Rudolfo Augusto Ersching Rüncos

Desempenho do SCTP em terminais multi-abrigados para transporte de tráfego VoIP em cenários com perdas

Curitiba/PR

fevereiro 2011

Rudolfo Augusto Ersching Rüncos

Desempenho do SCTP em terminais multi-abrigados para transporte de tráfego VoIP em cenários com perdas

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Engenharia Elétrica. Programa de Pós-Graduação em Engenharia Elétrica, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná.

Orientador: Prof. Dr. Eduardo Parente Ribeiro

Curitiba/PR

fevereiro 2011

RESUMO

A multiplicidade de meios de acesso e sua popularização estimula a exploração de sistemas multi-abrigados, que são sistemas com mais de uma porta disponível para acessar a internet. Esta característica pode trazer ganhos à resiliência da conexão e à qualidade de serviços. Serviços multimídia podem ser beneficiados pelo uso mais adequado das rotas disponíveis em sistemas multi-abrigados. O SCTP (*Stream Control Transmission Protocol*) é um protocolo de transporte que suporta conexões multi-abrigadas, e por esse motivo vem sendo alvo de estudos envolvendo otimização do uso de caminhos para melhoria de qualidade de serviços multimídia.

Este trabalho estuda o desempenho do SCTP juntamente com um mecanismo de troca automática de rotas baseado em atraso (*delay-centric*) para o transporte de tráfego VoIP (*Voice over IP*) em terminais multi-abrigados. Os cenários simulados envolvem atrasos e perdas variáveis. Dois parâmetros que apresentam forte influência sobre o comportamento do SCTP, o *PMR* (*Path.Max.Retrans*) e o *RTOmax* (Valor máximo do *Retransmission TimeOut*), têm seus valores variados com o objetivo de investigar sua influência na qualidade de chamadas VoIP. A métrica de desempenho utilizada é o MOS (*Mean Opinion Score*) estimado a partir do E-model. Os resultados indicam que abordagens mais agressivas, representadas por menores valores para os parâmetros estudados, proporcionam uma melhor qualidade nas chamadas VoIP.

Palavras chave: SCTP, multi-abrigado, delay-centric, VoIP

ABSTRACT

The increasing popularity of different media access technologies stimulates the exploration of multi-homed systems, which can access the internet via two or more gateways. This can provide services with both better resilience and quality. Multimedia services can benefit from a more adequate use of available routes in a multi-homed system. SCTP's (Stream Control Transmission Protocol) multi-homing support has made it a target for studies on enhancing quality of multimedia services through automatic path switching.

This work studies the performance of SCTP along with an automatic path switching mechanism based on delay (delay-centric) for VoIP (Voice over IP) transmissions in a multi-homed host. Simulated scenarios present variable delays and losses. The values of two parameters with strong influence over SCTP's behavior, *PMR* (Path.Max.Retrans) and *RTOmax* (Maximum Retransmission TimeOut value), are spanned in order to evaluate their influence in VoIP call quality. Performance is measured using MOS (Mean Opinion Score) estimated by the E-model. Results indicate that a more agressive aproach, meaning lower values for both parameters under study here, deliver a better quality for VoIP calls.

Key words: SCTP, multi-abrigado, delay-centric, VoIP

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACK	*Acknowledgement*, refere-se a um pacote que o receptor envia em resposta a um pacote recebido.
CBR	*Constant Bit Rate* (Taxa de Bits Constante)
C++	Linguagem de programação de alto nível de uso geral
E-model	Modelo computacional que busca prever o efeito subjetivo em função de medidas objetivas de diversos fatores em uma transmissão de voz.
G.107	Recomendação do ITU-T que padroniza o E-model.
G.711	Codec VoIP de taxa de bits constante.
IC	Intervalo de confiança
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
ITU-T	*International Telecommunication Union - Telecommunication Standardization Sector*
M/D/1	Notação de Kendall para um sistema de filas contendo um único servidor, tempo de serviço determinístico e intervalo entre chegadas de clientes aleatório seguindo uma distribuição de probabilidade exponencial.
MOS	*Mean Opinion Score*. Uma medida subjetiva de qualidade de chamadas telefônicas.
NS2	*Network Simulator 2*, um simulador de redes de computadores de eventos discretos.
Otcl	*Object oriented Tool Command Language* (Linguagem de Comandos de Ferramentas orientada a objetos)
PLC	*Packet Loss Concealing* (encobrimento de perdas de pacotes).
PMR	*Path.Max.Retrans*. Número máximo de perdas tolerado pelo SCTP antes do estabelecimento de inatividade do caminho.
PSI	Provedor de Servido de Internet
QoS	*Quality of Service* (qualidade de serviço)

RNG	*Random Number Generator* (Gerador de números aleatórios).
RTO	*Retransmission TimeOut*. Tempo limite esperado pelo SCTP antes de realizar uma retransmissão.
RTOini	Valor inicial do *RTO*.
RTOmax	Limite superior do parâmetro *RTO*.
RTOmin	Limite inferior do parâmetro *RTO*.
RTT	*Round Trip Time* ou tempo de ida e volta, que é o intervalo de tempo que leva desde a geração de um pacote até o recebimento de seu ACK.
RTTVAR	Uma medida da variação do *RTT*.
SACK	*Selective Acknowledgement* (ACK seletivo).
SCTP	*Stream Control Transmission Protocol*
SRTT	*Smooth Round Trip Time* é uma média suavizada do *RTT*.
TCP	*Transmission Control Protocol*, um protocolo de transporte orientado à conexão.
UDP	*User Datagram Protocol*, um protocolo de tranporte não orientado à conexão
VoIP	*Voice over Internet Protocol* (Voz sobre IP)

1 INTRODUÇÃO

As constantes inovações em tecnologias de transmissão de dados têm aumentado a gama de possibilidades de meios de acessos. São cada vez mais comuns dispositivos capazes de se conectar simultaneamente a mais de uma rede, como WiFi, 3G e WiMax. Geralmente cada meio de acesso se conecta à rede utilizando endereços IP distintos e possivelmente através de PSIs (Provedores de Serviço de Internet) diferentes, caracterizando assim um usuário multi-abrigado.

Um usuário ou sistema é multi-abrigado quando ele possui mais de uma interface de rede, física ou lógica, disponível para comunicação. Através de mecanismos de monitoramento e de troca de rotas, é possível aumentar a resiliência e estabilidade de uma conexão, garantindo a continuidade dos serviços mesmo no caso de alguma rota se tornar indisponível. Para dispositivos móveis isso pode ser utilizado como um mecanismo alternativo para realizar a troca de estações base à medida que as áreas de convergência vão mudando.

Concomitantemente, serviços de multimídia em tempo real, tais como VoIP (*Voice over Internet Protocol*), têm aumentado sua demanda. Esse tipo de serviço exige um baixo atraso no envio dos dados. Em sistemas multi-abrigados a utilização de mecanismos que monitorem o atraso das rotas e escolham a de menor atraso em tempo real pode trazer um aumento de qualidade para esses serviços. Esse tipo de mecanismo já está sendo estudado por diversos autores [1] [2] [3] e é comumente chamado de *delay-centric*. A partir deste ponto nesta dissertação esse mecanismo de troca automática de rotas baseado no atraso será referido apenas como *delay-centric*.

A camada mais propícia para a implementação de mecanismos que envolvam trocas de rotas ou estações base é a camada de transporte [4], pois ela é a camada de mais baixo nível que apresenta conhecimento sobre as condições das rotas. Além disso, implementar a troca de rotas na camada de transporte não exige mudanças no núcleo da rede, apenas nos pontos finais (terminais), o que torna essa solução mais viável economicamente. No entanto, o protocolo de transporte mais largamente utilizado por aplicações em tempo real, o UDP, não dispõe de

recursos adequados para se implementar esse tipo de solução.

O desenvolvimento de protocolos de transporte e mecanismos que suportem terminais multi-abrigados e permitam troca automática de interfaces de saída pode beneficiar tanto usuários de serviços multimídia como a própria rede. O benefício para os usuários terminais é evidente, pois utilizar o caminho que apresente as melhores condições de tráfego dentre os disponíveis resultará na melhor QoS possível para este usuário. Como os caminhos que apresentam boas condições de tráfego são usualmente aqueles que estão pouco utilizados, fazer com que o tráfego flua por esses caminhos ao invés de caminhos mais congestionados irá garantir um uso mais eficiente da rede, o que é uma vantagem para a própria rede.

Um protocolo que atualmente está ganhando popularidade e que apresenta suporte para conexões entre terminais multi-abrigados é o SCTP (*Stream Control Transmission Protocol*) [5]. Esse protocolo vem sendo utilizado largamente em estudos envolvendo troca de rotas ou pontos de acesso e melhoria da qualidade de serviços entre terminais multi-abrigados. O SCTP padrão também tem implementado um mecanismo de detecção de falhas sensível a perdas que realiza troca de rotas, além de diversos parâmetros que regulam seu funcionamento.

O emprego do SCTP e do *delay-centric* para aplicações multimídia ainda não foi suficientemente estudado. Foi pouco explorado o comportamento do SCTP com *delay-centric* em cenários com perdas, nos quais tanto o *delay-centric* quanto o mecanismo de detecção de falhas atuariam simultaneamente. Além disso, a otimização dos parâmetros do SCTP para transporte de tráfego multimídia é outra questão importante que necessita de mais estudos para ser alcançada. Esses assuntos são abordados no presente trabalho.

1.1 Objetivos

O objetivo deste trabalho é investigar o comportamento do SCTP com e sem o algoritmo do *delay-centric* em relação à qualidade do tráfego VoIP em diversas condições de tráfego de fundo e perdas. Alguns parâmetros que regulam o comportamento do SCTP são variados no intuito de investigar sua influência sobre a qualidade das chamadas VoIP. Um dos parâmetros estudados é o *PMR* (*Path.Max.Retrans*) que corresponde basicamente a um limiar de sensibilidade a perdas do mecanismo de detecção de falhas padrão do SCTP. Outro parâmetro que apresenta forte influência no mecanismo de detecção de falhas, de modo geral determinando os tempos de reação do SCTP, é o *RTO* (*Retransmission TimeOut*). Este parâmetro é calculado em tempo real durante uma associação, não podendo, portanto, ser configurado no início de uma associação SCTP. Mas, em compensação, seus limites tanto inferior (*RTOmin*) como superior

(*RTOmax*) podem e são variados neste trabalho. A métrica de qualidade adotada é o MOS (*Mean Opinion Score*) estimado a partir do E-model, um modelo computacional que busca prever a qualidade percebida pelo usuário em função de medidas objetivas de diversos fatores em uma transmissão de voz.

Neste trabalho são realizadas simulações que utilizam o protocolo SCTP como agente de transporte para tráfegos VoIP em terminais multi-abrigados. O SCTP é configurado para tráfego não-confiável, de modo que não há retransmissões de pacotes perdidos. O codec VoIP simulado é o G.711 com mecanismo de encobrimento de perdas de pacotes (*Packet Loss Concealing* ou PLC). Também é utilizada a implementação do algoritmo *delay-centric* para o NS2 (*Network Simulator 2*) feita por Gavriloff [3].

Todos os cenários simulados envolvem tráfegos de fundo, no intuito de gerar atrasos variáveis controlados, e perdas em rajadas em qualquer rota. Tanto o atraso médio como as perdas médias podem variar sua intensidade entre as diferentes rotas em cada simulação, e várias combinações de atraso e perdas médios são simulados. Dessa forma, tanto o mecanismo de detecção de falhas padrão do SCTP como o mecanismo do *delay-centric* podem atuar.

Ao final do trabalho são indicados os valores mais adequados para os parâmetros estudados no sentido de se obter a melhor qualidade de transmissão para o tráfego VoIP, além de idéias de trabalhos futuros baseadas nos resultados obtidos.

1.2 Trabalhos Relacionados

Kelly et al. [1] estudou o uso do SCTP para realização de trocas de rotas em redes sem fio. Ele propôs o algoritmo *delay-centric* para a tomada de decisão de troca de rotas baseada no atraso. Apesar de seus testes não envolverem cenários com perdas, o autor afirma que é viável a utilização do *delay-centric* no SCTP para aplicações em tempo real em terminais móveis. Noonan et al. [6] realizou mais simulações envolvendo o uso do *delay-centric* em cenários sem fio e concluiu, apesar da simplicidade destes, que aquele traz benefícios à qualidade de serviços sensíveis ao atraso transportados pelo SCTP.

Gavriloff [3] ampliou os estudos sobre o *delay-centric*, implementando sua versão de *delay-centric* para o módulo do SCTP no software de simulação de redes NS2. Suas simulações envolviam situações com variadas condições de tráfego de fundo, mas sem a presença de perdas. Nessas condições, o autor demonstra que o *delay-centric* traz grandes benefícios ao tráfego VoIP, usando como métrica de desempenho o MOS estimado através do E-model.

Fracchia et al. [7] propôs um novo método denominado WiSE para estimação de congestionamento e falhas de transmissão em cenários sem fio com perdas em rajadas. Seu enfoque, diferentemente de Kelly et al. [1], foi a transferência volumosa de dados. Através do uso de técnicas de estimação de largura de banda o autor afirma que o método permite um maior *throughput* com uma maior robustez a falhas, além de ser compatível com o TCP (*Transmission Control Protocol*).

Para levar em conta tanto atraso como perdas em transmissões de tráfegos multimídia, Fitzpatrick et al. [8] propôs o uso de um método de troca de rotas que ele denominou ECHO (*Endpoint Centric Handover*). Esse método tem como parâmetro de decisão a qualidade do tráfego VoIP, estimada a partir do E-model. Através do ECHO, tanto o atraso como as perdas são levados em conta indiretamente na tomada de decisão pelo melhor caminho.

Kashihara et al. [9] desenvolveu um método para troca de caminho principal numa associação SCTP visando melhoria de qualidade para transmissão de comunicações em tempo real. Seu método é baseado em regras para a realização de troca de rotas utilizando medições de largura de banda. Esse método foi desenvolvido para trazer melhoria de transmissão em cenários sem fio multi-abrigados cujos enlaces apresentam diferentes características quanto a atraso, largura de banda e perdas. Funasaka et al. [10] propôs um aprimoramento a esse método através da medição de atraso e otimização da estratégia de troca de rotas, realizando trocas apenas quando isso representar uma real vantagem ao tráfego. O autor afirma que o método é eficaz para situações em que a rede não apresenta mudanças rápidas de banda disponível ou atraso.

Estudos analíticos mostram que o mecanismo de detecção de falhas padrão do SCTP não realiza uma boa estimativa do tempo de falha [11]. No entanto, a escolha adequada dos parâmetros do SCTP pode melhorar sua performance em cenários com perdas. Qiao et al. [12] sugere baixos valores de *PMR* (0 ou 1) para melhorar a velocidade de transferência de dados volumosa. Caro et al. [13] chega à mesma conclusão e ainda propõe uma política alternativa de retransmissão. Apesar do aumento de falsos-positivos para falhas causado por valores mais baixos de *PMR*, o aumento na velocidade de reação a falhas de caminho compensa e garante um ganho do *throughput*.

Outro parâmetro que tem grande influência sobre o comportamento do SCTP em relação a falhas de caminho é seu parâmetro *RTOmax*. Fallon et al. [14] demonstra o bloqueio de transmissão causado em transmissões com perdas devido ao crescimento exagerado do *RTO*. Esse comportamento do SCTP, que a autora chama de contra-intuitivo, pode ser minimizado utilizando-se um valor menor de *RTOmax*, limitando assim o crescimento do *RTO*. Em outro

trabalho [15] a mesma autora propõe um novo método (AORAN - Adaptive Optimized RTO algorithm for wireless Access Networks) para calcular o *RTO* e evitar os problemas causados por seu aumento excessivo em redes sem fio.

Os trabalhos apresentados aqui não esgotam o assunto do transporte de tráfego VoIP através do SCTP. Apesar de servirem como base para o presente trabalho, nenhum deles apresenta a mesma abordagem que é proposta neste, que é o emprego do SCTP para o transporte de chamadas VoIP em cenários com perdas utilizando o MOS como métrica de desempenho. Outro aspecto que diferencia este trabalho é a abordagem estatística do desempenho através do uso das médias e intervalos de confiança de inúmeras simulações. Portanto, não é possível uma comparação direta dos resultados obtidos nos trabalhos citados com este.

1.3 Estrutura da dissertação

Esta dissertação contem cinco capítulos, além das referências bibliográficas e anexos. O primeiro capítulo traz uma contextualização do assunto, uma descrição sucinta sobre o estado da arte, alguns trabalhos desenvolvidos por outros pesquisadores acerca do mesmo assunto desta dissertação e os objetivos deste trabalho.

Alguns conceitos fundamentais para o entendimento do que foi estudado aqui são apresentados no capítulo 2. Nele são feitas breves revisões sobre o SCTP, o algoritmo *delay-centric*, o modelo de filas M/D/1, o modelo de Gilbert, o E-model, o MOS e sobre o *software* de simulação utilizado neste trabalho, o NS2.

O capítulo 3 detalha os cenários simulados e explica alguns procedimentos de cálculo utilizados para a obtenção de parâmetros de simulação e também dos resultados, que são apresentados no capítulo 4.

No capítulo 5, são apresentadas as conclusões mais relevantes acerca dos resultados obtidos. A dissertação é encerrada com a lista de referências bibliográficas e três anexos. O anexo A é uma tabela com valores padrão para todos os parâmetros envolvidos no cálculo do fator *R* do E-model. O anexo B é o *script* Otcl utilizado nas simulações do cenário 2 e o anexo C é o programa em C++ utilizado para realizar o cálculo do MOS das simulações do cenário 2.

2 REVISÃO DE CONCEITOS

2.1 SCTP

O SCTP, tal como o TCP, é um protocolo de transporte orientado a conexão. A motivação para o desenvolvimento desse novo protocolo veio da necessidade por um protocolo de transporte mais apropriado do que o TCP para aplicações de sinalização em telefonia. Em outubro de 2000 a IETF lançou a RFC2960 [5], primeiro documento que padronizou o SCTP. Posteriormente, outros documentos extenderam as funcionalidades deste protocolo [16] [17].

Assim como o TCP, o SCTP apresenta controle de congestionamento e permite a entrega confiável de dados. Além disso, ele permite transmissão ordenada ou não ordenada de dados, fornece serviço de fluxos múltiplos (*multi-streaming*) e, devido à extensões [17], permite também entrega não confiável de dados, semelhante ao UDP, ou mesmo com confiabilidade parcial.

Uma conexão SCTP entre dois terminais é denominada de associação. Uma mesma associação SCTP permite o uso de mais de um endereço IP para comunicação, o que significa a possibilidade do uso de mais de uma rota para comunicação. Ou seja, o SCTP fornece suporte para associações multi-abrigadas. Essa capacidade do SCTP é o que torna este protocolo uma excelente alternativa para a implementação de sistemas automáticos de troca de rotas. Além disso, ele tem por padrão implementado um mecanismo que realiza a troca de rotas quando ocorrem falhas, de forma a aumentar a resiliência da associação.

Ao iniciar uma associação entre dois terminais multi-abrigados, um dos endereços IP de cada terminal envolvido na associação é escolhido como primário. Assim, uma rota se torna a primária enquanto as demais tornam-se secundárias. A menos que a rota primária se torne inativa, todo tráfego da associação irá fluir por essa rota. Caso o caminho primário se torne inativo, o SCTP irá enviar os pacotes através de uma rota alternativa (se disponível) até o restabelecimento do caminho primário. Um caminho se torna inativo se ocorrerem sucessivas ausências de recebimento de confirmação de envio de dados, por exemplo, durante uma rajada

de perdas no caminho em questão.

2.1.1 *RTT*, *SRTT* e *RTO*

Para melhor entender o mecanismo de detecção de falhas do SCTP, é necessário antes entender a variável denominada *Retransmition TimeOut* (*RTO*). Cada caminho i apresenta o seu próprio RTO_i. O *RTO* determina o tempo máximo que o SCTP aguarda pela confirmação de recebimento de um pacote antes de retransmiti-lo. No início de uma associação o *RTO* é inicializado com o valor de *RTOini*, que por padrão é de 3 s.

Na seqüência, o valor do *RTO* é atualizado cada vez que uma medida de tempo de ida e volta (*Round Trip Time - RTT*) é realizada para o caminho em questão. O RTT é medido do instante em que um pacote é enviado até o instante em que seu ACK é recebido. Quando a primeira medida RTT_1 de RTT é realizada [18]:

$$SRTT = RTT_1 \tag{2.1}$$

$$RTTVAR = \frac{RTT_1}{2} \tag{2.2}$$

$$RTO = SRTT + 4 \cdot RTTVAR \tag{2.3}$$

onde $SRTT$ é o RTT suavizado e $RTTVAR$ é uma medida de variação do RTT.

A partir da segunda medida de RTT, quando uma nova medida RTT_{nova} é realizada, o *RTO* é atualizado como segue [18]:

$$RTTVAR_{nova} = (1-\beta).RTTVAR_{antiga} + \beta.|SRTT_{antiga} - RTT_{nova}| \tag{2.4}$$

$$SRTT_{nova} = (1-\alpha).SRTT_{antiga} + \alpha.RTT_{nova} \tag{2.5}$$

$$RTO = SRTT_{nova} + 4 \cdot RTTVAR_{nova} \tag{2.6}$$

onde β e α são constantes cujos valores recomendados são $1/4$ e $1/8$, respectivamente.

Existem ainda limites para o valor de *RTO*. Este não pode ser inferior a *RTOmin* e nem superior a *RTOmax*. Por padrão, $RTOmin = 1$ s e $RTOmax = 60$ s.

O RTT é aferido no caminho primário a partir dos pacotes de dados e seus ACKs. Já nos caminhos secundários isso se dá através dos pacotes de pulsação e ACK de pulsação.

2.1.2 Pacotes de pulsação

Durante o tempo em que a associação existir, o SCTP irá avaliar periodicamente a disponibilidade de todas as rotas de forma a manter sempre atualizado o estado (ativa ou inativa) de cada uma delas. No caminho primário isso se dá através dos pacotes de ACK enviados pelo terminal receptor. Já nos caminhos alternativos, essa avaliação é realizada através de pacotes chamados de pulsação. A resposta a esses pacotes, chamada de ACK da pulsação, indica que o caminho se encontra ativo. O intevalo de tempo entre o envio de pacotes de pulsação, H_i, é calculado para cada caminho da seguinte forma [18]:

$$H_i = RTO_i + HBinterval \cdot (1 + \delta) \tag{2.7}$$

Onde RTO_i é o valor atual do *RTO* para o caminho i, δ é um valor aleatório entre $-0{,}5$ e 0,5 e $HBinterval$ é um parâmetro constante cujo valor padrão é 60 s. Toda vez que um pacote de dados ou de pulsação é enviado, um temporizador é inicializado com o último valor calculado de H_i. Caso o temporizador expire, um novo pacote de pulsação é enviado. Isso ocorre também em caminhos inativos. Assim, se um ACK de pulsação é recebido em um caminho inativo, o estado desse caminho será modificado para ativo.

2.1.3 Mecanismo de detecção de falhas

Cada vez que um pacote é enviado, um temporizador com o valor de RTO_i é iniciado. Se o temporizador expirar e não for recebido um ACK do pacote enviado, é dito que ocorreu um estouro de temporização (*timeout*). Nessa situação, o contador de estouros de temporização do caminho em questão é incrementado, o RTO_i é duplicado, a janela de transmissão é reduzida e o pacote é retransmitido (caso o fluxo em questão esteja configurado para transmissão confiável). Esse processo irá se repetir até que um ACK seja recebido ou até que o valor do contador de estouros de temporização seja maior do que o valor do parâmetro *Path.Max.Retrans* (*PMR*), que por padrão é 5.

No primeiro caso, ao receber um ACK o contador de estouros de temporização é

zerado e a transmissão segue normalmente. Já no caso de o valor do contador ser maior do que *PMR*, o estado do caminho é alterado para inativo e um caminho alternativo é escolhido para a transmissão dos dados. Dessa forma, o *PMR* determina a tolerância do mecanismo de detecção de falhas à ausência de confirmações de recebimento de dados. Em outras palavras, ele regula a sensibilidade desse mecanismo. Se o SCTP estiver configurado para transmissão não confiável, ao invés de retransmitir o pacote que não teve ACK o SCTP irá transmitir o pacote seguinte (se houver). Mesmo quando configurado para transporte não confiável, o SCTP continua enviando ACKs ou SACKs para os pacotes recebidos com sucesso (tanto de dados como de pulsação). Dessa forma os mecanismos de controle de congestionamento e de detecção de falhas continuam atuando normalmente.

O *PMR* possui um papel fundamental como parâmetro regulador do mecanismo de detecção de falhas do SCTP. Ele influencia diretamente a velocidade de reação do SCTP para efetuar a troca de rotas quando ocorrem perdas (ou, mais precisamente, ausência de recebimento de ACK). O SCTP irá efetuar uma troca de rotas através desse mecanismo quando o número de perdas (estouros de temporização) subsequentes for maior do que o valor do *PMR*. Dessa forma, valores baixos de *PMR* correspondem a um comportamento mais agressivo do SCTP, ou seja, menor tolerância a perdas e uma troca de rotas mais rápida. Por outro lado, valores altos configuram um comportamento mais permissivo em relação a perdas. O valor padrão do *PMR*, que é de 5 unidades, é um valor considerado alto, ou seja, bastante tolerante a perdas.

2.2 *Delay-centric*

O algoritmo do *delay-centric* foi proposto originalmente por Kelly et al. [1]. Esse mecanismo foi desenvolvido para aumentar a qualidade no transporte de dados sensíveis ao atraso da rede, como no caso do VoIP. Ele funciona em conjunto com o SCTP e se baseia no monitoramento do atraso apresentado pelos caminhos envolvidos na associação e na troca automática para o caminho que apresente um menor atraso, de forma transparente a camadas superiores. Dessa forma o *delay-centric* altera o caminho primário para a rota que apresentar menor atraso dinamicamente.

O monitoramento do atraso é feito pela aferição do $SRTT$ dos diferentes caminhos, através dos pacotes de pulsação nos caminhos alternativos ou do próprio tráfego últil no caso do caminho primário. O valor do $SRTT$ é calculado e fornecido pelo SCTP. Caminhos que apresentem um menor $SRTT$ são redefinidos como primários. Ou seja, o *delay-centric* interfere em tempo real na determinação do caminho primário do SCTP. Isso é feito através de um

comando já existente do SCTP que permite à aplicação determinar qual caminho deve ser usado como primário.

Esse mecanismo pode ser implementado com um parâmetro opcional de histerese. Se utilizada, a histerese faz com que a troca de rotas ocorra quando a diferença entre o $SRTT$ do caminho primário e o $SRTT$ do caminho secundário seja maior do que o valor de histerese. Isso ajuda a evitar uma frequência de trocas de rota excessivamente alta [3], além de compensar o fato de que quando houver a troca de rotas o novo caminho poderá apresentar um aumento do atraso devido ao acréscimo do tráfego SCTP a este caminho.

2.3 O modelo de perdas de Gilbert

Para realizar simulações envolvendo perdas de pacotes fez-se necessária a escolha de um modelo de perdas. O modelo de perdas uniformes é frequentemente utilizado por sua simplicidade, mas ele não representa bem o comportamento das perdas observadas em filas de roteadores ou quando ocorre interferência em sistemas sem fio. Já o modelo de Gilbert para perdas de pacotes representa um bom compromisso entre realismo, facilidade de implementação e custo computacional [19] [20]. Portanto, como o foco deste trabalho não é encontrar o modelo mais preciso para representar as perdas, este foi o modelo escolhido para modelar as perdas.

O modelo de erros de Gilbert consiste de uma cadeia de Markov de 2 estados, um estado "bom" ou sem perdas, e outro estado "ruim" ou com perdas, conforme ilustrado pela figura 2.1. Três parâmetros definem esse modelo, que são $P1$, $P2$ e L. $P1$ é a probabilidade de mudança para o estado ruim, $P2$ é a probabilidade de mudança para o estado bom e L é a probabilidade de perda de pacote durante o estado "ruim". Durante o estado "bom" não há perdas de pacotes.

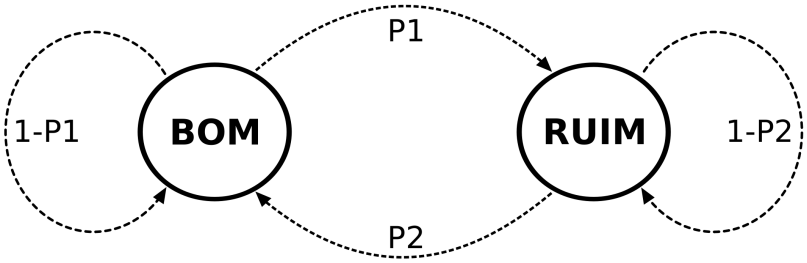

Figura 2.1: Modelo de Gilbert: uma cadeia de Markov de dois estados.

Nesse tipo de modelo, o sorteio para troca de estado deve ser realizado periodicamente. O instante em que o sorteio é realizado pode ser cada vez que um *bit* é gerado, cada vez que

um pacote é gerado, ou a intervalos de tempo constantes (independente da geração de pacotes). Para este trabalho, escolheu-se o último caso, em que o sorteio para troca de estado ocorre a intervalos de tempo constantes Δt, pois assim o custo computacional do modelo é menor do que no caso do intervalo de *bits* e ainda é possível extrair fórmulas simples para os fatores de interesse deste trabalho. Um deles é a perda média (P_{med}) que, supondo que o modelo atue durante um tempo que tenda ao infinito, é dada por:

$$P_{med} = L\left(\frac{P1}{P1+P2}\right) \tag{2.8}$$

Essa estimativa é independente de Δt, que determina a granularidade (resolução) temporal do modelo. Dessa forma, Δt influencia na duração média dos estados. A duração média do estado "bom" (T_{bom}) é dada por:

$$T_{bom} = \frac{\Delta t}{P1} \tag{2.9}$$

e de forma análoga, a duração do estado "ruim" (T_{ruim}) e dada por:

$$T_{ruim} = \frac{\Delta t}{P2} \tag{2.10}$$

2.4 O modelo de filas M/D/1

Existe formação de fila sempre que a taxa de chegada de clientes for maior do que a capacidade do sistema em serví-los. Em uma rede de pacotes, os clientes são os pacotes e os roteadores são os fornecedores de serviço. Num núcleo de rede, o intervalo de tempo entre a chegada de pacotes consecutivos em um roteador é aleatório. O tamanho dos pacotes também varia de forma aleatória. Como qualquer roteador tem uma capacidade limitada de escoamento de pacotes, podem ocorrer momentos nos quais a taxa de entrada é maior do que a taxa de saída. Nesses momentos, haverá a formação de uma fila na saída do roteador, e os pacotes sofrerão um atraso adicional ao esperar serem servidos nessa fila.

Quando se trabalha com simulação de redes de pacotes, é muito comum a necessidade de gerar tráfegos que simulem esse comportamento de um núcleo de rede de maneira controlada. A modelagem do tráfego de pacotes em um núcleo de rede é ainda assunto de pesquisa. Vários modelos já foram propostos [21] [22] [23] [24], mas não se chegou ainda a uma resposta final para essa questão.

Para este trabalho, foi necessário utilizar um modelo de tráfego de fundo que permita controlar a taxa de geração de pacotes, que simule de maneira satisfatória os momentos de pico de congestionamento encontrados em uma rede real e que seja de baixo custo computacional. O modelo escolhido foi o M/D/1, que além de alcançar esses requisitos, é um modelo de fácil implementação.

O modelo de filas M/D/1 (da notação de Kendall) é um modelo no qual o intervalo entre a chegada de pacotes segue uma distribuição exponencial, o tamanho dos pacotes é constante (tempo de serviço determinístico) e existe apenas um servidor, ou seja, os roteadores simulados apresentam apenas uma interface de entrada e uma de saída.

Para este modelo, a taxa de geração (chegada) de pacotes, λ, é [25]:

$$\lambda = \frac{1}{INTVmed} \tag{2.11}$$

onde $INTVmed$ é a média dos intervalos de geração de pacotes (distribuição exponencial). A taxa de escoamento de pacotes (μ) é dada por:

$$\mu = \frac{BW}{pktSIZE} \tag{2.12}$$

onde BW é a taxa de transmissão do enlace de saída do roteador em bits/s e $pktSIZE$ é o tamanho do pacote em bits. $pktSIZE$, BW e, consequentemente, μ são constantes, visto que trata-se do modelo M/D/1. Por sua vez, a ocupação do sistema (ρ) é dada por:

$$\rho = \frac{\lambda}{\mu} \tag{2.13}$$

Assim, segundo Banks [25], o atraso médio sofrido pelos pacotes (W) será:

$$W = \frac{\rho}{2\mu(1-\rho)} \tag{2.14}$$

Como μ é constante, W se torna, em última análise, dependente apenas de $INTVmed$. Esse resultado é utilizado para a implementação do gerador de tráfego de fundo, conforme será explicado adiante.

2.5 MOS e o E-model

A métrica definitiva de desempenho para qualquer ligação telefônica é o MOS (*Mean Opinion Score*) [26], aferido a partir de testes subjetivos realizados em condições adequadas e com um número significativo de indivíduos. Para se obter o MOS, cada indivíduo avalia a qualidade da chamada telefônica com uma nota de 1 a 5, conforme a tabela 2.1. O MOS é então calculado como a média aritmética de todos os indivíduos que avaliaram a chamada, resultando em um valor entre 1 e 5 que indica a qualidade da chamada.

Tabela 2.1: Escala MOS

MOS	classificação	degenerações
5	excelente	imperceptíveis
4	bom	perceptíveis mas não incomodam
3	razoável	incomodam levemente
2	pobre	incomodam
1	ruim	incomodam bastante

Entretanto, essas avaliações subjetivas são bastante custosas de serem realizadas e, para casos nos quais fossem necessárias muitas medições, chegam a ser inviáveis. É para resolver esse tipo de problema que o E-model foi desenvolvido. O E-model é um padrão definido pela ITU-T como G.107 [27] que utiliza várias informações e medições objetivas da rede para calcular um valor escalar de qualidade de chamada telefônica, o fator *R*, da seguinte forma:

$$R = R_0 - I_S - I_D - I_{e,eff} + A \tag{2.15}$$

R_0 : Relação sinal-ruído básica;

I_S : Fator degenerativo simultâneo de codificação;

I_D : Fator degenerativo devido à transmissão, incluindo atraso;

$I_{e,eff}$: Fator degenerativo de equipamento, incluindo perdas de pacotes VoIP [28];

A : Fator de vantagem.

Em relação ao cálculo do fator *R*, este segue os padrões da norma G.107 [27] e SG12 D.106 [28]. Dentre os fatores envolvidos no cálculo, I_D e $I_{e,eff}$ são os únicos que variam neste trabalho. O último é calculado da seguinte forma:

$$I_{e,eff} = I_e + (95 - I_e) \cdot \frac{P_{pl}}{P_{pl} + B_{pl}} \tag{2.16}$$

P_{pl} é a porcentagem de pacotes VoIP perdidos durante a chamada. I_e e B_{pl} são fatores dependentes do codec, sendo o último o fator de robustez a perdas.

O cálculo do fator I_D envolve uma série de etapas que incluem a medida do atraso médio sofrido pelos pacotes em ms (T). Todas as etapas são apresentadas na norma [27] e sua implementação em linguagem C++ consta no anexo C. O anexo A lista todas as variáveis envolvidas no cálculo do fator R com seus valores padrão.

O fator R, por sua vez, pode ser mapeado em valores de MOS de acordo com a seguinte equação:

$$MOS = \begin{cases} 1 & R \leq 0 \\ 1 + 0{,}035R + R(R-60)(100-R) \times 7 \times 10^{-6} & 0 < R < 100 \\ 4.5 & R \geq 100 \end{cases} \tag{2.17}$$

Neste trabalho, o MOS calculado a partir do fator R foi utilizado como métrica de desempenho dos cenários simulados, sendo assim utilizado como parâmetro de comparação. Essa é uma avaliação objetiva que não substitui a avaliação subjetiva do MOS, mas serve como parâmetro de comparação para qualidade de voz na ausência de uma medida subjetiva.

2.6 NS2: *Network Simulator 2*

Existem diversos elementos que formam as redes de computadores, tais como terminais, enlaces, roteadores, filas, entre outros. Esses elementos interagem de forma a transmitir dados encapsulados em pacotes seguindo diversos protocolos e algoritmos tais como protocolos de transporte, controles de congestionamento e regras de roteamento, sem contar ainda a interferência do próprio usuário nesse processo. Todo esse conjunto resulta em um sistema extremamente dinâmico. Em alguns casos, é possível obter modelos grosseiramente aproximados da rede, mas um estudo analítico mais detalhado do comportamento do tráfego de pacotes e de protocolos, mesmo em redes mais simples, é uma tarefa bastante difícil, senão impossível.

Devido a essa dificuldade, é muito comum o emprego de simuladores quando se deseja fazer uma análise mais aprofundada do comportamento de redes de computadores. Portanto, neste trabalho foi utilizado o simulador NS2 (*Network Simulator 2*).

O NS2 é um simulador de eventos discretos em redes de computadores frequentemente

utilizado em pesquisas devido à sua extensibilidade, pois é um software aberto escrito em C++. Mesmo assim, ele tem já implementados diversos elementos e protocolos de redes, permitindo assim a simulação de uma grande variedade de situações em redes tanto cabeadas quanto sem fio. Além disso ele apresenta geradores de tráfego, serviços e outras ferramentas que permitem a elaboração de cenários bastante complexos sem a necessidade de extender o código em C++.

Os *scripts* contendo a descrição da rede, como topologia, agentes, serviços e os parâmetros a serem simulados devem ser escritos na linguagem interpretada Otcl. O NS2 interpreta o *script*, executa a simulação e retorna os resultados, normalmente na forma de um arquivo de *trace* dos pacotes.

Neste trabalho, foi utilizada a versão 2.32 do NS2. Também foi utilizada a versão modificada do SCTP que retorna para o script Otcl os valores de $SRTT$ calculados e a implementação em Otcl do algoritmo do *delay-centric*, ambos conforme Gavriloff [3]. Cabe ressaltar que a modificação feita no SCTP apenas permite obter o valor de $SRTT$, não alterando de nenhuma forma o cálculo deste ou de qualquer outra variável. Portanto, em simulações nas quais o *delay-centric* não estiver atuando, o SCTP se comporta exatamente conforme sua implementação padrão da versão 2.32 do NS2.

3 METODOLOGIA

Inicialmente, foram realizados testes para garantir que o NS2 estava funcionando de acordo com o esperado. Esses testes de verificação envolveram o SCTP no modo de entrega não confiável, seu mecanismo de detecção de falhas e o algoritmo do *delay-centric*. Após a validação, foram desenvolvidos os cenários a serem simulados.

Cenários preliminares foram montados para se avaliar o tempo de processamento e a quantidade mínima de simulações que garantiriam um resultado estatisticamente relevante em relação ao MOS. Isso também serviu para determinar as faixas de valores dos parâmetros a serem utilizadas nas simulações.

Com base nessas simulações preliminares são montados os dois cenários finais de simulação. Cada cenário visa explorar uma questão diferente. O primeiro cenário corresponde ao estudo do *delay-centric* em conjunto com o mecanismo de detecção de falhas padrão do SCTP. O segundo cenário serve para se estudar a influência do parâmetro *RTOmax* no MOS.

Em cada cenário há parâmetros que podem variar para cada simulação e que são passados como argumentos de entrada. Portanto, para cada cenário são simulados diversos casos diferentes, onde cada caso é uma combinação específica de parâmetros. Por sua vez, cada caso é executado 100 (cem) vezes, cada vez usando uma semente diferente do gerador de números aleatórios (*Random Number Generator* ou RNG) do NS2. A rigor são utilizadas diferentes sub-faixas das sequências pseudo-aleatórias de uma mesma semente que gera sequências não correlacionadas. Essa quantidade de repetições (cem) permite estimar com uma boa precisão as médias dos valores de MOS, conforme avaliado em simulações preliminares.

3.1 Configuração das Simulações

O *script* em Otcl do primeiro cenário serve de base para o segundo, o que faz com que os cenários apresentem muitas características em comum. Portanto, o primeiro cenário será descrito em detalhes e sobre o segundo serão apresentadas as diferenças em relação ao primeiro.

Para maiores detalhes de implementação o anexo B é o *script* Otcl utilizado nas simulações do cenário 2.

3.1.1 Cenário 1: *delay-centric* e o *PMR*

A finalidade deste cenário é criar diversas condições de tráfego de fundo e perdas de forma a avaliar o comportamento do SCTP na transmissão de uma chamada VoIP. São variados os valores de *PMR* com e sem o uso do *delay-centric* na intenção de estudar o funcionamento do mecanismo de detecção de falhas e do *delay-centric*. Isso permite avaliar se sua atuação simultânea é desejável ou mesmo compatível quando existem perdas, além de permitir determinar o valor de *PMR* que apresenta o melhor desempenho para a transmissão do tráfego VoIP.

A topologia simulada neste cenário consta na figura 3.1. Nela, todos os enlaces são *full-duplex*, apresentam velocidades iguais a 1 Mbps, atraso de propagação de 10 ms e filas de saída com limite de 50 pacotes. Os agentes SCTP são conectados aos nós 8 e 9. Em todas as simulações o SCTP é configurado para tráfego não-confiável e não-ordenado, e a opção de SACKs atrasados é desativada.

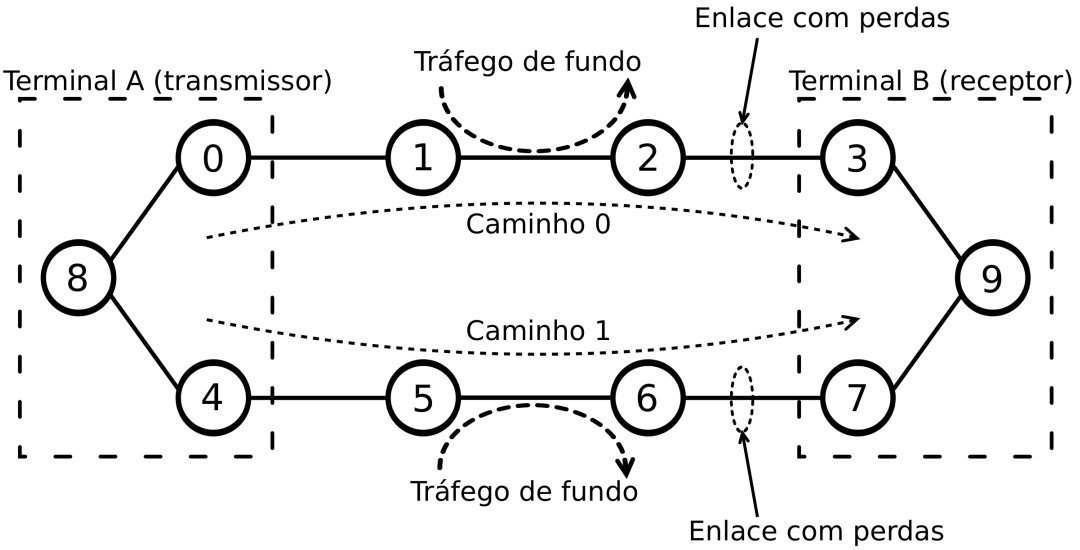

Figura 3.1: Topologia utilizada em todas as simulações. Cada link é *full-duplex*, com taxa de transmissão de 1Mbps e 10ms de atraso de propagação. O tráfego de fundo ocorre entre os nós 1 - 2 (caminho 0) e 5 - 6 (caminho 1). As perdas ocorrem nos enlaces entre os nós 2 - 3 e 6 - 7.

São mantidos constantes para todas as simulações, mas com valores diferentes do SCTP padrão, os seguintes parâmetros: $HBinterval = 0{,}1$ s, $RTOmin = 0{,}5$ s, $RTOini = 1$ s e $RTOmax = 10$ s. Estes três últimos parâmetros são configurados para valores menores do que os padrão (que são 1 s, 3 s e 60 s respectivamente) com base nas conclusões de Fallon et al. [14]

que sugerem valores menores para o *RTOmax*. Análises de simulações preliminares também indicaram que os valores padrão não são os mais apropriados.

O parâmetro *HBinterval*, por sua vez, influencia a freqüência com que são enviados os pacotes de pulsação no caminho secundário (ver seção 2.1.2). Seu valor padrão é de 60 s. Gavriloff [3] utilizou o valor de 1 s para esta variável, pois assim a taxa de atualização do *SRTT* do caminho alternativo para uso do *delay-centric* se torna maior, aumentando a reatividade e eficiência deste. Neste trabalho, optou-se por utilizar o menor valor que ainda mantivesse um tráfego não elevado no caminho secundário em relação ao tráfego VoIP. O valor escolhido aqui (0,1 s) garante que o tráfego gerado pelos pacotes de pulsação no caminho alternativo será de até cerca de 3% do tráfego VoIP no caminho primário.

Todos os demais parâmetros são mantidos padrão do SCTP, com exceção do *PMR* que é um parâmetro variável. No início de qualquer simulação, o caminho 0 é o escolhido como primário pelo SCTP.

O tráfego VoIP simulado representa o padrão G.711 [29]. Dessa forma, se utiliza um gerador de tráfego CBR conectado ao agente SCTP do nó 8 que gera pacotes de 160 bytes a cada 20 ms. É feito uso de um *buffer anti-jitter* fixo de 100 ms (5 pacotes). É também assumido o uso do algoritmo de PLC (*Packet Loss concealing*) conforme apêndice 1 do padrão G.711 [30]. Os efeitos do *buffer anti-jitter* fixo e do mecanismo de PLC são levados em conta no processamento dos *traces* para fins de cálculo do MOS (mais detalhes adiante).

Tráfegos de fundo são gerados entre os nós 1 e 2, e 5 e 6. Os tráfegos utilizam UDP com pacotes de 500 bytes e um intervalo médio entre pacotes calculado para fornecer um atraso médio adicional arbitrário na presença do tráfego VoIP (mais detalhes adiante). O atraso final experimentado por cada pacote é, portanto, o atraso de propagação e transmissão dos 3 enlaces somados (aproximadamente 35 ms e constante para todas as simulações) mais o atraso sofrido na fila dos nós 1 ou 5, dependendo do caminho que o tráfego VoIP estiver seguindo, devido ao tráfego de fundo. Dessa forma, o menor atraso que qualquer pacote pode experimentar durante uma simulação ocorre quando a fila (dos nós 1 ou 5) está vazia. Esse atraso mínimo é de aproximadamente 35 ms.

As perdas ocorrem entre os nós 2 e 3, e 6 e 7. Dessa forma os pacotes do tráfego de fundo não são perdidos, e os atrasos não são afetados pelas perdas. Os detalhes de implementação do modelo de perdas utilizado constam adiante.

Em todas as simulações, os tráfegos de fundo são iniciados no instante 1 s e interrompidos no instante 141 s. O início do tráfego VoIP é escolhido de forma aleatória para

cada simulação de maneira uniforme entre 10 s e 20 s, permanecendo ligado por 120 s. Essa duração (120 s) para o tráfego VoIP é suficiente para que os mecanismos sob estudo possam atuar diversas vezes ao longo de uma simulação, além de ser uma duração razoável para uma ligação telefônica.

São 7 os parâmetros que devem ser especificados em cada simulação: estado do *delay-centric* (ligado ou desligado), atraso médio na fila dos caminhos 0 e 1, valor do *PMR*, perdas médias nos caminhos 0 e 1 e sub-faixa do gerador de números aleatórios. Os valores utilizados para cada um desses parâmetros consta na tabela 3.1. Cada combinação de parâmetros gera uma simulação distinta. Todas as combinações foram simuladas, sendo que cada uma durou em média 9 s nos computadores utilizados.

Tabela 3.1: Parâmetros de entrada das simulações do cenário 1. Cada combinação resulta em uma simulação diferente.

Parâmetro de simulação	**Valores simulados**
Estado do *delay-centric*	ligado e desligado
Atraso médio na fila do caminho 0	5ms, 10ms, 25ms, 50ms e 75ms
Atraso médio na fila do caminho 1	5ms, 10ms, 25ms, 50ms e 75ms
PMR	0, 1, 2 e 5
Média de perdas do caminho 0	0.1%, 0.5%, 1%, 2%, 5% e 10%
Média de perdas do caminho 1	0.1%, 0.5%, 1%, 2%, 5% e 10%
Sub-faixa de RNG	0 a 99

3.1.2 Cenário 2: *RTOmin* e *RTOmax*

Neste cenário a finalidade é avaliar a influência dos parâmetros *RTOmin* e, principalmente, *RTOmax* na qualidade das chamadas VoIP em diferentes condições de tráfego de fundo e perdas. Também são variados o estado do *delay-centric* e os valores de *PMR* de forma a estender a dimensão das análises. A hipótese nesse caso é que a diminuição do valor de *RTOmax* provoca, em média, um aumento no MOS.

O *script* utilizado para este cenário é muito semelhante ao do cenário anterior. Características tais como a topologia, tráfego de fundo, perdas e agenda da simulação são exatamente iguais às do cenário 1. As únicas diferenças são que os parâmetros *RTOmin* e *RTOmax* também são argumentos de entrada, e o valor de *RTOini* não é o mesmo para todas as simulações, sendo definido de acordo com o valor de *RTOmax* de forma a não ser maior do que este. Também há uma regra nesse cenário que faz com que a simulação não seja realizada caso o valor passado para *RTOmax* seja menor do que o valor de *RTOmin*.

No cenário 2, portanto, são 9 os parâmetros que devem ser especificados para cada

simulação. A tabela 3.2 lista todos esses parâmetros com os respectivos valores utilizados nas simulações. Aqui também todas as combinações possíveis foram simuladas e o tempo médio de simulação continuou sendo de aproximadamente 9 s nos equipamentos utilizados.

Tabela 3.2: Parâmetros de entrada das simulações do cenário 2. Cada combinação resulta em uma simulação diferente.

Parâmetro de simulação	**Valores simulados**
Estado do *delay-centric*	ligado e desligado
Atraso médio na fila do caminho 0	5ms, 25ms e 75ms
Atraso médio na fila do caminho 1	5ms, 25ms e 75ms
PMR	0, 1, 2 e 5
Média de perdas do caminho 0	0.5%, 2% e 10%
Média de perdas do caminho 1	0.5%, 2% e 10%
RTOmin	0.02s, 0.08s, 0.1s e 0.5s
RTOmax	10s, 5s, 2s, 1s, 0.5s, 0.2s, 0.12s, 0.1s, 0.08s, 0.05s e 0.02s
Sub-faixa de RNG	0 a 99

3.2 Gerador de tráfego de fundo

O gerador de tráfego de fundo utilizado em todos os cenários é o mesmo. Seu objetivo é prover a simulação com um enlace de gargalo onde ocorre a formação de fila na saída do roteador de forma que os pacotes VoIP sofram um atraso adicional variável. Ele gera pacotes UDP de 500 bytes a um intervalo de tempo aleatório e que segue uma distribuição exponencial. Dessa forma, a fila gerada é do tipo M/D/1 (ver seção 2.4).

A média do intervalo de tempo entre a geração de pacotes é calculada de acordo com o atraso médio desejado para a fila. É possível reescrever a equação 2.14 da seguinte forma:

$$W = \frac{\rho}{2\mu(1-\rho)} \tag{3.1}$$

$$\frac{\rho}{1-\rho} = 2\mu W \tag{3.2}$$

$$\frac{1}{\rho} - 1 = \frac{1}{2\mu W} \tag{3.3}$$

$$\frac{1}{\rho} = \frac{1}{2\mu W} + 1 \tag{3.4}$$

$$\frac{1}{\rho} = \frac{2\mu W + 1}{2\mu W} \tag{3.5}$$

substituindo 2.13 e 2.11 em 3.5:

$$\mu . INTVmed = \frac{2\mu W + 1}{2\mu W} \tag{3.6}$$

$$INTVmed = \frac{2\mu W + 1}{2W\mu^2} \tag{3.7}$$

por sua vez, μ, que representa a capacidade de envio de pacotes em pacotes/segundo, é calculado da seguinte forma:

$$\mu = \frac{\left\{1000000 - 83200 \times \max\left[1, \left(\frac{Nfluxos}{Ncaminhos}\right)\right]\right\}}{4000} \tag{3.8}$$

onde $Nfluxos$ é o número de fluxos VoIP na simulação, $Ncaminhos$ é o número de caminhos da simulação e max[] é a função de máximo. As constantes da eq 3.8 são a largura de banda do enlace por onde circula o tráfego de fundo (1000000 bps ou 1 Mbps), o tamanho do pacote do tráfego de fundo em bits (4000 bits) e a taxa de bits do codec G.711 considerando o overhead SCTP (aproximadamente 83200 bps). Dessa forma é possível calcular em função do atraso médio desejado o valor de $INTVmed$, que é usado pelo gerador de tráfego de fundo.

A rigor, na equação 2.14, W se refere ao atraso que sofrem os pacotes UDP, mas é assumido aqui que esse atraso é aproximadamente o mesmo para pacotes VoIP, uma vez que eles passam pela mesma fila. O tratamento do atraso médio na fila se dar em relação aos pacotes UDP também permite considerar que o fluxo VoIP, por ser CBR, diminui a banda disponível para o tráfego de fundo, conforme a equação 3.8. Mas, como o atraso de interesse é aquele sofrido pelo fluxo VoIP, tomou-se o cuidado de descontar no mínimo 83200 bps da banda disponível, visto que para um dado fluxo VoIP esse é o mínimo de desconto que o caminho experimenta em qualquer instante.

3.3 Modelo de perdas em rajada

Como modelo de perdas foi escolhido o modelo de Gilbert com avaliação de troca de estados a intervalos de tempos regulares (ver seção 2.3). O intervalo de tempo entre as avaliações de estado Δt é o mesmo para todos os cenários e igual a $\Delta t = 0{,}005$ s.

Para que apenas a média de perdas (*Pmed*) seja o parâmetro de entrada para o modelo, são necessárias regras ou equações que relacionem os três parâmetros que determinam o modelo de Gilbert ($P1$, $P2$ e L). Além disso, os parâmetros devem ser tais que as durações médias dos estados estejam dentro das faixas desejadas. Procurou-se utilizar valores próximos aos utilizados por [31]. Dessa forma, as durações médias dos estados devem estar entre 1 s e 10 s para o estado sem perdas, e entre 25 ms e 250 ms para o estado com perdas.

O primeiro passo para a determinação dos parâmetros de Gilbert é a definição do parâmetro L, que nas simulações deste trabalho segue a seguinte regra:

$$L = \begin{cases} 0{,}8 & 0 < P_{med} < 0{,}05 \\ 0{,}75 + P_{med} & 0{,}05 \leq P_{med} < 0{,}15 \\ 0{,}9 & 0{,}15 \leq P_{med} < 0{,}5 \\ 0{,}55 + \frac{P_{med}}{2} & 0{,}5 \leq P_{med} < 0{,}7 \\ 1 & P_{med} \geq 0{,}7 \end{cases} \tag{3.9}$$

Essa regra designa valores adequados para L seja qual for o valor de perdas médias escolhido. Em seguida, se calcula o valor de $P1$ (probabilidade de mudar para o estado com perdas):

$$P1 = \frac{\sqrt{P_{med}}}{100L} \tag{3.10}$$

Essa heurística garante valores adequados para $P1$ dentro da faixa de valores de perdas médias utilizados nas simulações. Uma vez determinados L e $P1$, $P2$ é calculado a partir da equação 2.8, que pode ser reescrita da seguinte forma:

$$P2 = P1\left(\frac{L}{P_{med}} - 1\right) \tag{3.11}$$

Assim, $P2$ também alcança valores dentro da faixa estipulada para os valores de perdas médias simulados.

3.4 Cálculo do MOS

O cálculo do MOS está implementado em um programa escrito em C++ que toma o arquivo de *trace* gerado na simulação como entrada e calcula um MOS para cada fluxo VoIP presente no trace. Este cálculo é o mesmo para todos os cenários simulados. São dois os

parâmetros que influenciam o cálculo do MOS: o atraso médio dos pacotes em ms (T) e a porcentagem de pacotes perdidos (P_{pl}).

É levado em conta o *buffer anti-jitter* fixo de 100 ms (5 quadros VoIP). Assim, o instante de chegada do primeiro pacote VoIP transmitido com sucesso se torna a referência do *buffer anti-jitter*. Dessa forma, o áudio de todos os pacotes é reproduzido com um atraso (em relação à sua geração) igual ao atraso do primeiro pacote acrescentado de 100 ms. Pacotes que apresentem um atraso total até 100 ms maior do que o atraso sofrido pelo primeiro pacote são reproduzidos. Pacotes que sofram um atraso maior do que isto são descartados pelo *buffer anti-jitter*.

Como todos os pacotes que não são descartados são entregues à aplicação com um mesmo atraso (atraso do primeiro pacote + 100 ms), esse também é o valor do atraso médio dos pacotes (T) e não há *jitter* ($jitter = 0$ ms).

Determinar as perdas exige uma análise mais cuidadosa do *trace* gerado na simulação. Isso porque, devido ao mecanismo de controle de congestionamento do SCTP, os instantes de envio dos pacotes não necessariamente coincidem com a geração dos quadros VoIP. Além disso, podem haver mais de um quadro VoIP por pacote.

A estratégia adotada é a de avaliar cada quadro VoIP em relação ao seu tempo de chegada, independente do pacote que o contém. Se o quadro está em um pacote que sofreu uma perda na rede, esse quadro é contabilizado nas perdas. Também se o quadro chegar ao destino com um atraso superior ao permitido pelo *buffer anti-jitter* ele é contabilizado como uma perda. Caso nada disso aconteça, o quadro é reproduzido. Ver o anexo C para o algoritmo detalhado.

O P_{pl} é então calculado com base na quantidade de quadros enviados e na quantidade de quadros perdidos (na rede ou pelo *buffer anti-jitter*). Com os valores de T e P_{pl} em mãos, o fator R é calculado e, em seguida, o valor do MOS para o fluxo em questão, conforme já explicado na seção 2.5. Os valores dependentes do codec, que nos cenários simulados é o G.711 com PLC, são utilizados conforme a tabela de [32], $Ie = 0$ e $B_{pl} = 25{,}1$.

4 RESULTADOS

Inicialmente serão expostos alguns casos particulares que ajudarão a entender melhor os resultados obtidos envolvendo o MOS e as discussões posteriores. Esses primeiros casos que serão expostos foram extraídos de simulações do cenário 1, mas eles ilustram situações que ocorrem em ambos os cenários deste trabalho.

A figura 4.1 ilustra graficamente os dados obtidos de uma única simulação, na qual o atraso médio na fila dos dois caminhos é de 25 ms, as perdas médias são de 2%, o *delay-centric* está ativado e $PMR = 0$. Nesse gráfico, cada ponto corresponde a um pacote, com o instante no qual ele foi gerado representado pelo eixo x e o atraso pelo eixo y, ambos em segundos. A linha tracejada indica o valor máximo de atraso tolerado pelo *buffer anti-jitter*. Os pacotes descartados na rede são representados sobre o eixo x (atraso de 0s). Nesse gráfico, os pacotes que estão acima da linha tracejada foram descartados pelo *buffer anti-jitter* e, portanto, contaram como perdas adicionais para fins de cálculo do MOS. Cruzes azuis indicam pacotes que foram enviados pelo caminho 0 e circunferências vermelhas indicam pacotes que foram enviados pelo caminho 1.

Com o *delay-centric* ativado, a troca de rotas se torna bastante freqüente quando o tráfego de fundo é equivalente para os dois caminhos. Quando o atraso é simétrico, não há um caminho preferencial do ponto de vista de atraso. Assim, pequenas mudanças no atraso tendem a favorecer um caminho sobre o outro, e portanto levar à mudança de rota. Quando há uma assimetria maior entre os tráfegos de fundo, as mudanças tendem a diminuir em freqüência, pois nesses casos um caminho é mais vantajoso do ponto de vista de atraso do que o outro. A figura 4.2 mostra um caso de atraso assimétrico.

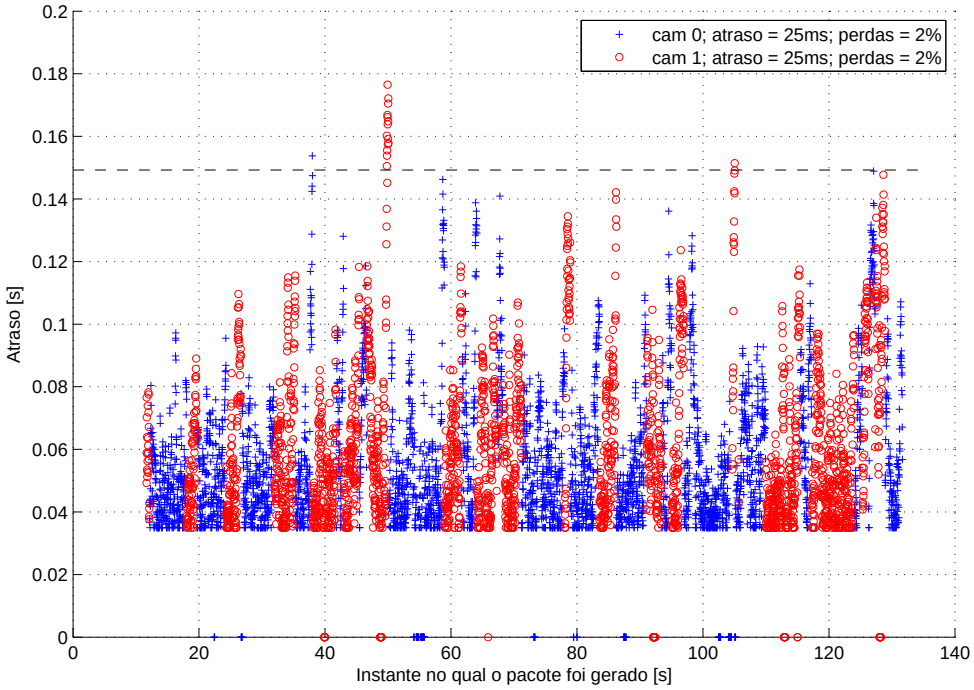

Figura 4.1: Atrasos dos pacotes (pacotes perdidos representados sobre o eixo x). Cenário 1 com *delay-centric* ligado e $PMR = 0$. Caminho 0: atraso = 25ms, perdas = 2%; caminho1: atraso = 25ms, perdas = 2%. A média do atraso dos pacotes é 58ms. MOS = 3,99.

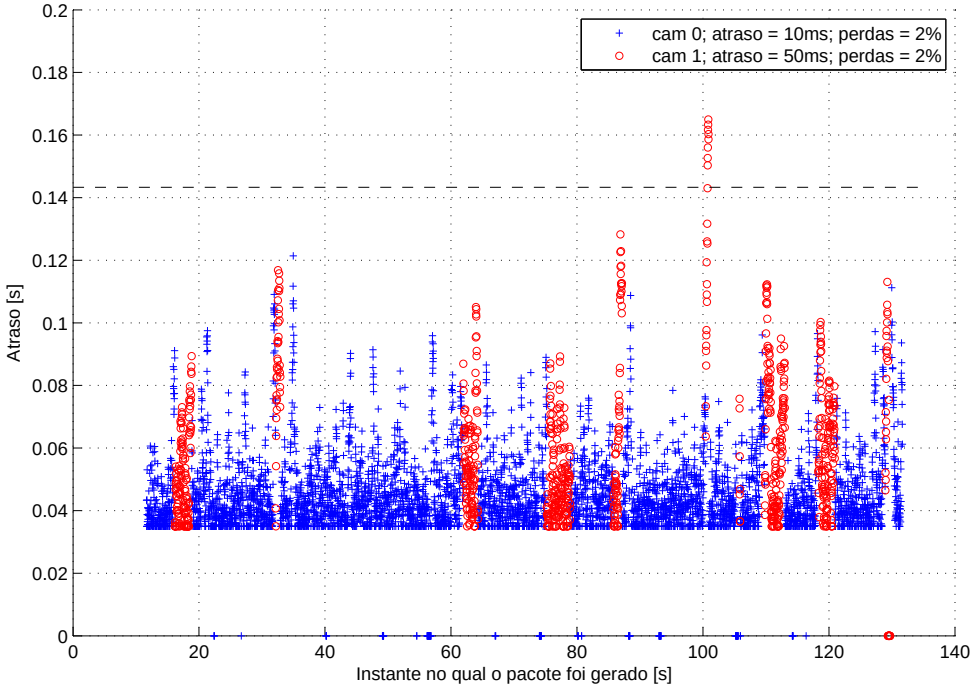

Figura 4.2: Atrasos dos pacotes. Cenário 1 com *delay-centric* ligado e $PMR = 0$. Caminho 0: atraso = 10ms, perdas = 2%; caminho1: atraso = 50ms, perdas = 2%. A média do atraso dos pacotes é 47ms. MOS = 4,05.

Nos casos em que o *delay-centric* estava desativado, a troca de rotas era ainda mais escassa, conforme exemplifica a figura 4.3. Este gráfico corresponde a um caso semelhante ao da figura 4.1, sendo que as únicas diferenças são o fato de o *delay-centric* estar desativado e o $PMR = 1$ nesta última. Outras observações relevantes, mas esperadas, são que sem o *delay-centric* há mais pacotes acima da linha limite do *buffer anti-jitter* e o atraso médio experimentado pelos pacotes é maior. De fato o *delay-centric* ajuda a evitar os momentos de pico de atraso, redirecionando o fluxo SCTP para o caminho alternativo. Isso causa uma diminuição do atraso médio, além de diminuir o *jitter*.

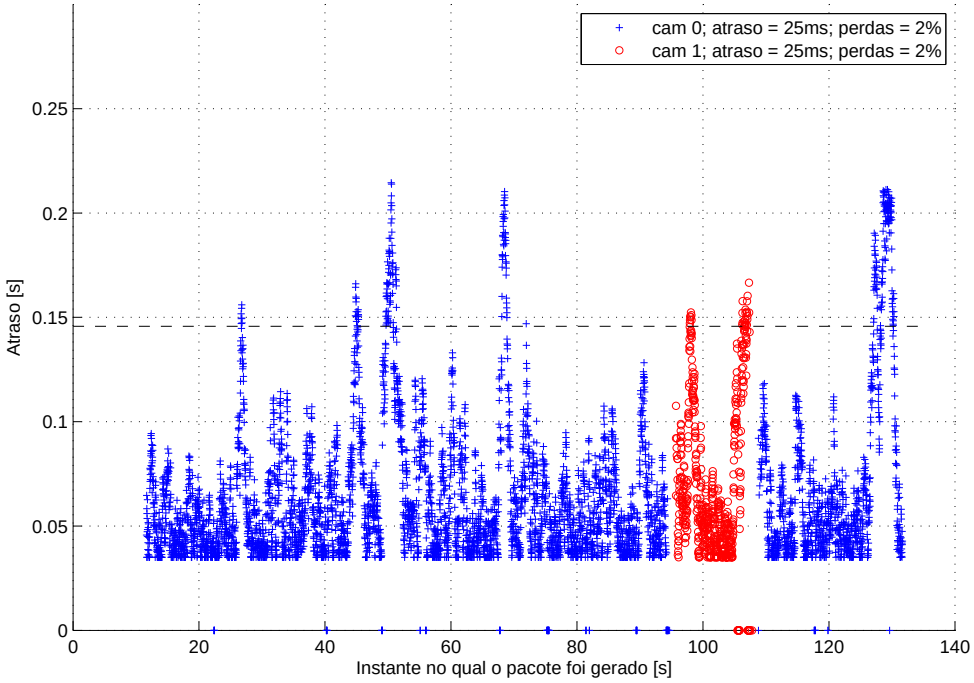

Figura 4.3: Atrasos dos pacotes. Cenário 1 com *delay-centric* desligado e $PMR = 1$. Caminho 0: atraso = 25ms, perdas = 2%; caminho1: atraso = 25ms, perdas = 2%. A média do atraso dos pacotes é 66ms. MOS = 3,31.

Além disso, o *delay-centric* tende a direcionar o fluxo sempre pelo caminho de menor atraso, independentemente do estado de perdas dos caminhos. Isso pode não ser vantajoso em casos em que o caminho de menor atraso apresentar altas perdas.

Durante a análise dos traces foram detectadas duas situações nas quais o fluxo VoIP cessa por um intervalo de tempo. A primeira aconteceu quando o *PMR* era 5. Nesse caso, quando ocorrem rajadas de perdas no caminho primário, a cada estouro de temporização o *RTO* é multiplicado por 2 e a janela de transmissão é reduzida para 2 pacotes. Se a rajada de perdas for longa o suficiente, o *RTO* pode alcançar valores altos, da ordem de até 16 s, antes do caminho ser marcado como inativo e o fluxo seguir pelo caminho alternativo. A situação pode

ser agravada caso, durante a rajada de erros, algum pacote e seu SACK for transmitido com sucesso, pois nesse caso o contador de estouros é reiniciado, causando uma interrupção ainda maior no fluxo VoIP. Para ilustrar essa situação, a figura 4.4 traz novamente o caso da figura 4.3, só que dessa vez com $PMR = 5$.

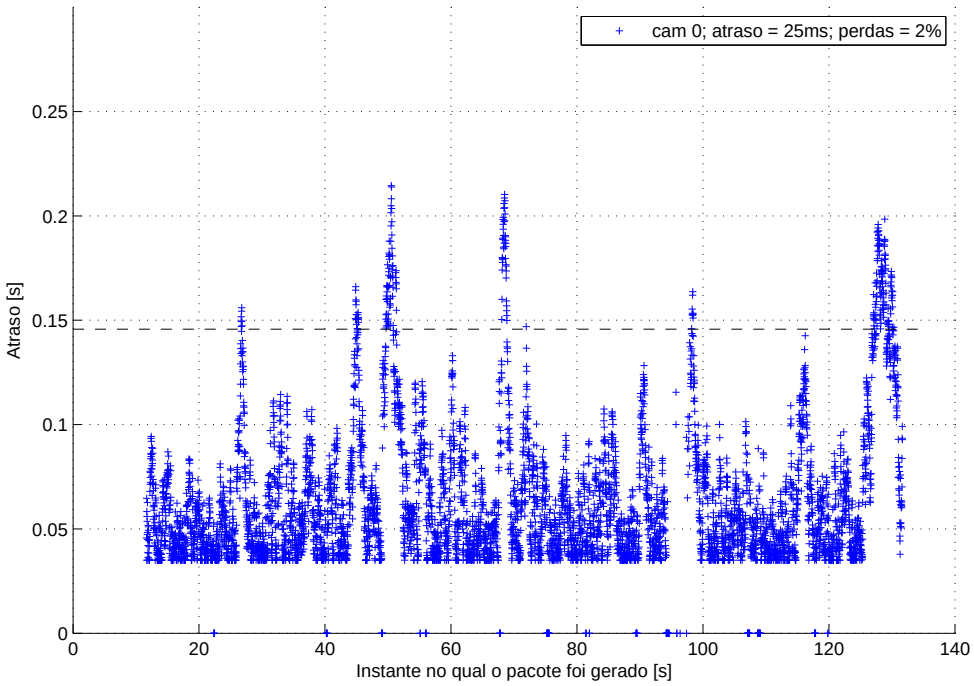

Figura 4.4: Atrasos dos pacotes. Cenário 1 com *delay-centric* desligado e $PMR = 5$. Caminho 0: atraso = 25ms, perdas = 2%; caminho1: atraso = 25ms, perdas = 2%. A média do atraso dos pacotes é 65ms. MOS = 3,32.

No caso da figura 4.3, em torno do instante 94 s o fluxo é redirecionado para o caminho 1 devido a uma rajada de perdas no caminho 0. Já no caso da figura 4.4 isso não acontece, e é observada uma interrupção na transmissão. Neste último caso, o número de estouros de *RTO* não chegou a superar o valor do *PMR* e o tempo entre transmissões sucessivas aumentou devido ao aumento do *RTO*, o que causou essa interrupção. A figura 4.5 traz um caso mais grave onde isso também aconteceu.

Na simulação da figura 4.5 o fluxo não é redirecionado para o caminho 1 em nenhum momento apesar das ocorrências de rajadas de perdas. Isso porque com *PMR* de 5 são necessários 6 estouros de temporização para o caminho ficar inativo. Cada vez que ocorre um estouro de *RTO*, este é dobrado e consequentemente também o é o tempo de envio do pacote seguinte. Isso ocasiona as interrupções que são observadas nesse caso, além de causar um acúmulo de dados no transmissor. Assim, quando cessam as perdas, ocorre um grande fluxo de dados enviado pelo SCTP, o que sobrecarrega o enlace de gargalo causando os picos de

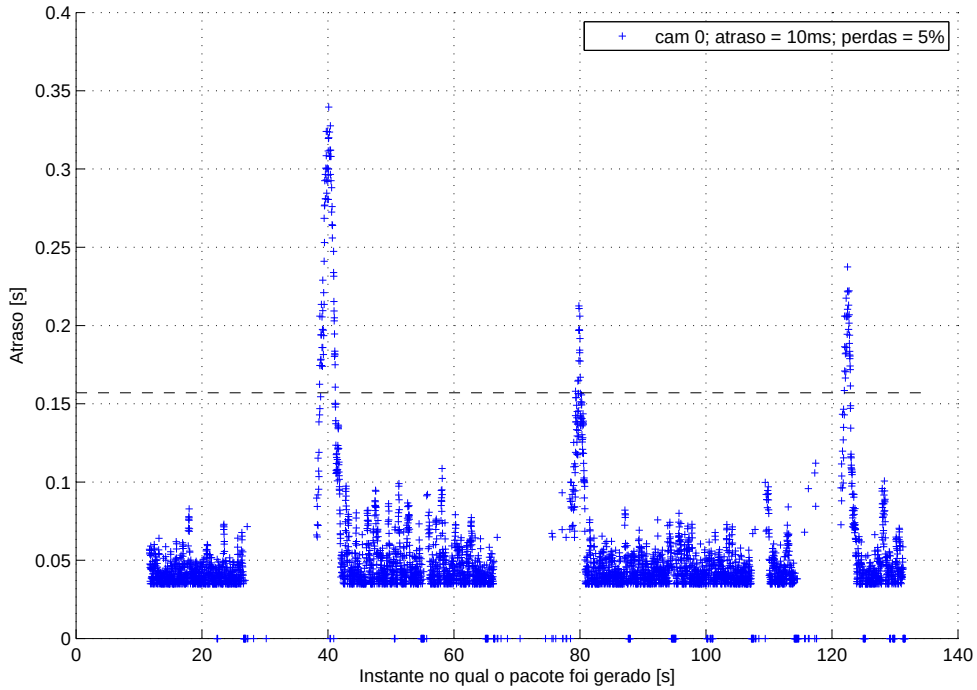

Figura 4.5: Atrasos dos pacotes. Cenário 1 com *delay-centric* desligado e $PMR = 5$. Caminho 0: atraso = 10ms, perdas = 5%; caminho1: atraso = 10ms, perdas = 5%. A média do atraso dos pacotes é 54ms. MOS = 1,75.

atraso logo após o término da rajada de perdas. Dessa forma, além das perdas na rede, ocorrem descartes pelo *buffer anti-jitter*, o que prejudica bastante a qualidade da chamada e resulta em um baixo valor de MOS.

Conforme será mostrado na sequência, em geral utilizar um valor alto (5) para o *PMR* resulta em uma pior qualidade de voz em relação a situações que utilizam valores menores de *PMR*. Assume-se que o principal motivo para isso é o problema ilustrado no caso da figura 4.5. Um outro parâmetro que tem uma forte influência nessas situações de interrupção por aumento excessivo do *RTO* é o *RTOmax*. Os resultados que mostram isso serão apresentados adiante.

A segunda situação de interrupção da transmissão ocorre quando o *PMR* é baixo, especialmente quando este é igual a zero. Nesses casos pode ocorrer de o tráfego VoIP cessar caso ocorram perdas nos dois caminhos e que levem ambos ao estado inativo. Nesse caso a associação entra no estado dormente até que algum pacote de pulsação seja respondido com um ACK de pulsação e algum dos caminhos se torne novamente ativo. A figura 4.6 ilustra essa situação com o mesmo caso da figura 4.3, só que agora com o $PMR = 0$.

No caso da figura 4.6, ao ocorrer perdas no caminho 1, este se torna inativo após o primeiro estouro de *RTO*. Como nesse instante o caminho 1 está também inativo, não há

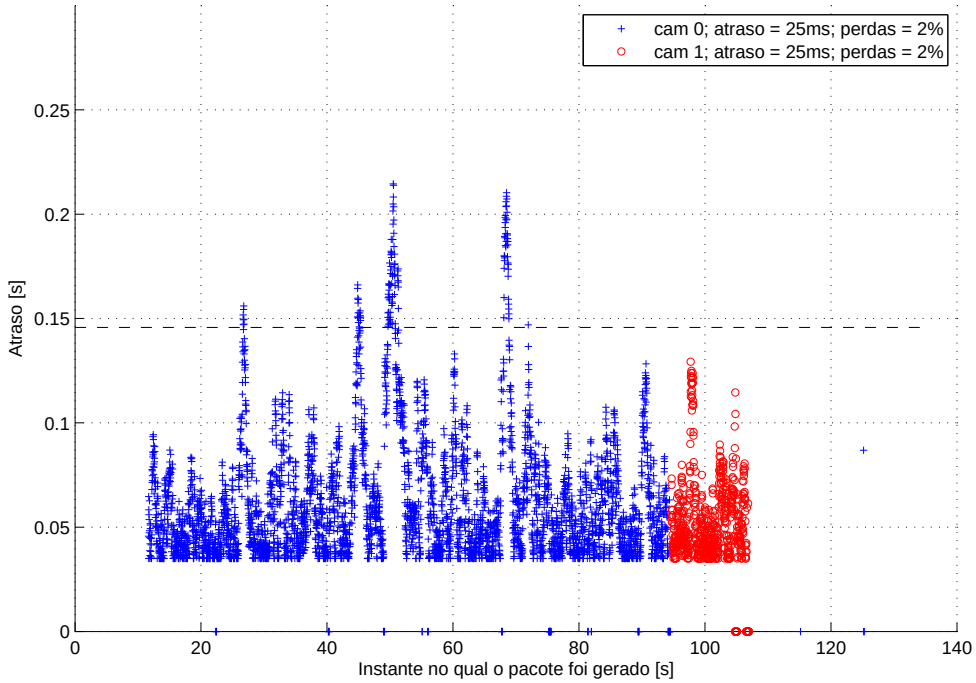

Figura 4.6: Atrasos dos pacotes. Cenário 1 com *delay-centric* desligado e $PMR = 0$. Caminho 0: atraso = 25ms, perdas = 2%; caminho1: atraso = 25ms, perdas = 2%. A média do atraso dos pacotes é 61ms. MOS = 2,69.

caminho disponível para enviar dados, e a associação entra no chamado estado dormente. Isso interrompe o fluxo VoIP a partir do instante 108 s aproximadamente. O estado dormente pode ser terminado se houver a resposta, por algum dos caminhos, de um pacote de pulsação, o que tornaria o caminho em questão ativo novamente. Portanto, nesse tipo de situação o parâmetro *RTOmax* também tem um papel importante, pois o intervalo entre envios de pacotes de pulsação também depende do valor do *RTO* (ver seção 2.1.2).

Portanto, perdas em rajadas podem gerar duas situações distintas de interrução de transmissão. Segundo observações realizadas por amostragem nos casos simulados, quanto maior for a duração média das rajadas (maior valor médio de perdas) mais a associação está sujeita a apresentar um desses casos de interrupção, sendo o tipo de interrupção dependente do valor do *PMR*. Em casos de perdas médias inferiores a 1% não foi detectado esse tipo de interrupção.

Todos os casos de interrupção ou de perdas são levados em consideração no cálculo do MOS e, portanto, são contemplados nos valores médios de MOS que serão apresentados nos gráficos a seguir. Como o MOS é o indicativo principal da qualidade da chamada, e a média do MOS obtido de várias sementes simuladas é o principal indicador do nível de qualidade dos

casos simulados, o impacto dessas situações está embutido nos resultados subsequentes, não tendo sido feitos, portanto, estudos mais aprofundados sobre as simulações individuais.

4.1 Cenário 1

Para analisar os resultados deste cenário foram utilizados gráficos como o da figura 4.7. Nele, o atraso nos dois caminhos permanece constante para todos os pontos (mas não necesssariamente o mesmo para os diferentes caminhos) e as perdas médias no caminho 0 também permanecem as mesmas. O eixo *x* representa diferentes valores de perdas médias no caminho 1 e o eixo *y* indica o valor do MOS. No gráfico constam 8 curvas, correspondendo aos 4 valores simulados para o *PMR* (0, 1, 2 e 5) com e sem o *delay-centric* ativado. Cada ponto do gráfico é calculado como a média das 100 simulações que utilizaram diferentes sub-faixas do RNG, e está presente também o intervalo de confiança (IC) de 95% (considerando a variância estimada a partir das 100 amostras como sendo igual a variância do MOS). Na legenda, as curvas denominadas "DCon" (linhas tracejadas) correspondem aos casos de *delay-centric* ativado, enquanto as denominadas "DCoff" (traço contínuo) correspondem aos casos de *delay-centric* desativado.

O gráfico da figura 4.7 ilustra um caso no qual o atraso nos dois caminhos é moderadamente alto (50 ms) e as perdas no caminho 0 são moderadas (2%). Dessa forma, ele representa um caso mediano usual. Ele também resume o resultado mais freqüentemente obtido nas simulações do cenário 1, no qual o menor valor de *PMR* se mostra o mais vantajoso e o uso do *delay-centric* traz ganhos no MOS na maioria dos casos. Nesse gráfico fica evidente a vantagem de se usar um $PMR = 0$.

Nesse caso ainda, enquanto as perdas médias no caminho 1 são inferiores às do caminho 0, o uso do *delay-centric* garante valores de MOS em torno de 3,5, o que representa uma qualidade razoável para chamadas VoIP. Para perdas de 2% ou superiores no caminho 1 há uma diminuição no ganho proporcionado pelo *delay-centric*. Quando as perdas no caminho 1 são de 10% o *delay-centric* traz, ao invés, um leve prejuízo.

Essa queda no MOS observada nos casos de *delay-centric* ativado quando as perdas no caminho 1 são mais intensas é justificada pelo aumento na freqüência de trocas de rota que o uso do *delay-centric* ocasiona, especialmente nesse caso em que os atrasos médios dos dois caminhos são iguais. Dependendo da intensidade do tráfego e das perdas, ocorre (nesse caso, para 10% de perdas no caminho 1) que o ganho trazido pelo *delay-centric* devido a diminuição do atraso médio é menos significativo do que o prejuízo causado pelo fato de os pacotes VoIP

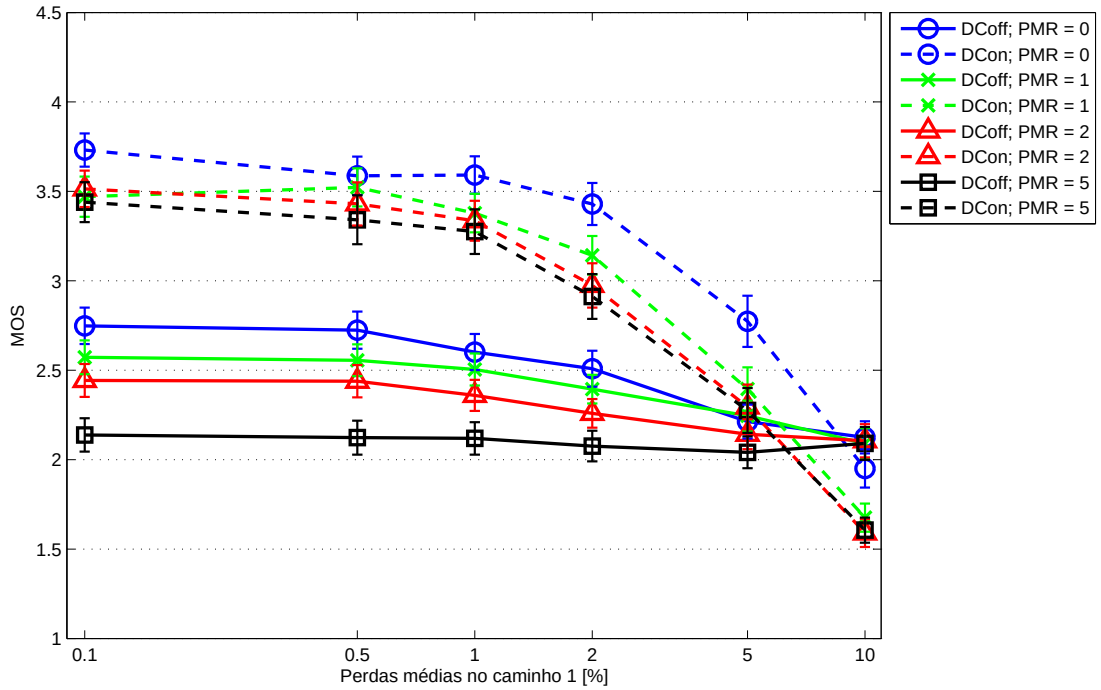

Figura 4.7: Cenário 1. Média do MOS com IC de 95% para casos de: atraso = 50ms e perdas = 2% no caminho 0; atraso = 50ms no caminho 1.

fluirem (durante mais tempo) pelo caminho de altas perdas. A figura 4.8 mostra esse tipo de situação de uma forma mais evidente.

No gráfico da figura 4.8, o caminho 0 apresenta boas condições de tráfego, com baixo atraso e baixas perdas, e o caminho 1 apresenta um atraso igualmente baixo. Nos casos em que as perdas no caminho 1 são baixas também (0,1% e 0,5%), não há distinção na qualidade da chamada, seja com ou sem o *delay-centric*. A partir de 1% de perdas no caminho 1, a qualidade diminui consideravelmente para os casos com *delay-centric* e se mantém alta para os casos sem. Neste caso, o *delay-centric* não traz ganhos porque o atraso já é baixo nos dois caminhos, e o aumento da freqüência de troca de rotas prejudica gravemente o fluxo VoIP, levando casos que sem ele tinham um MOS de 4,2 para 2. Isso reforça o fato de que o *delay-centric* não é sensível a perdas.

Ainda no gráfico da figura 4.8 é possível notar que o uso do $PMR = 0$ reduz o prejuizo causado pelo uso do *delay-centric*. Isso porque com $PMR = 0$ não há tolerância a perdas e a reação do SCTP é muito rápida, o que tende a evitar o fluxo VoIP de passar pelo caminho de altas perdas.

Na figura 4.9 consta uma situação semelhante à da figura 4.8, mas agora com altas perdas no caminho 0. Aqui também o *delay-centric* degrada a qualidade, especialmente quando

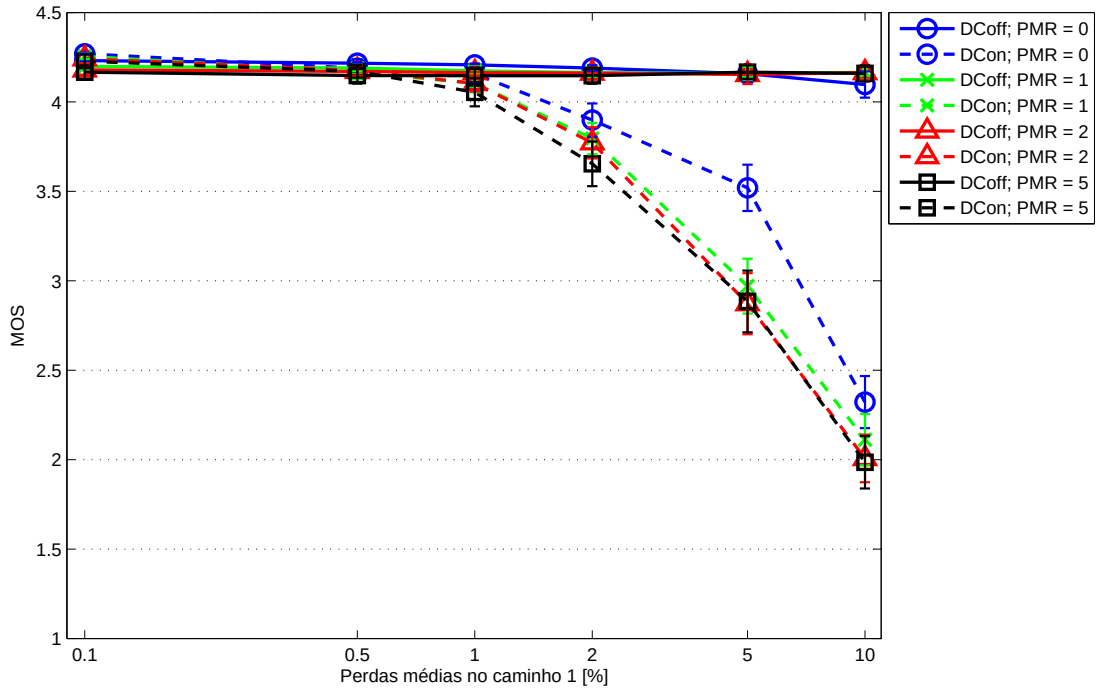

Figura 4.8: Cenário 1. Média do MOS com IC de 95% para casos de: atraso = 10ms e perdas = 0,5% no caminho 0; atraso = 10ms no caminho 1.

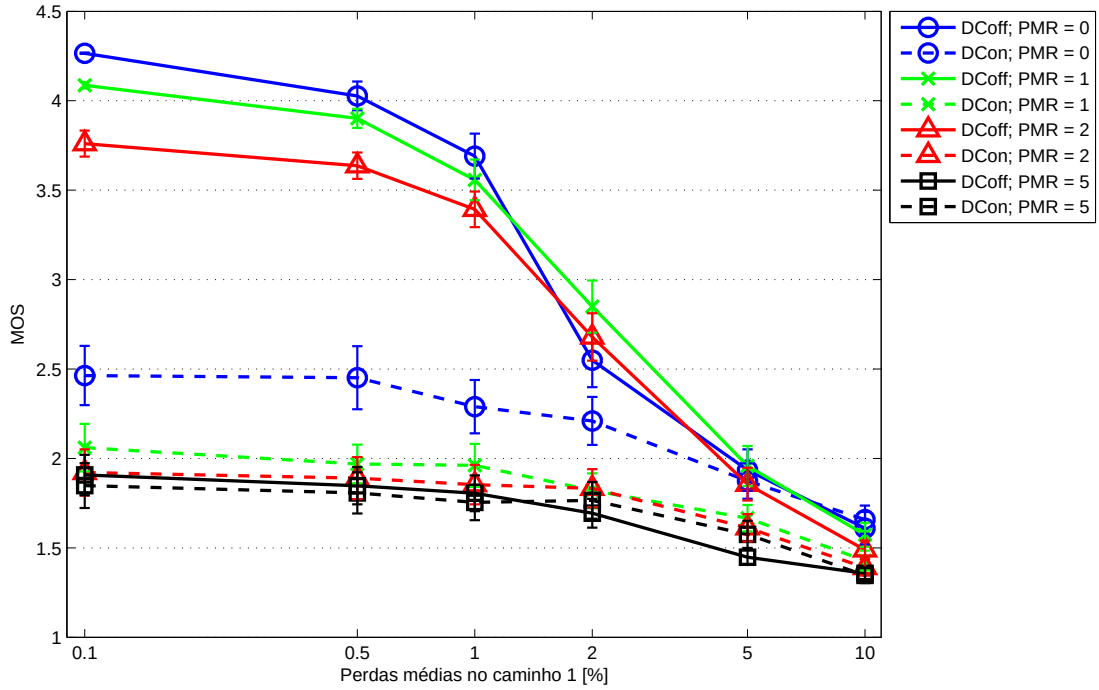

Figura 4.9: Cenário 1. Média do MOS com IC de 95% para casos de: atraso = 10ms e perdas = 10% no caminho 0; atraso = 10ms no caminho 1.

as perdas no caminho 1 são baixas. Os motivos para isso são aqueles já citados. O que se pode notar de diferente é que existe uma diferença considerável de comportamento, para os casos em que não há o *delay-centric* atuando, em relação aos valores de *PMR*. Nos pontos mais à esquerda do gráfico, onde o caminho 1 apresenta baixas perdas, quanto maior o valor do *PMR*, menor é o valor do MOS. Para valores maiores de *PMR* existe uma menor reatividade a perdas e uma tendência maior a permanecer no caminho inicial (caminho 0), que nesse caso apresenta altas perdas.

Para situações de tráfego de fundo intenso e baixas perdas em ambos os caminhos, como ocorre na figura 4.10, o uso do *delay-centric* traz um alto ganho de qualidade. Sua atuação garante que o fluxo VoIP estará sendo freqüentemente redirecionado para o caminho que apresentar, momentaneamente, um menor atraso. Conseqüentemente, quando as perdas em um dos caminhos aumenta (pontos mais à direita no gráfico), o ganho do *delay-centric* diminui. No caso de as perdas serem mais prejudiciais à chamada do que o atraso, o *delay-centric* pode chegar a prejudicar a qualidade. Nesses casos, o uso do $PMR = 0$ ajuda a evitar a degradação causada pelas perdas, garantindo em parte o ganho obtido com o uso do *delay-centric*.

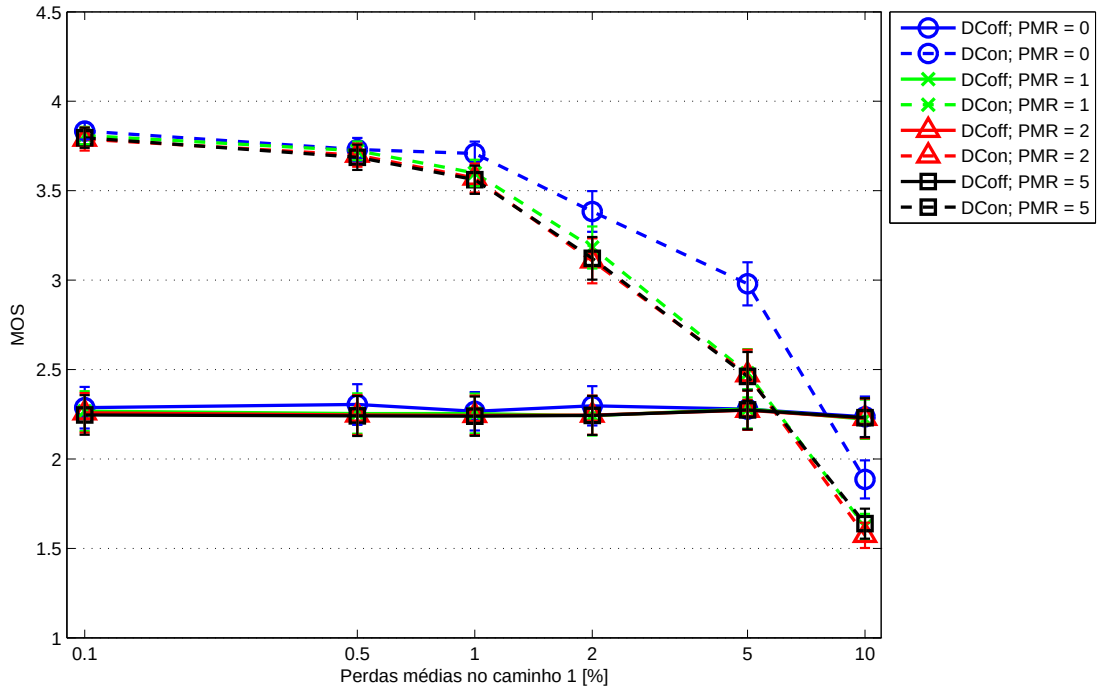

Figura 4.10: Cenário 1. Média do MOS com IC de 95% para casos de: atraso = 75ms e perdas = 0,5% no caminho 0; atraso = 75ms no caminho 1.

Tomando a mesma situação da figura 4.10, mas mudando a intensidade do tráfego de fundo no caminho 1 para um valor médio baixo (10 ms), é possível notar na figura 4.11 que não

há diferença considerável na qualidade para os casos simulados sem *delay-centric*, exceto para $PMR = 0$, em que há um leve ganho de qualidade em caso de baixas perdas no caminho 1. Isso mostra a tendência de permanecer no caminho inicial (caminho 0) que existe para casos sem *delay-centric* quando as perdas são baixas.

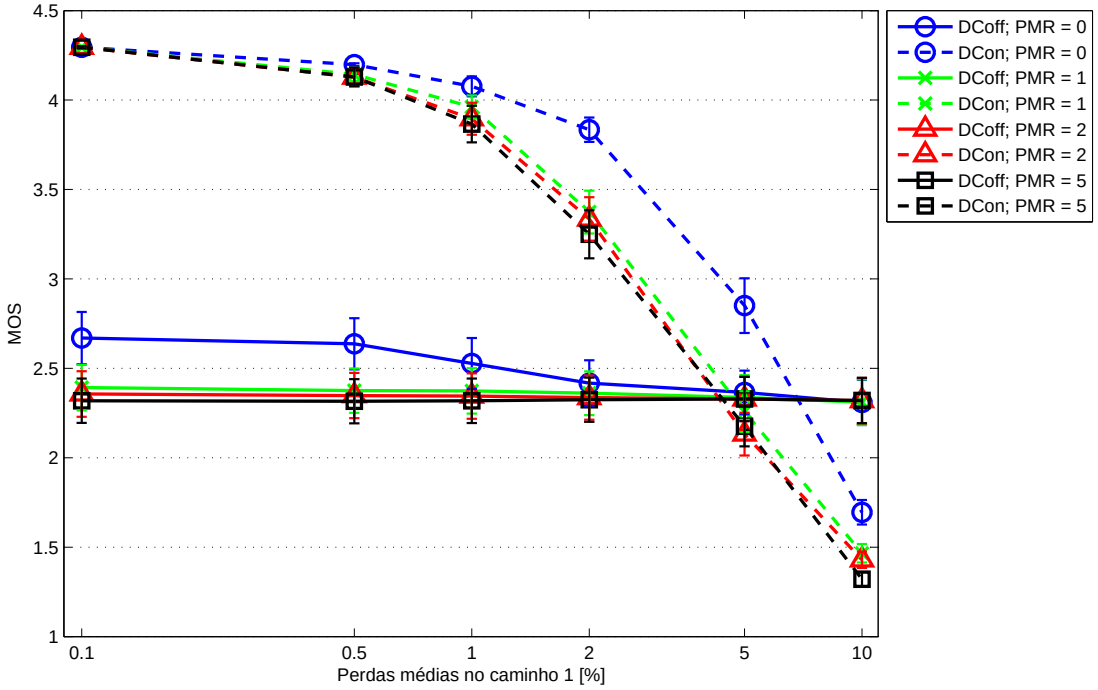

Figura 4.11: Cenário 1. Média do MOS com IC de 95% para casos de: atraso = 75ms e perdas = 0,5% no caminho 0; atraso = 10ms no caminho 1.

Já para o *delay-centric* ligado existe um ganho ainda mais pronunciado na qualidade em relação ao notado na figura 4.10 quando as perdas no caminho 1 são pequenas. Isso mostra como esse mecanismo tende a direcionar o fluxo pelo caminho 1. No entando, com o aumento das perdas no caminho 1 a queda na qualidade é mais pronunciada, sendo que a degradação causada quando estas são de 10% é maior do que aquela observada na figura 4.10.

Outra situação de tráfego de fundo assimétrico, mas agora invertendo sua intensidade nos caminhos é apresentada na figura 4.12. Nela, as perdas no caminho 0 são de 2%. Essa situação é interessante por ser uma das poucas situações em que o $PMR = 0$ pode piorar a qualidade da chamada. Isso foi observado apenas em situações como esta, em que o *delay-centric* não está atuando e o caminho 0 apresenta perdas moderadas e baixo atraso, enquanto o caminho 1 apresenta um valor elevado de atraso. O prejuízo à qualidade causado pelo atraso no caminho 1 é maior do que aquele causado pelas perdas no caminho 0. Sem o *delay-centric*, o caso de $PMR = 0$ vai levar o fluxo VoIP para o caminho 1, causando a

diminuição observada no MOS.

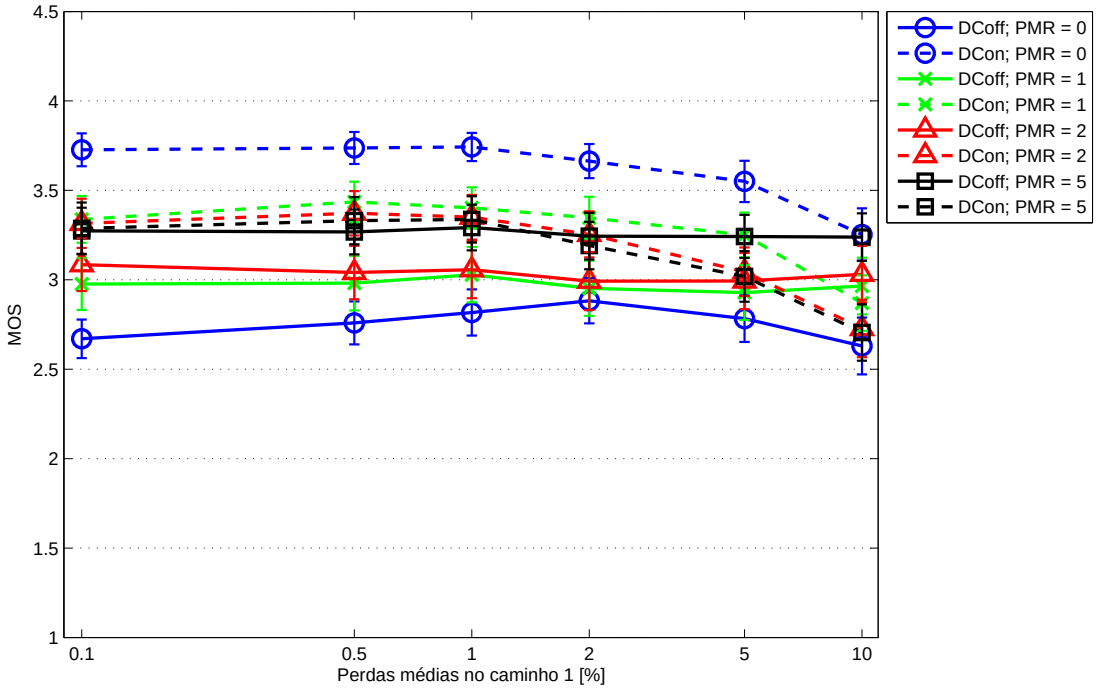

Figura 4.12: Cenário 1. Média do MOS com IC de 95% para casos de: atraso = 10ms e perdas = 2% no caminho 0; atraso = 75ms no caminho 1.

Nessa situação, com o *delay-centric* essa tendência se inverte e o uso do $PMR = 0$ passa a ser vantajosa em relação aos demais valores de *PMR*. Para estes, não há diferença significativa de qualidade, tanto com como sem o *delay-centric*. Mesmo o aumento das perdas no caminho 1 não provoca uma mudança de qualidade significativa.

A tabela 4.1 a seguir contém a média do MOS de todas as simulações realizadas neste cenário para os diferentes valores de *PMR*. Também constam os valores das médias do MOS para os casos exclusivos de *delay-centric* ativado e desativado.

Tabela 4.1: Média dos valores de MOS separados pelo valor de *PMR* para todas as simulações realizadas e para casos com *delay-centric* ativado e desativado.

PMR	**Média do MOS p/ todos os estados do *delay-centric***	**Média do MOS p/ *delay-centric* desativado**	**Média do MOS p/ *delay-centric* ativado**
0	3,19	3,09	3,29
1	3,04	3,05	3,03
2	2,97	2,96	2,97
5	2,82	2,69	2,94

O resultado da média obtida se forem desconsiderados todos os casos de perdas e

atrasos extremos se encontram na tabela 4.2. Nela foram consideradas todas as simulações com valores de atraso de 10 ms, 25 ms e 50 ms (em qualquer caminho e em qualquer combinação) e valores de perdas médias de 0,5%, 1% e 2% (em qualquer caminho e em qualquer combinação).

Tabela 4.2: Média dos valores de MOS separados pelo valor de *PMR* para simulações com valores medianos de atraso e perdas, além dos casos com *delay-centric* ativado e desativado.

PMR	**Média do MOS p/ todos os estados do *delay-centric***	**Média do MOS p/ *delay-centric* desativado**	**Média do MOS p/ *delay-centric* ativado**
0	3,70	3,47	3,92
1	3,58	3,40	3,76
2	3,52	3,32	3,72
5	3,43	3,20	3,67

O resultado geral das tabelas 4.1 e 4.2 mostra que para este cenário o valor de $PMR = 0$ é o que trouxe os melhores valores de MOS na média. Para valores maiores, o MOS é tanto menor quanto maiores forem os valores de *PMR*. Para este cenário, uma tolerância nula a perdas se mostrou a estratégia mais interessante do ponto de vista de qualidade de chamada VoIP.

Também fica evidente por essas tabelas que o uso do *delay-centric* é benéfico, em média, ao tráfego VoIP. Em todos os casos o MOS é mais alto quando do uso deste mecanismo, exceto para $PMR = 1$ na tabela 4.1. Mesmo assim, pode-se considerar esse um caso de empate estatístico, mantendo, assim, o emprego do *delay-centric* como favorável.

4.2 Cenário 2

Um dos problemas que resultam numa queda de qualidade da chamada VoIP detectados no cenário 1 diz respeito à interrupção da transmissão. Conforme explicado anteriormente, esse problema pode se dar devido a um aumento excessivo do *RTO* em casos de $PMR = 5$ ou devido ao estabelecimento do estado dormente da associação em casos de *PMR* muito baixo (especialmente $PMR = 0$).

A figura 4.13 mostra um exemplo do primeiro caso, no qual $PMR = 5$ e ocorre uma interrupção na transmissão de pacotes devido a uma rajada de perdas entorno dos 40 s da simulação (*delay-centric* desativado). Essa interrupção acarreta também num grande aumento do atraso nos pacotes enviados logo após esse intervalo devido a um acúmulo de dados no *buffer* de saída do SCTP. Nesse exemplo, o valor do *RTOmax* é de 10 s. Ou seja, o valor máximo permitido para o *RTO* é de 10 s.

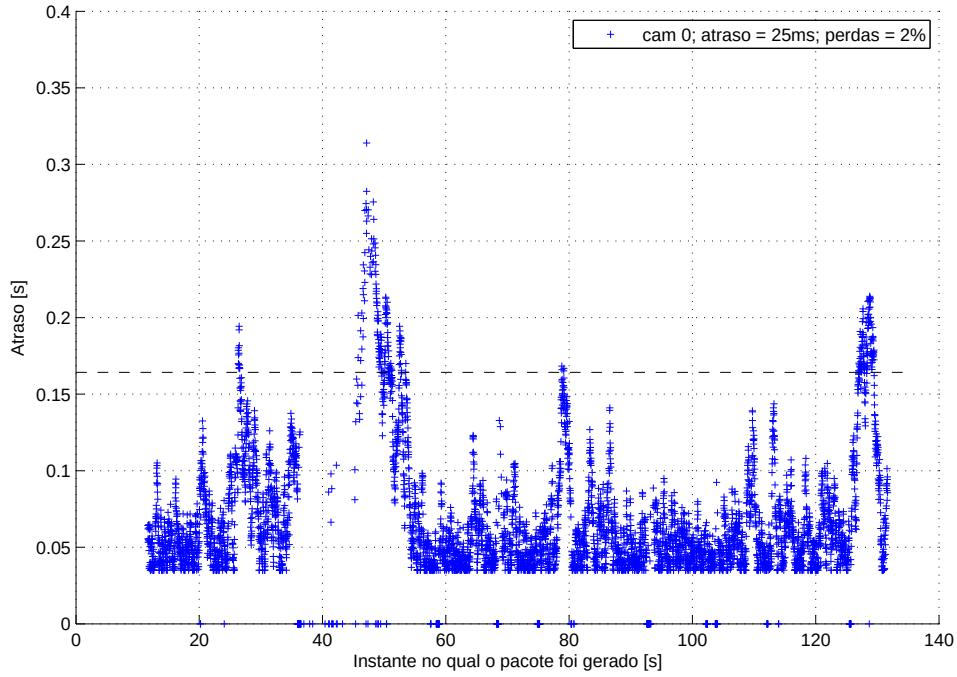

Figura 4.13: Atrasos dos pacotes. Cenário 2 com *delay-centric* desligado, $PMR = 5$, $RTOmin = 20$ms e $RTOmax = 10$s. Caminho 0: atraso = 25ms, perdas = 2%; caminho1: atraso = 25ms, perdas = 2%. A média do atraso dos pacotes é 70ms. MOS = 2,58.

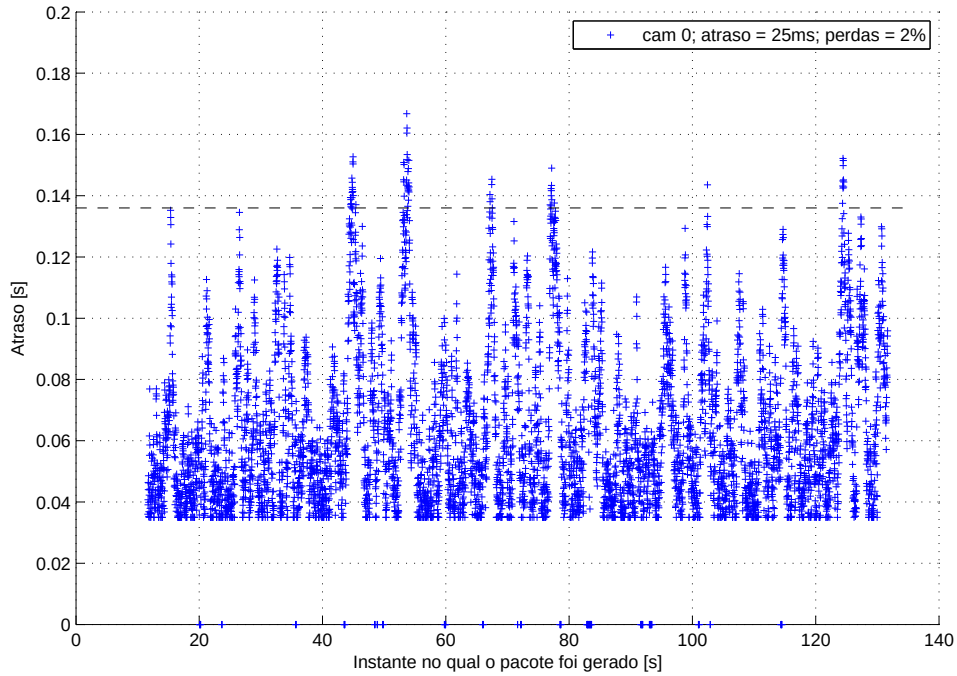

Figura 4.14: Atrasos dos pacotes. Cenário 2 com *delay-centric* desligado, $PMR = 5$, $RTOmin = 20$ms e $RTOmax = 0{,}1$s. Caminho 0: atraso = 25ms, perdas = 2%; caminho1: atraso = 25ms, perdas = 2%. A média do atraso dos pacotes é 63ms. MOS = 3,94.

Na figura 4.14 consta o mesmo caso da figura anterior, com a única diferença sendo o valor do *RTOmax*, que neste caso é de 0,1 s. Dessa forma é possível verificar que não ocorre a interrupção observada na figura 4.13. Com o *RTO* limitado a um valor pequeno, o SCTP não espera longos períodos para enviar um pacote mesmo após a ausência de um SACK e detecção de uma perda. Dessa forma, pequenas rajadas de perdas não prejudicam o fluxo VoIP além das próprias perdas, primeiramente porque assim que a rajada de perdas cessar o receptor voltará a receber os dados, o que significa um restabelecimento mais rápido da chamada. Além disso, não havendo interrupção, poucos dados serão mantidos no *buffer* de saída do SCTP, o que não causa o aumento no atraso observado no caso anterior.

O segundo motivo para a interrupção na transmissão de pacotes VoIP é exemplificado no gráfico da figura 4.15. Nesse caso, as perdas são altas nos dois caminhos, não há *delay-centric*, $PMR = 1$ e $RTOmax = 10$ s. Logo no início da simulação ocorre uma mudança de caminho devido ao caminho 0 se tornar inativo. Em seguida o mesmo ocorre com o caminho 1 e o fluxo é interrompido porque ambos os caminhos estão inativos, o que caracteriza o estado dormente da associação. Essa interrupção dura cerca de 15 s. O fluxo retorna pelo caminho 0, mas com um atraso alto logo após esse restabelecimento, o que causa uma interrupção na comunicação de mais de 20 s, devido ao descarte no *buffer anti-jitter*. Ao final da simulação também é observado um fenômeno desse tipo.

A figura 4.16 trata do mesmo caso do retratado pela figura 4.15, sendo a única diferença o valor do *RTOmax* que no caso da figura 4.16 é de 0,1 s. Como a única forma de uma associação SCTP sair do estado dormente é obtendo uma resposta a um pacote de pulsação em qualquer caminho, um *RTO* menor pode reduzir a duração do estado dormente. Isso porque o intervalo entre o envio de pacotes de pulsação depende do valor do *RTO* (ver equação 2.7). Portanto, no caso da figura 4.16, o estado dormente dura muito menos tempo, garantindo uma melhor continuidade da transmissão e, por conseqüência, uma maior qualidade da chamada VoIP.

O gráfico da figura 4.17 mostra o quanto o valor do *RTOmax* influencia, em média, uma chamada VoIP. Nesse gráfico cada ponto é calculado como a média do valor do MOS obtido com 100 simulações que usam sub-faixas diferentes do RNG. Os intervalos de confiança são de 95% e supõem que a variância do MOS pode ser estimada a partir das 100 amostras. As simulações que compõe esse gráfico apresentavam todas a mesma configuração de atrasos e perdas nos caminhos, além de um mesmo valor de *RTOmin*. O eixo y representa valores de MOS e o eixo x representa valores simulados para *RTOmax* (em escala logarítmica).

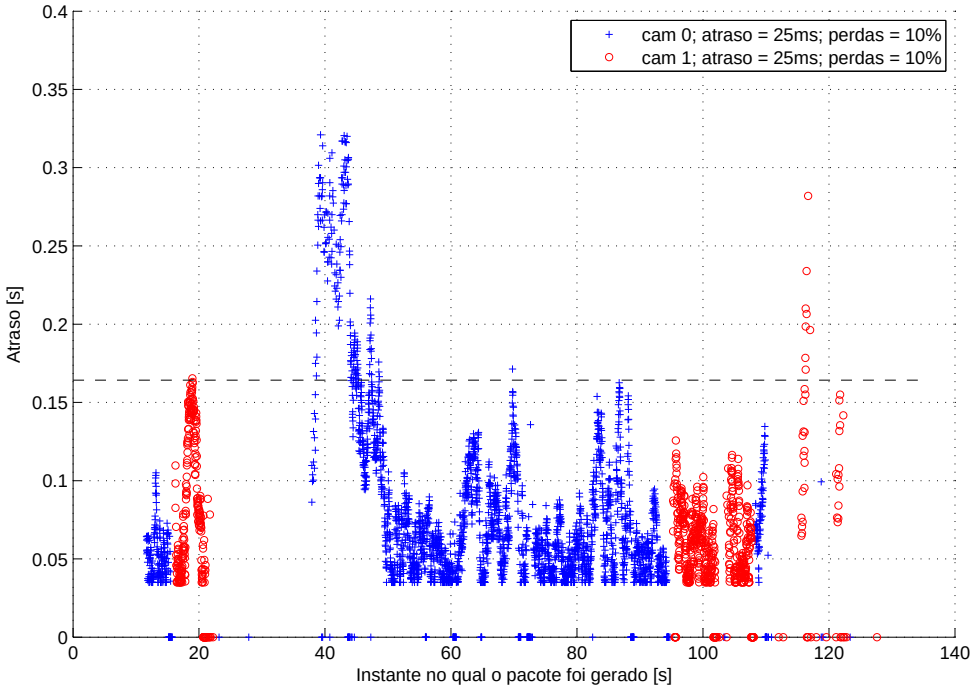

Figura 4.15: Atrasos dos pacotes. Cenário 2 com *delay-centric* desligado, $PMR = 1$, $RTOmin = 20$ms e $RTOmax = 10$s. Caminho 0: atraso = 25ms, perdas = 10%; caminho1: atraso = 25ms, perdas = 10%. A média do atraso dos pacotes é 79ms. MOS = 1,66.

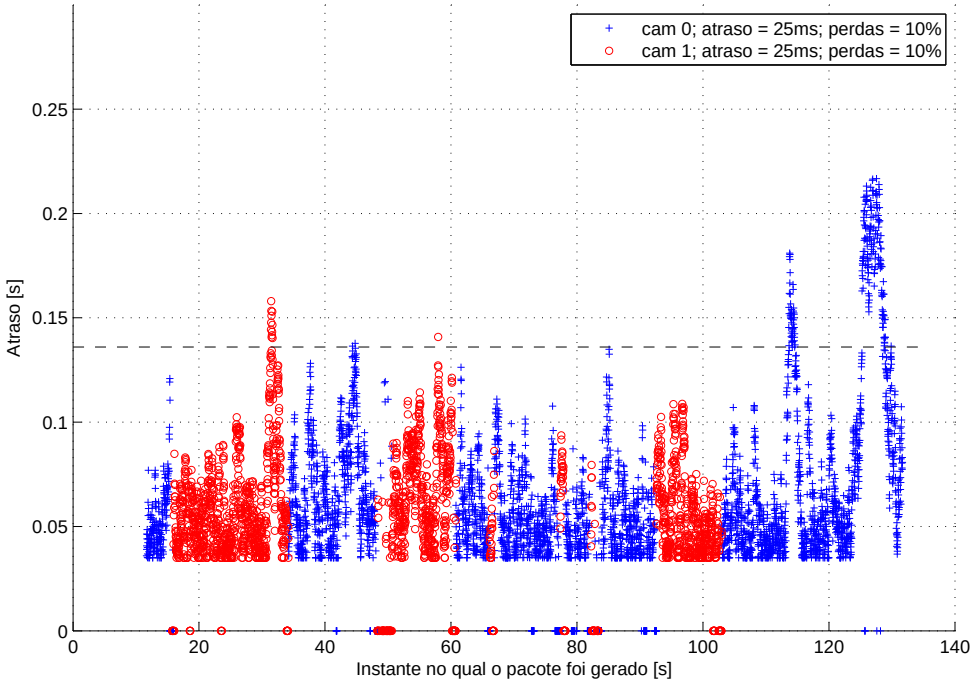

Figura 4.16: Atrasos dos pacotes. Cenário 2 com *delay-centric* desligado, $PMR = 1$, $RTOmin = 20$ms e $RTOmax = 0{,}1$s. Caminho 0: atraso = 25ms, perdas = 10%; caminho1: atraso = 25ms, perdas = 10%. A média do atraso dos pacotes é 64ms. MOS = 3,06.

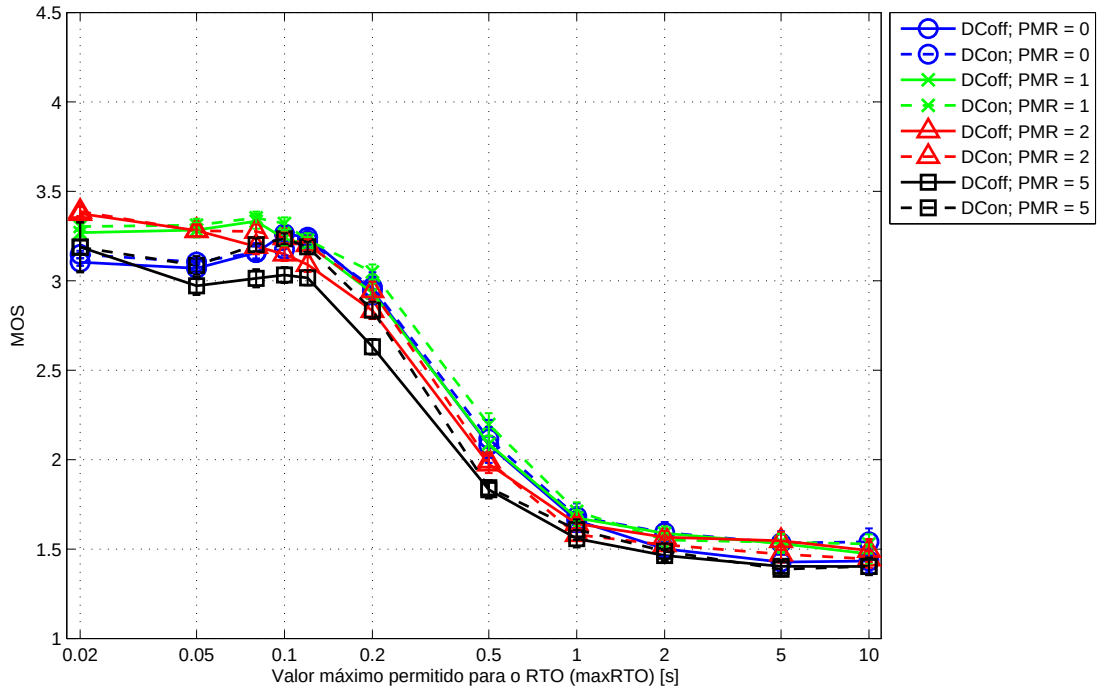

Figura 4.17: Cenário 2. Média do MOS com IC de 95% para casos de: atraso = 25ms e perdas = 10% no caminho 0; atraso = 25ms e perdas = 10% no caminho 1. $RTOmin = 20$ms

Na figura 4.17 consta um caso de atraso médio moderado (25 ms) e perdas altas (10%) em ambos os caminhos. Para esses casos de grandes perdas fica evidente a grande contribuição que baixos valores de *RTOmax* trazem à qualidade das chamadas para todos os valores de *PMR* e estados do *delay-centric*. Para valores de *RTOmax* entre 1 s e 0,1 s, a medida em que este diminui ocorre um aumento no MOS. Para valores maiores de *RTOmax* há pouca variação no MOS, assim como para valores menores do que 0,1 s.

O gráfico da figura 4.17 resume as principais características encontradas em relação à influência do parâmetro *RTOmax*. O ganho no MOS é observado diminuindo-se o valor do *RTOmax* até cerca de 0,1 s. Abaixo disso não se observa mais ganho no MOS. Em alguns casos ocorre até uma leve queda no MOS abaixo de 0,1 s, o que sugere este valor como um ponto ótimo para o *RTOmax* para este cenário especificamente. De acordo com o gráfico, diminuir o *RTOmax* de 10 s para 0,1 s trouxe o MOS médio das chamadas de um valor impraticável de cerca de 1,5 para um valor aceitável de 3,3. Isso sugere que mesmo em casos de grandes perdas de pacotes na rede o uso do SCTP pode ser viabilizado para transporte de tráfego VoIP se o *RTOmax* for drasticamente reduzido.

Outra observação em relação ao gráfico é que as curvas referentes ao $PMR = 1$ estão levemente acima das curvas de $PMR = 0$. Esse também foi um resultado padrão deste cenário.

Uma explicação para esse fato é que a rápida reação a perdas propiciada pelo $PMR = 0$ também ocorre com $PMR = 1$ quando o *RTO* está sob limites estreitos. Como o uso do $PMR = 1$ resulta em uma menor chance de ocorrência de estado dormente, a combinação dessas duas propriedades é uma explicação para a leve superioridade apresentada pelo $PMR = 1$ neste cenário. Entretanto, a diferença é muito pequena, e dada a pequena abrangência de casos simulados neste cenário, não se pode afirmar qual dos dois valores é a melhor opção.

Quando as perdas não são intensas o ganho alcançado pela redução do *RTOmax* é muito menor, conforme mostra o gráfico da figura 4.18. Nesse caso o atraso é o mesmo do caso da figura 4.17 em ambos os caminhos (25 ms) mas as perdas são moderadas (2% em ambos). Apesar do ganho ser menos evidente, ele está presente para valores menores de *RTOmax*. Novamente percebe-se que em 0,1 s está o ponto ótimo para *RTOmax*. Não só pelo valor médio de MOS maior, mas também pela diminuição dos intervalos de confiança entorno desse ponto, o que indica um comportamento mais regular da qualidade das chamadas.

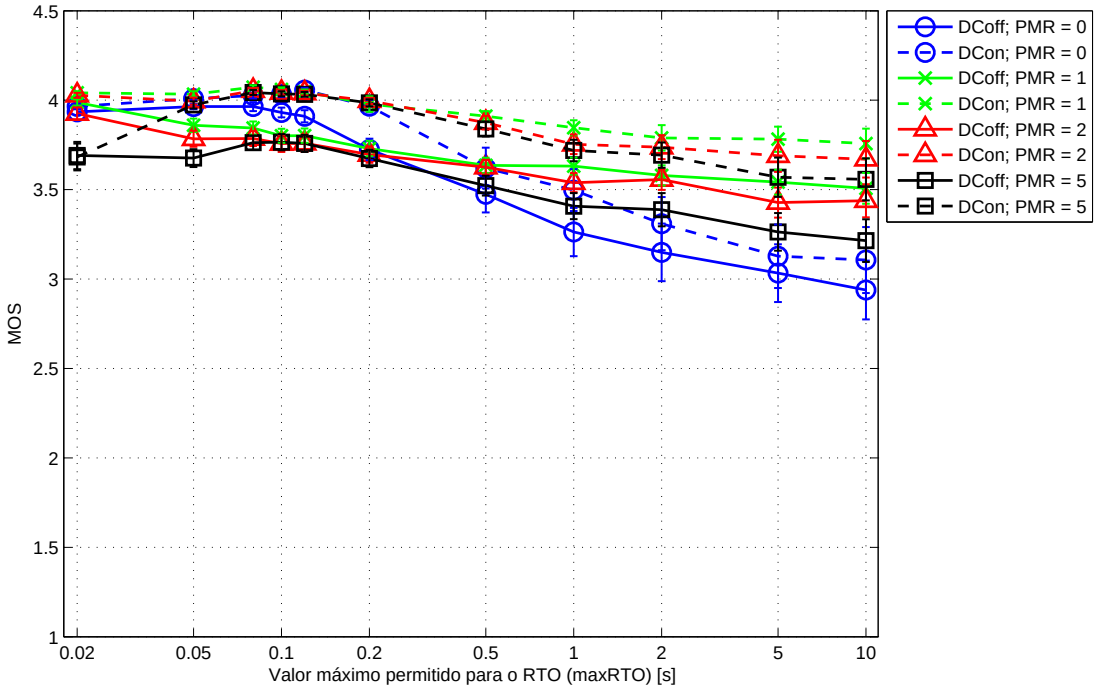

Figura 4.18: Cenário 2. Média do MOS com IC de 95% para casos de: atraso = 25ms e perdas = 2% no caminho 0; atraso = 25ms e perdas = 2% no caminho 1. *RTOmin* = 20ms

Ainda na figura 4.18 percebe-se a leve vantagem que o $PMR = 1$ leva em relação aos demais valores de *PMR*. Mesmo assim a diferença, especialmente para $RTOmax = 0{,}1$ s, é muito pequena e não permite assegurar uma tendência. Outra observação é que o ganho de MOS com a diminuição do *RTOmax* para os casos de $PMR = 0$ é o mais acentuado dentre todos

os casos simulados.

Na figura 4.19 é apresentado um caso no qual os atrasos são moderados e iguais nos dois caminhos. Já as perdas nos caminhos 0 e 1 são 2% e 10%, respectivamente. Novamente o *RTOmax* = 0,1 s é o ponto de máximo nesse gráfico do MOS. As curvas referentes aos valores de *PMR* maiores do que 0 e sem *delay-centric* apresentam valores maiores de MOS do que as demais. Isso porque esses são os casos que tendem a manter o tráfego no caminho 0, que nesse caso é o melhor por apresentar menores perdas. A medida que diminui o *RTOmax* essa vantagem também diminui até o ponto de máximo, no qual as diferenças se tornam pouco significativas.

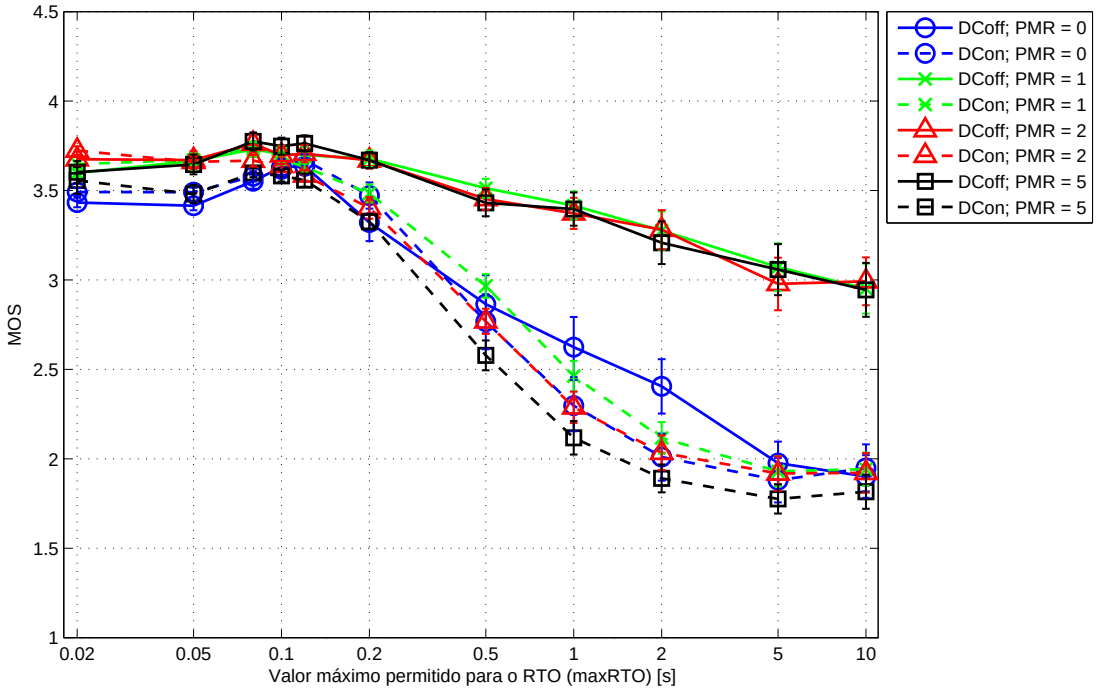

Figura 4.19: Cenário 2. Média do MOS com IC de 95% para casos de: atraso = 25ms e perdas = 2% no caminho 0; atraso = 25ms e perdas = 10% no caminho 1. *RTOmin* = 20ms

O contrário acontece no caso da figura 4.20. Nesse gráfico a diminuição do *RTOmax* aumenta as diferenças entre os casos simulados. Esse gráfico trata de casos nos quais ambos os caminhos apresentam atrasos e perdas elevados. Para *RTOmax = 0,1*, os casos que fazem uso do *delay-centric* conseguem alcançar um MOS de quase 3. Esse pode ser considerado, em alguns casos, um valor ainda aceitável de qualidade. Portanto pode-se afirmar que a escolha adequada do valor de *RTOmax* pode viabilizar uma chamada VoIP para usuários multi-abrigados mesmo em situações extremamente desfavoráveis (nesse caso com uso conjunto do *delay-centric*).

Em relação ao *RTOmin*, observou-se que ele não apresenta infuência significativa sobre

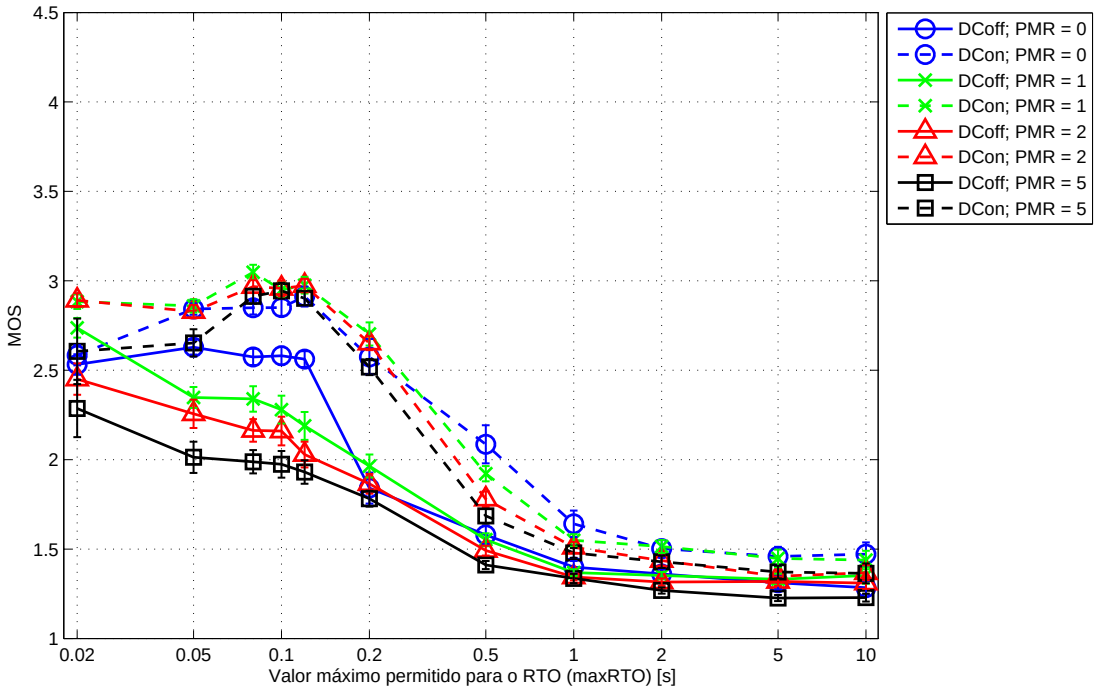

Figura 4.20: Cenário 2. Média do MOS com IC de 95% para casos de: atraso = 75ms e perdas = 10% no caminho 0; atraso = 75ms e perdas = 10% no caminho 1. *RTOmin* = 20ms

os valores de MOS para todos os casos simulados. A figura 4.21 mostra um caso similar ao da figura 4.17. A única diferença é que no caso da figura 4.21 o *RTOmin* = 0,1 s. Portanto, só são possíveis valores de *RTOmax* iguais ou maiores do que 0,1 s, pois não faria sentido que o limite superior do *RTO* fosse menor do que seu limite inferior.

Comparando os gráficos das figuras 4.21 e 4.17 é possível perceber que a única diferença é a abrangência de valores do *RTOmax*, pois no caso de *RTOmin* = 0,1 s, esse também é o menor valor de *RTOmax* que pode ser simulado. As curvas dos dois gráficos coincidem (estatisticamente), evidenciando a similaridade dos casos. Isso foi observado para todos os casos de diferentes *RTOmin* simulados. Portanto o valor do *RTOmin* não influencia na qualidade da chamada VoIP. Mesmo assim, esse parâmetro deve ser configurado de forma a permitir o valor desejado de *RTOmax*.

A tabela 4.3 contém as médias do MOS das simulações englobando todos os casos de atraso e perdas apenas para *RTOmin* = 0,02 s. Cada valor de MOS nessa tabela corresponde a um valor diferente de *RTOmax* (sentido das colunas) e de *PMR* (sentido das linhas) com ou sem *delay-centric*.

Essa tabela mostra que, em média, considerando valores tanto baixos como altos de atrasos e perdas em qualquer combinação, baixos valores de *RTOmax* (entorno de 0,1 s ou

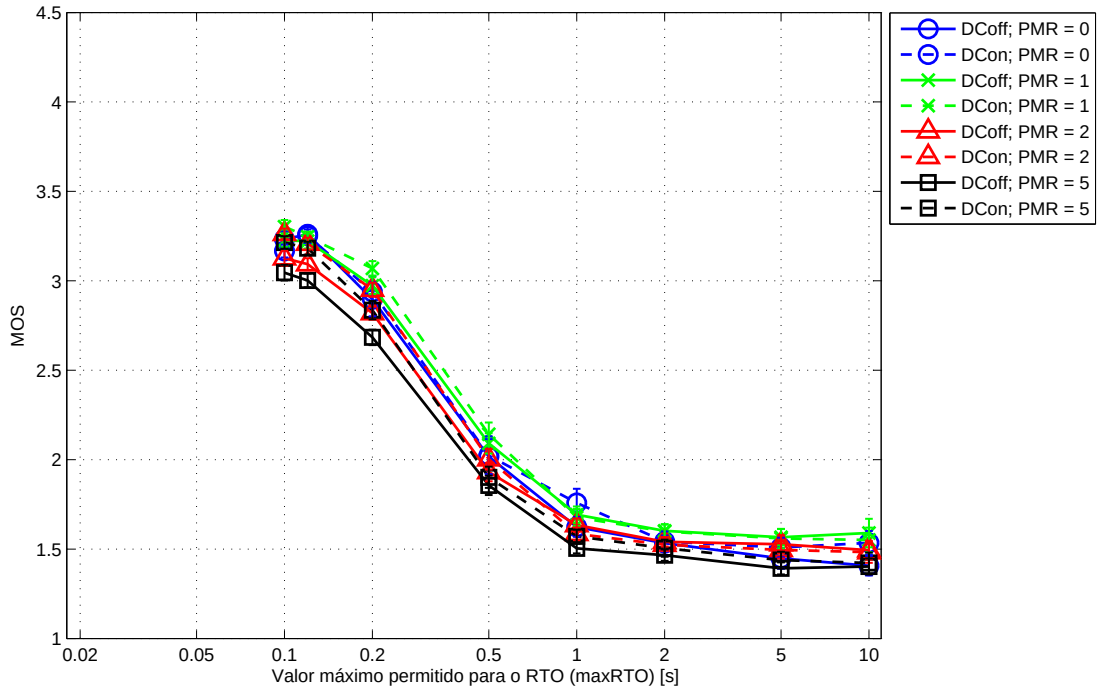

Figura 4.21: Cenário 2. Média do MOS com IC de 95% para casos de: atraso = 25ms e perdas = 10% no caminho 0; atraso = 25ms e perdas = 10% no caminho 1. *RTOmin* = 100ms

Tabela 4.3: Médias do MOS de todos os casos de atraso e perdas simulados no cenário 2 para *RTOmin* de 0,02 s classificadas por *RTOmax*, *PMR* e estados do *delay-centric*.

***RTOmax* (s)**	***Delay-centric* DESLIGADO**				***Delay-centric* LIGADO**			
	Valor de *PMR*				**Valor de *PMR***			
	0	**1**	**2**	**5**	**0**	**1**	**2**	**5**
10	2,359	2,989	2,984	2,741	2,785	3,015	2,953	2,875
5	2,383	3,006	2,994	2,761	2,784	3,015	2,956	2,878
2	2,497	3,043	3,028	2,845	2,881	3,088	3,017	2,936
1	2,606	3,084	3,059	2,926	3,050	3,217	3,126	3,027
0,5	2,771	3,155	3,138	3,044	3,321	3,454	3,343	3,241
0,2	3,124	3,282	3,270	3,194	3,700	3,743	3,703	3,646
0,12	3,481	3,291	3,282	3,197	3,807	3,827	3,798	3,776
0,1	3,525	3,324	3,307	3,216	3,773	3,843	3,811	3,788
0,08	3,559	3,366	3,322	3,259	3,770	3,873	3,834	3,795
0,05	3,491	3,428	3,296	3,247	3,682	3,826	3,807	3,714
0,02	3,413	3,627	3,606	3,278	3,455	3,764	3,834	3,430

0,08 s) proporcionam uma melhor qualidade para o tráfego VoIP em relação a valores maiores de *RTOmax*. O emprego do *delay-centric* ajudou também nesse sentido. Além disso, percebe-se que com *delay-centric* desligado os casos de $PMR = 0$ obtiveram uma média superior a dos demais valores de *PMR*. Por outro lado, nos casos em que o *delay-centric* estava ligado o $PMR = 1$ foi o que apresentou os mais altos valores de MOS, sendo estes inclusive os maiores

dentre todos os valores da tabela.

A tabela 4.4 tem o mesmo formato da tabela anterior, mas com os valores representando as médias dos MOS de todas as simulações apenas para os casos de perdas de 10%. Todos os casos de atraso em todas as combinações foram considerados na tabela 4.4. Conforme já mencionado, as vantagens da diminuição do valor de *RTOmax* ficam mais evidentes em casos de altas perdas. Portanto, em relação a tabela 4.3, na tabela 4.4 percebe-se uma variação maior de valores no sentido das colunas, com os maiores ocorrendo nos casos de *RTOmax* entorno de 0,08 s. No mais, as observações em relação a esta tabela se mantém as mesmas daquelas já mencionadas para a tabela anterior.

Tabela 4.4: Médias do MOS para casos de 10% de perdas, *RTOmin* de 0.02 s e todos os valores de atraso. As médias estão classificadas por *RTOmax*, *PMR* e estado do *delay-centric*.

***RTOmax* (s)**	***Delay-centric* DESLIGADO**				***Delay-centric* LIGADO**			
	Valor de *PMR*				**Valor de *PMR***			
	0	**1**	**2**	**5**	**0**	**1**	**2**	**5**
10	1,391	1,517	1,497	1,418	1,547	1,603	1,520	1,460
5	1,402	1,521	1,499	1,400	1,536	1,578	1,512	1,471
2	1,463	1,547	1,525	1,462	1,577	1,612	1,554	1,527
1	1,561	1,646	1,607	1,538	1,734	1,778	1,654	1,602
0,5	1,924	2,003	1,899	1,783	2,249	2,280	2,041	1,900
0,2	2,613	2,683	2,610	2,502	3,029	3,066	2,985	2,843
0.12	3,006	2,880	2,776	2,739	3,222	3,255	3,215	3,190
0,1	3,019	2,934	2,826	2,765	3,177	3,282	3,236	3,219
0,08	3,037	3,001	2,863	2,767	3,177	3,329	3,264	3,211
0,05	2,909	3,020	2,966	2,679	3,069	3,311	3,276	3,100
0,02	2,895	3,139	3,094	3,059	2,951	3,240	3,375	3,150

Tanto o *RTOmin* como, e principalmente, o *RTOmax* são parâmetros do SCTP que também apresentam uma forte influência sobre seu mecanismo de controle de congestionamento. Ao diminuir os valores desses parâmetros o controle de congestionamento do SCTP se torna menos atuante, pois seus tempos de espera para envio de dados se tornam menores. Dessa forma pode-se dizer que utilizar valores muito pequenos de *RTOmax* (e, por conseqüência, de *RTOmin*) torna o tráfego gerado pelo SCTP parecido com aquele produzido pelo UDP, conforme demonstrou Park et al. [33]. A diferença é que o SCTP ainda realiza trocas de rota, ao contrário do que aconteceria para um tráfego UDP. Nas simulações realizadas neste trabalho, quando $RTOmax = 0{,}02$ s pode-se considerar que o controle de congestionamento não apresenta mais nenhuma influência sobre o envio de dados, uma vez que o intervalo de geração dos pacotes VoIP é também de 0,02 s (20 ms) para o codec utilizado.

5 CONCLUSÕES

Neste trabalho foi estudado o desempenho do SCTP juntamente com o algoritmo de seleção automática de rotas baseado no menor atraso (*delay-centric*) para o tráfego VoIP (G.711 com PLC). Os cenários simulados contemplavam casos de usuários multi-abrigados cujas rotas continham tráfego de fundo aleatório (atraso) e perdas em rajada variáveis. Os principais parâmetros avaliados quanto à influência na qualidade de chamadas VoIP foram o *PMR* e o *RTOmax*, que representam respectivamente o limiar de sensibilidade a perdas do mecanismo de detecção de falhas padrão do SCTP e o limite superior do parâmetro *RTO*. A qualidade da chamada de cada simulação era medida usando o MOS estimado a partir do E-model.

O primeiro cenário simulava dois terminais que estabeleciam uma associação SCTP através de duas rotas (multi-abrigados). O tráfego VoIP era unidirecional e encontrava um enlace contendo tráfego de fundo que causava uma fila M/D/1. As perdas em rajadas seguiam o modelo de Gilbert. Através de parâmetros de entrada era possível configurar a simulação para diferentes valores de atraso, perdas e *PMR*, além de permitir ativar ou não o algoritmo do *delay-centric*. Todos os casos foram simulados com 100 sub-faixas diferentes do RNG com o intuito de se obter valores médios de MOS significativos.

O segundo cenário era semelhante ao primeiro. A diferença principal foi a inclusão dos parâmetros *RTOmin* e *RTOmax* como argumentos de entrada. Dessa forma foram simulados alguns dos casos do cenário anterior, mas para cada um foram avaliados diversos valores de *RTOmax*.

O uso do *delay-centric* traz um aumento significativo de qualidade em cenários envolvendo atrasos moderados ou altos. Esse mecanismo evita os momentos de pico de tráfego através do constante monitoramento do $SRTT$ nas rotas disponíveis e redirecionamento do fluxo VoIP, efetivamente reduzindo o atraso médio experimentado pelo fluxo VoIP. Quando o atraso médio nos dois caminhos é semelhante, o *delay-centric* aumenta consideravelmente a frequência de trocas de rota, enquanto que em casos de assimetria em relação ao atraso, ele mantém o fluxo durante a maior parte do tempo no caminho de menor atraso. Isso pode ter

efeitos deletérios caso os caminhos também apresentem altos níveis de perdas.

Em relação aos valores de *PMR*, para o primeiro cenário simulado o valor que permite um maior valor de MOS é $PMR = 0$. Com esse valor a reatividade do SCTP a perdas é alta, pois basta uma perda detectada para que o SCTP torne o caminho inativo e redirecione o fluxo de dados por outro. Esse comportamento se mostrou, numa média, tanto mais benéfico para a qualidade da chamada quanto maiores eram as perdas presentes nos caminhos. Já para o segundo cenário o uso do $PMR = 1$ touxe uma sutil vantagem sobre o $PMR = 0$. No entanto, essa diferença não permite determinar qual valor é o mais adequado ao tráfego VoIP. O que pode ser afirmado é que quando os limites do *RTO* são altos ($RTOmax \geq 10s$) é mais vantajoso o uso de $PMR = 0$. Já para baixos valores de *RTOmax* ($RTOmax < 1s$) tanto $PMR = 0$ como $PMR = 1$ são igualmente interessantes se combinados com o uso do *delay-centric*.

Perdas podem causar situações de interrupção de transmissão dos dados pelo SCTP, causando prejuízo não apenas pelas perdas propriamente ditas, mas também por uma demora na retomada da transmissão e pelo descarte no *buffer-anti-jitter* devido ao alto atraso sofrido pelos pacotes logo após essa retomada. Essas interrupções são devidas ao efeito do aumento excessivo do *RTO* que para altos valores de *PMR* permitia uma espera excessivamente longa para a melhora do caminho, e nos casos de baixo *PMR* retardava o término do estado dormente da associação.

A diminuição do *RTOmax* ajuda a evitar essas interrupções, trazendo ganho (em média) para todos os casos simulados no cenário 2 em que haviam perdas moderadas ou altas, independentemente do estado do *delay-centric* ou valor do *PMR*. Para esse cenário em específico há um valor ótimo entorno de $0{,}1s$ para o *RTOmax*. Quando as perdas são altas, este valor proporciona os maiores valores de MOS e uma diferença expressiva em relação ao MOS obtido com valores mais altos de *RTOmax*, tais como de $1s$ ou maiores. Apenas para perdas muito baixas é que não foi possível detectar o efeito benéfico da diminuição *RTOmax*.

Dos resultados obtidos pode-se concluir que o uso de valores menores tanto de *PMR* quanto de *RTOmax* propiciam uma melhor qualidade para o transporte de dados sensíveis ao atraso, tais como chamadas VoIP. O uso do *delay-centric* também traz vantagens nesse sentido na grande maioria dos casos, que também são os mais comumente encontrados na prática. A combinação de $PMR = 0$ ou $PMR = 1$, juntamente com um $RTOmax = 0{,}1s$ e com o *delay-centric* ativado resultou, nos cenários simulados, nos maiores valores de MOS.

Os dois mecanismos principais estudados no presente trabalho, o mecanismo de detecção de falhas do SCTP e o *delay-centric*, atuam simultaneamente e proporcionam melhora na qualidade para o tráfego VoIP na grande maioria das situações. No entanto, como cada

mecanismo é sensível apenas ou ao atraso ou a perdas, ocorrem situações em que não há concordância em suas atuações. O emprego de um algoritmo que possa conhecer as condições gerais de cada rota e controlar o uso e parâmetros dos dois mecanismos de troca de rotas poderia resultar em um aproveitamento ainda mais eficiente das rotas. Outra idéia nesse sentido é a avaliação e aprimoramento de um mecanismo de troca de rotas baseado no próprio MOS estimado para cada caminho, conforme proposto por Fitzpatrick et al. [8] . Essas abordagens teriam a vantagem de levar em conta, num mesmo algoritmo de decisão, tanto atraso quanto perdas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] KELLY, A. et al. Delay-centric handover in SCTP over WLAN. *Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE*, 2004.

[2] FITZPATRICK, J.; MURPHY, S.; MURPHY, J. An approach to transport layer handover of VoIP over WLAN. *IEEE*, 2005.

[3] GAVRILOFF, I. *Análise de aspectos envolvidos no mecanismo de seleção de caminho baseado em atraso para sistemas multiabrigados utilizando SCTP*. Dissertação (Mestrado) — UFPR, Curitiba, Brazil, March 2009.

[4] EDDY, W. At what layer does mobility belong? *Communications Magazine, IEEE*, v. 42, n. 10, p. 155–159, 2004. ISSN 0163-6804.

[5] STEWART, R. R. et al. Stream control transmission protocol. *IETF RFC 2960*, October 2000.

[6] NOONAN, J. et al. Simulations of multimedia traffic over SCTP modified for delay-centric handover. *5th World Wireless Congress (B3G)*, 2004.

[7] FRACCHIA, R. et al. WiSE: Best-path selection in wireless multihoming environments. *IEEE Transactions on Mobile Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 6, n. 10, p. 1130–1141, 2007. ISSN 1536-1233.

[8] FITZPATRICK, J. et al. Using cross-layer metrics to improve the performance of end-to-end handover mechanisms. *Computer Communication Preprint Online*, Elsevier, v. 10.1016, p. 13 Pages, 2009.

[9] KASHIHARA, S. et al. Path selection using active measurement in multi-homed wireless networks. *IEEE Proceedings of the 2004 International Symposium on Applications and the Internet (SAINT 04)*, 2004.

[10] FUNASAKA, J. et al. A study on primary path switching strategy of SCTP. *Proceedings on Autonomous Decentralized Systems, 2005. ISADS*, 2005.

[11] BUDZISZ, L. et al. An analytical estimation of the failover time in SCTP multihoming scenarios. *Wireless Communications and Networking Conference (WCNC)*, p. 3929 –3934, 2007. ISSN 1525-3511.

[12] QIAO, Y. et al. Performance analysis of multi-homed transport protocols with network failure tolerance. *IET Communications*, IET, v. 2, p. 336–345, 02/2008 2008.

[13] CARO, A. L.; AMER, P. D.; STEWART, R. R. Rethinking end-to-end failover with transport layer multihoming. *Annals of Telecommunications*, 2006.

[14] FALLON, S. et al. SCTP switchover performance issues in WLAN environments. *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, Performance Engineering Lab, Las Vegas,Nevada, p. 564 –568, 2008. ISSN 0197-2618.

[15] FALLON, S. et al. An adaptive optimized RTO algorithm for multi-homed wireless environments. *Wired/Wireless Internet Communications - Lecture Notes in Computer Science*, v. 5546, p. 133–145, 2009.

[16] STEWART, R. et al. Stream control transmission protocol (SCTP) specification errata and issues. *IETF RFC 4460*, IETF, n. 4460, April 2006. Disponível em: <http://www.ietf.org/rfc/rfc4460.txt>.

[17] STEWART, R. et al. *Stream Control Transmission Protocol (SCTP) Partial Reliability Extension*. IETF, May 2004. RFC 3758. (Request for Comments, 3758). Disponível em: <http://www.ietf.org/rfc/rfc3758.txt>.

[18] STEWART, R. R.; XIE, Q. *Stream Control Transmission Protocol (SCTP) - A reference Guide*. [S.l.]: Addison-Wesley, 2001.

[19] HASSLINGER, G.; HOHLFELD, O. The gilbert-elliott model for packet loss in real time services on the internet. 2008.

[20] WANG, H. S.; MOAYERI, N. Finite-state markov channel-a useful model for radio communication channels. *Vehicular Technology, IEEE Transactions on*, v. 44, n. 1, p. 163 –171, feb. 1995. ISSN 0018-9545.

[21] BARFORD, P.; CROVELLA, M. Generating representative web workloads for network and server performance evaluation. *Proceedings of the ACM SIGMETRICS*, 1998.

[22] KARAGIANNIS, T.; MOLLE, M.; FALOUTSOS, M. Long-range dependence: Ten years of internet traffic modeling. *IEEE INTERNET COMPUTING*, 2004.

[23] LI, M.; LIM, S. Modeling network traffic using generalized cauchy process. *Physica A: Statistical Mechanics and its Applications*, v. 387, n. 11, p. 2584 – 2594, 2008. ISSN 0378-4371.

[24] VISHWANATH, K.; VAHDAT, A. Swing: Realistic and responsive network traffic generation. *Networking, IEEE/ACM Transactions on*, v. 17, n. 3, p. 712 –725, jun. 2009. ISSN 1063-6692.

[25] BANKS, J. et al. *Discrete-Event System Simulation*. 3rd. ed. [S.l.]: Prentice-Hall, 2000.

[26] ITU-T. Methods for subjective determination of transmission quality. *ITU-T Recommendation P.800*, 1996.

[27] ITU-T. The e-model, a computational model for use in transmission planning. *G.107*, 2000.

[28] ITU-T. Estimates of Ie and Bpl parameters for a range of CODEC types. *SG12 D106*, January 2003.

[29] ITU.T. Pulse code modulation (PCM) of voice frequencies. *G.711*, November 1988.

[30] ITU-T. A high quality low-complexity algorithm for packet loss concealment with G.711. *G.711 appendix I*, September 1999.

[31] BHAGWAT, P. et al. Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs. *Wirel. Netw.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 3, n. 1, p. 91–102, 1997. ISSN 1022-0038.

[32] ITU-T. Provisional planning values for the equipment impairment factor Ie and packet-loss robustness factor Bpl. *G.113 - Appendix 1*, 2002.

[33] PARK, H. L.; KIM, M.; KIM, J.-S. Evaluation of stream control transmission protocol as a transport for VoIP over WLAN. *Advanced Communication Technology, The 9th International Conference on*, v. 3, p. 1613 –1617, 2007. ISSN 1738-9445.

ANEXO A – TABELA DE VALORES PADRÃO PARA OS PARÂMETROS DO E-MODEL ENVOLVIDOS NO CÁLCULO DO FATOR *R*

Nome	unidade	sigla	valor
Send Loudness Rating	dB	SLR	8
Receive Loudness Rating	dB	RLR	2
Overall LR OLR=SLR+RLR	dB	OLR	10
Sidetone Receive Side	dB	$STMR$	15
D-Factor, Receive Side	dB	D_r	3
Listener Side Tone, STMR+Dr=	dB	$LSTR$	18
D-Factor, Send Side	dB	D_s	3
Talker Echo Loudness Rating	dB	$TELR$	65
Weighted Echo Path Loss	dB	$WEPL$	110
Mean One Way Delay	ms	T	0
Round Trip Delay	ms	T_r	0
One Way absolute Delay	ms	T_a	0
Equipment Impairment Factor	-	I_e	0
Expectation Factor	-	A	0
Packet Loss Robustness Factor	-	B_{pl}	1
Random Packet Loss Probability	%	P_{pl}	0
Room Noise, Receive Side	dB(A)	P_r	35
Room Noise, Send Side	dB(A)	P_s	35
Circuit Noise	dBm	N_c	-70
Noise Floor	dBmp	N_{for}	-64
Quantizing Distortion Unit	-	QDU	1

Tabela A.1: Valores padrão dos parâmetros do E-model para redes IP. No caso do codec G.711 com PLC, que foi o utilizado neste trabalho, deve-se utilizar $I_e = 0$ e $B_{pl} = 25{,}1$.

ANEXO B – *SCRIPT* OTCL UTILIZADO NAS SIMULAÇÕES DO CENÁRIO 2

```
source ../tcl/topologia_v2.tcl
source ../tcl/dadossctp_v2.tcl
source ../tcl/otimizador.tcl
source ../tcl/progressmonitor.tcl
source ../tcl/GerTrafUDPMark.tcl

puts -nonewline "Parametros de simulacao ($argc): $argv ... "

#==================== PARÂMETROS DE ENTRADA

set estadoDC [lindex $argv 0];    #"DCoff" ou "DCon"
set atr(0) [lindex $argv 1];      #em SEGUNDOS [s]
set atr(1) [lindex $argv 2];      #em SEGUNDOS [s]
set vPMR [lindex $argv 3];        #inteiro
set perdas(0) [lindex $argv 4];   #em valor absoluto
set perdas(1) [lindex $argv 5];   #em valor absoluto
set vRTOmin [lindex $argv 6];     #em SEGUNDOS [s]
set vRTOmax [lindex $argv 7];     #em SEGUNDOS [s]
set SubStr [lindex $argv 8];      #inteiro



#==================== PARÂMETROS CONSTANTES

set Ncaminhos 2;          #numero de caminhos
set Nfluxos 1;            #numero de fluxos

set LBenlaces 1000000; #taxa de transmissão dos enlaces
set APenlaces 0.01;       #Atraso de propagação dos enlaces [s]
set vHBint 0.1;           #em SEGUNDOS [s]
set TpktVoIP 160;         #em BYTES
set LBVoIP 64000;         #em bps
set OHVoIP 48;            #OverHead do pacote VoIP [BYTES]
set TpktTF 500;           #em BYTES
set grao 0.005;           #granularidade temporal do modelo de Gilbert (dt)
set NivelRaj 100;         #recomendado entre 50 e 200. Valores maiores indicam rajadas de
                          #erros em média mais longas.
```

```
set DuraSimu 120;       #Duração da chamada VoIP em segundos


#=== GARANTINDO COERENCIA DOS VALORES DE RTOmin, RTOmax e RTOini

if { $vRTOmin > $vRTOmax } { exit 0 }
if { $vRTOmax >= 0.5 } { set vRTOini 0.5
} else { set vRTOini $vRTOmax }


#========= ORGANIZANDO PASTAS E NOMES DE ARQUIVOS

set caminho "./tr_c19_${estadoDC}_${atr(0)}_${atr(1)}/"
set nomeArq "tr_c19_${estadoDC}_${atr(0)}_${atr(1)}_${vPMR}_${perdas(0)}_\
${perdas(1)}_${vRTOmin}_${vRTOmax}_${SubStr}"
set ArqLOG "log_c19_${estadoDC}_${atr(0)}_${atr(1)}_${vPMR}"


if { [file exist "$caminho$ArqLOG.log"] } {
        set logF [open "$caminho$ArqLOG.log" r]
        set LogData [read $logF]
        close $logF

        # Procura se esse caso ja foi simulado
        set LogSimus [split $LogData "\n"]
        foreach line $LogSimus {
                if { $line == "$caminho$nomeArq.atr" } {
                        exit 0
                }
        }
} else {
        file mkdir "$caminho"
}


#Trace de atrasos dos pacotes de dados SCTP (pra ser usado adiante)
set datatr [open "$caminho$nomeArq.atr" w]


#============ INSTANCIANDO SIMULADOR
set ns [new Simulator]

$ns color 0 blue
$ns color 1 red
$ns color 2 green
$ns color 3 yellow
$ns color 4 magenta
$ns color 5 black
$ns color 6 orange
$ns color 7 purple
```

```
$ns color 8 gold
$ns color 9 brown


#===== CRIANDO RNG E CONFIGURANDO A SUBSTREAM CORRETA DO RNG
set RNG [new RNG]
for {set s 0} {$s < $SubStr} {incr s} { $RNG next-substream }


#=== CRIANDO CAMINHOS E CONFIGURANDO TRAFEGOS DE FUNDO E
#=== PERDAS EM RAJADAS

#— calculando variavel que determina os trafegos de fundo

if { ($Nfluxos/$Ncaminhos) <= 1 } {
        set kinterm [expr ($LBenlaces - ($LBVoIP*(($TpktVoIP +\
        $OHVoIP)/$TpktVoIP)))/($TpktTF*8)]
} else {
        set kinterm [expr ($LBenlaces - ($LBVoIP*(($TpktVoIP +\
        $OHVoIP)/$TpktVoIP)*($Nfluxos/$Ncaminhos)))/($TpktTF*8)]
}

if { ($kinterm <= 0) } {
        puts "ERRO: Rede não comporta o fluxo VoIP solicitado! (Taxa de dados\
        VoIP) > (taxa de transmissão dos caminhos somadas). Abortando simulacao."
        exit 0;
}

for {set c 0} {$c < $Ncaminhos} {incr c} {

        #— criando caminhos
        set topo($c) [new Topologia $ns 4]
        $topo($c) setBW $LBenlaces
        $topo($c) setDelay $APenlaces
        $topo($c) setQueueLimit 50
        $topo($c) build

        lappend listaTopo $topo($c)



        #— criando GERADORES DE TRAFEGO DE FUNDO
        set ATRmedTT($c) [expr round(1000*($atr($c) + 3*$APenlaces + 3*(($TpktVoIP +\
        $OHVoIP)/$LBenlaces)))]

        set IntPkt($c) [expr ($atr($c)*2*$kinterm + 1)/($atr($c)*2*$kinterm*$kinterm)]

        set trafFundo($c) [new Trafego/UDP/Mark $ns [$topo($c) getIdxNode 1] [$topo($c)\
        getIdxNode 2] $TpktTF $IntPkt($c) [expr $Nfluxos] $RNG]

        append TXTinfoCAM "CAMINHO ${c}: atraso medio total teorico =\
```

```
$ATRmedTT($c)ms; "


#— Trace de atrasos dos pacotes de dados SCTP neste caminho
set delay_tracer($c) [new Topologia/TraceDelay $ns $topo($c) $c $datatr]



#— criando GERADORES DE ERROS ALEATORIOS EM RAJADAS

#— CASO SEM PERDAS (0%): pular modelo de perdas
if {$perdas($c) == 0} {
        append TXTinfoCAM "SEM PERDAS (0%).\n"
        continue
}

append TXTinfoCAM "COM [expr round($perdas($c)*1000)]/mil de PERDAS\
(media).\n"


#— MODELO DE GILBERT CASO HAJAM PERDAS:
#— Calculo dos parametros adequados

#— LIMITES RECOMENDADOS: 0.02 < r < 0.2 ; 0.0005 < p < 0.005

#— PARAMETRO h = probabilidade de NAO perder pacote quando no
#— estado com perdas
if {$perdas($c) < 0.05} {
        set h($c) 0.2
} elseif {$perdas($c) <= 0.15} {
        set h($c) [expr 0.2 - 0.1*(($perdas($c) - 0.05)/0.1)]
} elseif {$perdas($c) <= 0.5} {
        set h($c) 0.1
} elseif {$perdas($c) <= 0.7} {
        set h($c) [expr 0.1 - 0.1*(($perdas($c) - 0.5)/0.2)]
} else {
        set h($c) 0
}

#— PARAMETRO p = probabilidade de mudar para estado COM perdas
set p($c) [expr (1/($NivelRaj*(1 - $h($c))))*sqrt($perdas($c))]

#— PARAMETRO r = probabilidade de mudar para estado SEM perdas
set r($c) [expr $p($c)*(((1 - $h($c))/($perdas($c))) - 1)]


#— CONFIGURANDO MODELOS DE PERDAS

#— criando variaveis aleatorias
set randomica_A($c) [new RandomVariable/Uniform]
$randomica_A($c) use-rng $RNG
```

```
        set randomica_B($c) [new RandomVariable/Uniform]
        $randomica_B($c) use-rng $RNG

        #— Criando o error model bom - sem perdas
        set em_A($c) [new ErrorModel]
        $em_A($c) unit pkt
        $em_A($c) set rate_ 0
        $em_A($c) ranvar $randomica_A($c)
        $em_A($c) drop-target [new Agent/Null]

        #— Criando o error model ruim - com perdas
        set em_B($c) [new ErrorModel]
        $em_B($c) unit pkt
        $em_B($c) set rate_ [expr 1 - $h($c)]
        $em_B($c) ranvar $randomica_B($c)
        $em_B($c) drop-target [new Agent/Null]


        #— Vetor de estados (error models)
        set mm_states($c) [list $em_A($c) $em_B($c)]

        #— Duraçoes dos estados
        set mm_periods($c) [list $grao $grao]

        #— Matriz de transição de estados
        set mm_transmx($c) [list [list [expr 1 - $p($c)]     $p($c)] [list     $r($c)\
        [expr 1 - $r($c)]]]

        set mm_trunit($c) pkt

        #— Utilizando transições baseadas em tempo
        set mm_sttype($c) time
        set mm_nstates($c) 2
        set mm_nstart($c) [lindex $mm_states($c) 0]

        set MSem($c) [new ErrorModel/MultiState $mm_states($c) $mm_periods($c)\
        $mm_transmx($c) $mm_trunit($c) $mm_sttype($c) $mm_nstates($c) $mm_nstart($c)]


        #conectando MULTImodels aos enlaces
        $ns link-lossmodel $MSem($c) [$topo($c) getIdxNode 2] [$topo($c) getIdxNode 3]
}

#puts -nonewline $TXTinfoCAM


#=================== CRIANDO AGENTES SCTP E FLUXOS VoIP

for {set f 0} {$f < $Nfluxos} {incr f} {

        #— instanciando agentes
```

```
        set dados($f) [new DadosSCTP $ns $listaTopo]

        #— configurando agente de origem
        set agenteSrc($f) [$dados($f) getAgenteSrc]
        $agenteSrc($f) set numUnrelStreams_ 1
        $agenteSrc($f) set unordered_ 1
        $agenteSrc($f) set useDelayedSacks_ 0
        $agenteSrc($f) set sackDelay_ 0
        $agenteSrc($f) set heartbeatInterval_ $vHBint
        $agenteSrc($f) set pathMaxRetrans_ $vPMR
        $agenteSrc($f) set class_ $f
        $agenteSrc($f) set minRto_ $vRTOmin
        $agenteSrc($f) set maxRto_ $vRTOmax
        $agenteSrc($f) set initialRto_ $vRTOini
        $agenteSrc($f) set reliability_ 0
        $agenteSrc($f) set dormantAction_ 0

        #— configurando agente de destino
        set agenteDst($f) [$dados($f) getAgenteDst]
        $agenteDst($f) set heartbeatInterval_ $vHBint

        #— criando gerador CBR para simular trafego VoIP
        set cbr_voz($f) [new Application/Traffic/CBR]
        $cbr_voz($f) set packetSize_ $TpktVoIP
        $cbr_voz($f) set rate_ $LBVoIP
        $cbr_voz($f) attach−agent $agenteSrc($f)


        $dados($f) connect

        #— Ativando delay-centric se for o caso
        if {$estadoDC == "DCon"} {

                set otimiz($f) [new Otimizador $dados($f)]
                $otimiz($f) set histerese_ 0.01

                set TXTinfoFLUX "Delay-Centric ATIVADO"

        } else {
                set TXTinfoFLUX "Delay-Centric DESATIVADO"
        }

        #— incluindo fluxo para aparecer nos traces de todos os caminhos
        for {set c 0} {$c < $Ncaminhos} {incr c} {
                $delay_tracer($c) add−flow $f [$agenteSrc($f) set class_]
        }

}

#puts $TXTinfoFLUX
```

```
#============== PROCEDIMENTO DE FINALIZACAO DA SIMULACAO
proc finish {} {
        global ns datatr delay_tracer caminho nomeArq ArqLOG estadoDC atr perdas\
        vPMR vRTOmin vRTOmax SubStr
        $ns flush-trace
        close $datatr




#       puts "calculando MOS..."
        if {![catch {exec ./calcMOSv3ARQ c19 $estadoDC $atr(0) $atr(1) $vPMR $perdas(0)\
        $perdas(1) $vRTOmin $vRTOmax $SubStr "${caminho}mos_c19_${estadoDC}_\
        ${atr(0)}_${atr(1)}_${vPMR}_0.1_G.711_PLC" 0.1 G.711_PLC >@stdout} msg]} {

                set logEscr [open "$caminho$ArqLOG.log" a]
                puts $logEscr "$caminho$nomeArq.atr"
                close $logEscr

                puts "ok. MOS calculado."
        } else {
                puts stderr "\n$msg"
                puts stderr "ERRO DURANTE CALCULO DO MOS. ESSA SIMULACAO NAO IRA\
                CONSTAR NO LOG PRINCIPAL."

                set logErro [open "$caminho$ArqLOG.ERRO" a]
                puts $logErro "$caminho$nomeArq.atr"
                close $logErro

                puts stderr "Esse caso ira constar no arquivo de erros\
                $caminho$ArqLOG.ERRO"
        }

        exit 0
}

#======== MONTANDO AGENDA DA SIMULACAO COM INICIOS ALEATORIOS

set random_schedule [new RandomVariable/Uniform]
$random_schedule use-rng $RNG

#— iniciando e cessando trafegos de fundo
for {set c 0} {$c < $Ncaminhos} {incr c} {
        $ns at 1 "$trafFundo($c) start"
        $ns at [expr $DuraSimu + 21] "$trafFundo($c) stop"
}

#— iniciando e cessando trafegos VoIP
for {set f 0} {$f < $Nfluxos} {incr f} {
        #— preambulo necessario para garantir consistencia dos traces
        $ns at [expr 0.1 + (0.1*($f/$Nfluxos))] "$cbr_voz($f) start"
```

```
        $ns at [expr 0.3 + (0.1*($f/$Nfluxos))] "$cbr_voz($f) stop"

        $ns at [expr 0.5 + (0.1*($f/$Nfluxos))] "$cbr_voz($f) start"
        $ns at [expr 0.7 + (0.1*($f/$Nfluxos))] "$cbr_voz($f) stop"

        #— main traffic
        set iniAleat($f) [expr 10 + (10*[$random_schedule value])]
        $ns at $iniAleat($f) "$cbr_voz($f) start"
        $ns at [expr $iniAleat($f) + $DuraSimu] "$cbr_voz($f) stop"
}

#— fim da simulacao
$ns at [expr $DuraSimu + 30] "finish"

#— rodando simulacao

$ns run
```

ANEXO C – PROGRAMA EM C++ PARA CALCULAR O MOS DE CADA SIMULAÇÃO REALIZADA NO CENÁRIO 2

```
// Programa para calcular o MOS a partir do arquivo de trace da simulação

#include <iostream>
#include <sstream>
#include <string>
#include <cstdlib>
#include <string.h>
#include <fstream>
#include <math.h>
#include <vector>
using namespace std;

//Definição da classe para os parâmetros envolvidos no cálculo do fator R e sua inicialização
//com valores padrão
class constantesITU {

        public:
        double SLR, RLR, OLR, STMR, Dr, LSTR, Ds, TELR, WEPL, T;
        double Tr, Ta, Ie, A, Bpl, Ppl, Pr, Ps, Nc, Nfor, QDU;

        constantesITU(){
                SLR = 8; RLR = 2; OLR = 10; STMR = 15; Dr = 3; LSTR = 18; Ds = 3;
                TELR = 65; WEPL = 110; T = 0; Tr = 0; Ta = 0; Ie = 0; A = 0;
                Bpl = 1; Ppl = 0; Pr = 35; Ps = 35; Nc = -70; Nfor = -64; QDU = 1;
        }
};

//Função que realiza o cálculo do fator R segundo a G.107
double fatorR (constantesITU dados)
{
        double R, lz, Nos, Pore, Nor, Nfo, No, Ro, Nt, X1;
        double Iolr, STMRo, Ist, Q, G, Y, Z, Iq, Is, Roe;
        double TERV, Re, Idte, Rle, Idle, X2, Idd, Id, Ie_eff;

        lz = log(10.0);
```

```
Nos = dados.Ps − dados.SLR − dados.Ds − 100 + 0.004*(dados.Ps − dados.OLR\
 − dados.Ds − 14)*(dados.Ps − dados.OLR − dados.Ds − 14);
Pore = dados.Pr + 10 * log(1 + pow(10.0, (10.0 − dados.LSTR)/10.0) ) / lz;
Nor = dados.RLR − 121 + Pore + 0.008 * (Pore − 35) * (Pore − 35);
Nfo = dados.Nfor + dados.RLR;
No = 10.0 * (log(pow(10, dados.Nc/10.0) + pow(10, Nos/10.0) + pow(10, Nor/10.0)\
 + pow(10, Nfo/10.0)) / lz);
Ro = 15 − 1.5 * (dados.SLR + No);
Nt = No − dados.RLR;
X1 = dados.OLR + 0.2 * (64 + Nt);
Iolr = 20 * ( pow( (1 + pow( (X1/8.0), 8.0 )), 0.125) − X1/8 );
STMRo = −10.0 * log(pow(10, (−dados.STMR / 10.0)) + exp(−dados.T / 4.0)\
 * pow(10, (−dados.TELR / 10.0))) / lz;
Ist = 12.0 * pow(1 + pow((STMRo − 13)/6.0, 8), 0.125) − 28 * pow(1\
 + pow((STMRo + 1)/19.4, 35.0), 1.0/35.0) − 13 * pow(1 + pow((STMRo\
 − 3)/33.0, 13), 1.0/13.0) + 29;
Q = 37 − 15 * log(dados.QDU)/lz;
G = 1.07 + 0.258 * Q + 0.0602 * Q * Q;
Y = (Ro − 100)/15.0 + 46.0/8.4 − G/9.0;
Z = (46.0/30.0) − (G/40.0);
Iq = 15 * log(1 + pow(10.0, Y) + pow(10.0, Z)) / lz;
Is = Iolr + Ist + Iq;
Roe = −1.5 * (No − dados.RLR);

if (dados.T <= 100) { TERV = 6 * exp(−0.3 * dados.T * dados.T); }
else { TERV = 0; }
TERV = TERV + (dados.TELR − 40.0 * log((1 + dados.T/10.0) / (1\
 + dados.T/150.0)) / lz);

if (dados.STMR < 9) { TERV = TERV + (Ist/2.0); }

Re = 80.0 + 2.5 * (TERV − 14);
Idte = ((Roe − Re)/2.0 + sqrt((Roe − Re)*(Roe − Re)/4.0 + 100) − 1) * (1\
 − exp(−dados.T));

if (dados.STMR > 20) { Idte = sqrt(Idte * Idte + Ist * Ist); }

Rle = 10.5 * (dados.WEPL + 7) * pow(dados.Tr + 1, −0.25);
Idle = (Ro − Rle)/2.0 + sqrt((Ro − Rle) * (Ro − Rle)/4.0 + 169);

if (dados.Ta <= 100) { X2 = 0; Idd = 0; }
else {
        X2 = log(dados.Ta/100.0) / log(2);
        Idd = 25 * (pow(1 + pow(X2, 6), 1/6.0) − 3 * pow(1 + pow(X2/3.0, 6),\
 1/6.0) + 2);
}

Id = Idte + Idle + Idd;
Ie_eff = dados.Ie + (95 − dados.Ie) * dados.Ppl / (dados.Bpl + dados.Ppl);
R = Ro − Is − Id − Ie_eff + dados.A;
```

```
        if (R < 0) { R = 0; }

        return (R);
}

//Função que realiza o mapeamento do fator R em valores de MOS
double fatorMOS (double R)
{
        if (R < 0) { return(1); }
        else if (R > 100) { return(4.5); }
        else { return( 1 + 0.035 * R + R * (R - 60) * (100 - R) * 0.000007 ); }
}

//Função que determina os valores de atraso e perdas a partir do arquivo de trace
double* calcMOSv3 (const char* camArq, double pjit, const char* codec)
{

        // carrega arquivo
        fstream arq (camArq, fstream::in);

        char linha[255], ch ('a');
        int nl = 0;
        int nc = 1;

        while (ch != '\n') {
                arq.get(ch);
                if (ch == ' ') nc++;
        }
        while (!arq.eof()) {
                arq.getline(linha,255);
                nl++;
        }

        arq.clear();
        arq.seekg(0 , ios::beg);

        vector< vector< double > > matriz (nl, vector< double > (nc));

        for (int i = 0; i < nl; i++) {
                for (int x = 0; x < (nc - 0); x++) {
                        arq >> matriz[i][x];
                }
        }

        arq.close();

        //remove pacotes iniciais
        int idx = 0;

        while (idx < matriz.size()) {
                if (matriz[idx][3] < 1) { matriz.erase(matriz.begin() + idx); }
```

```
        else { idx++; }
}

//determina quais sao os fluxos presentes nos dados
vector<int> fluxos (1, (int) matriz[0][1]);
vector<int> nlinflx (1);
nlinflx[0] = 1;

int flxdif = 0;

for (int i = 1; i < matriz.size(); i++) {
        flxdif = 0;
        for (int iflx = 0; iflx < fluxos.size(); iflx++) {
                if (fluxos[iflx] != (int) matriz[i][1]) { flxdif++; }
                else { nlinflx[iflx]++; }
        }
        if (flxdif == fluxos.size()) {
                fluxos.push_back((int) matriz[i][1]);
                nlinflx.push_back(1);
        }
}

// ordena fluxos
int valtmp, smaxrep, siter, sflg = 1;

for(smaxrep = 1; (smaxrep <= fluxos.size()) && sflg; smaxrep++) {
        sflg = 0;
        for (siter=0; siter < (fluxos.size() -1); siter++) {
                if (fluxos[siter+1] < fluxos[siter]) {
                        valtmp = fluxos[siter+1];
                        fluxos[siter+1] = fluxos[siter];
                        fluxos[siter] = valtmp;
                        valtmp = nlinflx[siter+1];
                        nlinflx[siter+1] = nlinflx[siter];
                        nlinflx[siter] = valtmp;
                        sflg = 1;
                }
        }
}


//cria objeto de dados e configura parâmetros e constantes do codec G.711

double SegTint = 0.02; // % intervalo entre segmentos de VoIP [segundos]
int SegSize = 160;     // % tamanho dos segmentos VoIP [bytes]
constantesITU dados;


if (!strcmp(codec, "G.711_PLC")) {      // G.711 com PLC
        dados.Bpl = 25.1;               // robustez a perda de pacotes
        dados.Ie = 0;                   // fator de deficiência do equipamento
```

```
} else if (!strcmp(codec, "G.723")) {      // G.723.1+VAD
        dados.Bpl = 16.1;                  // robustez a perda de pacotes
        dados.Ie = 15;                     // fator de deficiência do equipamento
} else if (!strcmp(codec, "G.729")) {      // G.729A+VAD
        dados.Bpl = 19;                    // robustez a perda de pacotes
        dados.Ie = 11;                     // fator de deficiência do equipamento
} else {                                   // G.711 sem PLC
        dados.Bpl = 4.3;                   // robustez a perda de pacotes
        dados.Ie = 0;                      // fator de deficiência do equipamento
}

double* MOS = new double[fluxos.size() + 1]; //para os valores MOS de saida.
//O primeiro elemento indica o numero de elementos após ele (número de valores
// do MOS calculados)
MOS[0] = fluxos.size();


//calcula o atraso fixo, perdas e MOS para cada fluxo
for (int f = 0; f < fluxos.size(); f++) {

        // extrai linhas referentes ao fluxo em questao e coloca em matflx
        vector< vector< double > > matflx (nlinflx[f], vector< double > (nc));

        idx = 0;
        for (int i = 0; i < matriz.size(); i++) {
                if (matriz[i][1] == fluxos[f]) {
                        matflx[idx] = matriz[i];
                        idx++;
                }
        }

        //ordena linhas por tempo de saida

        sflg = 1;

        for(smaxrep = 1; (smaxrep <= matflx.size()) && sflg; smaxrep++) {
                sflg = 0;
                for (siter=0; siter < (matflx.size() -1); siter++) {
                        if (matflx[siter+1][3] < matflx[siter][3]) {
                                matflx[siter+1].swap(matflx[siter]);
                                sflg = 1;
                        }
                }
        }

        //atraso fixo e busca do primeiro pacote recebido
        double atraso_fixo;

        int SegIniDesc = 0; //segmentos VoIP iniciais descartados, caso haja (para
                            // calculo posterior)
        idx = 0;
```

```
            while ((matflx[idx][4] < 1) && (idx < matriz.size())) {
                SegIniDesc += (matflx[idx][5] - ((((int)matflx[idx][5] - 48)%SegSize)\
                 + 48))/SegSize;
                idx++;  //indice do primeiro pacote transmitido com sucesso
            }
            atraso_fixo = matflx[idx][4] - matflx[idx][3] + pjit;

            //número total de segmentos
            int NsegTot = 0, NsegDesc;

            for (int i = 0; i < matflx.size(); i++) {
                NsegTot += matflx[i][5] - ((((int)matflx[i][5] - 48)%SegSize) + 48);
            }
            NsegTot /= SegSize;

            //descartes totais de segmentos VoIP
            NsegDesc = NsegTot;
            int idxSeg = 0;

            for (int l = 0; l < matflx.size(); l++) {
                if (idxSeg > NsegTot) { break; }

                //numero de segmentos no pacote
                int pktNS = ((matflx[l][5] - ((((int)matflx[l][5] - 48)%SegSize)\
                 + 48))/SegSize);

                if (matflx[l][4] == -1) {
                    idxSeg += pktNS;
                } else {
                    for (int s = 1; s <= pktNS; s++) {
                        if (matflx[l][4] <= (matflx[idx][4] + pjit\
                         + (SegTint*(idxSeg - SegIniDesc)))) {
                            NsegDesc -= 1;
                        }
                        idxSeg += 1;
                    }
                }
            }

            //configurando parametros para calculo do MOS
            dados.Ppl = ((double)NsegDesc/(double)NsegTot)*100;      // valor em %
            dados.T = atraso_fixo * 1000;                            // valor em [ms]
            dados.Ta = dados.T;
            dados.Tr = 2 * dados.T;

            //chamando funções para calculo do MOS
            MOS[f+1] = fatorMOS(fatorR(dados));

        }
        return(MOS);
}
```

```
//função principal que utiliza os argumentos de entrada
int main (int argc, char* argv[])
{
        if (argc < 13) {
                cout << "Numero insuficiente de parametros de entrada." << endl;
                return 1;
        }


        char *cena = argv[1];
        char *DC = argv[2];
        char *atr0 = argv[3];
        char *atr1 = argv[4];
        char *PMR = argv[5];
        char *per0 = argv[6];
        char *per1 = argv[7];
        char *RTOmin = argv[8];
        char *RTOmax = argv[9];
        char *seed = argv[10];
        char *ArqDEST = argv[11];
        double pjit = atof(argv[12]);
        char *codec = argv[13];

        double *MOS;
        char *CamArq;

        stringstream montastr;
        string caminho, ArqTR, STRdestBKP;

        //prepara caminho do arquivo de trace
        montastr << "./tr_" << cena << '_' << DC << '_' << atr0 << '_' <<\
         atr1 << '/';
        caminho = montastr.str();
        montastr.str("");
        montastr << "tr_" << cena << '_' << DC << '_' << atr0 << '_' << atr1\
         << '_' << PMR << '_' << per0 << '_' << per1 << '_' << RTOmin <<\
         '_' << RTOmax << '_' << seed << ".atr";
        ArqTR = montastr.str();
        montastr.str("");

        CamArq = new char [caminho.size() + ArqTR.size() + 1];
        strcpy(CamArq, caminho.c_str());
        strcat(CamArq, ArqTR.c_str());

        //calculo do MOS
        MOS = calcMOSv3(CamArq, pjit, codec);


        //escrevendo arquivos de saida
        montastr << ArqDEST << ".bkp";
```

```
        STRdestBKP = montastr.str();
        montastr.str("");

        for (int f = 1; f <= MOS[0]; f++) {
                fstream AD (ArqDEST, fstream::out | fstream::app);
                AD << atr0 << " " << atr1 << " " << PMR << " " << per0 <<\
                 " " << per1 << " " << RTOmin << " " << RTOmax << " " <<\
                 seed << " " << (f - 1) << " " << MOS[f] << endl;
                AD.close();

                fstream ADbkp (STRdestBKP.c_str(), fstream::out | fstream::app);
                ADbkp << atr0 << " " << atr1 << " " << PMR << " " << per0\
                 << " " << per1 << " " << RTOmin << " " << RTOmax << " "\
                 << seed << " " << (f - 1) << " " << MOS[f] << endl;
                ADbkp.close();
        }
        return 0;
}
```