Mecanismos para Recuperação de Perdas de Pacotes de Voz na Internet

Daniel Ratton Figueiredo - ratton@nce.ufrj.br *
Universidade Federal do Rio de Janeiro, COPPE/Sistemas

Bruno Ratton Brandi - brandi@land.ufrj.br †
Universidade Federal do Rio de Janeiro, Dept. de Computação

Edmundo de Souza e Silva - edmundo@nce.ufrj.br ‡
Universidade Federal do Rio de Janeiro, NCE, DCC e COPPE/Sistemas
Cx. P. 2324, Rio de Janeiro, RJ - 20001-970 - Brasil

Resumo

Transmissão de voz pela Internet está se tornando uma atividade comum entre seus usuários. Porém, diversos problemas relacionados com a qualidade de áudio, são enfrentados pelos aplicativos de tempo real. Dois principais fatores, responsáveis pela má qualidade de áudio, são o atraso inserido pela rede e a perda de pacotes. Para melhorar a qualidade do áudio é preciso recuperar os pacotes perdidos, porém, a eficiência do mecanismo de recuperação depende do processo de perda de pacotes na rede. Neste artigo, estuda-se o processo de perda, com o objetivo de avaliar diferentes algoritmos de recuperação de pacotes. Este estudo é baseado em medidas realizadas na Internet entre a UFRJ e a UCLA. Além disso, propõe-se um algoritmo para fazer esta recuperação e mostra-se que este algoritmo é mais eficiente que os atualmente usados.

Abstract

Multimedia applications, such as voice transmission, has been increasingly used over the Internet. There are still many issues under investigation, related to the quality of audio delivered. Two of the main issues considered are the jitter and the loss of audio packets. In order to improve the audio quality, recovery mechanisms must be used. However, the efficiency of these mechanisms is largely dependent in the loss process in the network. In this paper we study the packet loss process of audio streams aiming at comparing different recovery mechanisms. We also propose a new algorithm and show that its efficiency is significantly better than those currently used.

1 Introdução

A transmissão de dados de tempo real na Internet, como vídeo e voz, vem se tornando uma atividade comum entre seus usuários. Conferências de áudio e vídeo são cada vez mais divulgadas e assistidas no MBone [6]. O interesse por tais aplicativos deve-se ao aumento

*D.R. Figueiredo possui bolsa de mestrado da CAPES.
†B.R. Brandi possui bolsa de iniciação científica do CNPq.
‡Este trabalho conta com o apoio do CNPq/ProTeM e PRONEX.

da vazão nas interconexões da rede, à disponibilidade do hardware necessário (periféricos multimídia estão se tornando comuns) e ao baixo custo da comunicação.

Aplicativos para transmissão de voz na Internet já existem há alguns anos. Atualmente, uma grande variedade desses aplicativos, comerciais ou freeware, está disponível. Exemplos de aplicativos freeware para transmissão de voz são o vat [10], o nevot [16], o rat [9] e o FreePhone [18]. O primeiro, pioneiro na transmissão de voz pela Internet, é largamente utilizado pelo meio acadêmico. Entre os programas comerciais, destacam-se o Internet Phone da VocalTec, bastante utilizado na plataforma Windows, e o Cooltalk da Netscape. Além desses, existem vários outros aplicativos de voz disponíveis na Internet.

Um dos problemas da existência de uma ampla gama de aplicativos diferentes é a incompatibilidade entre eles. Em geral, o protocolo utilizado para transmissão de voz, e a codificação/compressão do áudio são proprietários, o que torna impossível a comunicação entre aplicativos de diferentes fabricantes. Uma tentativa de padronização de um protocolo de transporte para tráfego em tempo real está sendo feita pelo IETF. Atualmente, existe um draft do padrão RTP/RTL (Real-Time Transmission Protocol) [15, 14], que está em fase de aprovação. Alguns aplicativos freeware, como o vat e FreePhone, já implementam este padrão e podem se comunicar sem muita dificuldade.

Por serem de tempo real, esses aplicativos multimídia têm características bem específicas, e qualquer deturpação dessas características, como o atraso na transmissão dos dados, pode comprometer seu uso. Então, para um bom funcionamento destes aplicativos é necessário que a rede ofereça qualidade mínima de serviço. Entretanto, a Internet de hoje oferece uma única classe de serviço, a de *best effort*. Esta não oferece nenhuma garantia quanto a qualidade de serviço que o aplicativo irá receber. Parâmetros como banda passante mínima, retardo máximo e taxa de perda máxima de pacotes não são garantidos, comprometendo a transmissão de tráfego de tempo real.

Neste artigo são abordadas questões relacionadas à recuperação de pacotes de voz perdidos em transmissões pela Internet. Apresentam-se alguns algoritmos de recuperação de pacotes existentes e estuda-se o processo de perda de pacotes na rede para avaliar a eficácia dos mesmos. Por fim, propõe-se um algoritmo que realiza esta recuperação, e mostra-se que este algoritmo é mais eficiente que os atualmente usados. Estes estudos são baseados em medidas feitas na Internet entre o Brasil e os EUA.

Na próxima seção, apresentam-se os problemas existentes na transmissão de voz em tempo real por uma rede, e como esses podem ser resolvidos ou amenizados. Na seção 3 abordam-se os algoritmos utilizados para recuperar pacotes perdidos e as vantagens e desvantagens de cada um. Na seção 4, analisa-se o processo de perda de pacotes e as medidas estatísticas coletadas da rede. Na seção 5 propõe-se um novo esquema para recuperar pacotes perdidos e vê-se como diferentes esquemas de redundância se comportam quando submetidos às mesmas características da rede. Na seção 6 conclui-se o artigo.

2 Transmissão de Voz em Tempo Real

Em um aplicativo de voz de tempo real, a perda de pacotes e o atraso entre eles são fatores fundamentais que influenciam na recuperação do sinal transmitido e, consequentemente, na qualidade da voz que chega ao destinatário. Pacotes que são perdidos na rede causam falhas no sinal de voz no aplicativo que os recebe. Essas falhas podem prejudicar a compreensão da conversa caso as perdas sejam longas (i.e. de vários pacotes consecutivos) ou muito freqüentes. Porém, falhas raras que estejam muito espaçadas são imperceptíveis e

não comprometem a conversa. Por isso, para manter uma boa qualidade de voz, o número de pacotes que chegam entre duas falhas deve ser grande e o número de falhas deve ser pequeno.

O segundo fator fundamental para garantir uma boa qualidade de voz, está relacionado com o atraso aleatório inserido pela rede entre os pacotes. Este atraso, conhecido como *jitter*, mede a variação do tempo de chegada entre pacotes consecutivos em relação ao tempo de geração (constante) desses pacotes na fonte. Se atrasos entre pacotes são freqüentes e longos, então falhas na voz irão ocorrer no aplicativo destino. Para garantir uma boa qualidade de voz, é necessário manter o *jitter* baixo, em torno de zero.

Além dos problemas impostos pela rede, a degradação da qualidade do áudio também pode ser causada por outros motivos. O mau uso ou a má configuração dos microfones e dos alto-falantes podem ser responsáveis pela má qualidade do áudio. Este problema foi apontado pelo IETF e pelo projeto MICE [12]. Porém, esta degradação será amenizada à medida que os usuários vão se familiarizando com os aplicativos e periféricos, e os aplicativos vão se tornando amigáveis com interfaces mais fáceis de usar.

Ainda assim, a rede é a maior responsável pela má qualidade do áudio, por não oferecer a qualidade de serviço adequada. Para solucionar este problema, duas abordagens são consideradas na literatura. A primeira é a modificação do protocolo de transporte e das políticas de escalonamento de pacotes nos roteadores, para que suportem tráfego de tempo real. Com isto seria possível garantir a qualidade de serviço exigida pelo aplicativo. Para implementar esta modificação será necessário que a rede (protocolo + roteadores) suporte controle de admissão, reserva de banda, alocação prévia de recursos e disciplinas de escalonamento sofisticadas. Estes mecanismos ainda não foram implementados na Internet, sendo esta uma área de pesquisa onde novas propostas e análises de desempenho estão surgindo.

A segunda abordagem é implementar nos aplicativos de tempo real, algoritmos que melhorem a qualidade do serviço oferecido pela rede. O objetivo desses algoritmos é suavizar o impacto da mudança das características da rede ao longo do tempo, sobre a qualidade de serviço. Nos aplicativos de voz, esses algoritmos tentam reduzir o *jitter* e a taxa de perda de pacotes, que são os principais responsáveis pela qualidade do áudio.

Jitter existe em quase todas as redes, e várias propostas para solucionar ou minimizar o problema foram sugeridas. Em uma delas, foi visto que a utilização de um algoritmo simples, que utiliza um buffer na saída da rede, reduz significativamente o *jitter* [8, 7]. Neste algoritmo, os pacotes que chegam da rede não são consumidos imediatamente pelo aplicativo destino, mas armazenados em um buffer até que um certo número de pacotes seja acumulado. Só então o primeiro pacote que chegou é entregue ao destino. Desta forma, elimina-se por completo o *jitter* negativo e o *jitter* positivo pode ser minimizado. Porém, o atraso fim a fim pode prejudicar a interatividade da conversa. Por isto, o número de pacotes inicialmente acumulados no buffer não deve ser grande. Em [8] mostrou-se que este número pode ser pequeno e mesmo assim uma redução significativa do *jitter* é obtida.

O segundo tipo de algoritmo tenta recuperar pacotes perdidos durante a transmissão da voz. Esta recuperação pode ser feita de diversas maneiras através de diferentes algoritmos. Na próxima seção, apresentam-se os algoritmos mais utilizados.

3 Algoritmos para Recuperação de Perdas

A qualidade do áudio em aplicativos de voz é afetada, principalmente, pela perda dos pacotes na rede. Com isto é essencial que pacotes perdidos sejam, de alguma forma, recuperados. Alguns aplicativos de transmissão de voz já implementam esquemas de recuperação de pacotes perdidos. Recentemente, foi lançada uma versão do FreePhone que utiliza redundância para recuperar pacotes perdidos. O Internet Phone, também possui um esquema de redundância, porém proprietário, por ser um aplicativo comercial. O vat ainda não possui nenhum esquema para recuperação de perdas. Nesta seção são abordados alguns algoritmos para recuperação de pacotes.

Existem, basicamente, duas formas de recuperar pacotes perdidos em transmissões de dados. A primeira, consiste em retransmitir o pacote de dados perdido. A segunda, consiste em reconstruir os pacotes perdidos utilizando informação redundante que é enviada juntamente com o fluxo de dados.

O esquema de retransmissão dos pacotes perdidos é bastante conhecido e utilizado por diversos protocolos como, por exemplo, o protocolo ARQ (Automatic Repeat Request) (e.g. [1]). Todos esses esquemas utilizam a técnica de loop fechado para fazer a retransmissão, e somente os pacotes que não chegam ao destino são retransmitidos. Porém, uma retransmissão só pode ser disparada de duas maneiras. Ou por um *nack* vindo do aplicativo destino, informando que um pacote se perdeu, ou por esgotamento de um temporizador acionado no aplicativo origem que esperava por um *ack* do aplicativo destino, confirmando a chegada do pacote.

Este mecanismo de recuperação de pacotes perdidos não é adequado para transmissão de voz em tempo real, pois aumenta o retardo fim-a-fim. A retransmissão de um pacote perdido pode demorar significativamente, o que prejudica a interatividade da conversa, causando súbitas interrupções na fala. O atraso na retransmissão do pacote perdido está relacionado com o *round-trip time* da conexão. Em redes como a Intenet, onde este tempo pode ser muito alto em relação ao tempo de geração dos pacotes, esses algoritmos se tornam inviáveis.

A segunda forma de recuperar os pacotes perdidos, conhecida como FEC (Forward Error Correction), é baseada em loop aberto [2]. Neste mecanismo, redundância é enviada juntamente com os pacotes de dados, e em caso de perda, o pacote é reconstruído. A vantagem deste esquema é que não existe retransmissão, porém, sua desvantagem é que nem todos os pacotes perdidos podem ser recuperados. O número de perdas que o algoritmo irá recuperar, está diretamente relacionado com a quantidade de redundância que o mesmo irá enviar. Além disso, o *overhead* introduzido pela redundância pode ser grande, diminuindo a eficiência da transmissão, e portanto, a quantidade de informação redundante deve ser minimizada.

Existem diversas variações de esquemas do tipo FEC para recuperar erros. Um esquema bastante conhecido, e utilizado em outras áreas da computação, é a utilização do ou-exclusivo para compor a redundância. Neste esquema, divide-se a seqüência de pacotes em janelas. Em cada janela, a redundância é obtida através do ou-exclusivo de todos os pacotes pertencentes à janela. A redundância, neste caso, possui o mesmo tamanho dos pacotes de dados e é enviada juntamente com o primeiro pacote da próxima janela. Desta forma, o primeiro pacote de cada janela possui duas vezes o tamanho de um pacote de dados. O *overhead* deste esquema é dado por $1/k$, onde k é o tamanho da janela. A Figura 1 ilustra este exemplo onde o tamanho da janela é de 5 pacotes.

Utilizando este esquema, pode-se recuperar até 1 pacote em cada janela. Se mais de

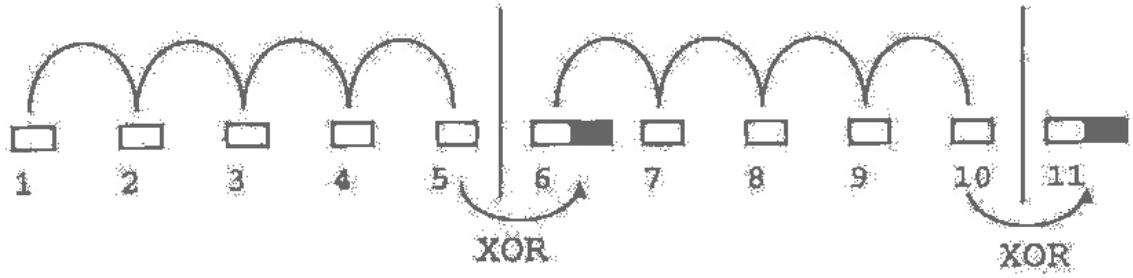

Figura 1: Esquema de redundância com janela de 5 pacotes.

um pacote for perdido na mesma janela então nenhum deles pode ser recuperado. Outra desvantagem deste esquema é o aumento do retardo fim-a-fim quando o tamanho da janela é aumentado, pois, para recuperar um pacote perdido, é necessário que cheguem todos os pacotes da janela atual e o primeiro pacote da próxima janela (que possui a redundância) para que a reconstrução possa ser feita.

Uma de suas vantagens é a simplicidade com que pode ser implementado. Além disso, conforme foi visto na seção 1, para eliminar o *jitter*, um buffer na saída da rede já é utilizado, e este mesmo buffer pode também ser usado para reconstruir os pacotes perdidos. Mantendo o tamanho da janela pequeno, não haverá retardos adicionais.

É importante ressaltar que a redundância em si, não está coberta por nenhum ou-exclusivo e que somente os pacotes contendo áudio podem ser recuperados. Desta forma, na figura 1, se os pacotes 6 e 3 se perdessem, apenas o pacote 6 seria recuperado. O pacote 3 não poderia ser recuperado, pois a redundância que protege esta janela foi perdida.

Um outro esquema de recuperação de pacotes, que utiliza FEC, foi recentemente proposto em [4, 5] e está implementado no aplicativo FreePhone. Neste mecanismo, o pacote n inclui, além da sua amostra de áudio, informação redundante do pacote $n-1$. A redundância enviada é comprimida, utilizando um algoritmo padrão para compressão de voz, para diminuir o *overhead* do esquema. Se o pacote n se perder, o destinatário aguarda até a chegada do pacote $n+1$, descomprime a informação redundante e recupera o pacote perdido. Este esquema recupera qualquer perda isolada, ou seja, perdas que envolvem apenas um pacote. Porém, é importante notar que caso todas as amostras de áudio estiverem sendo comprimidas antes de serem enviadas, o *overhead* deste mecanismo será de 100%, pois a redundância terá o mesmo tamanho do pacote original. É importante ressaltar ainda que este esquema é idêntico ao esquema do ou-exclusivo quando o tamanho de janela é igual a 1 pois, neste caso, todos os pacotes possuem a redundância do pacote anterior.

Para capturar perdas de mais de um pacote consecutivo, uma modificação do mecanismo acima também foi proposta em [5]. Aumentando a quantidade de informação redundante carregada em cada pacote. Desta forma, o pacote n possui a redundância dos pacotes $n-1$ e $n-2$. Porém, para diminuir o *overhead*, essas redundâncias podem ser comprimidas com algoritmos que reduzem ainda mais o tamanho da amostra de áudio ao custo de perder qualidade no momento da descompressão. Novamente, se todas as amostras já estiverem sendo comprimidas antes de serem enviadas, o *overhead* neste caso será de 200%.

Na próxima seção analisa-se e caracteriza-se o processo de chegada de pacotes de voz, passando pela Internet, através de medidas feitas na rede. Utilizando a seqüência dos pacotes que chegaram, bem como o tempo de chegada de cada pacote, é possível caracterizar o tráfego de voz e obter várias medidas estatísticas. A partir destas medidas, como a probabilidade de perda de um pacote, é possível avaliar e comparar diferentes esquemas

de redundância. Com base nessas medidas, propõe-se um esquema para recuperação de perdas que pode ser mais eficiente do que os algoritmos vistos.

4 Análise do Tráfego de Voz e Caracterização do Processo de Perda

Conforme foi visto, a qualidade do áudio em aplicativos de voz depende, principalmente, da perda de pacotes de voz e do *jitter* da rede. Viu-se, também, que vários algoritmos podem ser utilizados para minimizar a perda e reduzir o *jitter* melhorando a qualidade do áudio. Porém, a eficiência de todos esses algoritmos depende diretamente da qualidade de serviço que a rede está oferecendo. Além disso, todos apresentam algum compromisso para melhorar a qualidade do áudio. Este compromisso pode ser entre a interatividade da conversa e a redução do *jitter*, ou entre o *overhead* causado pelo envio de redundância e a recuperação de pacotes perdidos.

Um estudo do processo de chegada de pacotes de voz que atravessam a Internet se torna essencial para avaliar e comparar diferentes algoritmos. De posse de diversas distribuições e medidas estatísticas, torna-se mais fácil avaliar como os algoritmos, que se propõem a melhorar a qualidade do áudio, atuariam sob as mesmas condições.

Uma técnica bastante utilizada para redução da banda passante necessária para transmissão de voz, é o uso de algoritmos de compressão. Avalia-se, também, o impacto da compactação de voz sobre o processo de perda de pacotes imposto pela rede.

Vários estudos que caracterizam o processo de perda de pacotes e o atraso entre eles, na Internet, já foram realizados [3]. As medidas reportadas nestes trabalhos são média, variância e distribuição da perda de pacotes. Porém, medidas como a distribuição do tamanho da seqüência de sucesso e a distribuição do tamanho da perda consecutiva, não abordadas nestes estudos, são necessárias para comparar a eficiência de diferentes algoritmos de recuperação de pacotes, como veremos na seção abaixo.

Para fazer um estudo mais apurado e avaliar a qualidade de serviço da rede, uma bateria de testes foi feita. O objetivo era coletar dados referentes à transmissão de pacotes de voz na Internet e obter diversas medidas estatísticas.

4.1 Metodologia dos Testes

Todos os testes foram realizados entre a UFRJ/NCE e a Universidade da California, Los Angeles (UCLA) durante 7 dias consecutivos em três horários diferentes. Em cada horário foram enviados 10000 pacotes com dois tamanhos diferentes. Na primeira rodada, pacotes de 160 bytes foram enviados a intervalos constantes de 20ms, representando áudio codificado em PCM. Na segunda, pacotes de 32 bytes foram enviados a intervalos constantes de 20ms representando o áudio compactado pelo algoritmo GSM, padrão europeu para telefonia celular digital [13, 17]. Esses dois testes foram executados sequencialmente nesta ordem. A escolha de pacotes contendo 20ms de áudio segue a recomendação feita por [11], que indica que os pacotes de voz devem conter 16ms a 30ms. Além disso, diversos aplicativos, como o vat, nevot e vivavoz, utilizam pacotes contendo 20ms de áudio.

Os pacotes foram enviados sem que houvesse interrupção. Desta forma não consideramos algoritmos de detecção de silêncio e avaliamos o pior caso, já que a fonte de voz não é interrompida. A transmissão constante dos pacotes, sem que haja uma interrupção que

representa um período de silêncio, não afeta os resultados estatísticos pois vários outros tráfegos compartilham o mesmo caminho e a eventual interrupção da transmissão de voz não modificaria a carga total, muito maior, da rede.

Os testes foram realizados nos seguintes horários: às 8h na UFRJ (2h UCLA), em um horário de baixo tráfego na rede, às 13h (7h UCLA) horário de carga moderada e às 19h (13h UCLA) em um horário de maior tráfego na rede. Os pacotes dos testes foram transmitidos no sentido Rio de Janeiro - Los Angeles e no momento dos testes, existiam 16 roteadores entre as duas instituições. Além disso, o caminho feito pelos pacotes era composto por diversas redes heterogêneas, incluindo dois nós ATM. O caminho entre a UFRJ e a UCLA recebe tráfego de vários locais do mundo proporcionando uma rica mistura para nosso experimento. Testes entre a UFRJ e a PUC-RJ também foram realizados porém, a perda de pacotes entre estas instituições é praticamente inexistente. Em um trabalho futuro, pretendemos fazer medições entre a UFRJ e a UFMG, entre a UFRJ e a UFPe e entre a UFRJ e a UFG. Entretanto, o propósito deste artigo ainda continuará sendo válido.

Para realizar os testes, dois programas foram utilizados. Um deles, o programa cliente, emite pacotes numerados sequencialmente de tamanho fixo a uma taxa constante. O objetivo deste programa é simular uma fonte de voz sem períodos de silêncio.

O segundo programa, o servidor, recebe os pacotes que foram enviados pelo programa cliente e armazena o tempo de chegada de cada pacote. No final da transmissão, o servidor gera um arquivo contendo o número de seqüência dos pacote que chegaram assim como o tempo de chegada, em milisegundos, de cada pacote. Este arquivo mantém os pacotes na ordem de chegada.

Utilizamos o protocolo UDP para estabelecer a comunicação entre os dois programas. Desta forma, existe a possibilidade de perda de pacotes, retardo, duplicatas e chegadas fora de ordem. Este protocolo de transporte é utilizado para tráfego de tempo real em redes do tipo TCP/IP e faz parte do protocolo RTP (Real Time Protocol) [15].

4.2 Medidas Estatísticas

Ao final de uma transmissão, o programa servidor gera um arquivo contendo o *trace* daquela execução. Com este arquivo, várias medidas estatísticas podem ser obtidas. Estaremos interessados nas seguintes medidas:

- probabilidade de perda de pacote
- distribuição do tamanho da seqüência de sucesso
- distribuição do tamanho da seqüência formada por perdas consecutivas
- distribuição do *jitter*
- distribuição da distância dos pacotes fora de ordem

A probabilidade de perda de pacote é a razão entre o número de pacotes perdidos e o número de pacotes enviados.

Seja X a variável aleatória que indica o número de pacotes consecutivos que chegam entre duas perdas. $P[X = k]$ é a função probabilidade de massa (probability mass function, p.m.f.) de X.

	8hs	13hs	19hs
P[perda de pacote]	0.0158	0.0711	0.2614
$E[X]$	1277.66	27.30	4.25
$E[Y]$	1.19	1.16	1.43

Tabela 1: Medidas estatísticas para diferentes horários.

Seja Y a variável aleatória que indica o número de pacotes perdidos entre dois pacotes recebidos corretamente. $P[Y = k]$ é a função probabilidade de massa (probability mass function, p.m.f.) de Y.

Seja τ_n o tempo de chegada do n-ésimo pacote transmitido e T o intervalo (fixo) entre geração de pacotes de voz na fonte. A variável aleatória $J = (\tau_n - \tau_{n-1}) - T$, para $n = 2, \ldots$ é chamada de *jitter*. A $P[J > t]$ é a distribuição do *jitter* da seqüência transmitida.

Seja x_k o número de seqüência do k-ésimo pacote que chega ao destino. A medida a seguir define uma relação entre a ordem de chegada de um pacote e a ordem da seqüência original. Uma vez que no protocolo UDP não necessariamente os pacotes chegam na ordem enviada e portanto x_k pode ser diferente de k. Seja m o número de seqüência esperado para o próximo pacote. Se $x_k \geq m$ então $m = x_k + 1$ e caso contrário seu valor não é alterado. A variável $O(x_k) = m - x_k - 1$ indica a "distância" do pacote recebido em relação a sua posição original. Por exemplo, se $x_1 = 1, x_2 = 4, x_3 = 3, x_4 = 2, x_5 = 5$ então os pacotes x_3 e x_4 são considerados fora de ordem em relação ao pacote x_2, e $O(x_3) = 1$ e $O(x_4) = 2$, uma vez que x_4 chegou duas unidades atrasado em relação a x_2. A $P[O(x_n) = k | O(x_n) \geq 0]$ é a função probabilidade de massa (p.m.f.) desta "distância" dado que a ordem foi trocada.

A partir da p.m.f., os valores esperados de todas as medidas podem ser facilmente calculados. Além dessas medidas estatísticas, plotamos também diversos histogramas que ajudam a visualizar o comportamento da rede.

Uma medida bastante utilizada para caracterizar o processo de perda de pacotes de voz, e avaliar a qualidade de serviço da rede, é a fração de pacotes perdidos ou probabilidade de perda. Porém, esta medida não expressa o tamanho das rajadas de sucesso que, como veremos, é uma medida muito importante para avaliar a eficiência dos algoritmos que estudamos. Além disso, o valor esperado do tamanho da perda, também é uma medida que pode ser utilizada. Utilizaremos essas medidas para caracterizar o processo de perda de pacotes de voz na rede. Medidas referentes ao *jitter* não serão abordadas por estarem fora do escopo deste trabalho, e foram tratadas em [8].

Para apresentar os resultados, os *traces* foram agrupados por horário e por tamanho de pacote (PCM e GSM). A média amostral da distribuição de todas as medidas foi feita respeitando este agrupamento.

Apresentamos agora alguns gráficos e histogramas obtidos nos testes.

Os histogramas da figura 2 mostram o tamanho de cada perda nas seqüências transmitidas para três testes realizados em horários distintos. Todos os outros histogramas de testes realizados nos mesmos horários tiveram comportamento semelhante aos apresentados. Podemos notar que este tamanho se concentra entre 1 e 2 pacotes, o que pode ser comprovado pelo valor esperado do tamanho da perda apresentado na Tabela 1.

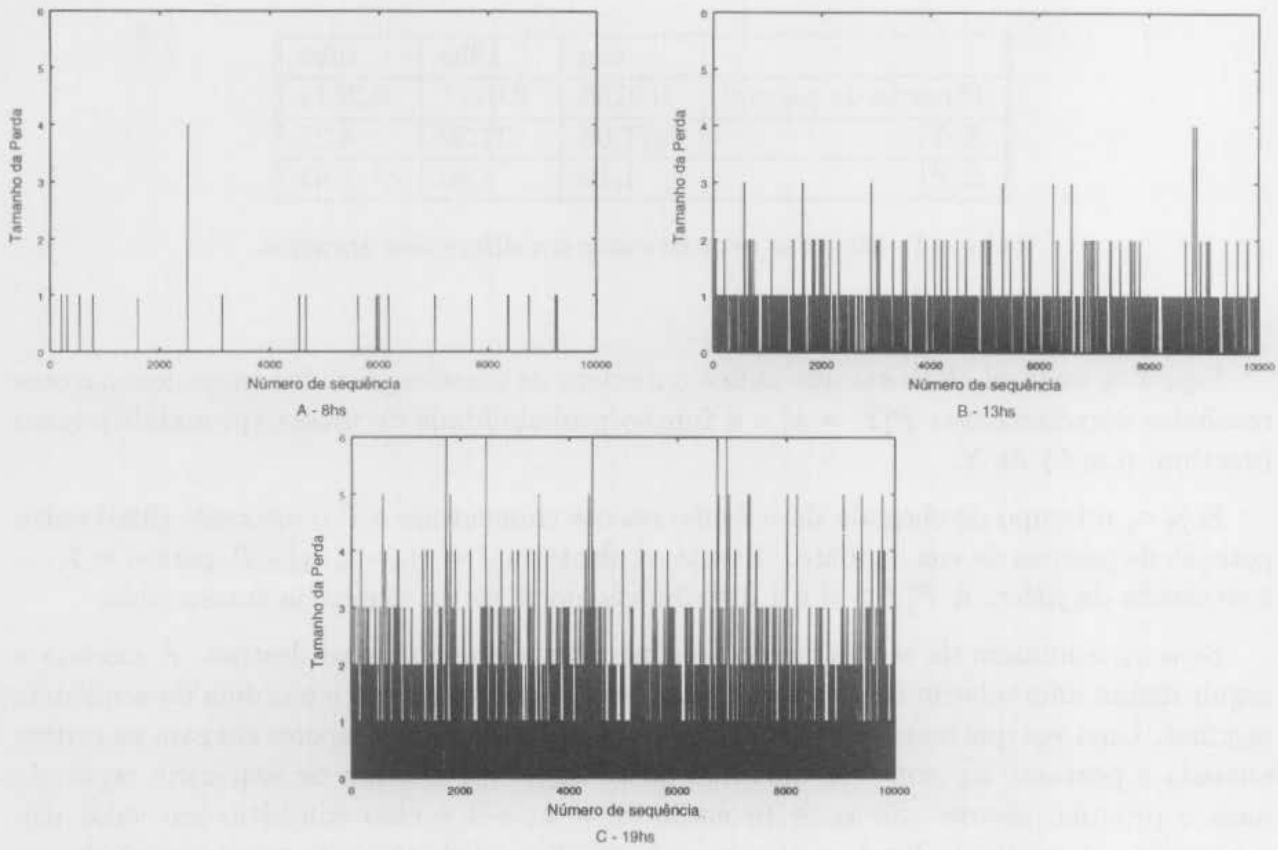

Figura 2: Histogramas de diferentes horários.

	PCM	GSM
P[perda de pacote]	0.0661	0.0711
$E[X]$	21.66	27.30
$E[Y]$	1.13	1.16

Tabela 2: PCM × GSM - média das 13hs.

Outro gráfico que também serve para avaliar a qualidade de serviço oferecida pela rede é o histograma da Figura 3. Neste gráfico apresentamos os valores de $O(x_n)$ para todos os pacotes que chegaram foram de sua ordem correta. Este histograma é referente a um dos testes feito às 19h, quando a rede se apresenta bastante congestionada. Outros testes realizados no mesmo horário apresentaram histogramas semelhantes a este.

Verifica-se que o número de pacotes fora de ordem não é grande. Neste histograma, 5.47% dos pacotes chegaram fora de ordem. Também podemos verificar que a distância entre a posição correta de chegada de um pacote e a posição em que este chegou é pequena, $\bar{O} = 1.08$.

A Figura 4 apresenta a função probabilidade de massa de Y, considerando todos os *traces* realizados às 13hs. Podemos verificar que as curvas PCM e GSM são quase que idênticas indicando que não há diferenças desta medida para as duas codificações. Além disso, a Tabela 2 mostra as outras medidas obtidas, comparando os dois métodos de codificação de voz neste mesmo horário. Em outros horários o comportameno entre os dois métodos também foi parecido.

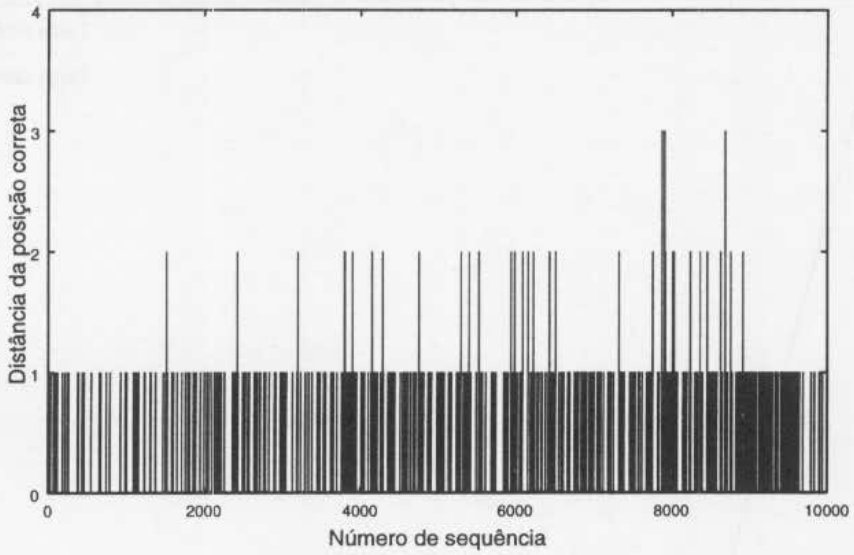

Figura 3: Distância dos pacotes fora de ordem.

4.3 Conclusão sobre resultados obtidos

Os experimentos realizados em [4], mostram que o valor esperado do tamanho da perda de pacotes, $E[Y]$, é baixo. Esta medida apresentou valores entre 1 e 2 para todos os nossos *traces*, mesmo em horários em que a rede estava com uma alta probabilidade de perda de pacotes, na ordem de 25%. Também, como era de se esperar, verifica-se que a rede possui características bastantes distintas, que variam no tempo. Durante o mesmo dia, é possível notar uma grande variação de todas as medidas estatísticas da rede, como mostra a Tabela 1.

O fato da rede possuir um comportamento variável em função do horário, indica que os algoritmos que se propõem a melhorar a qualidade do áudio devem ser adaptativos. Porém, para capturar os efeitos da rede, os algortimos devem, constantemente, avaliar a qualidade de serviço sendo oferecida. Esta tarefa não é trivial, pois questões como quais medidas utilizar para avaliar a qualidade de serviço e como adaptar os algoritmos baseados nessas medidas ainda constituem um problema em aberto.

Outro fato que pode ser observado, é que o tamanho do pacote, 32 ou 160 bytes, pouco influi na probabilidade de perda. Isto pode ser confirmado pela Tabela 2 e pelo gráfico 4. Além disso, todos os outros testes também apresentaram resultados similares para as duas codificações PCM e GSM.

A perda de um pacote pode ser causada por falta de espaço em buffer em um dos roteadores que compõem seu caminho pela rede, ou por erros nos bits do pacote causados pelos meios de transmissão. Erros causados pelos meios de transmissão de hoje são muito raros, e o aumento do tamanho do pacote (de 32 para 160 bytes) pouco influi na incidência desses erros. Este aumento também pouco influi na probabilidade de não haver espaço em buffer, já que o tamanho dos buffers são muito grandes em relação ao tamanho dos pacotes. Também foi observado em [3] que o processo de perda é aleatório quando o tráfego gerado pelos pacotes utiliza uma pequena fração da capacidade total do canal.

Pelas nossas medições, concluímos então, que a utilização da compressão do áudio não diminui a probabilidade de perda nem influi em medidas estatísticas relacionadas com a perda de pacotes. Entretanto, a compressão pode ser utilizada para redução da banda passante necessária para transmitir voz. Isto é necessário quando a capacidade dos meios

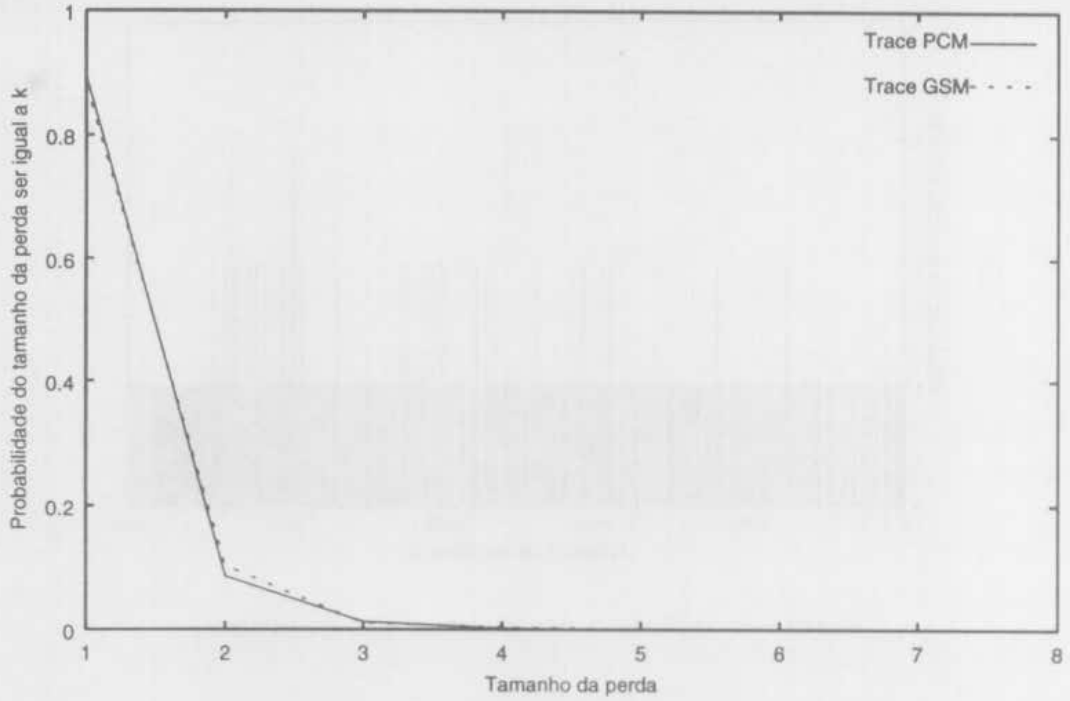

Figura 4: Comparação entre *traces* PCM e GSM.

de comunicação é baixa ou quando se deseja fazer um uso mais adequado dos meios de comunicação disponíveis (o que aconteceria se todos os usuários de aplicativos de voz evitassem o uso de algoritmos de compressão ...).

Para melhorar a qualidade de serviço oferecida pela rede é necessário utilizar, explicitamente, algoritmos que possam reconstruir pacotes perdidos, uma vez que comprimir o pacote não influi na probabilidade de perda do mesmo. Na próxima seção descrevemos o algoritmo por nós proposto e avaliamos sua eficiência.

5 Proposta de um Esquema de Redundância

Na seção 3 foram apresentados dois algoritmos para recuperação de pacotes perdidos utilizando FEC. Veremos agora uma modificação do esquema simples que utiliza ou-exclusivo para compor a redundância.

A modificação consiste em aumentar a redundância que protege cada janela. Desta forma, teremos mais de uma seqüência de pacotes dentro da mesma janela protegidos por um ou-exclusivo. A redundância referente a cada seqüência continua sendo enviada nos primeiros pacotes da próxima janela. Chamaremos de cadeia uma seqüência de pacotes protegida pela mesma redundância. Para formar o ou-exclusivo, as cadeias são intercaladas dentro da janela. A Figura 5 ilustra um exemplo de janela com 6 pacotes e duas redundâncias. Com isto o número de pacotes que pertencem a uma cadeia é inferior ao tamanho da janela.

O *overhead* deste esquema é dado por k/w onde k é o número de redundâncias (ou cadeias) em cada janela e w é o tamanho da janela.

A primeira janela no exemplo da Figura 5 possui duas cadeias de pacotes. A primeira é formada pelos pacotes 1,3,5 e o ou-exclusivo desses pacotes é enviado juntamente com o pacote 7. A segunda cadeia é formada pelos pacotes 2,4,6 e sua redundância é enviada juntamente com o pacote 8. Neste exemplo, os pacotes 7 e 8 possuem o dobro de seu tamanho

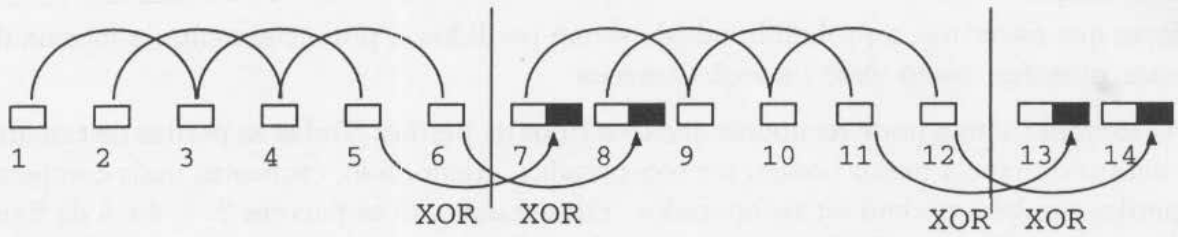

Figura 5: Janela de tamanho 6 com dois pacotes de redundância.

original, pois carregam, além da amostra de áudio, a redundância da janela anterior. Entretanto, como foi visto na seção anterior, o aumento no tamanho do pacote não afeta significativamente sua probabilidade de perda. Utilizando esta modificação, é possível recuperar perdas de tamanho maior que um. Na Figura 5, podemos recuperar qualquer perda (consecutiva) de tamanho dois. Isto só é possível pois as cadeias que compõem a redundância estão intercaladas, e uma perda de tamanho dois irá afetar somente um pacote em cada cadeia.

É importante notar que este esquema possui o mesmo *overhead* que o esquema de ou-exclusivo simples 1:3 (uma redundância e janela com três pacotes). Porém esses dois mecanismos têm comportamento completamente diferente. O esquema 1:3 nunca recupera perdas (consecutivas) de tamanho dois. O esquema 2:6 (duas redundâncias e janela com seis pacotes) não recupera duas perdas de pacote que ocorram na mesma cadeia (i.e. se os pacotes 1 e 5 se perdessem, o esquema 1:3 recuperaria ambos, enquanto o 2:6 não recuperia nenhum).

Vários esquemas com o mesmo *overhead* podem ser propostos, e a eficiência de cada um deles será diferente de acordo com as características da rede. A escolha do esquema mais eficiente para recuperar os pacotes perdidos depende, diretamente, do processo de perda de pacotes.

Uma segunda modificação pode ser feita de forma que dois esquemas distintos de redundância sejam misturados. O objetivo é recuperar perdas simples, de um pacote, e perdas de tamanho maiores do que um. Nesta proposta, todos os pacotes estão cobertos por dois ou-exclusivos, pois pertencem a duas cadeias diferentes, uma de cada esquema.

A Figura 6 ilustra um possível exemplo onde dois esquemas de redundância estão misturados. Todos os pacotes estão sendo protegidos pelos esquemas de redundância 1:2 e 3:6. O *overhead* desta nova proposta é obtido somando os *overheads* dos esquemas misturados. Neste exemplo *overhead* total é de 100% (1/2 + 3/6).

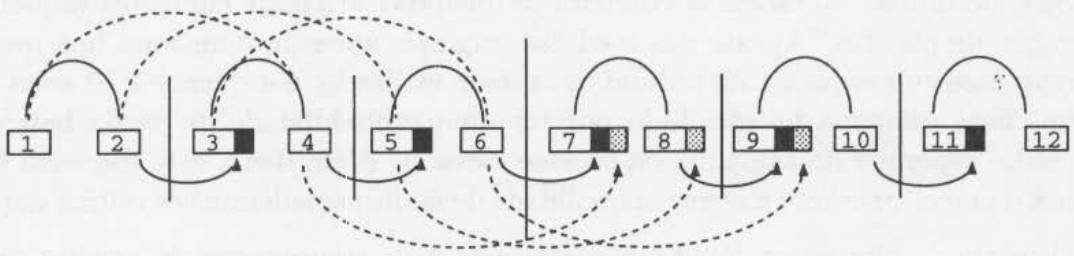

Figura 6: Mistura dos esquemas 1:2 e 3:6.

Os dois esquemas de redundância que foram misturados são independentes, e podem ser executados separadamente para gerar a redundância relativa a cada cadeia. A mistura de dois esquemas dá origem a pacotes maiores que carregam, junto com a amostra de áudio, duas redundâncias, uma de cada esquema. Ressaltamos que, apesar destes pacotes serem

maiores que os outros, a probabilidade de serem perdidos é, potencialmente, a mesma dos pacotes menores, como visto na seção anterior.

O exemplo acima pode recuperar diversos tipos de perdas. Todas as perdas de tamanho um dentro da janela maior podem ser recuperadas. Além disso, esquemas mais complexos de perdas também podem ser recuperados. Por exemplo, se os pacotes 2, 3, 4 e 5 da figura 6 forem perdidos é possível recuperá-los todos. Executamos, primeiramente, o esquema 1:2 e recuperamos o pacote 5. Recuperado o pacote 5, os pacotes 2, 3 e 4 podem ser recuperados utilizando o esquema 3:6, pois pertencem a cadeias diferentes. Lembrando que somente as amostras de áudio podem ser recuperadas.

A ordem de execução dos esquemas de recuperação influi nos pacotes que podem ser recuperados. Se no exemplo acima, executássemos o esquema 3:6 primeiro, então somente os pacotes 3, 4 e 5 seriam recuperados. O pacote 2 não seria recuperado pelo esquema 1:2, pois apesar do pacote 3 ter sido recuperado, a redundância contida nele se perdeu.

Novamente, podemos definir diversas misturas de esquemas de redundância que possuem o mesmo *overhead*, mas que apresentam um comportamento diferente. O esquema de 1:1, visto na seção 3, proposto por [5], também possui um *overhead* de 100%. Porém, este esquema só recupera perdas de tamanho 1, enquanto que no esquema da figura 6, com o mesmo *overhead*, perdas de tamanho 4 podem ser recuperadas!

O fator que determina o esquema de redundância mais eficiente e que deve ser utilizado é o processo de perda de pacotes. As medidas estatísticas referentes às características da rede irão refletir em algum esquema ótimo de redundância.

5.1 Comparação Entre Diferentes Esquemas de Redundância

Para avaliar e comparar os diferentes esquemas de redundância, utilizaremos os *traces* obtidos na seção 4. Através dos arquivos contendo o *trace* da chegada dos pacotes, é possível avaliar como cada esquema de reconstrução se comporta, utilizando-se do algoritmo de recuperação sendo estudado.

Utilizando o arquivo de *trace* com os pacotes já recuperados, podemos calcular todas as medidas estatísticas apresentadas na seção 4 e compará-las com o *trace* original, e com outros esquemas de redundância. Comparando as diversas medidas estatísticas obtidas de diferentes esquemas de redundância, é possível apontar o esquema mais eficiente.

Observa-se que no horário em que a rede está em baixa carga (às 8hs) um esquema simples de redundância, sem muito *overhead*, é suficiente para recuperar quase todos os pacotes perdidos. A tabela 3 compara as medidas originais com dois esquemas de recuperação de pacotes. Apesar das medidas originais apresentarem uma boa qualidade de serviço, caso um esquema de redundância fosse utilizado, o esquema 2:10 seria o mais eficiente. Este esquema foi escolhido por ter uma probabilidade de perda baixa e por ter um valor esperado do tamanho do sucesso elevado. Além disto, este esquema é o que apresenta o menor *overhead* e ofereceu qualidade de áudio semelhante aos outros esquemas.

No horário de alta carga (19hs) os algoritmos para recuperação de pacotes perdidos apresentam um comportamento bastante distinto. Utilizando o mesmo *overhead*, de 100%, esses algoritmos apresentaram características diferentes conforme mostra a tabela 4.

Neste horário, o algoritmo para recuperação que apresentou a melhor eficácia foi a mistura dos esquemas 1:2 e 3:6. Verificamos na Tabela 4 que este apresentou a menor probabilidade de perda e o maior valor esperado do tamanho da rajada de sucesso. A diferença entre a eficiência dos esquemas pode ser vista melhor no histograma da figura 7.

	original	1:1	2:4	2:10
P[perda de pacote]	0.0158	0.0031	0.0045	0.0077
$E[X]$	1277.66	5631.51	5452.53	5579.52
$E[Y]$	1.19	0.77	0.53	0.72

Tabela 3: Medidas estatísticas para diferentes esquemas de redundância às 8hs.

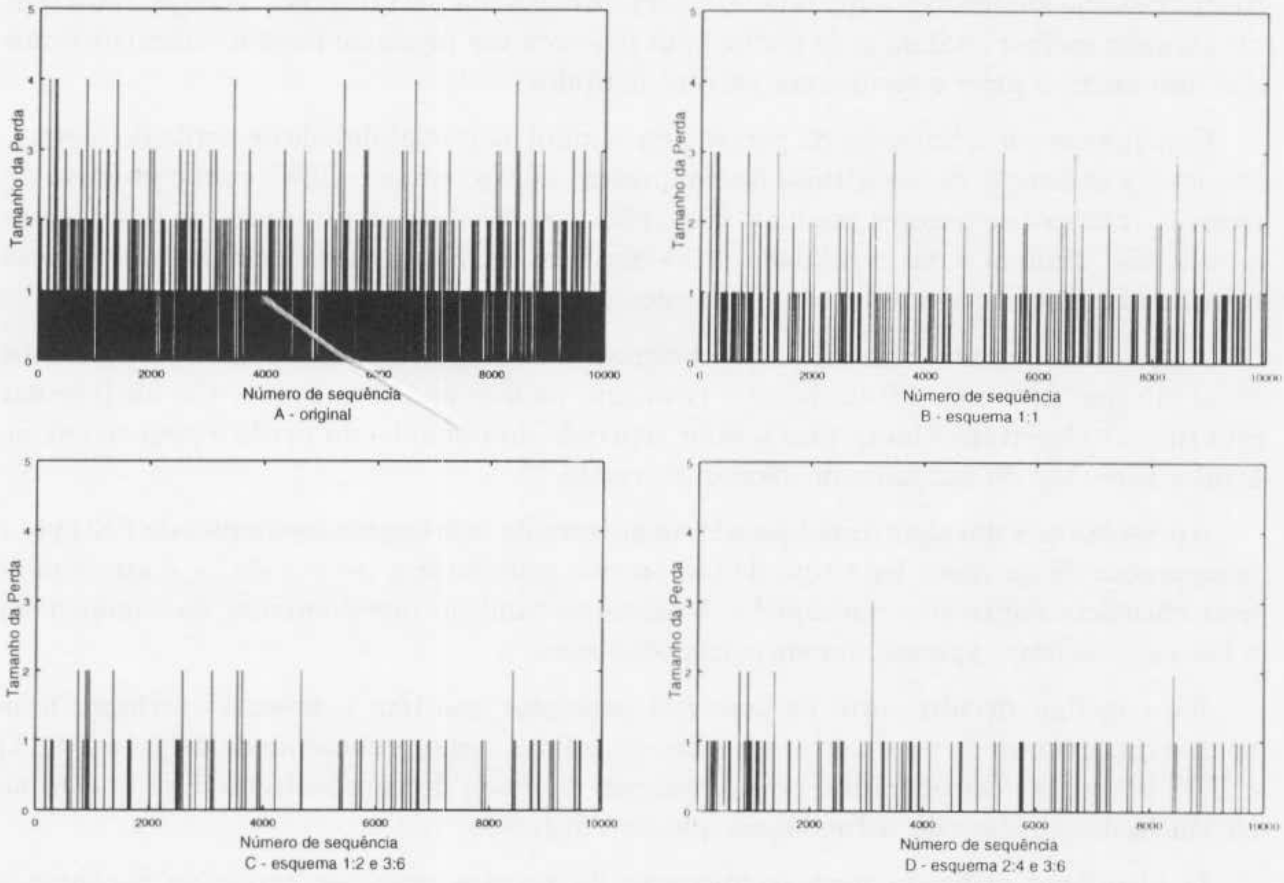

Figura 7: Comportamento de diferentes esquemas de redundância às 19hs.

Na Figura 7 o histograma A foi obtido às 19hs e apresenta as condições da rede sem nenhuma redundância. O histograma B foi gerado utilizando o esquema de recuperação 1:1, que foi proposto em [5] e está implementado na ferramenta FreePhone.

O histograma C foi gerado utilizando a mistura dos esquemas de redundância 1:2 e 3:6 e o histograma D foi gerado utilizando as misturas dos esquemas 2:4 e 3:6. Todos esses esquemas possuem 100% de *overhead*, e podemos verificar que o esquema C apresenta melhores resultados.

Em todos os testes feitos, a mistura de dois esquemas simples de redundância, apresentou resultados melhores do que um esquema único utilizando o mesmo *overhead*.

Não só as medidas estatísticas acima comprovam a eficiência do algoritmo proposto. Enfatizamos que, nos arquivos de *traces* estudados, a melhoria obtida é perfeitamente audível em trechos de conversas pré gravadas.

	original	1:1	1:2 e 3:6	2:4 e 3:6
P[perda de pacote]	0.1477	0.0265	0.0108	0.0122
$E[X]$	42.72	45.92	103.04	84.43
$E[Y]$	1.22	1.26	1.14	1.05

Tabela 4: Medidas estatísticas para diferentes esquemas de redundância às 19hs.

6 Conclusão

Neste trabalho analisamos a qualidade de serviço oferecida pela Internet de hoje. Para tentar garantir melhor qualidade de áudio, aplicativos de voz precisam fazer uso de algoritmos que diminuam o *jitter* e recuperem pacotes perdidos.

Concluímos que o tamanho do pacote pouco influi na probabilidade de perda do mesmo. Por isso, a utilização de algoritmos de compressão de voz, como o GSM, com o objetivo de reduzir o número de pacotes perdidos não é eficiente. Vimos que a compressão dos pacotes de voz não diminui a probabilidade de perda nem melhora outras medidas estatísticas relacionadas ao processo de perda de pacotes.

Algoritmos que utilizam FEC são eficientes na recuperação de pacotes perdidos nos casos em que as perdas são isoladas e envolvem poucos pacotes. Vimos que na Internet este tipo de algoritmo é ideal, pois o valor esperado do tamanho da perda é pequeno e que o valor esperado do tamanho do sucesso é grande.

Apresentamos um algoritmo baseado na mistura de dois esquemas simples de FEC para recuperação de pacotes. Este tipo de mecanismo generaliza a técnica de [5] e apresentou uma eficiência muito superior àquele. Mostramos também que diferentes esquemas, com o mesmo *overhead*, apresentam eficiências distintas.

Para melhor decidir entre os possíveis esquemas que têm o mesmo *overhead*, uma análise do processo de perda deve ser feita, e medidas como probabilidade de perda, $E[X]$ e $E[Y]$ devem ser consideradas, pois o processo de perda de pacotes influi diretamente na eficiência do esquema de redundância que será utilizado.

O algoritmo utilizado para recuperação de pacotes deve ser capaz de capturar a variância das características da rede e se adaptar a essas mudanças. O esquema utilizado para recuperar pacotes deve ser adaptativo. Atualmente, estamos investigando como escolher dinamicamente um algoritmo, dentro da família aqui estudada, de forma a minimizar as perdas e otimizar os recursos da rede.

Referências

[1] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, second edition, 1992.

[2] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, 1994.

[3] J-C. Bolot. Characterizing end-to-end packet delay and loss in the Internet. In *Proc. ACM Sigcomm*, pages 289–298, Sep. 1993.

[4] J-C. Bolot and A. Vega-García. The case for FEC-based error control for packet audio in the Internet. to appear in ACM Multimedia Systems.

[5] J-C. Bolot and A. Vega-García. Control mechanisms for packet audio in the Internet. In *Proc. IEEE Infocom*, Mar. 1996.

[6] Informações sobre o MBONE pode ser encontradas na URL http://www.mbone.com.

[7] M.C. Diniz and Edmundo de Souza e Silva. Análise de mecanismos de controle de jitter em redes ATM. In *XXIII Seminário Integrado de Software e Hardware*, pages 303–314, Aug. 1996.

[8] Daniel Ratton Figueiredo, Flavio Pimentel Duarte, and Edmundo de Souza e Silva. Implementação de um aplicativo de voz com ánalise e caracterização de seu tráfego. In *XXIV Seminário Integrado de Software e Hardware*, pages 531–540, Aug. 1997.

[9] Vicky Hardman and Isidor Kouvelas. URL http://www-mice.cs.ucl.ac.uk/mice/rat. Projects MICE and ReLaTe at the Department of Computer Science, University College London.

[10] Van Jacobson and Steven McCanne. URL http://www-nrg.ee.lbl.gov/vat. Lawrence Berkeley National Laboratory - University of California, Berkeley.

[11] N.S. Jayant. Effects of packet loss on waveform coded speech. In *Proc. Fifth Int. Conference on Computer Communications*, pages 275–280, Oct. 1980.

[12] The MICE project. URL http://www-mice.cs.ucl.ac.uk/mice/mice_home.html. Multimedia Integrated Conferencing for European Researchers, University College London.

[13] Moe Rahnema. Overview of the GSM system and protocol architecture. *IEEE Communications Magazine*, April 1993.

[14] H. Schulzrinne. RFC 1890. RTP profile for audio and video conferences with minimal control. Technical report, Audio-Video Transport Working Group and Network Working Group, Feb. 1996.

[15] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RFC 1889. RTP: A transport for real time applications. Technical report, Audio-Video Transport Working Group and Network Working Group, Feb. 1996.

[16] Henning Schulzrinne. Voice communication across the Internet: A network voice terminal. Technical report, Dept. of Electrical Engineering, University of Massachussets, Jul. 1992.

[17] John Scourias. Overview of the global system for mobile communications. Technical report, University of Waterloo, 1995.

[18] Andrés Vega-García and Sacha Fosse-Parisis. URL http://zenon.inria.fr/rodeo/fphone. High-Speed Networking Group at INRIA in Sophia Antipolis.