ANDRÉA GIORDANNA OLIVEIRA DO NASCIMENTO

COMPRESSÃO DE CABEÇALHO VISANDO A QUALIDADE DA FALA EM VOIP SOBRE REDES MESH

Manaus - AM
05 de maio de 2009

ANDRÉA GIORDANNA OLIVEIRA DO NASCIMENTO
Orientador: Edjair Mota

COMPRESSÃO DE CABEÇALHO VISANDO A QUALIDADE DA FALA EM VOIP SOBRE REDES MESH

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Informática.

ANDRÉA GIORDANNA OLIVEIRA DO NASCIMENTO

Manaus - AM
05 de maio de 2009

UNIVERSIDADE FEDERAL DO AMAZONAS

FOLHA DE APROVAÇÃO

Compressão de cabeçalho visando a qualidade da fala em VoIP sobre Redes Mesh

ANDRÉA GIORDANNA OLIVEIRA DO NASCIMENTO

Dissertação defendida e aprovada pela banca examinadora constituída por:

Dr.-Ing Edjair Mota – Orientador
Universidade Federal do Amazonas

Dr. Eduardo James Pereira Souto
Universidade Federal do Amazonas

Dr. Horácio Antonio Braga F. de Oliveira
Universidade Federal do Amazonas

Dr. Luiz Cláudio Schara Magalhães
Universidade Federal Fluminense

Manaus - AM, 05 de maio de 2009

A Deus. Aos meus pais, Noemia e Luiz Jorge,
pelo apoio incondicional em todos os momentos.
Aos meus irmãos, Joyce, Alexandre, Luiz, Junior e
Amanda, por serem a alegria da minha vida.

Agradecimentos

A Deus, pelas graças concedidas.

Ao meu orientador, Professor Edjair de Souza Mota, pela oportunidade concedida de realizar este trabalho, pelo estímulo, confiança, paciência e dedicação. Por ter dado a oportunidade e o incentivo de publicar em conferências internacionais e mostrar à comunidade científica as contribuições do nosso trabalho.

Aos meus familiares, pela paciência, apoio, carinho e ajuda em todos os momentos durante toda a minha vida, e principalmente nestes últimos dois anos.

Aos amigos do grupo de Redes de Computadores e Multimídia (GRCM) que estiveram presente durante esses dois anos, especialmente: Loide Mara, Arlen Nascimento, Gabriel Leitão, Cláudia Suzany, João Luis, Alexandre Passito, Valinda Maia, e principalmente ao Saulo Queiroz, pela ajuda e grandes discussões sobre o tema do trabalho e pelas contribuições em nossas publicações, e à Regeane Aguiar, pelo apoio e grandes contribuições com relação ao módulo do cálculo do MOS para o NS-2.

Aos colegas de mestrado, pela boa convivência, auxílio e incentivo, pelos momentos de descontração em meio às diversas atividades, durante esses dois anos.

À secretaria do DCC, pelo apoio administrativo.

Aos amigos André Venancio e Vanessa Souza, pelas conversas, bons conselhos e apoio em momentos difíceis.

À CAPES, pelo apoio financeiro.

Resumo

A tecnologia de Voz sobre IP tem sido amplamente utilizada na área de telefonia para realizar chamadas de voz através da Internet, tanto por empresas quanto por usuários domésticos. Por apresentar menor custo em comparação ao sistema de telefonia público, a tendência é que a convergência de serviços de voz e dados para um único meio se complete em breve. Portanto, é natural a preocupação com a qualidade desse serviço quando oferecido sobre redes sem fio, em particular sobre redes *mesh*, devido ao recente aumento do uso desse tipo de rede.

A utilização otimizada da largura de banda de redes sem fio afeta positivamente na qualidade de chamadas VoIP realizadas sobre este tipo de rede. A compressão de cabeçalhos para VoIP é uma alternativa para diminuir a utilização da largura de banda de redes sem fio para transmissão de informações de controle, aumentando assim a porcentagem de banda utilizada para transporte de carga útil, ou seja, dos dados de voz. No entanto, este mecanismo pode fazer com que o sistema VoIP fique menos tolerante à perdas, o que pode ser prejudicial em redes *mesh* sem fio, dada a sua característica de alta taxa de perda de pacotes.

Este trabalho sugere uma abordagem alternativa de compressão de cabeçalhos, que se utiliza de agregação de pacotes, com a intenção de eliminar esse problema de intolerância à perda de pacotes sem diminuir o ganho de compressão. É mostrada uma avaliação dessa abordagem em comparação com alguns algoritmos de compressão utilizados atualmente, baseada em simulações feitas utilizando o NS-2.

Os resultados obtidos pelas simulações mostram que a abordagem sugerida pode diminuir consideravelmente a perda de pacotes, com o custo de afetar negativamente no atraso dos pacotes. No entanto, é observado que apesar deste atraso imposto, a abordagem proposta apresenta vantagens com relação aos outros algoritmos testados.

Palavras-chave: VoIP, compressão de cabeçalhos, agregação de pacotes, *mesh*.

Abstract

Voice over IP technology has been widely used by companies and home users to carry voice calls over the Internet by companies and home users. Due to its lower cost in comparison to the public telephony system, the trend is that the convergence of voice and data services to one media will be completed soon. Therefore, it is natural to study the quality of service of VoIP systems over wireless networks, especially on *mesh* networks, due to the recent increase on the use of such a network.

The optimized use of the bandwidth of wireless networks positively affects the quality of VoIP calls on this type of network. VoIP header compression is an alternative to reduce the use of the bandwidth needed to transmit control information, thereby increasing the percentage of bandwidth used to carry payload, or voice. However, this mechanism can make the VoIP system less tolerant to packet loss, which can be harmful in wireless *mesh* networks, due to its high rate of packet loss.

This work suggests an alternative approach for header compression, using packet aggregation with the intention of eliminating the problem of intolerance to packet loss without reducing the compression gain. It is shown an evaluation of the suggested approach compared with some of compression algorithms used today, based on simulations made using NS-2.

The results obtained by simulation show that the suggested approach can significantly reduce packet loss, with the cost of increasing the delay of packets. However, it is observed that despite this additional delay, the suggested approach shows some advantages in comparison to the other algorithms.

Keywords: VoIP, header compression, packet aggregation, mesh.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o recente aumento do uso de redes locais sem fio (*Wireless Local Area Networks* - WLAN) [9], a utilização de uma grande variedade de serviços nessas redes tem aumentado cada vez mais. Considerando que muitos dispositivos móveis recentemente oferecidos no mercado incluem interfaces de rede sem fio, é cada vez mais comum perceber a utilização desse tipo de rede em ambientes corporativos ou residenciais, como uma extensão de redes já existentes que oferecem acesso a redes externas e/ou à Internet.

As redes em malha sem fio, ou redes *mesh*, vêm como uma alternativa para ampliar o alcance de redes sem fio infra-estruturadas já existentes, ou para oferecer conectividade em regiões ou áreas em que seria inviável ou indesejável a implantação de redes locais cabeadas, ou mesmo redes sem fio infra-estruturadas.

As redes *mesh* têm tido um considerável aumento de popularidade, pois suas vantagens vão desde a inerente tolerância a falhas de rede até a grande simplicidade de implantação, devido à sua característica de auto-configuração, além da ubiquidade oferecida por esse tipo de rede. Considerando que um *backbone mesh* é composto por roteadores sem fio, é possível oferecer uma grande área de cobertura, dependendo apenas da quantidade de roteadores e de seu posicionamento.

Com o aumento do uso de redes *mesh* ao redor do mundo, e da quantidade de serviços oferecidos nesse tipo de rede, é natural o interesse em se utilizar serviços de tempo real, como a tecnologia VoIP (*Voice over Internet Protocol*). Esta, por sua vez,

tem sido cada vez mais utilizada para a realização de chamadas telefônicas, por grandes empresas ou usuários domésticos de Internet.

O baixo custo da tecnologia VoIP é um potencial atrativo para a sua implantação em redes de diversos tipos, apesar de seu comportamento particular e exigente em relação às características da rede. Seu uso oferece a possibilidade de convergir a um único meio de comunicação vários serviços, antes necessariamente atrelados a tecnologias distintas, como voz, vídeo e dados.

No entanto, aplicações de tempo real tem requerimentos elevados em relação a características da rede, o que faz com que seu uso em redes sem fio torne-se mais difícil. Além disso, devido ao comportamento dinâmico da rede e dos canais característico de redes *mesh*, existem desafios significativos para o desenvolvimento e otimização desses serviços. Atraso, uso da largura de banda, *jitter* e perda de pacotes são fatores que influenciam diretamente na qualidade do serviço oferecido por aplicações de tempo real.

A implementação de compressão de cabeçalhos é uma alternativa para otimizar o uso da largura de banda da rede. É sabido que, em serviços VoIP, o tamanho do *payload* do pacote é comumente menor que o tamanho do cabeçalho, fazendo com que a porcentagem de informações de controle trafegada no canal seja superior a 50% da banda utilizada. O uso de compressão de cabeçalhos pode reduzir o tamanho do cabeçalho do pacote em mais de 90%, aumentando a porcentagcomem de uso da largura de banda para transmissão de carga útil.

No entanto, o uso de compressão de cabeçalhos em redes sem fio introduz um novo desafio, considerando que a sua utilização faz com que o tráfego VoIP fique mais sensível à perda de pacotes, como foi verificado durante o desenvolvimento deste trabalho. O tipo de compressão de cabeçalho mais comumente adotado utiliza-se da chegada de pacotes consecutivos para atualização de contexto. Esse método exige que as duas pontas da comunicação estejam sincronizadas para evitar que pacotes sejam descomprimidos erroneamente, ou sejam descartados, e a função da atualização de contexto é manter essa sincronização. Esse problema pode ser crucial para a qualidade da comunicação em ambientes muito congestionados, pois a perda de um único pacote pode

causar a perda de sincronização entre as pontas, e acarretar no descarte de todos os pacotes seguintes no descompressor até que a sincronização seja restabelecida.

Um algoritmo de compressão de cabeçalhos que não apresente a necessidade de sincronização de contextos elimina toda possibilidade de descarte de pacotes no descompressor devido à perda de pacotes, bem como todo processo necessário de atualização de contexto e restabelecimento de sincronização. No entanto, o custo de se implantar um algoritmo desse tipo pode ser refletido no ganho de compressão, que pode ser menor com relação aos algoritmos que necessitam de sincronização.

Na tentativa de manter um alto ganho de compressão, se utilizando de algoritmos que não fazem sincronização de contextos, a proposta do trabalho dessa dissertação é a utilização de uma técnica simples de compressão de cabeçalhos juntamente com uma técnica de agregação de pacotes, um mecanismo criado originalmente para diminuir o trabalho realizado pela camada de enlace de redes sem fio. A utilização de agregação de pacotes em conjunto com a compressão de cabeçalhos, como dois métodos interdependentes, pode aumentar o ganho de compressão pois a agregação oferece uma oportunidade de eliminar informações redundantes contidas nos cabeçalhos dos pacotes de voz que fazem parte de um mesmo pacote de agregação. Resultados de experimentos de simulação mostraram que essa técnica oferece grandes ganhos na qualidade da comunicação.

Portanto, o objetivo geral desta dissertação é avaliar a solução de compressão de cabeçalhos e agregação de pacotes proposta, em comparação com outras abordagens existentes. Para isso, um estudo confiável sobre as técnicas de compressão de cabeçalhos deverá ser realizado por meio de experimentos de simulação, bem como um estudo das técnicas de avaliação de desempenho, ambientes e métricas mais adequados, afim de oferecer resultados confiáveis para a pesquisa realizada.

Esta dissertação está dividida em 6 capítulos, dos quais o primeiro é esta introdução.

O Capítulo 2 apresenta os fundamentos teóricos necessários para o desenvolvimento deste trabalho, como uma breve descrição teórica de redes *mesh*, fundamentos de VoIP e compressão de cabeçalhos.

O Capítulo 3 descreve o problema do uso de compressão de cabeçalhos em redes *mesh*, a motivação principal para o desenvolvimento deste trabalho, bem como a abordagem proposta como solução para este problema.

O Capítulo 4 mostra como os experimentos foram conduzidos, descreve os cenários utilizados e as métricas escolhidas para fazer a avaliação dos algoritmos de compressão de cabeçalhos testados.

O Capítulo 5 traz os resultados dos experimentos, bem como uma discussão e análise dos mesmos de acordo com o funcionamento de cada algoritmo.

O Capítulo 6, por fim, apresenta as conclusões finais do trabalho, bem como sugestões para trabalhos futuros.

1.1 Motivação

Compressão de cabeçalhos e agregação de pacotes são duas técnicas utilizadas para melhorar o uso da largura de banda em redes sem fio. Existem muitos trabalhos que exploram a compressão de cabeçalhos em redes sem fio, em variados contextos, e não apenas para aplicações VoIP, como é possível verificar em [10, 11, 12, 13, 14, 15, 16]. São trabalhos que avaliam a compressão, seus conceitos, ou sugerem novas técnicas, para ambientes sem fio de múltiplos saltos ou de um único salto sem fio.

O mesmo pode ser observado para a agregação de pacotes. Trabalhos como [17, 18, 3] sugerem novas técnicas de agregação de pacotes para redes *mesh*, com o objetivo de melhorar a qualidade de chamadas VoIP.

O trabalho apresentado em [19] avalia a utilização de ambos os mecanismos. Os autores avaliam os efeitos de comportamento do algoritmo RoHC e agregação de pacotes em redes *mesh*. Porém, como em muitos outros trabalhos, a avaliação é feita sobre o uso dos mecanismos em suas funcionalidades originais, ou seja, como procedimentos isolados. A utilização de compressão de cabeçalhos e agregação de pacotes em um mesmo sistema para aumentar a qualidade do tráfego não é novidade, considerando que são procedimentos já desenvolvidos e estudados há alguns anos. Porém, para o uso de compressão e agregação como processos interdependentes, como apresentado nesta

dissertação, não se encontram muitos trabalhos na literatura atualmente.

Em [20], os autores sugerem o uso de um algoritmo de compressão de cabeçalhos denominado *Zero-Length Header Compression* (ZHC), que não necessita de sincronização entre os contextos e também faz uso de agregação de pacotes. Porém, o foco do trabalho apresentado é a qualidade de chamadas VoIP em geral, não oferecendo muitos detalhes quanto ao algoritmo de compressão de cabeçalhos e não apresentando uma avaliação comparativa do método quanto a outros algoritmos. Além disso, os autores desse trabalho sugerem o uso do ZHC de uma forma diferente, sem o uso de um mecanismo prévio de compressão para eliminar as informações estáticas.

A abordagem desta dissertação sugere o uso de compressão de cabeçalhos em pacotes agregados, através da eliminação de informações dinâmicas redundantes. Esse mecanismo depende do processo de agregação, o que faz com que o funcionamento efetivo de uma técnica dependa do funcionamento da outra. Porém, o preço a pagar por um mecanismo de compressão e agregação cooperativo cujo objetivo é diminuir a perda de pacotes no canal, é o atraso introduzido pelo mecanismo de agregação de pacotes. Avaliar se o impacto deste atraso imposto será prejudicial à comunicação VoIP é de suma importância para a validação da abordagem proposta neste trabalho.

1.2 Objetivos

O principal objetivo desta dissertação é avaliar o desempenho da abordagem proposta do uso de compressão de cabeçalhos estática com o auxílio da agregação de pacotes.

São objetivos secundários deste trabalho:

- Realizar a avaliação do desempenho do algoritmo RoHC U-mode, pelo fato de ser um algoritmo padronizado pelo IETF, apresentar um alto ganho de compressão, e apresentar o problema da propagação da perda.

- Realizar a avaliação do desempenho do mecanismo de compressão estática, por este não apresentar um ganho tão alto, mas no entanto não possuir o problema de dessincronização de contextos.

- E por fim, realizar uma análise comparativa entre todos esses diferentes mecanismos de compressão, observando qual deles oferece melhor desempenho em redes *mesh*.

Para cumprir este objetivo, utilizar-se-á um ambiente de simulação configurado adequadamente com os algoritmos a serem avaliados, utilizando-se de métricas gerais para avaliação de sistemas VoIP e métricas específicas para avaliação de algoritmos de compressão de cabeçalhos, e de cenários que exploram as principais características das redes *mesh*.

Capítulo 2

Fundamentos Teóricos

Este capítulo apresenta os principais conceitos teóricos envolvidos neste trabalho. As seções 2.1, 2.2 e 2.3 trazem conceitos relacionados a redes *mesh*, voz sobre IP e compressão de cabeçalhos, bem como seu funcionamento e principais desafios. A seção 2.4 mostra os principais trabalhos relacionados com o tema de compressão de cabeçalhos para VoIP, bem como uma descrição breve dos principais algoritmos utilizados.

2.1 Redes *mesh*

Por definição, redes *mesh* são redes que possuem sua topologia total ou parcialmente em malha, formadas por vários nós sem fio estáticos, que são capazes de se comunicar entre si sem o auxílio de nenhuma estrutura extra, e oferecem conectividade a dispositivos sem fio através de uma infra-estrutura distribuída [21].

Redes *mesh* complementam os padrões de WLAN existentes, como IEEE 802.11a/b/g/n para oferecer comunicações seguras e confiáveis entre todos os dispositivos sem fio possíveis, tanto PDAs (*Personal Digital Assistants*), telefones celulares, *laptops*, como outros dispositivos portáteis que possuírem uma interface de comunicação sem fio. Esse tipo de rede não é dependente de um único ponto responsável pela rede. Seus protocolos de roteamento devem trabalhar de forma auto-configurável e auto-organizável, para prover a infra-estrutura necessária à comunicação de todos os dispositivos interconectados.

As redes sem fio infra-estruturadas comumente usadas atualmente são formadas

por pontos de acesso (AP - *access points*) que oferecem conectividade sem fio, porém dependem de uma estrutura cabeada para ligar os pontos de acesso à rede. A principal característica das redes *mesh* é a ausência de cabos entre seus nós, fazendo assim com que sua implantação seja mais barata e mais fácil. As redes *mesh* podem ser usadas como alternativas às tradicionais redes cabeadas, e às redes sem fio infra-estruturadas, ou como uma extensão das mesmas, quando existe a necessidade de ampliar a área de abrangência de uma determinada rede.

Além da simplicidade de implantação, é possível citar como vantagem das redes *mesh* a alta tolerância a falhas de rede inerente de sua arquitetura [21]. A topologia em malha oferece a possibilidade de a rede se auto configurar em caso de falhas de nó ou enlace. Para isso, o protocolo de roteamento utilizado deve possuir formas de detectar falhas na rede, e deve oferecer suporte ao recálculo de rotas quando falhas forem detectadas.

Segundo [21], o conceito chave para o funcionamento de uma rede *mesh* é o roteamento. Os roteadores devem ser capazes de prover rotas ótimas através da rede, para que os clientes possam utilizar as melhores rotas através dos múltiplos enlaces sem fio. Os protocolos de roteamento devem levar em consideração as difíceis condições impostas pelo uso do ar como meio de propagação e seu comportamento altamente dinâmico, dar suporte a uma comunicação confiável e eficiente, além de não consumir muito da largura de banda disponível na rede com mensagens de controle.

Apesar da grande quantidade de serviços que podem ser oferecidos a usuários de redes sem fio, a implantação de serviços de tempo real, como VoIP, nesse tipo de rede não é trivial. As redes sem fio possuem características específicas bem diferentes de redes que se utilizam de uma infra-estrutura de cabos, devido principalmente ao meio de propagação utilizado.

Uma das dificuldades encontradas pela utilização de redes sem fio está relacionada à alocação das freqüências de transmissão, que dependem da regulamentação dos órgãos competentes de cada país. Além disso, os usuários de uma mesma tecnologia de rede sem fio devem operar na mesma freqüência, o que limita o número de usuários, o

alcance, a velocidade e a largura de banda. O uso otimizado da banda disponível pode maximizar o número de usuários utilizando um determinado serviço, como VoIP, ao mesmo tempo [13]. Por isso, existe um grande esforço da parte de pesquisadores em otimizar o uso da largura de banda disponível em redes sem fio, principalmente para o tráfego de serviços que exigem muito desse recurso.

As redes *mesh* podem ser classificadas em três tipo, de acordo com a topologia da rede [21, 22, 23]:

- *Rede de topologia plana:* em uma topologia plana, a rede é formada por nós que agem tanto como clientes quanto como roteadores. Todos os dispositivos da rede são vistos como pontos pertencentes a um mesmo nível, independente das funções que desempenham, sejam clientes, roteadores ou *gateways*. Os clientes se organizam entre si para prover roteamento, configuração da rede, para prover serviços e outras aplicações. Apesar de oferecer simplicidade, esse tipo de arquitetura não oferece escalabilidade, além de impor uma alta restrição de recursos [23].

- *Rede de topologia hierárquica:* redes de topologia hierárquica são aquelas em que os dispositivos estão separados em vários níveis de hierarquia, de acordo com suas funções na rede, onde os clientes ou usuários finais estão no nível mais baixo. Esses clientes se comunicam com um *backbone* formado por roteadores *mesh*. Na maioria dos casos, os nós *mesh* são roteadores dedicados. Isso quer dizer que esses dispositivos nunca vão ser a fonte ou destinatário de uma comunicação, e simplesmente farão o encaminhamento das comunicações vindas dos clientes. A responsabilidade de se auto-organizar e auto-configurar, além de prover conectividade e serviços, é então dos nós dedicados que fazem parte do *backbone*. Alguns desses elementos ainda podem ser *gateways* de rede, oferecendo acesso à Internet [21].

- *Rede de topologia híbrida:* essa classificação é dada às redes *mesh* que apresentam as duas topologias mostradas anteriormente [22]. Os autores de [21] e [23] classificam como redes *mesh* híbridas também as redes hierárquicas que se utilizam

de outras redes para comunicação, como por exemplo redes de telefonia celular ou redes Wi-MAX. A idéia desse tipo de arquitetura é manter a escalabilidade da rede, e oferecer melhor qualidade nas comunicações.

Os elementos de uma rede *mesh* que fazem parte da formação e da operação da rede, repassando os dados e participando da descoberta de rotas são denominados *Mesh Points* (MP). Esses elementos podem também assumir o papel de pontos de acesso para nós clientes, sendo nesse caso denominados *Mesh Access Points* (MAP). Os MAPs também são capazes de se auto-configurar, assim como os MPs. Essa característica permite que no momento em que um novo dispositivo é inicializado na rede, ele automaticamente procura por outros dispositivos em sua área de cobertura, e associa-se a eles. O acesso a uma rede cabeada para prover acesso à Internet ou a outras redes é feito por dispositivos denominados *Mesh Portal Points* (MPP), que agem como *gateways* e são capazes de fazer a conexão entre a rede *mesh*, e uma rede externa [24].

2.1.1 Perda de pacotes e vazão de dados em redes *mesh*

Pelo fato de serem formadas basicamente por roteadores sem fio que se comunicam entre si, caracterizando um ambiente de múltiplos saltos sem fio, a interferência em redes *mesh* tende a ser maior devido ao número de diferentes rádios numa mesma área de cobertura [22]. Isso certamente leva a uma taxa de perda de pacotes maior, o que pode prejudicar fortemente uma comunicação de voz sem cuidados especiais.

Redes sem fio de múltiplos saltos enfrentam o problema do *terminal escondido*, o que faz com que uma transmissão sem fio seja inibida por outra feita a partir de outro nó. Nós que estão dentro da área de alcance do nó destinatário, mas não têm conhecimento um do outro são nós escondidos. Segundo [23], esse é o problema responsável pela maior parte das colisões que ocorrem em redes sem fio. A Figura 2.1 mostra um exemplo de cenário onde existem terminais escondidos. Na Figura 2.1, podemos ver que o nó A está enviando um pacote para o nó B. No entanto, haverá uma colisão, pois o nó C,

que está no alcance do nó B, está enviando um pacote para o nó D, o que vai resultar na não recepção do pacote de A para B.

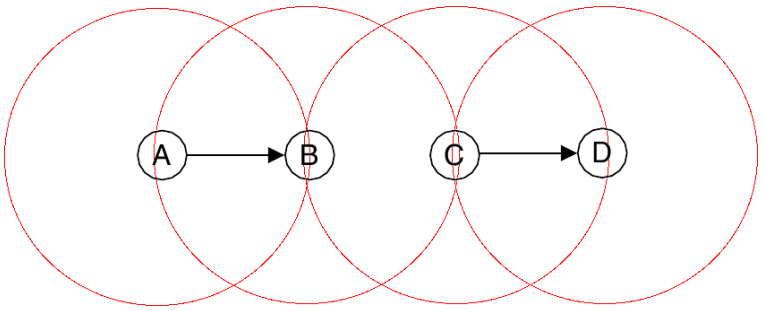

Figura 2.1: Problema do terminal escondido.

Outro problema que afeta a vazão de redes sem fio é o problema do *terminal exposto*, que faz com que um nó deixe de realizar uma transmissão por ouvir a transmissão de outro nó que está em sua área de alcance, no entanto sem saber se essa transmissão interferiria de fato na transmissão que seria feita. A Figura 2.2 mostra um exemplo de cenário onde ocorreu o problema do terminal exposto. Na Figura, o nó A deixou de transmitir seu pacote para B por estar ouvindo a transmissão que C está realizando para D. No entanto, a transmissão de C para D não interferiria na transmissão de A para B, pois os nós C e B estão fora de alcance um do outro.

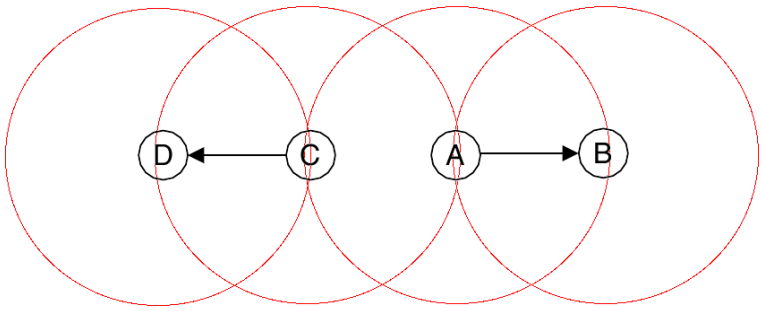

Figura 2.2: Problema do terminal exposto.

Ao se discutir a questão da capacidade de uma rede *mesh* é importante levar em consideração também o problema da eqüidade. Um nó em uma rede *mesh* deve transmitir o tráfego vindo de outros nós que precisa ser encaminhado, além do próprio tráfego. Portanto, além da competição com outros nós por transmissões realizadas para um mesmo nó destino, existe também uma competição inevitável entre o próprio tráfego,

e o que deve ser encaminhado. Esse tipo de competição não ocorre em redes sem fio infra-estruturadas, pois os nós estão sempre a um único salto de distância do ponto de acesso [1].

Considerando uma rede *mesh* com 3 nós, como mostrado na Figura 2.3, onde aos nós 1 e 2 é aplicada igualmente uma carga G, que será encaminhada ao nó 3, considerado o *gateway* da rede. Segundo [1], os tráfegos S_1 e S_2, originados nos nós 1 e 2 respectivamente, não vão ser encaminhados de forma justa. O comportamento esperado é que o nó 2 priorize o próprio tráfego S_2 diminuindo a vazão do tráfego S_1 originado no nó 1. Portanto, quanto mais longe do *gateway* estiver um nó, menor será a vazão do seu tráfego.

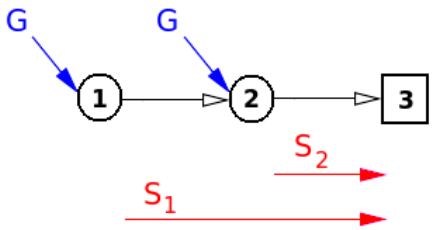

Figura 2.3: Rede de múltiplos saltos sem fio com carga originadas nos nós 1 e 2 ao *gateway* (nó 3) da rede. Fonte: [1].

Além disso, a vazão diminui rapidamente em uma rede *mesh*, conforme o número de saltos aumenta [21]. Vários fatores podem contribuir para a degradação da vazão, como características do protocolo MAC utilizado, o problema do terminal exposto, o problema do terminal escondido, e principalmente a auto-interferência.

A diminuição da vazão também se dá pela alta taxa de erros obtida em redes sem fio. Em comparação com transmissões realizadas em redes cabeadas, as transmissões de rádio são suscetíveis a fatores negativos, como o decaimento da potência do sinal propagado ao longo do caminho, além de outros. Por isso, medidas específicas devem ser tomadas para que os serviços oferecidos não sejam afetados pela perda de pacotes, que inevitavelmente ocorre nesse tipo de rede. Melhorar a utilização da rede através de medidas de otimização do uso da largura de banda pode ajudar nesse aspecto, já que quanto menos tempo uma informação necessitar para ser transmitida, menor a

probabilidade de ela sofrer algum dano durante a transmissão e chegar corrompida ao seu destinatário [13].

2.2 Voz sobre IP

Voz sobre IP (VoIP - *Voice over Internet Protocol*) é a tecnologia que permite o transporte de voz sobre redes de comutação de pacotes. É utilizada pela telefonia IP, e permite que a voz seja capturada, digitalizada, codificada, empacotada e enviada pela rede até um destinatário localizado em outro ponto.

A telefonia IP, por apresentar características de interatividade em tempo-real, é mais complexa do que aplicações multimídia de tempo contínuo, como por exemplo transmissões de rádio ou TV na Internet. Nessas últimas, a restrição com relação a tempo não é tão severa; para aplicações de telefonia IP, o atraso é um fator importante pois pode comprometer a interatividade entre os interlocutores, afetando a qualidade da comunicação.

Segundo [25], as aplicações VoIP possuem exigências específicas com relação às redes em que são implantadas:

- Atraso fim-a-fim: também chamado de atraso M2E (*mouth-to-ear*), é o atraso total da voz no sistema. Inclui todos os atrasos introduzidos pelo sistema VoIP, como atrasos de codificação e empacotamento, além dos atrasos introduzidos pela rede, como atraso de processamento, atraso de fila, atraso de transmissão e atraso de propagação [26].

- Variação de atraso ou *jitter*: a variação de atraso define a diferença no tempo de chegada dos pacotes no destinatário. A variação do atraso é importante pois afeta a cadência da reprodução da voz no destinatário. É causada pelos diferentes atrasos que cada pacote sofre na rede.

- Taxa de erro (FER - *Frame Error Rate*): define a taxa de quadros incompletos ou recebidos com erros. Para aplicações VoIP, a taxa de erro tolerada pode ser

relativamente alta, se comparada a outros tipos de aplicações. Porém, essa taxa não pode afetar a compreensão dos interlocutores.

A Figura 2.4 mostra os componentes básicos de uma comunicação VoIP. A voz é capturada e então periodicamente codificada. A cada período de fala capturado, é associado um *stream* de bits a uma dada taxa de produção, determinada pelo algoritmo utilizado pelo *codec* (CODificador/DECodificador). Após o processo de codificação, as amostras de voz digitalizadas são agrupadas em pacotes e a estes são adicionados cabeçalhos de protocolos da pilha TCP/IP, dos quais as aplicações VoIP fazem uso para seu funcionamento. São esses os cabeçalhos dos protocolos RTP, UDP, IP, e o protocolo de camada de enlace referente ao meio de transmissão utilizado.

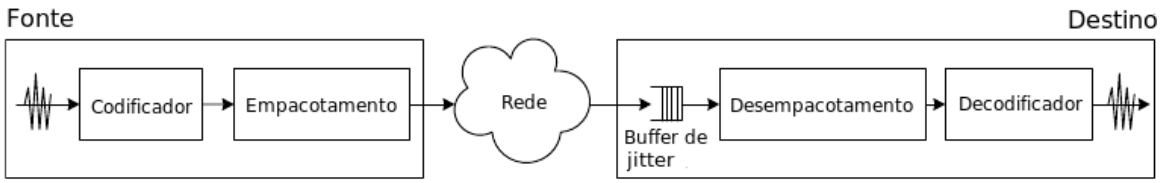

Figura 2.4: Componentes básicos de uma comunicação VoIP. Fonte: [2].

Após serem transmitidos pela rede de dados e atingirem o destinatário, os pacotes de voz são armazenados em um componente denominado *buffer* de *jitter*, que possui a função de eliminar a variação de atraso (*jitter*), descartar os pacotes de voz que chegam fora do tempo, e amenizar os atrasos da rede. No tempo certo os pacotes então são desempacotados, decodificados e reproduzidos.

Segundo [27], um atraso fim-a-fim abaixo de 150 ms é considerado aceitável para aplicações de voz interativas. Com atrasos na faixa de 150 ms até 400 ms o usuário já percebe problemas de interatividade na comunicação. Acima de 400 ms, o atraso é inaceitável e a comunicação é considerada inviável.

Outro fator importante para o uso de aplicações VoIP é a largura de banda disponível na rede. Os autores de [2] afirmam que este fator é de suma importância para sistemas VoIP, pois influencia na quantidade de chamadas que uma rede pode sustentar.

A taxa de codificação utilizada pelos *codecs* influencia diretamente no uso da banda. Alguns *codecs* utilizam técnicas de compactação com o objetivo de diminuir a quantidade de bits enviados a rede, ou seja, utilizam taxas de codificação mais baixas. A tabela 2.1 mostra alguns dos *codecs* mais usados padronizados pela *International Telecommunication Union - Telecommunication Standardization Sector* (ITU-T), a taxa usada para a codificação das amostras de voz, o tamanho em milisegundos dos pacotes, o tempo de conversão analógico digital (*look-ahead time*) e a largura de banda combinada, ou seja, para uma chamada bidirecional considerando links simétricos.

Tabela 2.1: Codecs mais usados definidos pela ITU-T. Fonte: [2].

Codec	**Taxa de bits (kbps)**	**Tamanho do pacote (ms)**	**Tempo de conversão A/D (ms)**	**Largura de banda combinada (kbps)**
G.711u	64.0	20	1.0	180.80
G.711a	64.0	20	1.0	180.80
G.729	8.0	20	25.0	68.80
G.723.1 (MPMLQ)	6.3	30	67.5	47.80
G.723.1 (ACELP)	5.3	30	67.5	45.80

Quanto maior a taxa de codificação utilizada pelo *codec*, maior será a largura de banda utilizada. Os codificadores de voz impõem além do atraso de codificação, perda de informações e de qualidade para realizar a compressão [6]. Portanto, existe uma situação de conflito entre a quantidade de bits enviados à rede e a qualidade da voz, pois quanto maior for a taxa de codificação, melhor será a qualidade, porém maior será o tamanho do pacote gerado, e maior será o uso da banda.

Segundo [6], dentre os mecanismos utilizados para prover uma utilização otimizada da banda disponível podem ser citados mecanismos de supressão de silêncio utilizados pelos *codecs* de voz, o agrupamento de vários quadros de amostras de voz em um único pacote, e a compressão de cabeçalhos, objeto de estudo deste trabalho.

2.3 Compressão de cabeçalhos

Aplicações VoIP usam cabeçalhos IP/UDP/RTP para realizar o transporte correto do fluxo de mídia entre as entidades participantes da comunicação. A voz é capturada,

digitalizada, codificada e comprimida, sendo dividida em quadros que são encapsulados em pacotes que serão enviados à rede [6].

A compressão de cabeçalhos é uma técnica empregada para aumentar a porcentagem de largura de banda utilizada para o transporte de dados. Considerando que os pacotes de voz geralmente apresentam carga útil de tamanho menor que os cabeçalhos [16], a porcentagem de informação de dados nos pacotes é menor que a porcentagem de informações de controle. A idéia principal das técnicas de compressão de cabeçalhos é diminuir a quantidade de informações de controle nos cabeçalhos dos pacotes e conseqüentemente sua porcentagem em relação ao tamanho do pacote, aumentando a largura de banda disponível para transporte de informações de dados (carga útil do pacote).

Um sistema de compressão de cabeçalhos é composto por duas entidades principais denominadas *compressor* e *descompressor*, que estão geralmente localizadas na extremidade fonte e na extremidade destino da comunicação, respectivamente. Estão localizados entre as camadas de enlace (camada 2) e de redes (camada 3) da pilha de protocolos TCP/IP dos dispositivos (Figura 2.5). O compressor recebe os pacotes da camada de redes, comprime os cabeçalhos, e envia os pacotes com cabeçalhos comprimidos para a camada de enlace. O descompressor faz o processo inverso, recebendo os pacotes com cabeçalhos comprimidos, descomprimindo-os e repassando-os para a camada de redes. Portanto, a compressão é aplicada aos cabeçalhos das camadas superiores à camada de enlace.

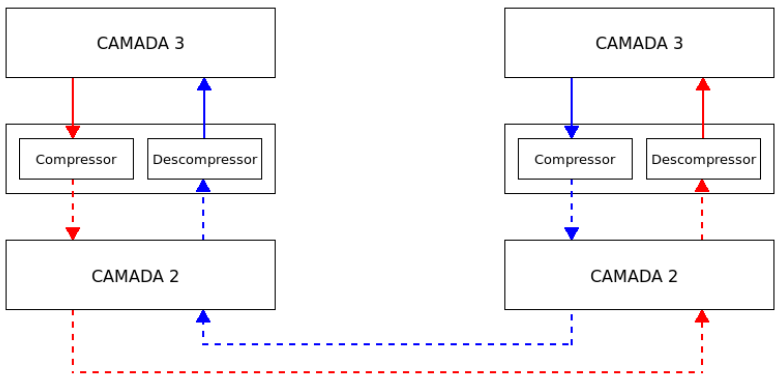

Figura 2.5: Posição do compressor e descompressor na pilha de protocolos TCP/IP.

A compressão de cabeçalhos se aproveita da redundância que existe entre os valores dos campos de cabeçalhos de pacotes consecutivos de um mesmo fluxo de dados. Considerando um fluxo de voz, as informações redundantes ocupam a maior parte dos cabeçalhos dos pacotes, e seus valores geralmente são iguais ou variam de uma forma pré-estabelecida. A compressão elimina dos cabeçalhos essa informação redundante e a armazena em estruturas de dados chamadas *contextos*, que ficam localizados nas extremidades da comunicação, ou seja, no compressor e no descompressor.

Os contextos de sessão armazenam informações que se referem ao estado atual dos fluxos multimídia. São estruturas que armazenam as informações que são retiradas dos cabeçalhos dos pacotes no momento da compressão, e que serão utilizadas para remontá-los no momento da descompressão. Cada registro dentro do contexto guarda informações sobre um fluxo de voz específico. Apesar de as informações redundantes não serem enviadas para a rede numa comunicação com compressão de cabeçalhos, elas devem ser enviadas ao destinatário pelo menos uma vez no início da comunicação, para que o descompressor possa armazená-las e utilizá-las para remontar os cabeçalhos originais a partir dos cabeçalhos comprimidos.

2.3.1 Estrutura e classificação dos cabeçalhos

Com relação aos valores assumidos durante uma sessão de voz, os campos de cabeçalhos IP/UDP/RTP apresentam diferentes comportamentos, que formam padrões conhecidos e garantem a base para o processo de compressão. De acordo com este padrão de comportamento, os campos de cabeçalhos podem ser classificados em [28]:

- *inferíveis*: são aqueles cujos valores podem ser calculados a partir de outros campos de cabeçalho. São classificados como inferíveis os campos de tamanho do pacote dos cabeçalhos IP e UDP, e de *checksum* do cabeçalho IP.

- *estáticos*: são aqueles que nunca mudam seu valor durante uma sessão. São considerados campos estáticos os campos de versão, endereço IP de fonte e destino, tamanho do cabeçalho, os campos de *flag* e o campo de protocolo do cabeçalho

IP, os campos de porta de fonte e destino do cabeçalho UDP, e os campos de versão, *padding*, fonte de sincronização (SSRC) e extensão do cabeçalho RTP.

- *dinâmicos*: são aqueles que mudam, ou podem mudar, durante a sessão. Podem assumir valores aleatórios, ou valores que mudam de alguma forma pré-estabelecida. São campos dinâmicos os campos de tipo de serviço, identificação e *time to live* do cabeçalho IP, o campo *checksum* do UDP, e os campos CSRC *counter*, *marker*, tipo de carga útil, número de sequência, *timestamp* e CSRC (se estiver presente) do cabeçalho RTP.

A Figura 2.6 mostra um esquema dos campos de cabeçalhos IP/UDP/RTP e sua classificação. A classificação dos campos de cabeçalhos em uma dessas categorias determina que tratamento irão receber no momento da compressão. Geralmente, o processo inicial de um sistema de compressão é a fase de estabelecimento de contextos, onde as informações dos cabeçalhos devem ser enviadas para o descompressor. Medidas particulares de cada algoritmo incluem a forma como esses valores serão enviados, sendo que a mais comum é enviar um cabeçalho descomprimido no início da sessão. Outras medidas particulares incluem métodos específicos para codificar os valores que deverão ser enviados nos cabeçalhos comprimidos, e como evitar falhas no processo de descompressão no caso de perda de pacotes na rede.

2.3.1.1 Campos estáticos

Os campos estáticos ocupam a maior parte dos cabeçalhos usados nos pacotes de voz. Apesar de conservarem seus valores durante toda uma sessão de voz, é possível definir diferentes tipos de campos estáticos. De acordo com os valores que assumem, [28] identificou duas sub-categorias de campos estáticos. São elas:

- *estáticos de definição*: são aqueles campos cujos valores identificam o fluxo de voz ao qual pertencem. São campos estáticos de definição os campos de endereço IP de fonte e destino, os campos de porta de fonte e destino, e o campo de fonte de sincronização (SSRC) do cabeçalho RTP.

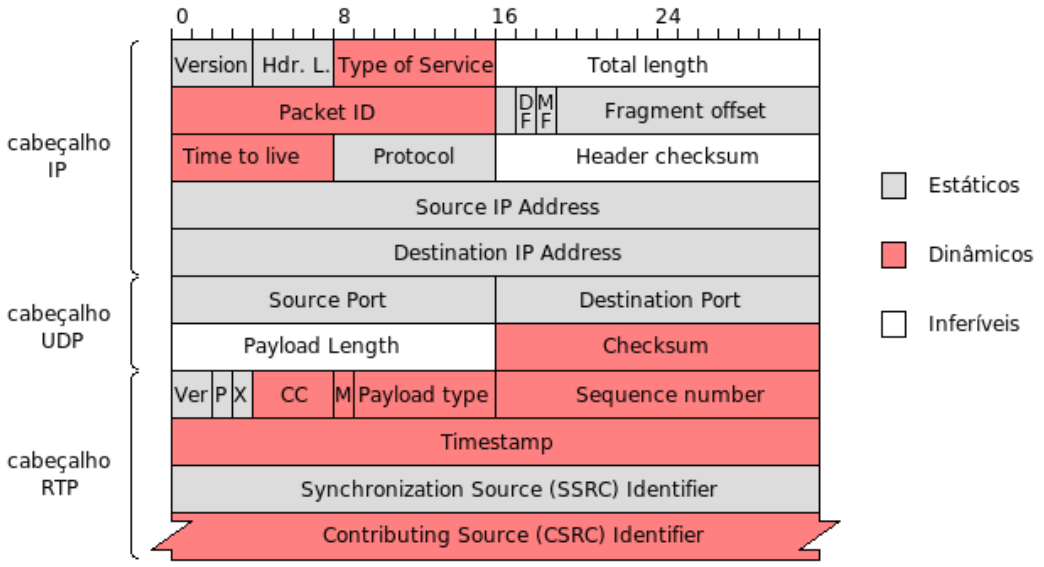

Figura 2.6: Classificação dos campos dos cabeçalhos IP/UDP/RTP.

- *estáticos conhecidos*: são os campos cujos valores são conhecidos *a priori* pelo sistema de compressão, e assumem o mesmo valor até para fluxos diferentes. Por esse motivo, os algoritmos de compressão geralmente não os enviam para o descompressor em momento algum da comunicação. São considerados campos estáticos conhecidos os campos de tamanho do cabeçalho IP, considerando que o cabeçalho IP não apresenta extensões em aplicações VoIP, o campo de *flag* reservado que sempre vai assumir o valor 0, os campos de *flag* de fragmentação do cabeçalho IP, já que para comunicações VoIP não se espera que haja fragmentação de pacotes devido ao tamanho usualmente pequeno dos pacotes, e também o campo de versão do RTP, já que existe apenas uma versão do RTP em uso atualmente, a versão 2.

Os campos estáticos são enviados para o descompressor apenas uma vez, no início da sessão, com excessão daqueles cujos valores são conhecidos, pois considera-se que o sistema de compressão já conhece esses valores *a priori*. Os campos estáticos que definem um fluxo são usados no compressor para identificar a qual contexto pertence um pacote recebido da camada de redes.

2.3.1.2 Campos dinâmicos

Para alguns campos dinâmicos, seu padrão de variação pode ser quantificado. São esses os campos cujos valores assumidos variam de uma forma previsível e constante, geralmente crescente, durante toda a sessão. Nesse caso, leva-se em consideração apenas a variação do valor (o delta), e não o próprio valor que o campo assume. Por isso, [13] assume também uma classe de campos denominada *delta*, formada pelos campos de cabeçalho que apresentam esse comportamento.

Devido aos diferentes comportamentos que os campos dinâmicos podem apresentar, segundo [28], eles podem ainda ser divididos em 5 sub-classes:

- *dinâmicos estáticos* ou *delta*: são campos que foram classificados como dinâmicos, porém seu padrão de mudança é simples, como por exemplo incremental. São os campos de variação previsível, e são aqui denominados estáticos porque seu valor delta é geralmente estático. São considerados campos delta o campo de identificação do cabeçalho IP quando seu valor é atribuído seqüencialmente, o campo de *checksum* do UDP quando não está presente, pois nesse caso seu valor é considerado 0, o campo de CSRC *counter*, quando não houver nenhuma fonte extra de mídia pois seu valor será sempre 0, e o campo de número de seqüência do RTP, pois é sempre incrementado de 1.

- *dinâmicos semi-estáticos*: são os campos dinâmicos que podem mudar ou não. Eles permanecem estáticos na maior parte do tempo da sessão. Um campo semi-estático ocasionalmente muda seu valor, porém retorna ao valor original depois de um certo número de pacotes enviados. É considerado campo semi-estático o campo *marker* do cabeçalho RTP, pois assume o valor 0 na maior parte do tempo, assumindo o valor 1 somente nos pacotes que marcam o início de um período de fala.

- *dinâmicos de mudança rara*: esses são os campos dinâmicos que assim como os semi-estáticos podem mudar ou não, porém quando mudam permanecem com o novo valor até o fim da sessão. São considerados campos dinâmicos de mudança

rara o campo de identificação do cabeçalho IP, quando seu valor é incrementado em mais de uma unidade, o campo tipo de serviço do cabeçalho IP, o campo de CSRC *counter*, quando houver fontes extras de mídia, os campos de tipo de carga útil, *timestamp* e CSRC (quando estiver presente) do cabeçalho RTP.

- *dinâmicos alternantes*: são os campos dinâmicos que alternam seus valores entre um pequeno número de valores diferentes. É classificado como campo dinâmico alternante o campo *time to live* do cabeçalho IP.

- *dinâmicos irregulares*: são aqueles em que não é possível identificar nenhum padrão de comportamento. São campos dinâmicos irregulares os campos de identificação do cabeçalho IP, quando seu valor é atribuído aleatoriamente e o campo de *checksum* do UDP, quando estiver presente.

Os campos classificados como dinâmicos semi-estáticos, de mudança rara e alternantes são enviados para o descompressor apenas no início da sessão, porém o sistema deve oferecer uma maneira de atualizar seus valores caso ocorra uma mudança. Com relação aos valores dinâmicos delta, por possuírem o valor de variação conhecido, são enviados uma única vez. Porém, os algoritmos oferecem alguma garantia de que o sistema é tolerante a perdas, pois a perda de um único pacote pode fazer com que os pacotes subseqüentes sejam remontados com os valores de referência errados. Os campos dinâmicos irregulares são enviados com seu valor original, em todos os pacotes do fluxo.

2.3.2 Problema da propagação da perda

Os contextos de compressor e descompressor guardam as informações de referência que serão utilizadas para remontar os cabeçalhos. Portanto, suas informações devem ser exatamente as mesmas, ou seja, os contextos devem estar *sincronizados*. Cada pacote enviado pela fonte é usado para atualizar o contexto do compressor, e analogamente cada pacote recebido no destinatário e corretamente descomprimido é utilizado para atualizar o contexto de descompressor.

Se não for possível manter a sincronização, o descompressor pode não conseguir descomprimir os cabeçalhos dos pacotes recebidos. Considerando que geralmente o descompressor se baseia em pacotes do mesmo fluxo anteriormente recebidos para atualizar seu contexto, a perda de um único pacote pode acarretar na dessincronização de contexto, e fazer com que o descompressor não consiga descomprimir os pacotes seguintes com sucesso, mesmo que cheguem a tempo e sem erros no descompressor, sendo obrigado a descartá-los. Nesse caso dizemos que a perda foi propagada, pois a perda de um único pacote fez com que o descompressor descartasse todos os pacotes seguintes (Figura 2.7).

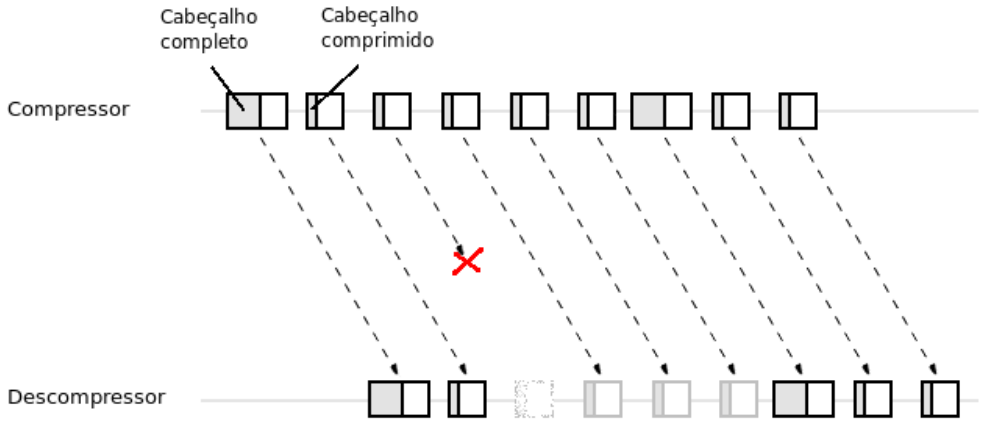

Figura 2.7: Problema da propagação da perda: quando um pacote é perdido, os pacotes seguintes são descartados até que o contexto seja sincronizado novamente.

Para amenizar o problema da propagação da perda, alguns algoritmos fazem uso de mensagens de atualização de contexto. São mensagens enviadas periodicamente, contendo todas as informações completas dos cabeçalhos. Quando o descompressor recebe uma mensagem de atualização, ele substitui todo o conteúdo atual do seu contexto pelo conteúdo da mensagem de atualização. Se ele estiver dessincronizado, ele usará a informação recebida para atualizar seus valores de referência e assim restabelecer a sincronização.

2.4 Trabalhos relacionados

O primeiro trabalho de compressão de cabeçalhos para VoIP publicado foi a *Request for Comments* (RFC) 2508 [29], que define o algoritmo *Compressed RTP* (CRTP). Foi desenvolvido inicialmente para enlaces seriais de baixa velocidade, onde o tráfego de voz e vídeo em tempo-real era potencialmente problemático. O algoritmo sugere a aplicação de compressão aos cabeçalhos IP/UDP/RTP, e sua proposta é a redução do tamanho dos cabeçalhos a aproximadamente 2 bytes, quando o campo *checksum* do UDP não estiver presente, e a 4 bytes caso contrário.

O CRTP foi desenvolvido baseado no único algoritmo de compressão de cabeçalhos existente até então, o *Compressed TCP* (CTCP) definido na RFC 1144 [30], que definia um algoritmo de compressão para os cabeçalhos IP e TCP em enlaces de baixa velocidade. O CRTP tem como principal característica a simplicidade de seu mecanismo.

O funcionamento do CRTP define o envio de uma mensagem inicial com todas as informações dos cabeçalhos (FULL_HEADER), utilizada para estabelecer o contexto no compressor e no descompressor. Em seguida os cabeçalhos são enviados comprimidos, carregando apenas as informações delta dos campos de cabeçalhos dinâmicos. Pacotes FULL_HEADER também são periodicamente enviados para o descompressor, com o objetivo de manter a sincronização entre os contextos, ou quando solicitado pelo descompressor através de um canal de *feedback* (retorno), no caso de o descompressor detectar que houve dessincronização de contexto.

O CRTP não apresentou boa performance em redes sem fio, por ter sido originalmente desenvolvido para enlaces confiáveis [31], e pela característica intrínseca das redes sem fio de apresentarem altas taxas de perdas de pacotes. Isso se dá pelo fato de o CRTP não oferecer nenhum mecanismo de recuperação a perdas de contexto, apresentando o problema da propagação da perda. Também influencia no mal desempenho do CRTP em redes sem fio o fato de as mesmas não necessariamente oferecerem um canal de *feedback* disponível para solicitação de recuperação de contexto.

Nas RFCs 3095 [28] e 4815 [32] é definido o algoritmo *Robust Header Compression*

(RoHC). Este algoritmo foi desenvolvido pelo *Internet Engineering Task Force* (IETF), para redes onde o CRTP não apresentou um bom desempenho, como as redes de celular sem fio. Sua proposta principal é a de um algoritmo que apresenta maior robustez que o CRTP nesse tipo de rede, sem que para isso seja necessário aumentar o tamanho dos cabeçalhos comprimidos.

O RoHC oferece três modos de operação: modo unidirecional (U-mode), modo bidirecional otimista (O-mode) e modo bidirecional confiável (R-mode). Os dois últimos fazem uso de um canal de *feedback*, assim como o CRTP, porém o modo U-mode define comunicação apenas no sentido do compressor para o descompressor. Isso introduz a possibilidade de se utilizar o algoritmo em enlaces onde o canal de *feedback* não existe, ou não é desejável utilizá-lo.

O modo U-mode trabalha com atualizações periódicas do contexto através de mensagens com os cabeçalhos completos enviadas para o descompressor. Os modos O-mode e R-mode trabalham com solicitação de atualização do contexto por parte do descompressor, quando for detectada a dessincronização. Segundo o trabalho apresentado em [12], o modo U-mode é o mais vantajoso para enlaces sem fio assimétricos, pois a atualização do contexto não depende da solicitação a partir do descompressor, através de um canal que pode não estar disponível (pelo fato de se tratar de enlaces assimétricos).

O algoritmo RoHC usa um método de codificação para os valores dos campos dinâmicos que são enviados nos cabeçalhos comprimidos, denominado *Window-Least Significant Bits* (W-LSB). Esse método de codificação é utilizado para os campos que sofrem mudanças pequenas, codificando e enviando apenas os bits menos significativos, que o descompressor utilizará para calcular o valor original do campo, junto com os últimos valores do campo descomprimidos com sucesso (a quantidade de valores utilizados como referência é a largura da janela utilizada pelo mecanismo). Este mecanismo, por fazer uso de uma janela de valores de referência, oferece uma certa tolerância com relação à perda de pacotes; porém se a perda de pacotes for em rajada e ultrapassar a largura da janela, a dessincronização de contexto será inevitável.

Para verificar se houve dessincronização de contexto, o RoHC implementa uma veri-

ficação sobre os campos dos cabeçalhos calculados pelo descompressor, chamada *Cyclic Redundancy Check* (CRC). Cada cabeçalho comprimido possui um campo que leva um valor de CRC calculado sobre os valores dos campos de cabeçalhos antes da compressão. Após receber o pacote, o descompressor recupera os valores dos campos com as informações do contexto, e executa novamente o cálculo do CRC. Se o valor for igual, a compressão foi considerada de sucesso, caso contrário é detectada a dessincronização de contexto.

O RoHC oferece um alto grau de compressão, e uma alta robustez, porém sua complexidade de implementação é bem mais alta com relação ao CRTP. Além disso, por ter sido implementado para redes de celulares, que apresentam normalmente um único enlace sem fio, ele considera que a rede entrega os pacotes em ordem, como o CRTP, o que pode não acontecer considerando redes IP de múltiplos saltos.

O *Enhanced Compressed RTP* (ECRTP), apresentado na RFC 3545 [31], foi desenvolvido então para redes com alto atraso, perda de pacotes e entrega de pacotes fora de ordem, na tentativa de cobrir as falhas do CRTP, e com o objetivo de oferecer uma opção também robusta porém mais simples com relação ao RoHC [31]. Apresenta algumas mudanças e extensões baseadas no CRTP, visando sua adaptação a redes com as características citadas.

As propostas do ECRTP são atingidas através de mecanismos de atualização de contexto mais eficientes que aqueles apresentados pelo CRTP, trazendo a possibilidade de manter a atualização dos contextos mesmo no caso de perda de alguns pacotes, e pela sugestão de um *checksum* para os cabeçalhos, como o cálculo CRC do RoHC, para que o descompressor possa detectar se realmente o contexto foi dessincronizado após a perda de um pacote. No entanto, apesar de todas essas melhorias, o ECRTP não atingiu a eficiência do RoHC, com relação a ganho de compressão, ficando como uma alternativa segura para redes em que um algoritmo de compressão mais simples é preferível em detrimento do grau de compressão atingido.

Apesar de os algoritmos de compressão citados implementarem mecanismos contra o problema da propagação da perda, ou seja, mecanismos para manter a sincronização de

contexto ou minimizar a quantidade de pacotes descartados devido à dessincronização, ainda assim para redes com altas taxas de perda de pacotes esses mecanismos podem não ser suficientes. Pensando nisso, em [10] foi definido um novo método para tentar aliviar o problema de dessincronização de contexto, denominado *Cooperative Header Compression* (CoHC).

O CoHC não é exatamente um algoritmo de compressão de cabeçalhos, mas um mecanismo que oferece uma maior tolerância a falhas causadas por perda de pacotes, que pode ser utilizado com qualquer algoritmo de compressão. Foi desenvolvido especificamente para dar suporte a algoritmos de compressão em redes *mesh*. Seu funcionamento define a utilização de uma estrutura extra que deve ser enviada em cada pacote comprimido. Essa estrutura extra, denominada *additional information container* (AIC), deve conter informações dos cabeçalhos de pacotes enviados anteriormente, que podem auxiliar o descompressor no caso de perda de pacotes.

Levando em consideração o fato de que em redes *mesh* múltiplas rotas podem ser formadas entre dois nós, o algoritmo foi desenvolvido para ser utilizado em múltiplos canais paralelos. Fazendo com que cada fluxo de voz diferente, entre o mesmo par fonte-destino, percorra um caminho diferente na rede, a idéia do mecanismo é que os pacotes de cada fluxo levem informações sobre os contextos dos outros fluxos. Se ocorrer perda de pacotes no caminho tomado por um dos fluxos, os outros poderão chegar até o destinatário com AICs carregando informações sobre os pacotes perdidos. No caso de haver apenas um fluxo de voz, o AIC deve conter informações sobre o contexto do próprio fluxo de voz, utilizado para comprimir o cabeçalho do pacote anterior.

No trabalho apresentado em [13], Abinader realizou experimentos com o algoritmo CRTP e com o mecanismo CoHC, e concluiu-se que o mecanismo pode realmente influenciar positivamente no tráfego VoIP. Porém, para que o mecanismo possa efetivamente usar múltiplas rotas paralelas em uma rede *mesh*, ele depende de um mecanismo de roteamento que lhe dê suporte.

Esse mecanismo, adicionado aos mecanismos próprios de cada algoritmo, pode aumentar a tolerância a falhas devido à perda de pacotes na rede, mas não eliminar

o problema da propagação da perda. No caso de redes muito congestionadas, onde perdas em rajada podem vir a ocorrer, esse mecanismo adicional pode ainda não ser suficiente para manter a qualidade de uma chamada de voz. Além disso, um cuidado especial deve ser tomado ao decidir que informações devem ser enviadas no AIC, pois isso implica em um aumento no tamanho dos cabeçalhos comprimidos, o que vai afetar o grau de compressão oferecido pelo algoritmo usado.

Capítulo 3

Descrição do Problema

Neste capítulo é descrito o principal problema que levou ao desenvolvimento deste trabalho, os principais objetivos do trabalho, e uma proposta de solução para o problema apresentado. A seção 3.1 mostra alguns dos problemas encontrados na aplicação de compressão de cabeçalhos em redes *mesh*, assim como o impacto do problema da propagação da perda para a qualidade de uma chamada de voz. A seção 3.2 descreve a solução apresentada e avaliada neste trabalho, proposta com o objetivo de enfrentar este problema utilizando duas técnicas conhecidas hoje em dia: compressão de cabeçalhos e agregação de pacotes. A seção 3.3 descreve, contextualiza e delimita os objetivos deste trabalho.

3.1 Compressão de cabeçalhos em redes *mesh*

Com o recente aumento do uso de redes *mesh*, a quantidade de diferentes serviços oferecidos nessas redes tem aumentado cada vez mais. Aplicações de tempo real, como VoIP (Voz sobre IP), são aplicações que estão sendo cada vez mais utilizadas nesses ambientes, apesar de seu comportamento exigente com relação às características da rede. Redes *mesh* têm sido implantadas em residências e ambientes empresariais como uma solução para extender redes sem fio infra-estruturadas, com a intenção de prover ubiqüidade no acesso à rede. Considerando que muitos dispositivos de comunicação recentemente oferecidos no mercado incluem tecnologias de interconexão sem fio, oferecer

VoIP aos usuários desses dispositivos vem ganhando grande importância.

Uma característica particular do tráfego VoIP é que pacotes de voz geralmente apresentam uma grande sobrecarga de informações de controle. A quantidade de carga útil do pacote geralmente é menor que o tamanho dos cabeçalhos anexados a ela, o que faz com que a quantidade de informações de controle trafegadas na rede seja maior que a quantidade de dados de voz. A tabela 3.1 mostra os tamanhos dos pacotes gerados pelos *codecs* mais utilizados. A tabela mostra que a maioria dos *codecs* gera pacotes cujo tamanho é menor que o tamanho dos cabeçalhos IP/UDP/RTP (40 bytes).

Tabela 3.1: Tamanhos dos pacotes gerados pelos *codecs* mais utilizados. Fontes: [6] e [7].

Codec	**Taxa de bits (kbps)**	**Tamanho do pacote (ms)**	**Tamanho do payload (bytes)**
G.711	64,0	20	160,0
G.726	32,0	20	80,0
G.728	16,0	20	40,0
G.729a	8,0	20	20,0
G.723.1 (MPMLQ)	6,3	30	23,625
G.723.1 (ACELP)	5,3	30	19,875
GSM-FR	13,0	20	32,5
GSM-HR	5,6	20	14,0
GSM-EFR	12,2	20	30,5

Sabendo das exigências e peculiaridades do tráfego VoIP, é interessante oferecer a esse tipo de aplicação o máximo de recursos de rede possível, na tentativa de garantir qualidade de serviço. Porém, dadas as características intrínsecas das redes *mesh*, como a alta taxa de perda de pacotes devido aos erros impostos pelo meio de transmissão e à sua estrutura de múltiplos saltos sem fio, torna-se um desafio garantir a qualidade de aplicações VoIP sobre esse tipo de rede.

Uma maneira de abordar o problema da qualidade de aplicações VoIP sobre redes sem fio é através da otimização do uso da largura de banda. Esse é um recurso limitado em redes sem fio, pois o meio de transmissão compartilhado é o ar, e os nós precisam usar a mesma freqüência para se comunicar. O uso otimizado da largura de banda do canal traz uma grande vantagem para aplicações VoIP: como a largura de banda é "economizada" para o uso de informações de controle, a banda que se torna disponível

aumenta a quantidade de informação útil trafegada, o que ajuda a aumentar a quantidade de chamadas VoIP simultâneas e de qualidade aceitável na rede. Uma das formas de se implementar um melhor uso da largura de banda sem fio disponível é através da compressão de cabeçalhos, objeto de estudo deste trabalho.

Por esse motivo, é extremamente indicado o uso de algoritmos de compressão de cabeçalhos robustos em redes sem fio, como o RoHC e o ECRTP. Porém, características desses algoritmos deixam em dúvida se seu comportamento seria satisfatório em redes de múltiplos saltos sem fio. Vale citar, por exemplo, que o RoHC é um algoritmo desenvolvido para redes com um único enlace sem fio, pois não tolera a chegada desordenada de pacotes no descompressor. Isso o torna perfeitamente aplicável a redes de telefonia celular, como as redes de terceira geração padronizadas pelo *3rd Generation Partnership Project* (3GPP). No entanto, esse comportamento não seria aceitável em redes *mesh*, pois essas, ao contrário das anteriores, são caracterizadas por múltiplos saltos sem fio, o que faz com que a implantação desses algoritmos torne-se potencialmente problemática.

Os algoritmos para compressão de cabeçalhos atualmente disponíveis oferecem um grande ganho de compressão, porém introduzem um novo problema relacionado ao fato de imporem a necessidade de sincronismo entre os dois elementos principais do sistema, compressor e descompressor. Esse comportamento diminui consideravelmente a tolerância característica das aplicações VoIP à perda de pacotes causando o problema da propagação da perda, ou seja, a perda de um único pacote pode ser suficiente para causar a dessincronização do contexto, e fazer com que todos os pacotes gerados em seguida sejam descartados no descompressor, por este não conseguir descomprimi-los corretamente, até que a sincronização seja restabelecida.

Outro problema está relacionado ao tipo de compressão de cabeçalhos utilizado. Como a compressão normalmente se aplica não apenas aos cabeçalhos de camada de aplicação e transporte, como também ao cabeçalho de camada de redes, para redes de múltiplos saltos sem fio dois tipos de compressão podem ser aplicados: *compressão salto-a-salto* e *compressão fim-a-fim*. A compressão salto-a-salto é aquela em que

cada ponto intermediário da rota entre um par fonte-destino executa a descompressão e nova compressão para extrair do cabeçalho de camada de redes a informação necessária para realizar o roteamento. A compressão fim-a-fim é possível quando os elementos intermediários são capazes de realizar o roteamento sem a necessidade de extrair a informação necessária dos cabeçalhos comprimidos. Nesse caso, o processo de compressão e descompressão é realizado apenas na fonte e no destinatário da comunicação.

A compressão de cabeçalhos salto-a-salto introduz um *overhead* de tempo na comunicação, e de espaço e processamento nos elementos intermediários. Cada par de roteadores ao longo do caminho deve estabelecer um par compressor-descompressor, e deve tratar de mantê-lo atualizado e tomar providências caso ocorra dessincronização de contextos. Portanto, se um único par ao longo do caminho perder a sincronização, toda a comunicação é comprometida. A compressão fim-a-fim por sua vez, introduz a necessidade de se obter mecanismos extras nos roteadores para que esses sejam capazes de rotear os pacotes com cabeçalhos comprimidos. Isso, no entanto, pode ser resolvido utilizando-se roteamento por rótulos, ao invés de se utilizar o roteamento da camada de redes.

Para o problema da propagação da perda por dessincronização de contexto, alguns algoritmos implementam mecanismos de resiliência de alta complexidade, como por exemplo o ECRTP [31] e o RoHC [28]. São utilizadas, por exemplo, mensagens periódicas para atualização do contexto, mensagens de confirmação ou solicitação de atualização (nesse caso o algoritmo necessita de um canal de retorno, nem sempre disponível no caso de redes sem fio), e armazenamento de informações de referência.

Porém, redes *mesh* oferecem uma alta taxa de erros no canal, devido às características do meio de transmissão utilizado. Como apenas um dispositivo pode transmitir por vez, quando mais de um elemento transmite ao mesmo tempo ocorre uma colisão [33]. Além disso, muitas outras coisas podem interferir na comunicação, como obstáculos no ambiente, e o recebimento da mesma informação através de caminhos diferentes no meio de propagação (problema do *multi-path fading*) [34]. Com essas características, o problema da propagação da perda pode se agravar, e os mecanismos de recuperação de

falhas utilizados pelos algoritmos podem não ser suficientes, principalmente no caso de perdas em rajada. Além disso, a largura de banda em redes sem fio é limitada, fazendo com que o número de usuários permitidos também seja limitado. O uso otimizado da largura de banda disponível pode maximizar o número de usuários da rede.

Alguns experimentos de simulação realizados inicialmente para este trabalho mostraram que o problema da propagação da perda pode aumentar consideravelmente a porcentagem de perda de pacotes na rede, de acordo com a quantidade de saltos ou carga de tráfego na rede. A Figura 3.1 mostra o gráfico da perda de pacotes em simulações de chamadas VoIP realizadas com o algoritmo RoHC U-mode, em um ambiente de rede *mesh* linear, com 6 nós. É possível identificar a porcentagem de perda de pacotes relacionada à problemas na rede, e a porcentagem de perda de pacotes devido à descarte de pacotes no descompressor. Esses experimentos foram realizados utilizando simulação em uma rede sem fio de múltiplos saltos linear com 6 nós, conforme será descrito no capítulo 4.

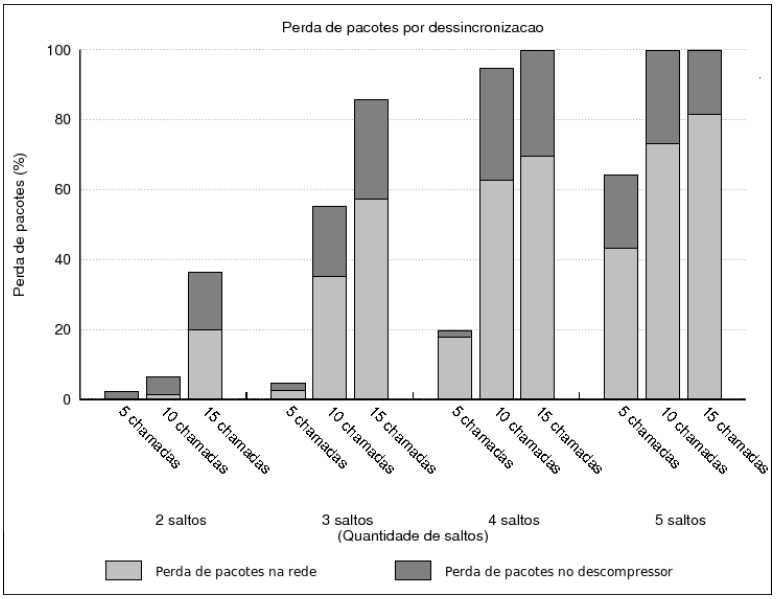

Figura 3.1: Porcentagem de perda de pacotes na rede e de perda por dessincronização.

No gráfico é possível observar que a perda por dessincronização chega a quase 30% do total de pacotes enviados pela fonte nos casos de maior número de chamadas simultâneas, ou seja, uma parte significativa das perdas de pacotes apresentadas está

relacionada a pacotes que chegaram em seu destinatário, porém foram descartados no descompressor por este estar dessincronizado. O problema da propagação da perda afeta consideravelmente as perdas de pacotes na rede. Por isso, neste trabalho é apresentado e avaliado um método que tem como objetivo eliminar o problema da propagação da perda, sem no entanto perder os ganhos oferecidos pela compressão de cabeçalhos ao sistema VoIP.

3.2 Solução proposta

Uma forma de solucionar o problema do descarte de pacotes no descompressor devido à dessincronização de contextos é eliminar totalmente a necessidade de manter os contextos de compressor e descompressor sincronizados. O problema da propagação da perda pode ser eliminado através da implementação de um algoritmo de compressão que mantenha nos contextos apenas as informações estáticas dos cabeçalhos IP/UDP/RTP, e não as dinâmicas, como implementam os algoritmos que necessitam de sincronização. Se os contextos armazenarem apenas informações estáticas, não haverá necessidade de sincronização. Esse tipo de compressão é denominada *compressão estática*.

A compressão estática oferece a vantagem de não precisar atualizar os contextos de compressor e descompressor, pois armazena apenas as informações estáticas, ou seja, aquelas que não mudam durante toda a sessão. Isso quer dizer que a perda de pacotes não fará com que pacotes seguintes sejam descartados no descompressor, eliminando assim o problema da propagação da perda. Outra vantagem apresentada pela compressão estática é a diminuição da quantidade de informações que devem ser armazenadas nos pontos onde ocorrem a compressão e a descompressão, já que o contexto passa a armazenar apenas as informações estáticas.

No entanto, o custo de se manter contextos sem a necessidade de sincronização se reflete no ganho de compressão, pois as informações dinâmicas trafegam no canal novamente e não são armazenadas nos contextos, como nos algoritmos convencionais [14]. Isso faz com que o tamanho do cabeçalho comprimido aumente com relação aos algoritmos de compressão convencionais, diminuindo o ganho de compressão atingido

(Fig. 3.2).

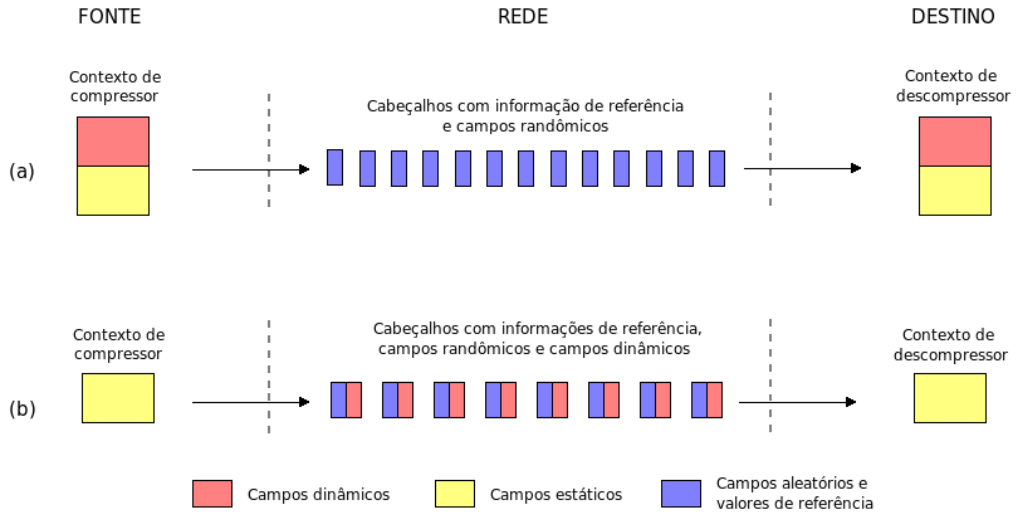

Figura 3.2: Comparação da localização das informações da compressão convencional (a) e da compressão estática (b).

A compressão estática pode diminuir o tamanho dos cabeçalhos para até 35% do tamanho original. Alguns algoritmos convencionais, que exigem sincronização, podem diminuir o tamanho dos cabeçalhos para menos de 10%. Experimentos realizados com compressão estática para este trabalho mostraram que apesar de esse mecanismo não apresentar o problema da propagação da perda, seu ganho de compressão não é grande o suficiente para oferecer ganhos significativos, em comparação aos algoritmos mais robustos. Portanto, é sugerido o uso de técnicas auxiliares para aumentar o ganho de compressão atingido, sem deixar de fazer uso do mecanismo de compressão estática.

A compressão de cabeçalhos estática se favorece da existência de campos cujos valores não mudam entre os pacotes de um fluxo de voz. Porém, alguns campos dinâmicos na maior parte do tempo de uma sessão também apresentam alguma redundância entre pacotes consecutivos, pois seguem um padrão de comportamento pré-estabelecido. Uma forma de oferecer um maior ganho de compressão para o mecanismo de compressão estática pode se aproveitar dessa redundância quase sempre presente.

Para se utilizar da redundância das informações dinâmicas sem retornar ao problema de sincronização de contextos e propagação da perda, após o processo de compressão estática é possível utilizar um mecanismo simples de *agregação de pacotes*. A agregação

de pacotes é uma técnica também utilizada para otimização do uso da largura de banda em redes sem fio. Seu objetivo principal é, através da agregação de vários pacotes, diminuir o *overhead* de tempo da camada de enlace imposto pelo MAC de redes sem fio 802.11, e assim diminuir o número de perda de pacotes por contenção na camada de enlace, e o aumento do número de retransmissões [3]. Além disso, a agregação também ajuda a diminuir o uso da largura de banda para trafegar informações de controle, pois diminui a quantidade de cabeçalhos MAC enviados à rede (Fig. 3.3).

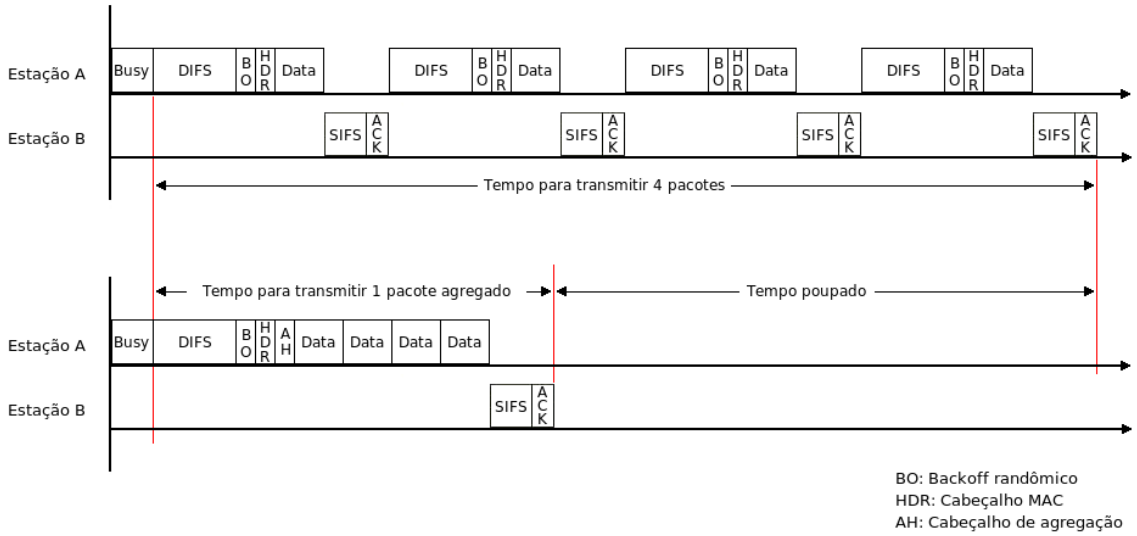

Figura 3.3: Vantagens da agregação de pacotes em redes 802.11. Fonte: [3].

A agregação de pacotes foi desenvolvida originalmente para unir pacotes com o mesmo destino em comum, a partir de uma mesma origem. Sendo assim, a agregação de pacotes convencional agrega pacotes de diferentes fluxos que possuem uma rota inteira, ou parte, em comum. Existem dois tipos de agregação de pacotes, a saber, agregação *fim-a-fim* e *salto-a-salto*. A agregação fim-a-fim é realizada quando os pacotes são agregados no primeiro ponto em comum da rota, e são desagregados apenas no último ponto em comum da rota, seguindo todo o percurso agregados em um mesmo pacote. A agregação salto-a-salto, como o nome diz, é realizada em cada roteador da rede, ou seja, cada roteador desagrega os pacotes, faz uma reorganização com pacotes vindos através de outros enlaces de entrada, e reagrega novamente aqueles que possuirem o mesmo enlace de saída [3].

Para que a agregação de pacotes possa contribuir para a compressão de cabeçalhos apenas pacotes do mesmo fluxo podem ser agregados. Por isso, a agregação de pacotes introduz um atraso de enfileiramento no processo, pois o compressor precisa esperar a formação de k pacotes, onde k é denominado *grau de agregação*, para formar um pacote de agregação. Esse atraso imposto se reflete na qualidade da chamada, como foi comprovado com os experimentos realizados neste trabalho, o que faz com que este tipo de mecanismo não seja a melhor opção em ambientes com poucos saltos sem fio, ou baixa carga de tráfego. Como foi possível observar a partir dos resultados deste trabalho, para cenários com maiores quantidades de saltos sem fio ou de chamadas simultâneas, o ganho oferecido pela agregação de pacotes compensa o atraso imposto pelo mecanismo. Por isso, é importante a utilização de um baixo grau de agregação, pois este valor é diretamente proporcional ao atraso que será imposto ao tráfego.

Feita a agregação, as informações dinâmicas redundantes entre os cabeçalhos dos pacotes que foram agregados são retiradas dos cabeçalhos comprimidos e mantidas em um único cabeçalho, mais externo, denominado *cabeçalho de agregação* (Fig. 3.4). São ditas informações redundantes aqui aquelas de campos que assumirem valores sequenciais ou que assumirem o mesmo valor para os pacotes agregados.

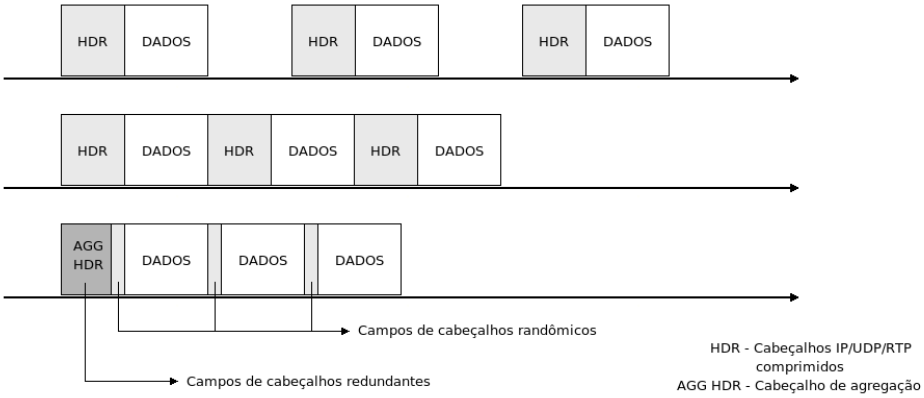

Figura 3.4: Solução cooperativa: compressão + agregação.

O cabeçalho de agregação é um cabeçalho que contém os campos dos cabeçalhos IP/UDP/RTP cujo valor é igual para todos os pacotes agregados. Portanto, quando o pacote de agregação atingir o destinatário, este será capaz de, a partir do cabeçalho

de agregação e dos campos randômicos, reconstituir o cabeçalho comprimido estaticamente de cada um dos pacotes agregados, e assim poderá prosseguir com o processo de desagregação e posterior descompressão estática normalmente. Os experimentos realizados para este trabalho mostraram que esse mecanismo de compressão e agregação pode aumentar o ganho de compressão de aproximadamente 60%, com apenas o uso da compressão estática, para mais de 80%.

Capítulo 4

Experimentos

Este capítulo descreve o ambiente utilizado para a realização dos experimentos descritos neste trabalho, bem como a metodologia de análise dos resultados, e as métricas utilizadas. A seção 4.1 detalha elementos de *hardware* e *software* utilizados para a execução dos experimentos. A seção 4.2 descreve o projeto de experimentos, cenários utilizados, e as métricas escolhidas para a avaliação dos algoritmos.

4.1 Ambiente de simulação

Segundo Jain [8], existem três técnicas que podem ser utilizadas para realizar avaliação de desempenho de sistemas em geral. São elas *modelagem analítica*, *simulação* e *medição*. A escolha da metodologia apropriada é fundamental para um trabalho de avaliação eficiente, e depende totalmente do sistema a ser avaliado.

Dentre os fatores que devem ser levados em consideração no momento da escolha da técnica está o estado atual do sistema. Medições só podem ser realizadas se o sistema, ou algo similar, já existir e estiver disponível para o avaliador. Outro fator que deve ser levado em consideração é o tempo disponível para a realização dos trabalhos. Geralmente, avaliações feitas através de medições são mais demoradas e oferecem um custo maior, pois exigem equipamento específico. A tabela 4.1 mostra os principais critérios que devem ser levados em consideração no momento da escolha do método de avaliação a ser utilizado.

Tabela 4.1: Principais critérios usados para escolher uma técnica de avaliação de desempenho. Fonte: [8].

Critério	**Modelagem Analítica**	**Simulação**	**Medição**
Estágio	Qualquer	Qualquer	Pós-prototipação
Tempo exigido	Pequeno	Médio	Varia
Ferramentas	Analistas	Linguagens computacionais	Instrumentação
Acurácia	Baixa	Moderada	Varia
Custo	Pequeno	Médio	Alto

Para os experimentos realizados neste trabalho, a técnica utilizada foi a simulação, por apresentar a configuração de critérios mais razoável para o sistema a ser avaliado. O ambiente foi composto pelo simulador de redes Network Simulator 2 (NS-2) [35], com algumas alterações (descritas na subseção 4.1.1), e pelo Akaroa-2 [36], ferramenta de cálculos estatísticos e paralelização de processamento.

Experimentos iniciais foram realizados em um ambiente de processamento paralelizado, formado por um servidor de simulação, com processador Intel Xeon 2.4GHz, 1 GB de RAM e 200 GB de disco rígido, configurado com o sistema operacional Linux Debian Lenny, e as ferramentas NS-2 e Akaroa-2. Foram utilizadas 5 máquinas clientes de simulação com processador Intel Pentium 4 de 2.8GHz, 1 GB de memória RAM e 40 GB de disco rígido, configuradas com sistema operacional Linux Ubuntu 7.04 Feisty Fawn, conectados através de um *switch* na rede local do laboratório do PPGI, caracterizando um ambiente com 6 máquinas (o servidor também foi usado como cliente) executando as simulações em paralelo.

As ferramentas NS-2 e Akaroa-2 foram exportadas a partir do servidor para as máquinas cliente através da rede, para que essas pudessem executar as simulações localmente, utilizando a ferramenta NFS *Kernel Server*, instalada em todas as máquinas. O Akaroa-2 foi a ferramenta responsável por gerenciar os processos em execução em cada uma das máquinas cliente.

Os experimentos finais, apresentados nesta dissertação, foram realizados em um *notebook* Acer modelo TravelMate6292, com processador Intel Core 2 Duo de 2.0GHz, 2 GB de memória RAM e 250 GB de disco rígido, configurado com o sistema operacional

Linux Ubuntu 8.04 Hardy Heron e as ferramentas NS-2 e Akaroa-2, que foi utilizado como servidor e clientes de simulação, caracterizando um ambiente com 2 processos executando simulações em paralelo.

4.1.1 Network Simulator

O NS-2 é um simulador de eventos muito utilizado na atualidade para simulação de redes cabeadas ou sem fio, que fornece apoio substancial para a simulação de TCP, roteamento, e protocolos *multicast* sobre redes cabeadas e sem fio (local e satélite). É atualmente suportado pelos grupos *Simulation Augmented by Measurement and Analysis for Networks* (SAMAN) e *Collaborative Simulation for Education and Research* (CONSER). É uma ferramenta implementada em C++ e em uma versão orientada a objetos do Tcl, chamada OTcl. Pode ser executado sobre as plataformas Linux, BSD, OS X e Windows [35].

Este simulador foi escolhido para fazer parte do ambiente de experimentos devido ao seu alto respaldo e utilização na comunidade acadêmica, e devido à boa representatividade dos seus resultados. Outro motivo levado em consideração foi o fato de que o NS-2 vem sendo desenvolvido e atualizado há mais de 15 anos, por vários grupos de pesquisa, inclusive de grandes empresas multinacionais. A estrutura do simulador, que permite a inclusão de novos módulos facilitanto assim a avaliação de novas tecnologias, foi outro fator importante na sua escolha.

A versão utilizada para a realização dos experimentos deste trabalho foi a versão 2.29. Esta versão foi escolhida devido à disponibilidade de um *patch* para melhoria na modelagem das camadas de enlace e física do simulador. Em [37], os autores mostraram a necessidade e a importância de uma boa modelagem a nível de camada de enlace e física para a qualidade dos resultados obtidos através de simulação de redes sem fio. Além desta, várias outras alterações foram aplicadas à versão original do NS-2, com o objetivo de melhorar a qualidade, rapidez e fidelidade das simulações e dos modelos simulados, além de acrescentar os módulos extras necessários aos nossos experimentos. As principais alterações serão descritas nas próximas subseções.

4.1.1.1 Modelagem do canal

Uma das principais alterações aplicadas ao NS-2 foi sobre a modelagem do canal sem fio. O trabalho apresentado em [37] mostrou que a escolha do modelo de propagação pode afetar nos resultados das simulações com NS-2, sendo importante então a escolha do modelo de propagação adequado.

O NS-2 oferece três modelos de propagação: *Free Space*, *Two-Ray ground* e *Log-normal shadowing* (ou apenas *shadowing*). Os modelos *Free Space* e *Two-Ray ground* são determinísticos, e dependem basicamente da distância, da força do sinal transmitido, e dos ganhos das antenas. Ambos assumem a existência de visada entre os dispositivos transmissor e receptor, e o *Two-Ray ground* além disso considera a existência de reflexo do sinal numa superfície plana. Porém, segundo [38], esses modelos são muito simplistas para modelar a propagação do sinal em redes sem fio.

O modelo *shadowing* depende de valores determínisticos e estocásticos para determinar a força do sinal dada uma distância entre dois dispositivos. Esse modelo leva em consideração a existência de obstáculos entre o transmissor e o receptor, oferecendo assim uma modelagem mais realística. Portanto, a força do sinal calculada pelo modelo *shadowing* depende de dois parâmetros que devem ser configurados de acordo com o ambiente a ser simulado: expoente α de degradação no percurso (*path loss exponent*) e o desvio padrão σ.

Porém, no trabalho apresentado em [39], muitos erros foram identificados no modelo *shadowing* do NS-2, relacionados à recepção e transmissão de pacotes, dentre outros problemas identificados. Este trabalho também apresenta todas as correções para os problemas citados.

Para nossos experimentos foram simulados canais 802.11b [9] do Instituto de Engenheiros Eletricistas e Eletrônicos (*Institute of Electrical and Electronics Engineers* - IEEE), por ser o padrão de redes sem fio mais utilizado na atualidade. Foi utilizado o modelo *shadowing* com as alterações sugeridas em [39], e a simulação dos canais foi configurada com base no trabalho apresentado em [40]. O modelo *shadowing* foi configurado com *path loss exponent* de $3,5$ e desvio padrão de $4,0$, segundo Rappaport

[41], valores utilizados para ambientes de área urbana.

4.1.1.2 Módulos de algoritmos de compressão de cabeçalhos

O simulador NS-2 não possui originalmente nenhum módulo de compressão de cabeçalhos. Portanto, durante o desenvolvimento deste trabalho, para que os experimentos pudessem ser realizados, módulos extras foram implementados para que fosse possível simular o funcionamento dos algoritmos de compressão a serem avaliados. Como o objetivo era a análise dos algoritmos sobre redes *mesh*, um novo módulo foi adicionado à estrutura original de um nó sem fio usada pelo NS-2.

A estrutura original de um nó sem fio simulado pelo NS-2 é apresentada na Figura 4.1. Quando um pacote passa a existir dentro de um nó como o descrito nessa imagem, ele atinge o ponto `entry_`. Se o nó em questão for o próprio nó de origem do pacote, ele veio de um elemento do tipo `Src/Sink`, que são os elementos responsáveis por gerar e receber todo tráfego da rede. O objeto `addr demux` é o elemento responsável por decidir para onde o pacote deve ser encaminhado, ou seja, ele decide se o pacote é destinado ao nó em questão ou não. Se for, o pacote é encaminhado ao objeto `port demux`, que vai decidir para qual elemento `Src/Sink` o pacote deve ser encaminhado.

Quando o pacote não é destinado ao nó em questão, ou seja, trata-se do próprio nó que gerou o pacote, ou trata-se de um nó intermediário, o pacote é enviado pelo `addr demux` para o elemento responsável por fazer o roteamento em camada de redes, o `RTAgent`. O `RTAgent` é o objeto que vai assumir o algoritmo de roteamento configurado no *script* utilizado para a simulação. Originalmente o NS-2, na versão utilizada, oferece os algoritmos de roteamento AODV, DSDV e DSR. O elemento `RTAgent` passa o pacote para o objeto `LL`, que é responsável por identificar que tipo de roteamento está sendo feito, e simular o uso do protocolo ARP para checar os endereços MAC do nó para o qual o pacote deve ser enviado.

Após passar pelo objeto `LL`, o pacote passa pela fila da camada de enlace, implementada pelo objeto `IFq`. O comportamento da fila também é configurável através do *script* de simulação. Em seguida o pacote atinge o objeto `MAC`, responsável por imple-

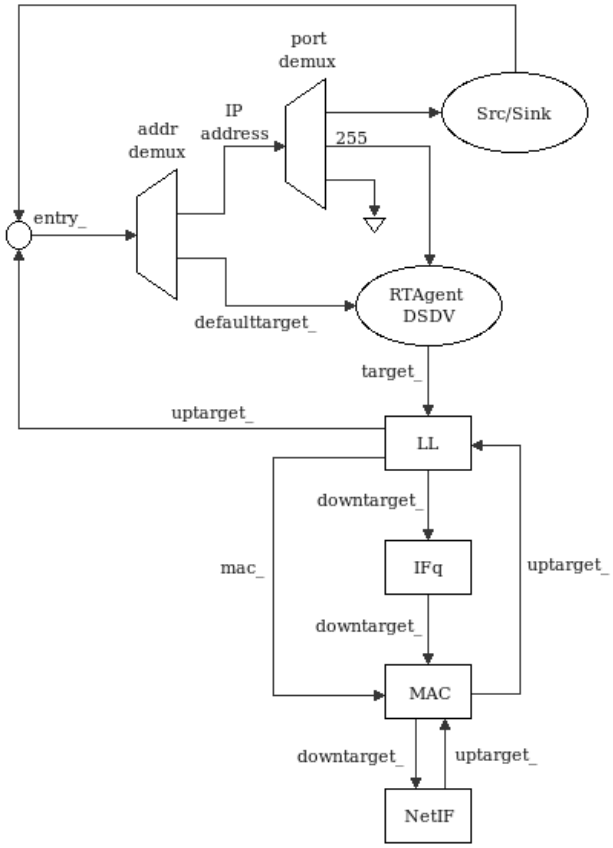

Figura 4.1: Estrutura de um nó sem fio no NS-2. Fonte: [4]

mentar todo o procedimento relacionado ao acesso ao meio, dependendo da tecnologia utilizada, no caso deste trabalho, a tecnologia 802.11. O `MAC` envia o pacote então para o objeto `NetIF`, que representa a interface pela qual o pacote será transmitido, e é o objeto responsável por entregar o pacote ao objeto `MAC` do dispositivo receptor.

Para simular o uso de algoritmos de compressão de cabeçalhos, foi criado um novo módulo, que foi incorporado à estrutura original de um nó sem fio no NS-2. A Figura 4.2 mostra a nova estrutura do nó sem fio com compressão de cabeçalhos, com o elemento `HC` representando o objeto responsável pela compressão.

Com o módulo implementado, é possível configurar o nó sem fio para utilizar os algoritmos de compressão de cabeçalhos RoHC U-mode, compressão estática e compressão estática com agregação de pacotes, além de configurá-lo para não utilizar nenhum mecanismo de compressão. É possível também configurar o algoritmo RoHC para trabalhar com o CoHC. A implementação foi feita em sua maior parte utilizando

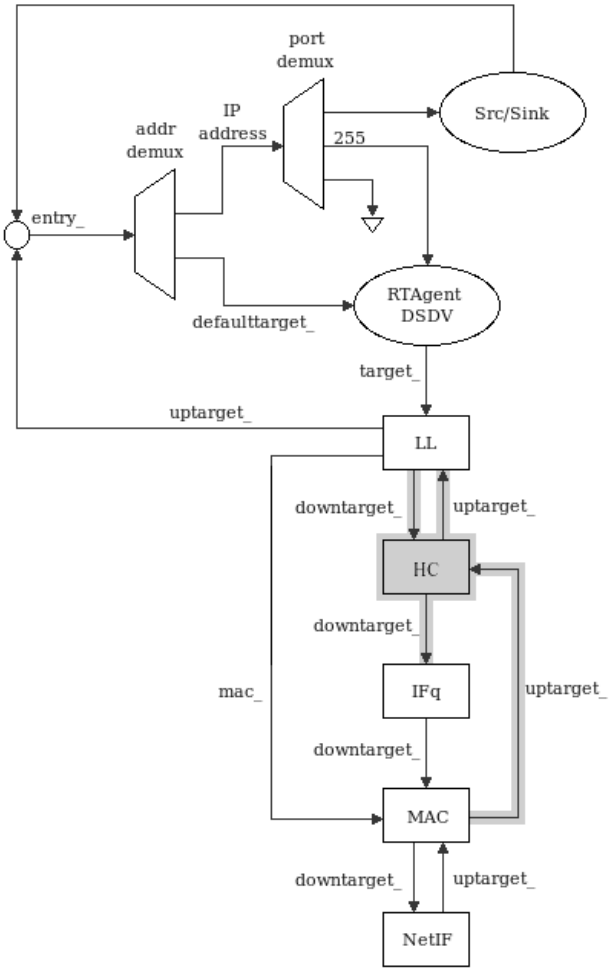

Figura 4.2: Estrutura de um nó sem fio no NS-2 com compressão de cabeçalhos. Em destaque, o novo elemento e a comunicação entre ele e os demais objetos.

a linguagem C++, porém a linguagem oTcl também foi utilizada para fazer a comunicação entre o módulo novo e os demais elementos que formam a estrutura do nó sem fio. O algoritmo de compressão pode ser escolhido e configurado diretamente a partir do *script* de simulação.

4.1.1.3 Roteamento

O roteamento é um dos pontos mais importantes quando trata-se de redes de múltiplos saltos sem fio. Fato que comprova isto é a grande quantidade de protocolos de roteamento desenvolvidos para *mobile ad-hoc networks* (MANETs). Como principais protocolos podemos citar o reativo *Ad-hoc On-Demand Distance Vector* (AODV), e o

pró-ativo *Optimized Link State Routing* (OLSR), desenvolvidos por grupos de trabalho do IETF. Apesar de as redes *mesh* apresentarem características particulares com relação às redes *ad-hoc*, os protocolos de roteamento originalmente desenvolvidos para MANETs têm sido utilizados em redes *mesh*, o que se tornou objeto de pesquisa de vários grupos de pesquisa.

Como visto na seção 3.1, existem dois tipos de compressão de cabeçalhos, salto-a-salto e fim-a-fim. Para os experimentos realizados neste trabalho, foi escolhida a utilização de compressão de cabeçalhos fim-a-fim, por não apresentar custo extra nos nós intermediários de uma rota entre um par fonte-destino. Para que a compressão de cabeçalhos fim-a-fim seja possível de ser realizada, é necessário que os roteadores da rede sejam capazes de rotear pacotes com cabeçalhos comprimidos. Como a compressão de cabeçalhos se aplica inclusive ao cabeçalho IP, isso significa dizer que os roteadores devem executar o roteamento dos pacotes sem precisar extrair as informações dos cabeçalhos IP.

Com o objetivo de satisfazer essa necessidade, optou-se por utilizar roteamento por rótulos, implementado através do *Multi-protocol Label Switching* (MPLS) [42]. O MPLS é conhecido por realizar o roteamento entre as camadas de redes e de enlace, caracterizando assim um roteamento em camada 2,5 (Figura 4.3). O MPLS funciona basicamente com a adição de um rótulo nos pacotes de tráfego (e nesse caso se torna indiferente ao tipo de dados transportado, podendo ser tráfego IP ou outro qualquer) no roteador de entrada do *backbone* (roteador de borda) e, a partir daí, todo o encaminhamento pelo *backbone* passa a ser feito com base neste rótulo, que é removido quando o pacote deixa o *backbone*.

A implementação de MPLS utilizada para o NS-2 está disponível em [43], denominada MPLS *Network Simulator* (MNS) versão 2.0, que está disponível para o NS-2 versão 2.26. Foi necessária uma pequena adaptação do módulo para sua utilização na versão 2.29, e na estrutura do nó sem fio do NS-2, pois o módulo original somente se aplicava a nós cabeados.

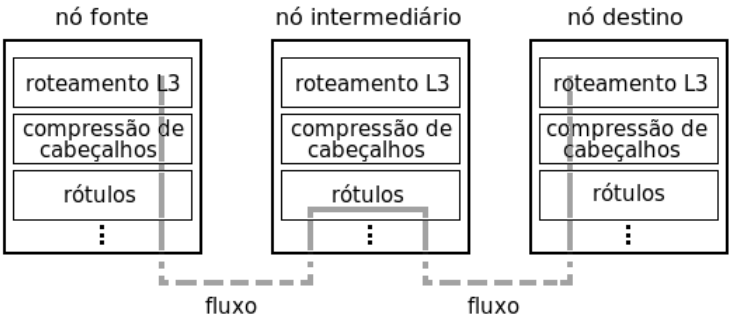

Figura 4.3: Roteamento por rótulos.

4.1.2 Akaroa

Segundo [44], amostras obtidas através de simulações estocásticas devem ser cuidadosamente trabalhadas antes que o avaliador possa fazer conclusões a respeito do comportamento do sistema. Inferência estatística é necessária quando um modelo de simulação correto produz resultados diferentes, porém também corretos, como é o caso de simulações estocásticas. Isso se deve à utilização de variáveis aleatórias nesse tipo de processo, e esse comportamento não deve ser ignorado.

Observações coletadas no período de transiente (momento inicial da simulação) não representam corretamente o parâmetro a ser avaliado. Esse é um período em que o sistema ainda não está estável, portanto suas observações não são confiáveis. O Akaroa-2 utiliza técnicas para identificar o final do período de transiente e início do período de estado estacionário das simulações. Então, suas estimativas são calculadas apenas a partir de amostras coletadas no estado estacionário.

Já as observações geradas após o estado de transiente das simulações, mesmo estando no estado estacionário, apresentam correlação em seus valores gerados sequencialmente. Isso se deve às condições iniciais dadas ao modelo, e à duração do processo de simulação. Por isso, quanto maior a duração da simulação e maior for a quantidade de amostras geradas, essa correlação tende a diminuir pois as amostras ficam distantes no tempo. Simulações de horizonte infinito como as administradas pelo Akaroa-2 calculam a variância das estimativas levando em consideração a correlação entre as amostras, afim de evitar resultados extremamente otimistas ou extremamentes pessimistas, que

divergem do verdadeiro comportamento do sistema. Para maiores detalhes, consulte [44].

O Akaroa-2 é uma ferramenta cujo principal objetivo é automatizar totalmente a paralelização de simulações e controlar a sua duração, levando em consideração todos esses fatores relacionados a transiente e correlação entre as amostras, além de considerar também a precisão exigida para os resultados. Seu funcionamento é simples, utilizando-se de múltiplas máquinas de simulação trabalhando paralelamente. Uma instância de um modelo de simulação sequencial é carregada em várias estações de trabalho diferentes, operando como máquinas de simulação conectadas através da rede, e uma unidade central é responsável por coletar assincronamente estimativas parciais de cada processador e calcular uma média geral.

O Akaroa-2 permite que o mesmo modelo de simulação seja executado em diferentes processadores em paralelo, com o objetivo de produzir observações independentes e identicamente distribuídas (IID), iniciando cada replicação da simulação com diferentes séries de números fornecidos por geradores de números pseudo-aleatórios (*Pseudo Random Number Generator* - PRNG).

Essencialmente, um processo mestre (Akmaster) é iniciado em um processador, que age como um gerente, enquanto um ou mais processos escravos (Akslave) são iniciados em cada estação que fará parte do experimento de simulação, formando um grupo de máquinas de simulação (Figura 4.4). O processo Akrun inicia a simulação, e o Akaroa-2 fica responsável pelas tarefas fundamentais relacionadas a carregar o mesmo modelo de simulação em todos as estações, controlando o experimento como um todo, e oferecendo um controle automatizado da precisão dos resultados da simulação.

No início, um teste de estacionariedade é aplicado localmente a cada replicação, para determinar o início do estado estacionário em cada série separadamente, e a partir de então as observações são coletadas. Cada escravo da simulação gera observações, e quando o montante de observações recolhidas é suficiente para produzir uma estimativa razoável, dizemos que um *checkpoint* foi atingido, e nesse momento o analisador local submete a estimativa calculada ao analisador global, localizado no processo Akmaster.

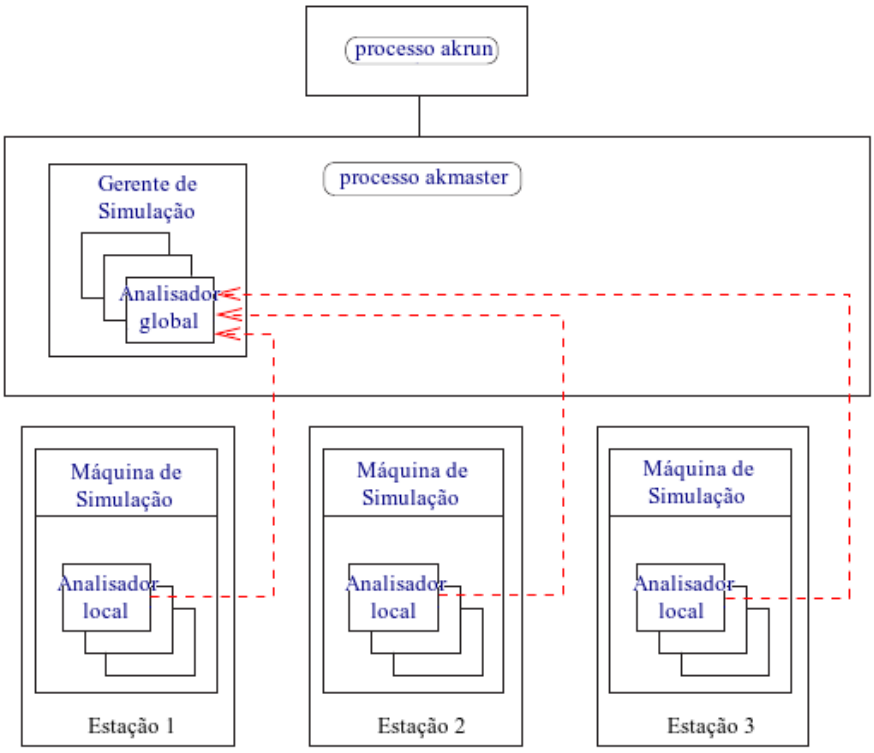

Figura 4.4: Esquema do Akaroa-2. Fonte: [5]

O analisador global calcula uma estimativa global, baseada em estimativas locais entregues pelos diferentes escravos, e verifica se a precisão necessária foi alcançada, caso em que a simulação global está concluída. Enquanto a precisão não é atingida, os escravos continuam trabalhando e gerando novas estimativas locais.

Dado o nível de confiança, a cada *checkpoint* do Akaroa-2, o intervalo de confiança é calculado, e a partir dele a precisão relativa atingida. Se a precisão relativa atingida for menor que a precisão máxima requerida, a simulação é encerrada. Senão, a simulação continua gerando observações até o próximo *checkpoint*. Para os experimentos realizados neste trabalho, foram exigidos um nível de confiança de 95%, e precisão relativa máxima de 5%.

4.2 Projeto de experimentos

Nesta seção, são definidos os principais aspectos dos experimentos realizados, como as métricas, os parâmetros e os fatores utilizados. A tabela 4.2 resume esses aspectos, e

as subseções seguintes detalham métricas e cenários escolhidos.

Tabela 4.2: Configuração dos experimentos.

Fatores	**1. Algoritmo de compressão de cabeçalhos**: Sem compressão (None), protocolo RoHC (RoHC), compressão estática (SHC), compressão estática + agregação (SHC+AG)
Parâmetros	**1. Cenário**: Linear (cenário 1) e Árvore (cenário 2) **2. Número de saltos (cenário 1)**: 2, 3, 4 e 5 saltos **3. Número de chamadas simultâneas (cenários 1 e 2)**: de 2 a 38 chamadas
Métricas	MOS, perda de pacotes, atraso na rede, ganho de compressão, eficiência de largura de banda
Carga do sistema	Tráfego CBR, bidirecional, com comportamento de chamadas VoIP de 60 segundos, utilizando o *codec* de voz G.729a, com períodos de fala e silêncio

O fator de avaliação é o algoritmo de compressão de cabeçalhos utilizado. Como parâmetros de influência no sistema foram escolhidos o cenário utilizado, o número de saltos, avaliado apenas no cenário linear, e o número de chamadas simultâneas, avaliado em ambos os cenários definidos.

4.2.1 Cenários

Para os nossos experimentos foi utilizada compressão de cabeçalhos fim-a-fim, por não necessitar que todos os nós realizem compressão e descompressão. Porém, a utilização de compressão fim-a-fim exige que os roteadores intermediários realizem o encaminhamento dos pacotes sem as informações do cabeçalho IP. Para suprir essa necessidade, foi utilizada uma configuração de roteamento em camada 2, 5. Atendendo a esta exigência principal para o funcionamento dos algoritmos, os cenários foram escolhidos então com o objetivo de explorar as características principais das redes *mesh*.

O primeiro cenário representa um *backbone mesh* com os roteadores posicionados em uma linha reta, como mostra a Figura 4.5. A distância entre nós consecutivos é de 25 metros. Neste cenário, os nós podem se comunicar diretamente apenas com seus vizinhos imediatamente anterior e posterior. A idéia que envolve a utilização deste

cenário está na característica das redes *mesh* de serem formadas por múltiplos saltos sem fio. Portanto, os experimentos realizados sobre este cenário têm por objetivo avaliar o comportamento dos algoritmos de compressão conforme o número de saltos sem fio na comunicação.

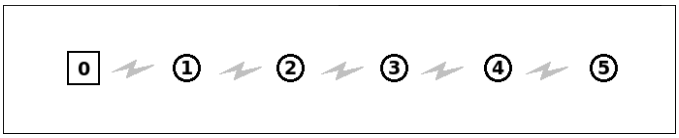

Figura 4.5: Cenário linear.

O segundo cenário representa um *backbone mesh* com os roteadores posicionados em formato de árvore (Figura 4.6). A distância entre os nós e cada um de seus vizinhos diretos é de 25 metros. Neste cenário, a idéia principal é avaliar os algoritmos em uma rede cujos roteadores precisam lidar com tráfegos vindos de fontes diferentes. Em redes *mesh* esse comportamento é comum. As chamadas VoIP foram geradas a partir dos nós-folha da árvore representada na figura (nós 3, 4, 5 e 6), com destino ao nó 0, em destaque. Esse comportamento do tráfego também é comum em muitas redes *mesh* que possuem um elemento *gateway*, que dá acesso a outras redes ou à Internet.

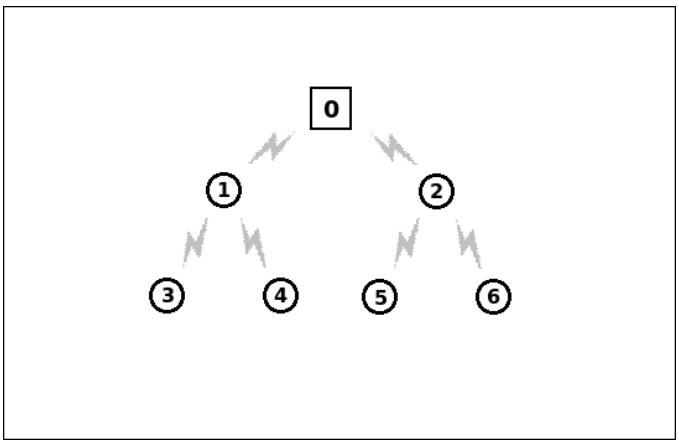

Figura 4.6: Cenário em árvore.

Nestes cenários, quatro algoritmos/abordagens de compressão de cabeçalhos foram testadas. A primeira não envolve nenhum mecanismo de compressão, sendo utilizada apenas para fins de comparação. Uma abordagem é com a utilização do RoHC U-mode, por ser um protocolo cujo funcionamento exige a utilização de sincronização de

contexto. Também foi avaliado um algoritmo simples de compressão estática, também para fins de comparação. Por último, a abordagem sugerida neste trabalho, que diz respeito ao uso de compressão estática com agregação de pacotes. O objetivo da utilização desses algoritmos é fazer uma análise comparativa entre eles, destacando em que ponto cada um se destaca em relação aos outros, e avaliar se a abordagem de compressão estática e agregação de pacotes sugerida neste trabalho é válida e oferece vantagens sobre as outras.

4.2.2 Métricas de avaliação

As métricas foram escolhidas levando em consideração o fato de o foco do trabalho ser o uso de compressão de cabeçalhos para aplicações VoIP. É sabido que este tipo de aplicação, por ser uma aplicação de tempo real, impõe algumas exigências sobre a rede onde é implantado. Dentre essas exigências, podemos citar a *perda de pacotes* e o *atraso na rede*. Altos atrasos podem comprometer a interatividade do sistema VoIP, e a perda de pacotes em grande quantidade pode fazer com que a comunicação entre os interlocutores se torne confusa. Portanto, esses fatores foram utilizados como métricas de avaliação para os algoritmos testados.

Também foram utilizadas métricas específicas para avaliar o comportamento de algoritmos de compressão de cabeçalhos. As métricas de *ganho de compressão* e *eficiência de largura de banda* são métricas que têm relação entre si pois são calculadas com base no tamanho do pacote e do seu cabeçalho antes e depois da compressão. Porém, a apresentação das duas é importante para o trabalho pois uma está relacionada apenas ao tamanho dos cabeçalhos (ganho de compressão) e a outra está relacionada ao ganho proporcionado com relação ao tamanho total do pacote (eficiência de largura de banda).

Como medida da qualidade da fala foi usado o *Mean Opinion Score* - MOS, calculado a partir do fator R, que leva em consideração vários fatores relacionado ao tráfego, como a perda de pacotes, o atraso, *jitter*, *codec* usado, etc.

De acordo com o comportamento do tráfego na rede, dado o ganho de compressão

e eficiência de largura de banda oferecidos pelo algoritmo de compressão utilizado, os valores de atraso e perda serão afetados. O quantitativo deste impacto poderá ser observado relacionando os valores dessas métricas específicas com os valores medidos de ganho de compressão e eficiência de largura de banda. Podemos dizer então que essas métricas influenciam o MOS, já que influenciam fatores que são utilizados para o cálculo da sua pontuação. As subseções a seguir dão descrições mais detalhadas sobre as métricas utilizadas.

4.2.2.1 Ganho de compressão

Indica o quanto o algoritmo proporciona em termos de compressão, ou seja, o quanto do cabeçalho original não foi enviado à rede. O ganho de compressão envolve duas medidas diretamente relacionadas com o funcionamento de um compressor em um algoritmo de compressão de cabeçalhos, e indica o quanto esta entidade proporciona em termos de compressão. A medida de ganho de compressão é dada pela razão entre o total de bytes ocupados por cabeçalhos IP/UDP/RTP que foi "economizado" em uma dada conexão (pelo uso de um algoritmo de compressão) e o total de bytes originalmente ocupados pelos cabeçalhos IP/UDP/RTP. Esta medida não possui unidade, e o seu valor oscila entre 0 e 1.

Um valor de ganho de compressão igual a zero corresponde a uma situação onde o algoritmo de compressão não consegue evitar a transmissão de nenhum byte através do canal, o que implica em nenhum alívio na utilização da largura de banda e indicando que o algoritmo não atende ao que se propõe fazer. Um valor igual a 1 indica uma situação ideal, onde um dado algoritmo consegue evitar que todos os bytes originalmente ocupados por cabeçalhos sejam transmitidos, e a largura de banda é totalmente dedicada à transmissão de dados úteis (*payload*). Um valor de ganho de compressão igual a 1 é impossível de ser atingido, considerando que pelo menos as informações estáticas precisam ser transmitidas pelo canal no momento de inicialização do contexto do descompressor. Já um valor de ganho de compressão igual a zero é obtido quando não há compressão de cabeçalhos.

O ganho de compressão (GC) é obtido através da relação [13]:

$$GC = \frac{T_s}{T_o} \tag{4.1}$$

onde T_s é o tamanho total de bytes economizados com a compressão, e T_o é o tamanho total de bytes ocupados pelos pacotes, contando com *payload* e cabeçalhos descomprimidos.

4.2.2.2 Eficiência de largura de banda

Outra métrica importante é a medida de eficiência da largura de banda, e indica o quão importante é a contribuição dada por um determinado algoritmo de compressão de cabeçalhos para a diminuição de utilização da largura de banda, ou seja, indica o quanto da banda utilizada foi liberada para o tráfego de carga útil devido à compressão dos cabeçalhos. É dada pela razão entre o total de bytes de *payload* transmitidos e o total de bytes efetivamente utilizados para transmissão, incluindo-se *payload* e cabeçalhos comprimidos. Seu valor não possui unidade, e varia de 0 a 1.

Um valor de eficiência de largura de banda igual a 0 é impossível de ser alcançado, pois o tamanho do *payload* é levado em consideração em seu cálculo. Porém, quanto mais próximo de 0 for o valor, menor é o ganho em termos de largura de banda proporcionado pela utilização do algoritmo. Da mesma maneira, um valor de eficiência de largura de banda igual a 1 é impossível de ser alcançado, já que não é possível reduzir o tamanho dos cabeçalhos a zero, mas quanto mais próximo de 1 for a medida tão mais relevante será a contribuição dada pela utilização de um dado algoritmo de compressão de cabeçalhos.

A eficiência de largura de banda (EB) é obtida através da relação [13]:

$$EB = \frac{T_u}{T_c} \tag{4.2}$$

onde T_u é o tamanho total dos bytes ocupados pelos dados de voz nos pacotes transmitidos, e T_c representa o tamanho total em bytes ocupados pelo pacote inteiro, levando

em consideração *payload* e cabeçalhos comprimidos.

4.2.2.3 Atraso na rede

O atraso dos pacotes é um fator primário de influência na qualidade de chamadas de voz na rede. É sabido que o limite de tempo para que o ouvido humano não perceba atraso na reprodução da fala é de 150 ms. Portanto, se a rede impõe atrasos muito grandes, o impacto deste fator na qualidade da chamada será perceptível.

Como o objetivo da avaliação realizada é analisar o comportamento do tráfego de voz na rede, dadas as alterações realizadas pelos algoritmos sobre os cabeçalhos dos pacotes, apenas o atraso de rede foi mensurado. O atraso de rede leva em consideração outros 3 tipos de atrasos: atraso de fila, atraso de propagação e atraso de serialização [6].

O atraso de fila é aquele que o pacote enfrenta quando precisa aguardar para ser transmitido por um roteador. O atraso de fila para um pacote específico depende da quantidade de outros pacotes que chegaram antes e que já estão na fila esperando pela transmissão no enlace. Se a fila estiver vazia e não houver nenhuma transmissão de pacote em curso, então o tempo de atraso de fila do pacote será zero. Por outro lado, se o tráfego estiver pesado e houver muitos pacotes esperando para serem transmitidos, o atraso de fila será longo.

Quanto ao atraso de serialização, ele é relacionado ao tamanho do pacote e à velocidade de transmissão do enlace, ou seja, é o tempo que o pacote leva até ser colocado totalmente no enlace. Quanto maior o tamanho do pacote, maior será o atraso de transmissão. A compressão de cabeçalhos pode diminuir o atraso de transmissão do pacote, tendo em vista que diminui o tamanho dos pacotes.

O tempo que o pacote leva para chegar de uma ponta à outra do enlace determina o atraso de propagação. Essa medida é relacionada ao tipo do enlace e ao seu tamanho, pois o bit se propaga à velocidade específica do meio.

4.2.2.4 Perda de pacotes

Outro fator que influencia na qualidade de aplicações de tempo real, inclusive VoIP, é a perda de pacotes na rede. As aplicações VoIP oferecem uma certa tolerância à perda de pacotes, pois pequenas perdas são imperceptíveis ao ouvido humano. Porém, essa tolerância é bem limitada, e a perda de pacotes em uma alta porcentagem pode oferecer um impacto negativo na qualidade da chamada, de forma a prejudicar o entendimento dos interlocutores.

A perda de pacotes em sistemas VoIP pode se dar por dois motivos [6]: erro de transmissão de bits, ou descarte de pacotes nos roteadores da rede, ou no *buffer* de compensação de *jitter*. Redes de dados baseadas no protocolo IP não garantem a entrega de pacotes. Se a rede estiver sobrecarregada, as filas dos roteadores vão crescer, e sempre que um pacote encontrar uma fila cheia, ele será descartado. Para aplicações VoIP, não cabe a retransmissão de pacotes, devido à sua característica de aplicação de tempo real. Além disso, mesmo que não tenham sido descartados durante o percurso na rede, os pacotes de voz que chegam a seu destino muito atrasados são considerados perdidos e descartados pelo *buffer* de *jitter*.

4.2.2.5 MOS

A pontuação MOS tem o objetivo de descrever quantitativamente a qualidade de uma reprodução de voz, levando em consideração vários fatores, dentre eles perda de pacotes, atraso, *codec* usado, *jitter*, etc. Portanto, a pontuação MOS, apresentada no nosso trabalho junto às métricas de perda e atraso, dará uma idéia do quanto essas métricas influenciam na qualidade da chamada como um todo. Possibilita também saber se a média da qualidade das chamadas realizadas pode ser considerada aceitável, tendo em vista que a pontuação MOS mínima para isso é de 3,6 [6].

O MOS foi calculado através do Modelo E, que calcula a qualidade da fala através do fator R. O MOS é uma pontuação específica para avaliar a qualidade da fala e varia de 1 (qualidade muito pobre) a 5 (qualidade excelente). O fator R é obtido com o

seguinte cálculo [6]:

$$R = R_o - I_s - I_d - I_{e,eff} + A$$

onde:

- R_o representa os efeitos da razão sinal-ruído;
- I_s representa perdas simultâneas ao sinal de voz;
- I_d representa perdas relacionadas ao atraso fim-a-fim;
- $I_{e,eff}$ representa perdas relacionadas à rede, ao *codec*, ou ao equipamento;
- A representa o fator de vantagem (e depende da tecnologia de comunicação usada).

O fator R é uma grandeza que varia de 0 a 100. De acordo com [6], o fator R pode ser convertido para a escala de pontuação MOS através da seguinte expressão:

$$\begin{aligned} \text{Para } R < 6,5 &: MOS = 1 \\ \text{Para } 6,5 \leq R \leq 100 &: MOS = 1 + 0,035R + 7.10^{-6}R(R-60)(100-R) \\ \text{Para } R > 100 &: MOS = 4,5 \end{aligned}$$

Normalmente, o fator R é descrito em categorias de valores, tal como pode ser visto na tabela 4.3. Sistemas cuja qualidade da fala seja avaliada em $R \leq 60$ não são recomendados, sendo desejável obter $R \geq 70$ [6].

Tabela 4.3: Categorias de qualidade da fala para o fator R. Fonte: [6].

Fator R	**MOS**	**Satisfação do usuário**
$90 \leq R < 100$	4,34 - 4,50	Muito satisfeitos
$80 \leq R < 90$	4,03 - 4,34	Satisfeitos
$70 \leq R < 80$	3,60 - 4,03	Alguns insatisfeitos
$60 \leq R < 70$	3,10 - 3,60	Muitos insatisfeitos
$0 \leq R < 60$	1,00 - 3,10	Quase todos insatisfeitos

Para os experimentos realizados neste trabalho, consideramos como pontuação mínima para uma chamada VoIP de qualidade aceitável aquelas que apresentaram MOS acima de 3,6.

Capítulo 5

Análise dos Resultados

Neste capítulo são mostrados os resultados da avaliação dos algoritmos de compressão de cabeçalhos, obtidos através das métricas e dos cenários propostos no capítulo anterior. A seção 5.1 mostra a análise dos resultados obtidos para o ganho de compressão dos algoritmos. A seção 5.2 mostra os resultados para a eficiência de largura de banda obtida em nossos experimentos. Na seção 5.3 são mostrados os resultados para a perda de pacotes na rede. A seção 5.4 mostra os valores obtidos para o atraso na rede. Por fim, a seção 5.5 analisa os valores para a pontuação MOS obtida.

5.1 Ganho de compressão

A medida de ganho de compressão é dada pela razão entre a quantidade de bytes que foram retirados dos cabeçalhos IP/UDP/RTP pelo processo de compressão em uma comunicação, e o total de bytes originalmente ocupados pelos cabeçalhos IP/UDP/RTP, como descrito em 4.2.2.1. Essa medida indica o quão eficiente é o mecanismo de compressão com relação a sua capacidade de diminuir o tamanho dos cabeçalhos; quanto maior o ganho de compressão de um algoritmo, maior sua capacidade de "compactar" os cabeçalhos. Como descrito anteriormente, a medida de ganho de compressão é feita no compressor, no lado da fonte do fluxo de voz. Portanto, seus valores não são afetados pelas características da rede, ou do meio de propagação utilizado.

Os valores apresentados aqui foram obtidos através da média aritmética dos valores

de ganho de compressão obtidos em todas as simulações realizadas no NS-2, calculados com base nas estatísticas oferecidas pelo módulo de compressão de cabeçalhos desenvolvido para a realização dos experimentos.

Os valores utilizados para calcular a média de ganho de compressão foram medidos com a utilização do Akaroa-2, utilizando precisão relativa máxima de 5% e 95% de confiança. A diferença entre os valores obtidos das diversas simulações foi considerado insignificante, já que todos os intervalos de confiança obtidos apresentaram sobreposição entre si. Isso garante que o cálculo da média aritmética sobre esse valores não invalida a confiança obtida através das simulações com o Akaroa-2.

Tabela 5.1: Ganho de compressão dos algoritmos.

Algoritmo	**Ganho de compressão**
RoHC	0,8645
Compressão Estática	0,6384
Compressão + Agregação	0,8274

A tabela 5.1 mostra os valores obtidos para o ganho de compressão dos algoritmos avaliados. O valor de ganho de compressão varia entre 0 e 1, onde um valor próximo de 1 significa que o algoritmo de compressão foi capaz de poupar quase todos os cabeçalhos de serem enviados à rede.

Com base nisso, e nos números mostrados na tabela 5.1, podemos dizer que o algoritmo RoHC foi o que apresentou o melhor ganho de compressão. O alto ganho de compressão apresentado pelo algoritmo RoHC se deve ao fato de seu processo de compressão eliminar dos cabeçalhos as informações estáticas e dinâmicas, mantendo apenas informações de identificação de contexto e as informações dinâmicas quando há mudança em seus valores.

O algoritmo RoHC é capaz de diminuir o tamanho dos cabeçalhos para até 2 bytes, o que poderia oferecer um ganho de compressão maior ainda do que o apresentado. Porém, pelo fato de eliminar as informações dinâmicas dos cabeçalhos, periodicamente o algoritmo RoHC U-mode precisa enviar mensagens de atualização de contexto, com o objetivo de recuperar o descompressor de uma possível perda de sincronização. Essas

mensagens de atualização possuem tamanho de cabeçalhos bem maior que os cabeçalhos comprimidos, chegando quase ao tamanho dos cabeçalhos originais.

A periodicidade com que mensagens de atualização são enviadas é um ponto de conflito de algoritmos de compressão de cabeçalhos que precisam de atualização de contexto. Quanto mais curto o período em que essas mensagens são enviadas, menor a possibilidade de o contexto de descompressor ficar desatualizado, porém, menor o ganho de compressão oferecido. Portanto, o envio dessas mensagens e a periodicidade com que são enviadas influenciam diretamente no ganho de compressão do RoHC. Em nossos experimentos, foram enviadas mensagens com cabeçalhos de atualização a cada 10 pacotes de cabeçalhos comprimidos, de acordo com o trabalho apresentado em [11].

O algoritmo de compressão estática mostrou o menor ganho de compressão. A compressão estática elimina dos cabeçalhos IP/UDP/RTP apenas as informações classificadas como estáticas e inferíveis, mantendo nos cabeçalhos as informações dinâmicas. Portanto, como esperado, o algoritmo de compressão estática não oferece um ganho de compressão tão alto quanto um algoritmo que comprime também os campos dinâmicos. O impacto dessa diferença de ganho de compressão no comportamento do tráfego de voz será avaliado com a análise das métricas de perda de pacotes, atraso e MOS.

A abordagem de compressão estática e agregação de pacotes mostrou um ganho de compressão quase tão alto quanto do algoritmo RoHC. Isso significa que o processo de agregação cumpriu a tarefa de aumentar o ganho de compressão do algoritmo de compressão estática. Apesar de o ganho de compressão obtido não ultrapassar aquele apresentado pelo RoHC, o valor apresentado se aproxima, e isso significa que o algoritmo de compressão estática e agregação conseguiu gerar cabeçalhos de pacotes quase tão pequenos quanto os cabeçalhos comprimidos pelo RoHC.

A compressão estática apresentou um ganho de compressão de 0,6384. O mecanismo de agregação de pacotes ofereceu um ganho de compressão extra devido à eliminação das informações dinâmicas redundantes dos cabeçalhos dos pacotes agregados. Neste caso, podemos afirmar que o ganho de compressão dessa abordagem é influenciado também pelo grau de agregação utilizado, que para nossos experimentos foi de dois

pacotes por pacote de agregação.

O grau de agregação utilizado é um ponto de conflito para a agregação de pacotes nesse contexto, pois quanto maior for o grau de agregação, maior será o ganho de compressão extra oferecido, porém maior também será o atraso imposto ao pacotes agregados. Na seção 5.4, que analisa os valores obtidos para o atraso na rede dos pacotes de voz em nossos experimentos, é possível visualizar o impacto do mecanismo de agregação de pacotes no atraso. A eliminação das informações redudantes entre os cabeçalhos dos pacotes agregados ofereceu um ganho de compressão extra ao mecanismo de compressão de cabeçalhos estática de 0,1890, o que significa uma diminuição extra de quase 20% do tamanho original dos cabeçalhos IP/UDP/RTP.

5.2 Eficiência de largura de banda

A medida de eficiência de largura de banda, como descrito em 4.2.2.2, é a métrica que indica o quanto de largura de banda foi utilizado para a transmissão de carga útil, quantificando assim a contribuição de cada algoritmo de compressão de cabeçalhos para um uso mais otimizado da largura de banda disponível. É obtido através da razão entre o total de bytes de carga útil transmitidos e o total de bytes efetivamente utilizados para a transmissão, incluindo *payload* e cabeçalhos. É uma grandeza que varia de 0 a 1, sendo que um valor de eficiência de largura de banda próximo de 1 significa que a contribuição do algoritmo de compressão foi quase total, e que a maior parte da largura de banda utilizada foi destinada à transmissão de carga útil.

Essa métrica, assim como a métrica de ganho de compressão, é calculada a partir de informações obtidas na fonte da comunicação, ou seja, no compressor, não sofrendo assim influência da rede ou do meio de transmissão utilizado. Os valores mostrados aqui foram calculados com base em estatísticas oferecidas pelo módulo de compressão de cabeçalhos implementado e incorporado ao NS-2 para a realização dos experimentos.

Os valores apresentados representam a média aritmética calculada a partir das estimativas obtidas de todas as simulações executadas pelo NS-2 com a utilização do Akaroa-2, exigindo 95% de confiança e precisão relativa máxima de 5%. O cálculo

da média aritmética sobre as estimativas não invalida a confiança obtida através da simulação com o Akaroa-2, devido ao fato de a diferença entre as estimativas ter sido considerada insignificante.

É importante citar que o valor de eficiência de largura de banda mostrado aqui leva em consideração apenas os cabeçalhos IP/UDP/RTP, não contabilizando os bytes utilizados pelos cabeçalhos de camada de enlace. A tabela 5.2 mostra os valores de eficiência de largura de banda obtidos para os algoritmos avaliados, e para as chamadas realizadas sem compressão de cabeçalhos.

Tabela 5.2: Eficiência de largura de banda dos algoritmos.

Algoritmo	**Eficiência de largura de banda**
Sem compressão	0,3333
RoHC	0,7868
Compressão Estática	0,5803
Compressão + Agregação	0,7435

A eficiência de largura de banda para chamadas sem compressão de cabeçalhos é apresentada para auxiliar na análise comparativa dos valores obtidos pelos algoritmos de compressão de cabeçalhos. O valor de 0,3333 de eficiência de largura de banda significa que apenas um terço da largura de banda foi utilizado para transmissão de carga útil dos pacotes.

Como mostrado na tabela, o algoritmo RoHC apresentou o maior valor de eficiência de largura de banda, com mais de 0,4 de diferença para a eficiência de largura de banda obtido das comunicações sem compressão de cabeçalhos. O alto valor de eficiência de largura de banda apresentado pelo RoHC se justifica pelo seu alto grau de compressão, como apresentado na seção anterior. Se um algoritmo oferece um alto grau de compressão, significa que seu mecanismo diminuiu consideravelmente o tamanho dos cabeçalhos comprimidos. Com a diminuição do tamanho dos cabeçalhos dos pacotes, aumenta a utilização da banda para a transmissão de carga útil, aumentando assim a eficiência de largura de banda.

O algoritmo de compressão estática mostrou o menor valor de eficiência de largura

de banda. A isso se deve seu baixo ganho de compressão, com relação aos outros algoritmos. Por não comprimir os campos de cabeçalhos dinâmicos, o algoritmo de compressão estática não é capaz de diminuir o tamanho dos cabeçalhos na mesma proporção em que o algoritmo RoHC.

No entanto, ainda assim o algoritmo de compressão estática apresenta melhorias com relação ao valor de eficiência de largura de banda de chamadas sem compressão de cabeçalhos. Um valor de eficiência de largura de banda de 0,5803 indica que o algoritmo de compressão estática foi capaz de diminuir o tamanho dos cabeçalhos para menos da metade do tamanho total do pacote. A partir de então, podemos dizer que a maior parte da banda foi utilizada para transmissão de carga útil dos pacotes.

O valor de eficiência de largura de banda apresentado pela abordagem de compressão de cabeçalhos e agregação de pacotes foi quase tão alto quanto o valor apresentado pelo algoritmo RoHC. Novamente isso se justifica pelo ganho de compressão obtido por essa abordagem, em comparação ao ganho de compressão do algoritmo RoHC. Como o ganho apresentado pela abordagem de compressão e agregação foi alto, a eficiência de banda obtida segue o mesmo comportamento, pois cabeçalhos de pacotes menores indicam que uma parte significativa da largura de banda deixou de ser dedicada à transmissão de informações de controle e passou a ser dedicada à transmissão de informações de dados.

5.3 Perda de pacotes

A perda de pacotes em redes sem fio pode ocorrer em dois pontos principais: nos roteadores ou nos enlaces da rede. A perda nos roteadores se dá devido à sobrecarga de pacotes nesses elementos, o que ocasiona a ocupação total de seus *buffers*. Quando isso ocorre, todos os pacotes recebidos são descartados até que os *buffers* liberem espaço para armazenar novos pacotes. Já os enlaces oferecem uma perda relacionada ao meio de transmissão utilizado, e os enlaces sem fio, por sua vez, são conhecidos por apresentarem altas taxas de perdas de pacotes.

Para aplicações VoIP, existe também o descarte de pacotes atrasados no *buffer* de

compensação de *jitter*, e para sistemas com compressão de cabeçalhos, pode haver também o descarte de pacotes por dessincronização de contexto, e pelo problema da propagação da perda, como descrito em 2.3.2.

Os valores apresentados aqui são estimativas obtidas através do Akaroa-2, com 95% de confiança e precisão relativa de 5%. Seus valores representam a porcentagem de pacotes perdidos em uma chamada de voz de 60 segundos, na presença de chamadas simultâneas, realizadas sobre os cenários apresentados.

O cenário linear, como mostrado em 4.2.1, é composto por 6 nós roteadores posicionados em uma linha reta, onde cada nó só pode se comunicar diretamente com seus nós vizinhos imediatos. Para avaliar o impacto da quantidade de saltos na perda de pacotes, foram realizados experimentos com 2, 3, 4 e 5 saltos sobre o cenário citado.

As Figuras 5.1, 5.2, 5.3 e 5.4 mostram os gráficos da perda de pacotes para os experimentos realizados com 2, 3, 4 e 5 saltos, respectivamente, no cenário linear. Os intervalos de quantidade de chamadas simultâneas foram escolhidos com o objetivo de apresentar o aumento da porcentagem de perda de pacotes desde uma porcentagem próxima de 0% até uma porcentagem próxima de 100%.

Como esperado, os valores de perda de pacotes para as chamadas realizadas sem compressão de cabeçalhos (None) foram os mais altos, na maioria dos casos. Essa diferença com relação às chamadas realizadas com compressão de cabeçalhos mostra que o uso de um mecanismo de compressão pode de fato influenciar positivamente no comportamento do tráfego do fluxo de voz em redes de múltiplos saltos sem fio. Isso confirma também o impacto negativo causado pelo *overhead* dos cabeçalhos IP/UDP/RTP originais imposto sobre o tráfego de voz.

O algoritmo de compressão estática (SHC) mostrou os valores de perda de pacotes mais altos, em comparação aos outros dois algoritmos de compressão (RoHC e SHC+AG). Podemos justificar este comportamento pelo ganho de compressão obtido por este algoritmo, que não foi tão alto quanto o ganho de compressão do algoritmo RoHC e da abordagem de compressão e agregação. Os algoritmos de compressão de cabeçalhos diminuem a porcentagem de informações de controle nos cabeçalhos, e con-

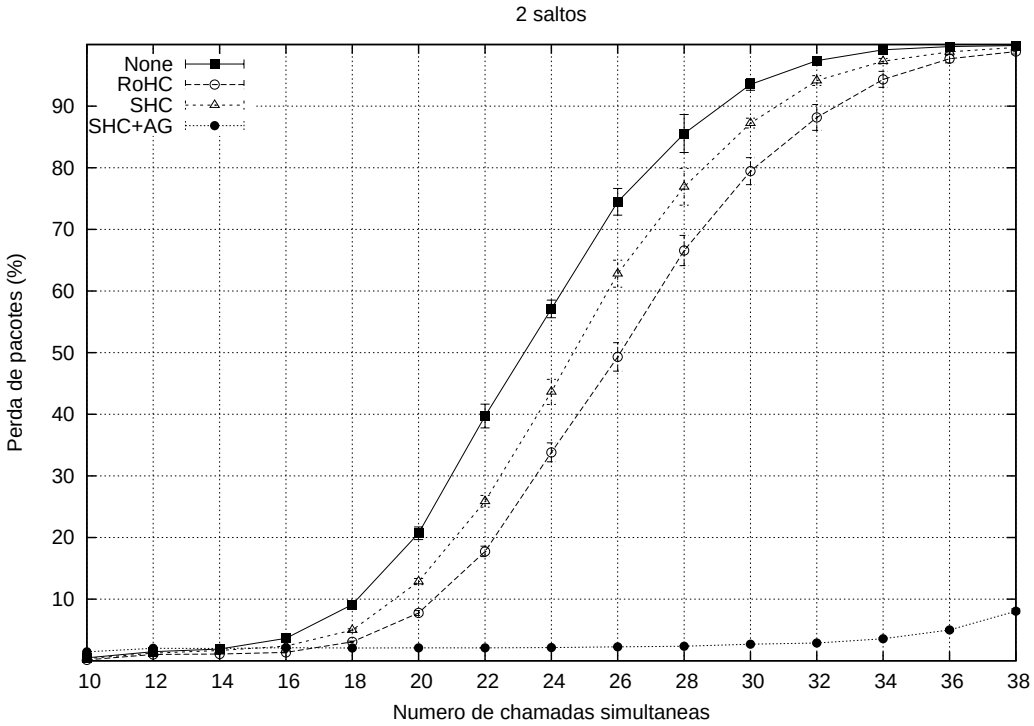

Figura 5.1: Perda de pacotes para chamadas de 2 saltos no cenário linear.

seqüentemente diminuem também o tamanho total dos pacotes. Essa diferença nos tamanhos dos pacotes é significativa para a perda de pacotes, como pode ser visualizada nos resultados dos nossos experimentos.

Por esse motivo, ainda assim os valores de perda do algoritmo SHC são mais baixos que os valores de perda apresentados pelas chamadas sem compressão, apresentando uma diferença de mais de 10% para 22, 24 e 26 chamadas simultâneas com 2 saltos sem fio, e de aproximadamente 20% para 8 chamadas com 5 saltos. O melhor desempenho do algoritmo SHC em comparação às chamadas sem compressão se deve à eficiência de largura de banda obtida por esse algoritmo, que identificou uma utilização da largura de banda de mais de 50% para carga útil dos pacotes.

O algoritmo RoHC mostrou valores de perda melhores que os valores de perda apresentados pelo algoritmo SHC, chegando a uma diferença de mais de 10% para 18 chamadas simultâneas com 3 saltos. A perda apresentada pelo RoHC poderia ter sido menor ainda, pois certamente uma porcentagem dessa perda se deve à dessincronização de contexto e o problema da propagação da perda, características de algoritmos de

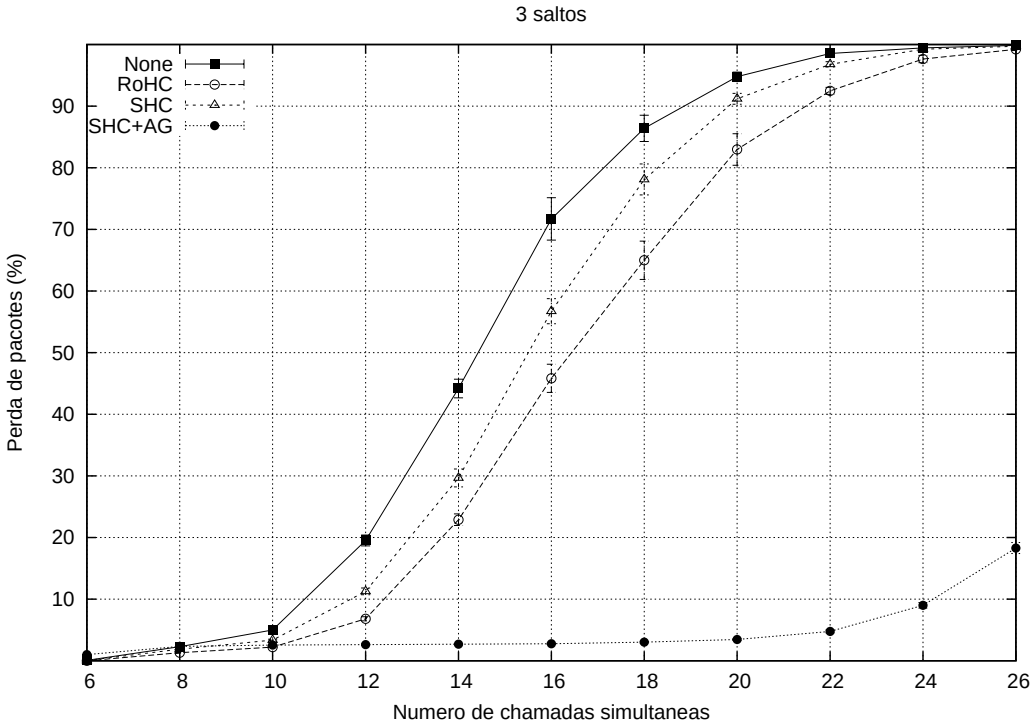

Figura 5.2: Perda de pacotes para chamadas de 3 saltos no cenário linear.

compressão que necessitam de atualização de contexto. Apesar desse problema, o alto ganho de compressão oferecido pelo RoHC fez com que a perda de pacotes apresentada se mostrasse mais baixa que a perda das chamadas sem compressão e das chamadas realizadas com o algoritmo SHC.

A abordagem de compressão de cabeçalhos e agregação de pacotes (SHC+AG) mostrou os menores valores para perda de pacotes. Os valores apresentados ficaram bem abaixo da perda apresentada pelos outros algoritmos, ficando inclusive abaixo dos 10% para a maioria dos experimentos realizados, e atingindo o máximo de 29% para 14 chamadas simultâneas com 5 saltos sem fio.

Apesar de a agregação aumentar o tamanho dos pacotes, o que poderia aumentar também a perda, esse procedimento também diminui a quantidade de pacotes enviados à rede, de uma forma proporcional ao grau de agregação utilizado. A agregação de pacotes de dois em dois, como utilizado em nossos experimentos, resulta na criação de pacotes um pouco maiores mas não o suficiente para impactar negativamente na perda de pacotes.

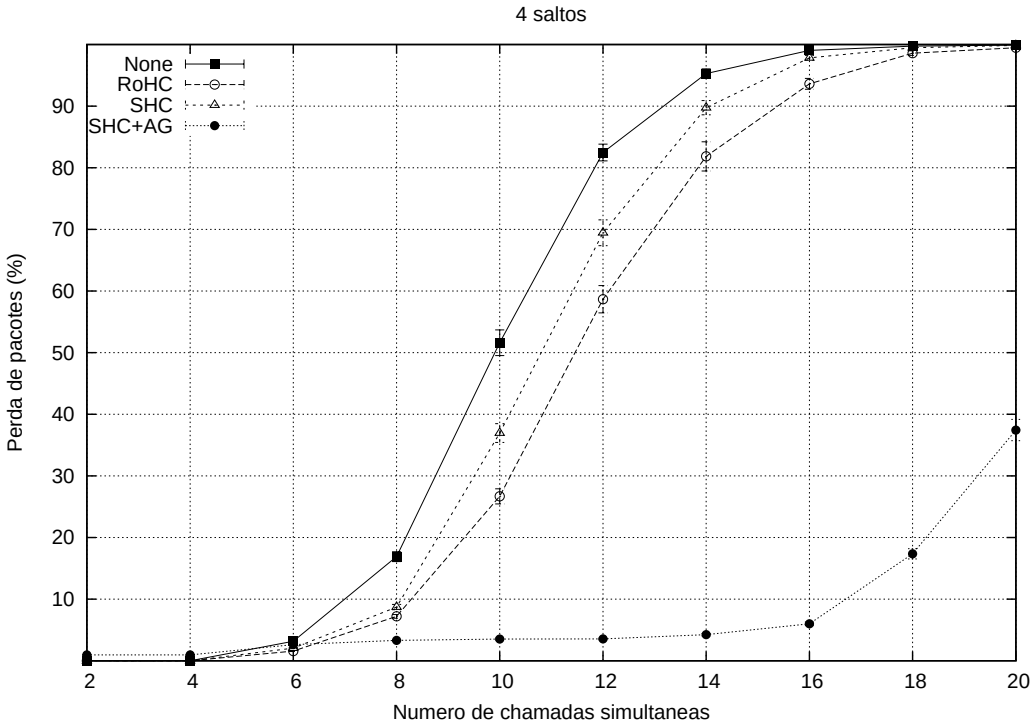

Figura 5.3: Perda de pacotes para chamadas de 4 saltos no cenário linear.

Além disso, a agregação diminui consideravelmente a quantidade de pacotes enviados à rede, diminuindo o trabalho da camada de enlace executado por pacote enviado, além de proporcionar a diminuição da quantidade de bytes enviados à rede, proposta chave do processo de compressão de cabeçalhos. Portanto, podemos dizer que além do alto ganho de compressão oferecido pela abordagem SHC+AG, o impacto positivo na perda de pacotes se deu também pelo processo de agregação por si só, que foi o principal responsável por manter o nível de perda de pacotes bem abaixo dos apresentados pelos outros algoritmos.

Com o aumento do número de saltos sem fio, notamos que o comportamento dos algoritmos se mantém o mesmo. No entanto, é possível notar que o número de chamadas simultâneas suportadas diminui. Para 10 chamadas simultâneas, por exemplo, a perda no gráfico de 2 saltos é de menos de 5%; já para o cenário de 5 saltos sem fio, a perda para 10 chamadas simultâneas fica acima de 75%. É possível notar também que o sistema fica menos tolerante ao acréscimo de chamadas simultâneas, ou seja, a perda de pacotes cresce mais rapidamente, quanto maior for o número de saltos sem fio.

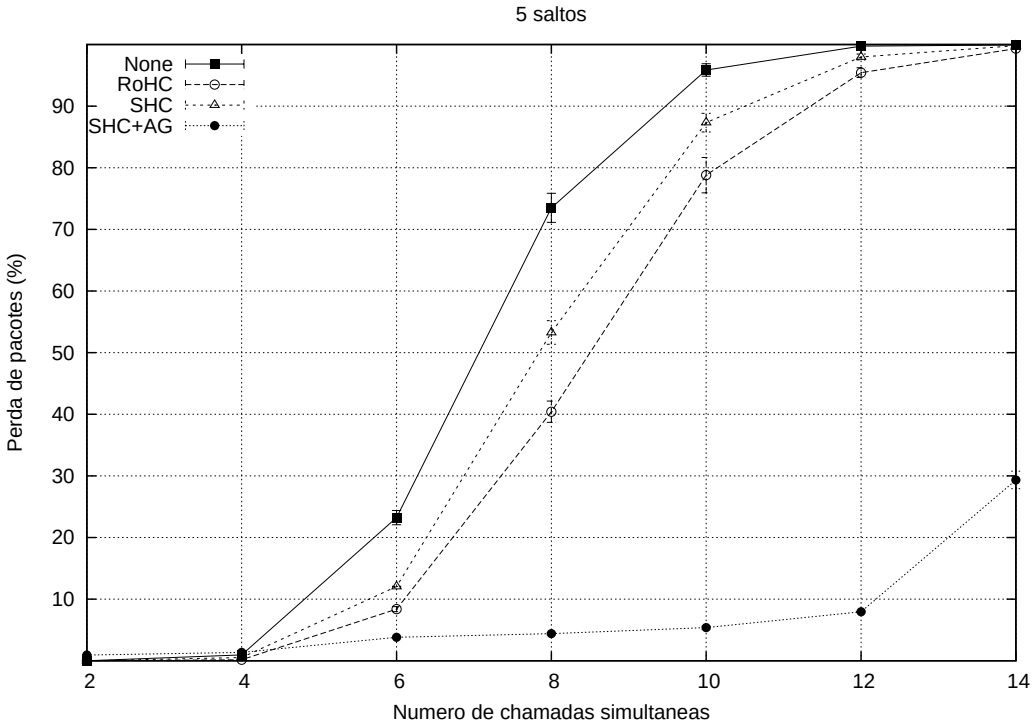

Figura 5.4: Perda de pacotes para chamadas de 5 saltos no cenário linear.

Para a abordagem SHC+AG, no entanto, esse efeito foi bem mais sutil. A perda de pacotes se manteve crescendo lentamente com o aumento da quantidade de chamadas simultâneas, mantendo-se em menos de 30% de perda em todos os casos, inclusive nos quais as outras configurações de compressão apresentaram perda de quase 100%. A essa diferença de comportamento, atribuimos novamente o fator de economia no trabalho realizado pela camada de enlace para transmitir os pacotes e à diminuição da quantidade de pacotes enviados à rede, graças à agregação.

É interessante notar também que os valores da perda de pacotes obtidos pela abordagem SHC+AG se mostraram um pouco maior que os demais para os experimentos com poucas chamadas simultâneas. A isso podemos atribuir o tamanho um pouco maior dos pacotes agregados, que apesar de termos concluído não ser prejudicial na maioria dos casos, para cenários com pouca carga podem apresentar uma perda de pacotes maior até que a perda apresentada por chamadas sem compressão de cabeçalhos. No entanto, podemos observar que ainda assim a perda não apresenta índices significativos a ponto de contribuir para a degradação da qualidade das chamadas de

voz.

A perda de pacotes, quando ocorre com uma certa freqüência e/ou acima de certo limite, afeta diretamente a qualidade da voz reproduzida no receptor, provocando pausas desconfortáveis para os interlocutores. A referência [6] define esse limite em 5% para toda a comunicação. Quando a taxa de perda de pacotes ultrapassa esse limite, problemas de inteligibilidade podem ocorrer, prejudicando a interatividade da conversa. A tabela 5.3 mostra a quantidade de chamadas simultâneas realizadas com média máxima de 5% de perda de pacotes por chamada, nos experimentos realizados no cenário linear.

Tabela 5.3: Quantidade de chamadas simultâneas realizadas com até 5% de perda de pacotes.

Algoritmo	**2 saltos**	**3 saltos**	**4 saltos**	**5 saltos**
Sem compressão	16 chamadas	10 chamadas	6 chamadas	4 chamadas
SHC	18 chamadas	10 chamadas	6 chamadas	4 chamadas
RoHC	18 chamadas	10 chamadas	6 chamadas	4 chamadas
SHC + Agregação	36 chamadas	22 chamadas	14 chamadas	8 chamadas

Além dos experimentos realizados sobre o cenário linear, foram realizados também experimentos utilizando o cenário em árvore. O cenário em árvore, descrito em 4.2.1, foi formado por 7 nós, ficando 4 deles à distância de 2 saltos da raiz, considerada um *gateway* da rede, e para onde todas as chamadas realizadas foram direcionadas.

Foram realizados dois experimentos sobre o cenário em árvore, um primeiro com chamadas simultâneas realizadas a partir de todos os nós em direção ao nó *gateway* (nó 0), formando assim 6 pares fonte-destino, e um segundo experimento com chamadas simultâneas originadas apenas nos 4 nós folhas (nós 3, 4, 5 e 6) da árvore, formando 4 pares fonte-destino.

As Figuras 5.5 e 5.6 mostram os valores de perda de pacotes obtidos nos dois experimentos realizados sobre o cenário em árvore. A quantidade de chamadas simultâneas mostrada nos gráficos é referente à quantidade de chamadas para cada par fonte-destino. Foram realizados experimentos com a quantidade de chamadas simultâneas variando de 2 a 8, quantidade em que a maioria dos algoritmos mostrou perda de

pacotes acima de 85%.

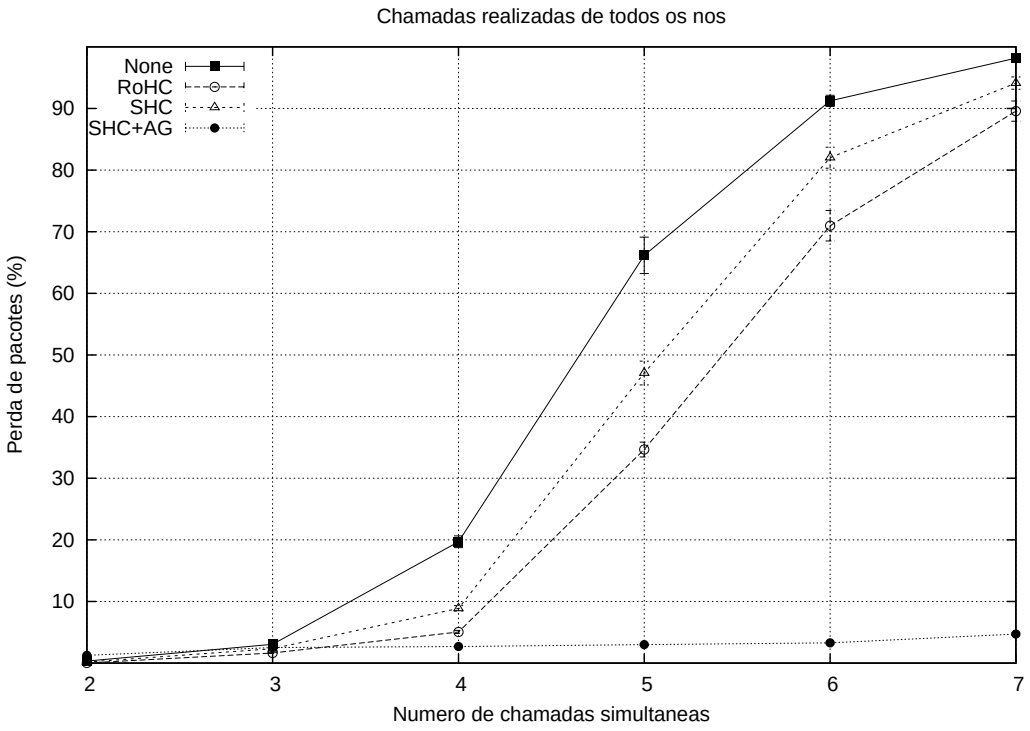

Figura 5.5: Perda de pacotes para chamadas realizadas sobre o cenário em árvore, a partir de todos os nós.

Podemos notar que no cenário em árvore o comportamento dos algoritmos se manteve idêntico ao comportamento mostrado nos experimentos sobre o cenário linear. As chamadas sem compressão de cabeçalhos apresentaram os maiores valores de perda de pacotes. A perda de pacotes para o algoritmo SHC se mostrou maior que para os demais algoritmos de compressão. O RoHC mostrou os melhores valores, depois da abordagem de compressão e agregação.

Novamente a abordagem SHC+AG mostrou valores de perda bem abaixo dos demais algoritmos. A justificativa para esse comportamento sobre o cenário em árvore é a mesma dada para o cenário linear. O funcionamento do algoritmo, principalmente relacionado ao mecanismo de agregação de pacotes é o principal responsável por esse comportamento na rede.

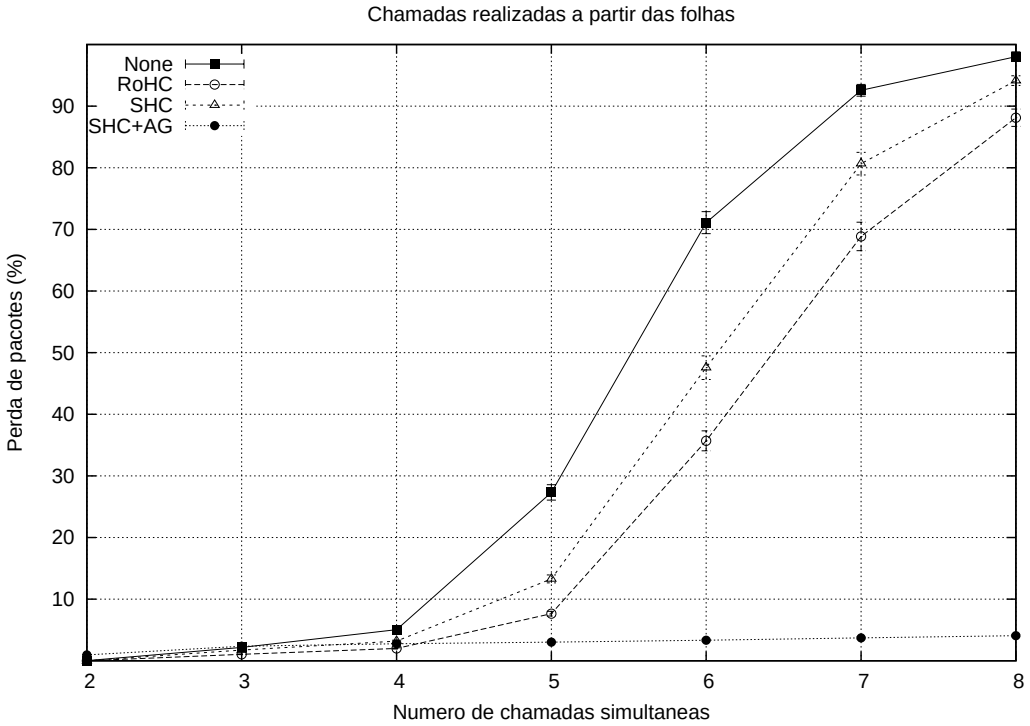

Figura 5.6: Perda de pacotes para chamadas realizadas sobre o cenário em árvore, a partir dos nós folha.

5.4 Atraso na rede

O atraso na rede envolve três tipos de atraso: atraso de enfileiramento, atraso de serialização e atraso de propagação [6]. Portanto, os principais elementos responsáveis pelos atrasos na rede são os roteadores, onde se formam as filas de transmissão, e os enlaces da rede, que vão determinar a velocidade com a qual os pacotes são transmitidos e propagados.

É uma métrica importante ao tratar-se de aplicações VoIP pois esse tipo de aplicação, caracterizada por ser de tempo real, possui restrições com relação ao tempo em que a voz leva para chegar de uma ponta da comunicação a outra; atrasos acima do aceitável podem comprometer a interatividade dos interlocutores e a qualidade da comunicação.

Como o objetivo da avaliação é analisar o comportamento do fluxo de voz na rede dadas as condições oferecidas pelos algoritmos de compressão de cabeçalhos, é importante citar que os valores mostrados aqui são para o atraso na rede, não levando em

consideração o atraso imposto pelo *codec* de voz (atraso de codificação e atraso de predição), que é constante durante toda uma comunicação utilizando o mesmo *codec*.

Os valores mostrados nesta seção são estimativas do atraso médio dos pacotes em chamadas simultâneas de 60 segundos realizadas sobre os cenários linear e em árvore.

O cenário linear é composto por 6 nós roteadores posicionados em uma linha reta, onde cada nó só pode se comunicar diretamente com seus nós vizinhos imediatos. Para avaliar o impacto da quantidade de saltos no atraso da rede, foram realizados experimentos com 2, 3, 4 e 5 saltos sem fio.

As Figuras 5.7, 5.8, 5.9 e 5.10 mostram os valores de atraso na rede para as chamadas realizadas com 2, 3, 4 e 5 saltos, respectivamente, no cenário linear. Os valores de atraso mostrados aqui foram mensurados a partir das mesmas chamadas utilizadas para medir a perda de pacotes na rede mostradas na seção anterior.

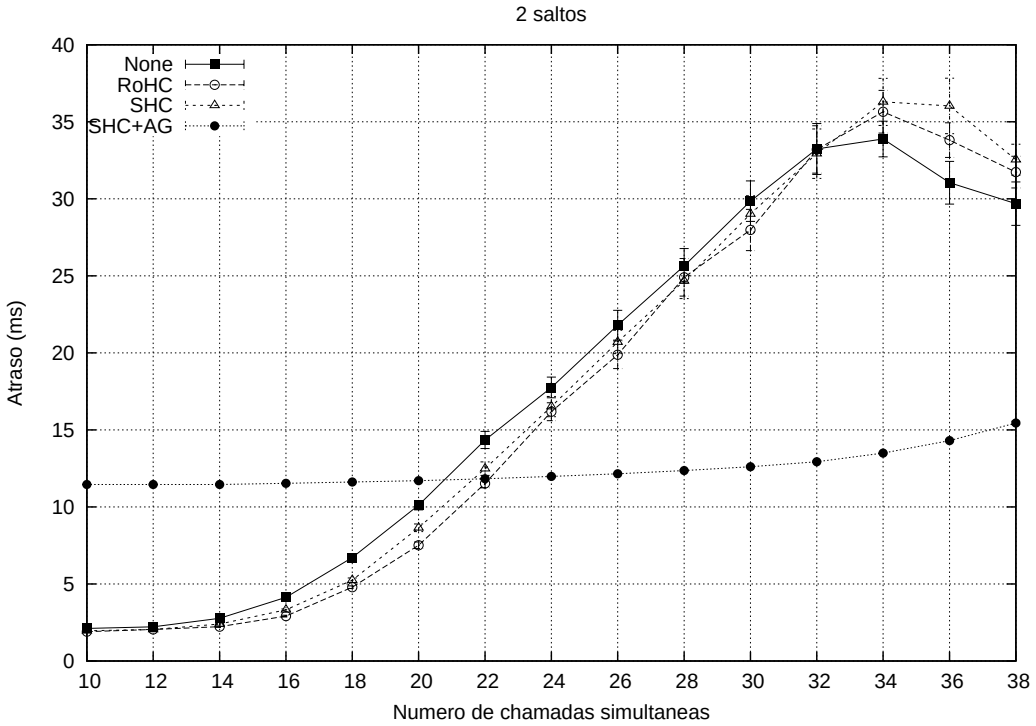

Figura 5.7: Atraso na rede para chamadas de 2 saltos no cenário linear.

Como esperado, os valores de atraso aumentaram conforme o número de chamadas simultâneas aumentou, independente da quantidade de saltos sem fio. Quanto maior a quantidade de chamadas simultâneas realizadas, maior será a carga de tráfego enviada

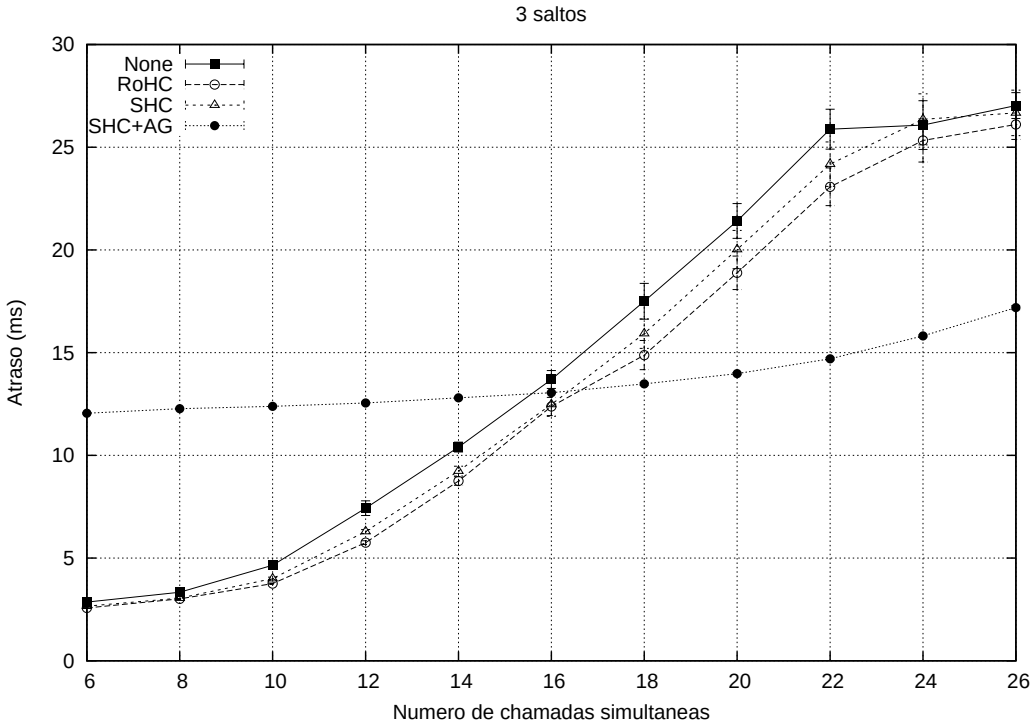

Figura 5.8: Atraso na rede para chamadas de 3 saltos no cenário linear.

à rede, aumentando assim a fila nos roteadores da rede e aumentado o atraso sofrido pelos pacotes ao percorrer o caminho da origem ao destino. As chamadas realizadas sem compressão de cabeçalhos (None) mostraram os maiores valores de atraso, se não levarmos em consideração a abordagem de compressão de cabeçalhos e agregação de pacotes (SHC+AG).

As chamadas realizadas sem compressão de cabeçalhos mostraram um atraso sutilmente maior do que os algoritmos RoHC e de compressão estática (SHC) em alguns casos, chegando a no máximo 5 ms de diferença no caso de 10 chamadas simultâneas com 5 saltos sem fio. Nos outros casos, a diferença apresentada pode ser considerada insignificante. Isso quer dizer que em alguns casos, a diferença do ganho de compressão oferecido pelos algoritmos de compressão RoHC e SHC foi grande o suficiente para gerar impacto no atraso sofrido pelos pacotes na rede, mas na maioria dos casos não.

Analisando os gráficos, podemos ver que a distância entre a curva das chamadas sem compressão e as curvas apresentadas para os algoritmos RoHC e SHC tende a aumentar conforme o número de saltos sem fio aumenta. Então, com base nesse comportamento

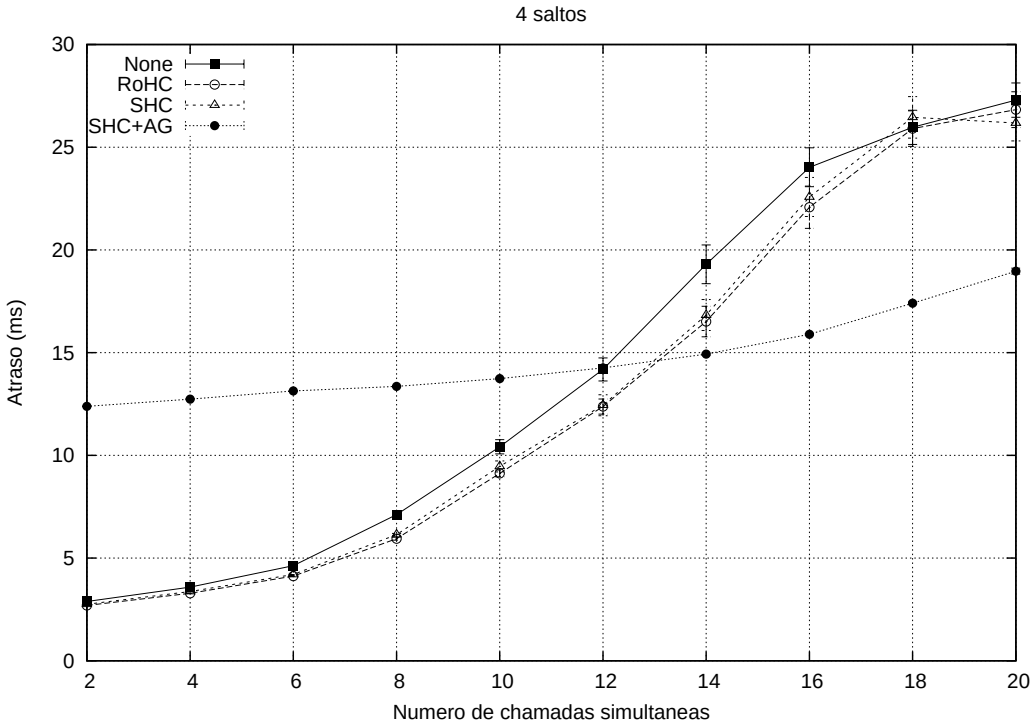

Figura 5.9: Atraso na rede para chamadas de 4 saltos no cenário linear.

é possível afirmar que, conforme o aumento do número de saltos, os algoritmos RoHC e SHC tendem a mostrar uma certa vantagem com relação à chamadas realizadas sem compressão.

Podemos notar através desses gráficos que o atraso volta a diminuir a partir de uma certa quantidade de chamadas simultâneas, independente da quantidade de saltos. Esse comportamento pode ser melhor observado no gráfico referente às chamadas realizadas com 2 saltos sem fio, onde a partir de 34 chamadas os valores do atraso diminuem, para as chamadas sem compressão e com os algoritmos RoHC e SHC. Esse comportamento pode ser justificado pela alta porcentagem de perda de pacotes desses algoritmos, nesse caso acima de 90%, que pode ser observada nos gráficos da seção anterior. Com a alta perda de pacotes, o atraso diminuiu porque os poucos pacotes que conseguiram atingir o destinatário não sofreram atraso tão alto na rede.

Os valores de atraso para as chamadas realizadas com o algoritmo RoHC e com o algoritmo SHC não mostraram diferença significativa entre si. Apesar de o RoHC ter apresentado ganho de compressão bem mais alto que o algoritmo SHC, o tamanho

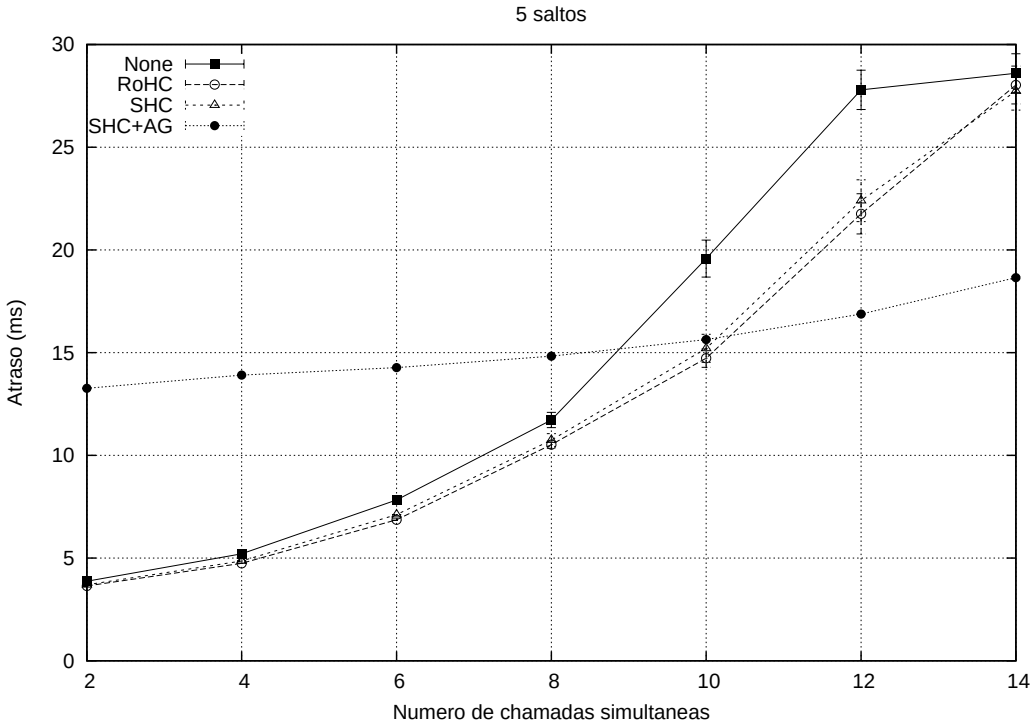

Figura 5.10: Atraso na rede para chamadas de 5 saltos no cenário linear.

menor dos pacotes não foi suficiente para mostrar uma diferença significativa no atraso nos cenários analisados. Com esses resultados, podemos dizer que a diferença de ganho de compressão e eficiência de banda apresentados pelos algoritmos RoHC e SHC não mostraram diferença significativa para o valor do atraso na rede.

O comportamento do atraso apresentado pela abordagem de compressão de cabeçalhos e agregação de pacotes (SHC+AG) é peculiar e diferente em comparação aos outros, pois apesar de mostrar um alto valor de atraso para poucas quantidades de chamadas simultâneas, esse valor varia bem pouco conforme o número de chamadas aumenta.

O atraso imposto pelo mecanismo de agregação de pacotes fez com que o atraso apresentado pela abordagem SHC+AG fosse mais alto para poucas chamadas simultâneas. No entanto, a vantagem da diminuição da quantidade de pacotes enviados à rede e conseqüentemente do *overhead* de camada de enlace, contribuiu para que o atraso na rede se mostrasse mais estável conforme o acréscimo de chamadas simultâneas. Essa estabilidade faz com que, a partir de um certo ponto, o atraso apresentado pela abor-

dagem SHC+AG deixe de ser o mais alto apresentado e passe a ser o mais baixo de todos.

O aumento da quantidade de saltos sem fio apresenta sobre o atraso o mesmo efeito apresentado sobre a perda de pacotes. A quantidade de chamadas simultâneas suportadas para um dado valor de atraso diminui conforme o aumento da quantidade de saltos. Por exemplo, para 10 chamadas simultâneas o atraso se mantém abaixo de 5 ms com 2 e 3 saltos sem fio, porém esse atraso aumenta para aproximadamente 10 ms com 4 saltos, e fica entre 15 e 20 ms para 5 saltos sem fio. O atraso aumenta dado o aumento de saltos sem fio na comunicação devido ao tempo que o pacote leva para ser processado em cada nó intermediário (tempo de enfileiramento, tempo de serialização e tempo de propagação); quanto maior for a quantidade de nós intermediários, maior será este atraso.

No geral, podemos afirmar que os algoritmos apresentaram o mesmo comportamento independente da quantidade de saltos sem fio na comunicação. Apesar de o atraso aumentar com o aumento do número de saltos e chamadas simultâneas, todos os valores apresentados em nossos experimentos não resultariam em um atraso fim-a-fim que ultrapassasse o máximo permitido para considerar um impacto negativo na qualidade da chamada, que seria um atraso de 150 ms [6].

Para os experimentos realizados sobre o cenário em árvore, as Figuras 5.11 e 5.12 mostram os resultados. O cenário em árvore é formado por 7 nós dispostos formando uma árvore de 3 níveis e 4 folhas, sendo a raiz da árvore o nó 0. Esses valores foram mensurados a partir das mesmas chamadas realizadas sobre o cenário em árvore utilizadas para medir a perda de pacotes. Portanto, foram realizadas chamadas com destino ao nó *gateway* (nó 0), em um primeiro momento a partir de todos os nós da rede, e em um segundo momento a partir dos nós folha.

Notamos nesses gráficos um comportamento idêntico ao mostrado nos gráficos de atraso sobre o cenário linear. O atraso aumenta com o aumento do número de chamadas simultâneas, e os valores de atraso para os algoritmos RoHC e SHC não apresentam uma diferença significativa entre si. Já as chamadas simultâneas realizadas sem compressão

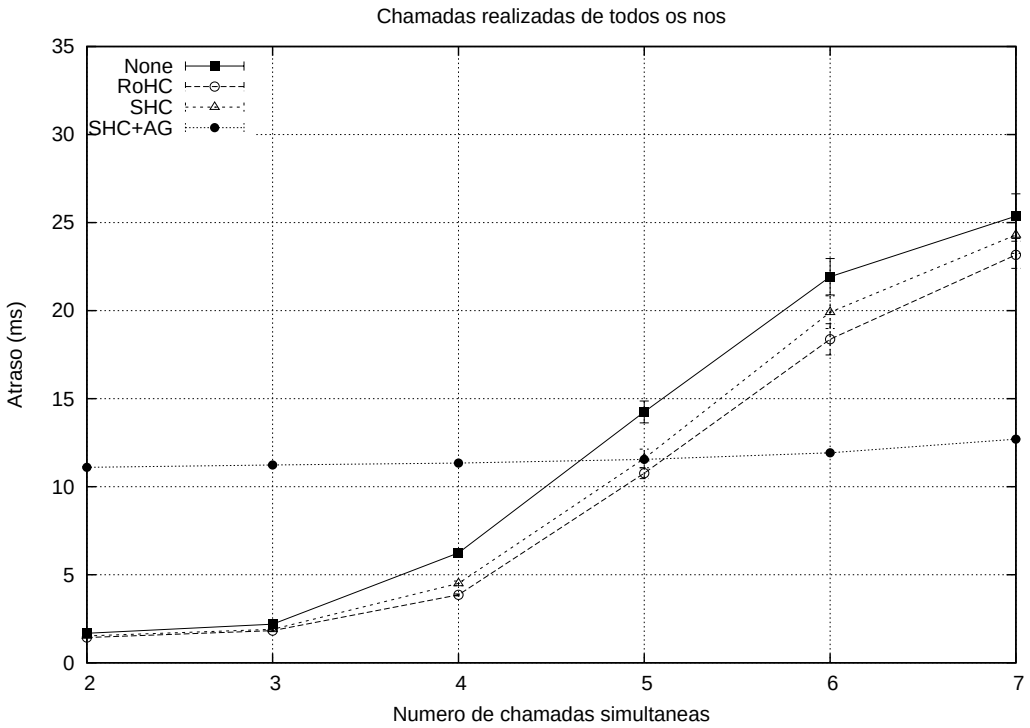

Figura 5.11: Atraso na rede para chamadas realizadas sobre o cenário em árvore, a partir de todos os nós.

apresentaram atraso um pouco acima dos demais algoritmos.

A abordagem SHC+AG apresentou o maior atraso para a maioria dos casos. No entanto, como no cenário linear, o atraso apresentado por essa abordagem é mais estável, aumentando pouco com o acréscimo de chamadas simultâneas. Para as demais configurações de compressão, o aumento no atraso é bem maior com o aumento do número de chamadas.

Novamente os valores de atraso apresentados não resultariam em um atraso fim-a-fim que pudesse ultrapassar os 150 ms permitidos para considerar uma chamada de qualidade aceitável.

5.5 *Mean Opinion Score* - MOS

A pontuação MOS tem o objetivo de quantificar a qualidade de uma reprodução de voz. Para VoIP, esta métrica representa a qualidade da chamada, e é obtida a partir do fator R, calculado através do Modelo E [6], que leva em consideração vários fatores

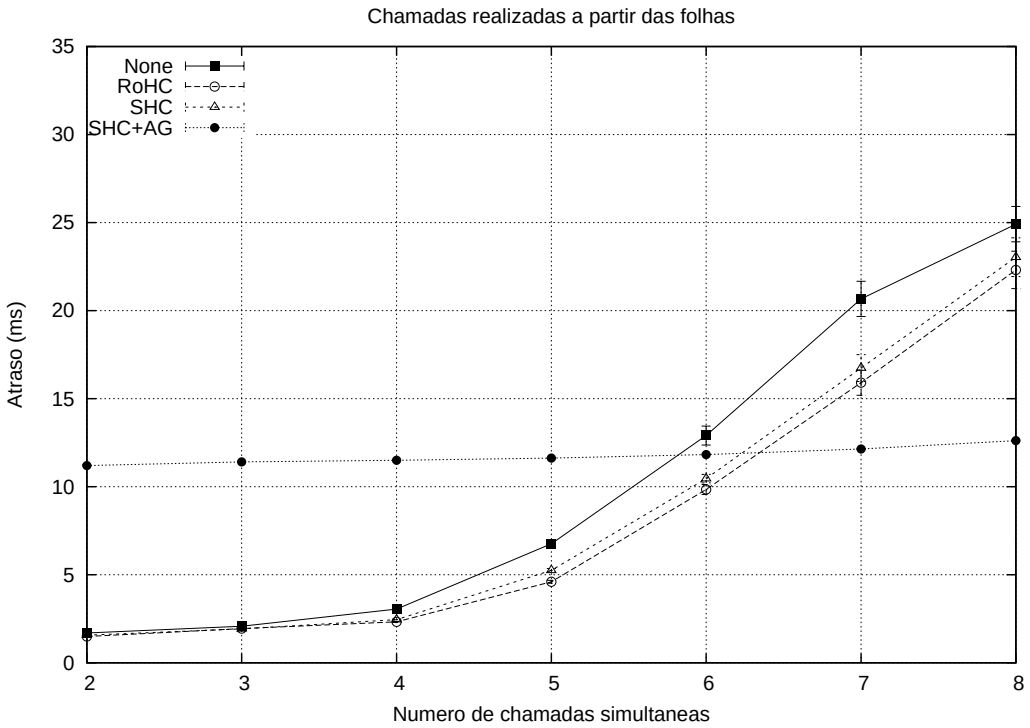

Figura 5.12: Atraso na rede para chamadas realizadas sobre o cenário em árvore, a partir dos nós folha.

relacionados ao fluxo de voz, desde os equipamentos utilizados pelos interlocutores até detalhes relacionados ao tráfego, como o *codec* utilizado, atrasos e perdas de pacotes na rede, além do *jitter*. A escala do MOS varia de 1 a 5, sendo 1 utilizado para as chamadas de qualidade mais pobre, e 5 usado para as chamadas de qualidade excelente.

Os valores mostrados aqui são estimativas obtidas através do Akaroa-2, com 95% de confiança e 5% de precisão relativa máxima exigida. Seus valores representam a média do MOS obtido para chamadas de voz bidirecionais de 60 segundos, realizadas na presença de chamadas simultâneas e sobre os cenários linear e em árvore.

O cenário linear é composto por 6 nós roteadores posicionados em uma linha reta, onde cada nó só pode se comunicar diretamente com seus nós vizinhos imediatos. Para avaliar o impacto da quantidade de saltos na pontuação MOS, foram realizados experimentos com 2, 3, 4 e 5 saltos sem fio sobre o cenário citado.

As Figuras 5.13, 5.14, 5.15 e 5.16 mostram os gráficos da pontuação MOS para os experimentos realizados com 2, 3, 4 e 5 saltos sem fio, respectivamente, no cenário

linear. Como é possível observar, as diferentes abordagens de compressão mostraram comportamento parecido nos 4 gráficos.

Como esperado, o valor do MOS diminui com o aumento do número de chamadas simultâneas, já que é esperado que a qualidade das chamadas caia com o aumento do número de fluxos de voz concorrentes. Os valores de MOS mostrados pelos algoritmos RoHC e de compressão estática (SHC) não mostraram diferença significativa entre si. Podemos justificar esse comportamento pelos valores obtidos para as métricas de atraso e perda de pacotes para essas chamadas, que não apresentaram uma diferença muito grande entre os dois algoritmos a ponto de refletir uma diferença significativa no MOS.

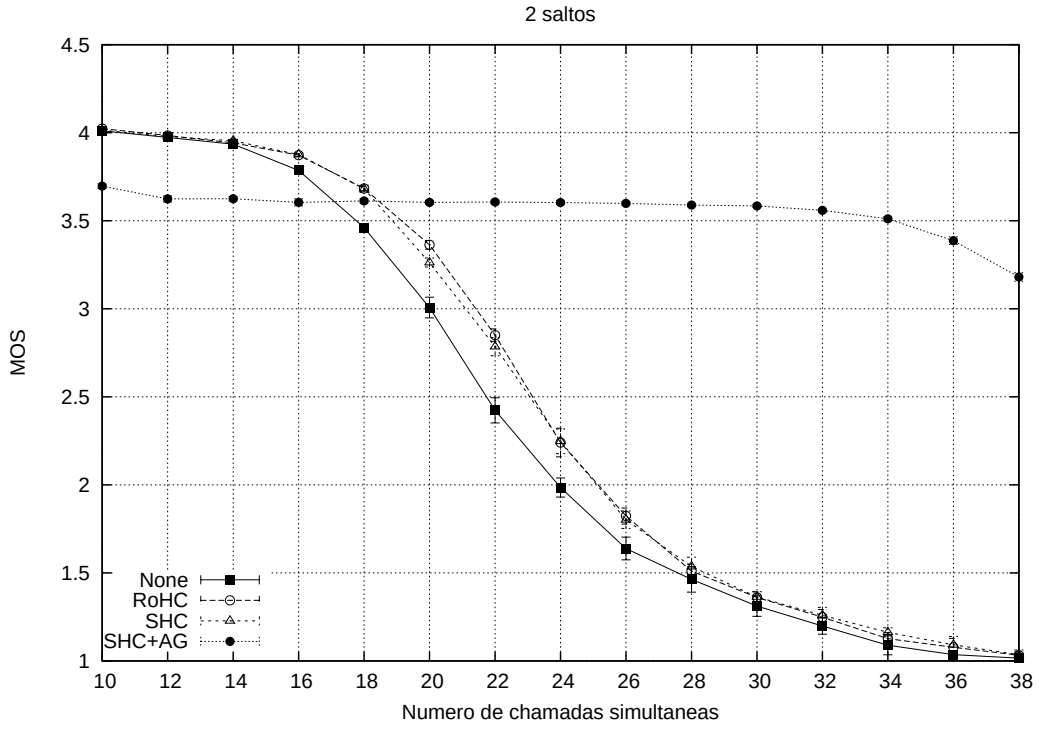

Figura 5.13: Pontuação MOS para chamadas de 2 saltos no cenário linear.

Já as chamadas realizadas sem compressão de cabeçalhos (None) mostraram valores de MOS um pouco mais baixos. Esse comportamento pode ser justificado mais uma vez pelo ganho de compressão oferecido pelos algoritmos RoHC e de compressão estática, que impactaram no atraso e na perda de pacotes, que apesar de não representarem um grande diferença entre si, representou uma diferença significativa com relação às chamadas sem compressão.

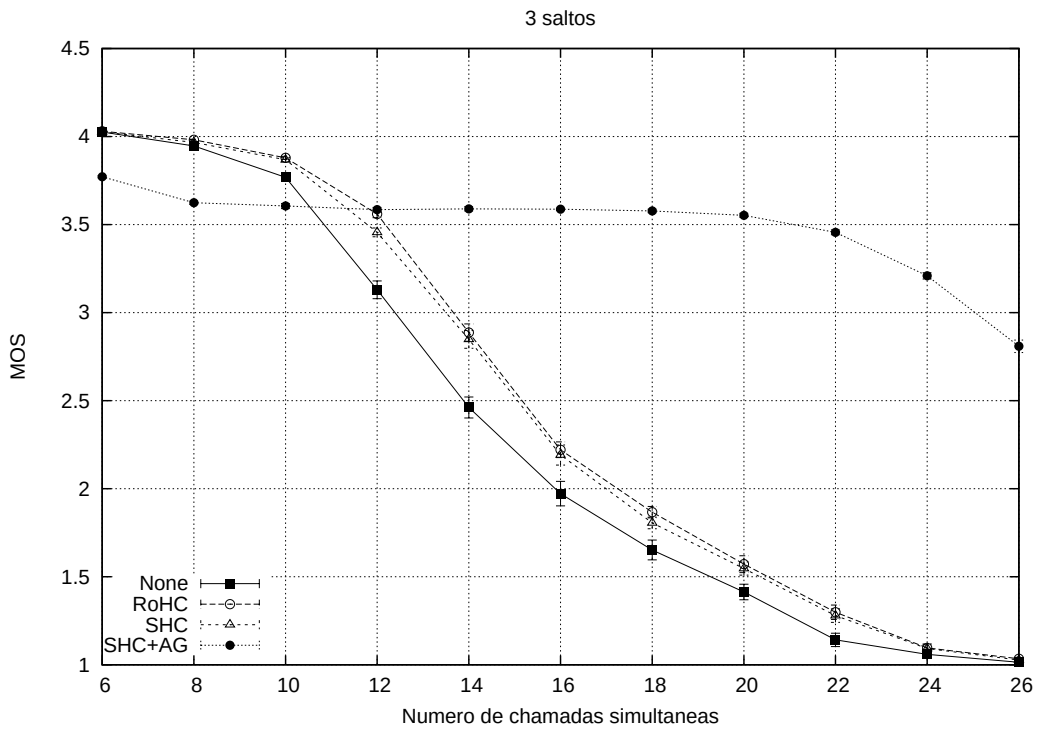

Figura 5.14: Pontuação MOS para chamadas de 3 saltos no cenário linear.

No entanto, é importante citar que não é possível fazer uma associação direta da pontuação MOS com a perda de pacotes apresentada. Isso se deve ao fato de que não apenas a porcentagem de pacotes perdidos é levada em consideração para o cálculo do MOS, mas também o momento em que as perdas ocorreram no decorrer da chamada. Perdas que ocorrem mais próximas do início da chamada possuem uma influência menor na pontuação do MOS, por serem eliminadas da memória do interlocutor com o decorrer da chamada. Além disso, também é levado em consideração a ocorrência de perdas consecutivas, ou seja, se houve perdas em rajada durante a transmissão, já que perdas esparsas não são detectadas pelo ouvido humano [6].

Para chamadas realizadas com a abordagem de compressão de cabeçalhos e agregação de pacotes (SHC+AG), o MOS apresentou um comportamento particular. Para os experimentos com poucas chamadas simultâneas, o MOS apresentado foi mais baixo do que o MOS das outras configurações de compressão. Podemos justificar este comportamento pelo alto atraso apresentado por essa abordagem, conseqüência do mecanismo de agregação de pacotes.

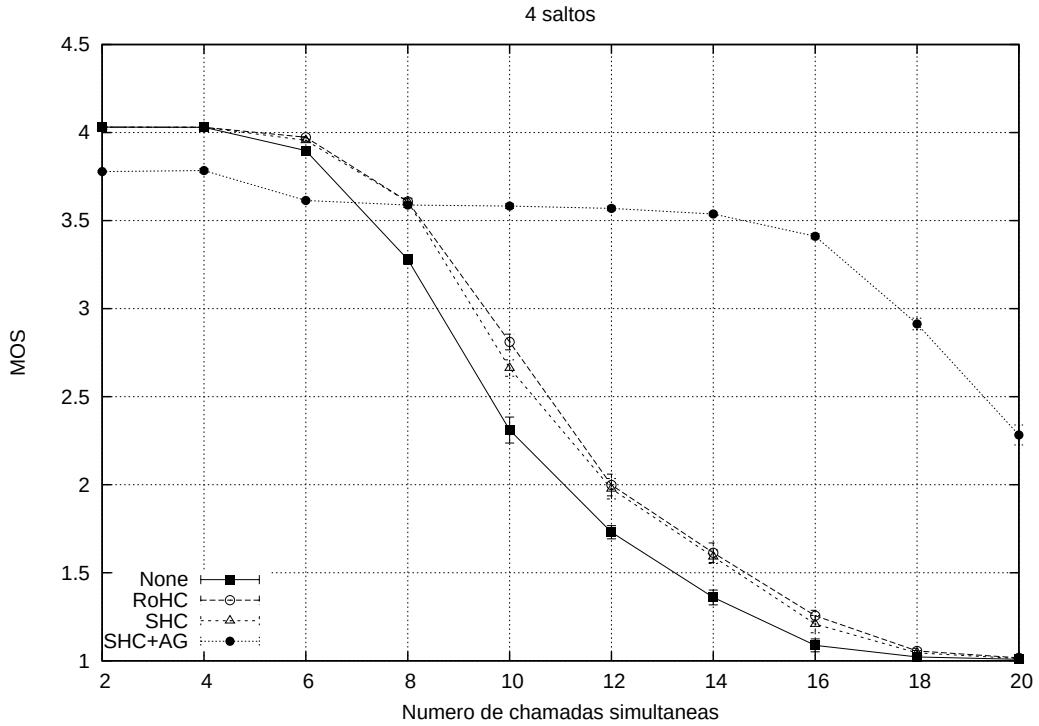

Figura 5.15: Pontuação MOS para chamadas de 4 saltos no cenário linear.

No entanto, com o acréscimo de chamadas simultâneas, o MOS apresentado pela abordagem SHC+AG diminui mais lentamente do que para os outros algoritmos, passando a ser, a partir de um dado momento, o maior MOS apresentado. Esse comportamento de maior estabilidade do MOS com relação ao acréscimo de chamadas simultâneas pode ser justificado pelo comportamento similar observado na perda de pacotes e no atraso dessas chamadas, influenciados pelo grau de compressão do algoritmo, e pelo mecanismo de agregação de pacotes.

O efeito do aumento do número de saltos sem fio pode ser observado na quantidade de chamadas toleradas. Para as chamadas com 2 saltos sem fio, podemos ver que com 18 chamadas simultâneas o MOS se mantém em torno de 3,5 para todos as configurações de compressão. Já para 3 saltos, esse número diminui para 12 chamadas, 8 chamadas com 4 saltos e 6 chamadas com 5 saltos. A tabela 5.4 mostra a quantidade de chamadas suportadas com MOS de no mínimo 3,6, considerado o mínimo para um chamada ser considerada de qualidade aceitável [6].

Além dos experimentos sobre o cenário linear, o MOS também foi calculado para os

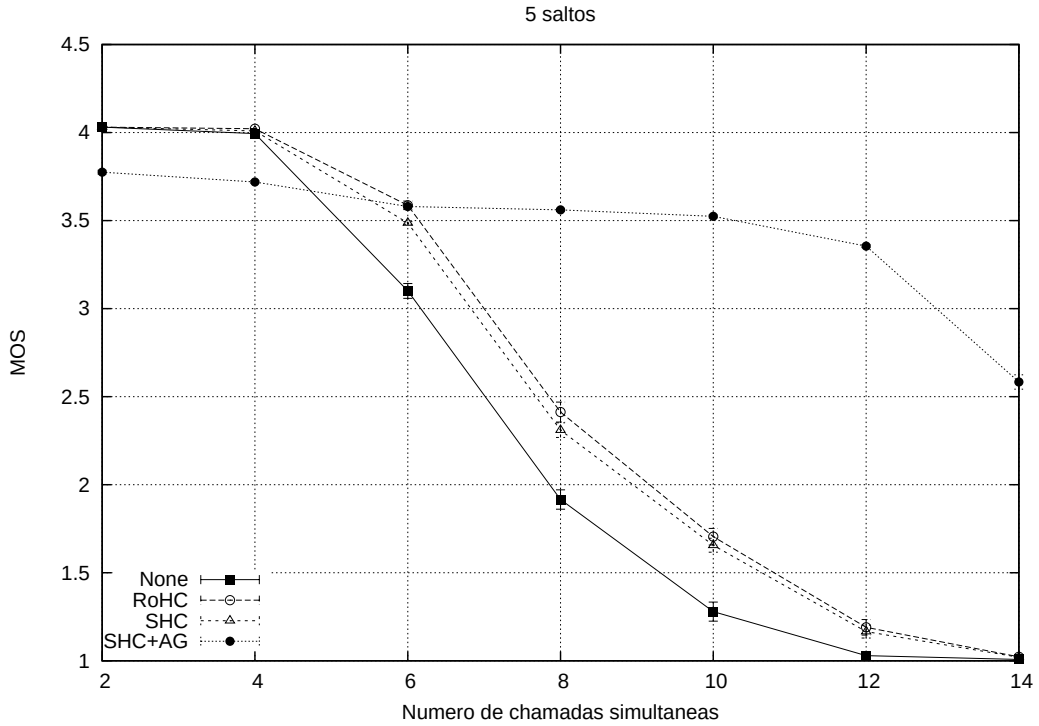

Figura 5.16: Pontuação MOS para chamadas de 5 saltos no cenário linear.

Tabela 5.4: Quantidade de chamadas simultâneas realizadas com no mínimo 3,6 de pontuação MOS.

Algoritmo	**2 saltos**	**3 saltos**	**4 saltos**	**5 saltos**
Sem compressão	16 chamadas	10 chamadas	6 chamadas	4 chamadas
SHC	18 chamadas	10 chamadas	8 chamadas	4 chamadas
RoHC	18 chamadas	12 chamadas	8 chamadas	6 chamadas
SHC + Agregação	32 chamadas	18 chamadas	12 chamadas	8 chamadas

experimentos conduzidos sobre o cenário em árvore. O cenário em árvore foi formado por 7 nós formando uma árvore de 3 níveis e 4 folhas, sendo a raiz da árvore considerada o *gateway* da rede (nó 0). Foram realizadas chamadas destinadas ao nó 0, a partir de todos os nós da rede, e a partir dos nós folhas, em momentos distintos. As Figuras 5.17 e 5.18 mostra os valores de MOS calculados sobre essas chamadas.

Como podemos observar, as chamadas realizadas sem compressão de cabeçalhos mostraram os menores valores de MOS na maioria dos casos. Em seguida, os algoritmos RoHC e SHC mostraram valores de MOS mais altos, mas não mostraram diferença significativa entre si. Isso reflete o mesmo comportamento mostrado sobre o cenário linear, e as justificativas são as mesmas: os valores de perda de pacotes e atraso apresen-

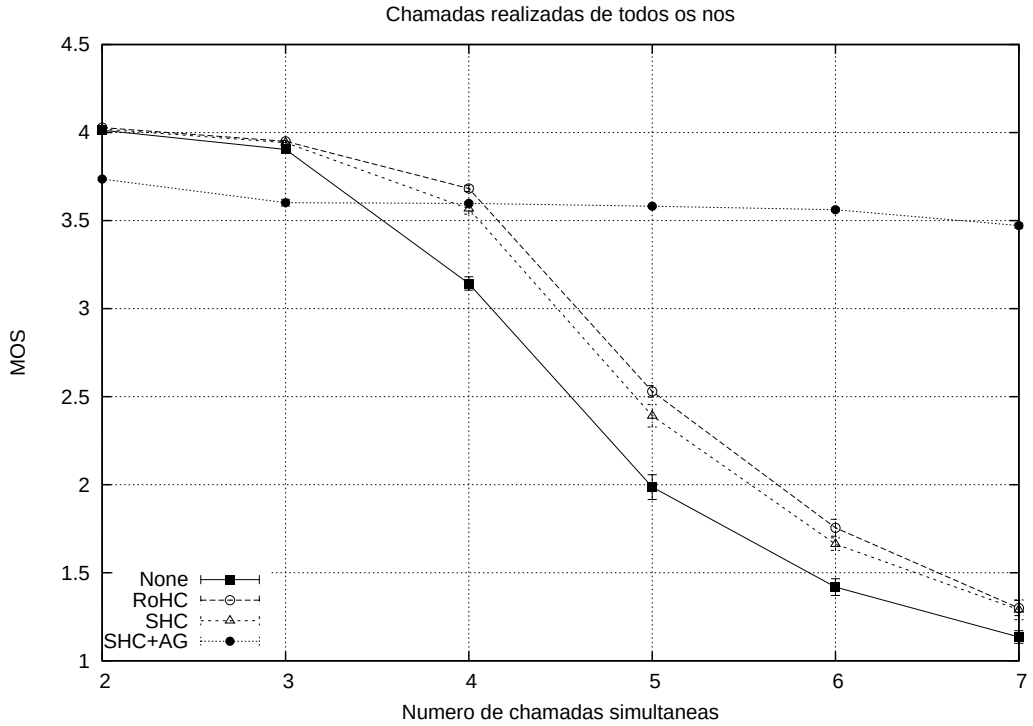

Figura 5.17: Pontuação MOS para chamadas realizadas sobre o cenário em árvore, a partir de todos os nós.

tado pelos dois algoritmos não foram diferentes o suficiente para apresentar diferença significativa no MOS.

Já a abordagem proposta SHC+AG mostrou os valores de MOS mais baixos para poucas chamadas simultâneas, e os valores mais altos com o acréscimo do número de chamadas. Mais uma vez o MOS apresentou esse comportamento mais estável com o aumento de chamadas, em comparação aos outros algoritmos. Isso se justifica pelo comportamento também mais estável apresentado pelas métricas de atraso e perda de pacotes.

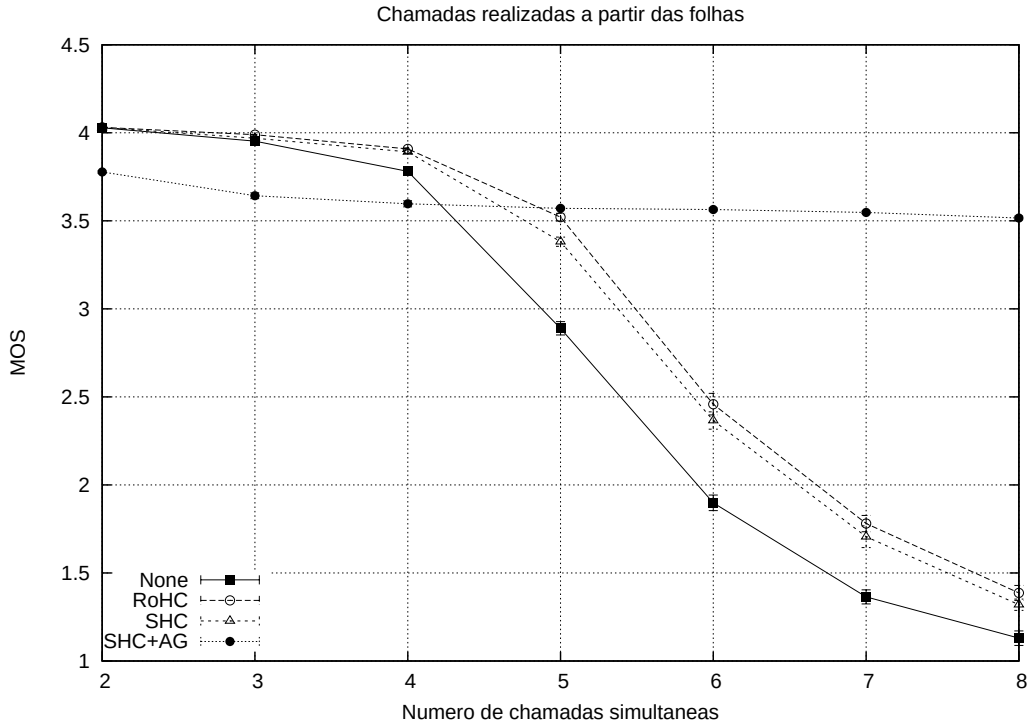

Figura 5.18: Pontuação MOS para chamadas realizadas sobre o cenário em árvore, a partir dos nós folha.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste capítulo estão contidas as conclusões obtidas a partir da análise dos resultados encontrados. São mostradas também as contribuições dadas por este trabalho à solução do problema do uso de compressão de cabeçalhos em redes *mesh*. Por último são mostradas sugestões de trabalhos futuros que podem contribuir na investigação dos problemas apresentados nesta dissertação.

6.1 Conclusões

Esta dissertação apresentou uma abordagem de compressão de cabeçalhos para VoIP, que se utiliza de um algoritmo de compressão de cabeçalhos estática e uma técnica simples de agregação de pacotes. A motivação desta dissertação está no fato de o uso otimizado da largura de banda ser algo de suma importância para redes sem fio, e no fato de os algoritmos de compressão de cabeçalhos para VoIP existentes atualmente não serem próprios para redes de múltiplos saltos sem fio.

A abordagem sugerida faz uso de um algoritmo de compressão de cabeçalhos estática para eliminar o problema da propagação da perda, presente nos algoritmos de compressão que fazem uso da memória recente do fluxo de voz. A agregação de pacotes foi uma alternativa utilizada para aumentar o ganho da compressão estática, através da eliminação de informações redundantes nos cabeçalhos dos pacotes agregados.

Foram apresentados também resultados de experimentos realizados com a inten-

ção de avaliar a abordagem sugerida, em comparação com um algoritmo que faz uso de memória recente (RoHC), um algoritmo de compressão estática puro, e chamadas realizadas sem compressão de cabeçalhos. Foram utilizadas 5 métricas de avaliação, sendo elas ganho de compressão, eficiência largura de banda, perda de pacotes, atraso na rede e MOS.

A partir dos resultados obtidos dos experimentos realizados para esta dissertação podemos concluir que a escolha de soluções de compressão de cabeçalhos para VoIP em redes *mesh* não é trivial. Esse tipo de rede, por ser constituída fundamentalmente de múltiplos saltos sem fio, apresenta desafios extras para a utilização de algoritmos de compressão.

Podemos checar essa dificuldade através dos resultados obtidos para o MOS, em todas as configurações de compressão utilizadas. Apesar de o MOS das chamadas realizadas com algoritmos de compressão ter sido um pouco maior que o MOS das chamadas sem compressão, essa diferença não foi tão grande. Isso mostra que para redes de múltiplos saltos sem fio as propostas de compressão existentes atualmente podem não ser suficientes para mostrar um aumento significativo na qualidade das chamadas de voz.

A escolha do algoritmo de compressão apropriado é também importante. Para os nossos cenários, a diferença entre os resultados dos algoritmos RoHC e de compressão estática não foi significativa na maioria dos casos, se levarmos em consideração as métricas de atraso na rede, perda de pacotes e MOS. Apesar de apresentar maior robustez com ganho de compressão e eficiência de banda maiores, o algoritmo RoHC apresenta complexidade maior que um algoritmo simples de compressão estática. Portanto, nos casos em que for preferível um algoritmo de compressão de cabeçalhos mais simples em detrimento do ganho de compressão atingido, um algoritmo de compressão estática pode ser a opção apropriada.

Com relação aos resultados obtidos pela abordagem proposta nesta dissertação, apesar de os valores de MOS não mostrarem melhoria com relação aos outros algoritmos de compressão, através dos resultados das métricas de perda de pacotes e atraso na

rede, podemos concluir que o uso dessa abordagem pode ser vantajoso. O atraso obtido ainda fica longe de ultrapassar o máximo permitido para chamadas de voz, e a perda de pacotes, bem abaixo da perda apresentada pelos outros algoritmos, pode ser de grande vantagem para o desempenho da rede, por aumentar a vazão de dados de voz.

Além disso, a estabilidade mostrada nas métricas de perda, atraso e MOS, dado o acréscimo de chamadas simultâneas, sugere que a utilização desse mecanismo pode representar um caminho para o desenvolvimento de um mecanismo de compressão de cabeçalhos apropriado para redes sem fio de múltiplos saltos.

6.2 Trabalhos futuros

Esta seção traz algumas sugestões de trabalhos futuros que investigam tanto as implicações das conclusões obtidas através dos resultados, quanto outros aspectos relacionados a compressão de cabeçalhos não abordados nesta dissertação.

6.2.1 Testes em ambiente real

Como trabalhos futuros, a primeira sugestão é a implementação e avaliação da abordagem proposta em um ambiente real. Apesar de todo cuidado tomado para reproduzir o mais fielmente possível um ambiente real para a realização dos nossos experimentos de simulação, através da utilização de um modelo de propagação mais adequado aos cenários sem fio utilizados, é sabido que um modelo de simulação nunca poderá reproduzir totalmente um ambiente real e oferecer um tempo de simulação aceitável.

Quanto maior é a quantidade de detalhes levados em consideração ao se elaborar um modelo de simulação, maior é a complexidade desse modelo, e maior será o tempo levado para a simulação ser executada. Portanto, alguns fatores não são levados em consideração para o modelo de simulação, com o objetivo de manter o tempo de simulação dentro do aceitável, sem que no entanto este modelo deixe de oferecer uma acurácia moderada dos resultados. Por isso, experimentos de medições em ambientes reais aumentariam a credibilidade dos resultados de simulações obtidos pelos experimentos de simulação realizados neste trabalho [8].

6.2.2 Análise do tempo de execução dos algoritmos

Outro fator que merece atenção é o tempo de execução dos algoritmos de compressão. Os experimentos realizados aqui não levaram em consideração o tempo de compressão e descompressão, visto que foi mensurado apenas o atraso dos pacotes na rede. No entanto, como o sistema de compressão descrito aqui implementa a compressão e descompressão nos roteadores do *backbone mesh*, é importante ressaltar que os procedimentos de compressão e descompressão, executados para cada pacote que entra ou sai do *backbone*, devem ser rápidos e eficientes, independente do mecanismo utilizado.

Portanto, um fator adicional para a análise comparativa dos algoritmos citados aqui seria através da medição do tempo de execução dos principais procedimentos de cada um, sobre plataformas idênticas às encontradas nos roteadores sem fio mais utilizados para a implantação de redes *mesh*. A intenção deste procedimento seria saber se os algoritmos não representariam carga além do suportado pelas plataformas dos roteadores, consumindo muito de tempo de execução ou de outros recursos do dispositivo.

6.2.3 Testes com outros *codecs*

Os *codecs* de voz utilizados influenciam diretamente no tamanho dos *payloads* dos pacotes de voz gerados, devido a taxa de codificação utilizada, ao tamanho dos quadros gerados, e à quantidade de quadros incluídos em cada pacote de dados. Em nossos experimentos, foi utilizado apenas o *codec* G.729a gerando quadros de 20 ms, e incluindo 1 quadro por pacote, o que resultou em pacotes de 20 bytes de *payload*.

Como a grande contribuição dos algoritmos de compressão de cabeçalhos é na diminuição do tamanho total dos pacotes, através da diminuição dos tamanhos dos cabeçalhos, a análise do comportamento desses algoritmos na presença de diferentes tamanhos de *payloads* poderia dar uma visão mais generalizada na análise, além de identificar qualquer variação no comportamento dos algoritmos em comparação ao que foi apresentado nesta dissertação.

6.2.4 Estudo de soluções de roteamento para a utilização de compressão fim-a-fim

Outro aspecto importante a ser analisado são soluções de roteamento para dar suporte a compressão de cabeçalhos fim-a-fim. Para o nosso trabalho, foi utilizado roteamento por rótulos implementado com o MPLS. No entanto, nossos experimentos não podem dar a conclusão de que esta é a melhor opção para realizar roteamento independente da camada de redes.

A pesquisa de soluções de encaminhamento de pacotes para dar suporte ao uso de compressão de cabeçalhos fim-a-fim em redes múltiplos saltos sem fio é importante, pois como foi dito anteriormente, a compressão feita salto-a-salto pode apresentar um custo adicional muito alto pela manutenção de pares compressor-descompressor para cada salto da rede, além do tempo de compressão e descompressão que seria feito em cada nó intermediário da rota entre a fonte e o destinatário.

6.2.5 Proposta de algoritmo de compressão de cabeçalhos adaptativo

Como foi observado nos resultados dos experimentos desta dissertação, para maiores quantidades de chamadas simultâneas a abordagem de compressão de cabeçalhos e agregação de pacotes apresenta os melhores valores de perda, atraso e MOS, e para quantidades menores de chamadas os outros algoritmos mostraram os melhores valores. Partindo desse pressuposto, podemos sugerir o estudo de um mecanismo de compressão capaz de trocar seu método principal de compressão de cabeçalhos a partir das condições da rede, ou do tráfego.

A idéia seria que, para redes menos carregadas fosse utilizado um mecanismo de compressão simples, e conforme a carga da rede aumentasse, os compressores tivessem a opção de trocar o algoritmo de compressão utilizado para um mais robusto, como o mecanismo de compressão e agregação apresentado aqui.

6.3 Publicações científicas

Como resultados do estudo desenvolvido nesta dissertação, podemos citar também os trabalhos científicos publicados. São eles 3 artigos em conferências internacionais *Qualis* A, e uma produção em andamento.

- NASCIMENTO, A.G.O. ; MOTA, E. ; QUEIROZ, S. ; MOTA, E. ; NASCIMENTO, A. ; NASCIMENTO, E.. "*Header compression for VoIP over multi-hop wireless mesh networks*". Em: *The IEEE Symposium on Computers and Communications*. ISCC 2008. Marrakesh, Marrocos. (Qualis A). ***Best paper award***.

- NASCIMENTO, A.G.O. ; MOTA, E. ; QUEIROZ, S. ; GALVAO, L.. "*Towards an Efficient Header Compression Scheme to Improve VoIP over Wireless Mesh Networks*". A ser publicado em: *The IEEE Symposium on Computers and Communications*. ISCC 2009. Sousse, Tunisia. (Qualis A)

- NASCIMENTO, A.G.O. ; MOTA, E. ; QUEIROZ S. ; NASCIMENTO, E.. "*An Alternative Approach for Header Compression Over Wireless Mesh Networks*". A ser publicado em: *Fifth International Symposium on Frontiers of Information Systems and Network Applications* (FINA). AINA 2009. Bradford, Reino Unido.

Além desses, atualmente ainda está em desenvolvimento um artigo para publicação em um *journal* internacional, ainda a ser definido.

Referências Bibliográficas

[1] J. Jun and M. L. Sichitiu, "The Nominal Capacity of Wireless Mesh Networks," *IEEE Wireless Communications*, vol. 10, no. 5, pp. 8–14, 2003.

[2] S. A. Ahson and M. Ilyas, *VoIP Handbook: Applications, Technology, Reliability and Security.* Boca Raton, FL, USA: CRC Press Taylor & Francis Group, 2009.

[3] K. Kim, S. Ganguly, R. Izmailov, and S. Hong, "On Packet Aggregation Mechanisms for Improving VoIP Quality in Mesh Networks," in *Proceedings of the Vehicular Technology Conference, VTC'06.* IEEE, 2006, pp. 891–895.

[4] "The NS Manual," 2007.

[5] E. Mota, "Performance of Sequential Batching-based Methods of Output Data Analysis in Distributed Steady-state Stochastic Simulation," Berlim, Alemanha, 2002.

[6] L. Carvalho, "Implementação do Modelo E para Avaliação Objetiva da Qualidade da Fala em Redes de Comunicação VoIP," Manaus, AM, Brasil, 2004.

[7] H. Schulzrinne and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control, Request for Comments 3551," 2003.

[8] R. Jain, *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling.* Wiley Computer Publishing, John Wiley & Sons, Inc., 1991.

[9] IEEE, "IEEE 802.11TM Wireless Local Area Networks," 2004. [Online]. Available: http://grouper.ieee.org/groups/802/11/

[10] F. Fitzek, T. Madsen, P. Popovski, and R. P. e M. Katz, "Cooperative IP Header Compression for Parallel Channels in a Wireless Meshed Networks," in *Proceedings of the International Conference on Communications - ICC'05*, vol. 2, 2005, pp. 1331–1335.

[11] P. Seeling, M. Reisslein, T. K. Madsen, and F. H. Fitzek, "Performance Analysis of Header Compression Schemes in Heterogeneous Wireless Multi—Hop Networks," *Wirel. Pers. Commun.*, vol. 38, no. 2, pp. 203–232, 2006.

[12] N. Fukumoto and H. Yamada, "Performance Enhancement of Header Compression over Asymmetric Wireless Links Based on the Objective Speech Quality Measurement," in *Proceedings of the 2007 International Symposium on Applications and the Internet - SAINT'07*. Washington, DC, USA: IEEE Computer Society, 2007, p. 16.

[13] Abinader Jr., F. M., "Avaliação de desempenho de algoritmos de compressão de cabeçalho cooperativos para aplicações VoIP sem fio," Tese de Mestrado, Manaus, AM, Brasil, 2006.

[14] C. Westphal and R. Koodli, "Stateless IP Header Compression," *ICC 2005: Proceedings of the 2005 IEEE International Conference on Communications*, vol. 5, pp. 3236 – 3241, 2005.

[15] T. K. Madsen, F. H. Fitzek, Y. Takatori, R. Prasad, and M. Katz, "Cooperative IP Header Compression using Multiple Access Points in 4G Wireless Networks," 2005.

[16] H. Jin, R. Hsu, and J. Wang, "Performance comparison of header compression schemes for RTP/UDP/IP packets," in *Proceedings of the IEEE Wireless Communications & Networking Conference - WCNC'04*, vol. 3, 2004, pp. 1691–1696.

[17] Y. Zhuang, K. Tan, V. Shen, and Y. Liu, "VoIP Aggregation in Wireless Backhaul Networks," in *Proceedings of the International Conference on Communications - ICC'06*, vol. 12. IEEE, 2006, pp. 5468–5473.

[18] M. Castro, P. Dely, J. Karlsson, and A. Kassler, "Capacity Increase for Voice over IP Traffic through Packet Aggregation in Wireless Multihop Mesh Networks," in *Proceedings of the Future Generation Communication and Networking - FGCN'07*, vol. 2. IEEE, 2007, pp. 350–355.

[19] S. Jung, S. Hong, and P. Park, "Effect of RObust Header Compression (ROHC) and Packet Aggregation on Multi-hop Wireless Mesh Networks," in *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology - CIT'06*. Washington, DC, USA: IEEE Computer Society, 2006, p. 91.

[20] S. Ganguly, V. Navda, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. R. Das, "Performance Optimizations for Deploying VoIP Services in Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2147–2158, 2006.

[21] Y. Zhang, J. Luo, and H. Hu, Eds., *Wireless Mesh Networking: Architectures, Protocols and Standards*, 1st ed. New York: Auerbach Publications, Taylor & Francis Group, 2006.

[22] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, March 2005.

[23] S. Misra, S. C. Misra, and I. Woungang, *Guide to Wireless Mesh Networks*. Springer Publishing Company, Incorporated, 2008.

[24] S. M. Faccin, C. Wijting, J. Kenckt, and A. Damle, "Mesh WLAN Networks: Concept and System Design," vol. 13, no. 2, pp. 10–17, 2006.

[25] H. Fathi, S. S. Chakraborty, and R. Prasad, *Voice over IP in Wireless Heterogeneous Networks - Signaling, Mobility and Security*. Springer Science+Business Media B.V., 2009.

[26] J. F. Kurose and K. W. Ross, *Redes de Computadores e a Internet*. São Paulo: Addison-Wesley, 2003.

[27] "ITU-T Recommendation G.114," Telecommunication Standardization Sector of ITU, Tech. Rep., 2003.

[28] C. Bormann, C. Burmeister, M. Degermark, H. Fukushima, H. Hannu, L.-E. Jonsson, R. Hakenberg, T. Koren, K. Le, Z. Liu, A. Martensson, A. Miyazaki, K. Svanbro, T. Wiebke, T. Yoshimura, and H. Zheng, "Robust Header Compression: Framework and four profiles, Request for Comments 3095," 2001.

[29] S. Casner and V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links, Request for Comments 2508," 1999.

[30] V. Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links, Request for Comments 1144," 1990.

[31] T. Koren, S. Casner, J. Geevarghese, B. Thompson, and P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering, Request for Comments 3545," 2003.

[32] L.-E. Jonsson, K. Sandlund, G. Pelletier, and P. Kremer, "Robust Header Compression: Corrections and Clarifications to RFC 3095, Request for Comments 4815," 2007.

[33] G. Franceschetti and S. Stornelli, *Wireless Networks: From the Physical Layer to Communication, Computing, Sensing and Control.* Orlando, FL, USA: Academic Press, Inc., 2006.

[34] V. Garg, Ed., *Wireless Communications and Networking*, 1st ed. San Francisco: Elsevier Inc., 2007.

[35] "The Network Simulator - NS-2." [Online]. Available: http://www.isi.edu/nsnam/ns/

[36] G. Ewing, K. Pawlikowski, and D. McNickle, "Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation," Warsaw, pp. 175–181, 1999.

[37] S. Queiroz, A. Nascimento, E. Mota, A. G. O. do Nascimento, L. Galvao, and E. Nascimento, "Impact Evaluation of Radio Propagation Models on Performance Parameters of Application Layer in Wireless Mesh Backbone Simulation," in *Proceedings of the IEEE Symposium on Computers and Communication - ISCC'2008*. Marrakesh, Morocco: IEEE Computer Society, 2008.

[38] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental Evaluation of Wireless Simulation Assumptions," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems - MSWiM'04*. New York, NY, USA: ACM, 2004, pp. 78–82.

[39] F. Schmidt-Eisenlohr, J. Letamendia-Murua, M. Torrent-Moreno, and H. Hartenstein, "Bug Fixes on the IEEE 802.11 DCF module of the Network Simulator Ns-2.28, Technical Report," 2006. [Online]. Available: http://www.telematica.polito.it/fiore/index.html

[40] W. Xiuchao, "Simulate 802.11b Channel within Ns-2," 2004. [Online]. Available: http://www.comp.nus.edu.sg/~wuxiucha/research/reactive/report/80211ChannelinNS_2new.pdf

[41] T. S. Rappaport, *Wireless Communication Principles and Practice*. Prentice Hall PTR, 2001.

[42] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture, Request for Comments 3031," 2001.

[43] M. Petersson, "MPLS Network Simulator versão 2.0 para NS-2 2.26," 2004. [Online]. Available: http://heim.ifi.uio.no/~johanmp/ns-2/mns-for-2.26.tar.gz

[44] K. Pawlikowski, "Steady-state simulation of queueing processes: survey of problems and solutions," *ACM Comput. Surv.*, vol. 22, no. 2, pp. 123–170, 1990.

Apêndice A

Experimentos adicionais

Diante do ótimo desempenho apresentado pela abordagem de compressão de cabeçalhos e agregação de pacotes sugerida neste trabalho, experimentos extras foram realizados para testar outros aspectos da abordagem, bem como para oferecer outras avaliações comparativas dos resultados obtidos. Os resultados apresentados neste capítulo levam em consideração apenas as métricas de perda de pacotes, MOS e atraso na rede. As condições e o ambiente em que esses experimentos foram realizados são as mesmas utilizadas para os experimentos apresentados no Capítulo 5.

Foi utilizado o cenário linear descrito em 4.2.1, com 6 nós sem fio dispostos em uma linha reta. As simulações foram realizadas utilizando o NS-2, e o Akaroa-2 com 95% de confiança e 5% de precisão relativa máxima.

A.1 Grau de agregação maior

Todos os experimentos realizados com a abordagem de compressão de cabeçalhos e agregação de pacotes mostrados no Capítulo 5 utilizaram grau de agregação de 2 pacotes, ou seja, os pacotes foram agregados de dois em dois. No entanto, através dos experimentos realizados não é possível afirmar que grau de agregação seria o mais adequado, levando em consideração características específicas do tráfego de voz, e o fato de a agregação se aplicar a pacotes de um mesmo fluxo.

Portanto, experimentos extras foram realizados com grau de agregação de 3 pacotes,

com 2, 3, 4 e 5 saltos sem fio sobre o cenário linear. As próximas subseções mostram os gráficos para perda de pacotes, atraso na rede e MOS, em comparação com os experimentos realizados com grau de agregação de 2 pacotes.

A.1.1 Perda de pacotes

As figuras A.1, A.2, A.3 e A.4 mostram os resultados para a perda de pacotes com o algoritmo de compressão e agregação, considerando graus de agregação 2 e 3.

Nos 4 gráficos podemos observar que as chamadas realizadas com grau de agregação 3 não mostraram uma diferença muito grande da perda apresentada pelas chamadas realizadas com grau de agregação 2, para a maioria dos experimentos.

No entanto, é possível observar que para as maiores quantidades de chamadas simultâneas a perda de pacotes para chamadas de grau de agregação 2 é maior, e apresenta um crescimento mais acentuado conforme o aumento da quantidade de chamadas. As chamadas com grau de agregação 3 apresentam um aumento na perda mais estável, conforme o aumento de chamadas simultâneas.

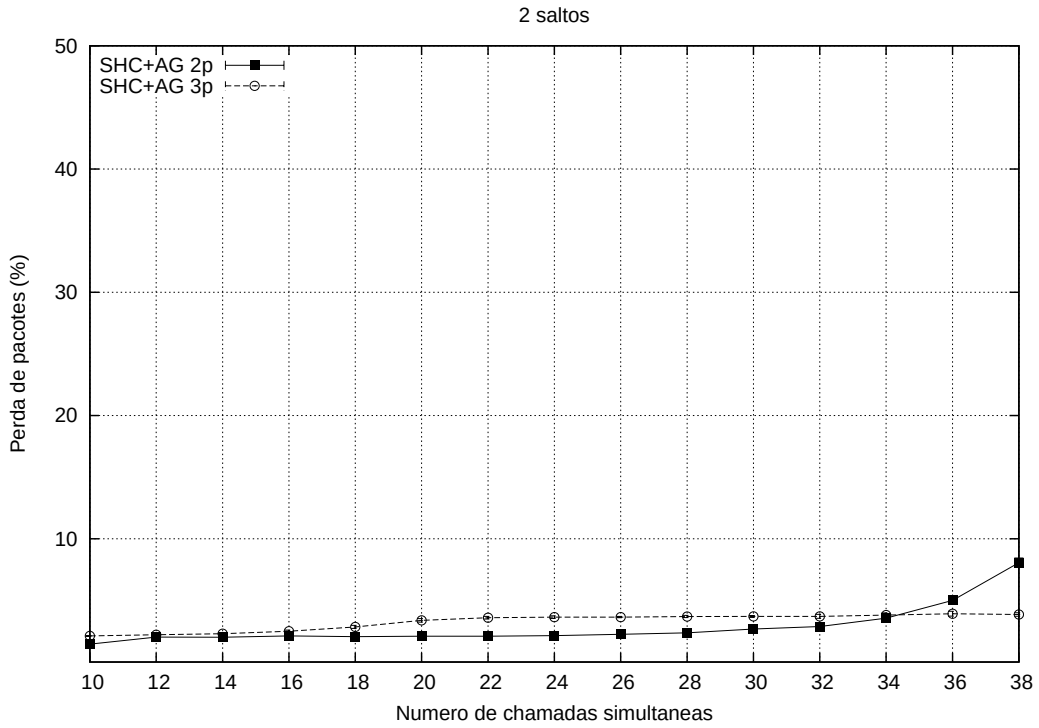

Figura A.1: Perda de pacotes para chamadas de 2 saltos com graus de agregação 2 e 3.

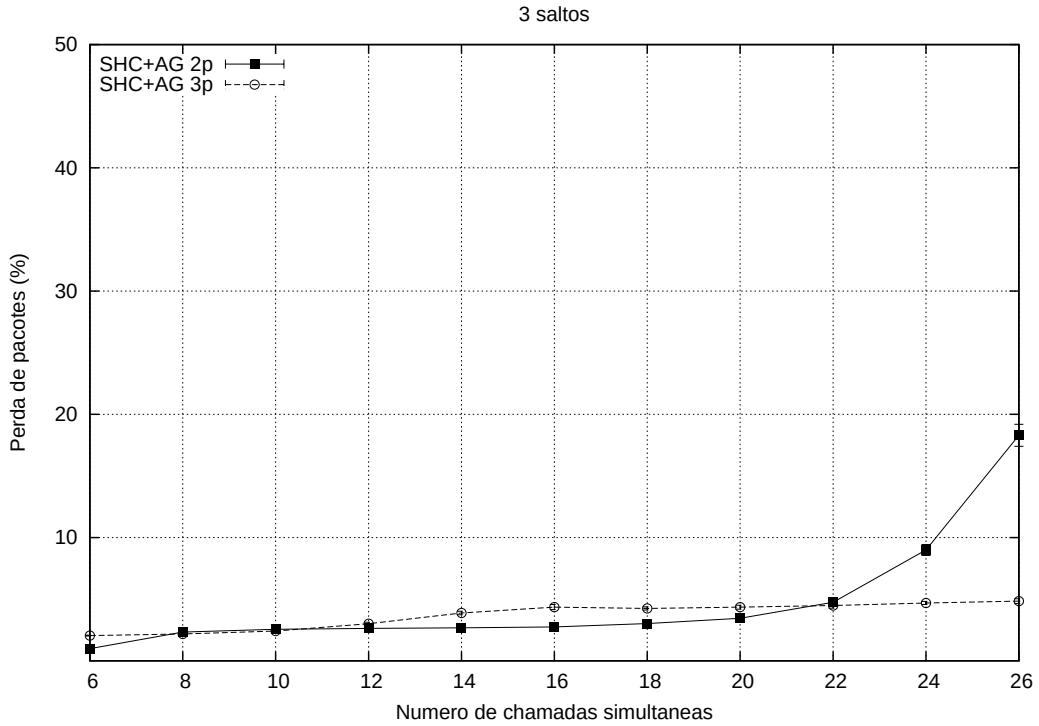

Figura A.2: Perda de pacotes para chamadas de 3 saltos com graus de agregação 2 e 3.

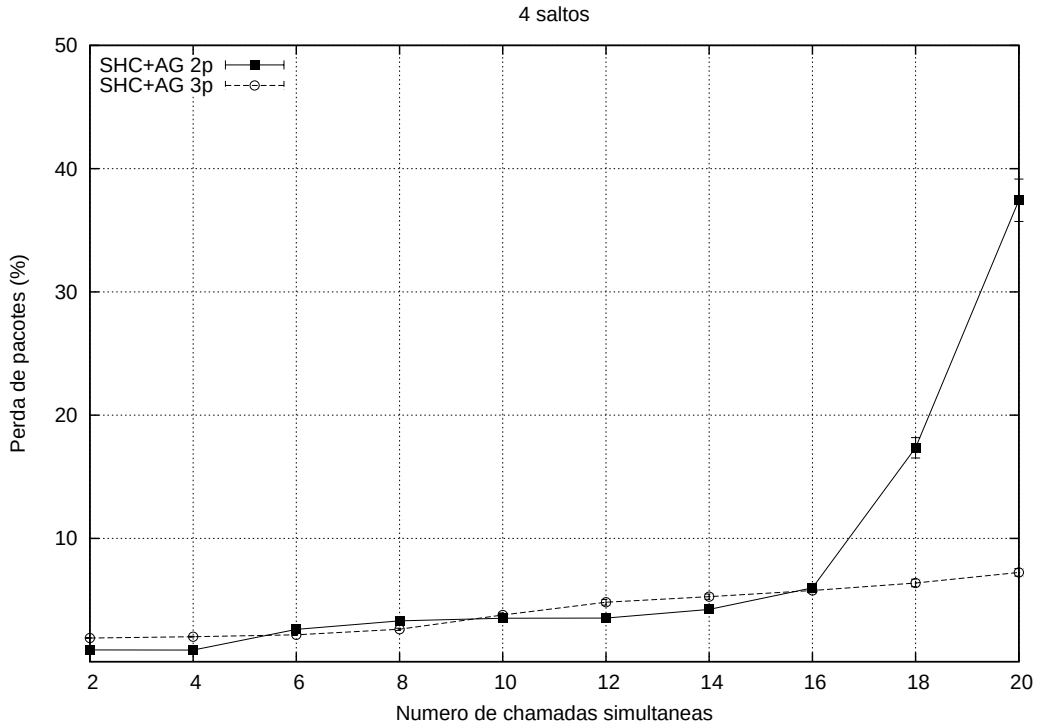

Figura A.3: Perda de pacotes para chamadas de 4 saltos com graus de agregação 2 e 3.

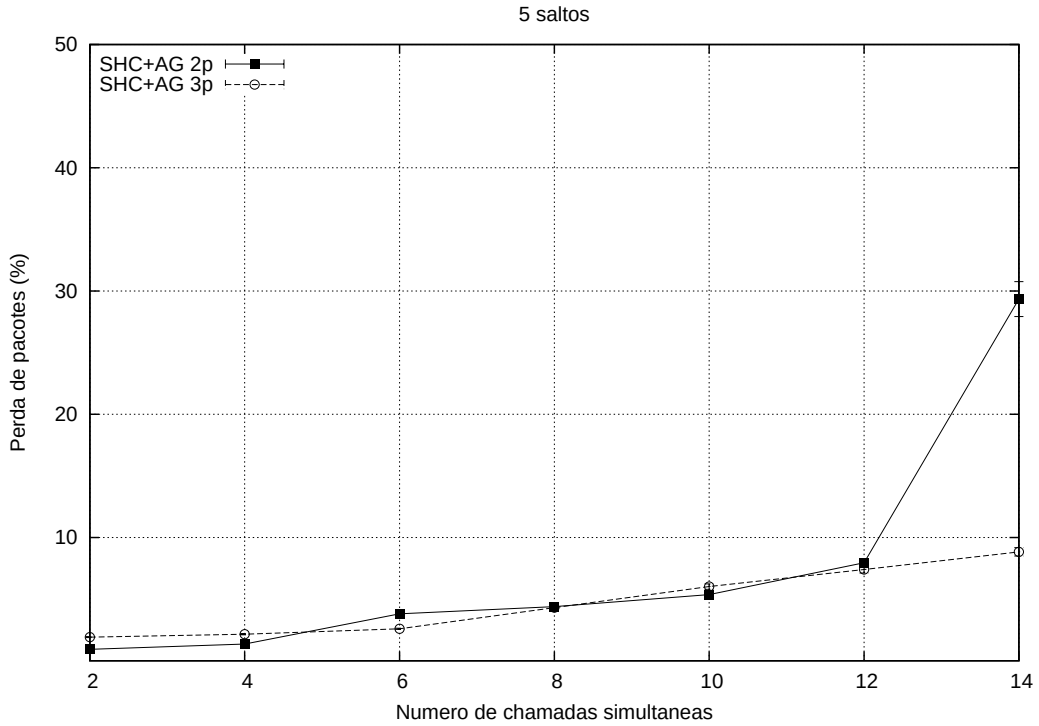

Figura A.4: Perda de pacotes para chamadas de 5 saltos com graus de agregação 2 e 3.

A.1.2 Atraso na rede

O grau de agregação influencia diretamente no atraso sofrido pelos pacotes de voz, devido ao fato de que a agregação é aplicada a pacotes do mesmo fluxo. Portanto, com o aumento do grau de agregação é esperado que o atraso médio sofrido pelos pacotes aumente, comportamento esse que pode ser verificado nos 4 gráficos mostrados nessa seção, para as chamadas realizadas com 2, 3, 4 e 5 saltos.

Os valores de atraso apresentados nos pacotes das chamadas realizadas com agregação de 3 pacotes foram mais altos. No entanto, em um comportamento análogo ao apresentado pelos gráficos da perda, é possível observar que o aumento no atraso, conforme a adição de chamadas simultâneas, é mais estável para as chamadas com grau de agregação 3.

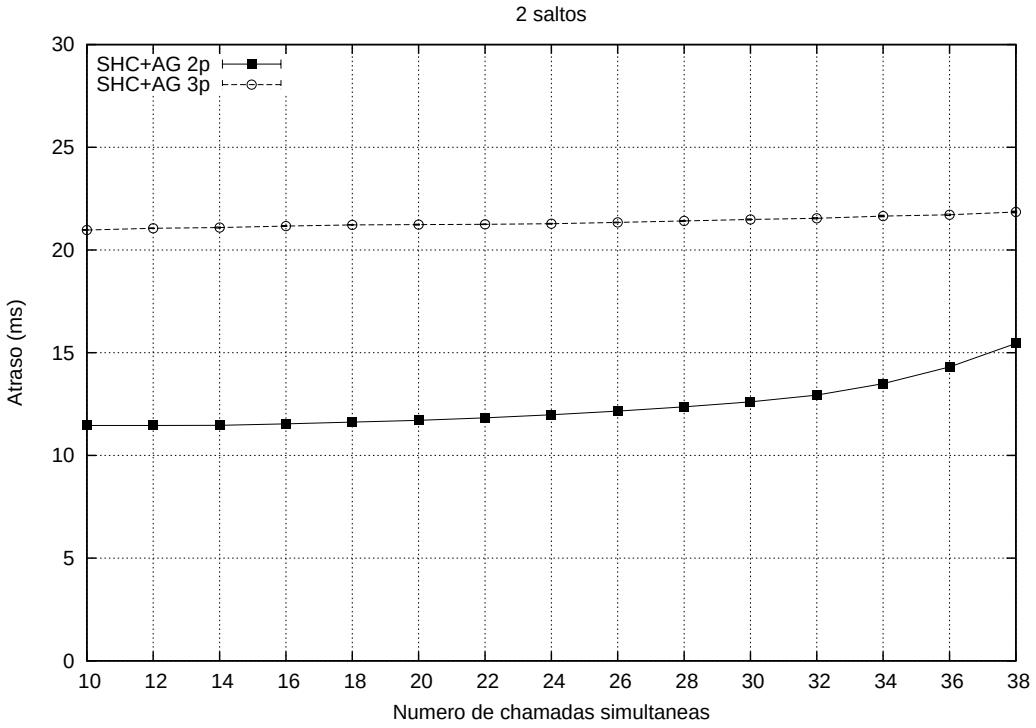

Figura A.5: Atraso na rede para chamadas de 2 saltos com graus de agregação 2 e 3.

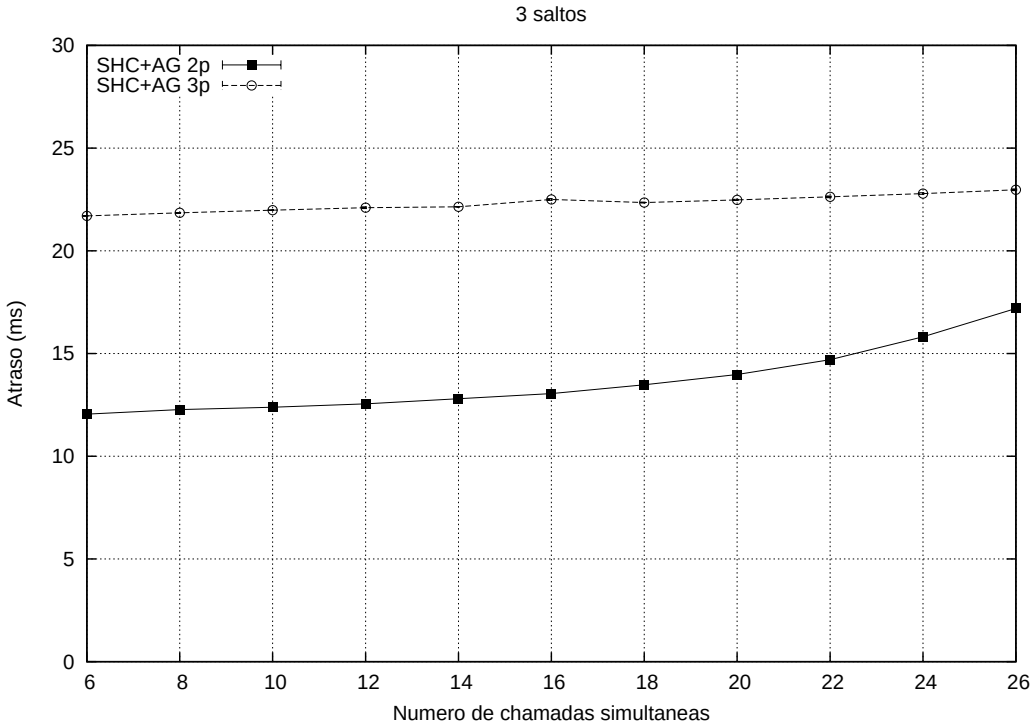

Figura A.6: Atraso na rede para chamadas de 3 saltos com graus de agregação 2 e 3.

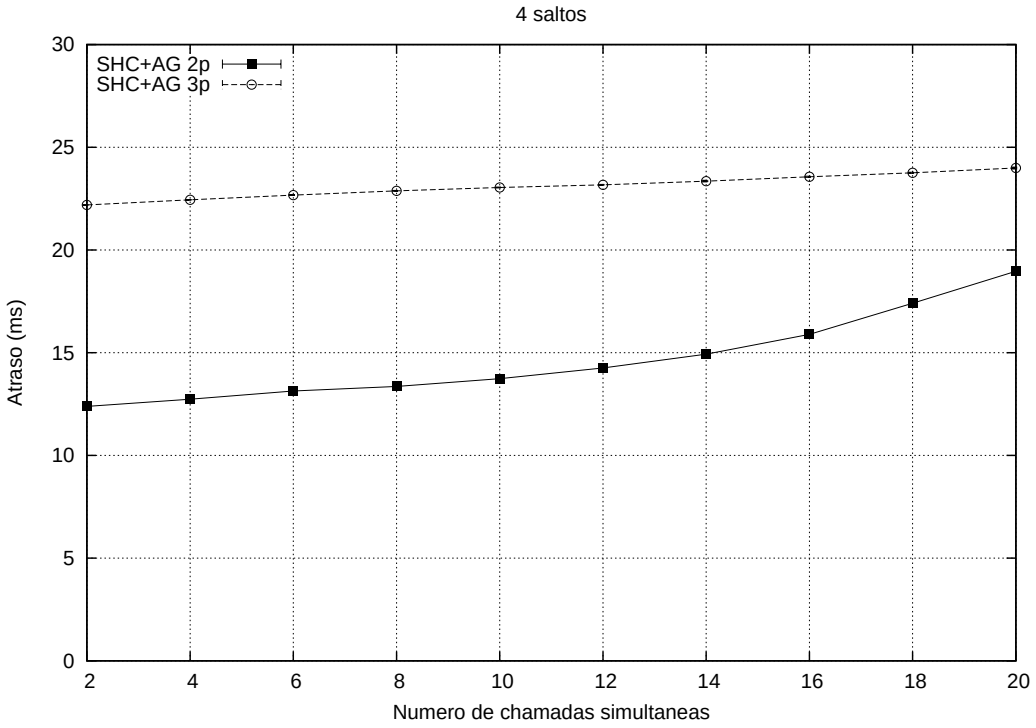

Figura A.7: Atraso na rede para chamadas de 4 saltos com graus de agregação 2 e 3.

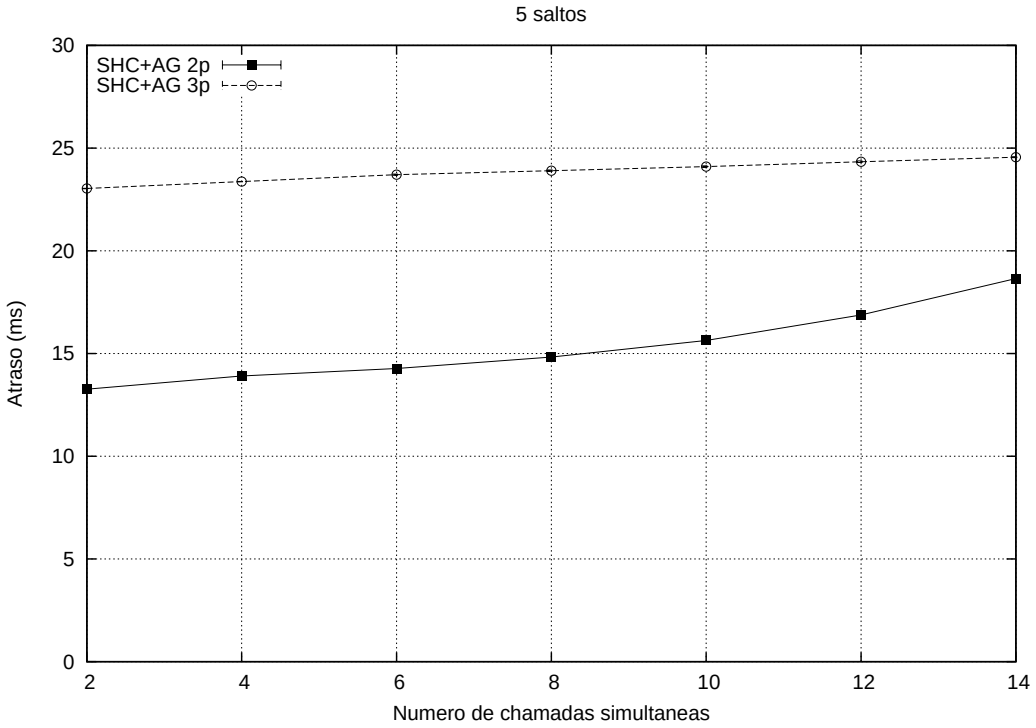

Figura A.8: Atraso na rede para chamadas de 5 saltos com graus de agregação 2 e 3.

A.1.3 MOS

Os valores de MOS mostrados aqui foram calculados para as mesmas chamadas que apresentaram perda e atraso apresentados nas subseções anteriores.

Nos gráficos é possível observar que os valores de MOS para as chamadas realizadas com grau de agregação 2 são mais altos que aqueles apresentados para as chamadas com grau de agregação 3, na maioria dos casos, ficando na maioria dos casos acima de 3,6, mínimo admitido para uma chamada ser considerada de qualidade boa [6].

No entanto, é possível notar uma diminuição mais constante e estável para as chamadas realizadas com grau de agregação 3, conforme o aumento no número de chamadas simultâneas. Esse comportamento reflete aquele apresentado pelos gráficos da perda de pacotes e do atraso na rede.

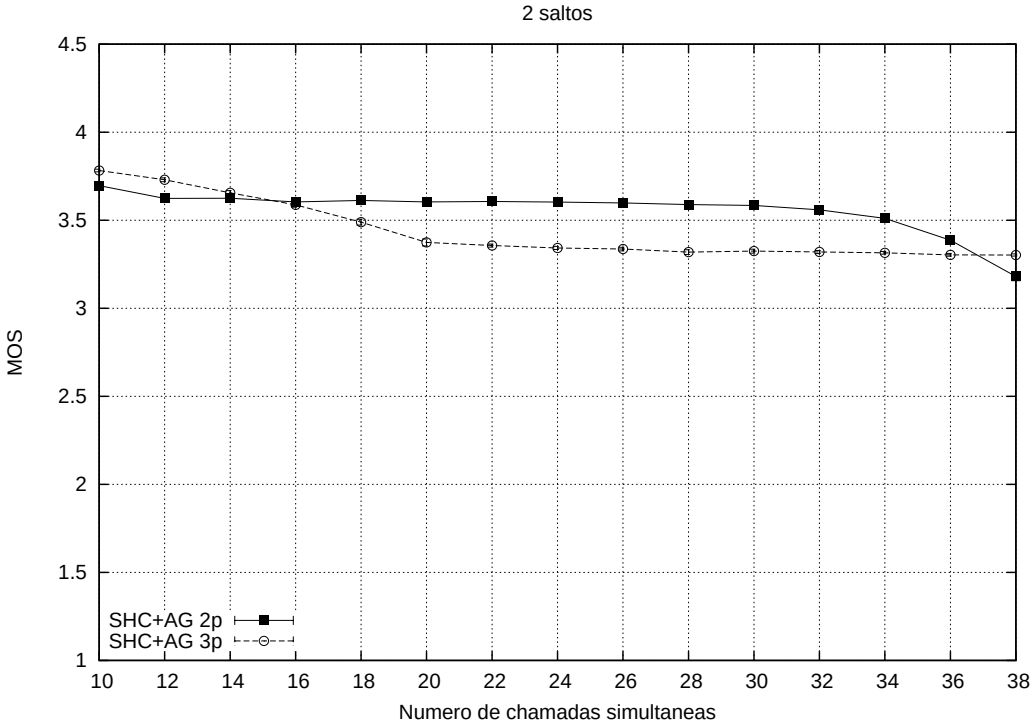

Figura A.9: MOS para chamadas de 2 saltos com graus de agregação 2 e 3.

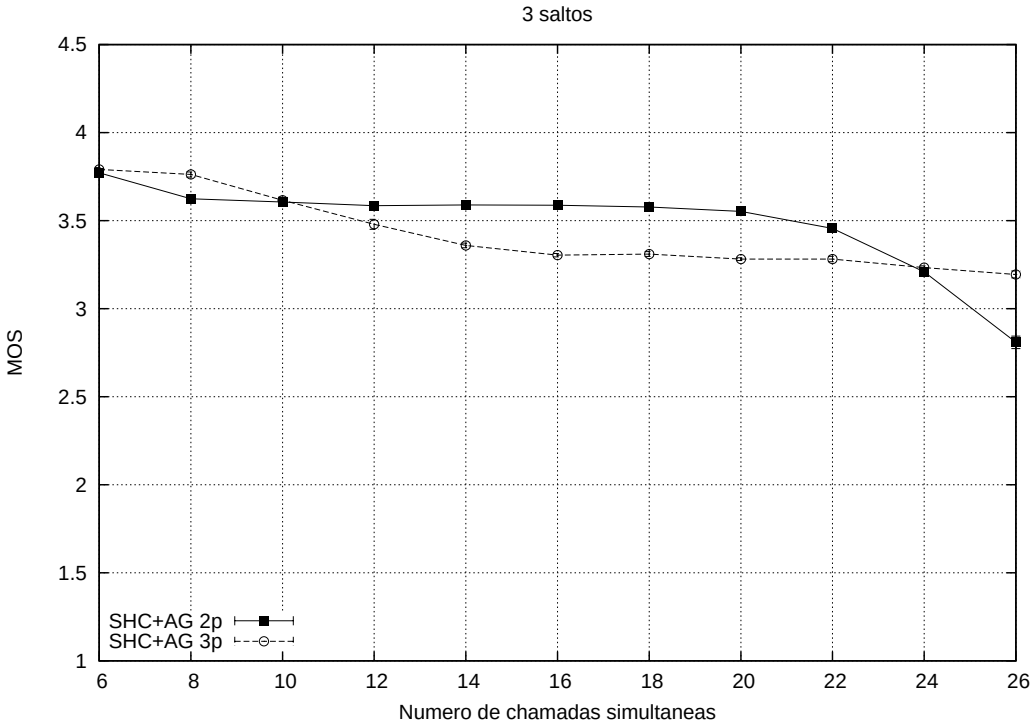

Figura A.10: MOS para chamadas de 2 saltos com graus de agregação 2 e 3.

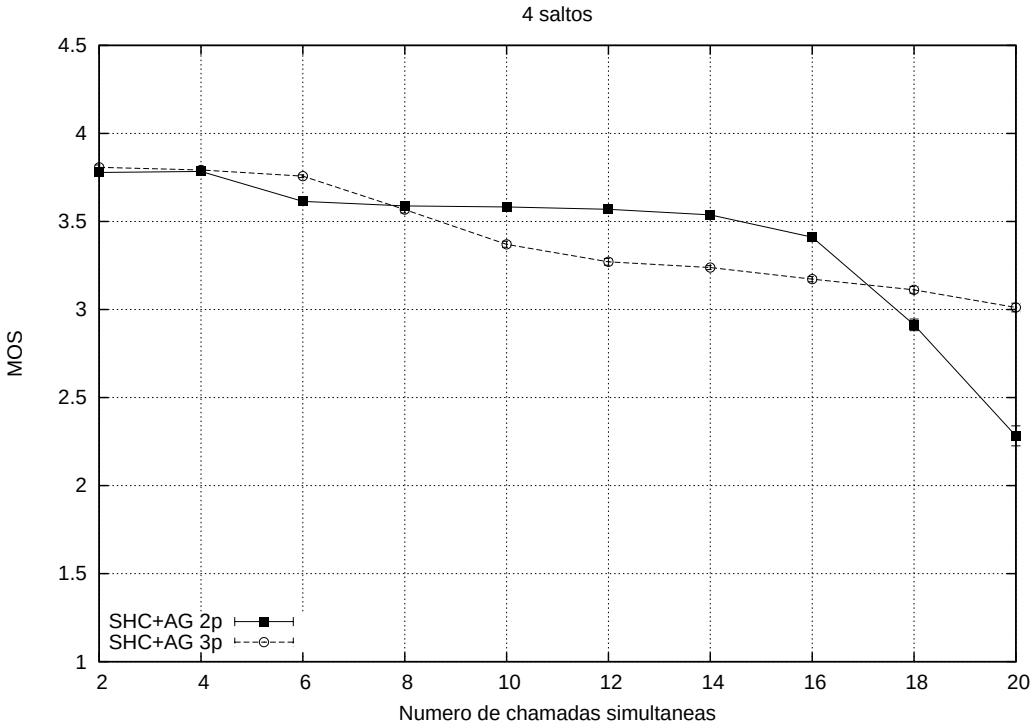

Figura A.11: MOS para chamadas de 2 saltos com graus de agregação 2 e 3.

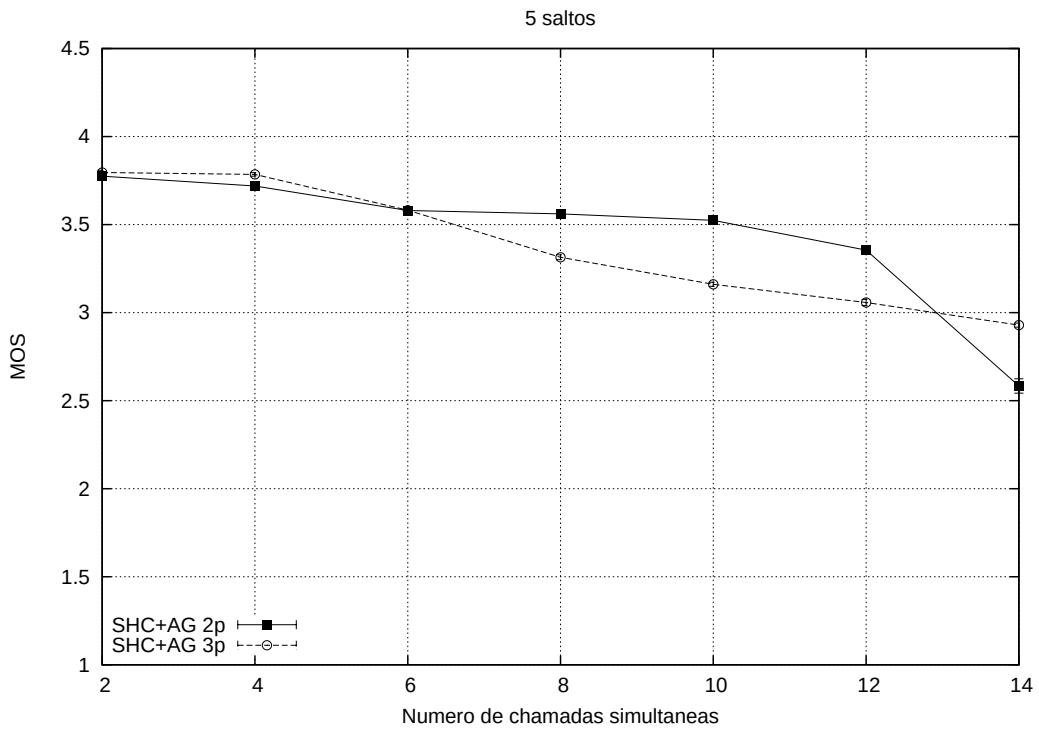

Figura A.12: MOS para chamadas de 2 saltos com graus de agregação 2 e 3.

Com base nos resultados mostrados aqui, podemos afirmar que o grau de agregação mais apropriado neste caso é o grau 2, utilizado inicialmente, já que o uso do grau 3 impõe um atraso cujo impacto pôde ser percebido no valor do MOS, fazendo com que o mesmo ficasse abaixo do limite adequado na maioria dos experimentos realizados.

A.2 Apenas agregação

A abordagem apresentada neste trabalho sugere o uso de compressão de cabeçalhos em conjunto com a agregação de pacotes para diminuir o tamanho dos cabeçalhos. No entanto, devido à grande diferença dos resultados obtidos através dessa abordagem, surge a necessidade de identificar que procedimento foi o responsável direto pelo comportamento apresentado.

Portanto, nesta seção são apresentados experimentos realizados apenas com a agregação de pacotes, já que os experimentos mostrados no capítulo 5 já mostram experimentos realizados apenas com a compressão de cabeçalhos (algoritmo de compressão

estática).

Foram realizados experimentos sobre o cenário linear, com 2, 3, 4 e 5 saltos. Os gráficos apresentados mostram perda de pacotes, atraso na rede e MOS das chamadas realizadas apenas com compressão, em comparação com os resultados obtidos pela abordagem de compressão de cabeçalhos e agregação de pacotes.

A.2.1 Perda de pacotes

Como é possível observar, o comportamento apresentado nos quatro gráficos se repete. Os valores de perda para as menores quantidades de chamadas simultâneas não apresentam diferença entre as chamadas realizadas com apenas o procedimento agregação, e as chamadas realizadas com compressão e agregação.

No entanto, com o aumento na quantidade de chamadas simultâneas, podemos ver que a perda de pacotes cresce mais acentuadamente para as chamadas realizadas com apenas agregação. É possível observar também que a diferença apresentada é maior conforme o aumento no número de saltos sem fio.

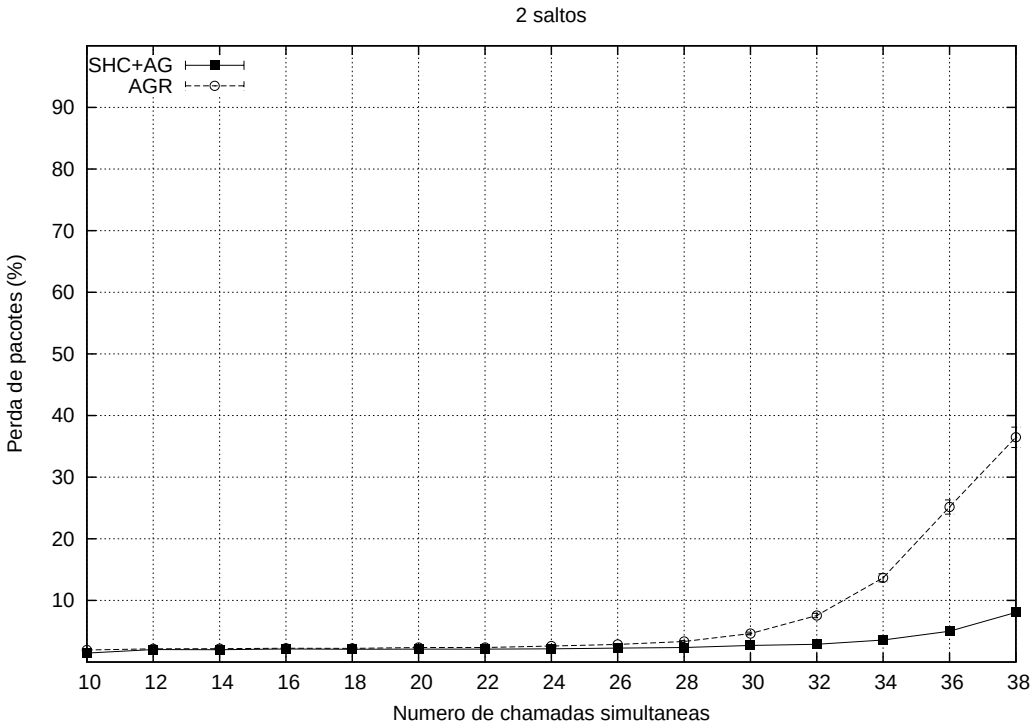

Figura A.13: Perda de pacotes para chamadas de 2 saltos com apenas agregação.

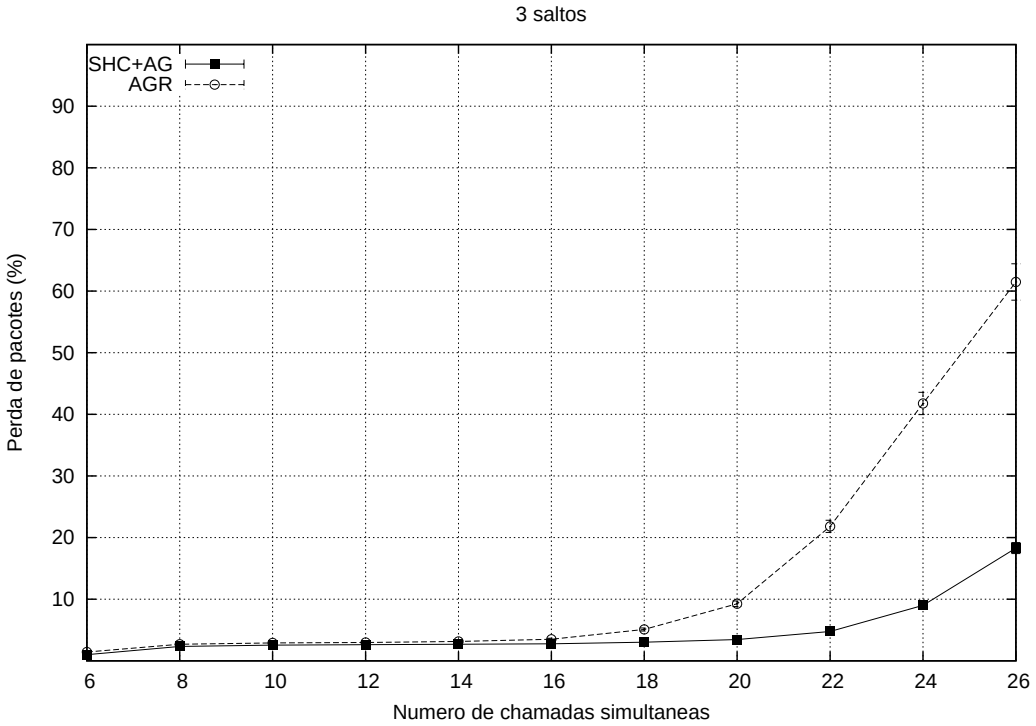

Figura A.14: Perda de pacotes para chamadas de 3 saltos com apenas agregação.

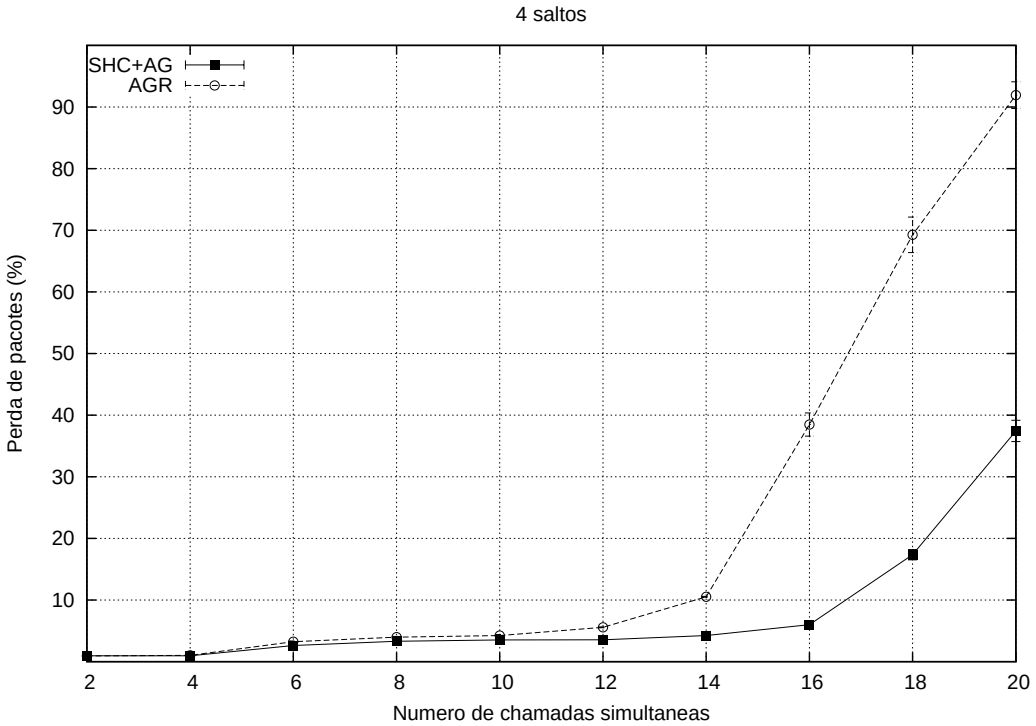

Figura A.15: Perda de pacotes para chamadas de 4 saltos com apenas agregação.

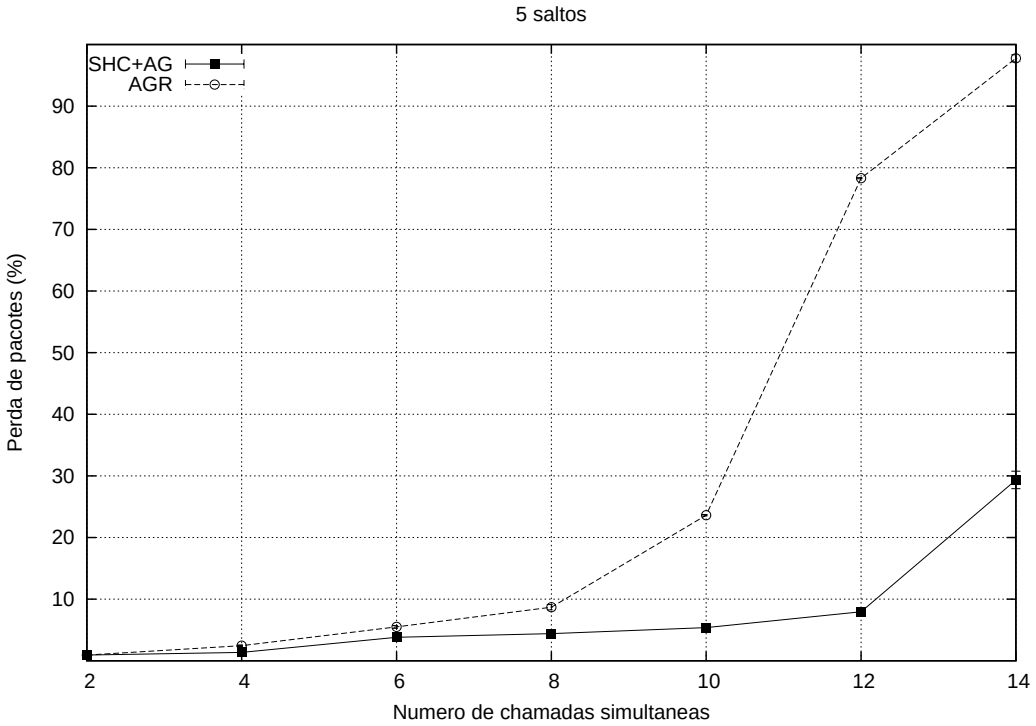

Figura A.16: Perda de pacotes para chamadas de 5 saltos com apenas agregação.

A.2.2 Atraso na rede

Para o atraso na rede, também é possível verificar o mesmo comportamento nos gráficos para as diferentes quantidades de saltos sem fio. O atraso apresentado pelas chamadas realizadas com agregação apenas foi maior, e com o aumento do número de chamadas simultâneas, a diferença em comparação ao atraso apresentado pelas chamadas realizadas com compressão e agregação também aumenta. É possível notar que essa diferença também aumenta conforme o aumento do número de saltos sem fio.

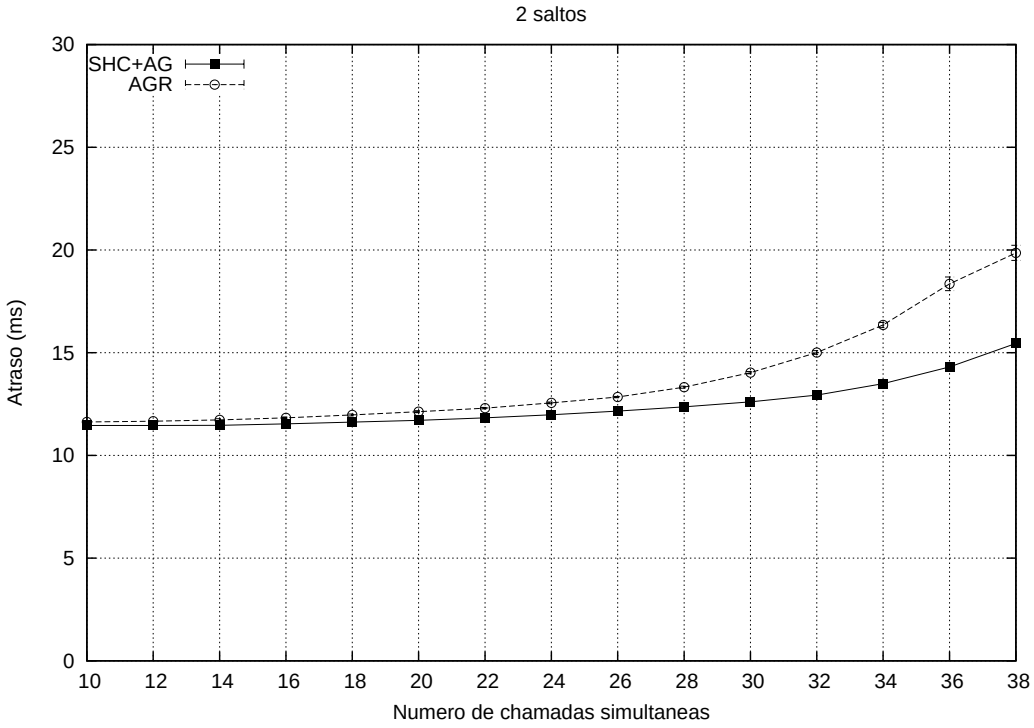

Figura A.17: Atraso na rede para chamadas de 2 saltos com apenas agregação.

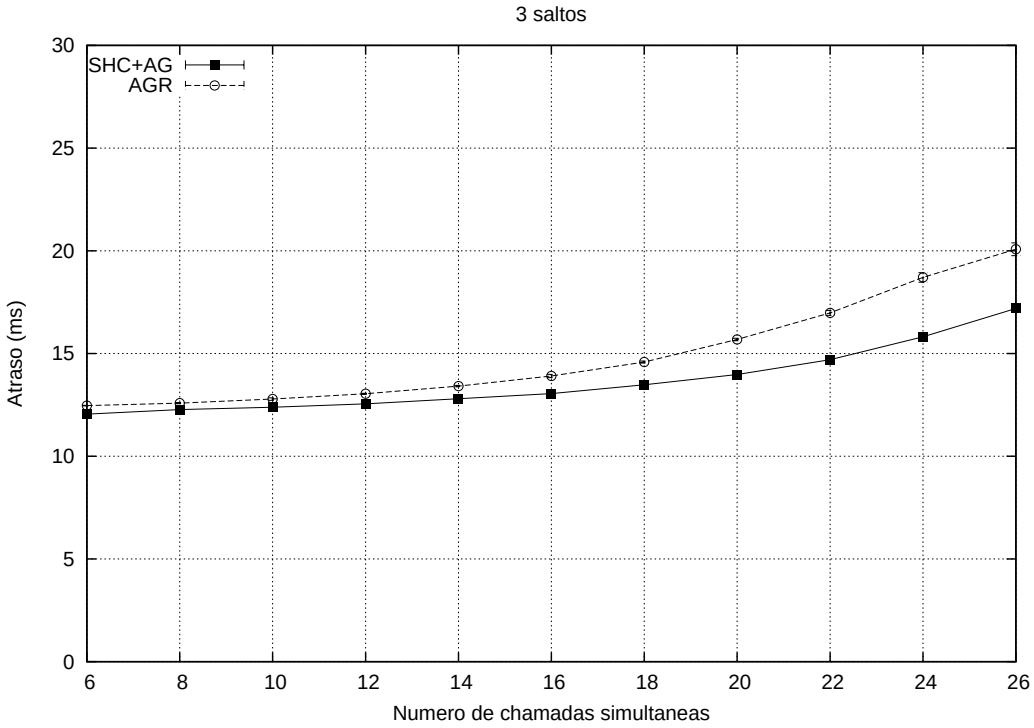

Figura A.18: Atraso na rede para chamadas de 3 saltos com apenas agregação.

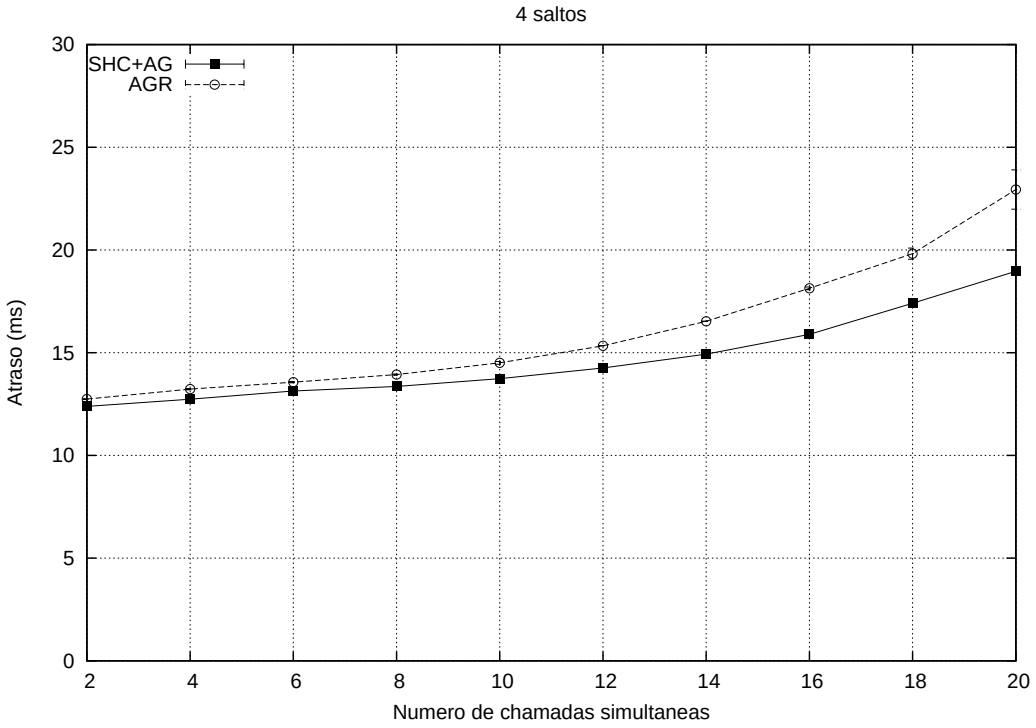

Figura A.19: Atraso na rede para chamadas de 4 saltos com apenas agregação.

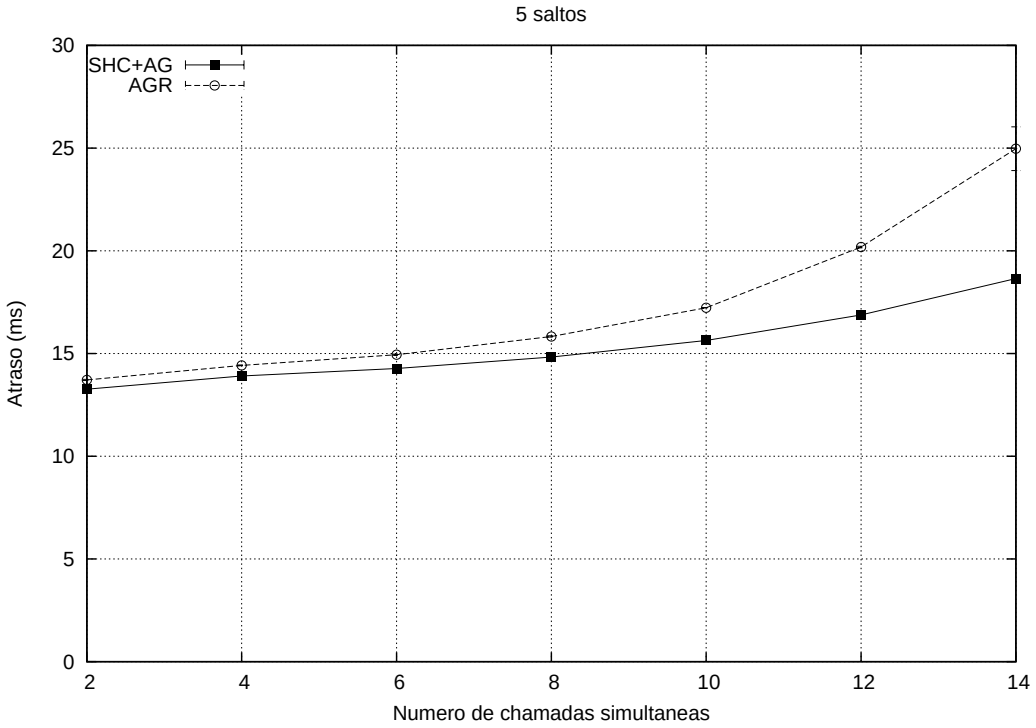

Figura A.20: Atraso na rede para chamadas de 5 saltos com apenas agregação.

A.2.3 MOS

Conforme pode ser observado nos gráficos abaixo, o MOS apresentado pelas chamadas realizadas com compressão e agregação foi maior que o MOS apresentado pelas chamadas realizadas com agregação apenas na maioria dos casos. Em alguns casos essa diferença pode ser considerada insignificante.

Porém, para as maiores quantidades de chamadas simultâneas, o MOS apresentado pelas chamadas realizadas com compressão e agregação foi maior que aquele apresentado pelas chamadas com agregação apenas.

Esse comportamento reflete o mesmo apresentado pelas métricas de perda de pacotes e atraso na rede. A justificativa para esse comportamento, apresentado pelas 3 métricas, se deve ao fato de que a compressão diminui o tamanho dos cabeçalhos, diminuindo consequentemente o tamanho total dos pacotes enviados a rede, o que reflete de maneira positiva na perda de pacotes e no atraso, e portanto, reflete também no MOS.

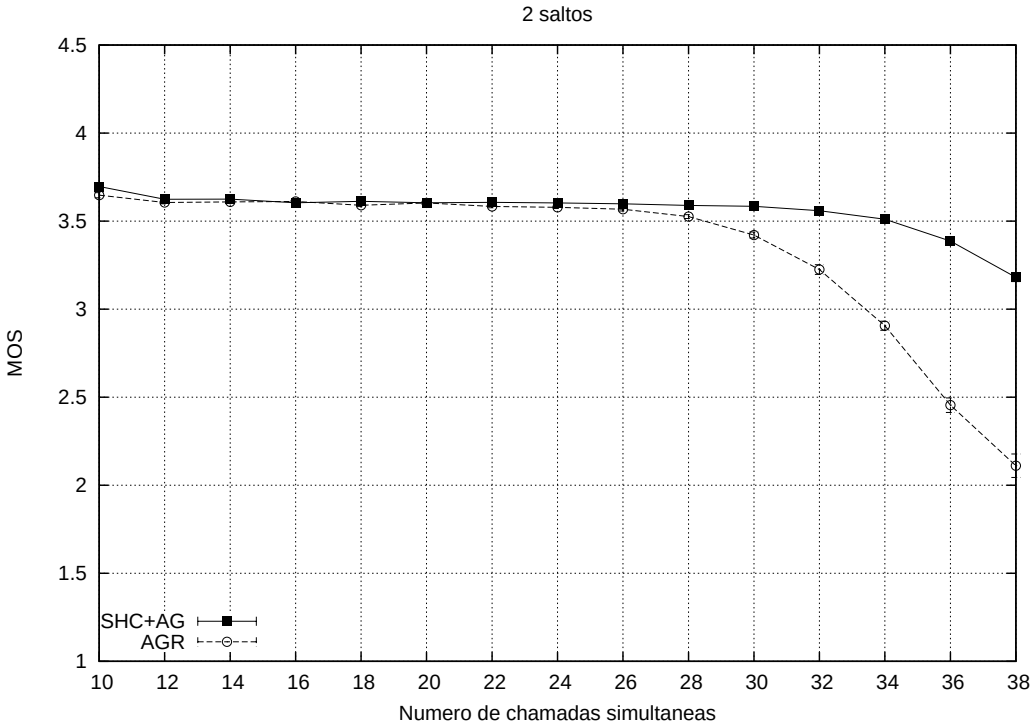

Figura A.21: MOS para chamadas de 2 saltos com apenas agregação.

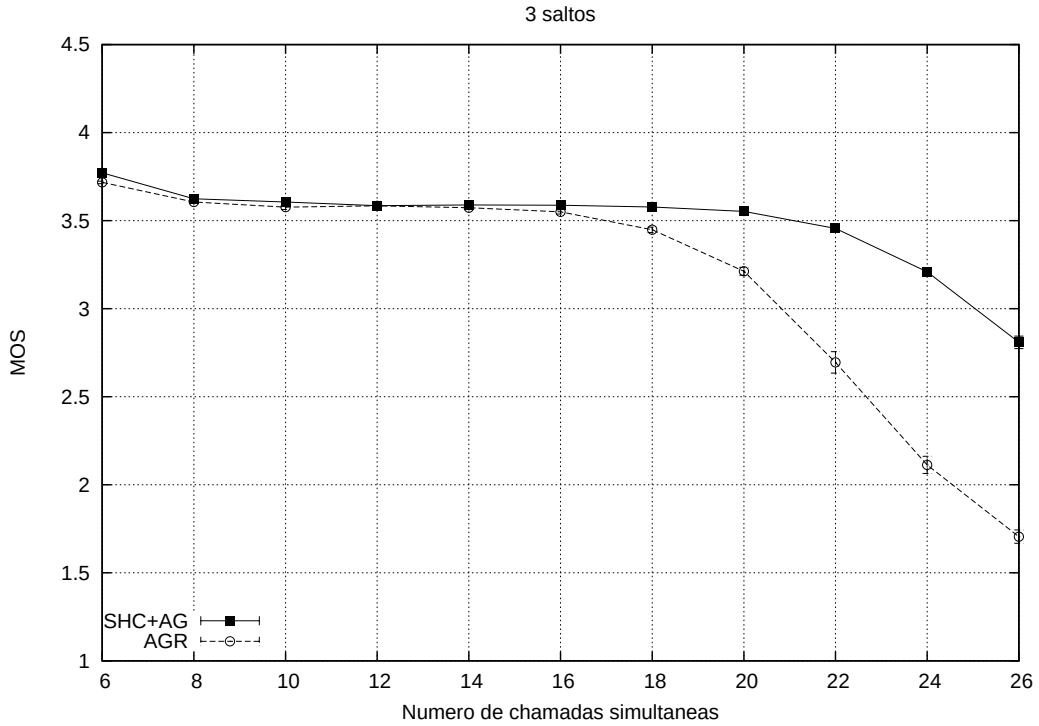

Figura A.22: MOS para chamadas de 3 saltos com apenas agregação.

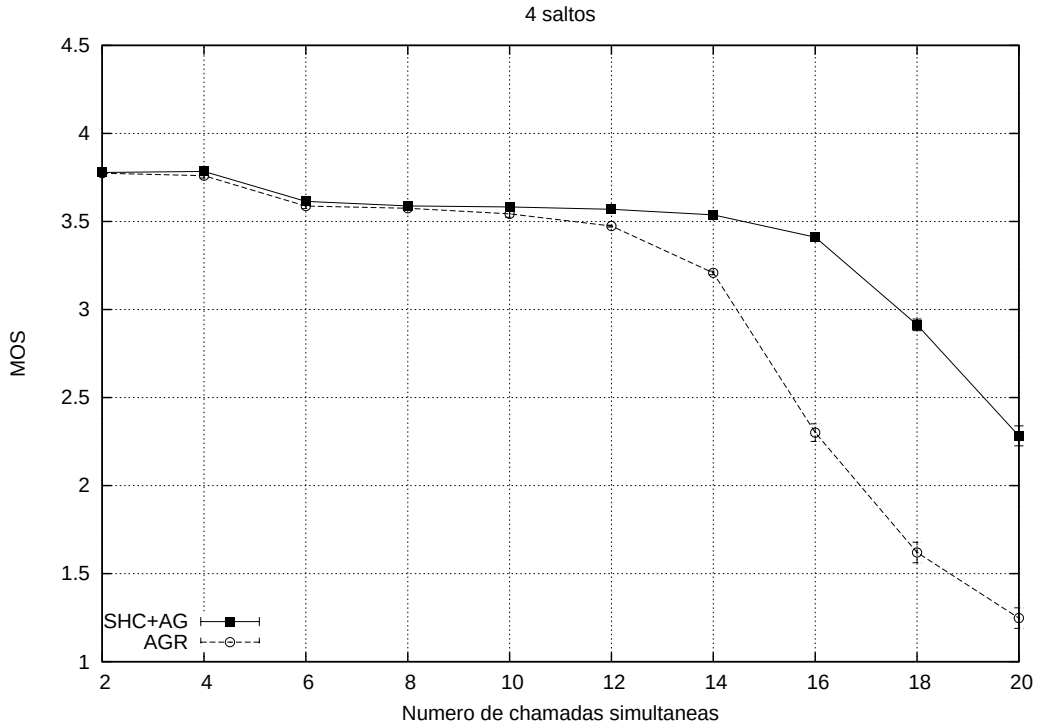

Figura A.23: MOS para chamadas de 4 saltos com apenas agregação.

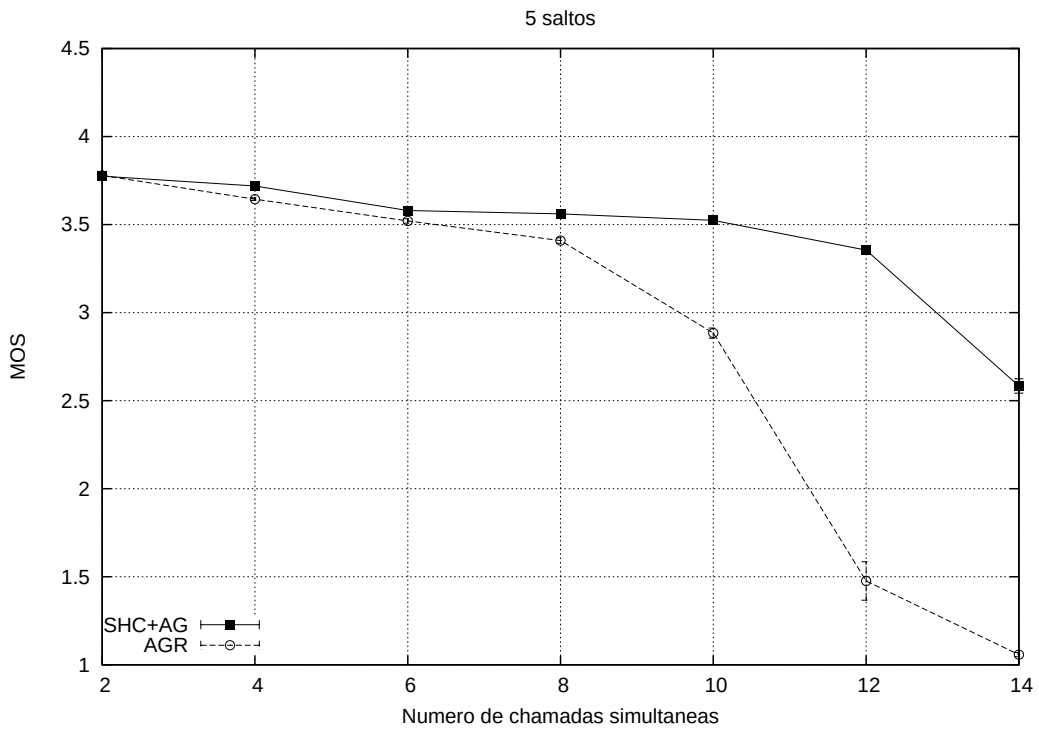

Figura A.24: MOS para chamadas de 5 saltos com apenas agregação.

A.3 Compressão e agregação X carga útil maior

Como a abordagem de compressão de cabeçalhos e agregação de pacotes proposta nesta dissertação se propõe a diminuir o *overhead* de cabeçalhos nos pacotes, o processo de agregação faz com que a quantidade de carga útil por pacote, após o processo, seja maior, já que pacotes consecutivos são agrupados, incluindo seus cabeçalhos e carga útil. Portanto, há de se questionar por que não fazer uso simplesmente da configuração dos *codecs* para gerar pacotes com carga útil de tamanho maior, eliminando assim a necessidade do processo de agregação.

Podemos justificar o uso da abordagem sugerida neste trabalho pelo fato de os processos de compressão de cabeçalhos e agregação de pacotes serem tomados nos roteadores da rede, o que faz com que o oferecimento do serviço seja inteiramente de responsabilidade da rede, não dependendo da configuração dos *codecs*, que usualmente se localizam nos sistemas finais. Além disto, a compressão de cabeçalhos aplicada ofe-

rece um ganho no tamanho do pacote final, ou seja, os pacotes gerados pela compressão e agregação são menores dos que os pacotes com carga útil maior.

Nesta seção são apresentados experimentos realizados sem compressão ou agregação de pacotes, utilizando apenas pacotes com carga útil maior, neste caso 2 vezes maior (40ms), para ser comparada aos experimentos realizados e apresentados neste trabalho (Capítulo 5), que mostrou os resultados para chamadas com compressão e agregação de pacotes de grau 2.

Foram realizados experimentos sobre o cenário linear, com 2, 3, 4 e 5 saltos, e as métricas utilizadas foram perda de pacotes, atraso na rede e MOS.

A.3.1 Perda de pacotes

Como é possível observar nos quatro gráficos, que apresentaram comportamento parecido, a perda de pacotes para as chamadas realizadas sem compressão ou agregação, apenas com a carga útil maior, mostraram os menores valores de perda de pacotes para as menores quantidades de chamadas simultâneas. Esse comportamento se dá pelo fato de que para as chamadas realizadas com agregação, a perda de um único pacote, para a aplicação VoIP representa a perda de dois pacotes.

No entanto, é possível observar que para as maiores quantidades de chamadas simultâneas, a perda de pacotes é menor para as chamadas realizadas com a abordagem proposta. Isso se dá pelo fato de que o tamanho dos pacotes gerados pela abordagem é menor, o que faz diferença para a perda de pacotes em ambientes mais congestionados.

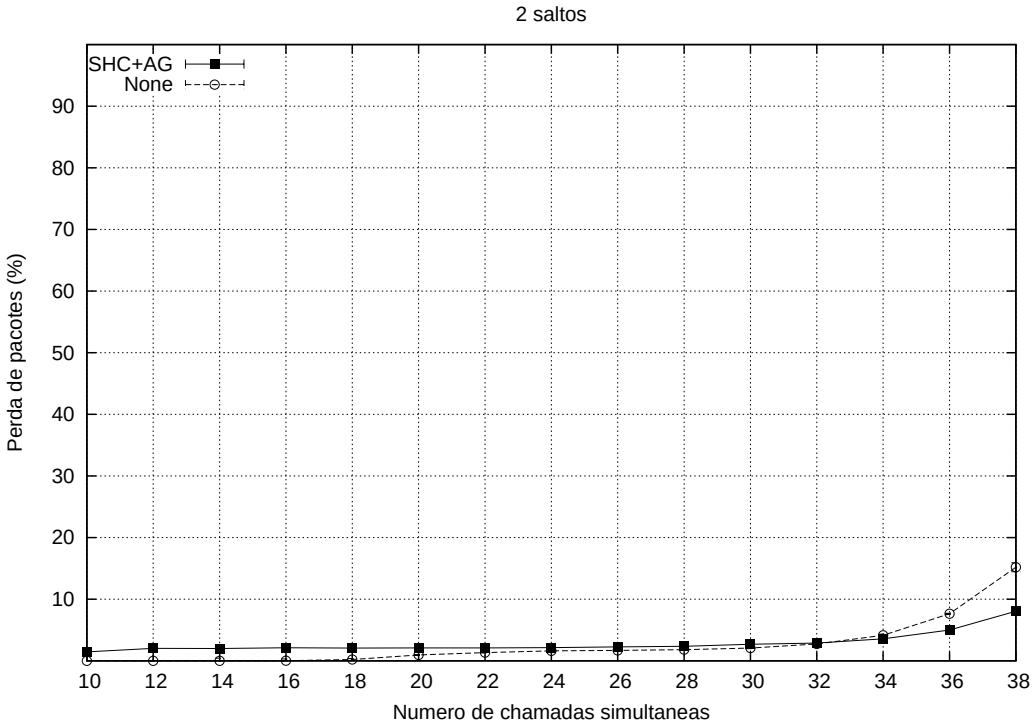

Figura A.25: Perda de pacotes para chamadas de 2 saltos com carga útil de 40ms.

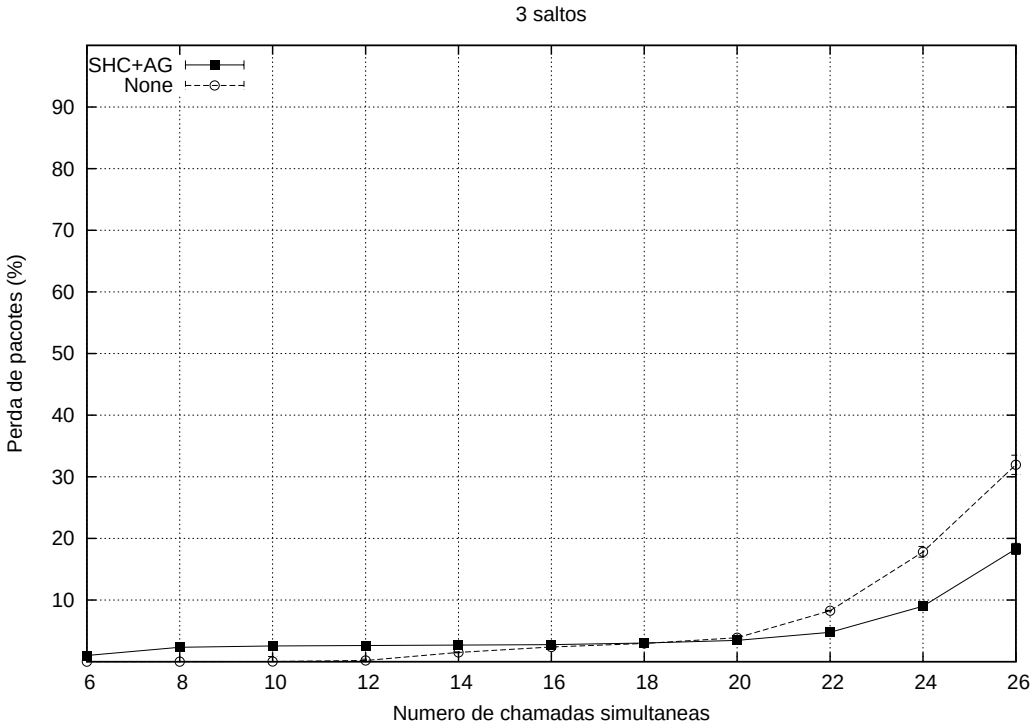

Figura A.26: Perda de pacotes para chamadas de 3 saltos com carga útil de 40ms.

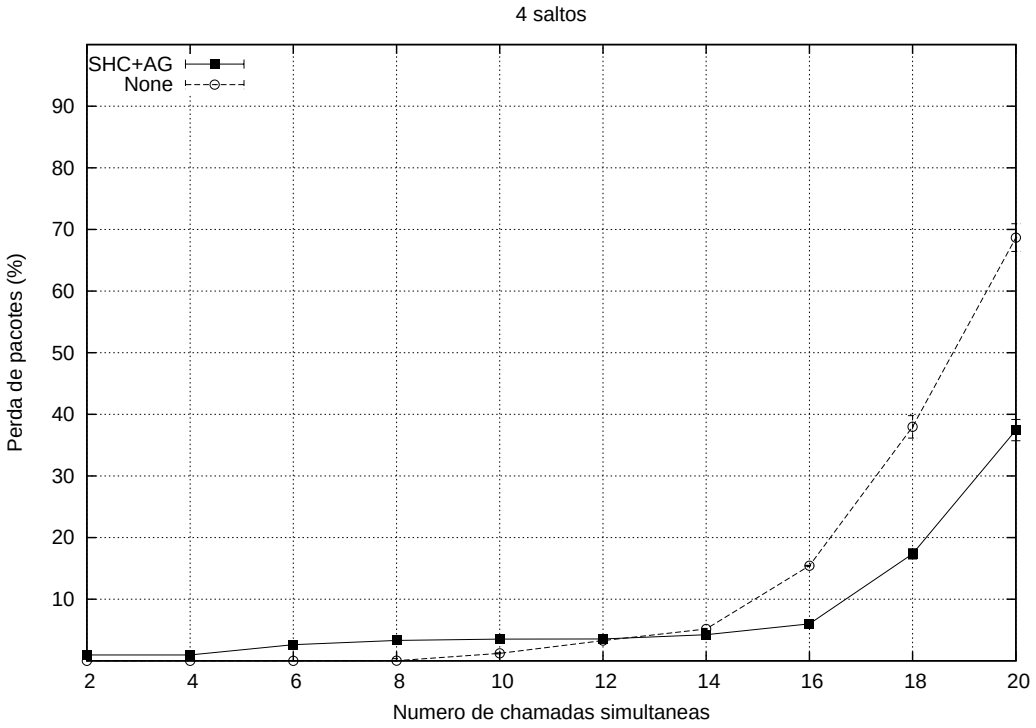

Figura A.27: Perda de pacotes para chamadas de 4 saltos com carga útil de 40ms.

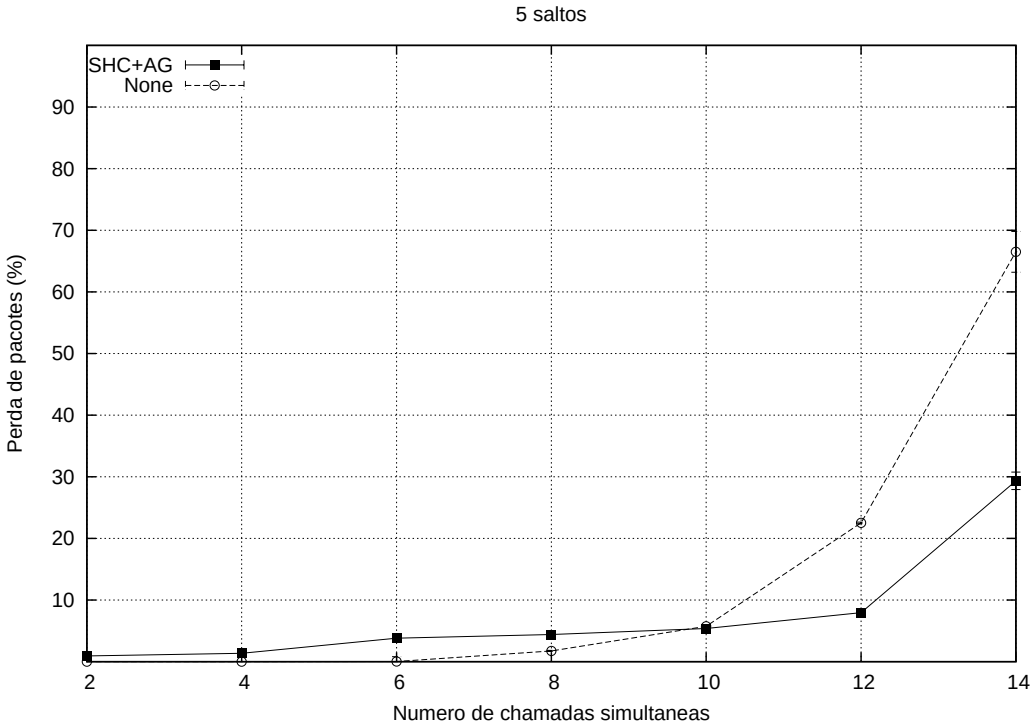

Figura A.28: Perda de pacotes para chamadas de 5 saltos com carga útil de 40ms.

A.3.2 Atraso na rede

As figuras a seguir mostram os valores para atrasos na rede apresentados pelas chamadas realizadas com a abordagem proposta de compressão e agregação, e as chamadas realizadas sem qualquer mecanismo extra, apenas com a carga útil de tamanho maior.

É possível observar que o comportamento se repete para os quatro gráficos. Como esperado, as chamadas realizadas com a abordagem proposta apresentaram atraso maior em todos os casos. Esse atraso é atribuído ao processo de agregação de pacotes, que impõe o atraso de enfileiramento, que é o tempo que um pacote precisa esperar pela formação de outro pacote para ser agregado.

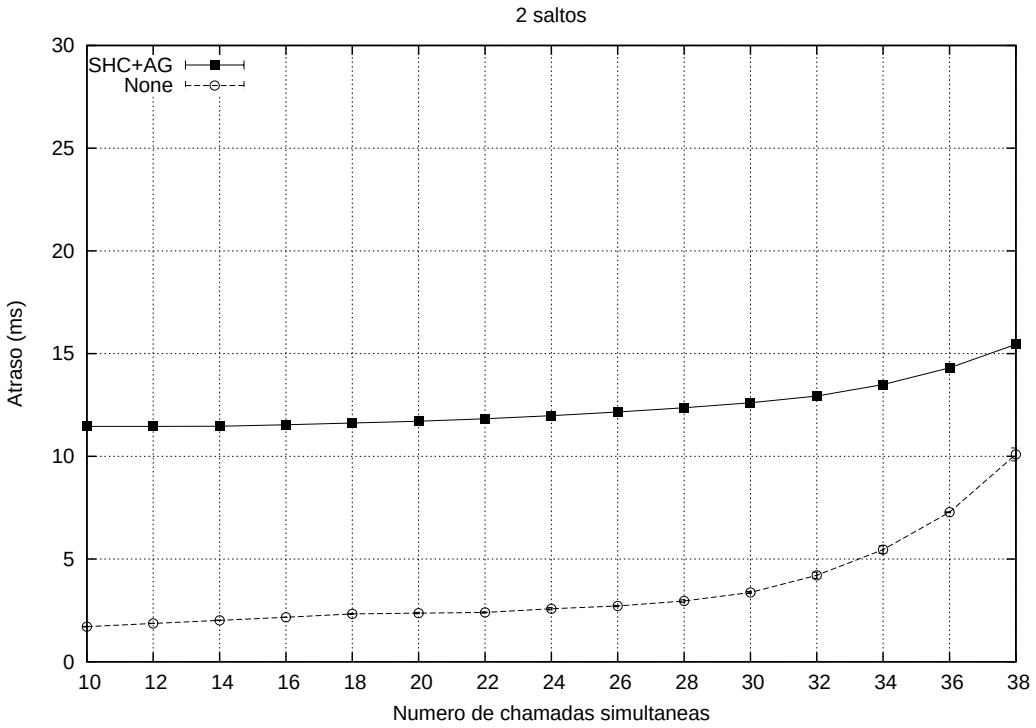

Figura A.29: Atraso na rede para chamadas de 2 saltos com carga útil de 40ms.

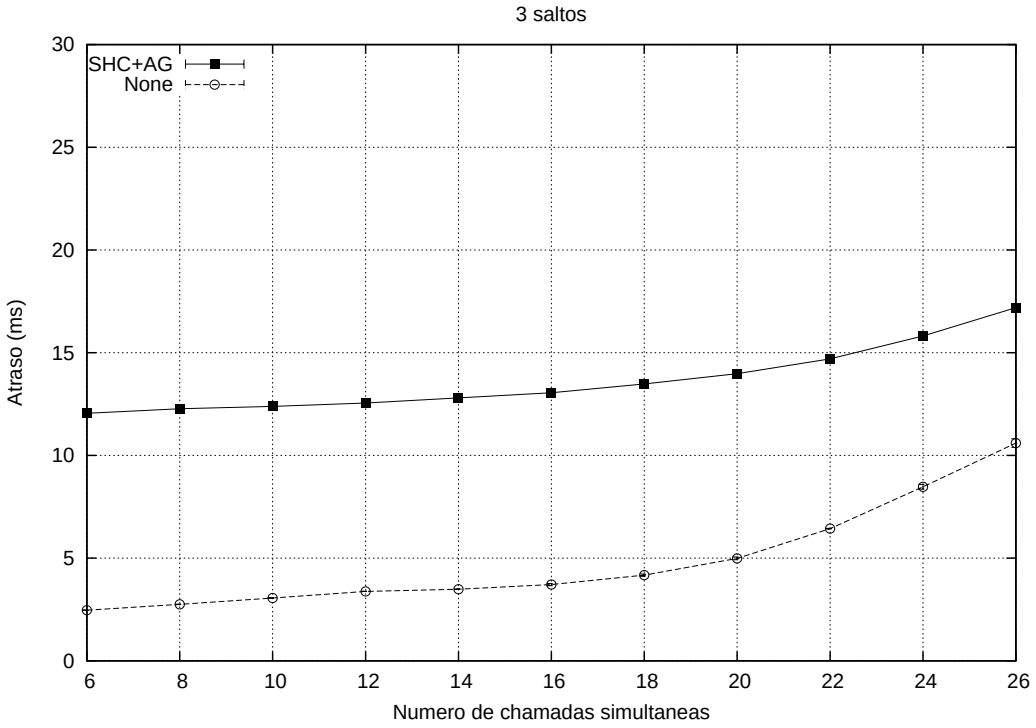

Figura A.30: Atraso na rede para chamadas de 3 saltos com carga útil de 40ms.

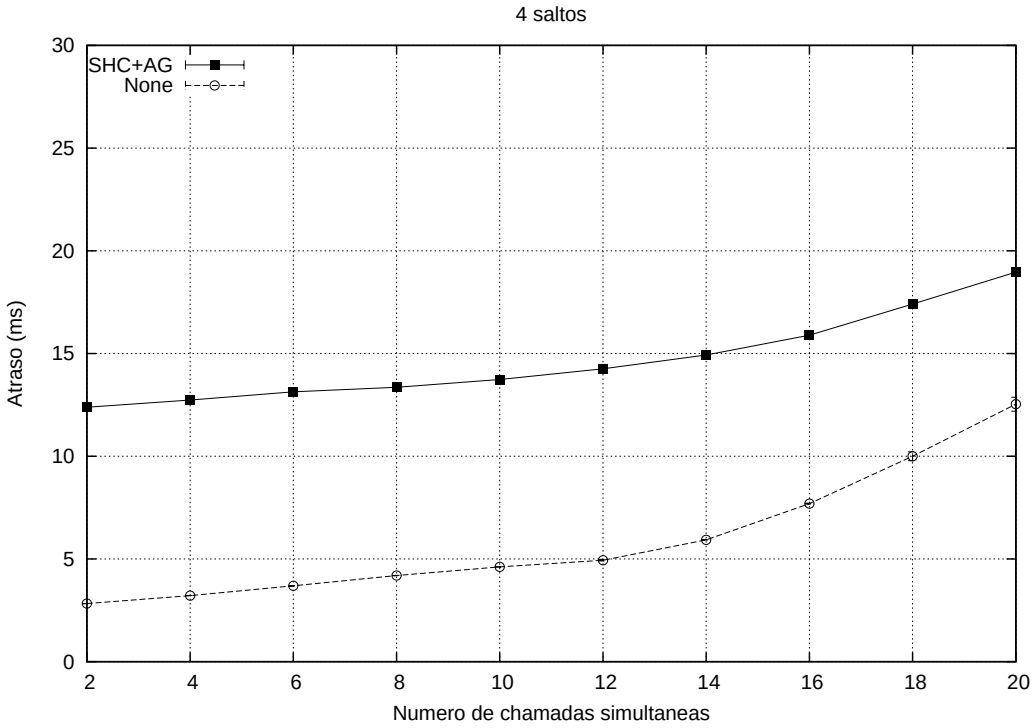

Figura A.31: Atraso na rede para chamadas de 4 saltos com carga útil de 40ms.

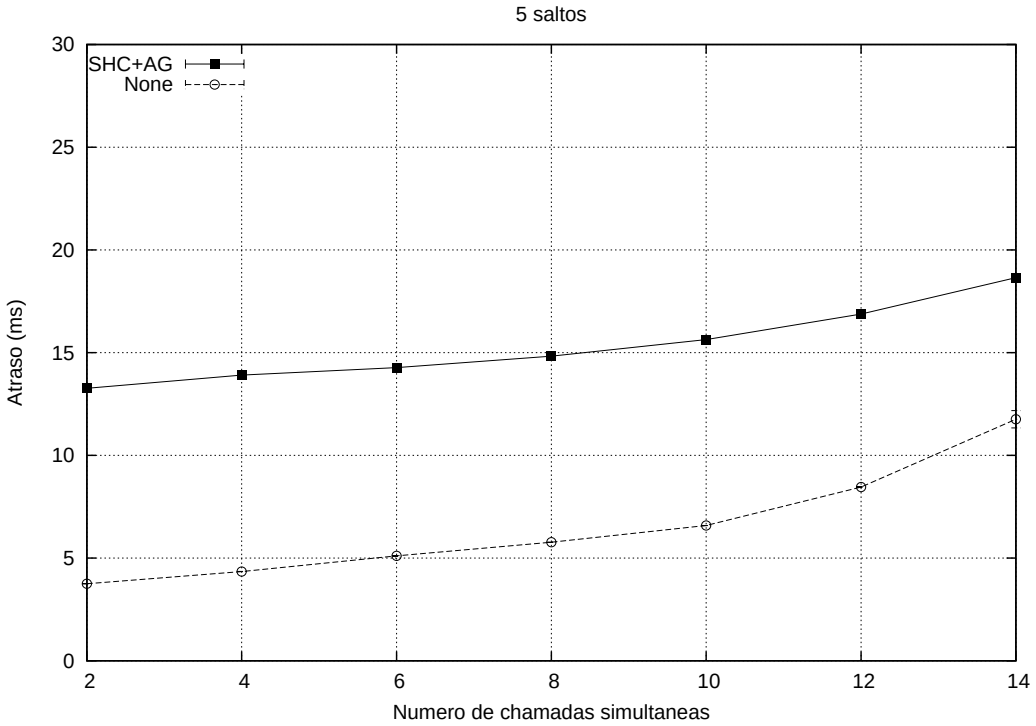

Figura A.32: Atraso na rede para chamadas de 5 saltos com carga útil de 40ms.

A.3.3 MOS

Os gráficos a seguir mostram os valores de MOS calculados para as mesmas chamadas cujos valores de perda e atraso foram mostrados nas subseções anteriores.

Como é possível observar nos gráficos, o comportamento permanece semelhante independente da quantidade de saltos sem fio. As chamadas realizadas sem qualquer mecanismo de compressão ou agregação, utilizando apenas carga útil dos pacotes maior, mostraram os maiores valores de MOS na maioria dos experimentos. Este comportamento era esperado, dado o comportamento apresentado pelas métricas de perda de pacotes e atraso na rede.

Já para as maiores quantidades de chamadas simultâneas, as chamadas realizadas com a abordagem de compressão e agregação apresentou os maiores valores. Esse comportamento, melhor observado nos gráficos para 4 e 5 saltos sem fio, se deu pelo fato de essas chamadas terem apresentado menor perda em comparação às chamadas realizadas apenas com a carga útil maior.

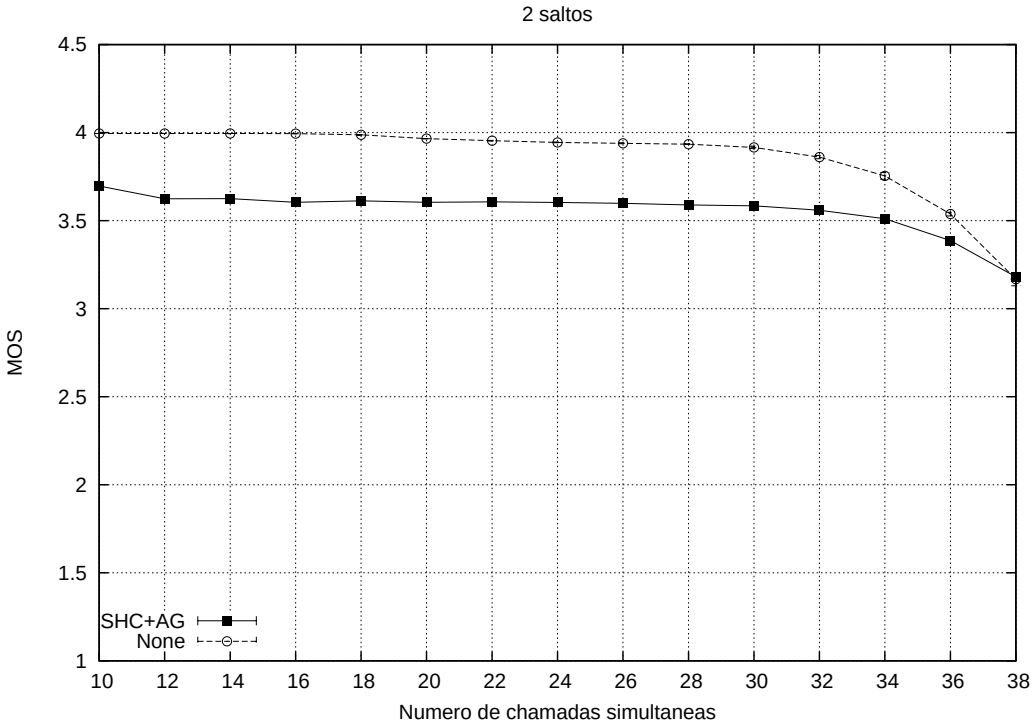

Figura A.33: MOS para chamadas de 2 saltos com carga útil de 40ms.

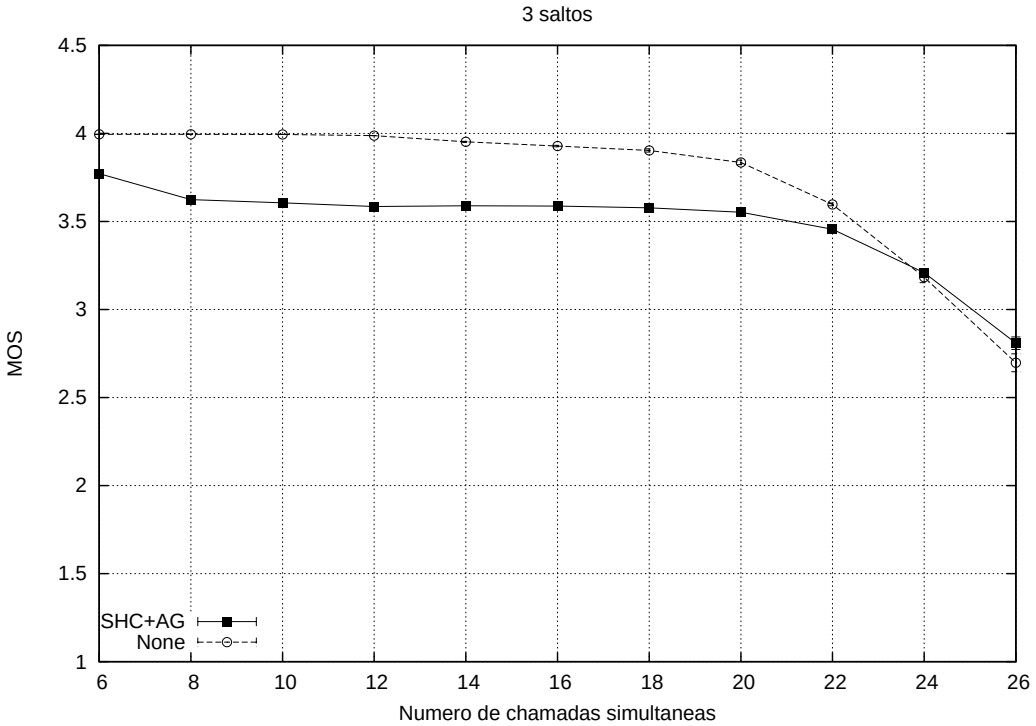

Figura A.34: MOS para chamadas de 3 saltos com carga útil de 40ms.

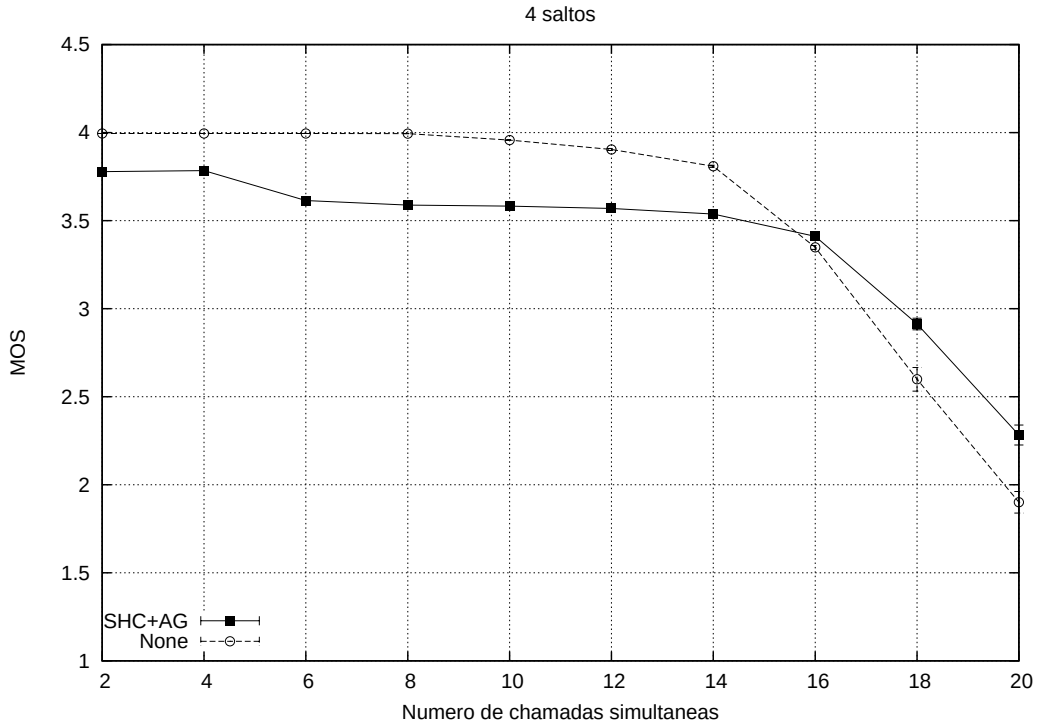

Figura A.35: MOS para chamadas de 4 saltos com carga útil de 40ms.

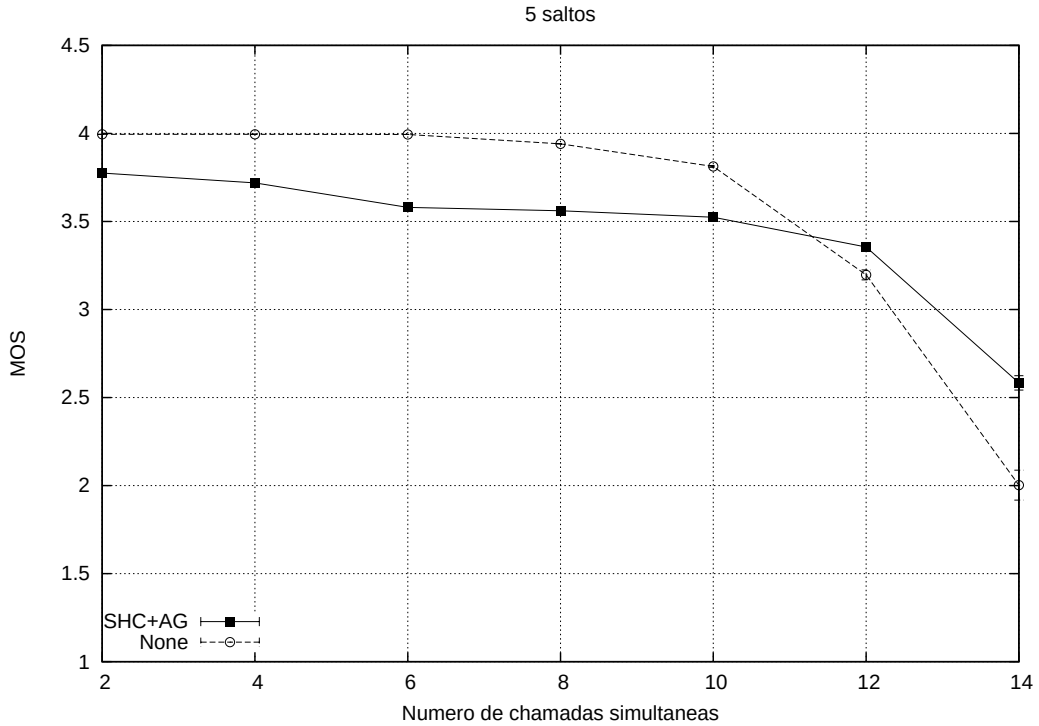

Figura A.36: MOS para chamadas de 5 saltos com carga útil de 40ms.

Apêndice B

Patches e *scripts* de simulação

B.1 *Patch* para NS-2 versão 2.29

O *patch* `patch-ns2.29-hdrcmp-pktagg-giordanna` disponível na página do Grupo de Redes de Computadores e Multimídia (GRCM - http://grcm.dcc.ufam.edu.br) aplica as seguintes alterações, dentre outras:

- Adiciona o módulo do protocolo de roteamento OLSR;
- Adiciona o módulo do Akaroa-2;
- Adiciona o módulo de cálculo do MOS;
- Adiciona o módulo MPLS para nós sem fio;
- Adiciona o módulo para compressão de cabeçalhos e agregação de pacotes.

Para aplicar o *patch*, faça o *download* do código do NS-2 versão 2.29 (http://sourceforge.net/projects/nsnam), faça o *download* do *patch* na página do GRCM, e digite no terminal:

```
$ tar xzf ns-allinone-2.29.tar.gz
$ patch -p0 < patch-ns2.29-hdrcmp-pktagg-giordanna
```

B.2 *Scripts*

B.2.1 Arquivo voice.tcl

O código abaixo deve ser copiado e colado em um arquivo de texto chamado `voice.tcl`, que deve estar no mesmo diretório do *script* de simulação.

```
#
# 10/21/1998  C-N. Chuah
# This file contains: on-off Markov model of voice sources
#                     random variables (uniforms & exponentials)
#                     CBR source

set uni [new RandomVariable/Uniform]
set expoLI [new RandomVariable/Exponential]
set expoMI [new RandomVariable/Exponential]
$uni set min_ 0
$uni set max_ 0

proc create_audio_src { nodeSrc audioDest pktSize numSource null1 udp0 cbr } {
    global ns_ framelength
    $ns_ attach-agent $nodeSrc $udp0

    $udp0 set packetSize_ $pktSize

    $cbr set packetSize_ $pktSize
    $cbr attach-agent $udp0
    $cbr set interval_ $framelength
    $cbr set class_ $numSource

    $ns_ attach-agent $audioDest $null1
    $ns_ connect $udp0 $null1
}

proc create_audio_model { numSource audioStartTime audioStopTime nodeSrc nodeDest pktSize startInSteadyState loss_agent udp0 cbr } {
    #global ns_ audioLambdaInv audioMuInv audioStartTime audioStopTime uni expoLI expoMI
    global ns_ audioLambdaInv audioMuInv uni expoLI expoMI

    set onProb [expr $audioMuInv / ($audioMuInv + $audioLambdaInv)]
    create_audio_src $nodeSrc $nodeDest $pktSize $numSource $loss_agent $udp0 $cbr ; #adicionado agente null1
    $expoMI set avg_ $audioMuInv
    $expoLI set avg_ $audioLambdaInv

    if {$startInSteadyState} {
       if {[$uni value] <= $onProb} {
           $ns_ at $audioStartTime "start_audio_source $cbr $audioStopTime"
       } else {
           set time [$expoLI value]
           puts "time --> $time"
           $ns_ at [expr $audioStartTime + $time] "start_audio_source $cbr $audioStopTime"
       }
```

```
    } else {
       $ns_ at $audioStartTime "start_audio_source $cbr $audioStopTime "
    }
    $ns_ at $audioStopTime "$cbr stop"
    $ns_ at $audioStopTime "$nodeSrc reset"
    $ns_ at $audioStopTime "$nodeDest reset"
}

proc start_audio_source { audioSource audioStopTime } {
    #global ns_ expoMI audioStopTime
    global ns_ expoMI
    $audioSource start
    set time [$expoMI value]
    if { [expr [$ns_ now] + $time] < $audioStopTime } {
        $ns_ at [expr [$ns_ now] + $time] "stop_audio_source $audioSource $audioStopTime"
    }
}

proc stop_audio_source { audioSource audioStopTime } {
    #global ns_ expoLI audioStopTime
    global ns_ expoLI
    $audioSource stop
    set time [$expoLI value]

    if { [expr [$ns_ now] + $time] < $audioStopTime } {
        $ns_ at [expr [$ns_ now] + $time] "start_audio_source $audioSource $audioStopTime"
    }
    # now toggle class from even to odd (or v.v.)
    set class [$audioSource set class_]
}

# note also that we account for the average percentage of time an
# audio source is "on" to determine how many will fit within th
# desired bandwidth...
# (if each source stays on with average time 1/mu, and off w/ avg
# time 1/lambda, then the fraction of time it's on is
# (1/mu)/(1/lambda + 1/mu)
# the fraction of time audio sources are really on:

set fractionOn [expr $audioMuInv / ($audioLambdaInv + $audioMuInv)]

if {$intBW != 0} {
#global color
    # compute the average inter-packet rate:
    set intPktRate [expr (8.0 * $intPktSize) / ($intBW * 1000)]
    # make the interfering traffic agent, set the class and packet size, attach
    # it to a null sink destination, and start it at the appropriate time:
    set intTraffic [new Agent/Message/Poisson $intPktRate]
    $intTraffic set class_  0
    $intTraffic set packetSize_ $intPktSize
    $ns_ attach-agent $sourceNode $intTraffic
    set intDest [new Agent/Null]
```

```
    $ns_ attach-agent $destNode $intDest
    $ns_ connect $intTraffic $intDest
    $ns_ at $intStartTime "$intTraffic start"
}
```

B.2.2 *Script* para cenário linear

O *script* abaixo foi utilizado para realizar os experimentos com chamadas sobre o cenário linear. Deve ser executado da seguinte forma:

```
$ ns script.tcl 0 <quantidade_de_saltos> <quantidade_de_chamadas_simultaneas>
```

```
# ======================================================================
# Definicoes das opcoes
# ======================================================================
set opt(chan)           Channel/WirelessChannel ;# channel type
set opt(prop)           Propagation/Shadowing ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy ;# network interface type
set opt(mac)            Mac/802_11 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue ;# interface queue type
set opt(ll)             LL ;# link layer type
set opt(ant)            Antenna/OmniAntenna ;# antenna model
set opt(ifqlen)         50 ;# max packet in ifq
set opt(nn)             7 ;# number of mobilenodes
set opt(adhocRouting)   OLSR ;# routing protocol
set opt(hdrcomp)        ALGB ;# compression algorithm - Options: ROHC, LECC (static compression),
                             ;# ALGB (static compression + packet aggregation)
set opt(x)              650 ;# x coordinate of topology
set opt(y)              650 ;# y coordinate of topology
set start               20 ;# instante de inicio da primeira chamada
set opt(tr)            /dev/null
set opt(cp)            ""
set opt(sc)            ""

ErrorModel80211 noise1  -104 ;# 1Mbps datarate thermal noise
ErrorModel80211 noise2  -101 ;# 2Mbps datarate thermal noise
ErrorModel80211 noise55 -97 ;# 5.5Mbps datarate thermal noise
ErrorModel80211 noise11 -92 ;# 11Mbps datarate thermal noise
ErrorModel80211 shortpreamble 1 ;# toggle 802.11 short preamble on/off
#ErrorModel80211 LoadBerSnrFile ber_snr_intersil.txt ;# use Intersil BER/SNR empirical curves
ErrorModel80211 LoadBerSnrFile ber_snr_choi.txt ;# use theoretical curves from del Prado
;# Pavon and Choi, "Link AdaptationStrategy
;# for IEEE 802.11 WLAN via Received Signal
;# Strength Measurement"

# ======================================================================
# Configuracao das Antenas Ganhos das antenas > irradiacao e recepcao
# ======================================================================
# Ganho Antena transmissora. Vivek: 13dBi Omniantenna total EIRP of 36dBm,
Antenna/OmniAntenna set X_ 0
```

```
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 3
Antenna/OmniAntenna set Gt_ 1.0;
Antenna/OmniAntenna set Gr_ 1.0;

# ======================================================================
# Interface wireless
# ======================================================================
Phy/WirelessPhy set CPThresh_   10.0 ;# capture threshold
Phy/WirelessPhy set CSThresh_   3.1622777e-14 ;# Carrier Sensing threshold
Phy/WirelessPhy set RXThresh_   3.1622777e-13 ;# receiver signal threshold
Phy/WirelessPhy set bandwidth_  11Mb ;# channel bandwidth (UNUSED!)
Phy/WirelessPhy set Pt_         0.031622777 ;# transmitter signal power (Watt)
Phy/WirelessPhy set freq_       2.472e9 ;# channel frequency (Hz)
Phy/WirelessPhy set L_          1.0

Propagation/Shadowing set pathlossExp_  3.5 ;# path loss exponent
Propagation/Shadowing set std_db_       4.0 ;# standard deviation
Propagation/Shadowing set dist0_        1.0 ;# reference small distance
Propagation/Shadowing set seed_         0

# ======================================================================
# Parametros da MAC
# ======================================================================
Mac/802_11 set CWMin_                 31
Mac/802_11 set CWMax_                 1023
Mac/802_11 set SlotTime_              0.000020 ;# 20us
Mac/802_11 set SIFS_                  0.000010 ;# 10us
Mac/802_11 set PreambleLength_        144 ;# 144 bit
Mac/802_11 set ShortPreambleLength_   72 ;# 72 bit
Mac/802_11 set PreambleDataRate_      1.0e6 ;# 1Mbps
Mac/802_11 set PLCPHeaderLength_      48 ;# 48 bits
Mac/802_11 set PLCPDataRate_          1.0e6 ;# 1Mbps
Mac/802_11 set ShortPLCPDataRate_     2.0e6 ;# 2Mbps
Mac/802_11 set RTSThreshold_          3000 ;# bytes
Mac/802_11 set ShortRetryLimit_       7 ;# retransmissions
Mac/802_11 set LongRetryLimit_        4 ;# retransmissions
Mac/802_11 set newchipset_            false ;# use new chipset, allowing a more recent
;# packet to be correctly received in place
;# of the first sensed packet
Mac/802_11 set dataRate_ 11Mb ;# 802.11 data transmission rate
Mac/802_11 set basicRate_ 2Mb ;# 802.11 basic transmission rate
Mac/802_11 set aarf_ true ;# 802.11 Auto Rate Fallback

# ======================================================================
# Parametros da camada de Rede: Protocolo de Roteamento OLSR-ETX/ML/MD
# ======================================================================
Agent/OLSR set use_mac_ true
Agent/OLSR set debug_ false
Agent/OLSR set willingness 3
Agent/OLSR set hello_ival_ 2
Agent/OLSR set tc_ival_ 5
```

```
# ======================================================================
# Parametros do compressor de cabeçalhos
# ======================================================================

# If you are not using any header compression scheme, comment all the lines

# for SHC+AGG
Agent/ALGB set total_calls_     [lindex $argv 0]

# for static compression
#Agent/LECC set total_calls_ [lindex $argv 0]

# for ROHC
#Agent/ROHC set large_cid_ true
#Agent/ROHC set cohc_ false ;# set true to use cooperative header compression
#Agent/ROHC set total_calls_     [lindex $argv 0]

# ======================================================================
# Adiciona somente os cabecalhos de interesse
# ======================================================================
remove-all-packet-headers ;# remove todos os cabecalhos do pacote
add-packet-header RTP
add-packet-header IP
add-packet-header OLSR
add-packet-header LL
add-packet-header Mac
add-packet-header LDP
add-packet-header MPLS
add-packet-header ALGB ;# for SHC+AG
#add-packet-header LECC ;# for static compression
#add-packet-header ROHC ;# for ROHC

# ======================================================================
# Criando O Objeto simulador ns_, trace e nam
# ======================================================================
set ns_ [new Simulator]
set ak [new Akaroa]
$ak AkDeclareParameters 5

set tracefd [open $opt(tr) w] ;#From Vivek
$ns_ use-newtrace
$ns_ trace-all $tracefd

# Criacao e configuracao da area
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# ======================================================================
# Criando O Objeto GOD-General Operations Director
# ======================================================================
create-god $opt(nn)
```

```
# ==================================================================
# NAM Configurations
# ==================================================================
# Abre o arquivo out.nam para escrita e rotula-o com nf
#set namtracens_ [open ufam.nam w]

# Escreve dados da simulacao relevantes para o nam no arquivo representado por nf.
#$ns_ namtrace-all-wireless $namtracens_ $opt(x) $opt(y)

# ==================================================================
# Configuracao e Criacao dos nos (eh necessario primeiro configura-los)
# ==================================================================
# Configuracao dos nos
$ns_ node-config -adhocRouting $opt(adhocRouting)\
-headerCompression $opt(hdrcomp)\
-llType $opt(ll) \
-macType $opt(mac) \
-ifqType $opt(ifq) \
-ifqLen $opt(ifqlen) \
-antType $opt(ant) \
-propType $opt(prop) \
-phyType $opt(netif) \
-channel [new $opt(chan)] \
-topoInstance $topo \
-wiredRouting OFF \
-agentTrace OFF \
-routerTrace OFF \
-macTrace OFF \
-toraDebug OFF \
-movementTrace OFF
#From olsr etx. Unica diferenca eh:
# -channelType $opt(chan) \

# Criacao dos nos
for {set i 0} { $i < $opt(nn)} {incr i} {
        set node_($i) [$ns_ mpls-node ]
        $node_($i) random-motion 0 ;# disable random motion
}

# ==================================================================
# Configurando posicionamento dos nos
# ==================================================================
#=========================>
$node_(0) set X_ 0.0
$node_(0) set Y_ 0.0
$node_(0) set Z_ 15.0
#=========================>
$node_(1) set X_ 25.0
$node_(1) set Y_ 0.0
$node_(1) set Z_ 15.0
#=========================>
```

```
$node_(2) set X_ 50.0
$node_(2) set Y_ 0.0
$node_(2) set Z_ 15.0
#=========================>
$node_(3) set X_ 75.0
$node_(3) set Y_ 0.0
$node_(3) set Z_ 15.0
#=========================>
$node_(4) set X_ 100.0
$node_(4) set Y_ 0.0
$node_(4) set Z_ 15.0
#=========================>
$node_(5) set X_ 125.0
$node_(5) set Y_ 0.0
$node_(5) set Z_ 15.0
#=========================>
$node_(6) set X_ 150.0
$node_(6) set Y_ 0.0
$node_(6) set Z_ 15.0

set numComun [lindex $argv 0]
set orig [lindex $argv 1]
set dest [lindex $argv 2]

# ============================================================
# MPLS stuff
# ============================================================

$ns_ enable-control-driven

for {set i 0} {$i < $opt(nn)} {incr i} {
set a $node_($i)
set m [eval $a get-module "MPLS"]
eval set LSRmpls$i $m
$m set enable_reroute_ 0
}

$ns_ at 6.0 "$ns_ configure-ldp-on-all-mpls-nodes"

switch $dest {
2 {
$ns_ at 10.0 "$LSRmpls0 setup-erlsp  2  1_2  1000"
$ns_ at 11.0 "$LSRmpls2 setup-erlsp  0  1_0  2000"
}
3 {
$ns_ at 10.0 "$LSRmpls0 setup-erlsp  3  1_2_3  1000"
$ns_ at 10.0 "$LSRmpls3 setup-erlsp  0  2_1_0  2000"
}
4 {
$ns_ at 10.0 "$LSRmpls0 setup-erlsp  4  1_2_3_4  1000"
$ns_ at 10.0 "$LSRmpls4 setup-erlsp  0  3_2_1_0  2000"
}
```

```
5 {
$ns_ at 10.0 "$LSRmpls0 setup-erlsp  5  1_2_3_4_5  1000"
$ns_ at 10.0 "$LSRmpls5 setup-erlsp  0  4_3_2_1_0  2000"
}
}

# ======================================================================
# Trafego de voz - Modelo ON/OFF
# ======================================================================

# parameters
set opt(chamsimul) 1 ;# numero de chamadas simultaneas
set opt(N) 1 ;# número de nós
set N  $opt(N) ;# numero de chamadas
set duration 60 ;# duracao da chamada em segundos
set interval 2 ;# intervalo fixo entre chamadas consecutivas
set bitrate 8
set fs 0.020
set codec G729

# Audio Source Constants
set audioMuInv 1.004 ;# media da duracao em segundos do periodo ON do modelo ON/OFF
set audioLambdaInv 1.587 ;# media da duracao em segundos do periodo OFF do modelo ON/OFF
set intBW 0
set audSrcBitrate $bitrate ;# taxa do codec em kbps
set Fs 8000 ;# quantidade de amostras geradas pelo codec a cada segundo
set sourceBitPerSample 8 ;# tamanho em bits de cada amostra gerada pelo codec
set framelength $fs ;# duracao do quadro de voz em segundos
set audioPktSize [expr round([expr $framelength * $audSrcBitrate * 1000 / 8.0])] ;# tamanho do pacote de voz em bytes

source "voice.tcl" ;# carrega o arquivo de geracao de trafego de voz

# seleciona uma rota
proc nnode {{range 7}} {
return [expr {int(rand()*$range)}]
}

##########################################################################################################################
# A cada iteracao do loop eh iniciado o fluxo de voz de uma comunicacao
# A comunicacao principal eh do nodo 0 -> nodo $dest
# As comunicacoes secundarias sao nodo i -> nodo N-j, onde i>=1 e j>=2
for {set i 0} {$i < $numComun} {incr i} {
puts "par $orig $dest"

set compression1 [[$ns_ get-node-by-id $orig] node-cagent]
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]

$ns_ at $start "makeacall 0 $i 0 $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 $i 1 $dest $node_($dest) $node_($orig) $compression2 $compression1"
}

# Calls generation
```

```
proc makeacall {j comun sentido dest nodeSource nodeDest compOrig compDest} {
global ns_ interval audioPktSize numCalls duration node_ fs codec

set udp($comun,$sentido) [new Agent/RTP]
set cbr($comun,$sentido) [new Application/Traffic/CBR]
set lossMonitor($comun,$sentido) [new Agent/ConsBuffer]
$lossMonitor($comun,$sentido) set offset_ 0.050 ;# buffer de jitter
$lossMonitor($comun,$sentido) framelength $fs
$lossMonitor($comun,$sentido) payloadtype $codec
$lossMonitor($comun,$sentido) set bytes_ 0 ;# pra calcular o goodput
set numCalls($comun,$sentido) 0 ;# inicializa o numero de chamadas para comunicacao i

set audioStartTime [expr [$ns_ now]]                    ;# instante de inicio da chamada
set audioStopTime [expr $audioStartTime + $duration]    ;# instante de termino da chamada

# Invoca o metodo start do nodo 0, o qual faz parte da comunicacao principal
incr numCalls($comun,$sentido)
set seqno [$cbr($comun,$sentido) set seqno_]

create_audio_model [expr $comun + 1] $audioStartTime $audioStopTime $nodeSource $nodeDest $audioPktSize 1 \\
                        $lossMonitor($comun,$sentido) $udp($comun,$sentido) $cbr($comun,$sentido)

$lossMonitor($comun,$sentido) start $seqno
$ns_ at $audioStopTime "stopcall $j $comun $sentido $dest $numCalls($comun,$sentido) $interval $nodeSource $nodeDest \\
                        $lossMonitor($comun,$sentido) $udp($comun,$sentido) $cbr($comun,$sentido) $compOrig $compDest"
}

# Finishes a call
proc stopcall {j comun sentido dest call_id interval nodeSource nodeDest lossMonitor udp cbr compOrig compDest} {
    global ns_ numCalls node_ ak duration count qtde numComun

set seqno [$cbr set seqno_]                 ;# obtem o seqno_ do ultimo pacote da chamada
$lossMonitor stop $seqno
set mos_ [$lossMonitor set mos_]
set percLost_ [$lossMonitor set percLost_]
set delayAvg_ [$lossMonitor set delayAvg_]
set bw_ [$lossMonitor set bytes_]

$ak AkObservation 1 $mos_
$ak AkObservation 2 $percLost_
if { $percLost_ != 100 } {
$ak AkObservation 3 $delayAvg_
}

set sport [$udp set agent_port_]
set metade [expr $sport / 2]
set sindi [expr fmod($metade, $numComun)]

set dport [$udp set dst_port_]
set metade [expr $dport / 2]
set dindi [expr fmod($metade, $numComun)]
```

```
set sentpkts [$compOrig compressor get-sent-pkts $sindi]
set pay [$compOrig compressor get-sent-bytes-pay $sindi]
set hea [$compOrig compressor get-sent-bytes-hea $sindi]

set recvpkts [$compDest decompressor get-recv-pkts $dindi]
set discpkts [$compDest decompressor get-disc-pkts $dindi]

set bytes_originais [expr 40 * $sentpkts]
set econo [expr $bytes_originais - $hea]
        if { $bytes_originais != 0 } { set compGain_ [expr $econo.0 / $bytes_originais] } else { set compGain_ 0 }
set total_bytes_pkt [expr $pay + $hea]
if { $total_bytes_pkt != 0 } { set bwEfccy_ [expr $pay.0 / $total_bytes_pkt] } else { set bwEfccy_ 0 }

if { $recvpkts != 0 } { set desFactor_ [expr $discpkts.0 / $recvpkts] } else { set desFactor_ 0 }

if { $sentpkts != 0 } { set desLoss_ [expr $discpkts.0 / $sentpkts] } else { set desLoss_ 0 }

$ak AkObservation 4 $compGain_
$ak AkObservation 5 $bwEfccy_

puts "Cha $j Sen $sentido Com $comun MOS = $mos_ Perda = $percLost_ Atraso = $delayAvg_ compGain = $compGain_ bwEfccy = $bwEfccy_"

$compOrig compressor reset-monitor $sindi
$compDest decompressor reset-monitor $dindi

incr j

$ns_ at [expr [$ns_ now] + $interval] "makeacall $j $comun $sentido $dest $nodeSource $nodeDest $compOrig $compDest"
}

proc stop {} {
global ns_ ;#tracefd namtracens_
# $ns_ flush-trace
# close $tracefd
# close $namtracens_
$ns_ "halt"
}

# ======================================================================
# Links com ajuste automatico de taxa de transmissao (via algor. ARF) e respecitivos de thresholds de SINR
# ======================================================================

# source connection-pattern and node-movement scripts
if { $opt(cp) == "" } {
puts "*** NOTE: no connection pattern specified."
set opt(cp) "none"
} else {
puts "Loading connection pattern..."
source $opt(cp)
}
if { $opt(sc) == "" } {
puts "*** NOTE: no scenario file specified."
```

```
set opt(sc) "none"
} else {
puts "Loading scenario file..."
source $opt(sc)
puts "Load complete..."
}

# define initial node position in nam
for {set i 0} {$i < $opt(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 20
}

# begin simulation
puts "Starting Simulation..."

$ns_ run
```

B.2.3 *Script* para cenário em árvore 1

O *script* abaixo foi utilizado para realizar os experimentos com chamadas a partir dos nós folha. Deve ser executado da seguinte forma:

```
$ ns script.tcl <quantidade_de_chamadas_simultaneas>
```

```
# ======================================================================
# Definicoes das opcoes
# ======================================================================
set opt(chan)           Channel/WirelessChannel ;# channel type
set opt(prop)           Propagation/Shadowing ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy ;# network interface type
set opt(mac)            Mac/802_11 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue ;# interface queue type
set opt(ll)             LL ;# link layer type
set opt(ant)            Antenna/OmniAntenna ;# antenna model
set opt(ifqlen)         50 ;# max packet in ifq
set opt(nn)             7 ;# number of mobilenodes
set opt(adhocRouting)   OLSR ;# routing protocol
set opt(hdrcomp)        ALGB ;# compression algorithm - Options: ROHC, LECC (static compression),
                             ;# ALGB (static compression + packet aggregation)
set opt(x)              650 ;# x coordinate of topology
set opt(y)              650 ;# y coordinate of topology
set start               20 ;# instante de inicio da primeira chamada
set opt(tr)            /dev/null
set opt(cp)            ""
set opt(sc)            ""

ErrorModel80211 noise1  -104 ;# 1Mbps datarate thermal noise
ErrorModel80211 noise2  -101 ;# 2Mbps datarate thermal noise
ErrorModel80211 noise55 -97 ;# 5.5Mbps datarate thermal noise
ErrorModel80211 noise11 -92 ;# 11Mbps datarate thermal noise
```

```
ErrorModel80211 shortpreamble 1 ;# toggle 802.11 short preamble on/off
#ErrorModel80211 LoadBerSnrFile ber_snr_intersil.txt ;# use Intersil BER/SNR empirical curves
ErrorModel80211 LoadBerSnrFile ber_snr_choi.txt ;# use theoretical curves from del Prado
;# Pavon and Choi, "Link AdaptationStrategy
;# for IEEE 802.11 WLAN via Received Signal
;# Strength Measurement"

# ======================================================================
# Configuracao das Antenas Ganhos das antenas > irradiacao e recepcao
# ======================================================================
# Ganho Antena transmissora. Vivek: 13dBi Omniantenna total EIRP of 36dBm,
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 3
Antenna/OmniAntenna set Gt_ 1.0;
Antenna/OmniAntenna set Gr_ 1.0;

# ======================================================================
# Interface wireless
# ======================================================================
Phy/WirelessPhy set CPThresh_   10.0 ;# capture threshold
Phy/WirelessPhy set CSThresh_   3.1622777e-14 ;# Carrier Sensing threshold
Phy/WirelessPhy set RXThresh_   3.1622777e-13 ;# receiver signal threshold
Phy/WirelessPhy set bandwidth_  11Mb ;# channel bandwidth (UNUSED!)
Phy/WirelessPhy set Pt_         0.031622777 ;# transmitter signal power (Watt)
Phy/WirelessPhy set freq_       2.472e9 ;# channel frequency (Hz)
Phy/WirelessPhy set L_          1.0

Propagation/Shadowing set pathlossExp_  3.5 ;# path loss exponent
Propagation/Shadowing set std_db_       4.0 ;# standard deviation
Propagation/Shadowing set dist0_        1.0 ;# reference small distance
Propagation/Shadowing set seed_         0

# ======================================================================
# Parametros da MAC
# ======================================================================
Mac/802_11 set CWMin_                31
Mac/802_11 set CWMax_                1023
Mac/802_11 set SlotTime_             0.000020 ;# 20us
Mac/802_11 set SIFS_                 0.000010 ;# 10us
Mac/802_11 set PreambleLength_       144 ;# 144 bit
Mac/802_11 set ShortPreambleLength_  72 ;# 72 bit
Mac/802_11 set PreambleDataRate_     1.0e6 ;# 1Mbps
Mac/802_11 set PLCPHeaderLength_     48 ;# 48 bits
Mac/802_11 set PLCPDataRate_         1.0e6 ;# 1Mbps
Mac/802_11 set ShortPLCPDataRate_    2.0e6 ;# 2Mbps
Mac/802_11 set RTSThreshold_         3000 ;# bytes
Mac/802_11 set ShortRetryLimit_      7 ;# retransmissions
Mac/802_11 set LongRetryLimit_       4 ;# retransmissions
Mac/802_11 set newchipset_           false ;# use new chipset, allowing a more recent
;# packet to be correctly received in place
;# of the first sensed packet
```

```
Mac/802_11 set dataRate_ 11Mb ;# 802.11 data transmission rate
Mac/802_11 set basicRate_ 2Mb ;# 802.11 basic transmission rate
Mac/802_11 set aarf_ true ;# 802.11 Auto Rate Fallback

# ======================================================================
# Parametros da camada de Rede: Protocolo de Roteamento OLSR-ETX/ML/MD
# ======================================================================
Agent/OLSR set use_mac_ true
Agent/OLSR set debug_ false
Agent/OLSR set willingness 3
Agent/OLSR set hello_ival_ 2
Agent/OLSR set tc_ival_ 5

# ======================================================================
# Parametros do compressor de cabeçalhos
# ======================================================================

# If you are not using any header compression scheme, comment all the lines

# for SHC+AGG
Agent/ALGB set total_calls_     [lindex $argv 0]

# for static compression
#Agent/LECC set total_calls_ [lindex $argv 0]

# for ROHC
#Agent/ROHC set large_cid_ true
#Agent/ROHC set cohc_ false ;# set true to use cooperative header compression
#Agent/ROHC set total_calls_     [lindex $argv 0]

# ======================================================================
# Adiciona somente os cabecalhos de interesse
# ======================================================================
remove-all-packet-headers ;# remove todos os cabecalhos do pacote
add-packet-header RTP
add-packet-header IP
add-packet-header OLSR
add-packet-header LL
add-packet-header Mac
add-packet-header LDP
add-packet-header MPLS
add-packet-header ALGB ;# for SHC+AG
#add-packet-header LECC ;# for static compression
#add-packet-header ROHC ;# for ROHC

# ======================================================================
# Criando O Objeto simulador ns_, trace e nam
# ======================================================================
set ns_ [new Simulator]
set ak [new Akaroa]
$ak AkDeclareParameters 5
```

```
set tracefd [open $opt(tr) w] ;#From Vivek
$ns_ use-newtrace
$ns_ trace-all $tracefd

# Criacao e configuracao da area
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# =====================================================================
# Criando O Objeto GOD-General Operations Director
# =====================================================================
create-god $opt(nn)

# =====================================================================
# NAM Configurations
# =====================================================================
# Abre o arquivo out.nam para escrita e rotula-o com nf
#set namtracens_ [open ufam.nam w]

# Escreve dados da simulacao relevantes para o nam no arquivo representado por nf.
#$ns_ namtrace-all-wireless $namtracens_ $opt(x) $opt(y)

# =====================================================================
# Configuracao e Criacao dos nos (eh necessario primeiro configura-los)
# =====================================================================
# Configuracao dos nos
$ns_ node-config -adhocRouting $opt(adhocRouting)\
-headerCompression $opt(hdrcomp)\
-llType $opt(ll) \
-macType $opt(mac) \
-ifqType $opt(ifq) \
-ifqLen $opt(ifqlen) \
-antType $opt(ant) \
-propType $opt(prop) \
-phyType $opt(netif) \
-channel [new $opt(chan)] \
-topoInstance $topo \
-wiredRouting OFF \
-agentTrace OFF \
-routerTrace OFF \
-macTrace OFF \
-toraDebug OFF \
-movementTrace OFF
#From olsr etx. Unica diferenca eh:
# -channelType $opt(chan) \

# Criacao dos nos
for {set i 0} { $i < $opt(nn)} {incr i} {
        set node_($i) [$ns_ mpls-node ]
        $node_($i) random-motion 0 ;# disable random motion
}
```

```
# ======================================================================
# Configurando posicionamento dos nos
# ======================================================================
#===========================>
$node_(0) set X_ 25.0
$node_(0) set Y_ 25.0
$node_(0) set Z_ 15.0
#===========================>
$node_(1) set X_ 0.0
$node_(1) set Y_ 25.0
$node_(1) set Z_ 15.0
#===========================>
$node_(2) set X_ 50.0
$node_(2) set Y_ 25.0
$node_(2) set Z_ 15.0
#===========================>
$node_(3) set X_ 0.0
$node_(3) set Y_ 0.0
$node_(3) set Z_ 15.0
#===========================>
$node_(4) set X_ 0.0
$node_(4) set Y_ 50.0
$node_(4) set Z_ 15.0
#===========================>
$node_(5) set X_ 50.0
$node_(5) set Y_ 50.0
$node_(5) set Z_ 15.0
#===========================>
$node_(6) set X_ 50.0
$node_(6) set Y_ 0.0
$node_(6) set Z_ 15.0

# ============================================================
# MPLS stuff
# ============================================================

$ns_ enable-control-driven

for {set i 0} {$i < $opt(nn)} {incr i} {
set a $node_($i)
set m [eval $a get-module "MPLS"]
eval set LSRmpls$i $m
$m enable-reroute simple-dynamic
}

$ns_ at 6.0 "$ns_ configure-ldp-on-all-mpls-nodes"

$ns_ at 10.0 "$LSRmpls0 setup-erlsp  3  1_3  1000"
$ns_ at 11.0 "$LSRmpls3 setup-erlsp  0  1_0  2000"

$ns_ at 12.0 "$LSRmpls0 setup-erlsp  4  1_4  3000"
$ns_ at 13.0 "$LSRmpls4 setup-erlsp  0  1_0  4000"
```

```
$ns_ at 10.0 "$LSRmpls0 setup-erlsp  5  2_5  5000"
$ns_ at 11.0 "$LSRmpls5 setup-erlsp  0  2_0  6000"

$ns_ at 12.0 "$LSRmpls0 setup-erlsp  6  2_6  7000"
$ns_ at 13.0 "$LSRmpls6 setup-erlsp  0  2_0  8000"

$ns_ at 14.0 "$LSRmpls0 setup-erlsp  1  1  9000"
$ns_ at 15.0 "$LSRmpls0 setup-erlsp  2  2  10000"

# ======================================================================
# Trafego de voz - Modelo ON/OFF
# ======================================================================

# parameters
set opt(chamsimul) 1 ;# numero de chamadas simultaneas
set opt(N) 1 ;# número de nós
set N  $opt(N) ;# numero de chamadas
set duration 60 ;# duracao da chamada em segundos
set interval 2 ;# intervalo fixo entre chamadas consecutivas
set numComun $opt(chamsimul) ;# numero de comunicacoes
set bitrate 8
set fs 0.020
set codec G729

set numChaSimul [lindex $argv 0]

# Audio Source Constants
set audioMuInv 1.004 ;# media da duracao em segundos do periodo ON do modelo ON/OFF
set audioLambdaInv 1.587 ;# media da duracao em segundos do periodo OFF do modelo ON/OFF
set intBW 0
set audSrcBitrate $bitrate ;# taxa do codec em kbps
set Fs 8000 ;# quantidade de amostras geradas pelo codec a cada segundo
set sourceBitPerSample 8 ;# tamanho em bits de cada amostra gerada pelo codec
set framelength $fs ;# duracao do quadro de voz em segundos
set audioPktSize [expr round([expr $framelength * $audSrcBitrate * 1000 / 8.0])] ;# tamanho do pacote de voz em bytes

source "voice.tcl" ;# carrega o arquivo de geracao de trafego de voz

# seleciona uma fonte
proc nsrc {{range 15}} {
return [expr {int(rand()*$range)}]
}

# seleciona uma destino
proc ndst {{range 4}} {
return [expr {int(rand()*$range)}]
}

#############################################################################################################
# A cada iteracao do loop eh iniciado o fluxo de voz de uma comunicacao
# A comunicacao principal eh do nodo 0 -> nodo $dest
```

```
# As comunicacoes secundarias sao nodo i -> nodo N-j, onde i>=1 e j>=2
for {set i 0} {$i < $numChaSimul} {incr i} {

set orig 0
set compression1 [[$ns_ get-node-by-id $orig] node-cagent]

# Chamadas dos nós folha
set dest 3
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 3 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 3 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 4
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 4 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 4 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 5
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 5 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 5 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 6
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 6 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 6 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
}

# Calls generation
proc makeacall {rod comun cha sentido orig dest nodeSource nodeDest compOrig compDest} {
global ns_ interval audioPktSize duration node_ fs codec

set udp($orig,$dest,$cha) [new Agent/RTP]
set cbr($orig,$dest,$cha) [new Application/Traffic/CBR]
set lossMonitor($orig,$dest,$cha) [new Agent/ConsBuffer]
$lossMonitor($orig,$dest,$cha) set offset_ 0.050 ;# buffer de jitter
$lossMonitor($orig,$dest,$cha) framelength $fs
$lossMonitor($orig,$dest,$cha) payloadtype $codec
$lossMonitor($orig,$dest,$cha) set bytes_ 0 ;# pra calcular o goodput

# puts $lossMonitor($orig,$dest,$cha)

set audioStartTime [expr [$ns_ now]]                    ;# instante de inicio da chamada
set audioStopTime [expr $audioStartTime + $duration]    ;# instante de termino da chamada

# Invoca o metodo start do nodo 0, o qual faz parte da comunicacao principal
set seqno [$cbr($orig,$dest,$cha) set seqno_]

create_audio_model [expr $comun + 1] $audioStartTime $audioStopTime $nodeSource $nodeDest $audioPktSize 1 \\
                        $lossMonitor($comun,$sentido) $udp($comun,$sentido) $cbr($comun,$sentido)

$lossMonitor($comun,$sentido) start $seqno
$ns_ at $audioStopTime "stopcall $j $comun $sentido $dest $numCalls($comun,$sentido) $interval $nodeSource $nodeDest \\
                        $lossMonitor($comun,$sentido) $udp($comun,$sentido) $cbr($comun,$sentido) $compOrig $compDest"
}
```

```
# Finishes a call
proc stopcall {rod comun cha sentido orig dest interval nodeSource nodeDest lossMonitor udp cbr compOrig compDest} {
    global ns_ node_ ak duration count qtde numChaSimul total_pares_

set seqno [$cbr set seqno_]                  ;# obtem o seqno_ do ultimo pacote da chamada
$lossMonitor stop $seqno
set mos_ [$lossMonitor set mos_]
set percLost_ [$lossMonitor set percLost_]
set delayAvg_ [$lossMonitor set delayAvg_]
set bw_ [$lossMonitor set bytes_]

$ak AkObservation 1 $mos_
$ak AkObservation 2 $percLost_
if { $percLost_ != 100 } {
$ak AkObservation 3 $delayAvg_
}

set sport [$udp set agent_port_]
set metade [expr $sport / 2]
set sindi [expr fmod($metade, $numComun)]

set dport [$udp set dst_port_]
set metade [expr $dport / 2]
set dindi [expr fmod($metade, $numComun)]

set sentpkts [$compOrig compressor get-sent-pkts $sindi]
set pay [$compOrig compressor get-sent-bytes-pay $sindi]
set hea [$compOrig compressor get-sent-bytes-hea $sindi]

set recvpkts [$compDest decompressor get-recv-pkts $dindi]
set discpkts [$compDest decompressor get-disc-pkts $dindi]

set bytes_originais [expr 40 * $sentpkts]
set econo [expr $bytes_originais - $hea]
        if { $bytes_originais != 0 } { set compGain_ [expr $econo.0 / $bytes_originais] } else { set compGain_ 0 }
set total_bytes_pkt [expr $pay + $hea]
if { $total_bytes_pkt != 0 } { set bwEfccy_ [expr $pay.0 / $total_bytes_pkt] } else { set bwEfccy_ 0 }

if { $recvpkts != 0 } { set desFactor_ [expr $discpkts.0 / $recvpkts] } else { set desFactor_ 0 }

if { $sentpkts != 0 } { set desLoss_ [expr $discpkts.0 / $sentpkts] } else { set desLoss_ 0 }

$ak AkObservation 4 $compGain_
$ak AkObservation 5 $bwEfccy_

puts "Rod $rod Com $comun Cha $cha Sen $sentido Orig $orig Dest $dest MOS = $mos_ Perda = $percLost_ Atraso = $delayAvg_ "

$compOrig compressor reset-monitor $sindi
$compDest decompressor reset-monitor $dindi

incr rod
```

```
$ns_ at [expr [$ns_ now] + $interval] "makeacall $rod $comun $cha $sentido $orig $dest $nodeSource $nodeDest $compOrig $compDest"
}

proc stop {} {
global ns_ ;#tracefd namtracens_
# $ns_ flush-trace
# close $tracefd
# close $namtracens_
$ns_ "halt"
}

# ======================================================================
# Links com ajuste automatico de taxa de transmissao (via algor. ARF) e respecitivos de thresholds de SINR
# ======================================================================

# source connection-pattern and node-movement scripts
if { $opt(cp) == "" } {
puts "*** NOTE: no connection pattern specified."
set opt(cp) "none"
} else {
puts "Loading connection pattern..."
source $opt(cp)
}
if { $opt(sc) == "" } {
puts "*** NOTE: no scenario file specified."
set opt(sc) "none"
} else {
puts "Loading scenario file..."
source $opt(sc)
puts "Load complete..."
}

# define initial node position in nam
for {set i 0} {$i < $opt(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 20
}

# begin simulation
puts "Starting Simulation..."

$ns_ run
```

B.2.4 *Script* para cenário em árvore 2

O *script* abaixo foi utilizado para realizar os experimentos com chamadas a partir de todos os nós. Deve ser executado da seguinte forma:

```
$ ns script.tcl <quantidade_de_chamadas_simultaneas>
```

```
# ======================================================================
# Definicoes das opcoes
# ======================================================================
set opt(chan)           Channel/WirelessChannel ;# channel type
set opt(prop)           Propagation/Shadowing ;# radio-propagation model
set opt(netif)          Phy/WirelessPhy ;# network interface type
set opt(mac)            Mac/802_11 ;# MAC type
set opt(ifq)            Queue/DropTail/PriQueue ;# interface queue type
set opt(ll)             LL ;# link layer type
set opt(ant)            Antenna/OmniAntenna ;# antenna model
set opt(ifqlen)         50 ;# max packet in ifq
set opt(nn)             7 ;# number of mobilenodes
set opt(adhocRouting)   OLSR ;# routing protocol
set opt(hdrcomp)        ALGB ;# compression algorithm - Options: ROHC, LECC (static compression),
                             ;# ALGB (static compression + packet aggregation)
set opt(x)              650 ;# x coordinate of topology
set opt(y)              650 ;# y coordinate of topology
set start               20 ;# instante de inicio da primeira chamada
set opt(tr)            /dev/null
set opt(cp)            ""
set opt(sc)            ""

ErrorModel80211 noise1  -104 ;# 1Mbps datarate thermal noise
ErrorModel80211 noise2  -101 ;# 2Mbps datarate thermal noise
ErrorModel80211 noise55 -97 ;# 5.5Mbps datarate thermal noise
ErrorModel80211 noise11 -92 ;# 11Mbps datarate thermal noise
ErrorModel80211 shortpreamble 1 ;# toggle 802.11 short preamble on/off
#ErrorModel80211 LoadBerSnrFile ber_snr_intersil.txt ;# use Intersil BER/SNR empirical curves
ErrorModel80211 LoadBerSnrFile ber_snr_choi.txt ;# use theoretical curves from del Prado
;# Pavon and Choi, "Link AdaptationStrategy
;# for IEEE 802.11 WLAN via Received Signal
;# Strength Measurement"

# ======================================================================
# Configuracao das Antenas Ganhos das antenas > irradiacao e recepcao
# ======================================================================
# Ganho Antena transmissora. Vivek: 13dBi Omniantenna total EIRP of 36dBm,
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 3
Antenna/OmniAntenna set Gt_ 1.0;
Antenna/OmniAntenna set Gr_ 1.0;

# ======================================================================
# Interface wireless
# ======================================================================
Phy/WirelessPhy set CPThresh_   10.0 ;# capture threshold
Phy/WirelessPhy set CSThresh_   3.1622777e-14 ;# Carrier Sensing threshold
Phy/WirelessPhy set RXThresh_   3.1622777e-13 ;# receiver signal threshold
Phy/WirelessPhy set bandwidth_  11Mb ;# channel bandwidth (UNUSED!)
Phy/WirelessPhy set Pt_         0.031622777 ;# transmitter signal power (Watt)
Phy/WirelessPhy set freq_       2.472e9 ;# channel frequency (Hz)
```

```
Phy/WirelessPhy set L_          1.0

Propagation/Shadowing set pathlossExp_  3.5 ;# path loss exponent
Propagation/Shadowing set std_db_       4.0 ;# standard deviation
Propagation/Shadowing set dist0_        1.0 ;# reference small distance
Propagation/Shadowing set seed_         0

# ======================================================================
# Parametros da MAC
# ======================================================================
Mac/802_11 set CWMin_                 31
Mac/802_11 set CWMax_                 1023
Mac/802_11 set SlotTime_              0.000020 ;# 20us
Mac/802_11 set SIFS_                  0.000010 ;# 10us
Mac/802_11 set PreambleLength_        144 ;# 144 bit
Mac/802_11 set ShortPreambleLength_   72 ;# 72 bit
Mac/802_11 set PreambleDataRate_      1.0e6 ;# 1Mbps
Mac/802_11 set PLCPHeaderLength_      48 ;# 48 bits
Mac/802_11 set PLCPDataRate_          1.0e6 ;# 1Mbps
Mac/802_11 set ShortPLCPDataRate_     2.0e6 ;# 2Mbps
Mac/802_11 set RTSThreshold_          3000 ;# bytes
Mac/802_11 set ShortRetryLimit_       7 ;# retransmissions
Mac/802_11 set LongRetryLimit_        4 ;# retransmissions
Mac/802_11 set newchipset_            false ;# use new chipset, allowing a more recent
;# packet to be correctly received in place
;# of the first sensed packet
Mac/802_11 set dataRate_ 11Mb ;# 802.11 data transmission rate
Mac/802_11 set basicRate_ 2Mb ;# 802.11 basic transmission rate
Mac/802_11 set aarf_ true ;# 802.11 Auto Rate Fallback

# ======================================================================
# Parametros da camada de Rede: Protocolo de Roteamento OLSR-ETX/ML/MD
# ======================================================================
Agent/OLSR set use_mac_ true
Agent/OLSR set debug_ false
Agent/OLSR set willingness 3
Agent/OLSR set hello_ival_ 2
Agent/OLSR set tc_ival_ 5

# ======================================================================
# Parametros do compressor de cabeçalhos
# ======================================================================

# If you are not using any header compression scheme, comment all the lines

# for SHC+AGG
Agent/ALGB set total_calls_     [lindex $argv 0]

# for static compression
#Agent/LECC set total_calls_ [lindex $argv 0]

# for ROHC
```

```
#Agent/ROHC set large_cid_ true
#Agent/ROHC set cohc_ false ;# set true to use cooperative header compression
#Agent/ROHC set total_calls_     [lindex $argv 0]

# ======================================================================
# Adiciona somente os cabecalhos de interesse
# ======================================================================
remove-all-packet-headers ;# remove todos os cabecalhos do pacote
add-packet-header RTP
add-packet-header IP
add-packet-header OLSR
add-packet-header LL
add-packet-header Mac
add-packet-header LDP
add-packet-header MPLS
add-packet-header ALGB ;# for SHC+AG
#add-packet-header LECC ;# for static compression
#add-packet-header ROHC ;# for ROHC

# ======================================================================
# Criando O Objeto simulador ns_, trace e nam
# ======================================================================
set ns_ [new Simulator]
set ak [new Akaroa]
$ak AkDeclareParameters 5

set tracefd [open $opt(tr) w] ;#From Vivek
$ns_ use-newtrace
$ns_ trace-all $tracefd

# Criacao e configuracao da area
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)

# ======================================================================
# Criando O Objeto GOD-General Operations Director
# ======================================================================
create-god $opt(nn)

# ======================================================================
# NAM Configurations
# ======================================================================
# Abre o arquivo out.nam para escrita e rotula-o com nf
#set namtracens_ [open ufam.nam w]

# Escreve dados da simulacao relevantes para o nam no arquivo representado por nf.
#$ns_ namtrace-all-wireless $namtracens_ $opt(x) $opt(y)

# ======================================================================
# Configuracao e Criacao dos nos (eh necessario primeiro configura-los)
# ======================================================================
# Configuracao dos nos
```

```
$ns_ node-config -adhocRouting $opt(adhocRouting)\
-headerCompression $opt(hdrcomp)\
-llType $opt(ll) \
-macType $opt(mac) \
-ifqType $opt(ifq) \
-ifqLen $opt(ifqlen) \
-antType $opt(ant) \
-propType $opt(prop) \
-phyType $opt(netif) \
-channel [new $opt(chan)] \
-topoInstance $topo \
-wiredRouting OFF \
-agentTrace OFF \
-routerTrace OFF \
-macTrace OFF \
-toraDebug OFF \
-movementTrace OFF
#From olsr etx. Unica diferenca eh:
# -channelType $opt(chan) \

# Criacao dos nos
for {set i 0} { $i < $opt(nn)} {incr i} {
        set node_($i) [$ns_ mpls-node ]
        $node_($i) random-motion 0 ;# disable random motion
}

# ==================================================================
# Configurando posicionamento dos nos
# ==================================================================
#=========================>
$node_(0) set X_ 25.0
$node_(0) set Y_ 25.0
$node_(0) set Z_ 15.0
#=========================>
$node_(1) set X_ 0.0
$node_(1) set Y_ 25.0
$node_(1) set Z_ 15.0
#=========================>
$node_(2) set X_ 50.0
$node_(2) set Y_ 25.0
$node_(2) set Z_ 15.0
#=========================>
$node_(3) set X_ 0.0
$node_(3) set Y_ 0.0
$node_(3) set Z_ 15.0
#=========================>
$node_(4) set X_ 0.0
$node_(4) set Y_ 50.0
$node_(4) set Z_ 15.0
#=========================>
$node_(5) set X_ 50.0
$node_(5) set Y_ 50.0
```

```
$node_(5) set Z_ 15.0
#=========================>
$node_(6) set X_ 50.0
$node_(6) set Y_ 0.0
$node_(6) set Z_ 15.0

# cores ==========================
$ns_ color 1 red
$ns_ color 2 blue
$ns_ color 3 yellow

# ============================================================
# MPLS stuff
# ============================================================

$ns_ enable-control-driven

for {set i 0} {$i < $opt(nn)} {incr i} {
set a $node_($i)
set m [eval $a get-module "MPLS"]
eval set LSRmpls$i $m
$m enable-reroute simple-dynamic
}

$ns_ at 6.0 "$ns_ configure-ldp-on-all-mpls-nodes"

$ns_ at 10.0 "$LSRmpls0 setup-erlsp  3  1_3  1000"
$ns_ at 11.0 "$LSRmpls3 setup-erlsp  0  1_0  2000"

$ns_ at 12.0 "$LSRmpls0 setup-erlsp  4  1_4  3000"
$ns_ at 13.0 "$LSRmpls4 setup-erlsp  0  1_0  4000"

$ns_ at 10.0 "$LSRmpls0 setup-erlsp  5  2_5  5000"
$ns_ at 11.0 "$LSRmpls5 setup-erlsp  0  2_0  6000"

$ns_ at 12.0 "$LSRmpls0 setup-erlsp  6  2_6  7000"
$ns_ at 13.0 "$LSRmpls6 setup-erlsp  0  2_0  8000"

$ns_ at 14.0 "$LSRmpls0 setup-erlsp  1  1  9000"
$ns_ at 15.0 "$LSRmpls0 setup-erlsp  2  2  10000"

#for {set i 0} {$i < $opt(nn)} {incr i} {
# set m LSRmpls$i
# $ns_ at 9.0 "$$m pft-dump"
# $ns_ at 9.0 "$$m lib-dump"
# $ns_ at 9.0 "$$m erb-dump"
#}

# ====================================================================
# Trafego de voz - Modelo ON/OFF
# ====================================================================
```

```
# parameters
set opt(chamsimul) 1 ;# numero de chamadas simultaneas
set opt(N) 1 ;# número de nós
set N  $opt(N) ;# numero de chamadas
set duration 60 ;# duracao da chamada em segundos
set interval 2 ;# intervalo fixo entre chamadas consecutivas
set numComun $opt(chamsimul) ;# numero de comunicacoes
set bitrate 8
set fs 0.020
set codec G729

set numChaSimul [lindex $argv 0]

# Audio Source Constants
set audioMuInv 1.004 ;# media da duracao em segundos do periodo ON do modelo ON/OFF
set audioLambdaInv 1.587 ;# media da duracao em segundos do periodo OFF do modelo ON/OFF
set intBW 0
set audSrcBitrate $bitrate ;# taxa do codec em kbps
set Fs 8000 ;# quantidade de amostras geradas pelo codec a cada segundo
set sourceBitPerSample 8 ;# tamanho em bits de cada amostra gerada pelo codec
set framelength $fs ;# duracao do quadro de voz em segundos
set audioPktSize [expr round([expr $framelength * $audSrcBitrate * 1000 / 8.0])] ;# tamanho do pacote de voz em bytes

source "voice.tcl" ;# carrega o arquivo de geracao de trafego de voz

# seleciona uma fonte
proc nsrc {{range 15}} {
return [expr {int(rand()*$range)}]
}

# seleciona uma destino
proc ndst {{range 4}} {
return [expr {int(rand()*$range)}]
}

##############################################################################################################
# A cada iteracao do loop eh iniciado o fluxo de voz de uma comunicacao
# A comunicacao principal eh do nodo 0 -> nodo $dest
# As comunicacoes secundarias sao nodo i -> nodo N-j, onde i>=1 e j>=2
for {set i 0} {$i < $numChaSimul} {incr i} {

set orig 0
set compression1 [[$ns_ get-node-by-id $orig] node-cagent]

# Chamadas dos nós internos
set dest 1
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 1 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 1 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 2
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 2 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
```

```
$ns_ at $start "makeacall 0 2 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"

# Chamadas dos nós folha
set dest 3
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 3 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 3 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 4
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 4 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 4 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 5
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 5 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 5 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
set dest 6
set compression2 [[$ns_ get-node-by-id $dest] node-cagent]
$ns_ at $start "makeacall 0 6 $i 0 $orig $dest $node_($orig) $node_($dest) $compression1 $compression2"
$ns_ at $start "makeacall 0 6 $i 1 $dest $orig $node_($dest) $node_($orig) $compression2 $compression1"
}

# Calls generation
proc makeacall {rod comun cha sentido orig dest nodeSource nodeDest compOrig compDest} {
global ns_ interval audioPktSize duration node_ fs codec

set udp($orig,$dest,$cha) [new Agent/RTP]
set cbr($orig,$dest,$cha) [new Application/Traffic/CBR]
set lossMonitor($orig,$dest,$cha) [new Agent/ConsBuffer]
$lossMonitor($orig,$dest,$cha) set offset_ 0.050 ;# buffer de jitter
$lossMonitor($orig,$dest,$cha) framelength $fs
$lossMonitor($orig,$dest,$cha) payloadtype $codec
$lossMonitor($orig,$dest,$cha) set bytes_ 0 ;# pra calcular o goodput

# puts $lossMonitor($orig,$dest,$cha)

set audioStartTime [expr [$ns_ now]]                    ;# instante de inicio da chamada
set audioStopTime [expr $audioStartTime + $duration]    ;# instante de termino da chamada

# Invoca o metodo start do nodo 0, o qual faz parte da comunicacao principal
set seqno [$cbr($orig,$dest,$cha) set seqno_]

create_audio_model [expr $comun + 1] $audioStartTime $audioStopTime $nodeSource $nodeDest $audioPktSize 1 \\
                        $lossMonitor($comun,$sentido) $udp($comun,$sentido) $cbr($comun,$sentido)

$lossMonitor($comun,$sentido) start $seqno
$ns_ at $audioStopTime "stopcall $j $comun $sentido $dest $numCalls($comun,$sentido) $interval $nodeSource $nodeDest \\
                        $lossMonitor($comun,$sentido) $udp($comun,$sentido) $cbr($comun,$sentido) $compOrig $compDest"
}

# Finishes a call
proc stopcall {rod comun cha sentido orig dest interval nodeSource nodeDest lossMonitor udp cbr compOrig compDest} {
    global ns_ node_ ak duration count qtde numChaSimul total_pares_
```

```
set seqno [$cbr set seqno_]                  ;# obtem o seqno_ do ultimo pacote da chamada
$lossMonitor stop $seqno
set mos_ [$lossMonitor set mos_]
set percLost_ [$lossMonitor set percLost_]
set delayAvg_ [$lossMonitor set delayAvg_]
set bw_ [$lossMonitor set bytes_]

$ak AkObservation 1 $mos_
$ak AkObservation 2 $percLost_
if { $percLost_ != 100 } {
$ak AkObservation 3 $delayAvg_
}

set sport [$udp set agent_port_]
set metade [expr $sport / 2]
set sindi [expr fmod($metade, $numComun)]

set dport [$udp set dst_port_]
set metade [expr $dport / 2]
set dindi [expr fmod($metade, $numComun)]

set sentpkts [$compOrig compressor get-sent-pkts $sindi]
set pay [$compOrig compressor get-sent-bytes-pay $sindi]
set hea [$compOrig compressor get-sent-bytes-hea $sindi]

set recvpkts [$compDest decompressor get-recv-pkts $dindi]
set discpkts [$compDest decompressor get-disc-pkts $dindi]

set bytes_originais [expr 40 * $sentpkts]
set econo [expr $bytes_originais - $hea]
        if { $bytes_originais != 0 } { set compGain_ [expr $econo.0 / $bytes_originais] } else { set compGain_ 0 }
set total_bytes_pkt [expr $pay + $hea]
if { $total_bytes_pkt != 0 } { set bwEfccy_ [expr $pay.0 / $total_bytes_pkt] } else { set bwEfccy_ 0 }

if { $recvpkts != 0 } { set desFactor_ [expr $discpkts.0 / $recvpkts] } else { set desFactor_ 0 }

if { $sentpkts != 0 } { set desLoss_ [expr $discpkts.0 / $sentpkts] } else { set desLoss_ 0 }

$ak AkObservation 4 $compGain_
$ak AkObservation 5 $bwEfccy_

puts "Rod $rod Com $comun Cha $cha Sen $sentido Orig $orig Dest $dest MOS = $mos_ Perda = $percLost_ Atraso = $delayAvg_ "

$compOrig compressor reset-monitor $sindi
$compDest decompressor reset-monitor $dindi

incr rod

$ns_ at [expr [$ns_ now] + $interval] "makeacall $rod $comun $cha $sentido $orig $dest $nodeSource $nodeDest $compOrig $compDest"
}
```

```
proc stop {} {
global ns_ ;#tracefd namtracens_
# $ns_ flush-trace
# close $tracefd
# close $namtracens_
$ns_ "halt"
}

# ======================================================================
# Links com ajuste automatico de taxa de transmissao (via algor. ARF) e respecitivos de thresholds de SINR
# ======================================================================

# source connection-pattern and node-movement scripts
if { $opt(cp) == "" } {
puts "*** NOTE: no connection pattern specified."
set opt(cp) "none"
} else {
puts "Loading connection pattern..."
source $opt(cp)
}
if { $opt(sc) == "" } {
puts "*** NOTE: no scenario file specified."
set opt(sc) "none"
} else {
puts "Loading scenario file..."
source $opt(sc)
puts "Load complete..."
}

# define initial node position in nam
for {set i 0} {$i < $opt(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 20
}

# begin simulation
puts "Starting Simulation..."

$ns_ run
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo