UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

DARLAN VIVIAN
EDUARDO ADILIO PELINSON ALCHIERI

ANÁLISE DE MÉTRICAS DE QoS PARA SLAs EM REDES AD HOC COM ROTEAMENTO SEGURO

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de bacharel em Ciências da Computação.

FLORIANÓPOLIS
2004

Darlan Vivian
Eduardo Adilio Pelinson Alchieri

ANÁLISE DE MÉTRICAS DE QoS PARA SLAs EM REDES AD HOC COM ROTEAMENTO SEGURO

Este trabalho de Conclusão de Curso foi julgado adequado para obtenção de Graduação em Ciência da Computação e aprovado em sua forma final junto a Universidade Federal de Santa Catarina

_____
Prof° Dr. Carlos Becker Westphall
Orientador

Apresentada à Banca Examinadora, integrada pelos Professores:

_____
Prof° Dr. Carlos Becker Westphall

_____
Profª Drª. Carla Merckle Westphall

_____
Profº Dr. Mario Antônio Ribeiro Dantas

AGRADECIMENTOS

Nossos sinceros agradecimentos aos nossos pais e familiares que, de uma forma ou de outra, contribuíram para o surgimento deste trabalho. Ao nosso orientador, Professor Carlos Becker Westphall, pela oportunidade de sua orientação no desenvolvimento de nossas atividades. Ao Professor Mario Antônio Ribeiro Dantas e a Professora Carla Merckle Westphall por ter aceitado o convite de participação na banca examinadora e com isso também contribuido na realização deste trabalho.

Aos nossos colegas e amigos, por todos os momentos de alegria e diversão que tivemos durante os anos em que estivemos morando aqui em Florianópolis, amizades estas que durarão para sempre.

A todos o nosso MUITO OBRIGADO!

RESUMO

O surgimento de novas tecnologias tem possibilitado grandes avanços nas comunicações, e desta forma proporcionado aos usuários serviços de melhor qualidade. Nos últimos anos a comunicação sem fio tem ganhado destaque entre as tecnologias para transmissão de dados.

Um fator crítico em relação ao uso desta tecnologia diz respeito à questão da segurança no roteamento em redes *ad hoc*, pois a maioria dos protocolos confia nos nodos durante o roteamento dos pacotes. Para solucionar ou ao menos minimizar os problemas causados por esta falta de segurança, vários protocolos de roteamento seguro tem sido propostos.

Neste contexto, é fundamental a existência de acordos de níveis de serviço, que são contratos estabelecidos entre o consumidor e o fornecedor do serviço de rede. Nestes contratos os clientes definem níveis de serviço, parâmetros e métricas que irão refletir o bom funcionamento da rede.

Este trabalho tem como objetivo obter informações sobre uma rede sem fio *Ad Hoc*, verificando o impacto causado pela adição de segurança nos protocolos de roteamento e analisar estes dados a fim de moldar as métricas de qualidade de serviço que compõem os SLAs.

Através de simulações de uma rede *Ad Hoc*, utilizando-se o *Network Simulator 2*, coletaram-se os valores correspondentes a algumas métricas de QoS, para analisar como está se comportando o tráfego, assim como os possíveis problemas que podem estar acontecendo na rede. Foram simuladas algumas métricas básicas de QoS: latência, *jitter*, vazão e perda de pacotes. Para cada uma destas métricas, variam-se parâmetros como, tipo de fila (FIFO, RED, DRR), protocolo de roteamento (DSDV, DSR, SEAD, *Ariadne*), número de nodos (5, 10, 15, 20) e tamanho de pacotes (64 e 256 *bytes*).

Com base no resultado das simulações, pode-se traçar o perfil de um ambiente *Ad Hoc*, com ou sem segurança, de acordo com as características desejadas, propondo-se um SLA.

Palavras-chave: protocolo de roteamento, QoS, SLA, segurança, redes *Ad Hoc*

ABSTRACT

The appearance of new technologies has been making possible great progresses in the communications, and this way providing to the users services of better quality. In the last years the wireless communication has been winning prominence among the technologies for data transmission.

A critical factor in relation to the use of this technology is the safety in ad hoc nets routing, because most of the protocols trusts the nodes during the routing of the packages. To solve or at least to minimize the problems caused by lack of safety, several protocols of safe routing have been proposed.

In this context, it is fundamental the existence of service level agreements, which is a contract established between the consumer and the supplier of the net service. In these contracts the customers define service levels, parameters and metric that will reflect the good operation of the wireless net.

This work has as objective to obtain information on a wireless Ad Hoc network, verifying the impact caused by the addition of security in the routing protocols and to analyze these information in order to mold the metric of quality of service that compose SLAs, guaranteeing the good operation of the network.

Through simulations of a wireless Ad Hoc network, being used Network Simulator 2, the values of some metric of QoS were collected, to analyze how it is behaving the traffic, as well as the possible problems that can be happening in the network. Some basic metric ones of QoS were simulated: latency, jitter, throughput and loss of packages. For each one of these metric ones, parameters are varied as, queue type (FIFO, RED, DRR), routing protocol (DSDV, DSR, SEAD, Ariadne), number of nodes (5, 10, 15, 20) and size of packages (64 and 256 bytes).

With base in the result of the simulations, it is possible to identify the profile of an Ad Hoc environment, with or without safety, in agreement with the wanted characteristics, proposing a SLA.

***Keywords**: rounting protocols, QoS, SLA, security, Ad Hoc networks*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

ABR	*Associativity Based Routing*
AODV	*Ad Hoc On-Demand Distance Vector Routing*
AP	*Access Point*
ASP	*Application Service Provider*
BSA	*Basic Service Area*
BSS	*Basic Service Set*
AES	*Advanced Encryption Standard*
CBR	*Constant Bit Rate*
CBWFQ	*Class-Based Weighted Fair Queue*
CCA	*Clear Channel Assessment*
CFP	*Contention Free Period*
CGSR	*Clusterhead Gateway Switch Routing*
CQ	*Custom Queue*
CRC	*Cyclic Redundancy Code*
CSMA/CA	*Carrier Sense Multiple Access Collision Avoidance*
CSMA/CD	*Carrier Sense Multiple Access Collision Detection*
CTS	*Clear To Send*
CW	*Contention Window*
DBPSK	*Differential Binary Phase Shift Keying*
DCF	*Distributed Coordination Function*
DES	*Data Encryption Standard*
DFS	*Dynamic Frequency Selection*
DIFS	*Distributed Interframe Space*
DoS	*Denial of Service*
DQPSK	*Differential Quadrature Phase Shift Keying*
DRR	*Déficit Round Robin*
DS	*Distribution System*
DSDV	*Destination-Sequenced Distance-Vector Routing*
DSR	*Dynamic Source Routing*
DSSS	*Direct Sequence Spread Spectrum*
EAP	*Extensible Authentication Protocol*
ESN	*Enhanced Security Network*

ESS	*Extended Service Set*
FCC	*Federal Communications Comission*
FHSS	*Frequency Hopping Spread Spectrum*
FIFO	*First In First Out*
FQ	*Fair Queue*
GFSK	*Gaussian Frequency Shift Keying*
HAP	*Hardware Access Point*
IBSS	*Independent Basic Service Set*
IEEE	*Institution of Electrical and Electronic Engineers*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IR	*Infrared Rays*
ISM	*Industrial, Scientific and Medical*
ISO	*International Organization for Standardization*
ISO/IEC	*Open Systems Interconnections Basic Reference Model*
LAN	*Local Area Network*
LBT	*Listening Before Talking*
LLC	*Logic Link Control*
LMR	*Lightweight Móbile Routing*
MAC	*Medium Access Control*
MAN	*Metropolitan Area Network*
MANET	*Mobile Ad Hoc Network*
MH	*Mobile Host*
NAV	Network Allocation Vector
NS-2	*Network Simulation 2*
OFDM	*Orthogonal Frequency Division Multiplexing*
OSI	*Open Systems Interconnect*
OTcl	*Object Tool Command Language*
PCF	*Point Coordination Function*
PDA	*Personal digital assistants*
PHY	*Physical Layer*
PIFS	*Point Coordination Interframe Space*
PPM	*Pulse Position Modulation*
QoS	*Quality of Service*

RED	*Random Early Detection*
RIP	*Routing Information Protocol*
RF	*Radio Frenquency*
RTS	*Request To Send*
SAODV	*Secure Ad Hoc On-Demand Distance Vector*
SAP	*Software Access Point*
SEAD	*Secure Efficient Ad hoc Distance vector*
SFQ	*Stichastic Fair Queue*
SIFS	*Short Interframe Space*
SLA	*Service Level Agreement*
SLM	*Service Level Management*
SLS	*Service Level Specifications*
SRP	*Static Routing Protocol*
SSR	*Signal Stability routing*
SSID	*Service Set Identifier*
TBF	*Token Bucker Filter*
TKIP	*Temporal Key Integrity Protocol*
TORA	*Temporally Ordered Routing Algorithm*
TPC	*Transmit Power Control*
TTL	*Time to Live*
UDP	*User Datagram Protocol*
VPN	*Virtual Private Networking*
WASP	*Wireless Application Service Provider*
WEP	*Wired Equivalent Privacy*
WFQ	*Weighted Fair Queue*
WLAN	*Wireless Local Area Network*
WLL	*Wireless Local Loop*
WMAN	*Wireless Metropolitan Area Network*
WPAN	*Wireless Personal Area Network*
WRED	*Weighted Random Early Detection*
WRP	*Wireless Routing Protocol*
WWAN	*Wireless Wide Area Network*

SUMÁRIO

1. INTRODUÇÃO

O surgimento de novas tecnologias possibilitou grandes avanços nas comunicações, proporcionando aos usuários, serviços com a melhor qualidade possível. Nos últimos anos a comunicação sem fio tem ganhado destaque entre as tecnologias para transmissão de dados, estando cada vez mais presente em nosso cotidiano.

O impacto causado pelas comunicações sem fio no comércio e no modo de vivermos tem sido superado somente pelo impacto causado pela *Internet*. Celulares, *pagers*, e PDAs (*Personal Digital Assistants*) se tornaram tão comuns em nossas vidas que é fácil esquecer que há 10 anos atrás eles eram raros. Mas a tecnologia de comunicações sem fios tem muito a desenvolver-se e a próxima fase será o complemento ou substituição da infra-estrutura de rede tradicional (cabeadas) como também a habilitação de infra-estruturas de redes que previamente só poderiam ser imaginadas. Sistemas de controle do estoque de produtos podem ser acessados a partir de *cybers* cafés, de dentro de restaurantes ou em aeroportos públicos; este comércio sem fios esta começando a desafiar o sistema de trocas, que o nosso mundo moderno adota, através do acesso de fontes centrais de informação e de comunicação diretamente entre usuários e entre dispositivos[OUE02].

As redes de computadores sem fio têm cada vez se tornando mais populares devido a sua conveniência e baixo custo do hardware [TIO01]. Usuários podem transferir arquivos, acessar e utilizar os serviços da Internet sem a necessidade ligar cabos, conectados a *switches* ou *hubs*, às portas dos computadores. Eles possuem a liberdade de movimentar-se por todos os lugares dentro do alcance da rede.

Um dos fatores que limitava o uso da tecnologia sem fio era a baixa taxa de transferência oferecida por esta, não atendendo de forma satisfatória as necessidades de comunicação, principalmente no meio empresarial. Através de pesquisas, que deram origem a novos padrões como o IEEE 802.11g, obteve-se um incremento na largura de banda das redes e desta forma estimulou-se a sua adoção em diversas aplicações.

Outro fator crítico em relação ao uso desta tecnologia está ligado a segurança, principalmente no que diz respeito ao roteamento em redes *ad hoc*, pois a maioria dos protocolos é cooperativa por natureza [ROY99], e confiam em seus vizinhos no roteamento de pacotes. Esta confiança permite que nodos maliciosos paralisem uma rede sem fio inserindo atualizações de roteamento erradas, reproduzindo antigas

informações de roteamento, trocando atualizações de roteamento ou publicando informações incorretas de roteamento. Enquanto estes ataques são possíveis em redes cabeadas, a natureza dos ambientes sem fio aumenta seus efeitos, e faz com que a sua detecção seja difícil. Várias pesquisas tem sido realizadas com o intuito de sanar ou pelo menos minimizar os problemas causados pela falta de segurança e desta forma alguns protocolos de roteamento seguro foram propostos.

A presença de sistemas sem fio em empresas, universidades e outras instituições está cada vez maior, comprovando a tendência de que nos próximos anos as redes sem fio substituirão ou serão adicionadas aos sistemas com fio já existentes. As redes sem fio surgem como a quarta revolução na computação, antecedida pelos centros de processamento de dados da década de sessenta, o surgimento dos terminais nos anos setenta e as redes de computadores na década dos oitenta [LIM02].

Para os grandes negócios, as telecomunicações tornaram-se uma ferramenta vital, demandando um alto nível de confiabilidade no que se refere aos serviços que são oferecidos aos seus usuários. Desta forma, a integridade da rede passa a ter um papel fundamental, sendo que além de um bom desempenho, deve ser garantido que o serviço seja oferecido com qualidade e confiabilidade de acordo com as necessidades estipuladas pelo cliente.

Neste contexto, surgem os acordos de níveis de serviços - SLAs (*Service Level Agreements*), que são contratos estabelecidos entre o consumidor e o fornecedor do serviço. Nestes contratos os clientes definem níveis de serviço, parâmetros e métricas que refletem o bom funcionamento da rede.

Este trabalho tem como objetivo obter informações através de simulações sobre uma rede sem fio *Ad Hoc*, verificando o impacto causado pela adição de mecanismos de segurança nos protocolos de roteamento e analisar estes dados com o objetivo de moldar as métricas de qualidade de serviço que compõem os SLAs.

1.1 Justificativa

Com a utilização cada vez maior das redes sem fio e o aumento da variedade dos serviços oferecidos, as relações contratuais entre fornecedores de serviços e clientes estão se tornando complexas. Desta forma, a formalização de acordos, nos quais estão definidos os parâmetros de qualidade com seus respectivos níveis desejados é de grande importância tanto para o fornecedor quanto para o consumidor [PER03].

A mudança dinâmica de topologia, ambiente aberto e a falta de uma infra-estrutura centralizada de segurança, tornam uma rede sem fio *Ad Hoc* extremamente vulnerável a presença de nodos maliciosos e a certos tipos de ataques.

Neste contexto, a satisfação dos clientes atuais e a conquista de novos clientes são fundamentais para se obter sucesso neste mercado. Devido a demanda de serviços diferenciados e com níveis de qualidade e segurança personalizados de acordo com as necessidades de cada cliente, é indispensável que exista uma política eficiente de gerência e acordo de níveis de serviço.

1.2 Objetivos

As redes sem fio *Ad Hoc* têm sido usadas em cenários dinâmicos onde não existe estrutura com fio. No entanto, apesar da sua crescente aceitação e uso, não existe muitos trabalhos relacionados a esta topologia. Sendo assim, o objetivo geral deste trabalho é obter e analisar, através de simulações, dados sobre o comportamento de uma rede *Ad Hoc* mediante certas métricas de Qualidade de Serviço - QoS (*Quality of Service*), levando em consideração o impacto causado pela adição de segurança nas comunicações, através de protocolos de roteamento seguro.

Para isso, os seguintes objetivos específicos deverão ser atingidos:

- Estudar as principais características dos sistemas de comunicação sem fio;
- Avaliar os aspectos da topologia *Ad Hoc*;
- Realizar estudos e análises em gerência de redes sem fio e de gerência e acordos de níveis de serviço;
- Utilizar, estudar e compreender o funcionamento de uma ferramenta de simulação, mais especificamente a ferramenta NS-2 (*Network Simulator*);
- Identificar quais são os parâmetros que tem uma maior influência na qualidade do serviço;
- Verificar o impacto nas métricas de QoS causado pelo uso de protocolos de roteamento seguro;
- Verificar as alterações nas métricas de QoS ao alterar os valores de alguns parâmetros como protocolo de roteamento e tamanho dos pacotes;
- Contribuir para o avanço científico nesta área;

1.3 Trabalhos Relacionados

Alguns trabalhos científicos já foram desenvolvidos e publicados na área de redes sem fio do tipo *Ad Hoc* considerando questões como QoS e segurança. Nestes trabalhos são realizadas novas propostas, assim como a análise das já existentes, para garantir níveis aceitáveis de QoS e segurança em uma rede sem fio *Ad Hoc*. Dentre os trabalhos utilizados como inspiração e fonte de conhecimento para este trabalho, destacam-se os mais relevantes:

- *Performance Evaluation of Security-Aware Routing Protocols for Clustered Mobile Ad Hoc Networks* [YIO04]: este artigo explora a eficácia de protocolos de roteamento seguro para o suporte na transmissão de dados multimídia em tempo real em redes sem fio *Ad Hoc* composta por clusters. Dois protocolos de roteamento seguro são testados, SEAD e *Ariadne*, em um ambiente simulado e o desempenho deles é comparado considerando várias condições de tráfego e topologias de rede.
- *Análise de Desempenho em Redes Wireless Ad Hoc e Estabelecimento de um Acordo de Nível de Serviço Pró-Ativo* [PER03]: apresenta um estudo das quatro métricas básicas de QoS (latência, jitter, vazão e perda de pacotes) em redes sem fio do tipo *Ad Hoc*, através de simulações utilizando o NS-2. Neste trabalho é proposto, com base nos resultados das simulações, um SLA pró-ativo, onde as rotinas deverão prever uma possível quebra do acordo e com isso tomar alguma decisão.
- *ARIADNE: A secure On-Demand Routing Protocol for Ad Hoc Networks* [YIH02a]: neste artigo são apresentados os tipos de ataques possíveis em um sistema, e uma descrição dos ataques nos protocolos de roteamento em redes *Ad Hoc*. É apresentado também o projeto e a avaliação de desempenho de um novo protocolo *on-demand* para o roteamento seguro, chamado *Ariadne*.
- *SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks* [YIH02b]: neste artigo é apresentado o projeto e avaliação de um novo protocolo do tipo *table-driven* (pró-ativo) de roteamento seguro chamado SEAD, o qual é baseado no protocolo DSDV.
- *A performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols* [BRO98]: neste artigo são apresentados os resultados de uma

simulação em nível de pacotes comparando quatro protocolos de roteamento para redes *Ad Hoc*: DSDV, TORA, DSR, e AODV. Os autores estenderam o simulador NS-2 para representar o modelo MAC e o comportamento da camada física do padrão IEEE 802.11, incluindo um modelo realístico do canal de transmissão sem fios. São discutidos os resultados obtidos nas simulações realizadas em redes compostas por 50 nodos móveis.

1.4 Organização do Trabalho

Com o objetivo de facilitar o entendimento dos assuntos aqui tratados, este trabalho foi dividido em 6 capítulos. O primeiro procurou situar o leitor sobre o que será tratado no restante deste trabalho destacando a justificativa e os objetivos traçados.

No segundo capítulo é apresentada uma visão geral sobre o funcionamento das redes de computadores sem fio, o padrão IEEE 802.11 é descrito juntamente com a arquitetura e as topologias que este tipo de rede pode assumir. Ainda neste capítulo, alguns conceitos relacionados à segurança são abordados.

No terceiro capítulo são apresentadas algumas características referentes à topologia de redes sem fio *Ad Hoc*, discutindo suas vantagens e desvantagens. Este capítulo destaca aspectos relacionados às questões de segurança no roteamento de pacotes. Protocolos de roteamento e políticas de filas também são abordados.

No quarto capítulo os acordos de níveis de serviços são descritos, destacando os tipos de SLAs e as métricas usadas no seu estabelecimento. Vários outros conceitos relacionados com SLAs também são apresentados.

No quinto capítulo a ferramenta utilizada e o ambiente onde foram realizadas as simulações são descritos. Ainda neste capítulo, os resultados das simulações são apresentados e discutidos.

Finalmente, no sexto capítulo, são feitas as conclusões do trabalho e sugestões de trabalhos futuros.

2. COMUNICAÇÃO SEM FIO – WIRELESS

O objetivo deste capítulo é dar uma visão geral sobre o funcionamento das redes sem fio, descrever o padrão IEEE 802.11 que as define, demonstrar quais são os componentes que compõem a arquitetura e as topologias que a rede pode ter. Alguns aspectos referentes à segurança nestes tipos de redes também são abordados.

2.1 Introdução

Nos últimos anos as redes sem fio têm sido cada vez mais utilizadas na comunicação entre dispositivos dos mais variados tipos e tamanhos, tais como computadores portáteis, de mão, telefones e sensores, nos mais distintos ambientes, como edifícios, campus de universidades e até mesmo em florestas e campos de batalha.

Por permitirem a mobilidade, estas redes facilitam a utilização do poder computacional, tornando transparente a disseminação da informação e a cooperação dos dispositivos na realização das mais variadas tarefas [RUB04].

Diferentes padrões e tecnologias de rede sem fio surgiram nos últimos anos para acomodar esta vasta gama de aplicações e coberturas, como por exemplo: Redes Locais sem Fio ou WLAN (*Wireless Local Area Network*), Redes Metropolitanas sem Fio ou WMAN (*Wireless Metropolitan Area Network*), Redes de Longa Distância sem Fio ou WWAN (*Wireless Wide Area Network*), redes WLL (*Wireless Local Loop*) e o novo conceito de Redes Pessoais Sem Fio ou WPAN (*Wireless Personal Area Network*).

As redes locais sem fio (WLANs) constituem-se como uma alternativa às redes convencionais com fio, fornecendo as mesmas funcionalidades, mas de forma flexível, de fácil configuração e com boa conectividade em áreas prediais ou de campus. Dependendo da tecnologia utilizada, rádio freqüência ou infravermelho, e do receptor, as redes WLANs podem atingir distâncias de até 18 metros [SIL98].

Ligações sem fio de alcance limitado são particularmente atraentes porque além de ser um meio conveniente para transmissão de voz, vídeo, e dados, podem prover soluções de rede de baixo custo dentro de casa ou no ambiente da empresa. De fato, um exame das casas nos Estados Unidos que possuem uma rede interna revela um aumento fixo na migração de redes com fio, baseado em telefone e instalações elétricas para redes do tipo sem fio (Figura 1)[SAN02].

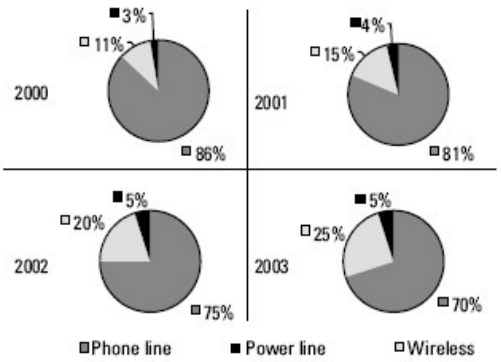


Figura 1 - Distribuição do uso do meio para estabelecimento de redes nas casas dos EUA

2.2 Rede Local sem Fio (WLAN)

Uma rede local sem fio pode ser definida como uma extensão de uma rede local cabeada tradicional, que converte os pacotes de dados em ondas de rádio ou raios infravermelhos e os envia para outros equipamentos da rede ou para um AP (*Acess Point*) que pode servir como uma conexão para uma LAN (*Local Area Network*).

A vantagem primária das redes locais sem fio (WLANs) é a habilidade que elas apresentam para se realizar a comunicação, através do ar, com uma rede cabeada ou com outros dispositivos que compõem a WLAN. Na figura 2 é possível visualizar o exemplo de uma WLAN típica [SIL98].

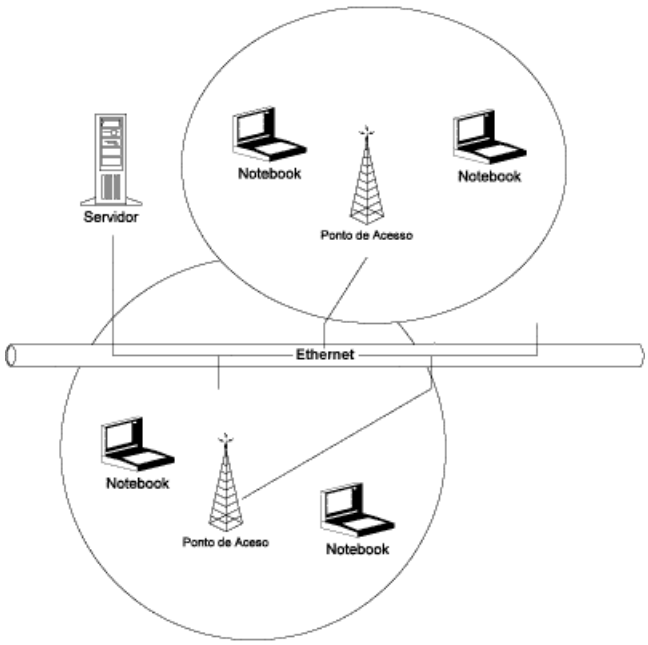


Figura 2 – WLAN típica

Neste ambiente típico, o transceptor (transmissor/receptor) ou ponto de acesso AP é conectado a uma rede local *Ethernet* convencional (com fio). Os pontos de acesso não apenas fornecem a comunicação com a rede convencional, como também intermediam o tráfego entre os dispositivos vizinhos, num esquema de micro células com *roaming* semelhante a um sistema de telefonia celular [SIL98].

As WLANs constituem-se como uma alternativa para as LANs, pois fornecem as mesmas funcionalidades, porém de forma flexível, são de fácil instalação e proporcionam uma boa conectividade. Geralmente são utilizadas em áreas prediais ou de campus em situações em que as áreas a serem cobertas pela rede são de difícil acesso, quando não é possível a instalação de cabos, como por exemplo, em construções antigas ou tombadas pelo patrimônio histórico ou ainda por ser economicamente inviável.

O grupo de trabalho IEEE 802.11, do IEEE (*Institute of Electrical and Electronics Engineers*), é responsável pela definição do padrão para as redes locais sem fio WLANs. O padrão proposto especifica três tipos (opções) de camadas físicas PHY (*Physical Layer*) e apenas uma subcamada MAC (*Medium Access Control*).

2.3 Padrão IEEE 802.11

IEEE é uma associação que desenvolve padrões para quase tudo que for eletro-eletrônico. Possui 36 sociedades técnicas que cobrem várias áreas de interesse, sendo que tópicos mais específicos são controlados por comitês especiais que se focalizam em uma tecnologia particular ou em algumas tecnologias para desenvolver padrões que serão usados para promover o avanço tecnológico.

O comitê de padrões IEEE 802 LAN/MAN desenvolve padrões de LAN (*Local Area Network*) e MAN (*Metropolitan Area Network)*. Os padrões mais amplamente usados são os da família *Ethernet* (802.3), *Token ring* (802.5), WLAN (802.11) e WPAN (802.15).

A conclusão do padrão IEEE 802.11 para WLANs em 1997 foi um primeiro passo importante no desenvolvimento evolutivo de tecnologias de gerência de redes sem fio. O padrão foi desenvolvido para maximizar a interoperabilidade entre diferentes tipos de WLANs como também para introduzir uma variedade de melhorias de desempenho e benefícios. Antes da adoção do padrão IEEE 802.11, a comercialização de equipamentos para redes sem fio era baseado em tecnologias proprietárias.

Como em todos os padrões 802.x, a especificação 802.11 cobre a operação do MAC e PHY, como ser visualizado na figura 3 [OUE02].

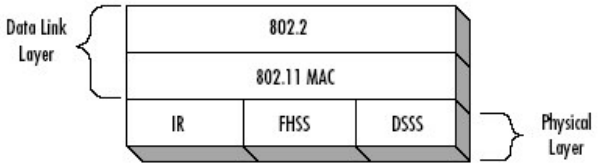


Figura 3 – Visualização das camadas MAC e PHY do padrão 802.11

Este padrão, assim como todos os protocolos da família 802.x, envolvem a camada física e de enlace do *Open Systems Interconnections Basic Reference Model* (ISO/IEC 7498-1:1994), conhecido como Modelo OSI e criado pela ISO (*International Organization for Standardization*) [ISO04]. Através da Figura 4 [WHA04] é possível visualizar uma comparação entre o modelo padrão de redes de computadores e o padrão IEEE 802.11.

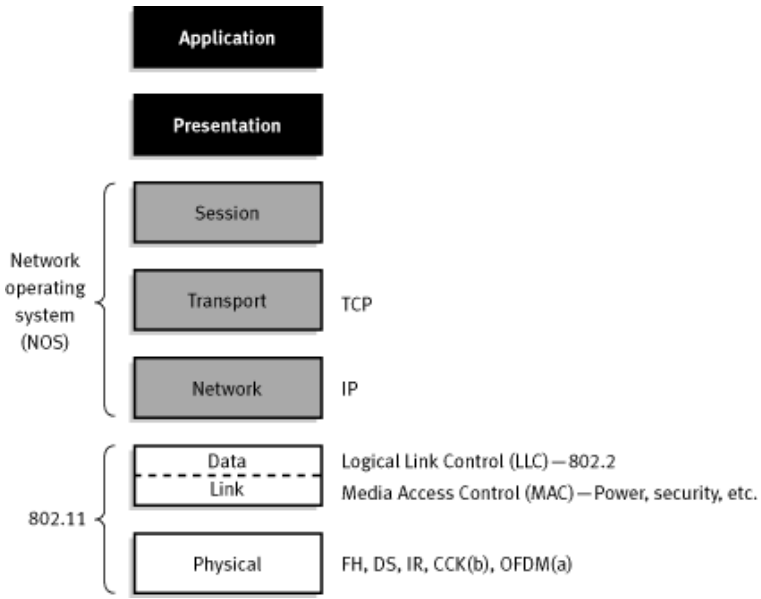


Figura 4 – Comparação entre o modelo OSI e o padrão IEEE 802.11

2.3.1 Camada Física 802.11

Devido ao fato de que muitos dispositivos utilizam as faixas ISM (*Industrial, Scientific and Medical*) em uma determinada área, é necessário que exista uma tecnologia para impedir que os vários sinais interfiram uns nos outros. Para contornar

este problema, foi desenvolvida uma tecnologia que permite que a largura da banda seja compartilhada conhecida como Espalhamento de Espectro (*Spread Spectrum*).

Esta tecnologia permite o espalhamento do sinal de rádio sobre um largo "espectro" de freqüências de rádio, minimizando o impacto de interferência. Na maioria dos casos, apenas pequenas partes da transmissão são corrompidas pela interferência, mas técnicas de codificação permitem que os dados sejam recuperados.

A técnica de *Spread Spectrum* consiste em codificar e modificar o sinal de informação executando o seu espalhamento no espectro de freqüências. O sinal espalhado ocupa uma banda maior que a informação original, porém possui baixa densidade de potência e, portanto, apresenta uma baixa relação sinal/ruído.

O *Spread Spectrum* utiliza as faixas de freqüências livres adotadas por vários países, inclusive o Brasil, denominadas internacionalmente como bandas ISM definidas nas faixas de 900 MHz, 2,4 GHz e 5,8 GHz (Figura 5).

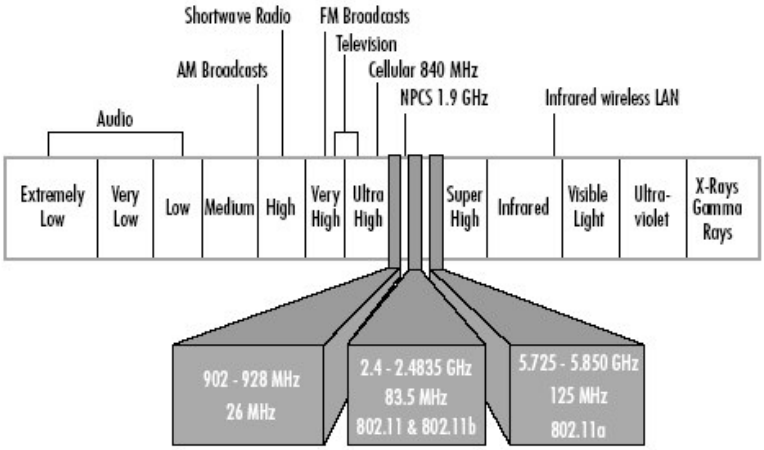


Figura 5 – Freqüências das bandas ISM

A camada física para 802.11 apresenta três opções: uma através de IR (*Infrared Rays*) e duas através de RF (*Radio Frenquency*). Devido a limitações de linha-de-visão, a transmissão infravermelha tem sido pouco estudada. A camada física de rádio freqüência é composta pelas técnicas de Espalhamento de Espectro por Salto em Freqüências – FHSS (*Frequency Hopping Spread Spectrum*) e Espalhamento de Espectro por Seqüência Direta – DSSS (*Direct Sequence Spread Spectrum*), A escolha de determinada técnica dependerá de vários fatores relacionados com a aplicação dos usuários e o ambiente onde a rede irá operar.

As redes *wireless* baseadas em radiofreqüência usam as faixas de freqüência ISM para a transmissão (Figura 5) [OUE02] . O padrão IEEE 802.11 define que as tecnologias de transmissão por espalhamento espectral operem na faixa de 2,400 a 2,4835 Ghz da banda ISM [IEE99a]. O responsável por esta regulamentação é o órgão FCC (*Federal Communications Comission*), que define que a banda ISM pode ser utilizada sem a necessidade de pedido de autorização para operação.

2.3.1.1 Espalhamento Espectral por Salto de Freqüência (FHSS)

O primeiro tipo de espalhamento de espectro desenvolvido é conhecido como FHSS (Figura 6) no qual ocorre a divisão da banda passante em subcanais de pequenas bandas (vários canais de freqüência). Os subcanais resultantes da divisão da banda passante são espaçados uniformemente sobre a faixa de freqüência de 83.5 Mhz, ocupando uma largura de banda de 1 Mhz.

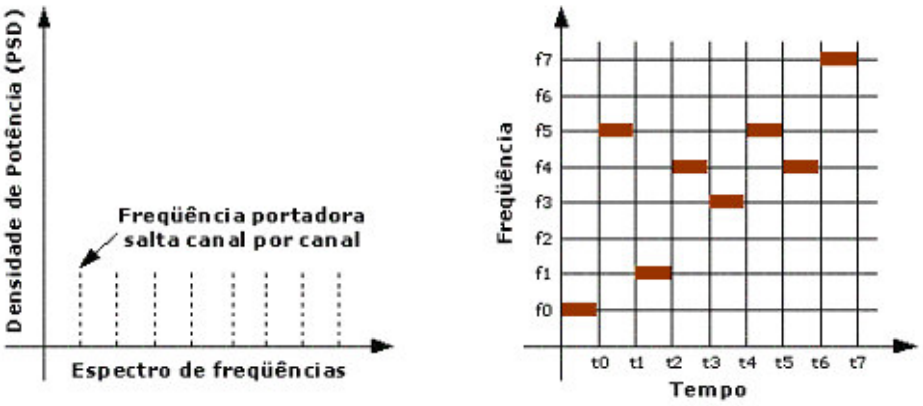


Figura 6 - Espalhamento Espectral por Salto de Freqüência

Através do salto de freqüência (processo de saltar rapidamente de uma freqüência para outra) a informação transmitida "salta" de um subcanal para outro numa seqüência pseudo-aleatória. Esta seqüência é determinada por um circuito gerador de códigos "pseudo-randômicos" que na verdade trabalha num padrão pré-estabelecido [RSS04].

Para poder sintonizar estes canais e receber os pacotes transmitidos, o receptor deverá estar sincronizado com o transmissor para saber previamente a seqüência de canais pelos quais o transmissor irá saltar.

Esta camada fornece operação de 1 Mbps, com 2 Mbps opcional. A versão de 1 Mbps utiliza 2 níveis da modulação GFSK (*Gaussian Frequency Shift Keying*), e a de 2 Mbps utiliza 4 níveis da mesma modulação [SIL98].

As vantagens desta técnica são [RSS04]:

1. Os canais que o sistema utiliza para operação não precisam ser seqüenciais.
2. A probabilidade de diferentes usuários utilizarem a mesma seqüência de canais é muito pequena.
3. A realização de sincronismo entre diferentes estações é facilitada em razão das diferentes seqüências de saltos.
4. Maior imunidade às interferências.
5. Equipamentos de menor custo.

As desvantagens desta técnica são [RSS04]:

1. Ocupação maior do espectro em razão da utilização de diversos canais ao longo da banda.
2. O circuito gerador de freqüências (sintetizador) possui grande complexidade.
3. O sincronismo entre a transmissão e a recepção é mais critico.
4. Baixa capacidade de transmissão, da ordem de 2 Mbit/s.

2.3.1.2 Espalhamento Espectral por Seqüência Direta (DSSS)

Nesta técnica de *Spread Spectrum* empregando a tecnologia de Seqüência Direta (Figura 7) diferentes transmissões são separadas por códigos e não por freqüência como ocorre no FHSS. Desta forma, o sinal de informação é multiplicado por um sinal codificador com característica pseudo-randômica conhecido como "*chip sequence*" ou pseudo-ruído ("*pseudo-noise*" ou *PN-code*). Este sinal codificador é um sinal binário gerado numa freqüência muito maior do que a taxa do sinal de informação. Ele é usado para modular a portadora de modo a expandir a largura da banda do sinal de rádio na freqüência transmitida [RSS04].

Em razão da utilização de uma grande largura de banda para transmissão, os sistemas em seqüência direta dispõem de poucos canais dentro da banda. Estes canais são totalmente separados de forma a não gerar interferência entre eles.

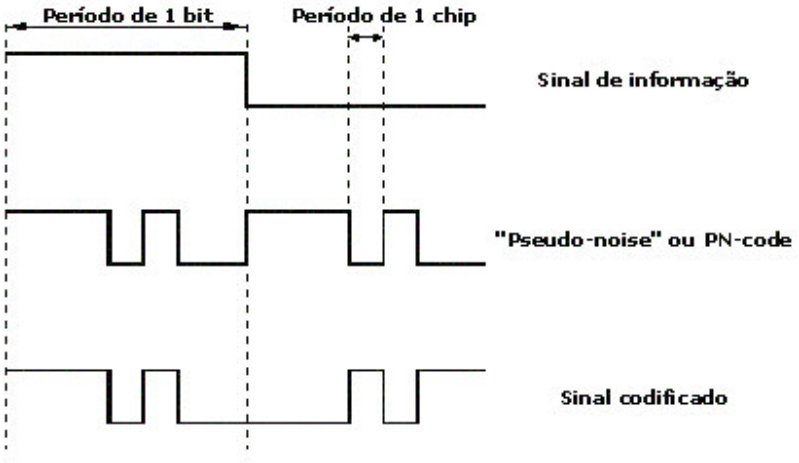


Figura 7 - Espalhamento Espectral por Seqüência Direta

Em um determinado momento, conforme uma seqüência, o transmissor envia os dados ciclicamente em diversos subcanais e para que o receptor possa receber os dados corretamente, deverá percorrer os subcanais na mesma ordem em que o transmissor os utiliza, ou seja, para recuperar os dados o receptor deverá saltar em sincronia com o transmissor. No entanto, a informação só será totalmente resgatada se a série de canais do transmissor for conhecida pelo receptor [SIL04b].

Esta camada provê operação em ambas as velocidades (1 e 2 Mbps). A versão de 1 Mbps utiliza a modulação DBPSK (*Differential Binary Phase Shift Keying*), enquanto que a de 2 Mbps usa modulação DQPSK (*Differential Quadrature Phase Shift Keying*) [SIL98].

As vantagens desta técnica são [RSS04]:

1. O circuito gerador de freqüência (sintetizador) é mais simples, pois não tem necessidade de trocar de freqüência constantemente.
2. O processo de espalhamento é simples, pois é realizado através da multiplicação do sinal de informação por um código.
3. Maior capacidade de transmissão, da ordem de 11 Mbit/s.

As desvantagens desta técnica são [RSS04]:

1. Maior dificuldade para manter o sincronismo entre o sinal *PN-code* gerado e o sinal recebido.
2. Maior dificuldade para solução dos problemas de interferências.
3. Equipamentos de maior custo.

2.3.1.3 Raios Infravermelhos (IR)

O espectro infravermelho já é usado há muito tempo na produção de controles remotos para alguns aparelhos como televisão e videocassete. Nos últimos anos, o uso de dispositivos para computadores residenciais, que utilizam raios infravermelhos, se tornou um fato comum. Dispositivos como teclados e mouses sem fio possibilitam a sensação de liberdade para trabalhar e jogar sem ficar amarrado ao computador. Muitos *laptops* têm sido fabricados com uma porta infravermelho que permite a troca de informação entre outro dispositivo portátil ou infravermelho através da transmissão infravermelha.

Semelhante a conexão infravermelha entre os *laptops*, os sinais infravermelhos são usados para transmitir dados em uma WLAN. Estas redes podem estar configuradas para funcionarem através de ligações ponto-a-ponto (usando conceito de linha de visão) ou através da difusão, onde os sinais são refletidos por algum tipo de superfície comum para todos o nodos [OUE02].

O sistema de transmissão IR é uma das três camadas físicas suportadas pelo padrão IEEE 802.11. Para a transmissão dos dados, esta camada utiliza raios com comprimento de onda muito alto (850 a 950 nm), um pouco abaixo do espectro da luz. Fornece operação a 1 Mbps, com 2 Mbps opcional. A versão de 1 Mbps usa modulação 16-PPM (*Pulse Position Modulation* com 16 posições), e a versão de 2 Mbps utiliza modulação 4-PPM.

A transmissão através de raios infravermelhos tem sido pouco utilizada na implementação de WLANs devido a alguns fatores [OUE02]:

1. Os raios infravermelhos podem ser facilmente obstruídos, pois não conseguem atravessar objetos sólidos.
2. Apesar de possibilitar taxas consideráveis de transmissão, a sua implementação apresenta um custo muito elevado.
3. Possui limitações de distância e cobertura, sendo que muitos dispositivos infravermelhos são necessários para cobrir a mesma área de cobertura suportada por um AP.

2.3.2 Camada MAC 802.11

O grupo de trabalho IEEE 802.11 foi formado com o objetivo de especificar um padrão internacional para as WLANs, que respeitasse alguns critérios: prover a interconexão com os sistemas pré-existentes e garantir tempos de respostas aceitáveis, sem comprometer a produtividade do usuário final. Para que estes objetivos sejam alcançados, a camada MAC (*Medium Access Control*) deve aparecer para a camada LLC (*Logic Link Control*) e superiores como qualquer outra rede 802.x, como por exemplo, uma rede *Ethernet* cabeada [SIL04b].

O mecanismo de acesso básico ao meio para 802.11 é o Acesso Múltiplo com Detecção de Portadora Evitando Colisões - CSMA/CA (*Carrier Sense Multiple Access Collision Avoidance*). O CSMA/CA é semelhante à Acesso Múltiplo com Detecção de Portadora com Detectação de Colisões - CSMA/CD (*Carrier Sense Multiple Access Collision Detection*) usado no padrão 802.3 (*Ethernet*), mas com algumas diferenças essenciais.

A camada MAC do 802.11 também define dois tipos de funções de acesso ao meio: a DCF (*Distributed Coordination Function*) e a PCF (*Point Coordination Function*) [RUB04]. A arquitetura da camada MAC pode ser visualizada na figura 8 [IEE99a].

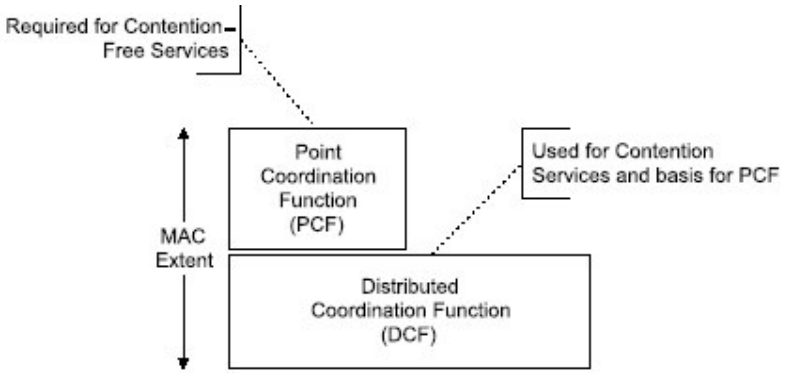


Figura 8 – Arquitetura da camada MAC

2.3.2.1 Mecanismo CSMA/CA

Diferente do padrão 802.3 (*Ethernet*) que envia um sinal até que uma colisão seja descoberta, o CSMA/CA toma um grande cuidado para não transmitir a menos que

tenha a atenção do receptor e nenhuma outra unidade esteja falando. Isto é conhecido como escutando antes de falar - LBT (*Listening Before Talking*).

Embora o método de acesso CSMA/CD seja muito utilizado nas redes IEEE 802.3, ele não é adequado às redes 802.11 pois nesse caso a detecção de colisões é muito difícil por assumir que todas as estações ouvem as outras, por requerer um rádio *full-duplex* de custo elevado e porque a taxa de erro de bit na camada MAC do 802.11 é de 10^{-5} [RUB04].

Antes de um pacote ser transmitido, o dispositivo sem fio escutará para ouvir se qualquer outro dispositivo está transmitindo. Se uma transmissão estiver acontecendo, o dispositivo esperará durante um determinado período de tempo, e então escuta novamente. Se ninguém mais estiver usando o meio, o dispositivo começará a transmitir. Caso contrário, esperará novamente durante um tempo randômico antes de escutar mais uma vez.

2.3.2.2 Função de Coordenação Distribuída (DCF)

O DCF, mecanismo básico de acesso ao meio no 802.11, é de modo simples, um CSMA/CA com reconhecimento positivo. Existem dois tipos de DCF no padrão: o baseado em CSMA/CA (obrigatório) e outro (opcional) que também utiliza pedidos e permissões para transmitir dados (*Request To Send* – RTS e *Clear To Send* - CTS) (Figura 10).

Uma estação que quer transmitir algum quadro, ouve o meio (detecta ou não a portadora). Caso o meio esteja livre após um determinado tempo chamado DIFS (*Distributed Interframe Space*), a estação transmite. Caso contrário, a transmissão é adiada e inicia-se um processo de *backoff*, no qual a estação escolhe um tempo aleatório uniformemente distribuído entre zero e o tamanho da CW (*Contention Window*), evitando assim colisões, e cria um temporizador de *backoff*. Esse método é conhecido como *backoff* exponencial binário. Este temporizador é decrementado periodicamente quando o meio está livre por mais de DIFS segundos, ou seja, não há nenhuma estação transmitindo. O período de decremento é dado pelo tempo de *slot* que corresponde ao atraso máximo de ida e volta dentro de um IBSS (*Independent Basic Service Set*). O temporizador é parado quando alguma transmissão é detectada no meio. Quando o temporizador expira, a estação envia o seu quadro [RUB04].

A estação receptora usa o método de verificação cíclica CRC (*Cyclic Redundancy Code*) para detectar erros e caso o pacote esteja correto, envia um pacote de reconhecimento (ACK). Esse ACK é enviado num tempo chamado espaço pequeno entre quadros SIFS (*Short Interframe Space*) após o recebimento do quadro anterior. Por definição, SIFS é menor que DIFS, ou seja, a estação receptora ouve o meio por SIFS para enviar o ACK (Figura 9). Caso a estação transmissora não receba o ACK, deduzirá que houve uma colisão, escalonará uma retransmissão e entrará no processo de *backoff*.

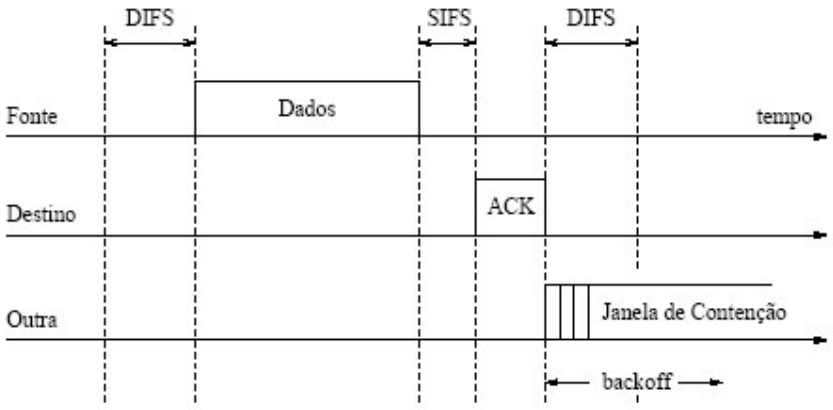


Figura 9 – Esquema básico de acesso no DCF

Para reduzir a probabilidade de colisões, a janela de contenção começa com um valor mínimo igual a 7 que é dado por *CWmin* e a cada transmissão não sucedida a janela de contenção aumenta para uma próxima potência de 2 menos 1, até que seja atingido um valor máximo predefinido de 255 chamado *CWmax*. Caso um número máximo de transmissões seja alcançado (sete), o pacote é descartado. Para evitar a captura do meio, caso a estação transmissora tenha mais algum pacote a transmitir, ela entra na fase de *backoff*.

O segundo tipo de DCF (Figura 10), que é opcional, inclui pacotes RTS e CTS para evitar problemas gerados por terminais "escondidos". Esse tipo de problema surge, por exemplo, quando uma estação *B* é capaz de receber quadros de dois diferentes transmissores, *A* e *C*, porém estes transmissores não podem se comunicar entre si. Nesse caso, o transmissor *A* pode achar que o meio está livre mesmo que *C* esteja transmitindo, o que resulta em colisão no receptor *B*.

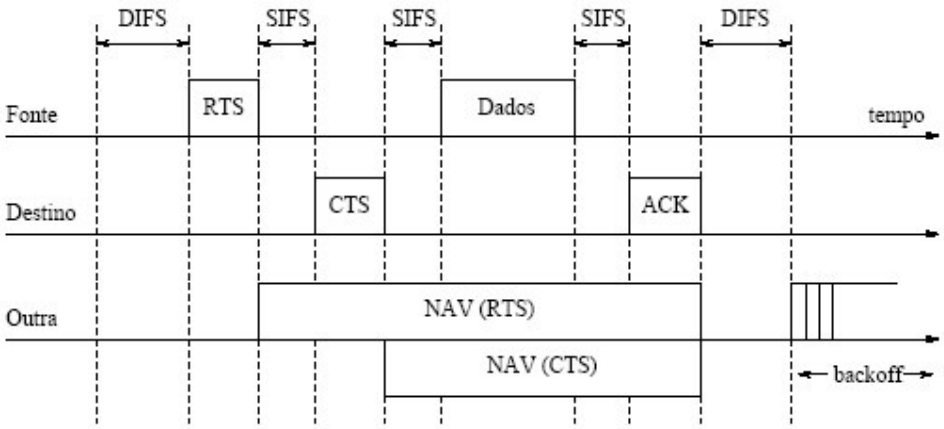


Figura 10 – DCF utilizando RTS e CTS

Nesse tipo de DCF, a detecção de portadora pode ser feita através do mecanismo físico (CCA) e virtualmente. O mecanismo de detecção virtual usa uma distribuição de informação de reserva do meio através da troca de quadros RTS e CTS antes do envio do dado [RUB04].

Os pacotes RTS e CTS contêm informações a respeito do nó de destino e de um tempo relativo ao envio do pacote de dados e de seu respectivo ACK. O uso de RTS e CTS é controlado por estação através de um limiar de RTS ($RTS_{threshold}$), através do qual uma estação pode não usar o RTS e o CTS, pode sempre utilizá-los ou ainda usá-los somente na transmissão de quadros maiores que o tamanho predeterminado.

Uma estação envia um RTS, após perceber que o meio está livre por pelo menos DIFS segundos, ao receptor antes da transmissão de um quadro para reservar o meio (Figura 10).

É evidente que a colisão de um quadro RTS de 20 *bytes* é menos severa e menos provável que uma colisão de quadros de dados que podem ter até 2346 *bytes*. O receptor responde com um CTS, após o meio estar livre por SIFS segundos, caso esteja pronto para receber. Todas as estações que ouvirem o RTS, o CTS, ou ambos, irão utilizar a informação da duração relativa ao pacote de dados para atualizar o vetor de alocação de rede NAV (*Network Allocation Vector*), que é utilizado para uma detecção virtual da portadora (Figura 10). Essa informação indica o período de tempo pelo qual uma transmissão não é iniciada pela estação, não importando se o CCA (*Clear Channel Assessment*) indique que o meio está livre. Desse modo, qualquer terminal escondido poderá adiar a sua transmissão para evitar colisões. Ao receber o CTS e esperar o meio estar livre por SIFS segundos, o transmissor inicia o envio do quadro, como no DCF

básico. Caso não receba o CTS, o transmissor entra na fase de *backoff* e retransmite o RTS [RUB04].

2.3.2.3 Função de Coordenação em um Ponto (PCF)

Outro tipo de acesso da camada MAC do 802.11 é o PCF. Apesar da implementação do DCF ser obrigatória, esse não é o caso do PCF. No modo PCF, um único ponto controla o acesso ao meio, através de consulta a cada estação, proporcionando a oportunidade de transmitir sem contenção [RUB04].

O coordenador de ponto, que opera e situa-se no AP, divide o tempo de acesso em períodos de super quadros. Cada super quadro compreende um período livre de contenção (modo PCF) e um período com contenção (modo DCF), como na Figura 11. Durante os períodos nos quais as estações estão no modo PCF, o coordenador de ponto consulta se cada estação tem algo a transmitir. As estações recebem dados quando são consultadas pelo coordenador de ponto.

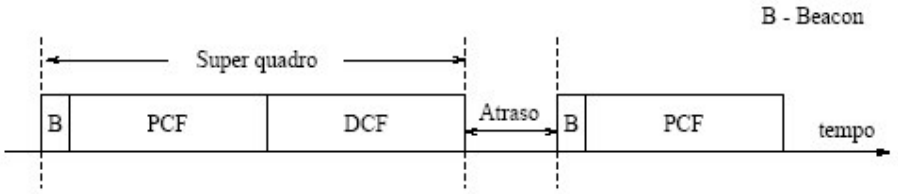


Figura 11 – Modos DCP e PCF operando juntamente

O coordenador de ponto inicia e controla o tempo livre de contenção. Ele escuta o meio por PIFS (*Point Coordination Interframe Space*) segundos e então começa um CFP (*Contention Free Period*) através da difusão de um sinal de *beacon* (Figura 11). Como, por definição, PIFS é menor que DIFS, nenhuma estação pode começar a enviar dados no modo DCF antes do coordenador de ponto.

Todas as estações adicionam a duração máxima do período de contenção ($CFP_{maxduration}$) aos seus respectivos NAVs. O período livre de contenção pode terminar a qualquer momento através do envio de um pacote *CFend* pelo coordenador de ponto. Isso ocorre freqüentemente quando a rede está com pouca carga. Além disso, o início de um período livre de contenção pode ser adiado por causa da transmissão de alguma estação no modo DCF (atraso na Figura 11).

Quando chega a vez de uma estação transmitir, o coordenador de ponto envia um pacote de dados, caso exista algum a ser enviado, dentro de um pacote de consulta (*piggyback*). O receptor envia de volta um ACK, também com dados se for o caso, depois de SIFS segundos. Após encerrar a transmissão a todas as estações contidas em uma lista de consultas, o coordenador de ponto reinicia o processo de consulta após PIFS segundos. Os usuários que estão sem transmitir por alguns ciclos são retirados da lista de consultas e são consultados de novo no início do próximo período livre de contenção.

2.3.2.4 Fragmentação

De modo a diminuir a probabilidade de erros devido ao enfraquecimento do sinal e ao ruído, quadros pequenos devem ser transmitidos. O MAC 802.11 provê suporte para a fragmentação de quadros em transmissões ponto-a-ponto e é responsável por remontar o quadro, o que torna o processo transparente para a camada superior. O padrão obriga que todos os receptores tenham suporte a fragmentação mas deixa como opcional a fragmentação nos transmissores. Um limiar de fragmentação ($Fragmentation_{threshold}$) é estabelecido, ou seja, um quadro é fragmentado se for maior que o limiar e o tamanho máximo de um fragmento também é dado por $Fragmentation_{threshold}$ [RUB04].

Caso não haja interrupção devido a limitação de ocupação do meio para uma camada física, os fragmentos de um quadro são enviados em rajada durante um período de contenção, utilizando uma única invocação do procedimento de acesso ao meio do DCF. Um fragmento é enviado SIFS segundos após o recebimento do ACK relativo ao fragmento anterior. A informação da duração no pacote de um fragmento indica o tempo necessário para a recepção do ACK do próximo fragmento, fazendo com que as outras estações que obtiverem essa informação não transmitam. O campo duração nos quadros de dados e no ACK especifica a duração total do próximo fragmento e do seu ACK. No caso do período livre de contenção, os fragmentos são enviados como quadros individuais [RUB04].

2.4 Variações do Padrão IEEE 802.11

Com os avanços das técnicas de processamento de sinais e a necessidade de melhorias no padrão 802.11, principalmente no que diz respeito à velocidade de transmissão, a IEEE aplicou-se no melhoramento deste padrão. Devido a isso, vários outros padrões com características diferentes surgiram, porém com a mesma arquitetura e tecnologia, estimulando novas pesquisas e a padronização de produtos suportados por estas redes.

2.4.1 802.11b

Em setembro de 1999, o IEEE ratificou uma revisão do padrão 802.11, chamado 802.11 Taxa Alta (HR/DSSS) ou 802.11b, que provê taxas mais altas de transmissão de dados, mantendo o protocolo 802.11. A arquitetura básica, características, e serviços do 802.11b são definidos pelo padrão 802.11 original sendo que a revisão da especificação afeta somente a camada física PHY, adicionando taxas mais altas de transferência de dados e conectividade mais robusta. [IEE99c]

A contribuição fundamental do 802.11b foi a adição para o padrão de WLANs da unificação da camada física que suporta duas novas velocidades, 5.5 Mbps e 11 Mbps. Para realizar isto, o DSSS teve que ser selecionado como técnica exclusiva da camada física para o padrão porque a freqüência não pode suportar velocidades mais altas sem violar a regulação atual do FCC. Desta forma, as redes baseadas no 802.11b vão operar a 1 e 2 Mbps usando DSSS, mas não irão operar a 1 e 2 Mbps usando FHSS.

802.11b WLANs apresentam troca dinâmica de taxa de transmissão, permitindo que a taxa de dados seja automaticamente ajustada para compensar interferências. Numa rede ideal, os usuários transmitem dados à taxa de 11 Mbps. Porém, quando os dispositivos são movidos além de uma certa distância ou se uma interferência significativa está presente, os dispositivos 802.11b transmitirão a velocidades mais baixas, como 5.5, 2, e 1 Mbps [OUE02].

2.4.2 802.11a

O padrão IEEE 802.11a é uma das extensões de camada físicas do padrão 802.11 original. Abandonando completamente o espectro de expansão, o 802.11a usa

uma técnica de codificação chamada de OFDM (*Orthogonal Frequency Division Multiplexing*). O equipamento 802.11 opera a 5 GHz e suporta taxas de transmissão de até 54 Mbps [IEE99b].

Quando o IEEE concluiu os padrões para redes de comunicação sem fio 802.11a e 802.11b em 1999, sua meta era criar uma tecnologia padrão. Este padrão deveria contornar múltiplos tipos de codificação física, freqüências e aplicações do mesmo modo que o padrão *Ethernet* 802.3 foi desenvolvido com sucesso para 10 Mbps, 100 Mbps e a 1 Gbps sobre fibra ótica e vários tipos de cabos de cobre [OUE02].

A desvantagem de usar a camada MAC do 802.11b é que o padrão 802.11a herda as mesmas ineficiências presentes na implementação do 802.11b. A camada MAC do 802.11b é apenas aproximadamente 70% eficiente. Atualmente o *throughput* máximo das implementações, baseadas no 802.11b, estão entre 5.5 e 6 Mbps. De forma idêntica, o padrão 802.11a que possui limite de 54 Mbps, apresenta *throughput* máximo entre 30 a 35 Mbps considerando o *overhead* adicional causado pela camada física. Mas ao contrário do padrão 802.11b, o 802.11a não têm que transmitir seus cabeçalhos a 1 Mbps. Assim, este padrão ganha aproximadamente 5% de eficiência se comparado ao 802.11b [OUE02].

2.4.3 802.11g

Neste padrão a camada física será uma extensão do IEEE 802.11b com uma taxa de transmissão de 54 Mbps usando a modulação OFDM. A especificação IEEE 802.11g é compatível com a especificação IEEE 802.11b. Este padrão permite a utilização mista da rede, ou seja, equipamentos que utilizam o padrão 802.11b operando a 11 Mbs podem compartilhar a mesma rede com os equipamentos que operam a 54 Mbs.

2.4.4 802.11d

O padrão IEEE 802.11d foi desenvolvido para áreas fora dos chamados cinco grandes domínios regulatórios (EUA, Canadá, Europa, Japão e Austrália). O 802.11d tem um frame estendido que inclui campos com informações dos países, parâmetros de freqüência e tabelas com parâmetros [FAG03].

2.4.5 802.11e

Inicialmente o objetivo deste padrão era prover segurança e qualidade de serviço para a camada MAC. Posteriormente, somente às questões relacionada com qualidade de serviço foram obordadas por este padrão, as quais são necessárias para suporte de voz, vídeo e dados.

2.4.6 802.11f

O IEEE 802.11f está definindo as recomendações práticas, mais que os padrões. Estas recomendações descrevem os serviços dos APs, as primitivas, o conjunto de funções e os protocolos que deverão ser compartilhados pelos múltiplos fornecedores para operarem em rede [FAG03].

2.4.7 802.11h

Na Europa, os radares e satélites usam a banda de 5 GHz, a mesma utilizada pelo padrão IEEE 802.11a. Isto significa que podem existir interferências com radares e satélites. O padrão 802.11h adiciona uma função de DFS (*Dynamic Frequency Selection*) e um TPC (*Transmit Power Control*) para o padrão 802.11a [FAG03].

2.4.8 802.11i

Este padrão tem o objetivo de melhorar as funções de segurança do protocolo da camada MAC, que agora é conhecido como ESN (*Enhanced Security Network*). Para isso os seguintes protocolos são avaliados: WEP (*Wired Equivalent Protocol*), TKIP (*Temporal Key Integrity Protocol*), AES (*Advanced Encryption Standard*), IEEE 802.1x para autenticação e criptografia.

O grupo de trabalho 802.11i está trabalhando na integração do AES dentro da camada MAC. O AES segue o padrão do DES (*Data Encryption Standard*). Como o DES o AES usa criptografia por blocos. Diferente do DES, o AES pode exceder as chaves de 1024 bits, reduzindo as possibilidades de ataques [FAG03].

2.5 Arquitetura 802.11

A topologia de uma rede sem fios é dinâmica, portanto, o endereço de destino nem sempre corresponde ao local do destino. Isto eleva a dificuldade no envio de *frames* pela rede para o destino planejado.

Para que uma WLAN ofereça suporte à mobilidade de estações de uma forma transparente para as camadas superiores, é necessário que exista uma interação entre alguns componentes da rede sem fio. Estes componentes compreendem o BSS (*Basic Service Set),* IBSS (*Independent Basic Service Set*)*,* as estações, o meio sem fio, o AP (*Access Poin*), o DS (*Distribution System*) e o ESS (*Extended Service Set*) [SIL04b].

As topologias BSS (Figura 12) consistem em pelo menos um AP conectado à infra-estrutura de uma rede cabeada e a um conjunto de estações sem fios que estão sob o controle direto de uma mesma função de coordenação. Esta função é quem determina quando cada MH (*Mobile Host*) pode enviar e receber dados utilizando o meio de transmissão sem fio. A área ocupada pelos membros de um BSS é denominada de BSA (*Basic Service Area*) [OUE02].

O AP age como o servidor lógico para uma única célula de LAN sem fio ou canal. A comunicação entre duas estações finais acontece a partir de uma estação para o AP e do AP para a outra estação.

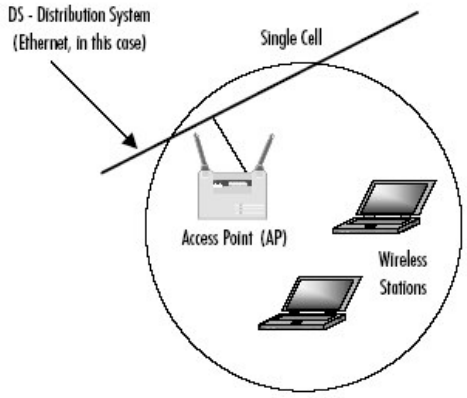


Figura 12 – Conjunto Básico de Serviços

As IBSS (Figura 13) também são conhecidas como configuração independente. Logicamente, uma configuração de IBSS é bastante semelhante a uma rede ponto-a-ponto de uma casa ou escritório na qual não existe a necessidade de que algum nodo funcione como servidor [OUE02].

As topologias IBSS incluem várias estações sem fio que se comunicam diretamente com outras estações, sem a intervenção de um AP ou qualquer conexão com uma rede cabeada. Isto é bastante útil para a configuração rápida e fácil de uma rede sem fio onde uma infra-estrutura sem fio não exista, como em quartos em hotéis e aeroportos.

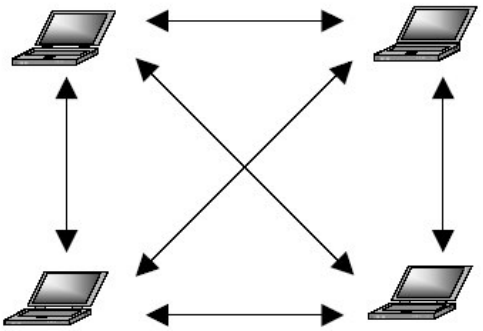


Figura 13 – Conjunto Básico de Serviços Independente

As topologias ESS (Figura 14) consistem em um conjunto de BSS (cada um com seu AP), geralmente conhecidas como células. Estas células normalmente estão conectadas através de um DS. O ESS é visto pela camada de protocolos superior (IP) como uma simples rede 802, do mesmo modo que uma rede *Ethernet* 802.3 usando *bridge* é vista como uma simples rede 802 pelas camadas de protocolo superiores [OUE02].

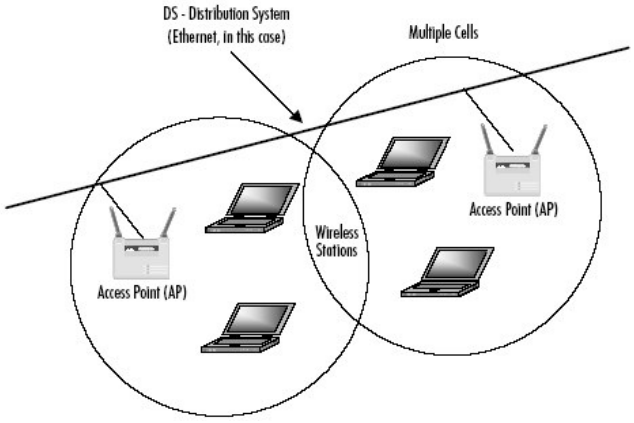


Figura 14 – Conjunto de Serviços Estendidos

Um sistema de distribuição de rede é absolutamente necessário se, por exemplo, bancos de dados, aplicações e serviços de impressão de uma rede são acessíveis somente através de uma rede cabeada.

2.6 Topologias das WLANs

Conforme especificações do padrão IEEE 802.11, as redes locais sem fio podem ser configuradas de dois modos distintos: com infra-estrutura e sem infra-estrutura. A seguir será descrito como as estações podem se organizar para se comunicar indiretamente ou diretamente entre si.

2.6.1 Redes com infra-estrutura

As redes com infra-estrutura são formadas pelas estações e pelos APs que são os responsáveis por boa parcela da funcionalidade da rede.

Nesta topologia, um nodo em uma BSS não é capaz de estabelecer uma comunicação diretamente com outro nodo sem que a informação passe por um AP centralizador (Figura 15). Quando uma estação dentro de um BSS deseja se comunicar com outra estação, a comunicação obrigatoriamente é interceptada pelo AP, para posteriormente ser enviada à estação de destino.

Aqui o funcionamento do ponto de acesso AP pode ser comparador a um *HUB*, provendo conectividade entre as estações sem fio. Pode-se conectar (*bridge*) uma *WLAN* com uma *LAN* cabeada, permitindo que as estações sem fio acessem os recursos de *LAN*, como servidores de arquivo ou conexão com a Internet. Existem dois tipos de APs (Figura 15): HAP (*Hardware Access Point*), que oferecem suporte para a maioria das características das redes sem fio e SAP (*Software Access Point*) que são executados em um computador equipado com uma interface de rede sem fio idêntica a usada em uma rede sem fio *Ad Hoc* ou ponto-a-ponto [WIR04].

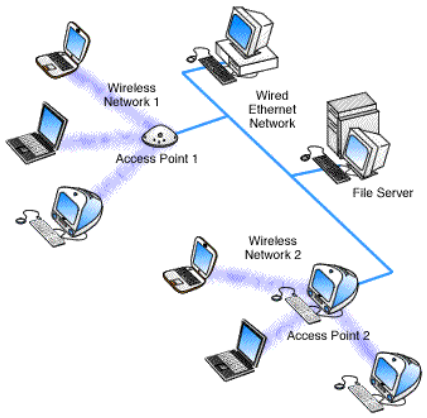


Figura 15 – Exemplo de WLAN com dois tipos de AP

Uma estação pode entrar em *roaming* (Figura 16), isto é, passar de uma BSS para outra através de um software e hardware que mantém uma conexão de rede fixa através do monitoramento da força do sinal proveniente do AP e procurando sempre um sinal de melhor qualidade. Normalmente isto é completamente transparente ao usuário o qual não percebe que um ponto de acesso diferente está sendo usado. Algumas configurações de ponto de acesso requerem autenticação de segurança ao trocar de ponto de acesso, normalmente na forma de requisição de uma senha [WIR04].

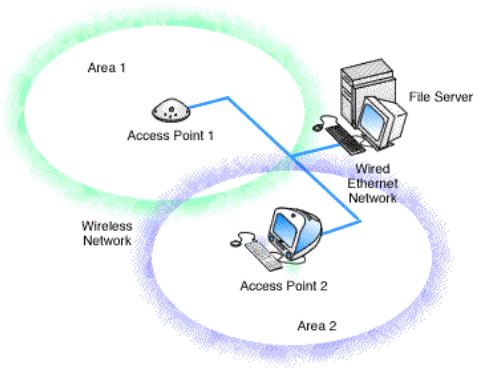


Figura 16 – Exemplo de *roaming*

2.6.2 Redes sem infra-estrutura (*Ad Hoc*)

As redes sem infra-estrutura, também são conhecidas como redes *Ad Hoc* ou MANET (*Mobile Ad Hoc Network*). Ao contrário das redes sem fio estruturadas, as redes *Ad Hoc* são formadas somente por estações móveis dentro de uma área restrita, que se comunicam sem a necessidade de um ponto de acesso AP, como pode ser visto na figura 17 [WIR04]. Nas redes MANET, o BSS é denominado de IBSS.

Figura 17 – Exemplo de rede *Ad Hoc*

Uma MANET é, portanto, um sistema autônomo de nodos móveis. O sistema pode operar em isolamento, ou pode apresentar *gateways* para se conectar com uma rede fixa. Os nodos são equipados com transmissores sem fio que utilizam antenas que podem ser unidirecionais [1] (broadcast), altamente direcional (ponto-a-ponto), possivelmente dirigível, ou a combinação destas possibilidades [IEE99b].

Em determinado ponto no tempo, dependendo das posições dos nodos e do padrão de cobertura de seus transmissores e receptores, do nível de potência da transmissão e do nível de interferência no subcanal, uma conectividade sem fio randômica (*Ad Hoc*) existe entre os nodos. Esta topologia pode mudar com o tempo devido à movimentação dos nodos ou ajuste dos parâmetros de transmissão e recepção. Caso os nodos não estejam na mesma área de cobertura do sinal, a rota entre eles pode ser formada por vários *hops* (saltos) através de um ou mais nodos na rede, caracterizando esta topologia como sendo *multihop* (múltiplos saltos).

As características e problemas das redes sem fio *Ad Hoc* serão apresentados detalhadamente no próximo capítulo.

2.7 Segurança em Wireless

Enquanto existem muitos trabalhos sobre a dinâmica dos protocolos de roteamento, a segurança em redes sem fio não foi muito explorada. Redes sem fios são geralmente mais propensas a ameaças de segurança física do que são as redes fixas (cabeadas). Existentes técnicas de segurança em nível de ligação (*link-level*) são freqüentemente utilizadas nas redes sem fio para reduzir estas ameaças. O que não pode ser evitado é a presença de nodos maliciosos que não respeitam os protocolos (eles poderão forjar pacotes, escutar o tráfego de outros, inventar erros, etc...).

Em um ambiente cabeado, a necessidade de acessar o meio físico (cabo) pode impedir alguém de conectar-se a rede, pois para isso será preciso entrar na construção para *plugar* um cabo na rede. Em uma WLAN, é impossível para o AP saber se a pessoa que opera o dispositivo sem fios está sentada dentro do edifício, passando, ou se esta em um estacionamento próximo. Por exemplo, na figura 18, o MH localizado fora da construção poderá acessar o AP da mesma forma como ele acessaria se estivesse dentro.

[1] Dispositivos que captam sinais de todas as direções.

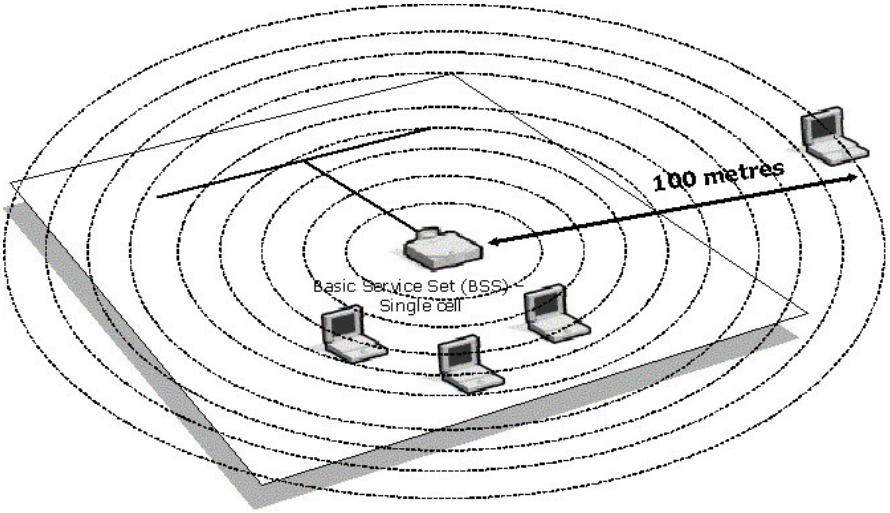


Figura 18 – Acesso de fora da construção

A segurança começa pela garantia das propriedades de confidencialidade, integridade e disponibilidade, e são comumente referenciadas por *The Big Three*, como podemos observar na figura 19. Alem destas propriedades outras também são necessárias como: autenticação, não repudiação, autorização.

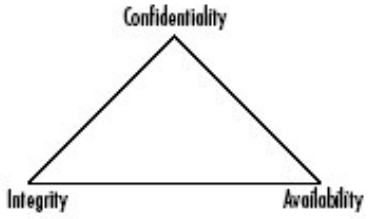


Figura 19 - *The Big Three*

O protocolo EAP (*Extensible Authentication Protocol*) foi desenvolvido para prover métodos de autorização e autenticação. O EAP pode ser configurado, desta forma pode suportar vários métodos para esquemas de autenticação, como cartões simbólicos (*token cards*), chave pública, certificados, e outros.

Quando um dispositivo sem fios está tentando conectar-se numa WLAN, envia um pedido de autenticação ao AP (Figura 20). Este pedido conterá o SSID (*Service Set Identifier*) da rede designada, ou um valor nulo caso esteja se conectando num sistema aberto. O AP aceitara ou negará a autenticação baseando-se no SSID, porém apenas isto não é seguro, pois o SSID de uma WLAN pode ser facilmente descoberto através de técnicas de *Sniffing*. Seguindo uma autenticação com sucesso, o dispositivo tentará associar-se ao AP. Neste momento pode-se fazer uma filtragem por meio do endereço

MAC e permitir o acesso apenas de determinados endereços, porém este endereço pode ser falsificado pelos nodos (um nodo intruso pode descobrir e utilizar o endereço de outro nodo).

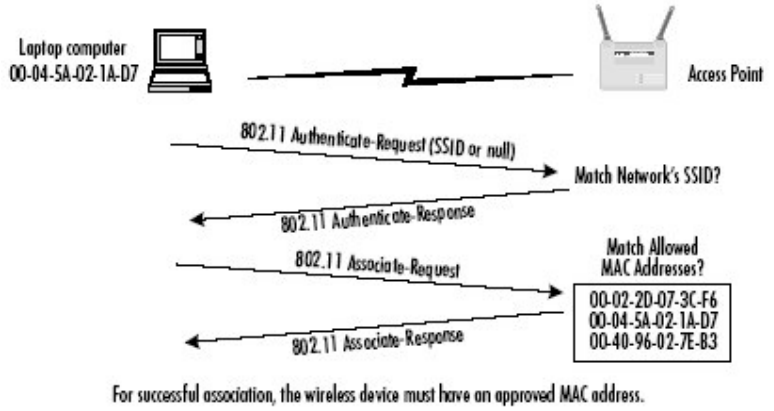


Figura 20 – Filtragem MAC

O uso de criptografia sempre foi o principal fator na segurança de informações e continuará a desempenhar um papel fundamental na segurança em redes sem fio, especialmente com a adoção de novos e melhores algoritmos de criptografia e sistemas de gerenciamento de chaves [OUE02]. Para proteger-se contra potenciais problemas de segurança usando criptografia, o padrão 802.11x tem uma função chamada WEP (*Wired Equivalent Privacy*), que é uma forma de criptografia que provê privacidade comparável com uma rede cabeada (Figura 21). Se informações devem ser enviadas através de uma rede sem fios de forma segura, o WEP deve ser usado, assegurando que os dados estarão protegidos com o mesmo grau de uma rede cabeada. Criptografia oferece a vantagem óbvia que o material protegido não pode ser usado sem as chaves necessárias para descriptografá-lo, no entanto possui algumas desvantagens como o consumo de maior tempo no processamento da informação. WEP oferece um nível razoável de segurança, desde que todos os fatores são usados corretamente, como cuidado no gerenciamento das chaves, evitando opções *default*.

Nunca foi pretendido que WEP desse segurança total, mas sim privacidade. Em casos que requerem altos graus de segurança outros mecanismos devem ser usados como autenticação, controle de acesso, senha e VPN (*Virtual Private Networking*).

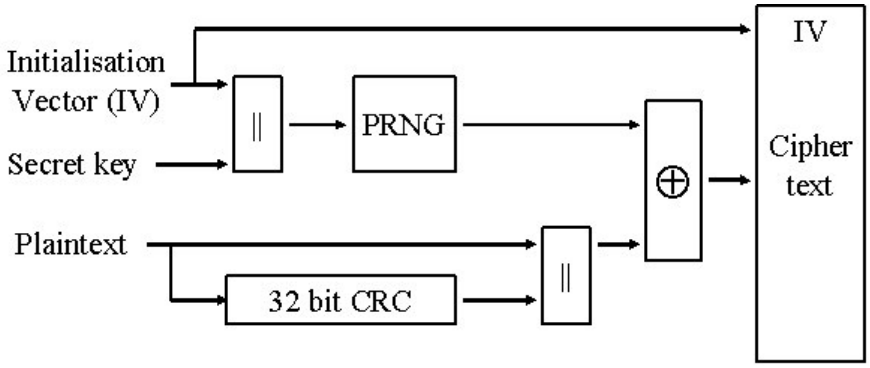


Figura 21 – WEP encryption

Também deve ser observado que técnicas tradicionais de VPN poderão ser implantadas sobre redes sem fios, da mesma forma como são usadas em redes cabeadas. Uma vasta descrição sobre segurança e mecanismos de segurança em redes sem fio pode ser encontrado em [OUE02].

2.8 Conclusões

As redes sem fios estão cada vez ocupando mais o espaço das redes cabeadas. Devido a deficiências em algumas propriedades não era muito aceita, no entanto a maioria delas já foi sanada ou pelo menos minimizada.

O continuo desenvolvimento da camada MAC e da camada física indica que no futuro a tecnologia *wireless* será confiável e robusta para aplicações críticas.

3. REDES SEM FIO *AD HOC*

O objetivo deste capítulo é apresentar algumas características da topologia de rede sem fio *Ad Hoc*, a qual é o principal foco de estudo deste trabalho. Alguns problemas, vantagens e desvantagens desta topologia são discutidos. Problemas de roteamento que envolvem questões de segurança, juntamente com alguns protocolos e tipos de filas também são abordados, destacando os que foram escolhidos para realizar as simulações presentes neste trabalho.

3.1 Introdução

Em uma rede sem fio *Ad Hoc*, os computadores (nodos) que formam esta rede cooperam entre si na transmissão de pacotes devido à limitação de transmissão individual de cada nodo. A rota de rede de um nodo remetente para um nodo destino pode requerer vários nodos intermediários criando uma rota *multihop* do remetente até o destinatário.

Redes *Ad Hoc* não requerem nenhuma administração centralizada ou uma infra-estrutura de rede fixa como estações base ou pontos de acesso e pode ser rapidamente configurada de acordo com as necessidades. Desta forma, este tipo de rede de computadores pode ser usada em cenários onde não exista infra-estrutura, ou onde a infra-estrutura existente não satisfaça as exigências da aplicação por razões como segurança, custo, ou qualidade. Exemplos de aplicações para redes *Ad Hoc* variam desde operações militares até a interação entre participantes presentes em uma reunião ou estudantes durante uma conferência.

3.2 Características

Devido as suas propriedades, as redes do tipo MANET possuem algumas características tais como [COR99]:

1. **Topologia dinâmica**: os nodos são livres para se mover arbitrariamente; assim, a topologia da rede (que é tipicamente *multihop*) pode mudar fortuitamente e rapidamente em intervalos de tempos imprevisíveis, podendo apresentar *links* tanto bidirecionais quanto unidirecionais.

2. **Limitação da Largura de Banda**: *links* sem fio continuarão tendo uma capacidade significativamente mais baixa se comparados aos *links* cabeados. Além disso, o *throughput* alcançado em uma comunicação sem fio é consideravelmente prejudicado devido aos efeitos causados por acessos múltiplos, desvanecimento do sinal, barulho e condições de interferência.
3. **Limitação de energia**: alguns ou todos os nodos em uma MANET precisam confiar em baterias ou outros meios exaustivos de energia. Para estes nodos, um dos critérios mais importante de projeto de sistema é a otimização do consumo de energia.
4. **Limitações na segurança**: redes sem fio móveis são geralmente mais propensas a ameaças de segurança física do que as redes com fio. Assim, o aumento da possibilidade de ocorrência de ataques de *eavesdropping* (escutar clandestinamente), *spoofing* e DoS (*Denial of Service*) devem ser considerados cuidadosamente.

3.3 Problemas abertos em redes sem fio *Ad Hoc*

As pesquisas nesta área estão longe de serem esgotadas, ainda tem muito pra ser descoberto com relação a estas redes. Muitos esforços têm sido despendidos no planejamento de protocolos de roteamento para prover uma efetiva e eficiente comunicação entre os nodos que fazem parte da rede. Entretanto existe outros tópicos que merecem investigação[AGR04]:

- Escalabilidade: até quanto uma rede *Ad Hoc* pode crescer sem que a comunicação e os serviços providos pela mesma se degradem?
- Endereçamento: o modo de endereçamento usado em redes cabeadas, e também em IP móvel, pode não ser adequado em MANET. Um novo esquema de endereçamento pode ser requerido para MANETs.
- Interoperabilidade com a *Internet*: como pode uma rede *Ad Hoc* acessar de forma eficiente a *Internet* para obter serviços avançados?
- Melhoramento da interação entre as camadas: uma melhor interação entre as camadas nos trará uma melhor performance.
- QoS: é possível rodar aplicações com requerimentos estritos de largura de banda e *delay* em uma MANET ?

- Segurança: como uma rede pode proteger-se de nodos maliciosos ou comprometidos?
- Controle do consumo de energia: como podemos maximizar a vida das baterias?

Muitos destes problemas já estão sendo pesquisados, no entanto ainda existe muito a ser feito.

3.4 Vantagens e Desvantagens

De acordo com [GTA04], várias vantagens e desvantagens podem ser citadas ao se comparar às redes *Ad Hoc* com as redes infra-estruturadas e com as redes fixas. Como vantagens pode-se citar:

a) rápida instalação: uma vez que as redes *Ad Hoc* podem ser estabelecidas dinamicamente em locais onde não haja previamente uma infra-estrutura de rede instalada;

b) tolerância à falhas: a permanente adaptação e reconfiguração das rotas em redes *Ad Hoc* permitem que perdas de conectividade entre os nós possam ser facilmente resolvidas desde que uma nova rota possa ser estabelecida;

c) conectividade: dois nós móveis podem se comunicar diretamente desde de que cada nó esteja dentro da área de alcance do outro. Em redes infra-estruturadas ou em redes fixas, mesmo que dois nós estejam próximos, é necessário que a comunicação passe pela estação de suporte à mobilidade (no caso de redes infra-estruturadas) ou, no caso de redes fixas, haver uma ligação por meio de cabo entre os dois nós;

d) mobilidade: esta é uma vantagem primordial com relação às redes fixas.

No entanto esta topologia possui algumas desvantagens, como:

a) roteamento: a mobilidade dos nós e uma topologia de rede dinâmica contribuem diretamente para tornar a construção de algoritmos de roteamento um dos principais desafios em redes *Ad Hoc*;

b) localização: outra questão importante em redes *Ad Hoc* é a localização de um nó, pois além do endereço da máquina não ter relação com a posição atual do nó, também não existem informações geográficas que auxiliem na determinação do posicionamento do nó;

c) taxa de erros: a taxa de erros associada a enlaces sem-fio é mais elevada;

d) banda passante: enquanto em meios cabeados a banda passante já chega em 1 Gbps, os enlaces sem-fio suportam tipicamente taxas de até 2 Mbps.

3.5 Protocolos de roteamento

Com os recentes avanços de desempenho dos computadores e das tecnologias de comunicação sem fio, é esperado ver o uso crescente e difundido de aplicações avançadas em redes móveis, muitas das quais envolverá o uso do IP (*Internet Protocol*). O objetivo das redes *Ad Hoc* é suportar operações robustas e eficientes pela adição de funcionalidades de roteamento entre os nodos móveis que formam a rede. Tais redes são previstas para serem dinâmicas, aleatórias e possuírem uma alta freqüência de variações.

A meta das redes móveis *Ad Hoc* é estender mobilidade no mundo dos autônomos, dos móveis, dos pertencentes ao domínio *wireless*, onde um conjunto de nodos, que pode ser tanto roteadores como *hosts*, formam a infra-estrutura de roteamento na rede [COR99].

Existe um grupo de trabalho do IETF (*Internet Engineering Task Force*) chamado *Mobile Ad Hoc Network Working Group* que trabalha na discussão e definição de protocolos de roteamento para redes móveis *Ad Hoc*.

A tarefa de roteamento em uma MANET é mais difícil do que em redes cabeadas, pois este depende de muitos fatores incluindo topologia, seleção de roteadores, iniciação da requisição, e características subjacentes específicas que podem servir de heurísticas para encontrar rapidamente e eficientemente o caminho pelo qual o pacote deve ser enviado.

Um dos principais desafios no projeto de um protocolo para redes *Ad Hoc* esta no fato de que um nodo precisa conhecer pelo menos a informação de localização de seus vizinhos para determinar uma rota para os pacotes, mas por outro lado, a topologia de rede pode mudar freqüentemente [JUB87]. Além disso, como o número de nodos da rede pode ser grande, encontrar a rota para os destinatários requer uma grande e freqüente troca de informações de controle entre os nodos. Assim, a quantidade de tráfego de atualizações pode ser bastante alta, e é ainda mais alta quando nodos com alta mobilidade estiverem presentes [AGR04]. Nodos de alta mobilidade podem causar um impacto tão grande no consumo de banda para manutenção das informações de roteamento que pode não sobrar largura da banda para a transmissão de pacotes de dados [COR96].

Existem vários protocolos de roteamento e eles podem ser divididos em dois grupos:

1. *Table-driven*: também chamados de pró-ativos, este grupo mantém as rotas de todos os possíveis destinatários em uma tabela de forma que quando um pacote precisar ser remetido, a rota já é conhecida e pode ser usada imediatamente. Fazem parte deste grupo os protocolos DSDV (*Destination-Sequenced Distance-Vector Routing*), WRP (*Wireless Routing Protocol*), CGSR (*Clusterhead Gateway Switch Routing*) e outros.

2. *On-demand*: também chamados de reativos, aqui as rotas dos destinatários apenas são descobertas sob demanda, isto é, um nodo não precisa conhecer a rota de um destinatário até que ele necessite enviar pacotes de dados para este destino. São exemplos de protocolos de roteamento pertencentes a este grupo o AODV (*AD HOC On-Demand Distance Vector Routing*), o DSR (*Dynamic Source Routing*), o LMR (*Lightweight Móbile Routing*), o TORA (*Temporally Ordered Routing Algorithm*), o ABR (*Associativity Based Routing*), o SSR (*Signal Stability routing*) e outros.

Cada protocolo de roteamento *Ad Hoc* possui vantagens e desvantagens, de acordo com determinadas situações, portanto não existe um que é melhor do que os outros [PER03]. No entanto o *Mobile Ad Hoc Network Working Group* especificou uma série de propriedades que um protocolo deve possuir [COR99], tais como:

- Operação distribuída: para evitar a centralização que leva à vulnerabilidade. Esta propriedade é essencial para o roteamento em uma rede *Ad Hoc*.
- Livre de *loops*: para que os pacotes não fiquem trafegando durante um período de tempo relativamente grande na rede, pode ser usada como solução uma variável do tipo TTL (*time to live*), mas uma abordagem melhor estruturada é mais indicada, por exemplo a utilização de número de seqüência (MANET).
- Operação baseada na demanda: Em vez de assumir uma distribuição uniforme de tráfego dentro da rede (e manter o roteamento entre todos os nodos a toda hora), deixa o protocolo de roteamento adaptar o padrão de tráfego baseado na demanda ou na necessidade básica. Se isto é feito inteligentemente, podem-se utilizar mais eficientemente os recursos de energia e a largura da banda da rede, às custas de aumentar o tempo de descobrir a rota.

- Operação pró-ativa: em certos contextos, a latência adicional causada pela operação baseada na demanda pode ser inaceitável. A operação pró-ativa é desejável nos casos onde os recursos de energia e a largura da banda permitirem.
- Segurança: sem alguma forma de segurança no nível de rede ou camada de ligação, um protocolo de roteamento MANET fica vulnerável a muitos tipos de ataques.
- Período de "dormência": como resultado da conservação de energia, ou de outra necessidade, nodos de um MANET podem parar de transmitir e\ou receber por períodos arbitrários de tempo. Um protocolo deve ser capaz de acomodar tais "períodos de sono" sem trazer conseqüências adversas.
- Suporte a enlaces (*links*) unidirecionais: ligações bidirecionais são tipicamente assumidas no projeto de protocolos de roteamento, no entanto muitos deles são incapazes de funcionar corretamente sobre ligações unidirecionais.

Segundo este mesmo grupo, algumas métricas podem ser levadas em consideração para determinar a performance de um protocolo, são elas:

- *throughput* e atraso fim-a-fim dos pacotes de dados;
- tempo para aquisição de uma rota, principalmente para algoritmos que trabalham sob demanda de rotas;
- porcentagem de pacotes entregues;
- eficiência do protocolo, que é o número de pacotes de controle necessários para que o protocolo funcione corretamente;
- capacidade de manipular e escolher a melhor rota baseada em parâmetros de QoS.

3.5.1 *Destination Sequenced Distance Vector Routing Protocol* (DSDV)

Um protocolo de roteamento por vetor de distância encontra os caminhos mais curtos entre os nodos de uma rede através da implementação distribuída do algoritmo clássico de *Bellman-Ford*. Os protocolos de vetor de distância são fáceis de implementar e são eficientes na utilização da memória e da capacidade de processamento da CPU de cada nodo [YIH02a]. O RIP (*Routing Information ProtocoI*) é um exemplo popular de protocolo de vetor de distância, o qual é extensamente usado

em redes IP de tamanho moderado. O roteamento por vetor de distância pode ser usado para o roteamento em uma rede *Ad Hoc* na qual cada nodo age como um roteador.

O protocolo DSDV é uma adaptação do RIP para o roteamento em redes *Ad Hoc*. O DSDV possui o atributo *sequence number*, para cada entrada na tabela de roteamento do protocolo RIP convencional. Através da utilização deste novo atributo, os nodos móveis podem identificar que a informação de uma rota está desatualizada evitando a formação de *loops* no roteamento.

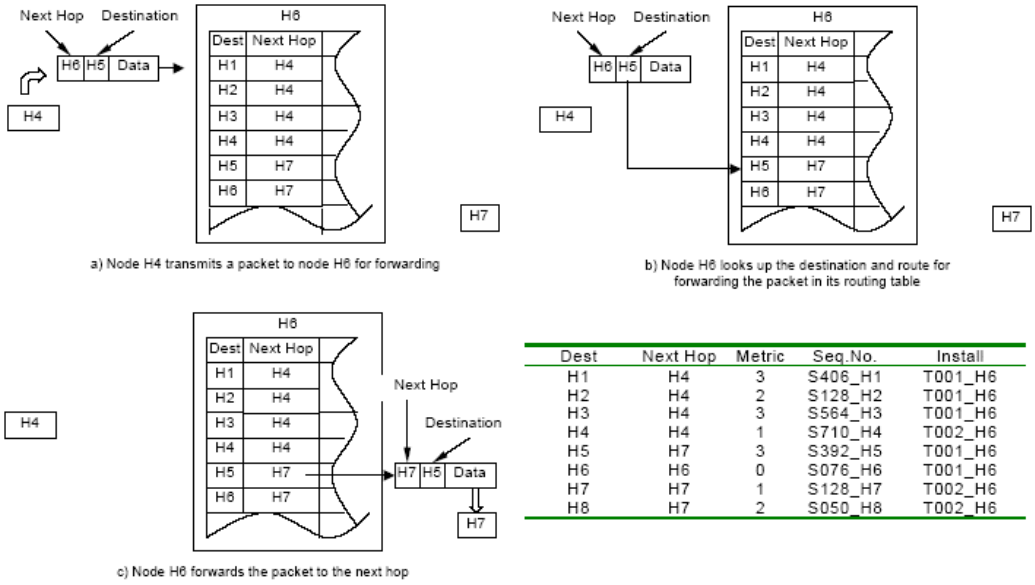
Cada nodo em uma rede *Ad Hoc* mantém uma tabela que lista todos os possíveis destinos dentro da rede. Cada entrada nesta tabela contém o endereço (identidade) de um possível destino, a menor distância conhecida (normalmente em número de *hops*) para aquele destino e o endereço do nodo vizinho que será o primeiro pulo neste caminho mais curto para aquele destino. À distância para o destino é conhecida como *metric* nas entradas da tabela. Quando um nodo estiver fazendo o roteamento de um pacote para algum destino, o nodo transmite o pacote para o roteador (nodo) vizinho indicado, e cada roteador usa sua própria tabela para realizar o próximo *hop* do pacote em busca do seu destino [PER94].

Periodicamente ou no instante em que a topologia da rede sofrer alguma mudança, cada nodo móvel transmite informações de roteamento fazendo *broadcasting* ou *multicasting* de um pacote de atualização da tabela de roteamento. Este pacote de atualização inicia com o atributo *metric* valendo um. Isto indica que cada vizinho receptor é um *hop* longe do outro nodo. Depois de receber o pacote de atualização, os nodos vizinhos atualizam as suas tabelas de roteamento incrementando de uma unidade o atributo *metric* e retransmitem este pacote para aos seus nodos vizinhos correspondentes. O processo será repetido até todos os nodos dentro da rede *Ad Hoc* receberem uma cópia do pacote de atualização [PER94].

Se os pacotes de atualização possuírem o mesmo *sequence number* para o mesmo nodo, será usado aquele com o menor valor para *metric* e a outra rota será descartada ou armazenada como uma rota menos preferível.

A figura 22 mostra um exemplo de uma rede *Ad Hoc* antes e depois do movimento dos nodos. O nodo H4 quer enviar um pacote ao nodo H5. O nodo H4 confere sua tabela de roteamento e identifica que o próximo *hop* é o nodo H6. Então H4 envia o pacote para H6 como mostrado na figura 22a. O nodo H6 procura o próximo *hop* para alcançar o nodo de destino H5 em sua tabela de roteamento quando recebe o pacote (figura 22b). O nodo H6 então retransmite o pacote para o próximo *hop* H7 como

demonstrado na tabela de roteamento da figura 22c. O procedimento de roteamento é repetido ao longo do caminho até que o pacote chegue ao nodo destino H5. É possível visualizar também na figura 22 a tabela de roteamento do nodo H6 antes do movimento dos nodos, sendo que o campo *Install* da tabela ajuda a determinar quando uma rota antiga deverá ser apagada [HEG03].

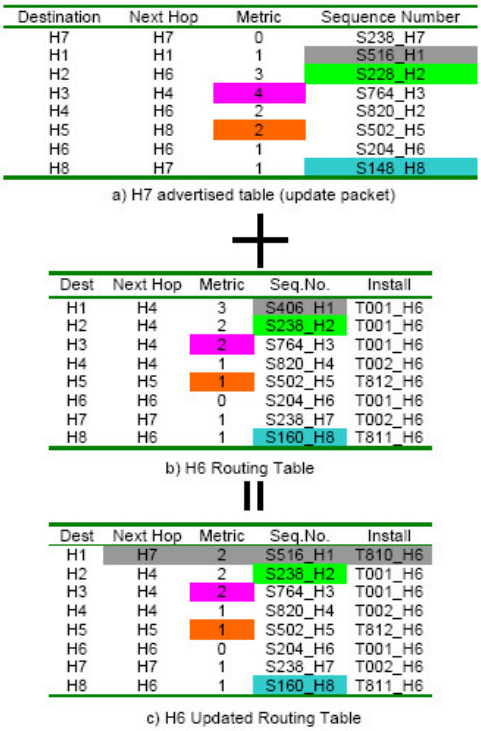


Dest	Next Hop	Metric	Seq.No.	Install
H1	H4	3	S406_H1	T001_H6
H2	H4	2	S128_H2	T001_H6
H3	H4	3	S564_H3	T001_H6
H4	H4	1	S710_H4	T002_H6
H5	H7	3	S392_H5	T001_H6
H6	H6	0	S076_H6	T001_H6
H7	H7	1	S128_H7	T002_H6
H8	H7	2	S050_H8	T002_H6

Figura 22 – Exemplo de roteamento de um pacote no DSDV

Existem dois tipos de pacotes de atualização: o *full dump*, que considera toda informação de roteamento disponível e o *incremental*, que só considera as informações de roteamento que sofreram mudança desde o último *full dump*. Na figura 23 é mostrado um exemplo de como um nodo manipula um pacote de atualização do tipo *incremental*.

Os pacotes de atualização *full dump* dificilmente são transmitidos quando um pequeno movimento de nodos móveis está acontecendo. Atualizações do tipo *incremental* são transmitidas entre atualizações *full dump* quando ocorrem mudanças parciais da tabela de roteamento como o recebimento de um novo *sequence number* ou mudanças significantes de rota (como mostrado na figura 23a) [HEG03].



Destination	Next Hop	Metric	Sequence Number
H7	H7	0	S238_H7
H1	H1	1	S516_H1
H2	H6	3	S228_H2
H3	H4	4	S764_H3
H4	H6	2	S820_H2
H5	H8	2	S502_H5
H6	H6	1	S204_H6
H8	H7	1	S148_H8

a) H7 advertised table (update packet)

Dest	Next Hop	Metric	Seq.No.	Install
H1	H4	3	S406_H1	T001_H6
H2	H4	2	S238_H2	T001_H6
H3	H4	2	S764_H3	T001_H6
H4	H4	1	S820_H4	T002_H6
H5	H5	1	S502_H5	T812_H6
H6	H6	0	S204_H6	T001_H6
H7	H7	1	S238_H7	T002_H6
H8	H6	1	S160_H8	T811_H6

b) H6 Routing Table

Dest	Next Hop	Metric	Seq.No.	Install
H1	H7	2	S516_H1	T810_H6
H2	H4	2	S238_H2	T001_H6
H3	H4	2	S764_H3	T001_H6
H4	H4	1	S820_H4	T002_H6
H5	H5	1	S502_H5	T812_H6
H6	H6	0	S204_H6	T001_H6
H7	H7	1	S238_H7	T002_H6
H8	H6	1	S160_H8	T811_H6

c) H6 Updated Routing Table

Figura 23 – Exemplo de atualização de informações de rota

3.5.2 *Dynamic Source Routing Protocol* (DSR)

O DSR é um protocolo de roteamento reativo, ou sob demanda. Sua característica principal, é a utilização de roteamento por fonte (*source routing*), onde o nó que está originando um pacote sabe toda a rota salto a salto até o destino. Assim, o DSR permite que a estação originadora do pacote determine o caminho que será utilizado pelo pacote na rede para chegar até o seu destino. Esse caminho é listado no cabeçalho do pacote de dados e é chamado de *source route*. O remetente então transmite o pacote para o primeiro *host* identificado na rota (executa o primeiro salto). Quando um *host* receber um pacote, e não é o destino final do pacote, simplesmente transmite o pacote ao próximo *host* identificado na rota do pacote (executa o próximo salto). Quando o pacote alcança seu destino final é entregue ao software de camada de rede naquele *host*.

Cada nó móvel que forma a rede *Ad Hoc* mantém uma *cache* de rotas na qual armazena as rotas descobertas. Quando um pacote é enviado a outro nó, o remetente primeiro confere sua *cache* para verificar se já não possui uma rota para o determinado destino. Se uma rota é achada o remetente usa esta rota para transmitir o pacote. Se

nenhuma rota é achada, o remetente tenta descobrir uma usando o protocolo de descoberta de rota (*route discovery protocol*). Enquanto espera pela descoberta da rota o nó continua seu processo normalmente, e armazena o pacote a ser roteado num buffer para poder transmiti-lo quando a rota for descoberta, ou pode descartar o pacote, deixando a tarefa de retransmissão para as camadas superiores do protocolo, caso necessário. A cada entrada na cache de rotas é associado um período de vencimento depois do qual a entrada é removida da cache.

Enquanto um nó estiver usando qualquer rota, a operação correta e contínua desta rota é monitorada por este nó. Se o remetente, o destinatário, ou qualquer um dos nós contidos na rota se mover para fora do alcance de transmissão, no próximo salto ou no salto anterior, esta rota não poderá mais ser utilizada para alcançar o nó destino. A rota também não estará mais avaliável se algum nó presente na rota falhar. Este monitoramento é chamado de manutenção da rota (*route maintenance*). Quando um problema com a rota em uso for detectado, o mecanismo de descoberta de rotas pode ser usado novamente para descobrir uma nova e correta rota para o destino.

Como descrito acima, o protocolo DSR é composto de dois mecanismos que trabalham juntos para permitir a descoberta e manutenção de rotas:

• *Route Discovery* é o mecanismo através do qual um nó A, que deseja enviar um pacote ao nó destino B, obtém uma rota para B. Este mecanismo apenas é usado quando o nó que deseja enviar um pacote ainda não possui uma rota para o nó destino.

Um nó que deseja descobrir uma rota executa um *broadcast* de um pacote de pedido de rota que será recebido por todos os nós dentro do alcance da transmissão. O pacote de pedido de rota identifica o nó destino da rota que esta sendo solicitada. Se a descoberta de rota tem êxito o nó remetente (nó que iniciou a descoberta da rota) recebe um pacote de resposta que lista uma seqüência de nós (pulos) da rede através dos quais o nó destino é alcançado. Para descobrir pedidos de requisição de rota duplicados, cada nó mantém uma lista com os pares (*initiator address, request id*) que foram recentemente recebidos. *Initiator address* é o endereço do nó que solicitou a descoberta da rota e *request id* é um identificador único da requisição.

Quando um nó recebe um pacote de pedido de rota, age de acordo com os seguintes passos:

1. Se o par (*initiator address, request id*) para esta requisição é encontrado na lista das requisições recebidas recentemente, então este pedido é descartado.

2. Se o endereço local esta contido na lista de nós por onde a pacote deve passar para chegar ao destino, então este pedido é descartado.

3. Se o endereço do destinatário é igual ao endereço local, então envia uma cópia da seqüência de nós por onde a mensagem passou para chegar a este nó (destino) para o nó que fez o pedido da rota. Caso este nó não seja o destino, mas conhece a rota para se chegar ao destino, esta rota é concatenada aos nós por onde a mensagem já passou é esta seqüência é enviada ao nó que fez o pedido da rota.

4. Senão adiciona o endereço local na lista de nós por onde a mensagem passou e executa um novo *broadcast* com a requisição.

A resposta enviada ao nó que fez o pedido da rota (*route reply*) é gerada quando a requisição chega no nó destino ou em um nó que possui em sua *cache* uma rota para o nó destino ainda não vencida. Para retornar esta resposta é preciso ter uma rota para o nó iniciador (nó que fez o pedido da rota). Caso a *cache* possua uma rota para o iniciador ela é usada, caso contrário, se *links* simétricos são suportados a resposta seguirá o caminho inverso da rota descoberta. Se *links* simétricos não são suportados uma nova descoberta de rota é iniciada transportando (*piggyback*) a resposta. Este mecanismo de descoberta de rotas e propagação da resposta é exemplificado nas figura 24 e 25, respectivamente.

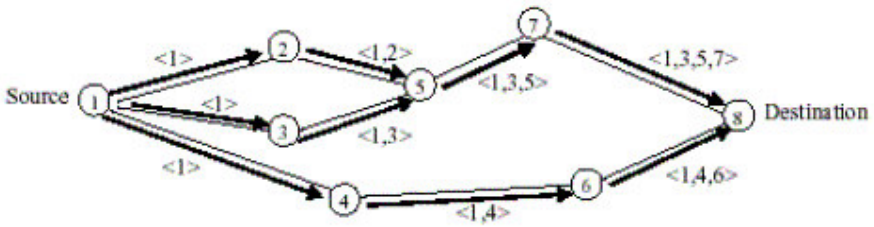


Figura 24 – Descobrimento da rota

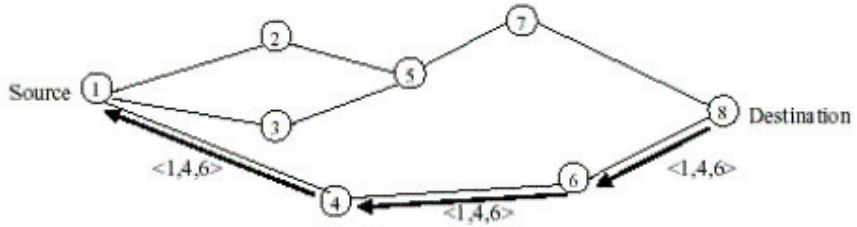


Figura 25– Propagação da resposta

- *Route Maintenance* é o mecanismo através do qual um nó é capaz de detectar, enquanto esta utilizando uma rota, se a topologia da rede sofreu mudanças e a rota não

pode mais ser usada por algum motivo, como uma quebra no link, isto é conseguido através do uso de pacotes de erros de rota (*route error packets*) e de reconhecimentos (*acknowledgments*). Este mecanismo apenas é utilizado quando um nó esta usando alguma rota.

Este protocolo utiliza roteamento dinâmico que se adapta rapidamente quando a movimentação dos nós é freqüente, e requer um pequeno ou nenhum *overhead* durante períodos nos quais os nós se movem com uma menor freqüência.

Uma descrição detalhada deste protocolo de roteamento, juntamente com outras análises e testes de performance podem ser encontrada em [JOH96] e [JOH01].

3.5.3 Principais ataques nos protocolos de roteamento

Os ataques realizados nos protocolos de roteamento em redes *Ad Hoc* podem ser classificados em dois tipos:

- ataques de quebra de rotas: este ataque tenta fazer com que pacotes de dados legítimos sejam roteados para caminhos disfuncionais. Um exemplo deste ataque é o *black hole*, onde o atacante, através do envio de pacotes de roteamento forjados, roteia todos os pacotes enviados para algum destino para ele e depois descarta estes pacotes.
- ataques de consumo de recursos: este ataque injeta pacotes na rede com o objetivo de consumir recursos como: largura de banda, poder de processamento e memória. Um exemplo deste ataque é o *routing loop*, onde o atacante envia pacotes de roteamento forjados, que cria um ciclo passando pelos nós sem achar o destino, consumindo recursos.

Outro exemplo de ataque de consumo de recursos é a inserção de pacotes de dados ou de controle extra na rede, os quais causarão um consumo de largura de banda e outros recursos. Se o protocolo de roteamento consegue evitar que um atacante insira *loops* no roteamento, e forçar um tamanho máximo para o roteamento, este ataque tem efeito limitado. A descrição de outros ataques e outra classificação para os diversos tipos de ataques podem ser encontrados respectivamente em [YIH02a] e [JUN04].

Para uma aplicação (camada de aplicação), estes dois tipos de ataques podem ser classificados como ataques de negação de serviço.

3.5.4 Protocolos de roteamento seguro

Muitos protocolos são propostos focados na idéia de encontrar o menor caminho entre dois nodos o mais rápido possível, entretanto existem aplicações que requerem mais do que a certeza da determinação da menor rota [YI801].

O projeto de protocolos de roteamento seguro para redes *Ad Hoc* apresenta algumas dificuldades, devido à natureza geralmente altamente dinâmica de uma rede *Ad Hoc* e devido à necessidade de operar eficazmente com recursos limitados, incluindo a largura da banda de rede, capacidade de processamento da CPU, memória e bateria (energia) de cada nodo na rede. Os protocolos de roteamento sem segurança para redes *Ad Hoc* são freqüentemente aperfeiçoados para propagar novas informações rapidamente de acordo com as novas modificações, exigindo interações do protocolo de roteamento mais rápidas e freqüentes do que as exigidas entre os nodos de uma rede tradicional (por exemplo, cabeada) [YIH02b].

A maioria dos protocolos de roteamento propostos na literatura está assumindo ambientes não hostis. Devido à mudança dinâmica de topologia, ambiente aberto e falta de infra-estrutura de segurança centralizada, um MANET é extremamente vulnerável a presença de nodos maliciosos e certos tipos de ataques que podem acontecer. Não há nenhuma garantia que o caminho de comunicação (escolhido no roteamento) é livre de nodos maliciosos, os quais não obedecerão ao protocolo empregado e tentarão interferir nas operações da rede. Focalizando estas preocupações, recentemente foram propostos vários protocolos de roteamento seguro: SAODV (*Secure Ad Hoc On-Demand Distance Vector*), *Ariadne*, SEAD (*Secure Efficient Ad hoc Distance vector*), SRP (*Static Routing Protocol*), entre outros. Alguns protocolos, como o SAODV, permitem o uso do IPSec nas transmissões em uma MANET.

SEAD e *Ariadne* são o estado-da-arte dos protocolos seguros para redes *Ad Hoc* os quais são representantes dos dois principais grupos de protocolos: pró-ativos e reativos. SEAD é um protocolo pró-ativo baseado no DSDV e o *Ariadne* é um protocolo reativo baseado no DSR. Ambos serão utilizados em nosso ambiente de simulação e desta forma, serão explicados de forma mais completa nos próximos itens deste capítulo.

3.5.4.1 *Secure Efficient Ad Hoc Distance Vector Routing Protocol* (SEAD)

SEAD é um novo protocolo pró-ativo de roteamento seguro para redes *Ad Hoc* que faz uso do roteamento por vetor de distância. Muitos protocolos de roteamento para redes *Ad Hoc* já foram implementados baseados no vetor de distância, mas geralmente considerando um ambiente confiável.

No projeto do SEAD [YIH02b] foram utilizadas cuidadosamente primitivas criptográficas em cada parte de suas funcionalidades para criar um protocolo eficiente, prático e que fosse robusto contra múltiplos tipos de ataques sem coordenação os quais podem gerar estados incorretos em qualquer nodo da rede. Juntamente com a segurança gerada pela aproximação existente entre a camada física e a camada MAC na pilha de protocolo de rede, o protocolo SEAD provê uma base para a operação segura de uma rede *Ad Hoc*.

A implementação do protocolo SEAD foi inspirada no DSDV-SQ, uma versão do protocolo DSDV. O DSDV-SQ apresenta um melhor desempenho como foi demonstrado em comparações realizadas com outras versões do protocolo DSDV através de simulações realizadas considerando redes *Ad Hoc* [BRO98] [JOH99].

SEAD trata de atacantes que modificam informações de roteamento transmitidas por broadcast durante a fase de atualização do protocolo DSDV-SQ. Desta forma, o roteamento pode ser interrompido se o atacante modificar os campos *sequence number* ou *metric* da mensagem de atualização da tabela de roteamento. *Replay attacks* também é tratado pelo SEAD.

Para garantir segurança no protocolo DSDV-SQ, SEAD faz o uso de cadeias de *hash* unilaterais eficientes ao invés de retransmitir usando operações de criptografia assimétrica. Entretanto o protocolo SEAD assume que algum nodo tenha um mecanismo para distribuir elementos autenticados da cadeia de *hash* os quais podem ser usados para autenticar todos os outros elementos da cadeia. Uma solução tradicional é garantir que a distribuição de chaves seja realizada por uma entidade confiável que assina certificados de chave pública para cada nodo; cada nodo então poderá usar sua chave pública para assinar um elemento da cadeia de *hash* e distribuí-lo [YIH02b].

A idéia básica do protocolo SEAD é autenticar o *sequence number* e o *metric* de uma mensagem de atualização da tabela de roteamento usando elementos da cadeia de *hash*. Além disso, o receptor das informações de roteamento SEAD também autentica o

remetente, assegurando que as informações de roteamento foram originadas do nodo correto.

Cada nodo usa um elemento autêntico específico de sua cadeia de *hash* em cada atualização de roteamento a qual é enviada para o próprio nodo (*metric* 0). Baseado neste elemento inicial, a cadeia de *hash* unilateral provê autenticação para o mais baixo salto no campo *metric* em outras atualizações de roteamento para este nodo. O uso do valor de *hash* correspondente ao *sequence number* e ao *metric* em uma entrada de atualização de roteamento previne que qualquer nodo anuncie uma rota para algum destino reivindicando um *sequence number* maior que o *sequence number* do próprio destino. Igualmente, um nodo não pode anunciar uma rota melhor que aquelas anunciadas, pois o *metric* existente em uma rota não pode ser diminuído devido à natureza da cadeia de *hash* unilateral [YIH02b].

Quando um nodo recebe uma atualização de roteamento, este confere a autenticidade da informação de cada entrada na atualização usando o endereço do destinatário, o *sequence number* e o *metric* da entrada recebida juntamente com o último valor de *hash* recebido da cadeia de *hash* do destinatário. Depois é feito o *Hashing* dos elementos recebidos quantas vezes for o correto confirmando a autenticidade da informação recebida caso o valor de *hash* calculado e o valor de *hash* autêntico forem iguais.

A fonte de cada mensagem de atualização de roteamento no SEAD também deve ser autenticada, caso contrário, pode permitir a criação de *loops* através do *impersonation attack*. Duas maneiras são propostas para realizar a autenticação: o primeiro é baseado em um mecanismo de autenticação por broadcast como o TESLA, e o segundo é baseado no uso de Códigos de Autenticação de Mensagem, assumindo a existência de uma chave secreta compartilhada entre cada par de nodos na rede.

3.5.4.2 *Ariadne Protocol*

O protocolo DSR forma a base para o projeto deste protocolo, por isto sua forma de funcionamento é do tipo *on-demand*, descobrindo as rotas dinamicamente conforme a necessidade. O *Ariadne* previne ataques por meio de autenticação com rotas descomprometidas (formadas por nós descomprometidos), previne também um grande número de tipos de ataques de negação de serviço. Além disso, o protocolo *Ariadne* não utiliza compromisso com os nós, confiando apenas em eficientes primitivas de

criptografia simétrica, não requerendo *hardware* confiável, nem alto poder de processamento.

As mensagens de roteamento podem ser autenticadas de três formas [YIH02a]: troca secreta entre cada par de nós, troca secreta entre os nós comunicantes combinada com autenticação *broadcast*, ou assinaturas digitais.

Uma forma de uso é a combinação do *Ariadne* com TESLA, um eficiente esquema de autenticação *broadcast* que requer uma sincronização fraca de tempo entre os nós da rede, de modo que um nó possa estimar o tempo de transmissão fim-a-fim para qualquer outro nó da rede.

A maioria dos ataques de rompimento de rota são causados por injeção maliciosa ou alteração de dados de roteamento. Para prevenir estes ataques cada nó que interpreta as informações de roteamento deve verificar a origem e integridade destes dados, isto é, autenticar os dados. Idealmente, o iniciador da descoberta de rota (*route discovery*) deve verificar a origem de cada dado contido na resposta (*route reply*).

Os mecanismos de descoberta de rotas e manutenção de rotas sofreram algumas alterações em relação aos usados pelo DSR, novos campos foram adicionados as mensagens no intuito de atingir os objetivos de segurança especificados por este protocolo.

Uma descrição detalhada das funcionalidades deste protocolo pode ser encontrada em [YIH02a], onde também são apresentados alguns testes de performance deste protocolo comparando-o com o DSR, seu originador.

3.6 Tipos de Filas

Gerenciamento de filas é o processo pelo qual o servidor de rede ou uma outra entidade, enfileira as chamadas e outras transações, determinando a ordem na qual estas transações devem ocorrer dentro do sistema. O gerenciamento de filas envolve a combinação de fatores como a priorização de chamadas e escolhas predefinidas.

Se o tráfego de entrada for maior do que o tráfego de saída da interface, os pacotes que entram são enfileirados de acordo com o algoritmo de enfileiramento. No contexto das filas podem ser avaliados, por exemplo, o descarte de pacotes e a latência. Existem vários tipos de filas que podem ser utilizadas em redes *wireless*: FIFO (*First In First Out*), SFQ (*Stichastic Fair Queue*), RED (*Random Early Detection*), FQ (*Fair Queue*), DRR (*Déficit Round Robin*), CQ (*Custom Queue*), WFQ (*Weighted Fair*

Queue), CBWFQ (*Class-Based Weighted Fair Queue*), WRED (*Weighted Random Early Detection*) e TBF (*Token* Bucker *Filter*). A seguir apresentaremos uma breve descrição de algumas delas.

- FIFO: este tipo de fila simples é o mais popular, sendo utilizado tanto no escopo das redes de computadores quanto dos sistemas operacionais, através do escalonamento de processos. Neste algoritmo os pacotes vão sendo armazenados na ordem em que eles chegam, e assim que possível são enviados nesta mesma ordem. Se chegarem pacotes depois da fila ter atingido o seu limite, estes serão descartados. A decisão sobre a prioridade de atendimento é determinada pela ordem de chegada. A disciplina FIFO não garante limites quanto à vazão, atraso, *jitter* ou perda de pacotes.
- SFQ: aqui o fluxo é dividido em várias sessões de filas, usando um algoritmo de *round robin* para impedir que uma sessão tome conta da fila de transmissão. Não existe uma fila para cada sessão, mas um número restrito de filas entre os quais os diversos fluxos são divididos através de um esquema de *hash*. Os fluxos que estiverem na mesma fila terão tratamento equivalente. A desvantagem deste esquema é que um fluxo mais importante será desfavorecido se cair na mesma fila de um fluxo de menor prioridade.
- RED: neste tipo de fila ocorre uma constante monitoração da carga da rede e quando um aumento do congestionamento é percebido (aumento da fila de entrada) pacotes começam a ser descartados estocasticamente e pacotes de sinalização são enviados aos dispositivos geradores do tráfego, para que estes diminuam o fluxo, diminuindo o congestionamento. Um outro nome para este tipo de fila é *Random Early Drop*.
- FQ: um tempo teórico é atribuído a cada pacote que chega na fila, e quando possível, os pacotes com tempo menor são enviados antes dos pacotes com tempo maior.
- DRR: este tipo de fila utiliza um *token* e diversas filas. Em cada ciclo o *token* vai passando de fila em fila e apenas a fila que estiver com o *token* poderá liberar pacotes em um determinado ciclo. Em cada ciclo as filas recebem um número fixo de créditos e só podem enviar até o limite deste crédito. Se transmitir menos, os créditos são guardados para o próximo ciclo e caso o pacote seja maior do que os créditos ganhos terá que esperar o próximo ciclo.

3.7 Conclusões

Neste capitulo foram abordados alguns aspectos referentes às redes sem fio *Ad Hoc*, bem como os problemas relacionados com esta topologia juntamente com as vantagens e desvantagens na sua utilização. Alguns protocolos de roteamento foram discutidos e por fim algumas políticas de filas foram analisadas.

Vários problemas envolvendo questões relacionadas com segurança que existiam neste tipo de topologia já foram sanados, ou minimizados, através de pesquisas que deram origem a uma nova categoria de protocolos, os protocolos de roteamento seguro. No entanto, ainda existe muito a ser pesquisado e descoberto no intuito de tornar esta tecnologia cada vez mais segura, estável e com uma alta performance.

4. ACORDOS DE NÍVEL DE SERVIÇO – SLA

O objetivo deste capítulo é apresentar e descrever os acordos de níveis de serviços, destacando os tipos de SLAs e as métricas usadas no seu estabelecimento.

4.1 Introdução

Cada vez mais as empresas estão contratando serviços baseados em níveis de serviços, conhecidos como SLAs (*Service Level Agreements*). Para administrar esses contratos é necessário que tanto as prestadoras de serviços como as empresas contratantes possuam um gerenciamento dos parâmetros contratos. A simplicidade e conhecimento pleno dos parâmetros contratados são fundamentais para a boa gestão do contrato, evitando conflitos pessoais e contratuais entre empresas e provedores dos serviços [FAG04].

A falta de estabilidade pavimenta o surgimento e conceito de SLAs e origina-se do fato de que pessoas querem pagar por um serviço que se comportará do mesmo modo o tempo todo.

SLAs estão se tornando um componente necessário no dia a dia de uma empresa que utiliza redes de computadores. Nestes contratos, os parâmetros de performance são especificados, os quais refletem o bom funcionamento da rede. Embora estes contratos normalmente englobem os serviços de telecomunicações, eles também podem incluir os serviços de tecnologia de informação e outros fatores referentes ao comércio dentro de uma companhia.

De uma maneira resumida, SLA é um contrato entre um provedor de serviços e o cliente, o qual estipula os níveis de serviços exigidos. Um SLA deve conter os níveis de serviços especificados, opções de suporte, as penalidades pela quebra do contrato (não fornecimento de serviços no nível especificado), possíveis recompensas por exceder o nível de serviço especificado, o tempo de manutenção, a que software ou hardware será fornecido e a que taxa.

4.2 Alguns conceitos

Quando estamos falando sobre SLAs alguns conceitos são importantes para a compreensão do assunto. São eles:

1. **Clientes:** também chamado de usuário final, é o indivíduo que usa o produto depois que ele for totalmente desenvolvido e comercializado. O termo é útil porque distingue duas classes de usuários, os que requerem um produto livre de *bugs* (usuários finais) e os que podem usar o mesmo produto para propósitos de desenvolvimento. Pode ser uma companhia que faz uso dos serviços oferecidos por um provedor ou uma organização que receber serviços de um provedor e fornece serviços a outros clientes, sendo assim também um provedor de serviços.
2. **Provedor de serviço (ASP):** em inglês *Application Service Provider*, é uma entidade que administra e distribui serviços e soluções a clientes por uma área larga. No intuito de fornecer determinados serviços, um provedor pode contatar outros provedores de acordo com suas necessidades. Os provedores podem ter caráter comercial, não lucrativo ou organizações governamentais. Os provedores podem ser divididos em 5 categorias [ASP04]:
   - Empresas: fornecem aplicações comerciais.
   - Local\Regional: fornecem uma grande variedade de serviços negócios menores em uma área local.
   - Especialistas: fornecem aplicações para uma necessidade específica, como serviços *Web* ou humanos recursos.
   - Mercado vertical: fornece suporte a uma indústria específica, como cuidado médico.
   - Negócios de volume: fornecem suporte a pequenos e médios negócios com grande volume de aplicações pré - embaladas.

   No contexto das redes sem fio, os provedores de serviços são chamados de WASP (*Wireless Application Service Provider*), os quais fornecem os mesmos serviços de um ASP para este tipo de rede.
3. **Ponto de acesso ao serviço:** é o lugar onde o serviço é fornecido ao cliente, este ponto delimita a responsabilidade do provedor de serviços.

4. **Serviço:** é um conjunto de operações que um cliente compra ou aluga junto ao provedor de serviços.
5. **Elemento de serviço:** formam a base para a composição dos serviços e são as entidades que implementam os serviços, podendo ser em *software* ou *hardware*.
6. **Relatório de desempenho:** é uma espécie de documento onde estão as informações de como o serviço contratado está sendo fornecido pelo provedor de serviços e utilizado pelo cliente.

4.3 Caderno de métricas

Compõe a parte técnica do SLA. É o local onde os indicadores são relacionados, descritos e definidos quantitativamente. No caderno de métricas definem-se também as metas de qualidades, especificando-se limiares para os indicadores [PER03]. No caderno de métricas, encontram-se os detalhes técnicos chamados de SLS (*Service Level Specifications*) onde estão definidas as métricas de performance (taxa de transmissão, latência, taxa de perdas de pacotes) e as agregações de fluxos (classificação, medição, etc...).

4.4 O que incluir em um SLA?

Os elementos mais importantes a serem incluídos num SLA, segundo [TEC04], são:

- O nível de disponibilidade necessário;
- O quanto poderá ser gasto;
- O nível de *throughput* necessário;
- O nível de *throughput* necessário em períodos de altas taxas de tráfego;
- O tempo de resposta do provedor quando uma falha ocorre;
- O nível de redundância necessário para manter o serviço;
- O nível de segurança desejado;
- A freqüência com que a performance deve ser avaliada.

Como pode ser observado, esta lista não é completa e pode mudar de acordo para acordo, dependendo das necessidades de cada cliente.

4.5 Tipos de SLA

Existem dois tipos de acordos de nível de serviços: SLA Externo e SLA Interno. O primeiro é bastante rigoroso, pois geralmente este contrato é firmado entre duas empresas. Já o segundo é um acordo entre diferentes áreas de uma mesma empresa, que visa assegurar qualidade nas diversas áreas da empresa, geralmente utilizado em grandes companhias.

Em [PER03] é proposto a gerencia e o estabelecimento de um SLA pró-ativo, o qual detecta a possibilidade de quebra em alguma métrica do acordo com antecedência (antes que realmente ocorra) e realiza determinadas ações no intuito de solucionar a possível degradação da rede. Caso esta solução não seja possível, novas ações são tomadas no sentido de renegociar o acordo com um servidor de SLAs.

4.6 Métricas para o estabelecimento de um SLA

As métricas para o estabelecimento de um SLA podem ser classificadas em independentes da tecnologia (percentual de disponibilidade, tempo de recuperação de falhas, tempo de instalação de serviços, etc...) e dependentes da tecnologia (perda, retransmissão, latência, etc...).

Dependendo do tipo de aplicação determinadas métricas serão ou não importantes, no escopo deste trabalho daremos atenção especial as seguintes métricas.

4.6.1 Vazão

Também denominada de *througput*, é uma medida de transmissão de dados que determina a quantidade de dados movida de um nó a outro da rede em um determinado período de tempo. Em termos práticos, as aplicações geram vazões que devem ser atendidas pela rede. A vazão na maioria das redes sofre variações no decorrer no tempo e é normalmente medida em Kbps ou Mbps.

4.6.2 Latência

É o tempo que um pacote leva para ir de um ponto a outro da rede, normalmente medido em milissegundos. Quanto menor for a latência, melhor o tempo de resposta da

rede, se a latência for muito grande torna-se difícil o uso de interatividade na rede. A forma de se calcular a latência é o tempo em que o pacote chega ao canal de comunicação (emissor) e a observação do ultimo *bit* do pacote (receptor).

Os principais responsáveis pela latência são o atraso de transmissão, codificação e decodificação e empacotamento e desempacotamento [PER03].

4.6.3 *Jitter*

É a variação do atraso. Em redes a pacotes, os fluxos são adicionalmente divididos em blocos de dados, e cada bloco é transmitido em seqüência. Se a rede é capaz de enviar todos os blocos com uma seqüência uniforme, então cada bloco deveria chegar no destino após um atraso uniforme. Muitas redes não garantem um atraso uniforme para seus usuários. Variações no atraso são comuns e causadas por muitos fatores, tais como: diferenças de tempo de processamento dos pacotes, diferenças de tempo de acesso à rede e diferenças de tempo de enfileiramento [SIL04a].

4.6.4 Perda de Pacotes

É o índice que mede a taxa de sucesso na transmissão de pacotes entre dois pontos da rede. Quanto menor a perda de pacotes, maior a eficiência da rede.

4.7 Conclusões

Os SLAs são uma excelente forma de contratar serviços, pois estabelece os parâmetros que devem ser atingidos pelo provedor. Porém, os contratos baseados em SLA devem ter um gerenciamento eficiente para evitar perdas e desgaste no relacionamento entre o cliente e o provedor de serviços.

5. SIMULAÇÕES

O objetivo deste capitulo é descrever a ferramenta utilizada e o ambiente onde foram realizadas as simulações, bem como apresentar e analisar os dados coletados. Neste trabalho foram avaliados três tipos de filas (FIFO, DRR e RED) e quatro protocolos de roteamento (DSR, DSDV, *Ariadne*, SEAD).

5.1 Introdução

Simulação é uma técnica muito utilizada para a avaliação de diversas características de sistemas. É usada principalmente quando o sistema a ser analisado ainda não esta disponível, principalmente em muitas situações onde esta é a forma mais fácil e menos custosa de se prever o comportamento de um sistema.

No entanto, apesar do uso de simulações nos proporcionar várias vantagens, esta abordagem apresenta algumas limitações dependendo do simulador utilizado.

5.2 *Network* Simulator (NS)

O *Network Simulator* é um *software* de simulação de redes. Ele trabalha na simulação de protocolos como o TCP, protocolos de *multicast* em redes com ou sem fio, simulando até mesmo roteamento de pacotes, entre outras coisas. Este é um aplicativo desenvolvido na Universidade de *Berkeley* em C++ e OTcl (*Object Tool Command Language*), sendo um *software* orientado a objeto. O NS é um simulador de redes dirigido por eventos que simula vários tipos de redes IP. Uma visão simplificada do NS pode ser observada na figura 22 [UNB04]. Como pode ser observada na figura 22, a entrada é um *script* escrito em OTcl.

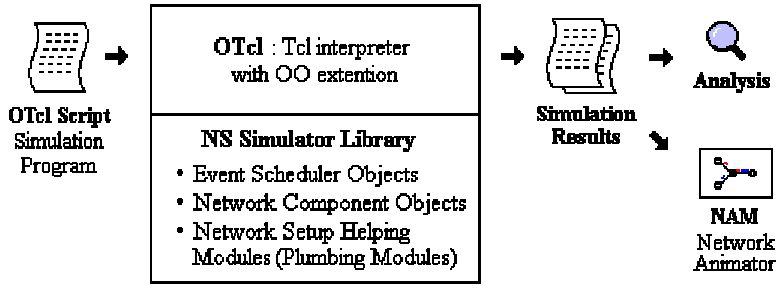


Figura 26 – Visão simplificada do NS

Quando uma simulação é feita, o NS produz um ou mais arquivos de saída baseadas em texto que contém dados da simulação detalhados. Os dados podem ser utilizados para análise de simulações ou como entrada para uma ferramenta de simulação gráfica chamada *Network Animator*, que possui uma interface gráfica bastante amigável e tem um controlador de velocidade. Atualmente o NS se encontra na versão 2.27, a qual foi utilizada nas simulações realizadas neste trabalho.

5.3 Ambiente de Simulação

Para realizar as simulações procurou-se modelar um sistema o mais próximo possível com um ambiente real. Este ambiente caracteriza-se por possuir um desempenho que atende as necessidades de comunicação tanto de usuários que o utilizam como dos próprios dispositivos *wireless* pertencentes ao mesmo.

Na realização das simulações alguns parâmetros foram variados de acordo com a análise a ser realizada, no entanto outros nunca sofreram alterações. A tabela 1 apresenta os parâmetros utilizados nas simulações, nela podemos observar quais foram os parâmetros que foram variados e quais permaneceram fixos ao longo das simulações.

Parâmetros que permaneceram fixos	**Parâmetros que foram variados**
Tipo de canal: *Wireless Channel*	Tipo de Fila: FIFO, DRR e RED
Modelo de propagação: *Two Ray Ground*	Protocolo de Roteamento: DSR, DSDV, *Ariadne* e SEAD
Tipo de interface: *Wireless Phy*	Número de estações móveis: 5, 10, 15, 20
Tipo de MAC: 802.11	Tamanho dos pacotes: 64 e 256 bytes
Tipo de camada de ligação: LL	
Modelo de antena: *Omni Antenna*	
Dimensões do ambiente: 300 X 300m	
Número máximo de pacotes na fila: 10	
Tempo de simulação: 400 segundos	
Protocolo de conexão: UDP	
Tipo de tráfego: Dados – CBR	
Largura de banda: 2 Mbps	
Modelo de mobilidade: *WayPoint*	

Tabela 1 – Parâmetros utilizados nas simulações

A obtenção dos padrões de comunicação e movimentação se deu através do uso de scripts presentes na distribuição do NS utilizada. O simulador utiliza estes padrões para variar a movimentação dos nós e a comunicação entre eles.

O parâmetro de mobilidade escolhido foi o *WayPoint*, o qual caracteriza-se por dividir o percurso de cada NM em períodos de movimentação e de pausa, ou seja, cada nó escolhe um destino e uma velocidade aleatórios na área estipulada, andando então para o destino e permanecendo lá durante o tempo de pausa, após volta a escolher outra posição e velocidade e assim sucessivamente até terminar o tempo da simulação. No padrão criado especificou-se uma velocidade de movimentação uniforme escolhida aleatoriamente entre 3 m/s e 10 m/s obtendo uma média de movimentação de 2.75 m/s, considerando o tempo de pausa que foi fixado em 30 segundos.

Este modelo de simulação exemplifica um ambiente wireless *multihop* com dimensões de 300 X 300m, onde existem 5, 10, 15 e 20 nós, os quais se comunicam com o auxilio de algum nó intermediário, quando necessário. Sendo assim, este ambiente assemelha-se, por exemplo, com um escritório ou fábrica, onde existe necessidade de comunicação tanto de usuários como de outros dispositivos *wireless*.

Os pacotes têm tamanho de 64 e 256 bytes, simulando apenas tráfego do tipo dados, mais especificamente CBR (*Constant Bit Rate*), que são enviados por um canal com largura de banda de 2 Mbps através de uma conexão UDP (*User Datagram Protocol*). O tempo de duração e ativação de cada conexão foi estabelecido de maneira aleatória, sendo assim cada uma tem a mesma probabilidade de ocorrer a qualquer momento.

Ainda foram feitas outras configurações como comprimento máximo da fila de 10 pacotes e tempo de simulação de 400 segundos.

Após a especificação destes parâmetros foi realizada a simulação de uma rede A*d Hoc*, os resultados obtidos juntamente com uma análise dos mesmos serão descritos nas próximas seções.

5.4 Ferramenta de Análise

Para a análise dos resultados da simulação optou-se pelo uso da ferramenta *Trace Graph* [MAL04], a qual lê e interpreta os arquivos gerados pelo simulador. Este *software* foi desenvolvido em Wroclaw *University of Technology - Poland* por Jaroslaw Malek e atualmente encontra-se na versão 2.02.

Este analisador realiza a análise das métricas de QoS que fazem parte do escopo deste trabalho da seguinte forma:

- Vazão: definida como o número de bytes (ou pacotes) que foram enviados dividido pelo tempo gasto.
- Latência: definida como sendo a diferença entre o tempo em que o pacote foi recebido e o tempo em que foi transmitido.
- Jitter: definido como sendo a diferença entre o atraso do pacote atual e o do próximo pacote. Sendo S(i) o tempo em que o pacote i foi enviado e R(i) o tempo em que o pacote i foi recebido, o jitter(i) =|(R(i+1)-S(i+1))-(R(i)-S(i))|.
- Taxa de Perdas: definida como o total de bytes (ou pacotes) perdidos durante a simulação. O analisador utilizado ainda permite visualizar separadamente o total de pacotes retirados\perdidos da\na fila (*dropped packets*) e o total de pacotes perdidos (que foram enviados e não chegaram no destino por outro motivo, como erro na rota por exemplo).

5.5 Resultados e análises

Neste tópico serão apresentados os resultados obtidos através das simulações, bem como fazer uma análise sobre estes resultados.

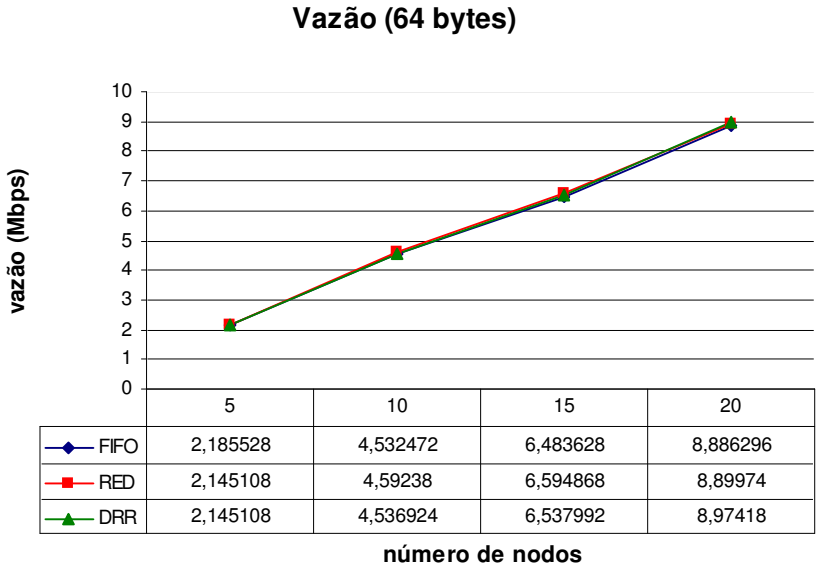
5.5.1 Análise das filas

Neste item serão analisadas as influências dos tipos de fila FIFO, RED e DRR sobre as métricas de QoS através de simulações. Para a realização destas simulações, foi fixado o DSDV como protocolo de roteamento e variou-se o tamanho dos pacotes (64 bytes e 256 bytes) e a quantidade de nodos na rede (5, 10, 15 e 20).

O número máximo de pacotes que podem ficar na fila foi definido com o valor 10, sendo que quando este valor for ultrapassado, cada fila irá descartar os pacotes de acordo com a sua política. Para a fila DRR, o parâmetro *quantum_*, que indica a quantidade em bytes que cada fluxo poderá enviar, foi fixado em 256 bytes.

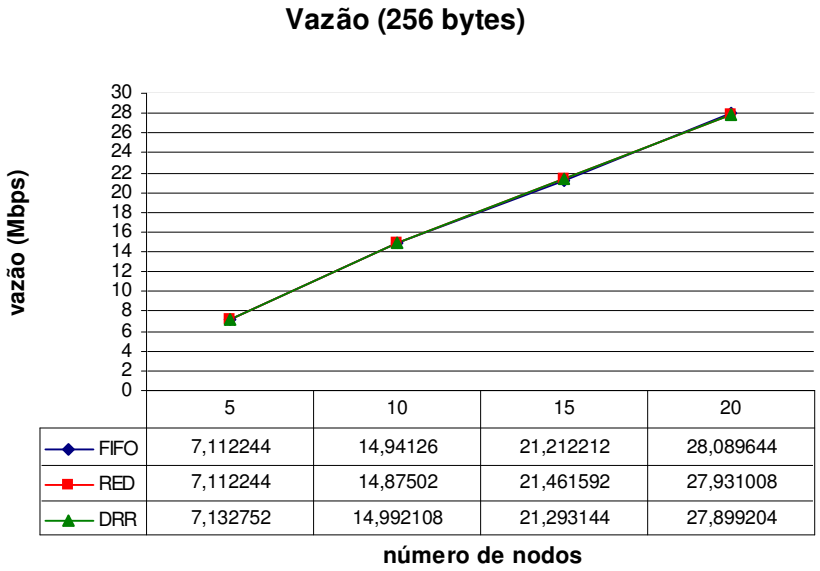
5.5.1.1 Vazão

Nesta métrica a variação dos tipos de fila não apresentou nenhuma influência considerável sobre a vazão da rede. Como pode ser observada nos gráficos 1 e 2, a vazão apresentou um crescimento linear, tanto para pacotes de 64 bytes e 256 bytes, de acordo com o aumento da quantidade de nodos que compõem a rede.



	5	10	15	20
FIFO	2,185528	4,532472	6,483628	8,886296
RED	2,145108	4,59238	6,594868	8,89974
DRR	2,145108	4,536924	6,537992	8,97418

Gráfico 1 – Vazão x Tipo de Fila (pacotes de 64 bytes)



	5	10	15	20
FIFO	7,112244	14,94126	21,212212	28,089644
RED	7,112244	14,87502	21,461592	27,931008
DRR	7,132752	14,992108	21,293144	27,899204

Gráfico 2 – Vazão x Tipo de Fila (pacotes de 256 bytes)

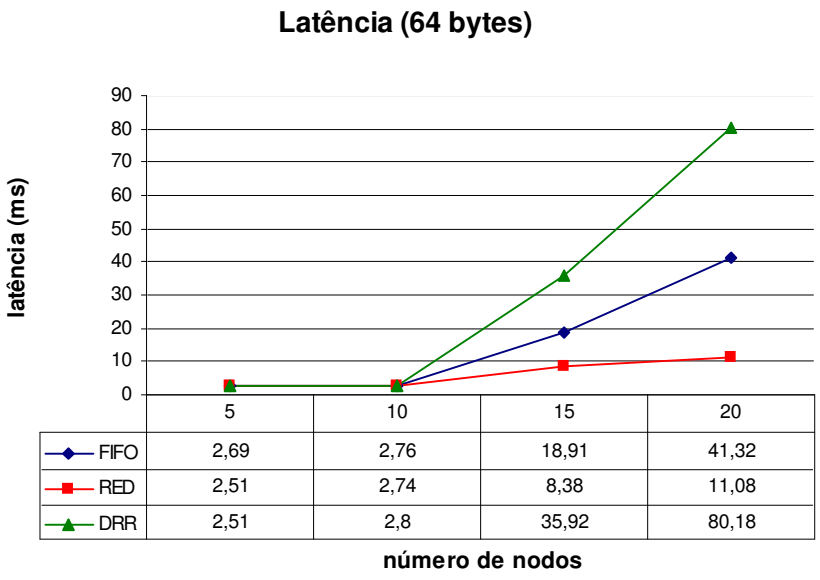
5.5.1.2 Latência

Como pode ser observado nos gráficos 3 e 4, a métrica latência apresentou um comportamento semelhante quando a rede *Ad Hoc* era formada por 5 e 10 nodos, independentemente do tipo de fila e do tamanho do pacote utilizado. Mas quando foram simuladas redes compostas por 15 e 20 nodos, as filas proporcionaram valores de latência diferenciados.

Tanto para pacotes de 64 bytes quanto para pacotes de 256 bytes, a fila do tipo RED apresentou os menores valores médios de latência principalmente na rede composta por 15 e 20 nodos. Desta forma, utilizando este tipo de fila a rede apresentou um melhor tempo de resposta, colaborando para uma comunicação de melhor qualidade entre os nodos.

Os maiores valores médios de latência foram obtidos quando foi utilizado o tipo de fila DRR. Portanto, a utilização desta fila prejudicou bastante a qualidade na comunicação entre os nodos, pois quando a latência apresenta valores elevados, torna-se difícil o uso de interatividade na rede.

A fila do tipo FIFO, na qual os pacotes vão sendo armazenados na ordem em que eles chegam, e assim que possível são enviados nesta mesma ordem, apresentou um desempenho intermediário em relação aos demais tipos de fila analisados.



	5	10	15	20
FIFO	2,69	2,76	18,91	41,32
RED	2,51	2,74	8,38	11,08
DRR	2,51	2,8	35,92	80,18

Gráfico 3 – Latência x Tipo de Fila (pacotes de 64 bytes)

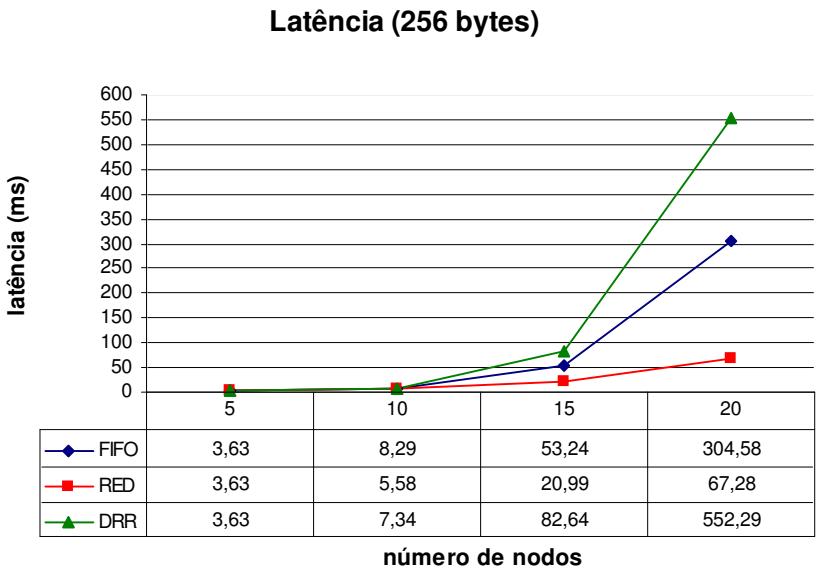


Gráfico 4 – Latência x Tipo de Fila (pacotes de 256 bytes)

5.5.1.3 *Jitter*

De forma parecida com a métrica latência, a métrica *jitter* apresentou um comportamento semelhante quando a rede *Ad Hoc* era formada por 5 e 10 nodos, como pode ser observado nos gráficos 5 e 6. Mas quando foram simuladas redes compostas por 15 e 20 nodos, as filas apresentaram valores de *jitter* diferenciados.

Tanto para pacotes de 64 bytes quanto para pacotes de 256 bytes, a fila do tipo RED apresentou os menores valores médios de *jitter* em redes compostas por 15 e 20 nodos. Desta forma, utilizando esta fila, a rede foi capaz de enviar os dados com uma boa uniformidade, contribuindo para uma melhor qualidade na comunicação entre os nodos.

Já o tipo de fila DRR apresentou os maiores valores de *jitter* para as redes compostas por 15 e 20 nodos. Este tipo de fila é composto por diversas filas e em cada ciclo as filas recebem um número fixo de créditos. Portanto, esta pode ser uma das razões para o seu baixo desempenho, pois estas filas só podem enviar até o limite deste crédito.

A fila FIFO, o tipo de fila mais simples e popular, apresentou um desempenho intermediário em relação aos demais tipos de fila analisados.

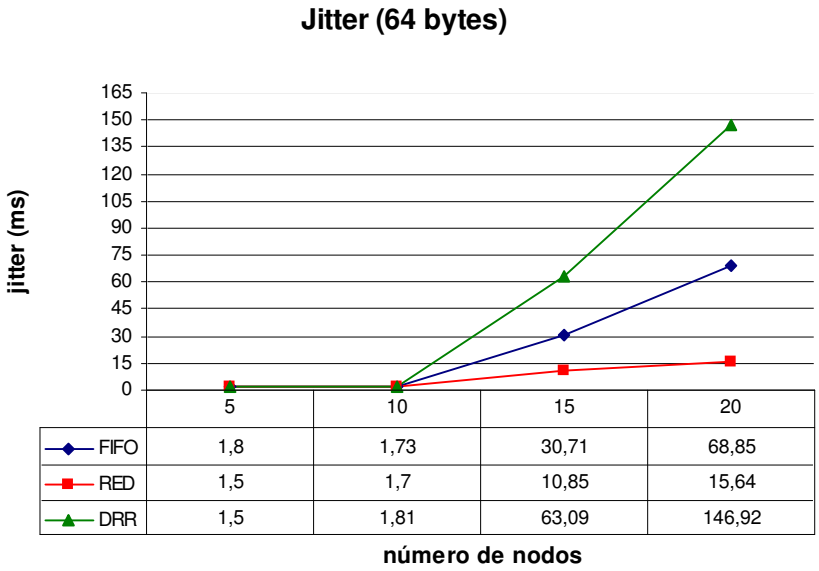


Gráfico 5 – *Jitter* x Tipo de Fila (pacotes de 64 bytes)

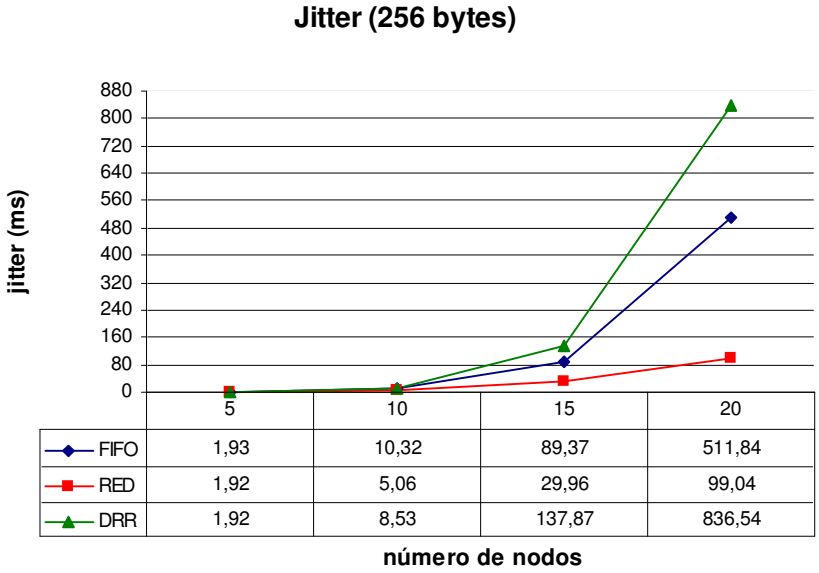


Gráfico 6 – *Jitter* x Tipo de Fila (pacotes de 256 bytes)

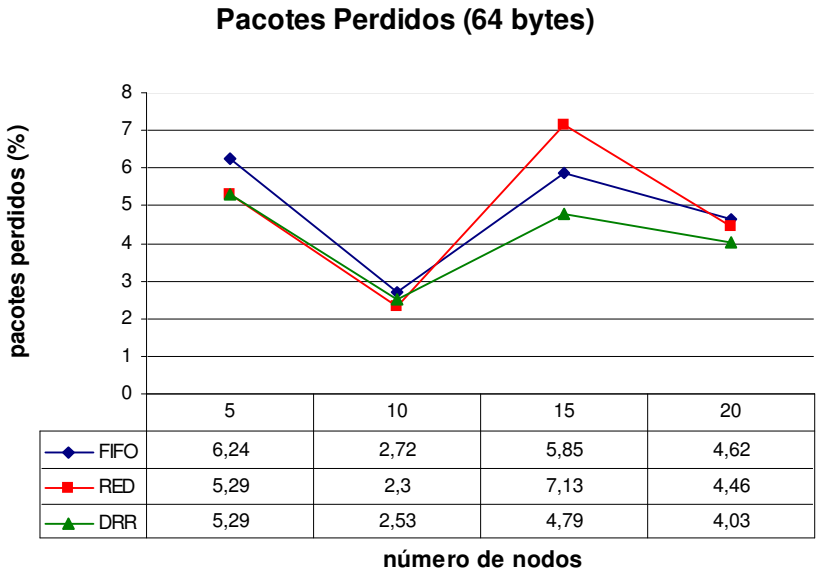
5.5.1.4 Perda de pacotes

A taxa de pacotes perdidos apresentou um comportamento semelhante nas redes compostas por 5, 10 e 15 nodos, independente do tipo de fila e do tamanho dos pacotes utilizado.

Na rede composta por 20 nodos, esta métrica apresentou um comportamento semelhante em relação aos tipos de fila mas diferente em relação ao tamanho dos pacotes. Quando o pacote tem 64 bytes, a taxa de pacotes perdidos é menor que a da rede com 15 nodos. Mas quando o pacote tem 256 bytes, a taxa de pacotes perdidos apresenta um grande incremento em relação à rede com 15 nodos (gráfico 8).

As menores taxas de pacotes perdidos foram obtidas com a rede composta por 10 nodos, independente do tipo de fila utilizado (gráfico 7 e 8). O fato de a rede com 10 nodos apresentar uma perda de pacotes menor que a rede composta por 5 nodos pode ter sido influenciada pelos arquivos de movimentação e de tráfego gerados para a simulação.

A maior taxa de pacotes perdidos para pacotes de 64 bytes aconteceu na rede composta por 15 nodos utilizando-se o tipo de fila RED. Já para pacotes de 256 bytes, a maior taxa de pacotes perdidos ocorreu na rede composta por 20 nodos utilizando-se o tipo de fila DRR.



	5	10	15	20
FIFO	6,24	2,72	5,85	4,62
RED	5,29	2,3	7,13	4,46
DRR	5,29	2,53	4,79	4,03

Gráfico 7 – Perda de Pacotes x Tipo de Fila (pacotes de 64 bytes)

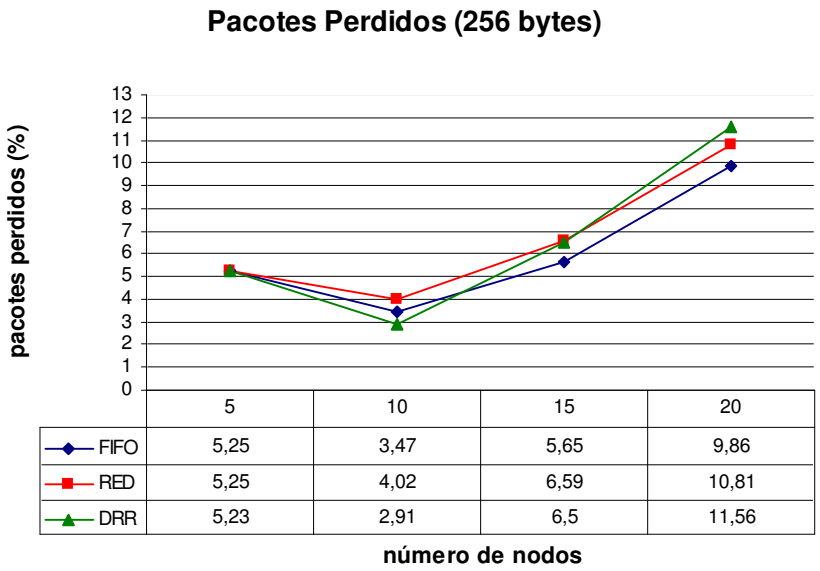


Gráfico 8 – Perda de Pacotes x Tipo de Fila (pacotes de 256 bytes)

5.5.2 Análise dos Protocolos de Roteamento

Em redes A*d Hoc* os protocolos de roteamento desempenham um papel fundamental que influencia diretamente o desempenho de toda a rede, pois estas redes estão sujeitas à perda de comunicação devido à mobilidade dos nodos ou por interferências.

Conforme apresentado na seção 3.5, não existe um consenso sobre qual é o melhor protocolo de roteamento, por isso optou-se por realizar simulações com protocolos tanto pró-ativos (*table-driven*) como reativos (*on-demand*). Como parte dos objetivos deste trabalho era verificar a influência nas métricas de QoS causada pela adoção de protocolos de roteamento seguro, outros protocolos com tais características também foram analisados. A tabela 2 apresenta os protocolos que foram simulados.

	Pró-ativo	**Reativo**
Sem segurança	DSDV	DSR
Com segurança	SEAD	*Ariadne*

Tabela 2 – Protocolos simulados

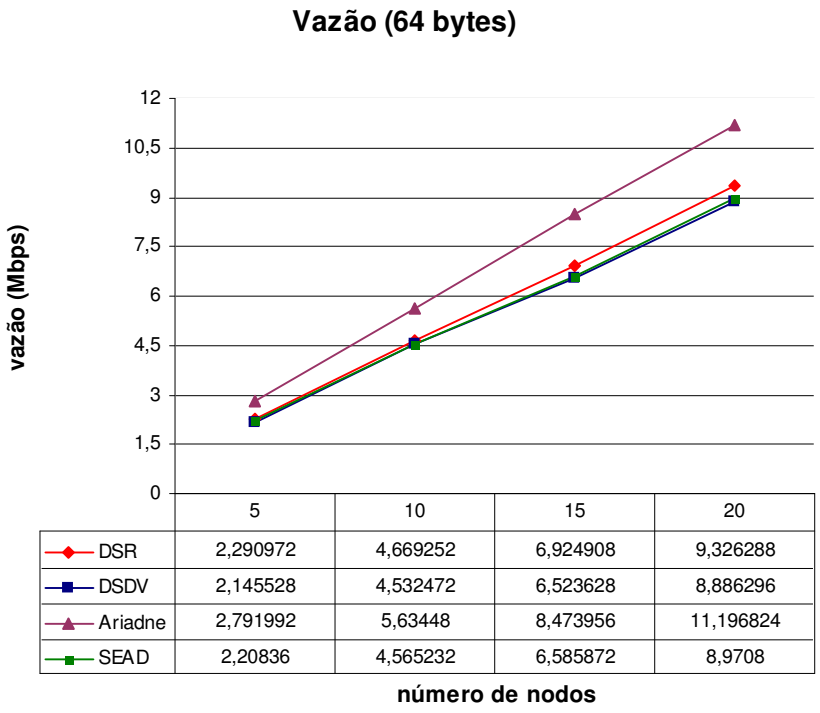
Como discutido na seção 3.5.4, o protocolo SEAD é baseado no DSDV e o *Ariadne* no DSR. O código fonte destes protocolos (SEAD e *Ariadne*) é originário do

Projeto Monarch [MON04], desenvolvido na CMU (*Carnegie Mellon University*) e foi disponibilizada pelos autores de [YOV04]. Para utilizar a implementação destes protocolos, foi necessário compilá-los juntamente com o código principal do NS.

Nas simulações destes protocolos utilizou-se o tipo de fila FIFO, com um número máximo de pacotes igual a 10. Foi escolhido este tipo fila por ter apresentado um desempenho intermediário em relação às outras filas analisadas e por ser a fila mais popular e simples entre os vários tipos existentes.

5.5.2.1 Vazão

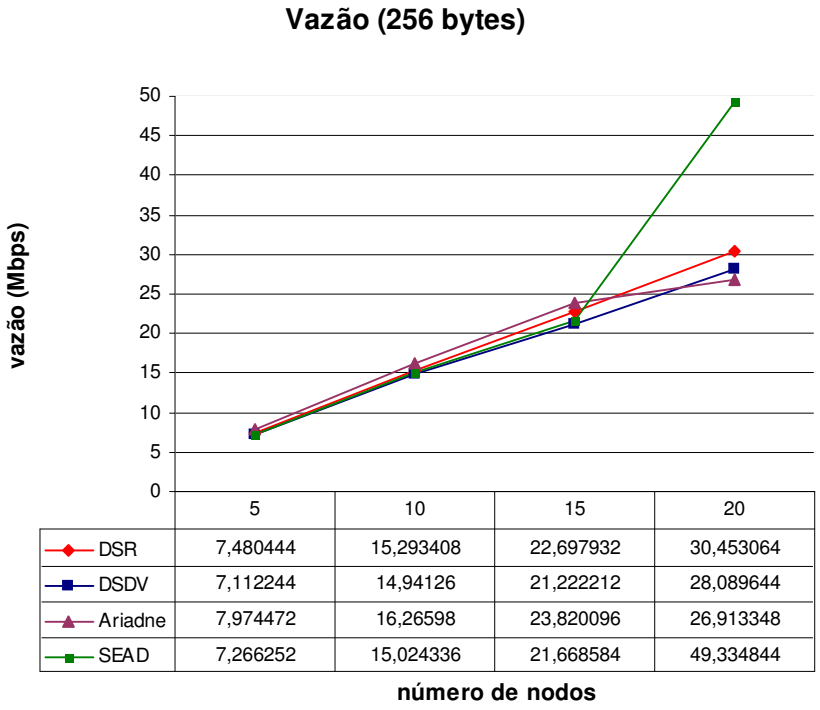
Quanto à vazão podemos observar que os protocolos DSR, DSDV e SEAD apresentaram um comportamento semelhante para os pacotes de 64 bytes, com uma pequena vantagem para o protocolo DSR. Já o protocolo *Ariadne* teve uma vazão maior do que os outros protocolos. O gráfico 9 apresenta os valores de vazão para cada um dos protocolos de roteamento analisados, para pacotes de 64 bytes.



	5	10	15	20
DSR	2,290972	4,669252	6,924908	9,326288
DSDV	2,145528	4,532472	6,523628	8,886296
Ariadne	2,791992	5,63448	8,473956	11,196824
SEAD	2,20836	4,565232	6,585872	8,9708

Gráfico 9 – Vazão x Protocolo de Roteamento (pacotes de 64 bytes)

No experimento com pacotes de 256 bytes tivemos um desempenho semelhante entre todos os protocolos quando a rede foi composta por 5, 10 e 15 nodos. No entanto, quando aumentamos o tamanho da rede para 20 nós, o protocolo SEAD apresentou uma melhor performance em relação a esta métrica. O gráfico 10 apresenta os valores de vazão para cada um dos protocolos de roteamento analisados, para pacotes de 256 bytes.



	5	10	15	20
DSR	7,480444	15,293408	22,697932	30,453064
DSDV	7,112244	14,94126	21,222212	28,089644
Ariadne	7,974472	16,26598	23,820096	26,913348
SEAD	7,266252	15,024336	21,668584	49,334844

Gráfico 10 – Vazão x Protocolo de Roteamento (pacotes de 256 bytes)

Em média todos os protocolos apresentaram um crescimento linear ao aumentarmos o número de nós presentes na rede.

5.5.2.2 Latência

A análise desta métrica é importante, pois quanto menor for a latência, melhor o tempo de resposta da rede. Caso a latência seja muito grande torna-se difícil o uso de interatividade na rede. Este é um fator que deve ser levando em consideração principalmente nas transmissões de áudio e vídeo.

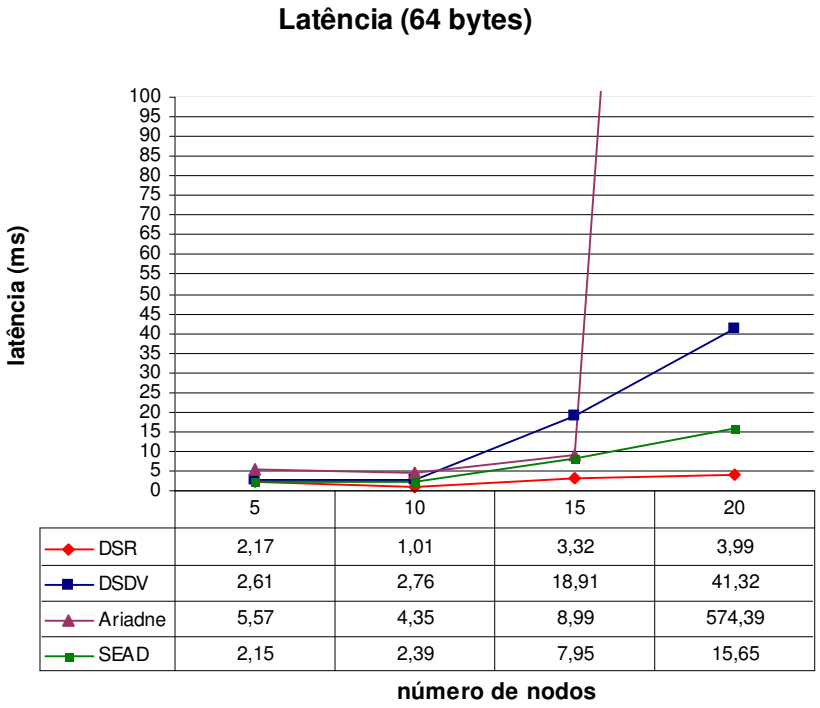
Observa-se que o protocolo DSR é o que possui a melhor performance tanto para pacotes de 64 bytes como para pacotes de 256 bytes. Em segundo lugar aparece o

protocolo SEAD, no entanto quando aumentamos a taxa de tráfego (20 nodos e pacotes de 256 bytes) a performance deste protocolo diminui drasticamente. Esta queda de desempenho também acontece com o protocolo *Ariadne*, sendo que este comportamento também foi observado em outros experimentos realizados em [YOV04] .

O principal fator que faz com que a performance do protocolo *Ariadne* seja inferior é o significativo aumento do *overhead* de roteamento quando aumentamos a taxa de tráfego, congestionando a rede. Já no protocolo SEAD o *overhead* permanece constante, pois mensagens de roteamento são enviadas periodicamente e não depende da taxa de tráfego.

O protocolo base do SEAD possui uma pequena diferença em alguns pontos em relação ao protocolo DSDV original [YIH02b], como por exemplo na forma de envio das mensagens de atualização de rotas, , isto pode ser um fator que faz com que em alguns casos o seu desempenho seja superior ao DSDV. Em outras simulações [YOV04] este comportamento não se verifica.

Os gráficos 11 e 12 referem-se aos valores médios de latência para cada um dos protocolos de roteamento analisados.



	5	10	15	20
DSR	2,17	1,01	3,32	3,99
DSDV	2,61	2,76	18,91	41,32
Ariadne	5,57	4,35	8,99	574,39
SEAD	2,15	2,39	7,95	15,65

Gráfico 11 – Latência x Protocolo de Roteamento (pacotes de 64 bytes)

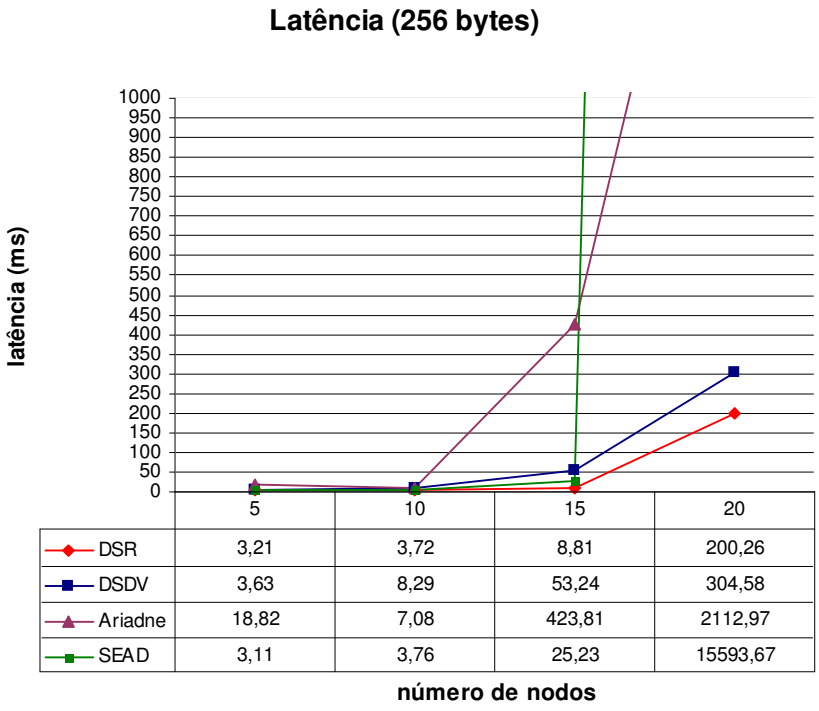


Gráfico 12 – Latência x Protocolo de Roteamento (pacotes de 256 bytes)

5.5.2.3 *Jitter*

O *jitter* é calculado como sendo a variação do atraso (latência), sendo assim, a sua análise nos permite observar se uma rede apresenta um comportamento constante ou se possui muitas variações.

O desempenho dos protocolos em relação a esta métrica foi semelhante ao apresentado em relação à latência. O protocolo DSR foi o que apresentou o melhor desempenho, seguido pelo SEAD. Novamente o protocolo *Ariadne* e, com menos intensidade, o protocolo SEAD apresentam problemas com o aumento do número de nodos presentes na rede e com o aumento do tamanho dos pacotes.

O protocolo DSDV, por sua vez, apresentou um desempenho intermediário e com o aumento do tamanho da rede e dos pacotes sua performance melhorou em relação aos outros protocolos.

As explicações para tais comportamentos são as mesmas que explicam o desempenho de cada protocolo em relação à latência. Os gráficos 13 e 14 apresentam os valore médios de *jitter* para cada um dos protocolos de roteamento analisados.

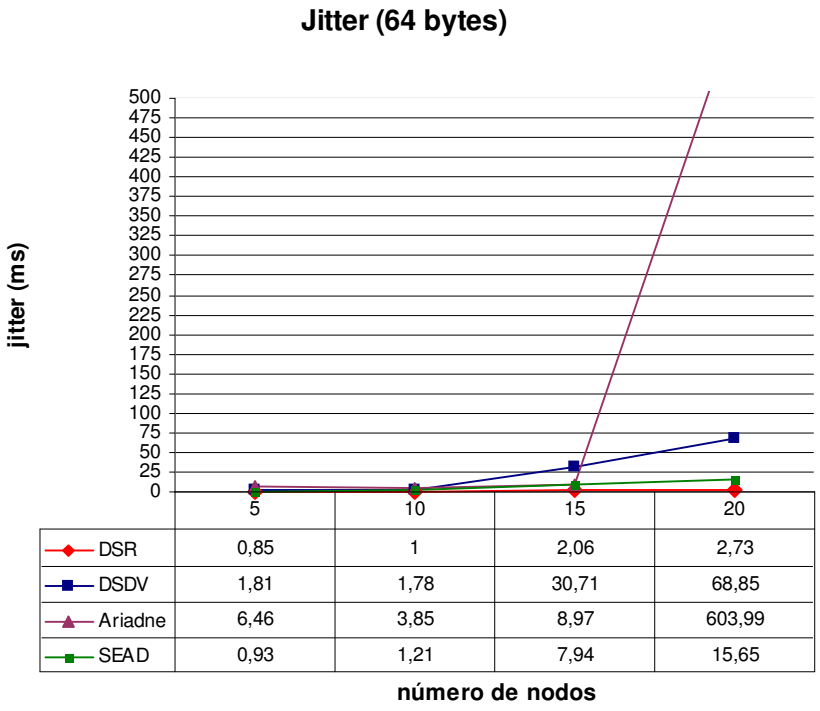


Gráfico 13 – *Jitter* x Protocolo de Roteamento (pacotes de 64 bytes)

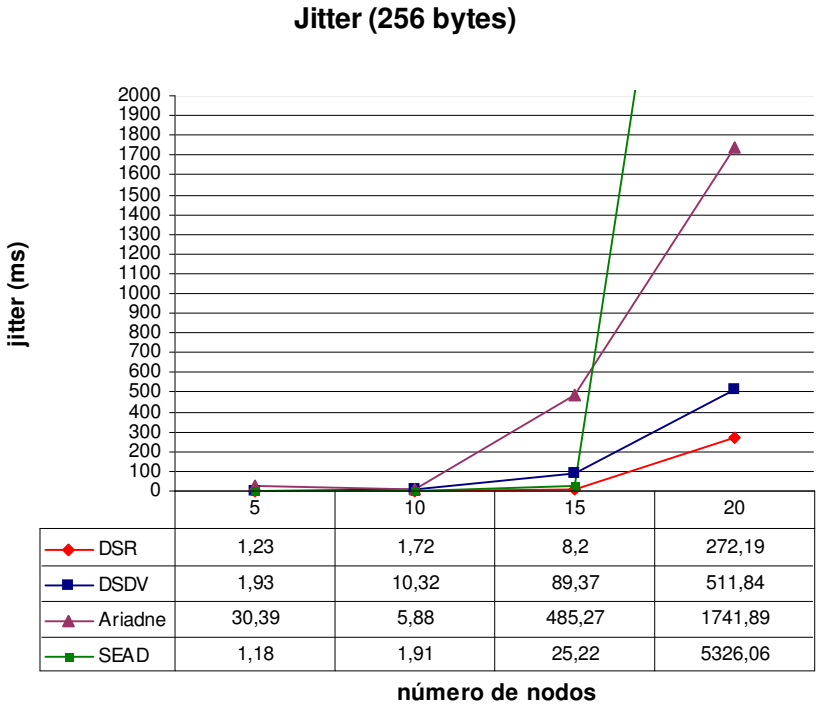


Gráfico 14 – *Jitter* x Protocolo de Roteamento (pacotes de 256 bytes)

5.5.2.4 Perda de Pacotes

Através da análise desta métrica podemos verificar se as filas, localizadas em cada nó da rede, estiveram sobrecarregadas. Outro fator que pode determinar a perda de pacotes é a ineficiência no roteamento dos pacotes, estando mais ligado ao protocolo utilizado no roteamento das mensagens.

O protocolo DSR foi o que apresentou o melhor desempenho, seguido pelo *Ariadne*, DSDV e SEAD, respectivamente. No entanto, para pacotes de 64 bytes, quando aumentamos o tamanho da rede para 20 nodos, o protocolo *Ariadne* mostrou uma performance inferior as DSDV. Para pacotes de 256 bytes, este fator é observado ainda com a rede num tamanho de 15 nodos e quando aumentamos para 20 nodos este protocolo foi o que apresentou o pior desempenho.

Os protocolos reativos (DSR e *Ariadne*), quando simulados com a rede contendo um menor número de nodos (5 e 10), apresentaram uma alta performance em relação a esta métrica, pois conseguem estabelecer as rotas com facilidade. Já os protocolos pró-ativos (DSDV e SEAD) mostraram um desempenho inferior, principalmente o SEAD. Os gráficos 15 e 16 apresentam os percentuais de perda de pacotes para cada um dos protocolos de roteamento analisados.

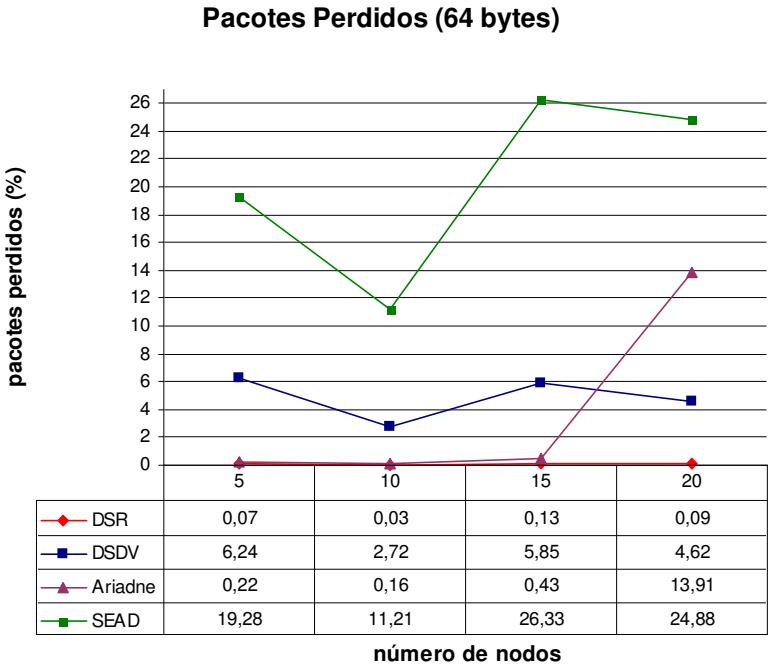


Gráfico 15 – Pacotes Perdidos x Protocolo de Roteamento (pacotes de 64 bytes)

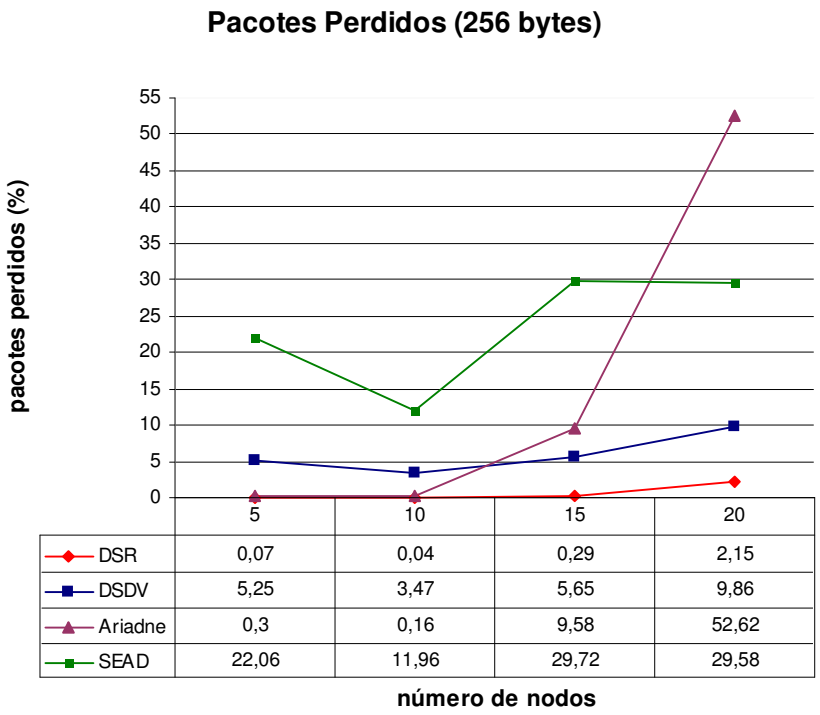


Gráfico 16 – Pacotes Perdidos x Protocolo de Roteamento (pacotes de 256 bytes)

5.6 Conclusões

Tanto para pacotes de 64 bytes quanto para pacotes de 256 bytes, a fila do tipo RED apresentou os menores valores médios para a métrica latência e para a métrica *jitter*. Isto pode ser explicado pelo fato desta fila realizar uma constante monitoração da carga na rede. Quando um aumento do congestionamento é percebido os pacotes começam a ser descartados estocasticamente e pacotes de sinalização são enviados aos dispositivos geradores do tráfego, para que estes diminuam o fluxo, diminuindo o congestionamento. Portanto, com a utilização deste tipo de fila a rede apresentou um melhor tempo de resposta e uma boa uniformidade no envio dos dados, colaborando para uma comunicação de melhor qualidade entre os nodos.

Sendo assim, a política de fila RED foi a que apresentou o melhor desempenho, apesar de ter descartado o maior número de pacotes em algumas situações. A fila FIFO teve uma performance intermediária e a que obteve o pior desempenho nas métricas analisadas foi a fila DRR.

Dos protocolos analisados o que teve um melhor desempenho foi o DSR. No entanto, vale lembrar que os protocolos *Ariadne* e SEAD apresentam mecanismos de

segurança, o que faz com que, em alguns casos, os seus desempenhos diminuam. Existem aplicações que exigem segurança nas comunicações e nestes casos é aconselhável a utilização de protocolos atentos às questões de segurança.

A diferença observada na performance entre os protocolos reativos (DSR e *Ariadne*) foi maior do que a observada entre os protocolos pró-ativos (DSDV e SEAD). Desta forma, podemos afirmar que nestas simulações o impacto causado pela adição de mecanismos de segurança foi maior no protocolo DSR (DSR → *Ariadne*) do que no protocolo DSDV (DSDV → SEAD).

6. CONCLUSÕES

As redes sem fio *Ad Hoc* estão sendo cada vez mais usadas nos mais variados ambientes, principalmente em locais onde não exista uma infra-estrutura de rede fixa ou o custo para sua construção é muito alto.

Devido a algumas características indesejáveis, como baixa taxa de vazão e problemas de segurança, este tipo de rede não era muito utilizada. Porém, através de muitas pesquisas e estudos, estes fatores, que limitavam o uso, começaram a ser sanados ou minimizados e a tendência é que a tecnologia sem fio seja cada vez seja mais aceita e utilizada.

Com a evolução das comunicações e o crescimento na quantidade e na variedade de aplicações, surgiram aquelas que necessitam de garantias em relação a comunicação. Desta forma, para estas aplicações, torna-se necessário o uso de SLAs, que quando estabelecido de uma forma correta são ferramentas poderosas de gerência.

Sendo assim, este trabalho teve como objetivo obter informações sobre o desempenho de redes sem fio *Ad Hoc* e analisá-las para poder moldar as métricas de QoS, as quais são usadas no estabelecimento dos SLAs.

Após a análise dos resultados obtidos com as simulações, a primeira conclusão que se chega é que, para estas simulações, a política de fila que apresentou os melhores resultados foi a RED e o protocolo com a melhor performance foi o DSR. No entanto, vários fatores devem ser analisados na escolha das configurações para uma rede sem fio *Ad Hoc*, como por exemplo a necessidade de segurança nas comunicações, bem como a não negação de serviço ou disponibilidade dos recursos.

Como já era esperado, ao adicionar mecanismos de segurança nos protocolos, os seus desempenhos tendem a degradar. No entanto, estes protocolos são importantes por exercerem funções de segurança no roteamento dos pacotes. Um ponto que merece destaque é o comportamento destes protocolos com o aumento do tráfego na rede (aumento do tamanho dos pacotes e do número de nós pertencentes à rede), pois quando isto acontece o desempenho dos mesmos diminui drasticamente. Esta queda no desempenho é mais acentuada no protocolo *Ariadne*, pois por se tratar de um protocolo reativo, com o aumento da taxa de tráfego na rede ocorre um significativo aumento no *overhead* de roteamento, congestionando a rede.

Em relação ao estabelecimento de um SLA, deverá ficar claro, para o cliente e principalmente para o provedor de serviços, que a utilização de protocolos de roteamento seguro implicará em valores mais altos para as métricas de latência e *jitter*. Como conseqüência disto, a rede apresentará um aumento no tempo de resposta e uma diminuição da uniformidade no envio de pacotes, prejudicando o desempenho da comunicação entre os nodos.

A continuidade deste trabalho poderia ser realizada através da avaliação de outros protocolos de roteamento (tanto seguros como não seguros). Além disso, poderia ser analisado o impacto causado no valor das métricas de QoS ao se utilizar diferentes padrões de mobilidade dos nós pertencentes à rede. Desta forma, poderia ser observado qual protocolo de roteamento seria mais eficiente em relação a este fator, pois com o aumento da mobilidade a tendência é que os valores das métricas de QoS degradem.

Outro trabalho interessante a ser desenvolvido seria analisar o desempenho dos protocolos de roteamento seguro em uma rede *Ad Hoc* quando um dos nodos fosse malicioso. O esperado é que a performance destes protocolos seja menor do que as apresentadas neste trabalho.

Além das quatro métricas analisadas neste trabalho, outros fatores de QoS poderiam ser abordados, como: *loss profile* e probabilidade de comunicação transparente.

REFERÊNCIAS

[AGR04] AGRAWAL, Dharma P.; CORDEIRO, Carlos M. **Mobile Ad hoc Networking**, Capitulo 3, OBR Research Center for Distributed and Mobile Computing, ECECS University of Cincinnati, Cincinnati, OH 45221-0030 – USA.

[ASP04] Página da **ASPnews**. Disponivel em: <http://www.aspnews.com/>. Acesso: 05 junho 2004.

[BRO98] BROCH, Josh.; MALTZ, David. A.; JOHNSON, David B.; YIH-CHUN, Hu, JETCHEVA, Jorgeta. G., **A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols**, in proceedings of MOBICOM 1998.

[COR96] CORSON, M.S.; MACKER, J.; BATSELL, S. **Architectural Considerations for Mobile Mesh Networking** In Proceedings of the IEEE MILCOM, October 1996.

[COR99] CORSON, S. RFC 2501: **Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations**, Janeiro 1999.

[FAG03] FAGUNDES, Eduardo M. **Wireless LAN**, Universidade Mackenzie. Disponível em: < http://www.efagundes.com/Artigos/Wireless_LAN.htm >. Acesso em: 10 outubro 2004.

[FAG04] FAGUNDES, Eduardo M. **Gestão de Contratos Com SLA**, Universidade Mackenzie. Disponível em: <http://www.efagundes.com/Artigos/ Gestao_de_Contratos_com_SLA.htm>. Acesso em: 9 setembro 2004.

[GTA04] **Redes *Ad Hoc*.** Disponível em:< http://www.gta.ufrj.br/~apaulo/seminarios/ adhoc-e-bluetooth/adhoc.html>. Acesso em: 10 outubro 2004.

[HEG03] HE,Guoyou, **Destination-Sequenced Distance Vector (DSDV) Protocol**, Networking Laboratory, Helsinki University of Technology. Disponível em: < http://citeseer.ist.psu.edu/531710.html >. Acesso em: 20 outubro 2004.

[IEE99a] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Wireless LAN Medium Access Control (MAC) and Physical (PHY) Specifications**. IEEE Standard 802.11, 1999.

[IEE99b] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Wireless LAN Medium Access Control (MAC) and Physical (PHY) Specifications**. IEEE Standard 802.11a, 1999.

[IEE99c] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Wireless LAN Medium Access Control (MAC) and Physical (PHY) Specifications.** IEEE Standard 802.11b, 1999.

[ISO04] **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**. Página da World Wide Web. Dispinivel em:<http://www.iso.org/>. Acesso em: 5 junho de 2004.

[JOH01] JOHNSON, David B.; MALTZ, D. A.; BROCH, J. **DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks**. In C. Perkins, editor, Ad Hoc Networking, chapter 5, pages 139--172. Addison-Wesley, 2001.

[JOH96] JOHNSON, David B.; MALTZ, D. A., **Dynamic Source Routing in Ad Hoc Wireless Networks**. In Imielinski and Korth, editors, Mobile Computing, volume 353. Kluwer Academic Publishers, 1996.

[JOH99] JOHANSSON, Per; LARSSON, Tony; HEDMAN, Nicklas; MIELCZAREK, Bartosz; DEGERMARK, Mikael, **Scenario-based Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks**, in proceedings of MOBICOM 1999.

[JUB87] JUBIN, J.; TRUONG, T. **Distributed Algorithm for Efficient and Interference-free Broadcasting in Radio Networks** in Proceedings of INFOCOM, January 1987, 21-32.

[JUN04] JUNIOR, Aurélio A.; DUARTE, Otto C. M. B. **Segurança no Roteamneto em Redes Móveis Ad Hoc.** Grupo de Teleinformática e Automação, Universidade Federal do Rio de Janeiro.

[LIM02] LIMA, A. **Proposta e Validação de um Mecanismo para Garantir QoS em Redes Sem Fio de Topologia Ad Hoc**. Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciências da Computação. Universidade Federal de Santa Catarina – UFSC, 2002.

[MAL04] MALEK, Jaroslaw. **Trace Graph.** Disponível em : <http://www.geocities. com/tracegraph/>. Acesso em 20 setembro de 2004.

[MON04] MONARCH PROJECT. **Wireless and Mobility Extensions to ns-2**. Disponível em: <http://www.monarch.cs.cmu.edu/cmu-ns.html>. Acesso em 7 Outubro de 2004.

[PER94] PERKINS, Charles E.; BHAGWAT, Pravin, **Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers**. In Proceedings of the ACM SIGCOMM, pages 234--244, August 1994.

[PER03] PEREIRA, M. C. **Análise de Desempenho em Redes Wireless Ad Hoc e Estabelecimento de um Acordo de Nível de Serviço Pró-Ativo**. 2003. 145f. Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciências da Computação. Universidade Federal de Santa Catarina – UFSC, 2003.

[OUE02] OUELLET, E; PADJEN, R.; PFUND, A. **Building a CISCO Wireless Lan**, Syngress Publishing, 2002. ISBN 1-928994-58-X.

[ROY99] ROYER,E. M.; TOH, C-k. "**A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks**", IEEE Personal Communications, Apr. 1999, Disponível em: < http://citeseer.ist.psu.edu/royer99review.html>. Acesso em: 22 outubro 2004.

[RSS04] **Aspectos Técnicos dos Rádios Spread Spectrum Aplicados aos Sistemas de Transmissão de Dados**. Disponível em: <http://www.teleco.com.br/ tutoriais/tutorialss/pagina_2.asp>. Acesso em: 10 maio 2004.

[RUB04] RUBINSTEIN, Marcelo G.; REZENDE, José F. **Qualidade de Serviço em Redes 802.11**, Disponível em:<http://www.gta.ufrj.br/ftp/gta/TechReports/ RuRe02.pdf>. Acesso em: 8 abril 2004.

[SAN02] SANTOS, Héctor J. De los. **RF MEMS Circuit Design for Wireless Communications**, Artech House, 2002. ISBN 1-58053-329-9

[SIL98] SILVA, Adailton J. S. **As Tecnologias de Redes Wireless**. RNP News Generation, Vol 2, No. 5, Maio, 1998. Disponível em: <http://www.rnp.br/newsgen/9805/wireless.html>. Acesso em: 20 julho 2004.

[SIL04a] SILVA, Madalena P. **Estabelecimento de um SLA através de Desempenho de Redes Sem Fio**. Sociedade Lageana de Educação, Departamento de Ciências Exatas e Tecnológica.

[SIL04b] SILVA, Madalena P. **Análise de Desempenho e Diferenciação na Camada MAC do Padrão IEEE 802.11**. Dissertação de mestrado, Janeiro, 2004.

[TEC04] **TechRepublic: What to Include in your Service Level Agreement with your Wireless ISP.** Disponivel em:<http://techrepublic.com.com/5100-6296-1037286.html>. Acesso em: 15 maio 2004.

[TIO01] TIONG, Eng . **IEEE 802.11b Wireless LAN: Security Risks**,version 1.0, September 20, 2001. Disponível em: <www.sans.org/rr/whitepapers/ wireless/151.php>. Acesso em: 21 outubro 2004.

[UNB04] **O Que é o NS?** Disponível em:<http://www.redes.unb.br/Projetos_grad/ns2/ oquee.html>. Acesso em: 15 maio 2004.

[WHA04] **What the Heck is this 802.11 && 802.11b?**. Disponível em: <http://ntrg.cs.tcd.ie/undergrad/4ba2.01/group1/802.htm>. Acesso em: 5 junho 2004.

[WIR04] **Wireless Networking, Vicomsoft**, White Papers. Disponível em: http://www.vicomsoft.com/knowledge/reference/wireless1.html>. Acesso em: 10 maio 2004.

[YI801] YI, Seung; NALDURG, Prasad; KRAVETS, Robin. **A Security-Aware Routing Protocol for Wireless Ad Hoc Networks**, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

[YIH02a] YIH-CHUN, Hu; PERRIG, Adrian; JOHNSON, David B., **ARIADNE: A secure On-Demand Routing Protocol for Ad Hoc Networks**, in proceedings of MOBICOM 2002.

[YIH02b] YIH-CHUN, Hu; JOHNSON, David B.; PERRIG, Adrian, **SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks**, in the 4th IEEE Workshop on Mobile Computing Systems and Applications, 2002.

[YOV04] YOVANOF, Gregory S.; ERIKCI, Kerem. **Performance Evaluation of Security-Aware Routing Protocols for Clustered Mobile Ad Hoc Networks**, International Workshop on Wireless Ad-Hoc Networks, 2004.

ANEXO A – SCRIPTS TCL

```
# Copyright (c) 2004 Federal University of Santa Catarina, Brazil.
#
#              Darlan Vivian              Eduardo A. P. Alchieri
#     darlan@inf.ufsc.br                          edu@inf.ufsc.br
#
# All rights reserved.
#
# ======================================================================
# Define options
# ======================================================================
set val(chan)    Channel/WirelessChannel ;# channel type
set val(prop)    Propagation/TwoRayGround ;# radio-propagation model
set val(netif)   Phy/WirelessPhy  ;# network interface type
set val(mac)     Mac/802_11       ;# MAC type
#set val(ifq)    Queue/RED        ;# Interface queue type
#set val(ifq)    Queue/DRR        ;# Interface queue type
set val(ifq)     CMUPriQueue      ;# Interface queue type
set val(ll)      LL               ;# Link layer type
set val(ant)     Antenna/OmniAntenna  ;# Antenna type
set val(x)       300.00  ;# X dimension of the topography
set val(y)       300.00  ;# Y dimension of the topography
set val(ifqlen)  10               ;# max packet in ifq
set val(seed)    0.0              ;# seed
#set val(adhocRouting) DSR        ;# ad-hoc routing protocol
set val(adhocRouting)  DSDV       ;# ad-hoc routing protocol
set val(nn)            5    ;# how many nodes are simulated
#set val(nn)           10 ;# how many nodes are simulated
#set val(nn)           15 ;# how many nodes are simulated
#set val(nn)           20 ;# how many nodes are simulated
#set val(cp)           "cbr/cbr-5-64"          ;# traffic model
set val(cp)           "cbr/cbr-5-256"         ;# traffic model
#set val(cp)           "cbr/cbr-10-64"         ;# traffic model
#set val(cp)           "cbr/cbr-10-256"        ;# traffic model
#set val(cp)           "cbr/cbr-15-64"         ;# traffic model
#set val(cp)           "cbr/cbr-15-256"        ;# traffic model
#set val(cp)           "cbr/cbr-20-64"         ;# traffic model
#set val(cp)           "cbr/cbr-20-256"        ;# traffic model
set val(sc)           "scen/scen-5"           ;# mobility model
#set val(sc)           "scen/scen-10"          ;# mobility model
#set val(sc)           "scen/scen-15"          ;# mobility model
# set val(sc)          "scen/scen-20"          # mobility model
set val(stop)         400.0                   ;# simulation time (sec)
set name_queue        "FIFO"      ;           # queue name
#set name_queue       "RED"                   ;# queue name
#set name_queue       "DRR"                   ;# queue name
#set name_tracefile   "TCC-5-64-$name_queue-$val(adhocRouting)"
set name_tracefile    "TCC-5-256-$name_queue-$val(adhocRouting)"
#set name_tracefile   "TCC-10-64-$name_queue-$val(adhocRouting)"
#set name_tracefile   "TCC-10-256-$name_queue-$val(adhocRouting)"
#set name_tracefile   "TCC-15-64-$name_queue-$val(adhocRouting)"
#set name_tracefile   "TCC-15-256-$name_queue-$val(adhocRouting)"
#set name_tracefile   "TCC-20-64-$name_queue-$val(adhocRouting)"
#set name_tracefile   "TCC-20-256-$name_queue-$val(adhocRouting)"
# ======================================================================
# Main Program
# ======================================================================
# To Queue DRR: indicates in bytes how much each flow can send its turn
#$val(ifq) set            quantum_ 256
#set transimission rate to 2Mbps
$val(mac) set dataRate_ 2000000
#
# Initialize Global Variables
#
# create simulator instance
set ns_                  [new Simulator]
# setup topography object
set topo     [new Topography]
# Create channel
set chan_1_ [new $val(chan)]
# create trace object for ns and nam
set tracefd [open $name_tracefile.tr w]
```

```
#set namtrace    [open $name_tracefile.nam w]
$ns_ trace-all $tracefd
#$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
# define topology
$topo load_flatgrid $val(x) $val(y)
#
# Create God
#
set god_ [create-god $val(nn)]
#
# define how node should be created
#
#global node setting
$ns_ node-config  -adhocRouting $val(adhocRouting) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channel $chan_1_ \
                                -topoInstance $topo \
                                        -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                                        -movementTrace ON

#
#  Create the specified number of nodes [$val(nn)] and "attach" them
#  to the channel.
for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0                      ;# disable random motion
}
#
# Define node movement model
#
puts "Loading connection pattern..."
source $val(cp)
#
# Define traffic model
#
puts "Loading scenario file..."
source $val(sc)



# Define node initial position in nam
for {set i 0} {$i < $val(nn)} {incr i} {
   # 20 defines the node size in nam, must adjust it according to your scenario
   # The function must be called after mobility model is defined
   $ns_ initial_node_pos $node_($i) 20
}
#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at $val(stop).0 "$node_($i) reset";
}
$ns_ at  $val(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
puts $tracefd "M 0.0 nn $val(nn) x $val(x) y $val(y) rp $val(adhocRouting)"
puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"
puts "Starting Simulation..."
$ns_ run
```

#Padrões de Comunicação

```
# Arquivo que contém o padrão de comunicação para 5 nós e pacotes de 64 bytes (cbr-5-64)

#
# nodes: 5, max conn: 10, send rate: 0.25, seed: 9
#
#
```

```
# 3 connecting to 4 at time 154.69328709630915
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 64
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 154.69328709630915 "$cbr_(0) start"
#
# 3 connecting to 2 at time 34.627089991526255
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 64
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 34.627089991526255 "$cbr_(1) start"
#
# 4 connecting to 0 at time 62.686458613111853
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 64
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 62.686458613111853 "$cbr_(2) start"
#
# 2 connecting to 1 at time 23.75412509811767
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 64
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 23.75412509811767 "$cbr_(3) start"
#
# 1 connecting to 3 at time 25.75412509811767
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 64
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 25.75412509811767 "$cbr_(4) start"
#
# 2 connecting to 0 at time 03.75412509811767
```

```
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 64
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 03.75412509811767 "$cbr_(5) start"
#
# 0 connecting to 1 at time 33.75412509811767
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 64
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 33.75412509811767 "$cbr_(6) start"
#
# 0 connecting to 2 at time 93.75412509811767
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 64
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 93.75412509811767 "$cbr_(7) start"
#
# 1 connecting to 4 at time 77.75412509811767
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 64
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 77.75412509811767 "$cbr_(8) start"
#
# 0 connecting to 4 at time 102.75412509811767
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 64
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 102.75412509811767 "$cbr_(9) start"
#
#Total sources/connections: 05/10
#
```

```
# Arquivo que contém o padrão de comunicação para 5 nós e pacotes de 256 bytes (cbr-5-256)

#
# nodes: 5, max conn: 10, send rate: 0.25, seed: 9
#
#
# 3 connecting to 4 at time 154.69328709630915
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 154.69328709630915 "$cbr_(0) start"
#
# 3 connecting to 2 at time 34.627089991526255
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 256
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 34.627089991526255 "$cbr_(1) start"
#
# 4 connecting to 0 at time 62.686458613111853
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 256
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 62.686458613111853 "$cbr_(2) start"
#
# 2 connecting to 1 at time 23.75412509811767
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 256
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 23.75412509811767 "$cbr_(3) start"
#
# 1 connecting to 3 at time 25.75412509811767
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 256
```

```
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 25.75412509811767 "$cbr_(4) start"
#
# 2 connecting to 0 at time 03.75412509811767
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 256
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 03.75412509811767 "$cbr_(5) start"
#
# 0 connecting to 1 at time 33.75412509811767
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 256
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 33.75412509811767 "$cbr_(6) start"
#
# 0 connecting to 2 at time 93.75412509811767
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 256
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 93.75412509811767 "$cbr_(7) start"
#
# 1 connecting to 4 at time 77.75412509811767
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 256
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 77.75412509811767 "$cbr_(8) start"
#
# 0 connecting to 4 at time 102.75412509811767
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 256
$cbr_(9) set interval_ 0.125
```

```
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 102.75412509811767 "$cbr_(9) start"
#
#Total sources/connections: 05/10
#

# Arquivo que contém o padrão de comunicação para 10 nós e pacotes de 64 bytes (cbr-10-64)

#
# nodes: 10, max conn: 20, send rate: 0.25, seed: 9
#
#
# 3 connecting to 4 at time 24.69328709630915
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 64
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 24.69328709630915 "$cbr_(0) start"
#
# 3 connecting to 5 at time 34.627089991526255
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 64
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 34.627089991526255 "$cbr_(1) start"
#
# 4 connecting to 5 at time 62.686458613111853
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 64
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 62.686458613111853 "$cbr_(2) start"
#
# 4 connecting to 6 at time 20.051991092065347
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 64
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 20.051991092065347 "$cbr_(3) start"
#
# 5 connecting to 6 at time 140.70807377840768
```

```
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 64
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 140.70807377840768 "$cbr_(4) start"
#
# 6 connecting to 7 at time 85.909864728297052
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 64
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 85.909864728297052 "$cbr_(5) start"
#
# 8 connecting to 9 at time 4.3617707511232098
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 64
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 4.3617707511232098 "$cbr_(6) start"
#
# 8 connecting to 9 at time 132.75412509811767
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 64
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 132.75412509811767 "$cbr_(7) start"
#
# 9 connecting to 5 at time 103.18451128117019
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 64
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 103.18451128117019 "$cbr_(8) start"
#
# 9 connecting to 0 at time 152.5049444346246
#
```

```
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 64
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 152.5049444346246 "$cbr_(9) start"
#
# 0 connecting to 8 at time 14.69328709630915
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 64
$cbr_(10) set interval_ 0.125
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 14.69328709630915 "$cbr_(10) start"
#
# 1 connecting to 7 at time 44.69328709630915
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 64
$cbr_(11) set interval_ 0.125
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 44.69328709630915 "$cbr_(11) start"
#
# 2 connecting to 1 at time 23.75412509811767                   /1
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 64
$cbr_(12) set interval_ 0.125
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 23.75412509811767 "$cbr_(12) start"
#
# 7 connecting to 2 at time 53.75412509811767
#
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(13)
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 64
$cbr_(13) set interval_ 0.125
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 53.75412509811767 "$cbr_(13) start"
#
# 8 connecting to 0 at time 39.75412509811767                   /2
#
set udp_(14) [new Agent/UDP]
```

```
$ns_ attach-agent $node_(8) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 64
$cbr_(14) set interval_ 0.125
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 39.75412509811767 "$cbr_(14) start"
#
# 1 connecting to 3 at time 25.75412509811767
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 64
$cbr_(15) set interval_ 0.125
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 25.75412509811767 "$cbr_(15) start"
#
# 7 connecting to 0 at time 78.75412509811767                    /9
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 64
$cbr_(16) set interval_ 0.125
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 78.75412509811767 "$cbr_(16) start"
#
# 6 connecting to 0 at time 13.75412509811767
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 64
$cbr_(17) set interval_ 0.125
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
$ns_ connect $udp_(17) $null_(17)
$ns_ at 13.75412509811767 "$cbr_(17) start"
#
# 2 connecting to 0 at time 03.75412509811767
#
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 64
$cbr_(18) set interval_ 0.125
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $null_(18)
$ns_ at 03.75412509811767 "$cbr_(18) start"
#
# 6 connecting to 3 at time 33.75412509811767
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(19)
```

```
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 64
$cbr_(19) set interval_ 0.125
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 33.75412509811767 "$cbr_(19) start"
#
#Total sources/connections: 10/20
#



# Arquivo que contém o padrão de comunicação para 10 nós e pacotes de 256 bytes (cbr-10-256)

#
# nodes: 10, max conn: 20, send rate: 0.25, seed: 9
#
#
# 3 connecting to 4 at time 24.69328709630915
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 24.69328709630915 "$cbr_(0) start"
#
# 3 connecting to 5 at time 34.627089991526255
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 256
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 34.627089991526255 "$cbr_(1) start"
#
# 4 connecting to 5 at time 62.686458613111853
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 256
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 62.686458613111853 "$cbr_(2) start"
#
# 4 connecting to 6 at time 20.051991092065347
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 256
$cbr_(3) set interval_ 0.125
```

```
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 20.051991092065347 "$cbr_(3) start"
#
# 5 connecting to 6 at time 140.70807377840768
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 256
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 140.70807377840768 "$cbr_(4) start"
#
# 6 connecting to 7 at time 85.909864728297052
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 256
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 85.909864728297052 "$cbr_(5) start"
#
# 8 connecting to 9 at time 4.3617707511232098
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 256
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 4.3617707511232098 "$cbr_(6) start"
#
# 8 connecting to 9 at time 132.75412509811767
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 256
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 132.75412509811767 "$cbr_(7) start"
#
# 9 connecting to 5 at time 103.18451128117019
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 256
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
```

```
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 103.18451128117019 "$cbr_(8) start"
#
# 9 connecting to 0 at time 152.5049444346246
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 256
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 152.5049444346246 "$cbr_(9) start"
#
# 0 connecting to 8 at time 14.69328709630915
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 256
$cbr_(10) set interval_ 0.125
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 14.69328709630915 "$cbr_(10) start"
#
# 1 connecting to 7 at time 44.69328709630915
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 256
$cbr_(11) set interval_ 0.125
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 44.69328709630915 "$cbr_(11) start"
#
# 2 connecting to 1 at time 23.75412509811767                    /1
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 256
$cbr_(12) set interval_ 0.125
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 23.75412509811767 "$cbr_(12) start"
#
# 7 connecting to 2 at time 53.75412509811767
#
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(13)
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 256
$cbr_(13) set interval_ 0.125
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
```

```
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 53.75412509811767 "$cbr_(13) start"
#
# 8 connecting to 0 at time 39.75412509811767                    /2
#
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 256
$cbr_(14) set interval_ 0.125
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 39.75412509811767 "$cbr_(14) start"
#
# 1 connecting to 3 at time 25.75412509811767
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 256
$cbr_(15) set interval_ 0.125
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 25.75412509811767 "$cbr_(15) start"
#
# 7 connecting to 0 at time 78.75412509811767                    /9
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 256
$cbr_(16) set interval_ 0.125
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 78.75412509811767 "$cbr_(16) start"
#
# 6 connecting to 0 at time 13.75412509811767
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 256
$cbr_(17) set interval_ 0.125
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
$ns_ connect $udp_(17) $null_(17)
$ns_ at 13.75412509811767 "$cbr_(17) start"
#
# 2 connecting to 0 at time 03.75412509811767
#
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 256
$cbr_(18) set interval_ 0.125
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
```

```
$ns_ connect $udp_(18) $null_(18)
$ns_ at 03.75412509811767 "$cbr_(18) start"
#
# 6 connecting to 3 at time 33.75412509811767
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(19)
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 256
$cbr_(19) set interval_ 0.125
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 33.75412509811767 "$cbr_(19) start"
#
#Total sources/connections: 10/20
#


# Arquivo que contém o padrão de comunicação para 15 nós e pacotes de 64 bytes (cbr-15-64)
#
# nodes: 15, max conn: 19, send rate: 0.25, seed: 9
#
#
# 3 connecting to 4 at time 154.69328709630915
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 64
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 154.69328709630915 "$cbr_(0) start"
#
# 3 connecting to 5 at time 34.627089991526255
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 64
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 34.627089991526255 "$cbr_(1) start"
#
# 4 connecting to 5 at time 62.686458613111853
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 64
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 62.686458613111853 "$cbr_(2) start"
#
# 4 connecting to 6 at time 20.051991092065347
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(3)
```

```
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 64
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 20.051991092065347 "$cbr_(3) start"
#
# 5 connecting to 6 at time 140.70807377840768
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 64
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 140.70807377840768 "$cbr_(4) start"
#
# 6 connecting to 7 at time 85.909864728297052
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 64
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 85.909864728297052 "$cbr_(5) start"
#
# 8 connecting to 9 at time 4.3617707511232098
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 64
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 4.3617707511232098 "$cbr_(6) start"
#
# 8 connecting to 9 at time 132.75412509811767
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 64
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 132.75412509811767 "$cbr_(7) start"
#
# 9 connecting to 5 at time 103.18451128117019
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(8)
set null_(8) [new Agent/Null]
```

```
$ns_ attach-agent $node_(5) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 64
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 103.18451128117019 "$cbr_(8) start"
#
# 9 connecting to 0 at time 152.5049444346246
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 64
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 152.5049444346246 "$cbr_(9) start"
#
# 0 connecting to 8 at time 14.69328709630915
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 64
$cbr_(10) set interval_ 0.125
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 14.69328709630915 "$cbr_(10) start"
#
# 1 connecting to 7 at time 44.69328709630915
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 64
$cbr_(11) set interval_ 0.125
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 44.69328709630915 "$cbr_(11) start"
#
# 2 connecting to 1 at time 23.75412509811767                    /1
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 64
$cbr_(12) set interval_ 0.125
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 23.75412509811767 "$cbr_(12) start"
#
# 7 connecting to 2 at time 53.75412509811767
#
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(13)
```

```
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 64
$cbr_(13) set interval_ 0.125
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 53.75412509811767 "$cbr_(13) start"
#
# 8 connecting to 0 at time 33.75412509811767                   /2
#
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 64
$cbr_(14) set interval_ 0.125
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 33.75412509811767 "$cbr_(14) start"
#
# 1 connecting to 3 at time 25.75412509811767
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 64
$cbr_(15) set interval_ 0.125
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 25.75412509811767 "$cbr_(15) start"
#
# 7 connecting to 0 at time 78.75412509811767                   /9
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 64
$cbr_(16) set interval_ 0.125
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 78.75412509811767 "$cbr_(16) start"
#
# 6 connecting to 0 at time 13.75412509811767
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 64
$cbr_(17) set interval_ 0.125
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
$ns_ connect $udp_(17) $null_(17)
$ns_ at 13.75412509811767 "$cbr_(17) start"
#
# 2 connecting to 0 at time 03.75412509811767
#
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
```

```
$cbr_(18) set packetSize_ 64
$cbr_(18) set interval_ 0.125
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $null_(18)
$ns_ at 03.75412509811767 "$cbr_(18) start"
#
# 6 connecting to 3 at time 33.75412509811767
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(19)
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 64
$cbr_(19) set interval_ 0.125
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 33.75412509811767 "$cbr_(19) start"
#
# 3 connecting to 10 at time 43.75412509811767
#
set udp_(20) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(20)
set null_(20) [new Agent/Null]
$ns_ attach-agent $node_(10) $null_(20)
set cbr_(20) [new Application/Traffic/CBR]
$cbr_(20) set packetSize_ 64
$cbr_(20) set interval_ 0.125
$cbr_(20) set random_ 1
$cbr_(20) set maxpkts_ 10000
$cbr_(20) attach-agent $udp_(20)
$ns_ connect $udp_(20) $null_(20)
$ns_ at 43.75412509811767 "$cbr_(20) start"
#
# 10 connecting to 7 at time 80.75412509811767
#
set udp_(21) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(21)
set null_(21) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(21)
set cbr_(21) [new Application/Traffic/CBR]
$cbr_(21) set packetSize_ 64
$cbr_(21) set interval_ 0.125
$cbr_(21) set random_ 1
$cbr_(21) set maxpkts_ 10000
$cbr_(21) attach-agent $udp_(21)
$ns_ connect $udp_(21) $null_(21)
$ns_ at 80.75412509811767 "$cbr_(21) start"
#
# 12 connecting to 14 at time 90.75412509811767
#
set udp_(22) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(22)
set null_(22) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(22)
set cbr_(22) [new Application/Traffic/CBR]
$cbr_(22) set packetSize_ 64
$cbr_(22) set interval_ 0.125
$cbr_(22) set random_ 1
$cbr_(22) set maxpkts_ 10000
$cbr_(22) attach-agent $udp_(22)
$ns_ connect $udp_(22) $null_(22)
$ns_ at 90.75412509811767 "$cbr_(22) start"
#
# 11 connecting to 5 at time 25
#
set udp_(23) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(23)
set null_(23) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(23)
set cbr_(23) [new Application/Traffic/CBR]
$cbr_(23) set packetSize_ 64
```

```
$cbr_(23) set interval_ 0.125
$cbr_(23) set random_ 1
$cbr_(23) set maxpkts_ 10000
$cbr_(23) attach-agent $udp_(23)
$ns_ connect $udp_(23) $null_(23)
$ns_ at 25 "$cbr_(23) start"
#
# 4 connecting to 13 at time 2
#
set udp_(24) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(24)
set null_(24) [new Agent/Null]
$ns_ attach-agent $node_(13) $null_(24)
set cbr_(24) [new Application/Traffic/CBR]
$cbr_(24) set packetSize_ 64
$cbr_(24) set interval_ 0.125
$cbr_(24) set random_ 1
$cbr_(24) set maxpkts_ 10000
$cbr_(24) attach-agent $udp_(24)
$ns_ connect $udp_(24) $null_(24)
$ns_ at 2 "$cbr_(24) start"
#
# 8 connecting to 11 at time 10
#
set udp_(25) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(25)
set null_(25) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(25)
set cbr_(25) [new Application/Traffic/CBR]
$cbr_(25) set packetSize_ 64
$cbr_(25) set interval_ 0.125
$cbr_(25) set random_ 1
$cbr_(25) set maxpkts_ 10000
$cbr_(25) attach-agent $udp_(25)
$ns_ connect $udp_(25) $null_(25)
$ns_ at 10 "$cbr_(25) start"
#
# 0 connecting to 12 at time 150
#
set udp_(26) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(26)
set null_(26) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(26)
set cbr_(26) [new Application/Traffic/CBR]
$cbr_(26) set packetSize_ 64
$cbr_(26) set interval_ 0.125
$cbr_(26) set random_ 1
$cbr_(26) set maxpkts_ 10000
$cbr_(26) attach-agent $udp_(26)
$ns_ connect $udp_(26) $null_(26)
$ns_ at 150 "$cbr_(26) start"
#
# 14 connecting to 9 at time 39
#
set udp_(27) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(27)
set null_(27) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(27)
set cbr_(27) [new Application/Traffic/CBR]
$cbr_(27) set packetSize_ 64
$cbr_(27) set interval_ 0.125
$cbr_(27) set random_ 1
$cbr_(27) set maxpkts_ 10000
$cbr_(27) attach-agent $udp_(27)
$ns_ connect $udp_(27) $null_(27)
$ns_ at 39 "$cbr_(27) start"
#
# 13 connecting to 2 at time 60
#
set udp_(28) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(28)
set null_(28) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(28)
set cbr_(28) [new Application/Traffic/CBR]
$cbr_(28) set packetSize_ 64
$cbr_(28) set interval_ 0.125
```

```
$cbr_(28) set random_ 1
$cbr_(28) set maxpkts_ 10000
$cbr_(28) attach-agent $udp_(28)
$ns_ connect $udp_(28) $null_(28)
$ns_ at 60 "$cbr_(28) start"
#
# 13 connecting to 4 at time 55
#
set udp_(29) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(29)
set null_(29) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(29)
set cbr_(29) [new Application/Traffic/CBR]
$cbr_(29) set packetSize_ 64
$cbr_(29) set interval_ 0.125
$cbr_(29) set random_ 1
$cbr_(29) set maxpkts_ 10000
$cbr_(29) attach-agent $udp_(29)
$ns_ connect $udp_(29) $null_(29)
$ns_ at 55 "$cbr_(29) start"
#
#Total sources/connections: 10/30
#

# Arquivo que contém o padrão de comunicação para 15 nós e pacotes de 256 bytes (cbr-15-256)

#
# nodes: 15, max conn: 19, send rate: 0.25, seed: 9
#
#
# 3 connecting to 4 at time 154.69328709630915
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 154.69328709630915 "$cbr_(0) start"
#
# 3 connecting to 5 at time 34.627089991526255
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 256
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 34.627089991526255 "$cbr_(1) start"
#
# 4 connecting to 5 at time 62.686458613111853
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 256
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 62.686458613111853 "$cbr_(2) start"
#
```

```
# 4 connecting to 6 at time 20.051991092065347
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 256
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 20.051991092065347 "$cbr_(3) start"
#
# 5 connecting to 6 at time 140.70807377840768
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 256
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 140.70807377840768 "$cbr_(4) start"
#
# 6 connecting to 7 at time 85.909864728297052
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 256
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 85.909864728297052 "$cbr_(5) start"
#
# 8 connecting to 9 at time 4.3617707511232098
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 256
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 4.3617707511232098 "$cbr_(6) start"
#
# 8 connecting to 9 at time 132.75412509811767
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 256
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 132.75412509811767 "$cbr_(7) start"
#
# 9 connecting to 5 at time 103.18451128117019
```

```
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 256
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 103.18451128117019 "$cbr_(8) start"
#
# 9 connecting to 0 at time 152.5049444346246
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 256
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 152.5049444346246 "$cbr_(9) start"
#
# 0 connecting to 8 at time 14.69328709630915
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 256
$cbr_(10) set interval_ 0.125
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 14.69328709630915 "$cbr_(10) start"
#
# 1 connecting to 7 at time 44.69328709630915
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 256
$cbr_(11) set interval_ 0.125
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 44.69328709630915 "$cbr_(11) start"
#
# 2 connecting to 1 at time 23.75412509811767                    /1
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 256
$cbr_(12) set interval_ 0.125
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 23.75412509811767 "$cbr_(12) start"
#
# 7 connecting to 2 at time 53.75412509811767
#
```

```
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(13)
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 256
$cbr_(13) set interval_ 0.125
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 53.75412509811767 "$cbr_(13) start"
#
# 8 connecting to 0 at time 33.75412509811767                   /2
#
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 256
$cbr_(14) set interval_ 0.125
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 33.75412509811767 "$cbr_(14) start"
#
# 1 connecting to 3 at time 25.75412509811767
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 256
$cbr_(15) set interval_ 0.125
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 25.75412509811767 "$cbr_(15) start"
#
# 7 connecting to 0 at time 78.75412509811767                   /9
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 256
$cbr_(16) set interval_ 0.125
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 78.75412509811767 "$cbr_(16) start"
#
# 6 connecting to 0 at time 13.75412509811767
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 256
$cbr_(17) set interval_ 0.125
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
$ns_ connect $udp_(17) $null_(17)
$ns_ at 13.75412509811767 "$cbr_(17) start"
#
# 2 connecting to 0 at time 03.75412509811767
#
set udp_(18) [new Agent/UDP]
```

```
$ns_ attach-agent $node_(2) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 256
$cbr_(18) set interval_ 0.125
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $null_(18)
$ns_ at 03.75412509811767 "$cbr_(18) start"
#
# 6 connecting to 3 at time 33.75412509811767
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(19)
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 256
$cbr_(19) set interval_ 0.125
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 33.75412509811767 "$cbr_(19) start"
#
# 3 connecting to 10 at time 43.75412509811767
#
set udp_(20) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(20)
set null_(20) [new Agent/Null]
$ns_ attach-agent $node_(10) $null_(20)
set cbr_(20) [new Application/Traffic/CBR]
$cbr_(20) set packetSize_ 256
$cbr_(20) set interval_ 0.125
$cbr_(20) set random_ 1
$cbr_(20) set maxpkts_ 10000
$cbr_(20) attach-agent $udp_(20)
$ns_ connect $udp_(20) $null_(20)
$ns_ at 43.75412509811767 "$cbr_(20) start"
#
# 10 connecting to 7 at time 80.75412509811767
#
set udp_(21) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(21)
set null_(21) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(21)
set cbr_(21) [new Application/Traffic/CBR]
$cbr_(21) set packetSize_ 256
$cbr_(21) set interval_ 0.125
$cbr_(21) set random_ 1
$cbr_(21) set maxpkts_ 10000
$cbr_(21) attach-agent $udp_(21)
$ns_ connect $udp_(21) $null_(21)
$ns_ at 80.75412509811767 "$cbr_(21) start"
#
# 12 connecting to 14 at time 90.75412509811767
#
set udp_(22) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(22)
set null_(22) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(22)
set cbr_(22) [new Application/Traffic/CBR]
$cbr_(22) set packetSize_ 256
$cbr_(22) set interval_ 0.125
$cbr_(22) set random_ 1
$cbr_(22) set maxpkts_ 10000
$cbr_(22) attach-agent $udp_(22)
$ns_ connect $udp_(22) $null_(22)
$ns_ at 90.75412509811767 "$cbr_(22) start"
#
# 11 connecting to 5 at time 25
#
set udp_(23) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(23)
```

```
set null_(23) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(23)
set cbr_(23) [new Application/Traffic/CBR]
$cbr_(23) set packetSize_ 256
$cbr_(23) set interval_ 0.125
$cbr_(23) set random_ 1
$cbr_(23) set maxpkts_ 10000
$cbr_(23) attach-agent $udp_(23)
$ns_ connect $udp_(23) $null_(23)
$ns_ at 25 "$cbr_(23) start"
#
# 4 connecting to 13 at time 2
#
set udp_(24) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(24)
set null_(24) [new Agent/Null]
$ns_ attach-agent $node_(13) $null_(24)
set cbr_(24) [new Application/Traffic/CBR]
$cbr_(24) set packetSize_ 256
$cbr_(24) set interval_ 0.125
$cbr_(24) set random_ 1
$cbr_(24) set maxpkts_ 10000
$cbr_(24) attach-agent $udp_(24)
$ns_ connect $udp_(24) $null_(24)
$ns_ at 2 "$cbr_(24) start"
#
# 8 connecting to 11 at time 10
#
set udp_(25) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(25)
set null_(25) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(25)
set cbr_(25) [new Application/Traffic/CBR]
$cbr_(25) set packetSize_ 256
$cbr_(25) set interval_ 0.125
$cbr_(25) set random_ 1
$cbr_(25) set maxpkts_ 10000
$cbr_(25) attach-agent $udp_(25)
$ns_ connect $udp_(25) $null_(25)
$ns_ at 10 "$cbr_(25) start"
#
# 0 connecting to 12 at time 150
#
set udp_(26) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(26)
set null_(26) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(26)
set cbr_(26) [new Application/Traffic/CBR]
$cbr_(26) set packetSize_ 256
$cbr_(26) set interval_ 0.125
$cbr_(26) set random_ 1
$cbr_(26) set maxpkts_ 10000
$cbr_(26) attach-agent $udp_(26)
$ns_ connect $udp_(26) $null_(26)
$ns_ at 150 "$cbr_(26) start"
#
# 14 connecting to 9 at time 39
#
set udp_(27) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(27)
set null_(27) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(27)
set cbr_(27) [new Application/Traffic/CBR]
$cbr_(27) set packetSize_ 256
$cbr_(27) set interval_ 0.125
$cbr_(27) set random_ 1
$cbr_(27) set maxpkts_ 10000
$cbr_(27) attach-agent $udp_(27)
$ns_ connect $udp_(27) $null_(27)
$ns_ at 39 "$cbr_(27) start"
#
# 13 connecting to 2 at time 60
#
set udp_(28) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(28)
set null_(28) [new Agent/Null]
```

```
$ns_ attach-agent $node_(2) $null_(28)
set cbr_(28) [new Application/Traffic/CBR]
$cbr_(28) set packetSize_ 256
$cbr_(28) set interval_ 0.125
$cbr_(28) set random_ 1
$cbr_(28) set maxpkts_ 10000
$cbr_(28) attach-agent $udp_(28)
$ns_ connect $udp_(28) $null_(28)
$ns_ at 60 "$cbr_(28) start"
#
# 13 connecting to 4 at time 55
#
set udp_(29) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(29)
set null_(29) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(29)
set cbr_(29) [new Application/Traffic/CBR]
$cbr_(29) set packetSize_ 256
$cbr_(29) set interval_ 0.125
$cbr_(29) set random_ 1
$cbr_(29) set maxpkts_ 10000
$cbr_(29) attach-agent $udp_(29)
$ns_ connect $udp_(29) $null_(29)
$ns_ at 55 "$cbr_(29) start"
#
#Total sources/connections: 10/30
#

# Arquivo que contém o padrão de comunicação para 20 nós e pacotes de 64 bytes (cbr-20-64)
#
# nodes: 20, max conn: 500, send rate: 0.125, seed: 789
#
#
# 0 connecting to 4 at time 36.801180828735781
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 64
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 36.801180828735781 "$cbr_(0) start"
#
# 0 connecting to 17 at time 14.710156756783908
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(17) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 64
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 14.710156756783908 "$cbr_(1) start"
#
# 1 connecting to 12 at time 33.4597601194213
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 64
$cbr_(2) set interval_ 0.125
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
```

```
$ns_ connect $udp_(2) $null_(2)
$ns_ at 33.4597601194213 "$cbr_(2) start"
#
# 1 connecting to 6 at time 154.62318933318517
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 64
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 154.62318933318517 "$cbr_(3) start"
#
# 2 connecting to 0 at time 91.370741450866106
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 64
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 91.370741450866106 "$cbr_(4) start"
#
# 2 connecting to 19 at time 2.1195241446232069
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 64
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 2.1195241446232069 "$cbr_(5) start"
#
# 3 connecting to 5 at time 142.37367788440253
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 64
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 142.37367788440253 "$cbr_(6) start"
#
# 3 connecting to 18 at time 49.59253445714365
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(18) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 64
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
```

```
$ns_ at 49.59253445714365 "$cbr_(7) start"
#
# 4 connecting to 7 at time 134.03483506014331
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 64
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 134.03483506014331 "$cbr_(8) start"
#
# 4 connecting to 11 at time 81.460011155093099
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 64
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 81.460011155093099 "$cbr_(9) start"
#
# 5 connecting to 6 at time 41.379350778357754
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 64
$cbr_(10) set interval_ 0.125
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 41.379350778357754 "$cbr_(10) start"
#
# 5 connecting to 17 at time 30.630132523658745
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(17) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 64
$cbr_(11) set interval_ 0.125
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 30.630132523658745 "$cbr_(11) start"
#
# 6 connecting to 15 at time 60.248110760118863
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(15) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 64
$cbr_(12) set interval_ 0.125
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 60.248110760118863 "$cbr_(12) start"
```

```
#
# 6 connecting to 16 at time 76.761934476328051
#
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(16) $null_(13)
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 64
$cbr_(13) set interval_ 0.125
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 76.761934476328051 "$cbr_(13) start"
#
# 6 connecting to 1 at time 171.20345262400969
#
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 64
$cbr_(14) set interval_ 0.125
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 171.20345262400969 "$cbr_(14) start"
#
# 7 connecting to 10 at time 28.212300216877974
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(10) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 64
$cbr_(15) set interval_ 0.125
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 28.212300216877974 "$cbr_(15) start"
#
# 7 connecting to 0 at time 79.08115726387183
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 64
$cbr_(16) set interval_ 0.125
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 79.08115726387183 "$cbr_(16) start"
#
# 8 connecting to 18 at time 34.597689963224198
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(18) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 64
$cbr_(17) set interval_ 0.125
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
$ns_ connect $udp_(17) $null_(17)
$ns_ at 34.597689963224198 "$cbr_(17) start"
#
```

```
# 8 connecting to 19 at time 49.041115031131135
#
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 64
$cbr_(18) set interval_ 0.125
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $null_(18)
$ns_ at 49.041115031131135 "$cbr_(18) start"
#
# 9 connecting to 1 at time 85.076934725547645
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(19)
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 64
$cbr_(19) set interval_ 0.125
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 85.076934725547645 "$cbr_(19) start"
#
# 9 connecting to 3 at time 22.619450093535452
#
set udp_(20) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(20)
set null_(20) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(20)
set cbr_(20) [new Application/Traffic/CBR]
$cbr_(20) set packetSize_ 64
$cbr_(20) set interval_ 0.125
$cbr_(20) set random_ 1
$cbr_(20) set maxpkts_ 10000
$cbr_(20) attach-agent $udp_(20)
$ns_ connect $udp_(20) $null_(20)
$ns_ at 22.619450093535452 "$cbr_(20) start"
#
# 10 connecting to 12 at time 97.666381633685148
#
set udp_(21) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(21)
set null_(21) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(21)
set cbr_(21) [new Application/Traffic/CBR]
$cbr_(21) set packetSize_ 64
$cbr_(21) set interval_ 0.125
$cbr_(21) set random_ 1
$cbr_(21) set maxpkts_ 10000
$cbr_(21) attach-agent $udp_(21)
$ns_ connect $udp_(21) $null_(21)
$ns_ at 97.666381633685148 "$cbr_(21) start"
#
# 10 connecting to 13 at time 45.064882321779095
#
set udp_(22) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(22)
set null_(22) [new Agent/Null]
$ns_ attach-agent $node_(13) $null_(22)
set cbr_(22) [new Application/Traffic/CBR]
$cbr_(22) set packetSize_ 64
$cbr_(22) set interval_ 0.125
$cbr_(22) set random_ 1
$cbr_(22) set maxpkts_ 10000
$cbr_(22) attach-agent $udp_(22)
$ns_ connect $udp_(22) $null_(22)
$ns_ at 45.064882321779095 "$cbr_(22) start"
#
# 11 connecting to 7 at time 15.20295392498511
```

```
#
set udp_(23) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(23)
set null_(23) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(23)
set cbr_(23) [new Application/Traffic/CBR]
$cbr_(23) set packetSize_ 64
$cbr_(23) set interval_ 0.125
$cbr_(23) set random_ 1
$cbr_(23) set maxpkts_ 10000
$cbr_(23) attach-agent $udp_(23)
$ns_ connect $udp_(23) $null_(23)
$ns_ at 15.20295392498511 "$cbr_(23) start"
#
# 11 connecting to 14 at time 103.57831639404331
#
set udp_(24) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(24)
set null_(24) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(24)
set cbr_(24) [new Application/Traffic/CBR]
$cbr_(24) set packetSize_ 64
$cbr_(24) set interval_ 0.125
$cbr_(24) set random_ 1
$cbr_(24) set maxpkts_ 10000
$cbr_(24) attach-agent $udp_(24)
$ns_ connect $udp_(24) $null_(24)
$ns_ at 103.57831639404331 "$cbr_(24) start"
#
# 12 connecting to 14 at time 65.86812784702897
#
set udp_(25) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(25)
set null_(25) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(25)
set cbr_(25) [new Application/Traffic/CBR]
$cbr_(25) set packetSize_ 64
$cbr_(25) set interval_ 0.125
$cbr_(25) set random_ 1
$cbr_(25) set maxpkts_ 10000
$cbr_(25) attach-agent $udp_(25)
$ns_ connect $udp_(25) $null_(25)
$ns_ at 65.86812784702897 "$cbr_(25) start"
#
# 12 connecting to 15 at time 133.39811292169526
#
set udp_(26) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(26)
set null_(26) [new Agent/Null]
$ns_ attach-agent $node_(15) $null_(26)
set cbr_(26) [new Application/Traffic/CBR]
$cbr_(26) set packetSize_ 64
$cbr_(26) set interval_ 0.125
$cbr_(26) set random_ 1
$cbr_(26) set maxpkts_ 10000
$cbr_(26) attach-agent $udp_(26)
$ns_ connect $udp_(26) $null_(26)
$ns_ at 133.39811292169526 "$cbr_(26) start"
#
# 13 connecting to 2 at time 53.39811292169526
#
set udp_(27) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(27)
set null_(27) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(27)
set cbr_(27) [new Application/Traffic/CBR]
$cbr_(27) set packetSize_ 64
$cbr_(27) set interval_ 0.125
$cbr_(27) set random_ 1
$cbr_(27) set maxpkts_ 10000
$cbr_(27) attach-agent $udp_(27)
$ns_ connect $udp_(27) $null_(27)
$ns_ at 53.39811292169526 "$cbr_(27) start"
#
# 13 connecting to 4 at time 12.39811292169526
#
```

```
set udp_(28) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(28)
set null_(28) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(28)
set cbr_(28) [new Application/Traffic/CBR]
$cbr_(28) set packetSize_ 64
$cbr_(28) set interval_ 0.125
$cbr_(28) set random_ 1
$cbr_(28) set maxpkts_ 10000
$cbr_(28) attach-agent $udp_(28)
$ns_ connect $udp_(28) $null_(28)
$ns_ at 12.39811292169526 "$cbr_(28) start"
#
# 14 connecting to 5 at time 3.39811292169526
#
set udp_(29) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(29)
set null_(29) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(29)
set cbr_(29) [new Application/Traffic/CBR]
$cbr_(29) set packetSize_ 64
$cbr_(29) set interval_ 0.125
$cbr_(29) set random_ 1
$cbr_(29) set maxpkts_ 10000
$cbr_(29) attach-agent $udp_(29)
$ns_ connect $udp_(29) $null_(29)
$ns_ at 3.39811292169526 "$cbr_(29) start"
#
# 14 connecting to 6 at time 7.39811292169526
#
set udp_(30) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(30)
set null_(30) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(30)
set cbr_(30) [new Application/Traffic/CBR]
$cbr_(30) set packetSize_ 64
$cbr_(30) set interval_ 0.125
$cbr_(30) set random_ 1
$cbr_(30) set maxpkts_ 10000
$cbr_(30) attach-agent $udp_(30)
$ns_ connect $udp_(30) $null_(30)
$ns_ at 7.39811292169526 "$cbr_(30) start"
#
# 15 connecting to 8 at time 97.39811292169526
#
set udp_(31) [new Agent/UDP]
$ns_ attach-agent $node_(15) $udp_(31)
set null_(31) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(31)
set cbr_(31) [new Application/Traffic/CBR]
$cbr_(31) set packetSize_ 64
$cbr_(31) set interval_ 0.125
$cbr_(31) set random_ 1
$cbr_(31) set maxpkts_ 10000
$cbr_(31) attach-agent $udp_(31)
$ns_ connect $udp_(31) $null_(31)
$ns_ at 97.39811292169526 "$cbr_(31) start"
#
# 15 connecting to 9 at time 27.39811292169526
#
set udp_(32) [new Agent/UDP]
$ns_ attach-agent $node_(15) $udp_(32)
set null_(32) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(32)
set cbr_(32) [new Application/Traffic/CBR]
$cbr_(32) set packetSize_ 64
$cbr_(32) set interval_ 0.125
$cbr_(32) set random_ 1
$cbr_(32) set maxpkts_ 10000
$cbr_(32) attach-agent $udp_(32)
$ns_ connect $udp_(32) $null_(32)
$ns_ at 27.39811292169526 "$cbr_(32) start"
#
# 16 connecting to 2 at time 11.39811292169526
#
set udp_(33) [new Agent/UDP]
```

```
$ns_ attach-agent $node_(16) $udp_(33)
set null_(33) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(33)
set cbr_(33) [new Application/Traffic/CBR]
$cbr_(33) set packetSize_ 64
$cbr_(33) set interval_ 0.125
$cbr_(33) set random_ 1
$cbr_(33) set maxpkts_ 10000
$cbr_(33) attach-agent $udp_(33)
$ns_ connect $udp_(33) $null_(33)
$ns_ at 11.39811292169526 "$cbr_(33) start"
#
# 17 connecting to 7 at time 1.39811292169526
#
set udp_(34) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(34)
set null_(34) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(34)
set cbr_(34) [new Application/Traffic/CBR]
$cbr_(34) set packetSize_ 64
$cbr_(34) set interval_ 0.125
$cbr_(34) set random_ 1
$cbr_(34) set maxpkts_ 10000
$cbr_(34) attach-agent $udp_(34)
$ns_ connect $udp_(34) $null_(34)
$ns_ at 1.39811292169526 "$cbr_(34) start"
#
# 17 connecting to 11 at time 20.39811292169526
#
set udp_(35) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(35)
set null_(35) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(35)
set cbr_(35) [new Application/Traffic/CBR]
$cbr_(35) set packetSize_ 64
$cbr_(35) set interval_ 0.125
$cbr_(35) set random_ 1
$cbr_(35) set maxpkts_ 10000
$cbr_(35) attach-agent $udp_(35)
$ns_ connect $udp_(35) $null_(35)
$ns_ at 20.39811292169526 "$cbr_(35) start"
#
# 18 connecting to 9 at time 29.39811292169526
#
set udp_(36) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(36)
set null_(36) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(36)
set cbr_(36) [new Application/Traffic/CBR]
$cbr_(36) set packetSize_ 64
$cbr_(36) set interval_ 0.125
$cbr_(36) set random_ 1
$cbr_(36) set maxpkts_ 10000
$cbr_(36) attach-agent $udp_(36)
$ns_ connect $udp_(36) $null_(36)
$ns_ at 29.39811292169526 "$cbr_(36) start"
#
# 18 connecting to 12 at time 10.39811292169526
#
set udp_(37) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(37)
set null_(37) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(37)
set cbr_(37) [new Application/Traffic/CBR]
$cbr_(37) set packetSize_ 64
$cbr_(37) set interval_ 0.125
$cbr_(37) set random_ 1
$cbr_(37) set maxpkts_ 10000
$cbr_(37) attach-agent $udp_(37)
$ns_ connect $udp_(37) $null_(37)
$ns_ at 10.39811292169526 "$cbr_(37) start"
#
# 19 connecting to 3 at time 15.39811292169526
#
set udp_(38) [new Agent/UDP]
$ns_ attach-agent $node_(19) $udp_(38)
```

```
set null_(38) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(38)
set cbr_(38) [new Application/Traffic/CBR]
$cbr_(38) set packetSize_ 64
$cbr_(38) set interval_ 0.125
$cbr_(38) set random_ 1
$cbr_(38) set maxpkts_ 10000
$cbr_(38) attach-agent $udp_(38)
$ns_ connect $udp_(38) $null_(38)
$ns_ at 15.39811292169526 "$cbr_(38) start"
#
# 19 connecting to 0 at time 56.39811292169526
#
set udp_(39) [new Agent/UDP]
$ns_ attach-agent $node_(19) $udp_(39)
set null_(39) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(39)
set cbr_(39) [new Application/Traffic/CBR]
$cbr_(39) set packetSize_ 64
$cbr_(39) set interval_ 0.125
$cbr_(39) set random_ 1
$cbr_(39) set maxpkts_ 10000
$cbr_(39) attach-agent $udp_(39)
$ns_ connect $udp_(39) $null_(39)
$ns_ at 56.39811292169526 "$cbr_(39) start"

#
#Total sources/connections: 20/40
#

# Arquivo que contém o padrão de comunicação para 20 nós e pacotes de 256 bytes (cbr-20-256)


#
# nodes: 20, max conn: 500, send rate: 0.125, seed: 789
#
#
# 0 connecting to 4 at time 36.801180828735781
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.125
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 36.801180828735781 "$cbr_(0) start"
#
# 0 connecting to 17 at time 14.710156756783908
#
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(17) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 256
$cbr_(1) set interval_ 0.125
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 14.710156756783908 "$cbr_(1) start"
#
# 1 connecting to 12 at time 33.4597601194213
#
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 256
$cbr_(2) set interval_ 0.125
```

```
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 33.4597601194213 "$cbr_(2) start"
#
# 1 connecting to 6 at time 154.62318933318517
#
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 256
$cbr_(3) set interval_ 0.125
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 154.62318933318517 "$cbr_(3) start"
#
# 2 connecting to 0 at time 91.370741450866106
#
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 256
$cbr_(4) set interval_ 0.125
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 91.370741450866106 "$cbr_(4) start"
#
# 2 connecting to 19 at time 2.1195241446232069
#
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(5)
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 256
$cbr_(5) set interval_ 0.125
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 2.1195241446232069 "$cbr_(5) start"
#
# 3 connecting to 5 at time 142.37367788440253
#
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 256
$cbr_(6) set interval_ 0.125
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 142.37367788440253 "$cbr_(6) start"
#
# 3 connecting to 18 at time 49.59253445714365
#
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(3) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(18) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 256
$cbr_(7) set interval_ 0.125
$cbr_(7) set random_ 1
```

```
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 49.59253445714365 "$cbr_(7) start"
#
# 4 connecting to 7 at time 134.03483506014331
#
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(8)
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 256
$cbr_(8) set interval_ 0.125
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 134.03483506014331 "$cbr_(8) start"
#
# 4 connecting to 11 at time 81.460011155093099
#
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(4) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 256
$cbr_(9) set interval_ 0.125
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 81.460011155093099 "$cbr_(9) start"
#
# 5 connecting to 6 at time 41.379350778357754
#
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 256
$cbr_(10) set interval_ 0.125
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 41.379350778357754 "$cbr_(10) start"
#
# 5 connecting to 17 at time 30.630132523658745
#
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(17) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 256
$cbr_(11) set interval_ 0.125
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 30.630132523658745 "$cbr_(11) start"
#
# 6 connecting to 15 at time 60.248110760118863
#
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(15) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 256
$cbr_(12) set interval_ 0.125
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
```

```
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 60.248110760118863 "$cbr_(12) start"
#
# 6 connecting to 16 at time 76.761934476328051
#
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(16) $null_(13)
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 256
$cbr_(13) set interval_ 0.125
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 76.761934476328051 "$cbr_(13) start"
#
# 6 connecting to 1 at time 171.20345262400969
#
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 256
$cbr_(14) set interval_ 0.125
$cbr_(14) set random_ 1
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 171.20345262400969 "$cbr_(14) start"
#
# 7 connecting to 10 at time 28.212300216877974
#
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(10) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 256
$cbr_(15) set interval_ 0.125
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 28.212300216877974 "$cbr_(15) start"
#
# 7 connecting to 0 at time 79.08115726387183
#
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 256
$cbr_(16) set interval_ 0.125
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 79.08115726387183 "$cbr_(16) start"
#
# 8 connecting to 18 at time 34.597689963224198
#
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(18) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 256
$cbr_(17) set interval_ 0.125
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
```

```
$ns_ connect $udp_(17) $null_(17)
$ns_ at 34.597689963224198 "$cbr_(17) start"
#
# 8 connecting to 19 at time 49.041115031131135
#
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 256
$cbr_(18) set interval_ 0.125
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $null_(18)
$ns_ at 49.041115031131135 "$cbr_(18) start"
#
# 9 connecting to 1 at time 85.076934725547645
#
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(19)
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 256
$cbr_(19) set interval_ 0.125
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 85.076934725547645 "$cbr_(19) start"
#
# 9 connecting to 3 at time 22.619450093535452
#
set udp_(20) [new Agent/UDP]
$ns_ attach-agent $node_(9) $udp_(20)
set null_(20) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(20)
set cbr_(20) [new Application/Traffic/CBR]
$cbr_(20) set packetSize_ 256
$cbr_(20) set interval_ 0.125
$cbr_(20) set random_ 1
$cbr_(20) set maxpkts_ 10000
$cbr_(20) attach-agent $udp_(20)
$ns_ connect $udp_(20) $null_(20)
$ns_ at 22.619450093535452 "$cbr_(20) start"
#
# 10 connecting to 12 at time 97.666381633685148
#
set udp_(21) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(21)
set null_(21) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(21)
set cbr_(21) [new Application/Traffic/CBR]
$cbr_(21) set packetSize_ 256
$cbr_(21) set interval_ 0.125
$cbr_(21) set random_ 1
$cbr_(21) set maxpkts_ 10000
$cbr_(21) attach-agent $udp_(21)
$ns_ connect $udp_(21) $null_(21)
$ns_ at 97.666381633685148 "$cbr_(21) start"
#
# 10 connecting to 13 at time 45.064882321779095
#
set udp_(22) [new Agent/UDP]
$ns_ attach-agent $node_(10) $udp_(22)
set null_(22) [new Agent/Null]
$ns_ attach-agent $node_(13) $null_(22)
set cbr_(22) [new Application/Traffic/CBR]
$cbr_(22) set packetSize_ 256
$cbr_(22) set interval_ 0.125
$cbr_(22) set random_ 1
$cbr_(22) set maxpkts_ 10000
$cbr_(22) attach-agent $udp_(22)
$ns_ connect $udp_(22) $null_(22)
```

```
$ns_ at 45.064882321779095 "$cbr_(22) start"
#
# 11 connecting to 7 at time 15.20295392498511
#
set udp_(23) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(23)
set null_(23) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(23)
set cbr_(23) [new Application/Traffic/CBR]
$cbr_(23) set packetSize_ 256
$cbr_(23) set interval_ 0.125
$cbr_(23) set random_ 1
$cbr_(23) set maxpkts_ 10000
$cbr_(23) attach-agent $udp_(23)
$ns_ connect $udp_(23) $null_(23)
$ns_ at 15.20295392498511 "$cbr_(23) start"
#
# 11 connecting to 14 at time 103.57831639404331
#
set udp_(24) [new Agent/UDP]
$ns_ attach-agent $node_(11) $udp_(24)
set null_(24) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(24)
set cbr_(24) [new Application/Traffic/CBR]
$cbr_(24) set packetSize_ 256
$cbr_(24) set interval_ 0.125
$cbr_(24) set random_ 1
$cbr_(24) set maxpkts_ 10000
$cbr_(24) attach-agent $udp_(24)
$ns_ connect $udp_(24) $null_(24)
$ns_ at 103.57831639404331 "$cbr_(24) start"
#
# 12 connecting to 14 at time 65.86812784702897
#
set udp_(25) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(25)
set null_(25) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(25)
set cbr_(25) [new Application/Traffic/CBR]
$cbr_(25) set packetSize_ 256
$cbr_(25) set interval_ 0.125
$cbr_(25) set random_ 1
$cbr_(25) set maxpkts_ 10000
$cbr_(25) attach-agent $udp_(25)
$ns_ connect $udp_(25) $null_(25)
$ns_ at 65.86812784702897 "$cbr_(25) start"
#
# 12 connecting to 15 at time 133.39811292169526
#
set udp_(26) [new Agent/UDP]
$ns_ attach-agent $node_(12) $udp_(26)
set null_(26) [new Agent/Null]
$ns_ attach-agent $node_(15) $null_(26)
set cbr_(26) [new Application/Traffic/CBR]
$cbr_(26) set packetSize_ 256
$cbr_(26) set interval_ 0.125
$cbr_(26) set random_ 1
$cbr_(26) set maxpkts_ 10000
$cbr_(26) attach-agent $udp_(26)
$ns_ connect $udp_(26) $null_(26)
$ns_ at 133.39811292169526 "$cbr_(26) start"
#
# 13 connecting to 2 at time 53.39811292169526
#
set udp_(27) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(27)
set null_(27) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(27)
set cbr_(27) [new Application/Traffic/CBR]
$cbr_(27) set packetSize_ 256
$cbr_(27) set interval_ 0.125
$cbr_(27) set random_ 1
$cbr_(27) set maxpkts_ 10000
$cbr_(27) attach-agent $udp_(27)
$ns_ connect $udp_(27) $null_(27)
$ns_ at 53.39811292169526 "$cbr_(27) start"
```

```
#
# 13 connecting to 4 at time 12.39811292169526
#
set udp_(28) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(28)
set null_(28) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(28)
set cbr_(28) [new Application/Traffic/CBR]
$cbr_(28) set packetSize_ 256
$cbr_(28) set interval_ 0.125
$cbr_(28) set random_ 1
$cbr_(28) set maxpkts_ 10000
$cbr_(28) attach-agent $udp_(28)
$ns_ connect $udp_(28) $null_(28)
$ns_ at 12.39811292169526 "$cbr_(28) start"
#
# 14 connecting to 5 at time 3.39811292169526
#
set udp_(29) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(29)
set null_(29) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(29)
set cbr_(29) [new Application/Traffic/CBR]
$cbr_(29) set packetSize_ 256
$cbr_(29) set interval_ 0.125
$cbr_(29) set random_ 1
$cbr_(29) set maxpkts_ 10000
$cbr_(29) attach-agent $udp_(29)
$ns_ connect $udp_(29) $null_(29)
$ns_ at 3.39811292169526 "$cbr_(29) start"
#
# 14 connecting to 6 at time 7.39811292169526
#
set udp_(30) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(30)
set null_(30) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(30)
set cbr_(30) [new Application/Traffic/CBR]
$cbr_(30) set packetSize_ 256
$cbr_(30) set interval_ 0.125
$cbr_(30) set random_ 1
$cbr_(30) set maxpkts_ 10000
$cbr_(30) attach-agent $udp_(30)
$ns_ connect $udp_(30) $null_(30)
$ns_ at 7.39811292169526 "$cbr_(30) start"
#
# 15 connecting to 8 at time 97.39811292169526
#
set udp_(31) [new Agent/UDP]
$ns_ attach-agent $node_(15) $udp_(31)
set null_(31) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(31)
set cbr_(31) [new Application/Traffic/CBR]
$cbr_(31) set packetSize_ 256
$cbr_(31) set interval_ 0.125
$cbr_(31) set random_ 1
$cbr_(31) set maxpkts_ 10000
$cbr_(31) attach-agent $udp_(31)
$ns_ connect $udp_(31) $null_(31)
$ns_ at 97.39811292169526 "$cbr_(31) start"
#
# 15 connecting to 9 at time 27.39811292169526
#
set udp_(32) [new Agent/UDP]
$ns_ attach-agent $node_(15) $udp_(32)
set null_(32) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(32)
set cbr_(32) [new Application/Traffic/CBR]
$cbr_(32) set packetSize_ 256
$cbr_(32) set interval_ 0.125
$cbr_(32) set random_ 1
$cbr_(32) set maxpkts_ 10000
$cbr_(32) attach-agent $udp_(32)
$ns_ connect $udp_(32) $null_(32)
$ns_ at 27.39811292169526 "$cbr_(32) start"
#
```

```
# 16 connecting to 2 at time 11.39811292169526
#
set udp_(33) [new Agent/UDP]
$ns_ attach-agent $node_(16) $udp_(33)
set null_(33) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(33)
set cbr_(33) [new Application/Traffic/CBR]
$cbr_(33) set packetSize_ 256
$cbr_(33) set interval_ 0.125
$cbr_(33) set random_ 1
$cbr_(33) set maxpkts_ 10000
$cbr_(33) attach-agent $udp_(33)
$ns_ connect $udp_(33) $null_(33)
$ns_ at 11.39811292169526 "$cbr_(33) start"
#
# 17 connecting to 7 at time 1.39811292169526
#
set udp_(34) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(34)
set null_(34) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(34)
set cbr_(34) [new Application/Traffic/CBR]
$cbr_(34) set packetSize_ 256
$cbr_(34) set interval_ 0.125
$cbr_(34) set random_ 1
$cbr_(34) set maxpkts_ 10000
$cbr_(34) attach-agent $udp_(34)
$ns_ connect $udp_(34) $null_(34)
$ns_ at 1.39811292169526 "$cbr_(34) start"
#
# 17 connecting to 11 at time 20.39811292169526
#
set udp_(35) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(35)
set null_(35) [new Agent/Null]
$ns_ attach-agent $node_(11) $null_(35)
set cbr_(35) [new Application/Traffic/CBR]
$cbr_(35) set packetSize_ 256
$cbr_(35) set interval_ 0.125
$cbr_(35) set random_ 1
$cbr_(35) set maxpkts_ 10000
$cbr_(35) attach-agent $udp_(35)
$ns_ connect $udp_(35) $null_(35)
$ns_ at 20.39811292169526 "$cbr_(35) start"
#
# 18 connecting to 9 at time 29.39811292169526
#
set udp_(36) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(36)
set null_(36) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(36)
set cbr_(36) [new Application/Traffic/CBR]
$cbr_(36) set packetSize_ 256
$cbr_(36) set interval_ 0.125
$cbr_(36) set random_ 1
$cbr_(36) set maxpkts_ 10000
$cbr_(36) attach-agent $udp_(36)
$ns_ connect $udp_(36) $null_(36)
$ns_ at 29.39811292169526 "$cbr_(36) start"
#
# 18 connecting to 12 at time 10.39811292169526
#
set udp_(37) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(37)
set null_(37) [new Agent/Null]
$ns_ attach-agent $node_(12) $null_(37)
set cbr_(37) [new Application/Traffic/CBR]
$cbr_(37) set packetSize_ 256
$cbr_(37) set interval_ 0.125
$cbr_(37) set random_ 1
$cbr_(37) set maxpkts_ 10000
$cbr_(37) attach-agent $udp_(37)
$ns_ connect $udp_(37) $null_(37)
$ns_ at 10.39811292169526 "$cbr_(37) start"
#
# 19 connecting to 3 at time 15.39811292169526
```

```
#
set udp_(38) [new Agent/UDP]
$ns_ attach-agent $node_(19) $udp_(38)
set null_(38) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(38)
set cbr_(38) [new Application/Traffic/CBR]
$cbr_(38) set packetSize_ 256
$cbr_(38) set interval_ 0.125
$cbr_(38) set random_ 1
$cbr_(38) set maxpkts_ 10000
$cbr_(38) attach-agent $udp_(38)
$ns_ connect $udp_(38) $null_(38)
$ns_ at 15.39811292169526 "$cbr_(38) start"
#
# 19 connecting to 0 at time 56.39811292169526
#
set udp_(39) [new Agent/UDP]
$ns_ attach-agent $node_(19) $udp_(39)
set null_(39) [new Agent/Null]
$ns_ attach-agent $node_(0) $null_(39)
set cbr_(39) [new Application/Traffic/CBR]
$cbr_(39) set packetSize_ 256
$cbr_(39) set interval_ 0.125
$cbr_(39) set random_ 1
$cbr_(39) set maxpkts_ 10000
$cbr_(39) attach-agent $udp_(39)
$ns_ connect $udp_(39) $null_(39)
$ns_ at 56.39811292169526 "$cbr_(39) start"

#
#Total sources/connections: 20/40
#
```

#Padrões de Movimentação

```
# Exemplo de padrão de movimentação gerado pelo script setdest que se encontra na districuição do NS.

# Padrão de movimentação para 5 nodos.

#
# nodes: 5, speed type: 1, min speed: 3.00, max speed: 10.00
# avg speed: 2.75, pause type: 1, pause: 30.00, max x: 300.00, max y: 300.00
#
$node_(0) set X_ 118.580766991898
$node_(0) set Y_ 96.224346667217
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 49.873105663249
$node_(1) set Y_ 82.659892899277
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 169.966768043369
$node_(2) set Y_ 198.750465768412
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 163.777889512899
$node_(3) set Y_ 283.263418455327
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 161.381136432122
$node_(4) set Y_ 193.011119623276
$node_(4) set Z_ 0.000000000000
$ns_ at 0.000000000000 "$node_(0) setdest 93.168456614725 108.783574525030 6.284457479177"
$ns_ at 0.000000000000 "$node_(2) setdest 50.705261329816 56.817221506029 9.504991914155"
$ns_ at 0.000000000000 "$node_(3) setdest 190.062097304714 232.358729008263 8.728533302326"
$god_ set-dist 0 1 1
$god_ set-dist 0 2 1
$god_ set-dist 0 3 1
$god_ set-dist 0 4 1
$god_ set-dist 1 2 1
$god_ set-dist 1 3 1
$god_ set-dist 1 4 1
$god_ set-dist 2 3 1
$god_ set-dist 2 4 1
$god_ set-dist 3 4 1
$ns_ at 4.510560148745 "$node_(0) setdest 93.168456614725 108.783574525030 0.000000000000"
$ns_ at 6.563534010992 "$node_(3) setdest 190.062097304714 232.358729008263 0.000000000000"
$ns_ at 19.504176000135 "$node_(2) setdest 50.705261329816 56.817221506029 0.000000000000"
$ns_ at 30.000000000000 "$node_(1) setdest 201.711375023230 268.572460246056 9.170025555968"
$ns_ at 30.000000000000 "$node_(4) setdest 45.567466523040 39.739568290727 9.142195435375"
```

```
$ns_ at 34.510560148745 "$node_(0) setdest 227.199542116724 249.372027364881 8.876024121235"
$ns_ at 36.563534010992 "$node_(3) setdest 133.291865448632 169.063693090633 8.556032227272"
$ns_ at 46.500878128848 "$node_(3) setdest 133.291865448632 169.063693090633 0.000000000000"
$ns_ at 49.504176000135 "$node_(2) setdest 63.570252159287 78.417879601689 3.223648719843"
$ns_ at 51.013188393746 "$node_(4) setdest 45.567466523040 39.739568290727 0.000000000000"
$ns_ at 53.216644961355 "$god_ set-dist 1 4 2"
$ns_ at 53.306468070643 "$god_ set-dist 0 4 2"
$ns_ at 56.176394987581 "$node_(1) setdest 201.711375023230 268.572460246056 0.000000000000"
$ns_ at 56.394309623485 "$node_(0) setdest 227.199542116724 249.372027364881 0.000000000000"
$ns_ at 57.303266399514 "$node_(2) setdest 63.570252159287 78.417879601689 0.000000000000"
$ns_ at 76.500878128848 "$node_(3) setdest 190.611338719530 233.992840901602 9.185737543887"
$ns_ at 81.013188393746 "$node_(4) setdest 294.006484433391 68.710522840682 4.021943444052"
$ns_ at 85.929640818061 "$node_(3) setdest 190.611338719530 233.992840901602 0.000000000000"
$ns_ at 86.176394987581 "$node_(1) setdest 218.640663585919 84.427625257742 3.360000377027"
$ns_ at 86.394309623485 "$node_(0) setdest 36.772033001432 146.969317195382 7.778852208983"
$ns_ at 87.303266399514 "$node_(2) setdest 108.897968509881 122.027862486848 6.471767536860"
$ns_ at 87.595779625328 "$god_ set-dist 0 4 1"
$ns_ at 88.868830412755 "$god_ set-dist 1 4 1"
$ns_ at 97.022431729918 "$node_(2) setdest 108.897968509881 122.027862486848 0.000000000000"
$ns_ at 114.189548834417 "$node_(0) setdest 36.772033001432 146.969317195382 0.000000000000"
$ns_ at 115.929640818061 "$node_(3) setdest 27.378937065326 113.120979892361 8.314001925595"
$ns_ at 127.022431729918 "$node_(2) setdest 298.470361759382 23.296973492431 6.571231996598"
$ns_ at 138.044905015630 "$god_ set-dist 0 4 2"
$ns_ at 139.372141027374 "$god_ set-dist 3 4 2"
$ns_ at 140.359853463882 "$node_(3) setdest 27.378937065326 113.120979892361 0.000000000000"
$ns_ at 141.212517258561 "$node_(1) setdest 218.640663585919 84.427625257742 0.000000000000"
$ns_ at 143.202649067569 "$node_(4) setdest 294.006484433391 68.710522840682 0.000000000000"
$ns_ at 144.189548834417 "$node_(0) setdest 82.685043569302 49.649938598086 4.707080099283"
$ns_ at 150.465688012209 "$god_ set-dist 0 4 1"
$ns_ at 154.053529876236 "$god_ set-dist 2 3 2"
$ns_ at 159.549303635063 "$node_(2) setdest 298.470361759382 23.296973492431 0.000000000000"
$ns_ at 167.050019930264 "$node_(0) setdest 82.685043569302 49.649938598086 0.000000000000"
$ns_ at 170.359853463882 "$node_(3) setdest 226.074556685870 71.739221020080 5.748332981871"
$ns_ at 171.212517258561 "$node_(1) setdest 59.604549430717 172.757213617093 8.485189477661"
$ns_ at 173.202649067569 "$node_(4) setdest 289.301987102910 39.630684848684 8.298101729947"
$ns_ at 173.907884187205 "$god_ set-dist 3 4 1"
$ns_ at 176.597163028807 "$god_ set-dist 2 3 1"
$ns_ at 176.752608410990 "$node_(4) setdest 289.301987102910 39.630684848685 0.000000000000"
$ns_ at 188.902033217960 "$god_ set-dist 1 2 2"
$ns_ at 189.549303635063 "$node_(2) setdest 51.857051040574 42.268966532300 7.164573333513"
$ns_ at 190.826577825230 "$god_ set-dist 1 4 2"
$ns_ at 192.652134868154 "$node_(1) setdest 59.604549430717 172.757213617093 0.000000000000"
$ns_ at 194.645378708253 "$god_ set-dist 1 2 1"
$ns_ at 197.050019930264 "$node_(0) setdest 149.042284535429 299.731288674313 7.028875269463"
$ns_ at 205.667325718864 "$node_(3) setdest 226.074556685870 71.739221020080 0.000000000000"
$ns_ at 206.752608410990 "$node_(4) setdest 204.522810978988 153.801689830318 4.344938802707"
$ns_ at 210.656421163092 "$god_ set-dist 1 4 1"
$ns_ at 222.652134868154 "$node_(1) setdest 179.815607774455 218.137254419864 9.345921265881"
$ns_ at 224.072223782829 "$node_(2) setdest 51.857051040574 42.268966532300 0.000000000000"
$ns_ at 230.255529013295 "$god_ set-dist 0 2 2"
$ns_ at 233.860364212741 "$node_(0) setdest 149.042284535429 299.731288674313 0.000000000000"
$ns_ at 235.667325718864 "$node_(3) setdest 3.454285072743 82.542152718820 7.945818181087"
$ns_ at 236.400530849419 "$node_(1) setdest 179.815607774455 218.137254419864 0.000000000000"
$ns_ at 239.481707349028 "$node_(4) setdest 204.522810978988 153.801689830318 0.000000000000"
$ns_ at 254.072223782829 "$node_(2) setdest 129.702495103264 30.596199576745 5.572628429650"
$ns_ at 260.715117804172 "$god_ set-dist 0 3 2"
$ns_ at 263.717581262901 "$node_(3) setdest 3.454285072743 82.542152718820 0.000000000000"
$ns_ at 263.860364212741 "$node_(0) setdest 213.680931074925 93.440421027040 6.695642841676"
$ns_ at 266.400530849419 "$node_(1) setdest 92.255174808455 198.684037296632 6.094081753299"
$ns_ at 266.701411211541 "$god_ set-dist 0 3 1"
$ns_ at 267.028397059315 "$god_ set-dist 0 2 1"
$ns_ at 268.197646672855 "$node_(2) setdest 129.702495103264 30.596199576745 0.000000000000"
$ns_ at 269.481707349028 "$node_(4) setdest 276.021693416990 10.812257614208 7.535775954239"
$ns_ at 281.118967635855 "$node_(1) setdest 92.255174808455 198.684037296632 0.000000000000"
$ns_ at 283.650763221908 "$god_ set-dist 3 4 2"
$ns_ at 288.907101166542 "$god_ set-dist 1 4 2"
$ns_ at 290.696365202882 "$node_(4) setdest 276.021693416990 10.812257614208 0.000000000000"
$ns_ at 293.717581262901 "$node_(3) setdest 104.275744312758 111.617667464650 3.737444729635"
$ns_ at 296.147130796214 "$node_(0) setdest 213.680931074925 93.440421027040 0.000000000000"
$ns_ at 298.197646672855 "$node_(2) setdest 78.347482228716 284.749505931488 7.127838624779"
$ns_ at 303.877586243197 "$god_ set-dist 3 4 1"
$ns_ at 311.118967635855 "$node_(1) setdest 254.484707893063 194.107895128998 3.853946927148"
$ns_ at 315.855008964430 "$god_ set-dist 1 4 1"
$ns_ at 320.516717792605 "$god_ set-dist 2 4 2"
$ns_ at 320.696365202882 "$node_(4) setdest 10.750868133311 241.571039577661 5.727598477467"
$ns_ at 321.792973137187 "$node_(3) setdest 104.275744312758 111.617667464650 0.000000000000"
```

```
$ns_ at 326.147130796214 "$node_(0) setdest 32.444141374058 63.817263016009 8.618051106938"
$ns_ at 334.574715650584 "$node_(2) setdest 78.347482228716 284.749505931488 0.000000000000"
$ns_ at 336.596093316660 "$god_ set-dist 2 4 1"
$ns_ at 347.456100000201 "$node_(0) setdest 32.444141374059 63.817263016009 0.000000000000"
$ns_ at 350.938425007963 "$god_ set-dist 0 1 2"
$ns_ at 351.792973137187 "$node_(3) setdest 190.838346792007 242.880980372253 3.271187373299"
$ns_ at 353.230098075925 "$node_(1) setdest 254.484707893063 194.107895128998 0.000000000000"
$ns_ at 364.574715650584 "$node_(2) setdest 256.140099344109 265.391906211506 6.364827213782"
$ns_ at 377.456100000201 "$node_(0) setdest 183.369512120462 95.049097684727 3.238236975054"
$ns_ at 379.887492502183 "$god_ set-dist 0 1 1"
$ns_ at 381.978556647487 "$god_ set-dist 0 2 2"
$ns_ at 382.082273862475 "$node_(4) setdest 10.750868133311 241.571039577661 0.000000000000"
$ns_ at 383.230098075925 "$node_(1) setdest 264.306818641680 45.733422975184 3.626744370526"
$ns_ at 384.970734836232 "$god_ set-dist 1 4 2"
$ns_ at 392.673406962339 "$node_(2) setdest 256.140099344109 265.391906211506 0.000000000000"
$ns_ at 396.437385077303 "$god_ set-dist 0 2 1"
$ns_ at 399.859905439863 "$node_(3) setdest 190.838346792007 242.880980372253 0.000000000000"
#
# Destination Unreachables: 0
#
# Route Changes: 29
#
# Link Changes: 29
#
# Node | Route Changes | Link Changes
#    0 |            12 |           12
#    1 |            11 |           11
#    2 |            10 |           10
#    3 |             8 |            8
#    4 |            17 |           17
#
```

ANEXO B – ARTIGO

Análise de Métricas de QoS para SLAs em Redes *Ad Hoc* com Roteamento Seguro

Darlan Vivian, Eduardo Adilio Pelinson Alchieri, Carlos Becker Westphall

INE – Departamento de Informática e Estatística
UFSC – Universidade Federal de Santa Catarina
Campus Universitário, C. P. 476 – CEP 88040-970 – Trindade – Florianópolis – SC

{darlan,edu,westphall}@inf.ufsc.br

Abstract. *In the last years the wireless communication has been winning prominence among the technologies for data transmission. The use of Ad Hoc wireless networks involves some factors as safety and establishment of SLAs. In this work they are obtained and analyzed, through simulations, data about the behavior of a Ad Hoc network by some metric of QoS that compose SLAs, taking into account the impact caused by the addition of mechanisms of safety in the communications, through protocols of safe routing.*

Resumo. *Nos últimos anos a comunicação sem fio tem ganhado destaque entre as tecnologias para transmissão de dados. A utilização de redes sem fio Ad Hoc envolve alguns fatores tais como segurança e estabelecimento de SLAs. Neste trabalho são obtidos e analisados, através de simulações, dados sobre o comportamento de uma rede Ad Hoc mediante certas métricas de QoS que compõem os SLAs, levando em consideração o impacto causado pela adição de mecanismos de segurança nas comunicações, através de protocolos de roteamento seguro.*

1. Introdução

O surgimento de novas tecnologias possibilitou grandes avanços nas comunicações, proporcionando aos usuários, serviços com a melhor qualidade possível. Nos últimos anos a comunicação sem fio tem ganhado destaque entre as tecnologias para transmissão de dados, estando cada vez mais presente em nosso cotidiano. Empresas, universidades e outras instituições estão adotando cada vez mais esta tecnologia, comprovando a tendência de que nos próximos anos as redes sem fio substituirão ou serão adicionadas aos sistemas com fio já existentes.

Um dos fatores que limitava o uso da tecnologia sem fio era a baixa taxa de transferência oferecida por esta, não atendendo de forma satisfatória as necessidades de comunicação, principalmente no meio empresarial. Através de pesquisas, que deram origem a novos padrões como o IEEE 802.11g, obteve-se um incremento na largura de banda das redes e desta forma estimulou-se a sua adoção em diversas aplicações.

Outro fator crítico em relação ao uso desta tecnologia está ligado a segurança, principalmente no que diz respeito ao roteamento em redes *Ad Hoc*, pois a maioria dos protocolos é cooperativa por natureza [Royer and Toh, 1999], e confiam em seus vizinhos no roteamento de pacotes. Esta confiança permite que nodos maliciosos paralisem uma rede sem fio inserindo atualizações de roteamento erradas, reproduzindo antigas informações de roteamento, trocando atualizações de roteamento ou publicando informações incorretas de roteamento. Enquanto estes ataques são possíveis em redes cabeadas, a natureza dos ambientes sem fio aumenta seus efeitos, e faz com que a sua detecção seja difícil. Várias pesquisas tem sido realizadas com o intuito de sanar ou pelo

menos minimizar os problemas causados pela falta de segurança e desta forma alguns protocolos de roteamento seguro foram propostos.

Para os grandes negócios, as telecomunicações tornaram-se uma ferramenta vital, demandando um alto nível de confiabilidade no que se refere aos serviços que são oferecidos aos seus usuários. Desta forma, a integridade da rede passa a ter um papel fundamental, sendo que além de um bom desempenho, deve ser garantido que o serviço seja oferecido com qualidade e confiabilidade de acordo com as necessidades estipuladas pelo cliente.

Com a utilização cada vez maior das redes sem fio e o aumento da variedade dos serviços oferecidos, as relações contratuais entre fornecedores de serviços e clientes estão se tornando complexas. Desta forma, a formalização de acordos, nos quais estão definidos os parâmetros de qualidade com seus respectivos níveis desejados é de grande importância tanto para o fornecedor quanto para o consumidor [Pereira, 2003].

Neste contexto, surgem os SLAs (*Service Level Agreements*), que são contratos estabelecidos entre o consumidor e o fornecedor do serviço. Nestes contratos os clientes definem níveis de serviço, parâmetros e métricas que refletem o bom funcionamento da rede.

Este trabalho tem como objetivo obter informações, através de simulações, sobre uma rede sem fio *Ad Hoc* mediante certas métricas de QoS (*Quality of Service*), verificando o impacto causado pela adição de mecanismos de segurança nos protocolos de roteamento e analisar estes dados com o objetivo de moldar as métricas de qualidade de serviço que compõem os SLAs.

O texto está organizado da seguinte forma: na seção 2 temos uma breve descrição sobre redes locais sem fio *Ad Hoc*, dando ênfase às questões que envolvem o roteamento de pacotes. A seção 3 apresenta os principais conceitos referentes a SLAs, destacando as métricas de QoS analisadas neste trabalho. A seção 4 descreve o ambiente de simulação, ainda nesta seção são apresentados e analisados os dados obtidos. Finalmente, a seção 5 relata alguns experimentos similares da literatura e na seção 6 temos as conclusões do trabalho e perspectivas de trabalhos futuros.

2. Redes Locais Sem Fio *Ad Hoc*

As WLANs (*Wireless Local Area Network*) constituem-se como uma alternativa para as LANs (*Local Area Network*), pois fornecem as mesmas funcionalidades, porém de forma flexível, são de fácil instalação e proporcionam uma boa conectividade. Por permitirem a mobilidade, estas redes facilitam a utilização do poder computacional, tornando transparente a disseminação da informação e a cooperação dos dispositivos na realização das mais variadas tarefas.

Conforme especificações do padrão IEEE 802.11, as redes locais sem fio podem ser configuradas de dois modos distintos: com infra-estrutura e sem infra-estrutura. As redes com infra-estrutura são formadas pelas estações e pelos APs (*Access Point*) que são os responsáveis por boa parcela da funcionalidade da rede, pois nesta topologia uma estação não é capaz de estabelecer uma comunicação diretamente com outra sem que a informação passe por um AP. Já as redes sem infra-estrutura, também conhecidas como redes *Ad Hoc* ou MANET (*Mobile Ad Hoc Network*), são formadas somente por estações móveis (nodos) dentro de uma área restrita, que se comunicam sem a necessidade de um AP.

Nas redes sem fio *Ad Hoc* o endereço de destino nem sempre corresponde ao local do destino, isto eleva a dificuldade no envio de informações pela rede. Em determinado ponto no tempo, dependendo das posições dos nodos e do padrão de cobertura de seus transmissores e receptores, do nível de potência da transmissão e do nível de interferência no subcanal, uma conectividade sem fio randômica (*Ad Hoc*)

existe entre os nodos. Esta topologia pode mudar com o tempo devido à movimentação dos nodos ou ajuste dos parâmetros de transmissão e recepção. Nestas redes as estações cooperam entre si na transmissão de pacotes, pois caso os nodos não estejam na mesma área de cobertura do sinal, a rota entre eles pode ser formada por vários *hops* (saltos) através de um ou mais nodos na rede, caracterizando esta topologia como sendo *multihop* (múltiplos saltos).

Por não requererem nenhuma administração centralizada ou uma infra-estrutura de rede fixa como estações base ou pontos de acesso, estas redes podem ser rapidamente configuradas de acordo com as necessidades. Exemplos de aplicações para redes *Ad Hoc* variam desde operações militares até a interação entre participantes presentes em uma reunião ou estudantes durante uma conferência.

Devido as suas propriedades, as redes do tipo MANET possuem algumas características tais como [Corson, 1999]:

- Topologia dinâmica: os nodos são livres para se mover arbitrariamente; assim, a topologia da rede (que é tipicamente *multihop*) pode mudar fortuitamente e rapidamente em intervalos de tempos imprevisíveis.
- Limitação da Largura de Banda: *links* sem fio possuem tracidionalmente uma capacidade significativamente mais baixa se comparados aos *links* cabeados. Além disso, o *throughput* alcançado em uma comunicação sem fio é consideravelmente prejudicado devido aos efeitos causados por acessos múltiplos, desvanecimento do sinal, barulho e condições de interferência.
- Limitação de energia: alguns ou todos os nodos em uma MANET precisam confiar em baterias ou outros meios exaustivos de energia. Para estes nodos, um dos critérios mais importante de projeto de sistema é a otimização do consumo de energia.
- Limitações na segurança: redes sem fio móveis são geralmente mais propensas a ameaças de segurança física do que as redes com fio. Assim, o aumento da possibilidade de ocorrência de ataques de *eavesdropping* (escutar clandestinamente), *spoofing* e DoS (*Denial of Service*) devem ser considerados cuidadosamente.

Devido a estas características, várias pesquisas têm sido desenvolvidas no planejamento de protocolos de roteamento para prover uma efetiva e eficiente comunicação entre os nodos que fazem parte da rede.

2.1. Protocolos de Roteamento

A tarefa de roteamento em uma rede *Ad Hoc* é mais difícil do que em redes cabeadas, pois este depende de muitos fatores incluindo topologia, seleção de roteadores, iniciação da requisição, e características subjacentes específicas que podem servir de heurísticas para encontrar rapidamente e eficientemente o caminho pelo qual os pacotes devem ser enviados.

Um dos principais desafios no projeto de um protocolo para redes *Ad Hoc* está no fato de que um nodo precisa conhecer pelo menos a informação de localização de seus nodos vizinhos para determinar uma rota para os pacotes. Além disso, como o número de nodos da rede pode ser grande, encontrar a rota para os destinatários requer uma grande e freqüente troca de informações de controle entre os nodos. Assim, a quantidade de tráfego de atualizações pode ser bastante alta, e é ainda maior quando nodos com alta mobilidade estiverem presentes [Agrawal and Cordeiro, 2004].

Existem vários protocolos de roteamento os quais podem ser classificados em dois grupos:

- *Table-driven*: também chamados de pró-ativos, este grupo mantém as rotas de todos os possíveis destinatários em uma tabela de forma que quando um pacote precisar ser remetido, a rota já é conhecida e pode ser usada imediatamente. Como exemplo, temos o protocolo DSDV (*Destination-Sequenced Distance-Vector Routing*) [He, 2003].

- *On-demand*: também chamados de reativos, aqui as rotas dos destinatários apenas são descobertas sob demanda, isto é, um nodo não precisa conhecer a rota de um destinatário até que ele necessite enviar pacotes de dados para este destino. Faz parte deste grupo o protocolo DSR (*Dynamic Source Routing*) [Johnson et al., 2001].

Cada protocolo de roteamento *Ad Hoc* possui suas vantagens e desvantagens, de acordo com determinadas situações e desta forma não existe um que seja melhor do que os outros. No entanto o *Mobile* Ad *Hoc Network Working Group* especificou uma série de propriedades que um protocolo deve possuir [Corson, 1999], tais como:

- Operação distribuída: para evitar a centralização que leva à vulnerabilidade. Esta propriedade é essencial para o roteamento em uma rede *Ad Hoc*.

- Livre de *loops*: para que os pacotes não fiquem trafegando durante um período de tempo relativamente grande na rede, pode ser adotada como solução uma variável do tipo TTL (*time to live*). Mas uma abordagem melhor estruturada seria mais indicada, como por exemplo a utilização de número de seqüência.

- Operação baseada na demanda: Em vez de assumir uma distribuição uniforme de tráfego dentro da rede (e manter o roteamento entre os nodos a todo o momento), permitiria ao protocolo de roteamento se adaptar ao padrão de tráfego baseado na demanda ou na necessidade básica. Se isto for realizado inteligentemente, podem-se utilizar mais eficientemente os recursos de energia e a largura da banda da rede, às custas do aumento no tempo de descobrimento de rota.

- Operação pró-ativa: em certos contextos, a latência adicional causada pela operação baseada na demanda pode ser inaceitável. A operação pró-ativa é desejável nos casos onde os recursos de energia e a largura da banda permitirem.

- Segurança: sem alguma forma de segurança no nível de rede ou na camada de ligação, os protocolos de roteamento para redes *Ad Hoc* ficam vulneráveis a muitos tipos de ataques.

- Período de "dormência": como resultado da conservação de energia, ou de outra necessidade, nodos de um MANET podem parar de transmitir e\ou receber por períodos arbitrários de tempo. Um protocolo deve ser capaz de acomodar tais "períodos de dormência" sem trazer conseqüências adversas.

- Suporte a enlaces (*links*) unidirecionais: ligações bidirecionais são tipicamente assumidas no projeto de protocolos de roteamento, no entanto muitos deles são incapazes de funcionar corretamente sobre ligações unidirecionais.

A mudança dinâmica de topologia, ambiente aberto e a falta de uma infra-estrutura centralizada de segurança, tornam uma rede sem fio *Ad Hoc* extremamente vulnerável a presença de nodos maliciosos e a certos tipos de ataques. Não há nenhuma garantia que o caminho de comunicação (escolhido no roteamento) é livre de nodos maliciosos, os quais não obedecerão ao protocolo empregado e tentarão interferir nas operações da rede. Em [Yih-Chun et al., 2002a] pode ser encontrada a descrição de

vários ataques realizados nos protocolos de roteamento em redes *Ad Hoc*, os quais podem ser classificados em dois tipos:

- Ataques de quebra de rotas: este ataque tenta fazer com que pacotes de dados legítimos sejam roteados para caminhos disfuncionais. Um exemplo deste ataque é o *black hole*, onde o atacante, através do envio de pacotes de roteamento forjados, roteia todos os pacotes enviados para algum destino para si próprio e depois descarta estes pacotes.

- Ataques de consumo de recursos: este ataque injeta pacotes na rede com o objetivo de consumir recursos como: largura de banda, poder de processamento e memória. Um exemplo deste ataque é o *routing loop*, onde o atacante envia pacotes de roteamento forjados, que cria um ciclo passando pelos nós sem achar o destino, consumindo recursos.

A maioria dos protocolos de roteamento propostos na literatura assume ambientes não hostis. No entanto, devido as estas vulnerabilidades, várias pesquisas foram realizadas no intuito de solucionar este problema, surgindo assim uma nova categoria de protocolos de roteamento em redes *Ad Hoc* chamada de protocolos de roteamento seguro.

O projeto de protocolos de roteamento seguro para este tipo de rede apresenta algumas dificuldades, devido à natureza geralmente altamente dinâmica e à necessidade de operar eficazmente com recursos limitados, incluindo a largura da banda de rede, capacidade de processamento da CPU, memória e bateria (energia) de cada nodo na rede.

Focalizando estas preocupações, recentemente foram propostos vários protocolos de roteamento seguro. Dentre eles daremos destaque ao SEAD (*Secure Efficient Ad hoc Distance vector*) [Yih-Chun, 2002b] que é um protocolo pró-ativo baseado no DSDV e ao *Ariadne* [Yih-Chun, 2002a] que é um protocolo reativo baseado no DSR.

3. Acordos de Níveis de Serviços - SLAs

Cada vez mais as empresas estão contratando serviços baseados em níveis de serviços, conhecidos como SLAs. Para administrar esses contratos é necessário que tanto as prestadoras de serviços como as empresas contratantes possuam um gerenciamento dos parâmetros contratos. A simplicidade e conhecimento pleno dos parâmetros contratados são fundamentais para a boa gestão do contrato, evitando conflitos pessoais e contratuais entre empresas e provedores dos serviços.

A falta de estabilidade pavimenta o surgimento e conceito de SLAs e origina-se do fato de que pessoas querem pagar por um serviço que se comportará do mesmo modo o tempo todo. Estes acordos estão se tornando um componente necessário no dia a dia de uma empresa que utiliza redes de computadores. Nestes contratos, os parâmetros de performance são especificados, os quais refletem o bom funcionamento da rede. Embora estes contratos normalmente englobem os serviços de telecomunicações, eles também podem incluir os serviços de tecnologia de informação e outros fatores referentes ao comércio dentro de uma companhia.

De uma maneira resumida, SLA é um contrato entre um provedor de serviços e o cliente, o qual estipula os níveis de serviços exigidos. Um SLA deve conter os níveis de serviços especificados, opções de suporte, as penalidades pela quebra do contrato (não fornecimento de serviços no nível especificado), as possíveis recompensas por exceder o nível de serviço especificado, o tempo de manutenção, a que software ou hardware será fornecido e a que taxa. Como pode ser observado, esta lista não é completa e pode mudar de acordo para acordo, dependendo das necessidades de cada cliente.

Existem dois tipos de acordos de nível de serviços: SLA Externo e SLA Interno. O primeiro é bastante rigoroso, pois geralmente este contrato é firmado entre duas empresas. Já o segundo é um acordo entre diferentes áreas de uma mesma empresa, que visa assegurar qualidade nas diversas áreas da empresa, sendo geralmente utilizado em grandes companhias.

Em [Pereira, 2003] é proposto a gerência e o estabelecimento de um SLA pró-ativo, o qual detecta a possibilidade de quebra em alguma métrica do acordo com antecedência (antes que realmente ocorra) e realiza determinadas ações no intuito de evitar a possível degradação da rede. Caso esta solução não seja possível, novas ações são tomadas no sentido de renegociar o acordo com um servidor de SLAs.

As métricas para o estabelecimento de um SLA podem ser classificadas em independentes da tecnologia (percentual de disponibilidade, tempo de recuperação de falhas, tempo de instalação de serviços, etc) e dependentes da tecnologia (perda, retransmissão, latência, etc). Dependendo do tipo de aplicação determinadas métricas serão ou não importantes. No escopo deste trabalho será dado maior atenção as seguintes métricas:

- Vazão: também denominada de *througput*, é uma medida de transmissão de dados que determina a quantidade de dados movida de um nó a outro da rede em um determinado período de tempo. Em termos práticos, as aplicações geram vazões que devem ser atendidas pela rede. A vazão na maioria das redes sofre variações no decorrer no tempo e é normalmente medida em Kbps ou Mbps.

- Latência: tempo que um pacote leva para ir de um ponto a outro da rede, normalmente medido em milissegundos. Quanto menor for a latência, melhor o tempo de resposta da rede, se a latência for muito grande torna-se difícil o uso de interatividade na rede. A forma de se calcular a latência é o tempo em que o pacote chega ao canal de comunicação (emissor) e a observação do ultimo *bit* do pacote (receptor). Os principais responsáveis pela latência são o atraso de transmissão, codificação, decodificação, empacotamento e desempacotamento [Pereira 2003].

- *Jitter*: é a variação do atraso. Em redes a pacotes, os fluxos são adicionalmente divididos em blocos de dados, e cada bloco é transmitido em seqüência. Se a rede é capaz de enviar todos os blocos com uma seqüência uniforme, então cada bloco deveria chegar no destino após um atraso uniforme. Muitas redes não garantem um atraso uniforme para seus usuários. Variações no atraso são comuns e causadas por muitos fatores, tais como: diferenças de tempo de processamento dos pacotes, diferenças de tempo de acesso à rede e diferenças de tempo de enfileiramento.

- Perda de pacotes: índice que mede a taxa de sucesso na transmissão de pacotes entre dois pontos da rede. Quanto menor a perda de pacotes, maior a eficiência da rede.

4. Simulações

Simulação é uma técnica muito utilizada para a avaliação de diversas características de sistemas. É usada principalmente quando o sistema a ser analisado ainda não está disponível, principalmente em muitas situações onde esta é a forma mais fácil e menos custosa de se prever o comportamento de um sistema. O *software* utilizado nas simulações foi o NS-2 (*Network Simulator*), o qual é um aplicativo de simulação de redes dirigido por eventos.

4.1. Ambiente de Simulação

Para realizar as simulações procurou-se modelar um sistema o mais próximo possível com um ambiente real. Este ambiente caracteriza-se por possuir um desempenho que atende as necessidades de comunicação tanto de usuários que o utilizam como dos próprios dispositivos *wireless* pertencentes ao mesmo.

Na realização das simulações alguns parâmetros foram variados de acordo com a análise a ser realizada, no entanto outros nunca sofreram alterações. A tabela 1 apresenta os parâmetros utilizados nas simulações, nela podemos observar quais foram os parâmetros que sofreram variações e quais permaneceram fixos ao longo das simulações.

A obtenção dos padrões de comunicação e movimentação se deu através do uso de *scripts* presentes na distribuição do NS utilizada. O simulador utiliza estes padrões para variar a movimentação dos nós e a comunicação entre eles.

Table 1. Parâmetros utilizados nas simulações

Tipo de canal: Wireless Channel	Tipo de tráfego: Dados – CBR
Modelo de propagação: Two Ray Ground	Largura de banda: 2 Mbps
Tipo de interface: Wireless Phy	Protocolo de Roteamento: DSR, DSDV, *Ariadne* e SEAD
Tipo de MAC: 802.11	Número de estações móveis: 5, 10, 15, 20
Tipo de camada de ligação: LL	Tamanho dos pacotes: 64 e 256 bytes
Modelo de antena: Omni Antenna	Tipo de Fila: FIFO
Dimensões do ambiente: 300 X 300m	Número máximo de pacotes na fila: 10
Modelo de mobilidade: WayPoint	Tempo de simulação: 400 segundos
Protocolo de conexão: UDP	

O parâmetro de mobilidade escolhido foi o *WayPoint*, o qual caracteriza-se por dividir o percurso de cada nodo em períodos de movimentação e de pausa, ou seja, cada nó escolhe um destino e uma velocidade aleatórios na área estipulada, andando então para o destino e permanecendo lá durante o tempo de pausa, após volta a escolher outra posição e velocidade e assim sucessivamente até terminar o tempo da simulação. No padrão criado especificou-se uma velocidade de movimentação uniforme escolhida aleatoriamente entre 3 m/s e 10 m/s obtendo uma média de movimentação de 2.75 m/s, considerando o tempo de pausa que foi fixado em 30 segundos.

Este modelo de simulação exemplifica um ambiente *wireless multihop* com dimensões de 300 X 300m, onde existem 5, 10, 15 e 20 nós, os quais se comunicam com o auxilio de algum nó intermediário, quando necessário. Sendo assim, este ambiente assemelha-se, por exemplo, com um escritório ou fábrica, onde existe necessidade de comunicação tanto de usuários como de outros dispositivos *wireless*.

Os pacotes têm tamanho de 64 e 256 bytes, simulando apenas tráfego do tipo dados, mais especificamente CBR (*Constant Bit Rate*), que são enviados por um canal com largura de banda de 2 Mbps através de uma conexão UDP (*User Datagram Protocol*). O tempo de duração e ativação de cada conexão foi estabelecido de maneira aleatória, assim cada uma tem a mesma probabilidade de ocorrer a qualquer momento.

A política de fila utilizada nestas simulações foi a FIFO, aonde os pacotes vão sendo armazenados na ordem em que eles chegam, e assim que possível são enviados nesta mesma ordem. Ainda foram realizadas outras configurações como comprimento máximo da fila de 10 pacotes e tempo de simulação de 400 segundos.

Conforme apresentado na seção 2.1, não existe um consenso sobre qual é o melhor protocolo de roteamento, por isso optou-se por realizar simulações com

protocolos tanto pró-ativos (*table-driven*) como reativos (*on-demand*). Como parte dos objetivos deste trabalho era verificar a influência nas métricas de QoS causada pela adoção de protocolos de roteamento seguro, outros protocolos com tais características também foram analisados. A tabela 2 apresenta os protocolos que foram simulados.

Table 2. Protocolos simulados

	Pró-ativo	Reativo
Sem segurança	**DSDV**	**DSR**
Com segurança	**SEAD**	***Ariadne***

Como discutido anteriormente, o protocolo SEAD é baseado no DSDV e o *Ariadne* no DSR. O código fonte destes protocolos (SEAD e *Ariadne*) é originário do Projeto Monarch [Monarch Project, 2004], desenvolvido na CMU (*Carnegie Mellon University*) e foi disponibilizada pelos autores de [Yovanof and Erikci, 2004].

4.2. Resultados e Análises

Quanto à vazão podemos observar que os protocolos DSR, DSDV e SEAD apresentaram um comportamento semelhante para os pacotes de 64 bytes, com uma pequena vantagem para o protocolo DSR. Já o protocolo *Ariadne* teve uma vazão maior do que os outros protocolos. No experimento com pacotes de 256 bytes tivemos um desempenho semelhante entre todos os protocolos quando a rede foi composta por 5, 10 e 15 nodos. No entanto, quando aumentamos o tamanho da rede para 20 nós, o protocolo SEAD apresentou uma melhor performance em relação a esta métrica. Em média todos os protocolos apresentaram um crescimento linear ao aumentarmos o número de nós presentes na rede. O gráfico 1 apresenta os valores de vazão para cada um dos protocolos de roteamento analisados, para pacotes de 64 e 256 bytes.

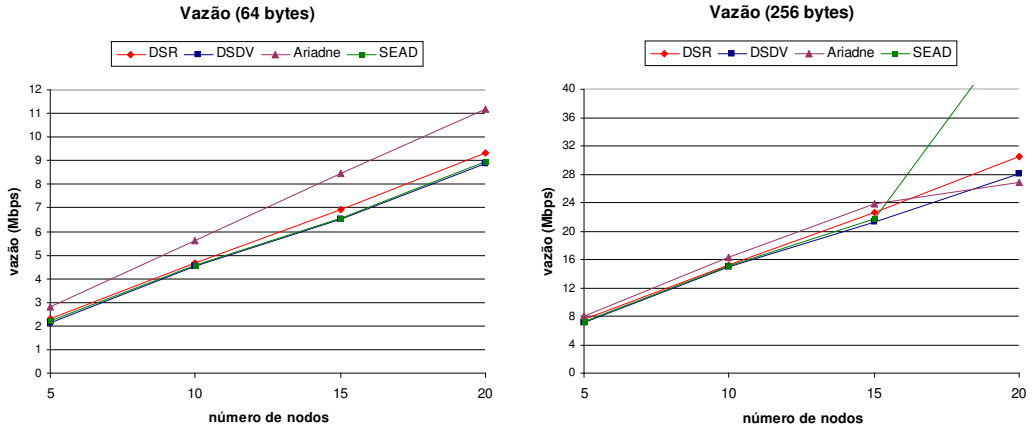


Gráfico 1. Vazão x Protocolo de Roteamento (pacotes 64 e 256 bytes)

A análise da latência é importante, pois quanto menor, melhor o tempo de resposta da rede. Caso a latência seja muito grande torna-se difícil o uso de interatividade na rede. Este é um fator que deve ser levando em consideração principalmente nas transmissões de áudio e vídeo. O gráfico 2 refere-se aos valores médios de latência para cada um dos protocolos de roteamento analisados.

Observa-se que o protocolo DSR é o que possui a melhor performance tanto para pacotes de 64 bytes como para pacotes de 256 bytes. Em segundo lugar aparece o protocolo SEAD, no entanto quando aumentamos a taxa de tráfego (20 nodos e pacotes de 256 bytes) a performance deste protocolo diminui drasticamente. Esta queda de desempenho também acontece com o protocolo *Ariadne*, sendo que este

comportamento também foi observado em outros experimentos realizados em [Yovanof and Erikci, 2004].

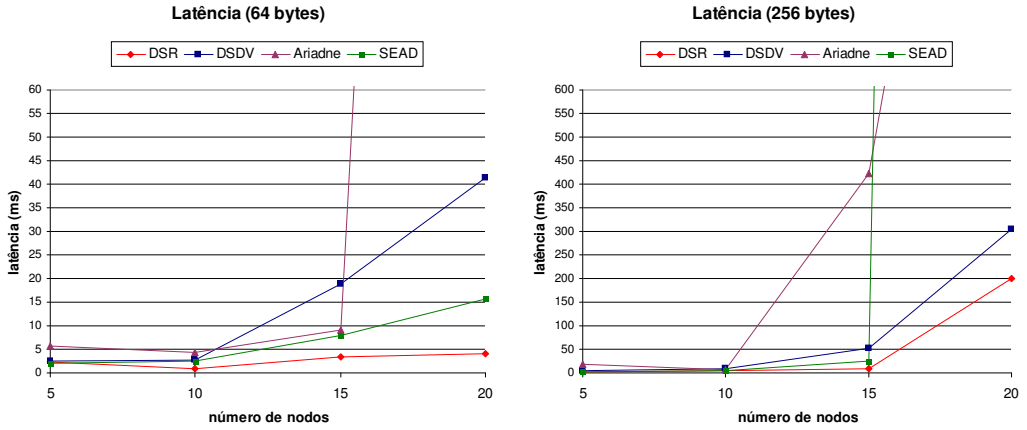


Gráfico 2. Latência x Protocolo de Roteamento (pacotes 64 e 256 bytes)

O principal fator que faz com que a performance do protocolo *Ariadne* seja inferior é o significativo aumento do *overhead* de roteamento quando aumentamos a taxa de tráfego, congestionando a rede. Já no protocolo SEAD o *overhead* permanece constante, pois mensagens de roteamento são enviadas periodicamente e não depende da taxa de tráfego.

O protocolo base do SEAD possui uma pequena diferença em alguns pontos em relação ao protocolo DSDV original [Yih-Chun, 2002b], como por exemplo na forma de envio das mensagens de atualização de rotas, isto pode ser um fator que faz com que em alguns casos o seu desempenho seja superior ao DSDV. Em outras simulações [Yovanof and Erikci, 2004] este comportamento não se verifica.

A métrica *jitter* é calculada como sendo a variação do atraso (latência), sendo assim, a sua análise nos permite observar se uma rede apresenta um comportamento constante ou se possui muitas variações. O gráfico 3 apresenta os valore médios de *jitter* para cada um dos protocolos de roteamento analisados.

O desempenho dos protocolos em relação a esta métrica foi semelhante ao apresentado em relação à latência. O protocolo DSR foi o que apresentou o melhor desempenho, seguido pelo SEAD. Novamente o protocolo *Ariadne* e, com menos intensidade, o protocolo SEAD apresentam problemas com o aumento do tráfego na rede.

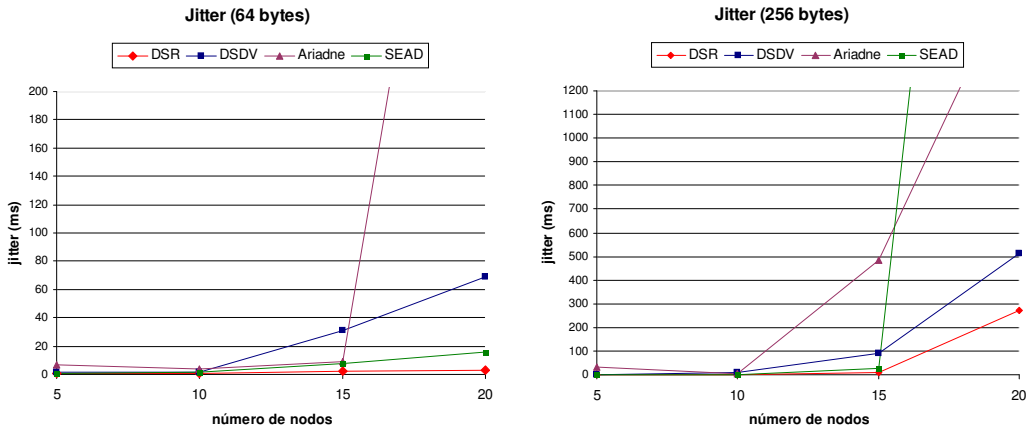


Gráfico 3. *Jitter* x Protocolo de Roteamento (pacotes 64 e 256 bytes)

O protocolo DSDV, por sua vez, apresentou um desempenho intermediário e com o aumento do tamanho da rede e dos pacotes sua performance melhorou em relação aos outros protocolos. As explicações para tais comportamentos são as mesmas que explicam o desempenho de cada protocolo em relação à latência.

Através da análise da métrica perda de pacotes, podemos verificar se as filas localizadas em cada nó da rede estiveram sobrecarregadas. Outro fator que pode determinar a perda de pacotes é a ineficiência no roteamento dos pacotes, estando mais ligado ao protocolo utilizado no roteamento das mensagens. O gráfico 4 apresenta os percentuais de perda de pacotes para cada um dos protocolos de roteamento analisados.

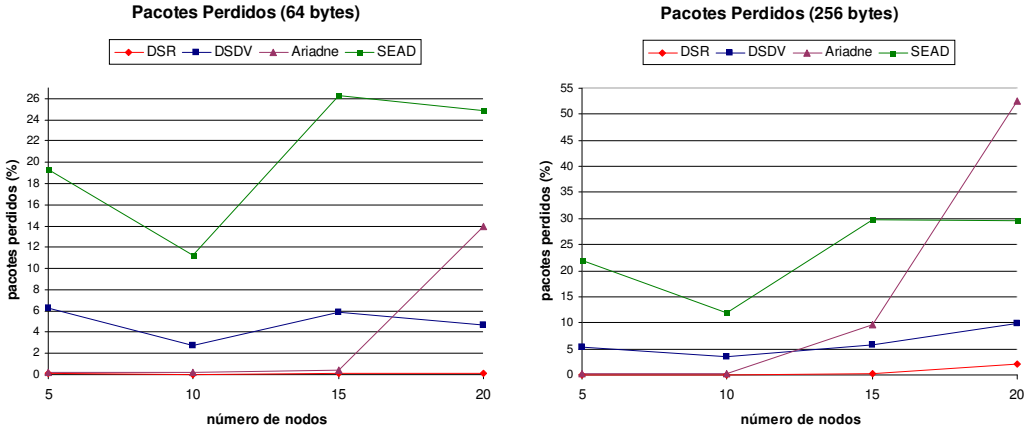


Gráfico 4. Pacotes Perdidos x Protocolo de Roteamento (pacotes 64 e 256 bytes)

O protocolo DSR foi o que apresentou o melhor desempenho, seguido pelo *Ariadne*, DSDV e SEAD, respectivamente. No entanto, para pacotes de 64 bytes, quando aumentamos o tamanho da rede para 20 nodos, o protocolo *Ariadne* mostrou uma performance inferior as DSDV. Para pacotes de 256 bytes, este fator é observado ainda com a rede num tamanho de 15 nodos e quando aumentamos para 20 nodos este protocolo foi o que apresentou o pior desempenho.

Os protocolos reativos DSR e *Ariadne*, quando simulados com a rede formada por 5 e 10 nodos, apresentaram uma alta performance em relação a esta métrica, pois a rede é pequena e com isso conseguem estabelecer as rotas com facilidade. Já os protocolos pró-ativos (DSDV e SEAD) mostraram um desempenho inferior, principalmente o SEAD.

5. Trabalhos Relacionados

Alguns trabalhos científicos já foram desenvolvidos e publicados na área de redes sem fio do tipo *Ad Hoc* considerando questões como QoS e segurança. Nestes trabalhos são realizadas novas propostas, assim como a análise das já existentes, para garantir níveis aceitáveis de QoS e segurança nestas redes. Dentre os trabalhos utilizados como inspiração e fonte de conhecimento para este trabalho, destacam-se os mais relevantes:

- *Performance Evaluation of Security-Aware Routing Protocols for Clustered Mobile Ad Hoc Networks* [Yovanof and Erikci, 2004]: este artigo explora a eficácia de protocolos de roteamento seguro para o suporte na transmissão de dados multimídia em tempo real em redes sem fio *Ad Hoc* composta por *clusters*. Dois protocolos de roteamento seguro são testados, SEAD e *Ariadne*, em um ambiente simulado e o desempenho deles é comparado considerando várias condições de tráfego e topologias de rede.

- *Network Mobility and Protocol Interoperability in Ad Hoc Networks* [DaSilva, 2004]: apresenta o projeto e implementação de um novo protocolo para redes *Ad Hoc*, um conjunto de soluções para administração de redes baseado em políticas, e discussões para o gerenciamento de chaves e desenvolvimento de *IPsec* em uma MANET. A eficiência do sistema é avaliada através de experimentos em uma rede sem fio *Ad Hoc*.
- Análise de Desempenho em Redes *Wireless Ad Hoc* e Estabelecimento de um Acordo de Nível de Serviço Pró-Ativo [Pereira, 2003]: apresenta um estudo das quatro métricas básicas de QoS (latência, *jitter*, vazão e perda de pacotes) em redes sem fio do tipo *Ad Hoc*, através de simulações utilizando o NS-2. Neste trabalho é proposto, com base nos resultados das simulações, um SLA pró-ativo, onde as rotinas deverão prever uma possível quebra do acordo e com isso tomar alguma decisão.
- ARIADNE: *A secure On-Demand Routing Protocol for Ad Hoc Networks* [Yih-Chun et al., 2002a]: neste artigo são apresentados os tipos de ataques possíveis em um sistema, e uma descrição dos ataques nos protocolos de roteamento em redes Ad Hoc. É apresentado também o projeto e a avaliação de desempenho de um novo protocolo *on-demand* para o roteamento seguro, chamado *Ariadne*.
- SEAD: *Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks* [Yih-Chun et al., 2002b]: neste artigo é apresentado o projeto e a avaliação de um novo protocolo do tipo *table-driven* de roteamento seguro chamado SEAD, o qual é baseado no protocolo DSDV.

6. Conclusões e Trabalhos Futuros

As redes sem fio *Ad Hoc* estão sendo cada vez mais usadas nos mais variados ambientes, principalmente em locais onde não exista uma infra-estrutura de rede fixa ou o custo para sua construção é muito alto. Devido a algumas características indesejáveis, como baixa taxa de vazão e problemas de segurança, este tipo de rede não era muito utilizado. Porém, através de muitas pesquisas e estudos, estes fatores, que limitavam o uso, começaram a ser sanados ou minimizados e a tendência é que a tecnologia sem fio seja cada vez seja mais aceita e utilizada.

Com a evolução das comunicações e o crescimento na quantidade e na variedade de aplicações, surgiram aquelas que necessitam de garantias em relação a comunicação. Desta forma, para estas aplicações, torna-se necessário o uso de SLAs, que quando estabelecido de uma forma correta são ferramentas poderosas de gerência. Sendo assim, este trabalho teve como objetivo obter informações sobre o desempenho de redes sem fio *Ad Hoc* e analisá-las para poder moldar as métricas de QoS, as quais são usadas no estabelecimento dos SLAs.

Após a análise dos resultados obtidos com as simulações, a primeira conclusão que se chega é que, para estas simulações, o protocolo que apresentou a melhor performance foi o DSR. No entanto, vale lembrar que os protocolos *Ariadne* e SEAD apresentam mecanismos de segurança, o que faz com que, em alguns casos, os seus desempenhos diminuam. Em vista disto, vários fatores devem ser analisados na escolha das configurações para uma rede sem fio *Ad Hoc*, como por exemplo a necessidade de segurança nas comunicações.

Como já era esperado, ao adicionar mecanismos de segurança nos protocolos, os seus desempenhos tendem a degradar. No entanto, estes protocolos são importantes por exercerem funções de segurança no roteamento dos pacotes. Um ponto que merece destaque é o comportamento destes protocolos com o aumento do tráfego na rede, pois

quando isto acontece o desempenho dos mesmos diminui drasticamente. Esta queda no desempenho é mais acentuada no protocolo *Ariadne*, pois por se tratar de um protocolo reativo, com o aumento da taxa de tráfego na rede ocorre um significativo aumento no *overhead* de roteamento, congestionando a rede. A diferença observada na performance entre os protocolos reativos (DSR e *Ariadne*) foi maior do que a observada entre os protocolos pró-ativos (DSDV e SEAD). Desta forma, podemos afirmar que nestas simulações o impacto causado pela adição de mecanismos de segurança foi maior no protocolo DSR (DSR → *Ariadne*) do que no protocolo DSDV (DSDV → SEAD).

Em relação ao estabelecimento de um SLA, deverá ficar claro, para o cliente e principalmente para o provedor de serviços, que a utilização de protocolos de roteamento seguro implicará em valores mais altos para as métricas de latência e *jitter*. Como conseqüência disto, a rede apresentará um aumento no tempo de resposta e uma diminuição da uniformidade no envio de pacotes, prejudicando o desempenho da comunicação entre os nodos.

A continuidade deste trabalho poderia ser realizada através da avaliação de outros protocolos de roteamento (tanto seguros como não seguros). Além disso, poderia ser analisado o impacto causado no valor das métricas de QoS ao se utilizar diferentes padrões de mobilidade. Desta forma, poderia ser observado qual protocolo de roteamento seria mais eficiente em relação a este fator, pois com o aumento da mobilidade a tendência é que os valores das métricas de QoS degradem. Outro trabalho interessante a ser desenvolvido seria analisar o desempenho dos protocolos de roteamento seguro em uma rede *Ad Hoc* quando um dos nodos fosse malicioso. O esperado é que a performance destes protocolos seja menor do que as apresentadas neste trabalho.

7. Referências

Agrawal, D. P. and Cordeiro, C. M. (2004) "Mobile Ad hoc Networking", capítulo 3, OBR Research Center for Distributed and Mobile Computing, ECECS University of Cincinnati, Cincinnati, OH 45221-0030 – USA.

Corson, S. (1999) "RFC 2501: Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", Janeiro 1999.

DaSilva, L. A., Midkiff, S. F., Park, J. S., Hadjichristofi, G. C. and Davis, N. J. (2004) "*Network Mobility and Protocol Interoperability in Ad Hoc Networks* ", IEEE Communications Magazine, p. 88-96, Nov. 2004.

He, G. (2003) "Destination-Sequenced Distance Vector (DSDV) Protocol", Networking Laboratory, Helsinki University of Technology. Disponível em: < http://citeseer.ist.psu.edu/531710.html >. Acesso em: 20 outubro 2004.

Johnson, D. B., Maltz, D. A. and Broch, J. (2001) "DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks". In C. Perkins, editor, Ad Hoc Networking, chapter 5, pages 139--172. Addison-Wesley, 2001.

Monarch Project (2004) "Wireless and Mobility Extensions to ns-2", Disponível em: <http://www.monarch.cs.cmu.edu/cmu-ns.html>. Acesso em 7 Outubro de 2004.

Pereira, M. C. (2003) "Análise de Desempenho em Redes Wireless Ad Hoc e Estabelecimento de um Acordo de Nível de Serviço Pró-Ativo", 145f. Dissertação de Mestrado submetida à Universidade Federal de Santa Catarina. UFSC, 2003.

Royer, E. M. and Toh, C-k. (1999) "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", IEEE Personal Communications, Apr. 1999, Disponível em: < http://citeseer.ist.psu.edu/royer99review.html>. Acesso em: 22 outubro 2004.

Yih-Chun, H., Perrig, A. and Johnson, D. B. (2002a) "ARIADNE: A secure On-Demand Routing Protocol for Ad Hoc Networks", in proceeding of MOBICOM 2002.

Yih-Chun, H., Johnson, D. B. and Perrig, A. (2002b) "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", in the 4th IEEE Workshop on Mobile Computing Systems and Applications, 2002.

Yovanof, G. S. and Erikci, K. (2004) "Performance Evaluation of Security-Aware Routing Protocols for Clustered Mobile Ad Hoc Networks", International Workshop on Wireless Ad-Hoc Networks, 2004.