

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

**FERRAMENTA PARA COMUNICAÇÃO VOIP USANDO
O PADRÃO H.323 EM REDES COM SERVIDORES NAT**

RODRIGO JEAN DE SOUZA
WAGNER CUNHA

FLORIANÓPOLIS, NOVEMBRO DE 2004

RODRIGO JEAN DE SOUZA

WAGNER CUNHA

**FERRAMENTA PARA COMUNICAÇÃO VOIP USANDO
O PADRÃO H.323 EM REDES COM SERVIDORES NAT**

Trabalho de Conclusão de Curso apresentado ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Sistemas de Informação, orientado pelo Professor Dr. Roberto Willrich

FLORIANÓPOLIS, NOVEMBRO DE 2004

RODRIGO JEAN DE SOUZA
WAGNER CUNHA

**FERRAMENTA PARA COMUNICAÇÃO VOIP USANDO
O PADRÃO H.323 EM REDES COM SERVIDORES NAT**

BANCA EXAMINADORA

Prof. Roberto Wilrich, Dr.
Orientador

Prof. Vítório Mazzola, Dr.

Prof. Mário A. R. Dantas, Dr.

FLORIANÓPOLIS, NOVEMBRO DE 2004

“Não importa o tamanho da montanha, ela não pode tapar o sol”
Provérbio chinês

Agradecimentos

Aos nossos pais e irmãos que apostaram em nosso potencial durante toda a faculdade, sempre nos dando apoio em todas as situações.

À todos os professores que participaram deste curso e que contribuíram para a nossa formação acadêmica e profissional.

Ao nosso professor e orientador Roberto Willrich, pela atenção dispensada durante a realização deste trabalho.

Aos nossos colegas de curso, por termos compartilhado angústias, aprendizados e vitórias durante todo o percurso, em especial ao nosso amigo Royquener Reuter pelo companheirismo.

Àqueles que colaboraram direta ou indiretamente na realização deste trabalho, em especial aos amigos Thiago Paim de Campos, Jean Holetz, Carlos Schwochow, Paulo Schwochow e Anita Martins.

Resumo

O uso da Internet para a comunicação de voz vem crescendo muito nos últimos anos principalmente pelo alto custo da telefonia convencional. Economizar é a prioridade dos usuários, que também são estimulados a aderir a esta modalidade de comunicação pela proximidade cada vez maior com a Internet.

O modelo de transmissão de mensagens de voz mais difundido é o H.323, desenvolvido pelo International Telecom Union (ITU-T), que define padrões para redes de computadores e telecomunicações.

Com o aumento contínuo do número de usuários na Internet, os endereços IP públicos tornaram-se insuficientes. Para resolver esse problema, passaram a ser usados servidores NAT, os quais acabaram dificultando a operação do padrão H.323 que é complexo e exige comunicação ponto-a-ponto. Nos NATs não é possível fazer esse tipo de comunicação, por isso, o H.323 não funciona para os usuários em redes com dispositivos NAT.

O presente trabalho propõe uma solução eficiente para transpor os dispositivos NAT e estabelecer comunicação entre dois pontos de rede separados por tradutores de endereços, o que é necessário para o funcionamento do H.323.

Palavras-chave: Voz sobre IP, Protocolo H.323, NAT, Dispositivos de Tradução de Endereço.

Lista de Figuras

| | |
|---|----|
| Figura 1 – Processo de amostragem | 4 |
| Figura 2 – Processo de quantificação e codificação | 5 |
| Figura 3 – Qualidade dos codecs através da escala MOS | 7 |
| Figura 4 – Datagrama UDP | 11 |
| Figura 5 – Padrões para controle e transmissão de voz | 11 |
| Figura 6 – Escopo do H.323 | 16 |
| Figura 7 – Cone NAT | 21 |
| Figura 8 – Symmetric NAT | 22 |
| Figura 9 – Relaying | 24 |
| Figura 10 – Conexão reversa | 24 |
| Figura 11 – UDP Hole Punching em diferentes NATs | 25 |
| Figura 12 – UDP Hole Punching em um mesmo NAT | 26 |
| Figura 13 – UDP Hole Punching em múltiplos NATs | 28 |
| Figura 14 – UDP Hole Punching: Predição de porta UDP | 30 |
| Figura 15 – UDP Hole Punching: Técnica “N+1” | 30 |
| Figura 16 – Funcionamento da ferramenta | 37 |

Lista de Tabelas

| | |
|--|----|
| Tabela 1 – Codecs recomendados pelo ITU-T | 6 |
| Tabela 2 – H.323: Fluxo de mensagens | 16 |
| Tabela 3 – Faixas de endereços privados | 19 |
| Tabela 4 – Tabela de mapeamento dos dispositivos NAT | 21 |
| Tabela 5 – Portas de entrada utilizadas pelo protocolo H.323 | 33 |
| Tabela 6 – Ambiente de testes | 41 |

Índice

| | | |
|---------------|--|-----------|
| 1 | Introdução | 1 |
| 2 | Representação Digital de Voz | 3 |
| 2.1. | Processo de Digitalização de Áudio..... | 3 |
| 2.1.1. | Amostragem | 3 |
| 2.1.2. | Quantificação | 4 |
| 2.1.3. | Codificação..... | 4 |
| 2.1.4. | A Taxa de Bits Gerada pela Codificação..... | 5 |
| 2.1.5. | Codecs de digitalização da voz..... | 6 |
| 3 | Transmissão de Voz sobre IP | 8 |
| 3.1. | TCP vs UDP | 10 |
| 3.2. | UDP..... | 10 |
| 3.3. | Protocolos de transmissão de voz..... | 11 |
| 3.3.1. | SIP | 12 |
| 3.3.2. | MGCP..... | 12 |
| 3.3.3. | Megaco | 12 |
| 3.3.4. | Protocolo H.323..... | 12 |
| 3.4. | RTP/RCTP – Real Time Protocol/Real Time Control Protocol..... | 17 |
| 4 | NAT (Network Address Translation) | 19 |
| 4.1. | Tipos de NAT | 21 |
| 4.1.1. | Cone NAT | 21 |
| 4.1.2. | Symmetric NAT | 22 |
| 4.1.3. | NAT Cone Full..... | 22 |
| 4.1.4. | Restricted Cone NAT | 22 |
| 4.1.5. | Port-Restricted Cone NAT | 23 |
| 4.2. | Técnicas para comunicação ponto-a-ponto através do NAT..... | 23 |
| 4.2.1. | Relaying..... | 23 |
| 4.2.2. | Conexão reversa | 24 |
| 4.2.3. | UDP Hole Punching | 25 |
| 4.2.3.1. | Cenários..... | 25 |
| 4.2.3.2. | Consistência das reservas de portas..... | 29 |

| | |
|---|-----------|
| 4.2.3.3. Predição da porta UDP | 29 |
| 5 Incompatibilidade do protocolo H.323 com os servidores NAT | 32 |
| 5.1. Portas dinâmicas | 32 |
| 5.2. Área de dados | 33 |
| 6 Funcionamento da ferramenta | 35 |
| 6.1. Tunelamento de pacotes | 35 |
| 6.2. Detalhes do funcionamento | 36 |
| 6.2.1. Estabelecimento do canal UDP | 38 |
| 6.2.2. Descoberta dos endereços IP públicos | 38 |
| 6.2.3. Alteração do endereço IP de origem no cabeçalho dos pacotes | 38 |
| 6.2.4. Tunelamento dos pacotes | 39 |
| 6.2.5. Alteração do endereço IP de destino no cabeçalho dos pacotes | 39 |
| 6.2.6. Injeção do pacote recebido | 39 |
| 6.2.7. Alteração dos endereços IP dentro dos dados do pacote | 39 |
| 6.3. Implementação da ferramenta | 40 |
| 6.4. Testes realizados | 41 |
| 7 Conclusão | 42 |
| 8 Referências Bibliográficas | 43 |
| 9 Anexos | 45 |
| 9.1. Artigo | 45 |
| 9.2. Código fonte | 52 |

1 Introdução

O crescimento da Internet, que nos últimos anos se popularizou como um meio de comunicação de alcance global e de baixo custo, aliado a um aumento constante na velocidade dos enlaces utilizados, criou um ambiente que viabiliza a criação de uma tecnologia para aplicações de transmissão de áudio digital em tempo real. Esta tecnologia chama-se Voz sobre IP ou VoIP [Monteiro 2000].

Em redes que não oferecem garantia de qualidade de serviço, como é o caso da Internet, o VoIP sofre uma série de problemas comuns às transmissões em tempo real sobre um protocolo não confiável como atrasos variáveis e perda de pacotes [Comer 2000]. Soluções baseadas em *Internet Protocol* (IP) têm sido propostas para substituir com inúmeras vantagens os modelos de telefonia convencional. Com a utilização de redes de pacotes para transportar voz elimina-se a necessidade da presença de um circuito. Assim, a voz é empacotada e transmitida por meio de redes de computadores juntamente com os dados. O IP é o protocolo usado nesse processo.

Atualmente o padrão H.323, proposto pela International Telecom Union (ITU-T), é o mais difundido no mercado para transmitir voz através da Internet. Esse modelo, focado na conexão e no controle da chamada, é uma pilha de protocolos, que são separados da transmissão de conteúdo (voz) entre os computadores.

O número crescente de usuários da Internet gerou o problema da falta de endereços IP, contornado com os servidores NAT (*Network Address Translation*), que possibilitam o acesso à Internet para diversas máquinas em uma rede local com apenas um endereço IP válido.

Os servidores NAT, no entanto, causaram problemas na comunicação de voz do padrão H.323 já que esse protocolo necessita de comunicação ponto-a-ponto. Os servidores NAT, em função de seus procedimentos de funcionamento, não oferecem comunicação direta entre o usuário da rede interna e a Internet, o que inviabiliza a utilização do padrão H.323.

O objetivo deste trabalho é justamente analisar a comunicação de voz através da Internet e propor uma ferramenta para que os usuários do padrão H.323 possam se comunicar mesmo que façam uso de servidores NAT.

A implementação dessa ferramenta baseia-se em técnicas de tunelamento e filtragem de pacotes e *UDP Hole Punching* para criar o canal de comunicação que será usado para estabelecer a sessão H.323.

2 Representação Digital de Voz

A voz humana é uma onda sonora que pode ser transmitida por diversos meios - sólidos, líquidos ou gasosos. Quanto mais denso o meio, melhor a transmissão. O som é audível aos seres humanos quando a sua frequência varia de 20 a 20.000 Hertz (Hz). As principais frequências da voz humana estão na faixa de 300 a 3.400 Hz.

No processo de emissão, transmissão e recepção de mensagens de voz sobre IP, o essencial é que o computador entenda a voz dos participantes para poder enviá-la pelo canal de comunicação. Para que isso seja possível, é preciso fazer a digitalização da voz, em que o sinal analógico criado pelo microfone é transformado em um sinal digital que pode ser manipulado pelo computador.

Terminado esse processo, os bits que carregam as informações da voz são enviados ao computador de destino, onde novamente são convertidos para o sinal analógico e emitidos pelas caixas de som de forma perceptível ao ouvido humano.

2.1. Processo de Digitalização de Áudio

2.1.1. Amostragem

No processo de amostragem, feito com um clock, um conjunto discreto de valores analógicos é amostrado em intervalos temporais de periodicidade constante. A frequência de relógio é chamada de taxa ou frequência de amostragem. O valor fatiado é mantido constante até o próximo intervalo. Isso é realizado por meio de circuitos *sampling and hold*. Cada parte é analógica em amplitude, ou seja, possui qualquer valor em um domínio contínuo, mas isso é discreto no tempo - dentro de cada intervalo, a amostra tem apenas um valor.

O Teorema de Nyquist [Matsuda 2002] afirma que se um sinal analógico contém componentes de frequência até f Hz, a taxa de amostragem deve ser de no mínimo $2f$ Hz, mas, na prática, essa frequência é um pouco maior. Um exemplo é que os principais componentes de frequência da voz humana estão dentro de 3,1 kHz, e, por isso, os sistemas de telefonia analógicos limitam o sinal transmitido a 3,1 kHz. Mas também é

comum o uso de uma frequência de amostragem de 8 kHz para a conversão desse sinal em digital.

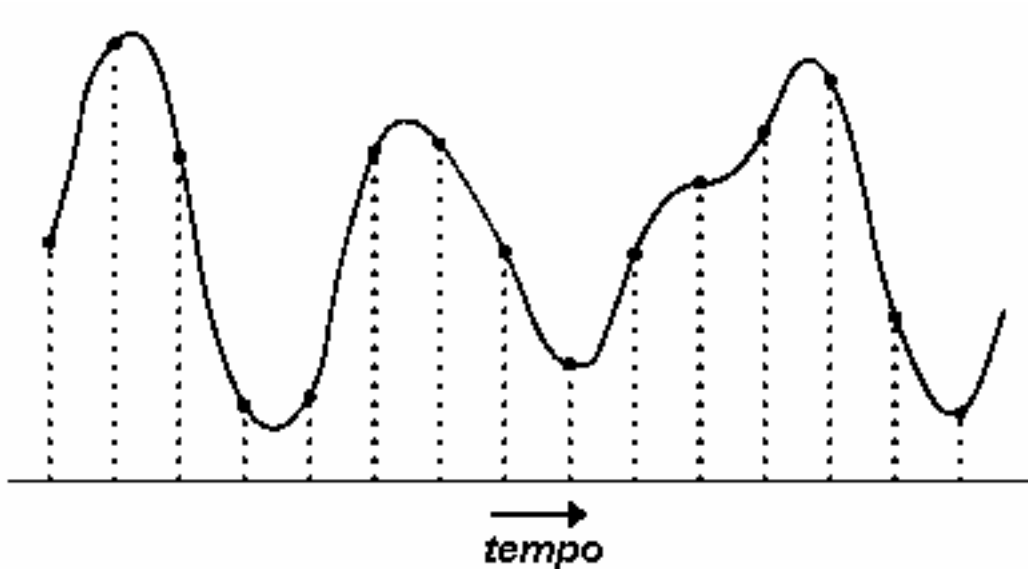


Figura 1 – Processo de amostragem

2.1.2. Quantificação

Na quantificação os valores de amostras contínuas são convertidos em valores discretos. Nesse processo, o domínio do sinal é dividido em um número fixo de intervalos numerados de mesmo tamanho. A cada amostra dentro de um intervalo, é atribuído o valor do intervalo. Passo de quantificação é o nome do tamanho desse intervalo. A técnica que utiliza o mesmo passo de quantificação é chamada modulação PCM (*Pulse Coded Modulation*).

2.1.3. Codificação

Na codificação, um conjunto de dígitos binários, chamado de *code-word*, é associado a cada valor quantificado. Usando-se 3 bits é possível codificar até 8 níveis. Cada amostra, então, é representada por essa quantidade de bits.

Em aplicações de telefonia, a digitalização da voz humana pode utilizar 16 bits por amostra, implicando em 65.536 (2^{16}) passos de quantificação. Em outras vezes, apenas 8 quantificações por bits são necessárias, produzindo 256 passos de quantificação.

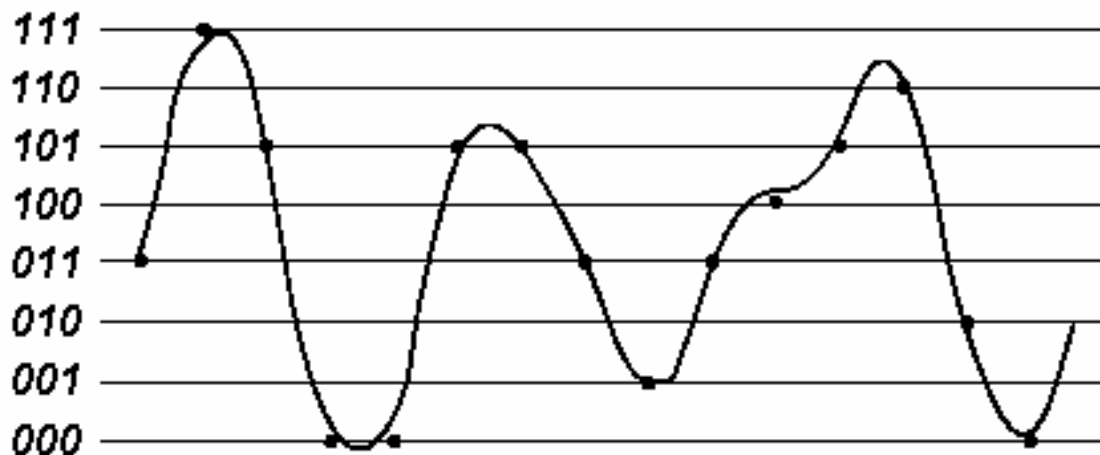


Figura 2 – Processo de quantificação e codificação

2.1.4. A Taxa de Bits Gerada pela Codificação

O produto entre a taxa de amostragem e o número de bits usados no processo de quantificação resulta na taxa de bits. Se uma frequência for de 8 kHz e 8 bits por amostra, a taxa de bits deverá ser de 64 kbps.

Continuamente deve ser tirada uma amostra do áudio digital em uma taxa fixa. Cada amostra é representada por um número fixo de bits. Quanto maior a taxa e o número de bits por amostragem, maior é a qualidade do áudio restituído, mas maior também é a taxa de bits gerada.

2.1.5. Codecs de digitalização da voz

Reduzir a taxa de transmissão de bits, ao mesmo tempo em que mantém o máximo possível de qualidade subjetiva original do sinal, é o objetivo dos codificadores de voz. A seguir estão os *codecs* de digitalização de voz recomendados pelo ITU-T:

Tabela 1 – Codecs recomendados pelo ITU-T

| Recomendação | Codificação | Taxa (Kpbs) | Quadro/look-ahead (ms) | Ano |
|--------------|-------------|-------------|------------------------|------|
| G.711 | PCM | 64 | 0,125 / 0 | 1972 |
| G.726 | ADPCM | 40,32,24,16 | 0,125 / 0 | 1990 |
| G.728 | LD-CELP | 16 | 0,625 / 0 | 1992 |
| G.729A | CS-ACELP | 8 | 10 / 5 | 1996 |
| G.723 1 | MP-MLQ | 6,3 | 30 / 7,5 | 1996 |
| G.723 1 | ACELP | 5,3 | 30 / 7,5 | 1996 |

A qualidade de um sinal de voz pode ser determinada através da escala MOS (*Mean Opinion Score*), na qual os ouvintes atribuem valores entre 0 e 5 ao sinal de voz de acordo com a sua opinião sobre a qualidade do sinal.

O gráfico abaixo compara os *codecs* anteriormente apresentados através da escala MOS:

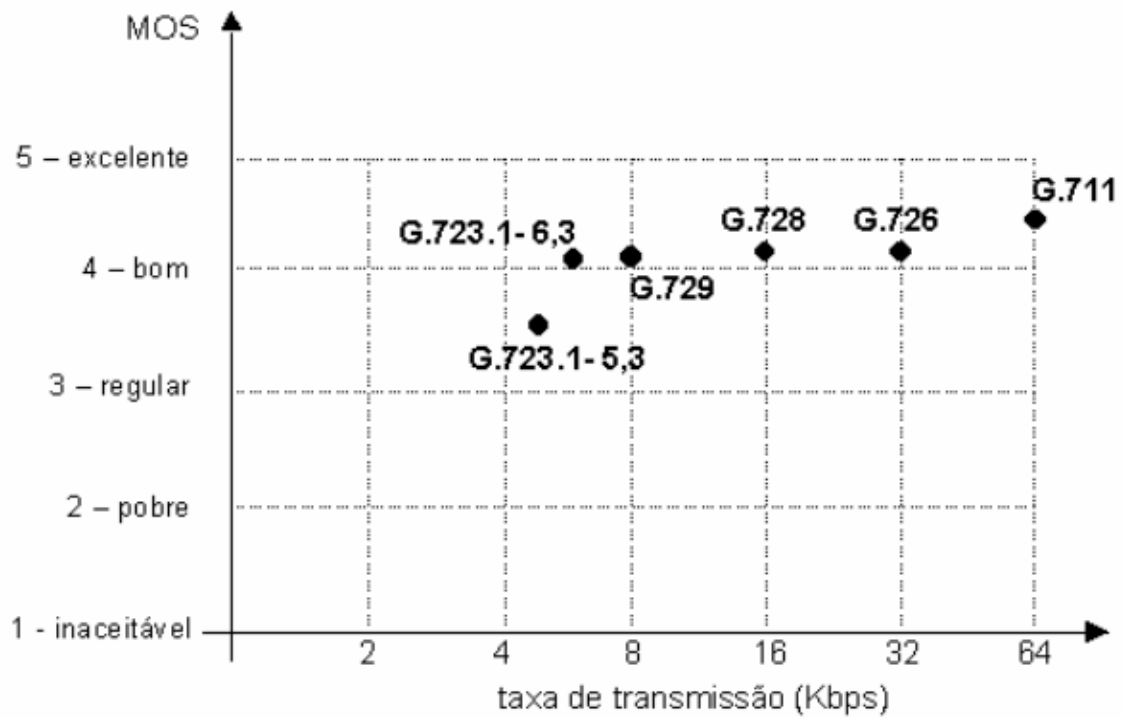


Figura 3 – Qualidade dos codecs através da escala MOS

Pode-se observar que os *codecs* G.723.1 e G.729 são os mais indicados para a compressão de pacotes para transmissão via Internet. Isso porque o tamanho do pacote foi bastante reduzido por meio da compactação. O *codec* G.728 pode ser usado para uma melhor qualidade de voz, mas gera um aumento na taxa de transmissão.

3 Transmissão de Voz sobre IP

Transformar voz em mais uma aplicação IP dentro de uma rede de dados que utilize IP como protocolo de nível de rede. Esse é o conceito da tecnologia Voz sobre IP, que permite, de acordo com [Monteiro 2001], a digitalização e a codificação de voz e o empacotamento de dados IP para a transmissão em uma rede IP. Tudo isso dentro de uma mesma rede, anárquica e dispersa, a custos relativamente baixos. Por causa do volume de dados gerados por uma aplicação VoIP, essa tecnologia é mais facilmente encontrada em redes corporativas privadas. Se a rede base para transporte desta aplicação for a Internet, não é aconselhável o uso para fins profissionais, já que o protocolo IP não oferece padrões de Qualidade de Serviço (QoS). Isso compromete a clareza da voz, que é dependente do tráfego de dados existentes no momento da conversa.

Ao comparar aplicações de dados, excluindo-se multimídia, e aplicações de voz, nota-se que a principal diferença é que uma aplicação de voz é sensível ao atraso. Em uma rede IP não é possível garantir um atraso constante, o que pode transformar uma aplicação de voz em tempo real, como por exemplo, uma ligação telefônica, em um serviço de baixa qualidade com a voz entrecortada e muitas vezes ininteligível. A capacidade de uma rede em oferecer atrasos constantes é chamado de QoS.

[Shulzrinne 2001] afirma que a tecnologia de Voz sobre IP vem se popularizando rapidamente. Institutos de pesquisa apontam crescimento exponencial da utilização dessa tecnologia, principalmente em médias e grandes empresas.

[Monteiro 2000] diz que transportar voz sobre uma rede de dados não é uma solução recente. Há anos a Embratel comercializa soluções baseadas em equipamentos TDM, mas o custo é alto, porque há grande desperdício de largura de banda.

Soluções baseadas em *Frame-Relay* também já existem há algum tempo. Mas como *Frame-Relay* é um protocolo de enlace, a aplicação de voz fica disponível apenas nos pontos remotos, onde é implantar essa tecnologia. A tecnologia ATM também vem

sendo adotada para este fim, mas o seu crescimento não tem sido maior porque o custo também é alto e o nível de capilaridade é baixo.

Segundo [Shulzrinne 2000], basicamente três elementos contaram para o fortalecimento da tecnologia de Voz Sobre IP: o desenvolvimento e a padronização de protocolo que permite QoS em redes IP; a produção acelerada de métodos de compressão de voz e a explosão da Internet.

Há alguns anos já é esperado, independente da tecnologia adotada, o movimento de integração entre voz e dados na mesma infra-estrutura de rede. As vantagens são claras, pois os custos envolvidos na manutenção de equipes técnicas, infra-estruturas diferenciadas e ligações internacionais são reduzidos com a integração. Contribuíram muito para Voz sobre IP (VoIP) tornar-se uma realidade o aumento do leque de novas aplicações, da disseminação de microcomputadores pessoais (para funcionamento como terminal multimídia), das capilaridades redes IP e da banda de transmissão disponível para o usuário.

O uso da VoIP em larga escala só é freado porque é grande a diferença de preço entre o terminal telefônico convencional e um equipamento para uso de VoIP. Além disso, a alta disponibilidade das redes telefônicas convencionais aliadas à falta de qualidade de serviço e de confiabilidade da rede, originalmente herdada do IP, são aspectos de peso na comparação entre as soluções.

Mesmo com as desvantagens e com aos enormes benefícios introduzidos pela integração entre telefonia e IP, ainda é almejada uma mudança no cenário atual da comunicação de voz e de dados atual. Espera-se por uma realidade integrada em larga escala, onde os meios de transmissão deverão servir aos dois “mundos” de forma transparente ao usuário.

3.1. TCP vs UDP

O protocolo UDP foi escolhido em relação ao TCP porque para as aplicações VoIP a entrega dos dados em tempo é mais importante que a entrega dos dados sem erros. Pacotes TCP são entregues em ordem e sem erros. O protocolo retransmite os pacotes defeituosos ou faltantes automaticamente. Isto provoca atrasos que dificulta o controle de *jitter*. Já o protocolo UDP não garante nem a ordem, nem a correção, nem a integridade dos pacotes, que podem chegar corrompidos ou simplesmente não chegar. Supondo que apenas um pequeno percentual de pacotes cheguem danificados, o protocolo UDP/IP é utilizado para a comunicação VoIP.

3.2. UDP

O UDP - uma interface para o protocolo IP - transmite os dados de forma não orientada à conexão e é restringido a portas e *sockets*. Esse protocolo substitui o protocolo TCP quando a transferência de dados não precisa estar submetida a serviços como controle de fluxo. Servir de multiplexador ou de demultiplexador para o tráfego de informações do IP é a função básica do UDP. Assim como o TCP, o UDP trabalha com portas que orientam adequadamente o tráfego de informação a cada aplicação de nível superior. As portas são as seguintes:

- a) Porta de destino: é uma parte do datagrama (uma extremidade) que indica o aplicativo ao qual deve-se enviar a informação que chega;
- b) Porta de origem: localiza-se no outro extremo do datagrama e indica o aplicativo que enviou a mensagem. Pode ser usado para um reenvio ou, quando não utilizado, é preenchido com zeros.

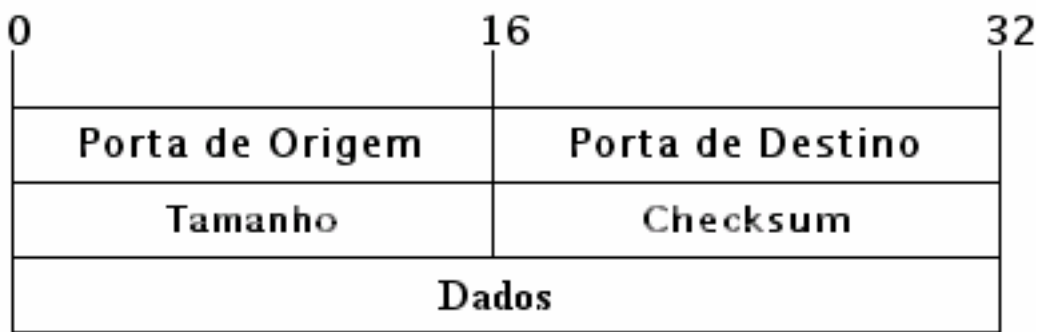


Figura 4 – Datagrama UDP

3.3. Protocolos de transmissão de voz

Para enviar os pacotes de voz de um ponto ao outro, é necessário aplicar técnicas após a digitalização. Além do envio, deve haver um controle de qualidade para que a comunicação seja feita de forma inteligível. O controle do atraso e da chegada de pacotes faz parte das técnicas de transmissão voz. Vários padrões são utilizados para transmissão e controle de pacotes de voz. O padrão H.323 será referência neste trabalho, já que é o mais comum. Veja abaixo um diagrama em blocos com os padrões de transmissão de voz mais conhecidos:

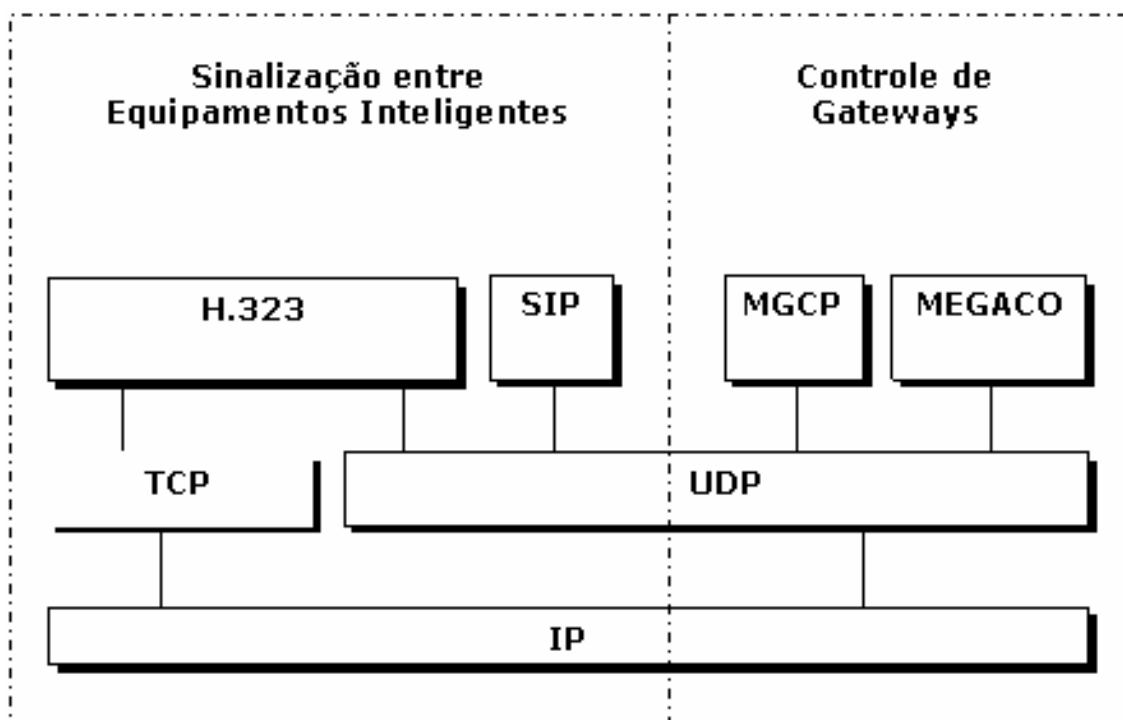


Figura 5 – Padrões para controle e transmissão de voz

3.3.1. SIP

Foi a IETF (Internet Engineering Task Force) que apresentou o SIP, um protocolo de sinalização mais avançado, porém de trato mais simplificado. Extensível e leve, mas escalável, o SIP tem como principal característica a forte integração com TCP/IP. No lançamento mais recente do sistema operacional Windows-XP pela Microsoft a implementação de uma pilha de SIP atestou o papel e a importância que o protocolo poderá assumir nas redes convergentes.

3.3.2. MGCP

Muito utilizado para comunicação entre *SoftSwitches* e *Media Gateways*, o MGCP é um padrão "de facto". No entanto, a comunicação entre *Media Gateways* é feita em geral por uma variação de SIP, conhecida como SIP-T.

3.3.3. Megaco

Baseado no sistema H.248 do ITU, o Megaco representa uma evolução do MGCP. Os dois são complementares ao SIP e ao H.323, embora também possam controlar dispositivos não-inteligentes, por meio de implementações específicas.

3.3.4. Protocolo H.323

[Craig 2001] diz que diariamente surgem novas oportunidades e novos desafios com os avanços nos sistemas de comunicações e com a popularização de serviços sobre as redes de computadores. Antigamente, os aplicativos de intercâmbio de dados não precisavam de suporte para executar, por exemplo, operações em tempo real. Mas essa realidade mudou.

Nesse contexto temos o padrão H.323, que descreve como áudio, vídeo, dados e informações de controle podem ser gerenciados em uma rede baseada em pacotes para

disponibilizar serviços de conversação. O padrão H.323 provê uma arquitetura de ação de dados multimídia, para redes baseadas no protocolo IP. O H.323 permite ainda a operação conjunta de produtos de multimídia e aplicações de fabricantes diferentes. Assim, os usuários podem se comunicar sem preocupação com a velocidade da rede.

A International Telecom Union (ITU-T), organismo que define padrões para redes de computadores e telecomunicações, recomenda o H.323. Estas redes incluem TCP/IP em cima de *Ethernet*, *Fast Ethernet* e *Token Ring*. A primeira versão da especificação do H.323 foi aprovada em 1996 pelo Grupo de estudos 16 do ITU e a segunda em janeiro de 1998. O H.323 faz parte de uma série de padrões de comunicações que permitem vídeo conferência e VoIP através de redes.

A flexibilidade, de acordo com [Domingues 2000], é uma das principais características do H.323, que permite a aplicação de voz, vídeo conferência e multimídia. Esse padrão está se tornando popular no mercado corporativo por diversas razões:

a) O H.323 determina padrões de Voz para uma infra-estrutura existente, além de ser projetada para compensar o efeito de latência em LANs, permitindo que os clientes usem aplicações de voz sem mudar a infra-estrutura de rede;

b) Estão cada vez mais velozes as redes baseadas em IP, além da largura de banda para redes com arquitetura *Ethernet* estarem migrando de 10 Mbps para 100 Mbps. A *Gigabit Ethernet* também está progredindo no mercado;

c) O H.323 provê padrões de interoperabilidade entre LANs e outras redes;

d) Como o fluxo de dados em redes pode ser administrado, com o H.323, o gerente de rede pode restringir a quantidade de largura de banda disponível para conferências e voz. O suporte à comunicação *Multicast* ainda reduz exigências de largura de banda;

e) Muitas empresas de comunicação e organizações, incluindo a Intel, Microsoft, Cisco e IBM, apóiam a especificação H.323. Os esforços destas companhias estão gerando um nível mais alto de consciência no mercado.

[Nóbrega 2001] diz que a ITU-T propôs o padrão H.323, sendo mais difundido atualmente, especialmente por ser o precursor da Telefonia IP e ser o primeiro padrão a tratar deste tema. As principais características deste padrão são:

- a) Especifica algoritmos padrões de compressão, conhecidos como áudio *codecs* ou *vocoders*, que devem ser implementados para garantir compatibilidade;
- b) Cria protocolos para controlar as chamadas, estabelecer canais de comunicação e negociar qualidade de serviço;
- c) Possibilita interoperabilidade com outros terminais de voz, como telefonia convencional, RDSI e voz sobre ATM, entre outros, permitindo assim a construção de *gateways*;
- d) Descreve elementos ativos do sistema e suas funções.

Em suas diversas funcionalidades, o protocolo H.323 utiliza uma família de recomendações ITU-T: H.225 para conexão, H.245 para controle, H.332 para conferências, H.335 para segurança, H.246 para interoperabilidade com RTPC e a série H.450.x para serviços suplementares. Todos esses padrões fazem parte da série H de recomendações. Também verifica-se no H.323, os elementos que compõem uma rede de telefonia IP. Estes elementos podem ser definidos da seguinte forma:

Terminal H.323 – Computador onde está implementado o serviço de telefonia IP. Atua como terminal de serviço de telefonia IP, como terminal de voz, vídeo e dados, através de recursos multimídia. Esses são os clientes da LAN que fornecem comunicação em tempo real nas duas direções. Todos os terminais H.323 têm que suportar H.245, Q.931, *Registration, Admission and Status* (RAS) e RTP. Os terminais H.323 também podem incluir o protocolo de conferência de dados T.120, codificadores de vídeo e suporte para MCU. Um terminal H.323 pode se comunicar com outro terminal, um *gateway* ou um MCU;

Gateway H.323 – Elemento que permite a interoperabilidade entre duas redes e é situado entre uma rede IP e uma rede de telecomunicações, como o sistema telefônico convencional (RTPC), a rede integrada de serviços digitais (RDSI) e a rede de telefonia

celular. Um *gateway* H.323 é um ponto final da rede que fornece comunicação em tempo real nas duas direções entre terminais H.323 em uma rede IP, terminais ITU em uma rede comutada ou para outro *gateway* H.323. Eles executam a função de translação entre diferentes formatos de dados. Os *gateways* são opcionais em uma LAN onde os terminais se comunicam diretamente. Mas quando os terminais precisam se comunicar com um ponto final em outra rede, a comunicação se faz via *gateway* através dos protocolos H.245 e Q.931.

Gatekeeper – É o componente mais importante de um sistema H.323 e executa a função de gerente. Atua como ponto central para todas as chamadas dentro de sua zona (é a agregação do *gatekeeper* e dos terminais registrados nela) e fornece serviços aos pontos finais registrados. Abaixo estão algumas funcionalidades dos *gatekeepers*:

a) Tradução de endereços: tradução de um endereço alias – que fornece um método alternativo de endereçamento de um ponto e pode ser um endereço de e-mail, um número telefônico ou algo similar – para um endereço de transporte. Isso é feito usando-se uma tabela de tradução que pode ser atualizada através de mensagens de registro;

b) Controle de admissão: o *gatekeeper* pode permitir ou negar acesso baseado em autorização de chamada, endereço de fonte e destino, etc;

c) Sinalização de chamada: o *gatekeeper* controla o processo de sinalização entre dois pontos finais que querem se conectar;

d) Autorização de chamada: o *gatekeeper* pode rejeitar chamadas de um terminal devido a falhas de autorização através do uso de sinalização H.225. As razões para rejeição podem ser acessos restritos durante alguns períodos de tempos ou acesso de certos terminais ou *gateways*;

e) Gerenciamento de largura de faixa: controle do número de terminais que podem acessar simultaneamente a rede. Através do uso da sinalização H.225, o *gatekeeper* pode rejeitar chamadas de um terminal devido à limitação de largura de faixa;

f) Gerenciamento da chamada: O *gatekeeper* pode manter uma lista de chamadas H.323 em andamento. Essa informação pode ser necessária para indicar que um

terminal chamado está ocupado e fornecer informações para a função de gerenciamento de largura de faixa.

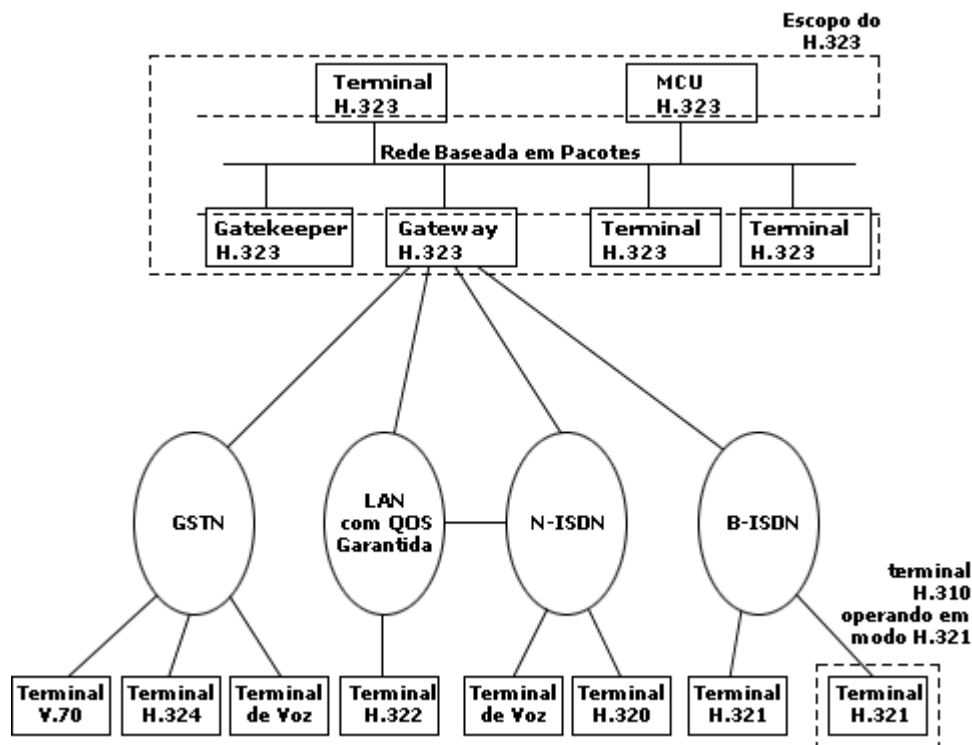


Figura 6 – Escopo do H.323

Para que seja estabelecida uma sessão, o padrão H.323 envia e recebe uma série de mensagens entre as estações envolvidas na comunicação. Uma sessão H.323 típica entre dois pontos tem o seguinte fluxo de mensagens:

Tabela 2 – H.323: Fluxo de mensagens

| Usuário A | Mensagem | Usuário B |
|---|---|-----------|
| Conecta-se em B na porta fixa 1720 - protocolo Q.931 | | |
| | Q.931 <i>Setup</i> → Endereço e Porta de A Endereço e Porta de B (1720) | |
| | ← Q.931 <i>Alerting</i> | |
| | ← Q.931 <i>Connect</i> Endereço e porta H.245 (de B) | |
| Conecta-se em B no endereço e porta recebidos - protocolo H.245 | | |

| | | |
|--|---|--|
| | \leftrightarrow Mensagens H.245 para negociação de capacidades e definição do ponto <i>Master/Slave</i> . | |
| | \leftarrow H.245 <i>OpenLogicalChannel</i> Endereço e porta RTCP (de B) | |
| | \rightarrow H.245 <i>OpenLogicalChannelAck</i> Endereço e porta RTP (de A) Endereço e porta RTCP (de A) | |
| | \rightarrow H.245 <i>OpenLogicalChannel</i> Endereço e porta RTCP (de A) | |
| | \leftarrow H.245 <i>OpenLogicalChannelAck</i> Endereço e porta RTP (de B) Endereço e porta RTCP (de B) | |
| | \leftrightarrow Fluxo RTP/ RTCP | |

3.4. RTP/RCTP – Real Time Protocol/Real Time Control Protocol

O RTP, segundo [Nóbrega 2001], é um protocolo padrão para transporte de dados com características de tempo real, como vídeo e áudio, que pode ser usado em diferentes serviços como mídia sob demanda e interativos. O protocolo é composto por uma parte de transmissão de dados e outra de controle, chamada RTCP (Real Time Control Protocol). A parte de dados consiste em um protocolo leve, que provê suporte para aplicações com características de tempo real, incluindo reconstrução temporal de mensagens, detecção de perdas, segurança, selo de tempo e identificação de conteúdo.

O protocolo de transporte acompanha o fluxo de bits gerados pelo codificador de mídia, normalmente telefonia IP, quebrando-os em pacotes, enviando-os pela rede e reproduzindo o fluxo de bits no receptor. O processo é complexo porque os pacotes podem ser perdidos, terem atrasos variados ou serem entregues fora de ordem. O protocolo de transporte deve permitir ao receptor detectar essas perdas. Ele deve também transportar informações de temporização para que o receptor possa fazer também a compensação para o atraso.

Seqüenciamento, sincronismo intramídia, identificação de conteúdo, identificação de quadro e identificação de origem são algumas funcionalidades do RTP.

Já o RTCP, que acompanha o RTP, provê informações adicionais sobre seus participantes, tais como retorno de informações de qualidade de serviço, sincronismo intermídia e identificação do usuário. De acordo com [Oliveira 2001], o RTCP necessita que todos os participantes enviem as informações periodicamente. O protocolo usa o mesmo endereço do RTP, porém em porta diferente. Nem todas as aplicações RTP utilizam o RTCP.

Aplicações em tempo real, tais como VoIP e fluxo de vídeo, têm um número de requisitos que as distinguem dos serviços de dados tradicionais da Internet:

a) Seqüência: caso cheguem fora de ordem, os pacotes devem ser reordenados em tempo real no receptor. Se perdido, o pacote deve ser detectado e compensado sem retransmissão;

b) Sincronização intra-mídia: o intervalo de tempo que existe entre pacotes sucessivos deve ser transmitida ao receptor como informação de controle. Nenhum dado, por exemplo, é geralmente enviado durante períodos de silêncio na fala. A duração desse silêncio deve ser reconstruída adequadamente. Se um número de diferentes mídias está sendo usado em uma única sessão, deve haver meios de sincronizá-lo. Com isso, é possível acertar o sinal de voz com o de vídeo. Isso também é conhecido como *lip-sync*;

c) Identificação do *payload*: na Internet, freqüentemente é preciso modificar a deflação de mídia dinamicamente para ajustá-la à disponibilidade de largura de faixa ou a novos usuários que se juntam ao grupo. Algum tipo de mecanismo é necessário para identificar a codificação utilizada em cada pacote;

d) Identificação de frame: vídeo e voz são enviados em unidades lógicas chamadas *frames*. É indispensável indicar para o receptor onde é o início ou fim do frame, de forma a auxiliar no sincronismo da entrega dos dados.

Os serviços descritos acima são providos por um protocolo de transporte. Usado para isso na Internet, o RTP tem dois componentes: o próprio RTP e o RTCP.

4 NAT (Network Address Translation)

O problema de falta de endereços IPv4 na Internet foi resolvido pelo NAT. Cada computador que acessa a Internet deve ter o protocolo TCP/IP configurado. Para isso, cada computador da rede interna precisa de um endereço IP válido na Internet. E não há endereços IPs suficientes.

A solução para essa questão veio com a criação do NAT. Com ele, os computadores da rede interna, utilizam os chamados endereços privados, que não são válidos na Internet. Pacotes que tenham como origem ou como destino um endereço na faixa dos endereços privados não serão encaminhados e sim descartados pelos roteadores, configurados para isso. As faixas de endereços privados são definidas na [RFC 1597] da seguinte maneira:

Tabela 3 – Faixas de endereços privados

| Limite inferior | Limite superior |
|------------------------|------------------------|
| 10.0.0.0 | 10.255.255.255 |
| 172.16.0.0 | 172.31.255.255 |
| 192.168.0.0 | 192.168.255.255 |

A vantagem que as empresas têm com os endereços privados é de utilizar a mesma faixa como forma de endereçamento de sua rede interna já que esses endereços privados não podem ser utilizados diretamente na Internet. Qualquer empresa pode ter endereços nas faixas definidas.

Com o uso do NAT, a empresa fornece acesso à Internet para um grande número de computadores da rede interna com um número bem menor de endereços IP, válidos na Internet. Com um servidor NAT, uma rede de, por exemplo, 50 computadores com endereçamento privado podem ter acesso à Internet usando um único endereço IP válido: o endereço IP da interface externa do NAT. Com isso, há uma grande economia de endereços IP: tem-se 50 computadores acessando a Internet (configurados com

endereços IP privados) e utilizando um único endereço IP válido, que é o da interface externa do servidor configurado como NAT.

Traduzir os endereços privados, que não são válidos na Internet, para o endereço válido da interface pública do servidor é a principal função do NAT. Quando um cliente acessa a Internet, no pacote de informação enviado por este cliente está registrado o endereço IP da rede interna, por exemplo: 10.10.0.5. Porém, este pacote não pode ser enviado para a Internet com este endereço IP como origem, senão no primeiro roteador este pacote será descartado, já que o endereço 10.10.0.5 não é um IP válido na Internet.

O NAT substitui o endereço IP de origem por um dos endereços IP da interface externa do NAT (endereço fornecido pelo provedor de Internet e, portanto, válido na Internet) para que o pacote possa ser enviado para a Internet. Esse processo é chamado de tradução de endereços.

No caso de resposta retornar, o NAT a repassa para o cliente que originou o pedido. Para saber a qual cliente destina-se uma resposta, o NAT, ao executar a função de tradução de endereços, associa um número de porta, que é único, com cada um dos computadores da rede interna.

Se um cliente interno tenta se comunicar com a Internet, o NAT substitui o endereço do cliente, como endereço de origem, por um endereço válido na Internet. Mas além do endereço, também é associada uma porta de comunicação. O NAT mantém uma tabela interna onde ficam registradas as associações IP/Porta privados e IP/Porta públicos.

Quando a resposta retorna, o NAT consulta uma tabela interna e, pela identificação da porta, sabe para qual computador da rede interna deve ser enviada a referida resposta, uma vez que a porta de identificação está associada a um endereço IP da rede interna.

Assim, ao mesmo tempo, diversos computadores da rede interna podem acessar a Internet, com um único endereço IP ou com uma quantidade de endereços IP bem menor do que o número de computadores da rede interna. A diferenciação é feita por meio de uma atribuição de porta de comunicação diferente, associada com cada IP da rede interna.

Tabela 4 – Tabela de mapeamento dos dispositivos NAT

| Origem | Interface externa NAT | IP Remoto |
|------------------|------------------------------|---------------------|
| 192.168.1.1:1029 | 140.116.72.219:1029 | 140.116.72.72:23 |
| 192.168.1.1:1030 | 140.116.72.219:1030 | 150.162.160.63:8080 |
| 192.168.1.2:1029 | 140.116.72.219:30029 | 140.116.72.72:21 |
| 192.168.1.3:1030 | 140.116.72.219:30030 | 140.116.72.72:21 |

4.1. Tipos de NAT

4.1.1. Cone NAT

Esse NAT utilizará a mesma porta externa para subseqüentes requisições da estação com IP privado para a dada porta interna, após ter estabelecido uma associação IP/Porta privados para IP/Porta públicos. Enquanto a porta interna da estação continuar a mesma, o NAT manterá a porta externa da primeira requisição mesmo que o IP de destino seja alterado.

Cone NAT

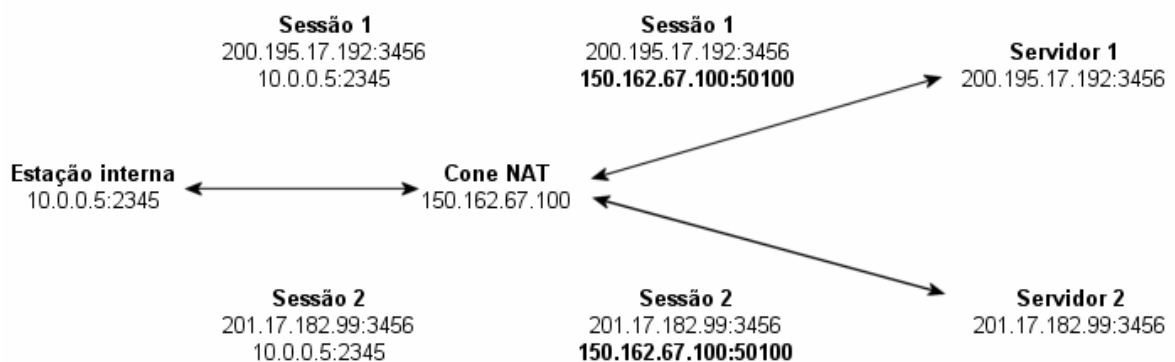


Figura 7 – Cone NAT

4.1.2. Symmetric NAT

Ao contrário do Cone NAT, o Symmetric NAT não mantém a associação entre IP/Porta privados e IP/Porta públicos. Se a porta interna não for alterada, mas o IP de destino sim, o NAT estabelecerá uma nova porta externa.

Symmetric NAT

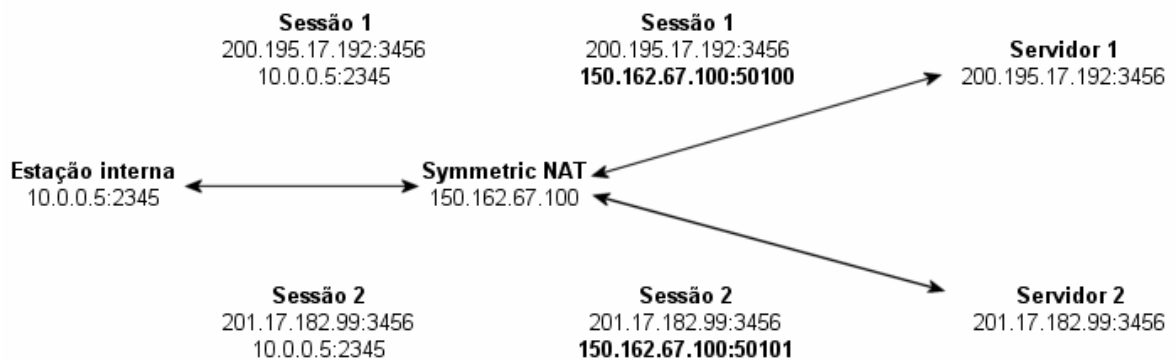


Figura 8 – Symmetric NAT

4.1.3. NAT Cone Full

O NAT Cone Full também é conhecido como NAT promiscuo. Após o estabelecimento da associação (IP/Porta privados – IP/Porta públicos), qualquer requisição externa dirigida ao par IP/Porta públicos, será enviada à estação interna que iniciou a requisição.

4.1.4. Restricted Cone NAT

Nesse tipo de NAT, refina o princípio do firewall que rejeita tráfego entrante não solicitado, as requisições entrantes somente serão enviadas à estação interna que previamente enviou mensagens à estação externa que enviou a requisição.

4.1.5. Port-Restricted Cone NAT

Esse tipo de NAT somente envia as requisições externas à estação interna se o par IP/Porta destino da requisição previamente enviada combinar com as informações da requisição entrante.

4.2. Técnicas para comunicação ponto-a-ponto através do NAT

Três técnicas são usadas para implementar a comunicação ponto-a-ponto através de dispositivos NAT. As duas primeiras, apesar de resolverem os problemas da comunicação ponto-a-ponto, possuem desvantagens que as tornam inviáveis para o uso na Internet. A terceira, chamada de *UDP Hole Punching*, é eficiente e possui vantagens sobre as demais técnicas.

4.2.1. Relaying

O *Relay* é a técnica mais garantida e ao mesmo tempo menos eficiente de comunicação entre pontos de rede separados por NAT. Baseia-se na adaptação da comunicação ponto-a-ponto para o paradigma cliente/servidor.

Esse método utiliza um terceiro ponto de rede, acessível (permite conexões de entrada) por ambos os pontos que desejam se comunicar e que irá atuar como servidor. Dessa forma, os pontos conectam-se no servidor e trocam mensagens através dele: o servidor repassa as mensagens enviadas pelos clientes.

A vantagem dessa técnica é que a conexão sempre funcionará enquanto os pontos conseguirem acessar o servidor. A desvantagem é a latência gerada na comunicação, bem como o desperdício de banda e o processamento do servidor.

Relaying

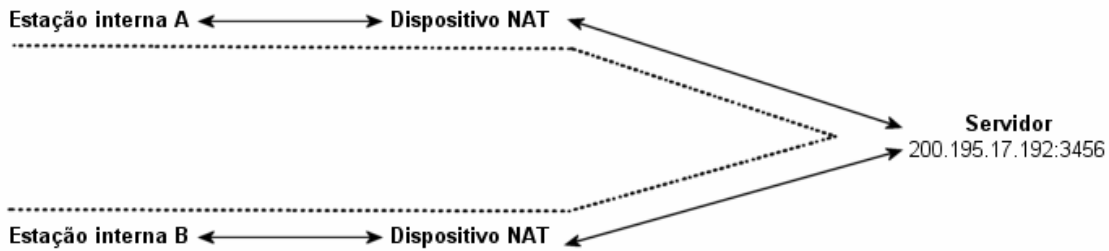


Figura 9 - Relaying

4.2.2. Conexão reversa

Esse procedimento é útil apenas quando um dos pontos que desejam se comunicar está atrás do NAT. Ele também considera a existência de um terceiro ponto – um servidor - acessível a ambos os clientes.

O ponto que está atrás do NAT deve manter uma conexão aberta com o servidor, para que possa ser avisado de que outro usuário deseja iniciar uma conexão. Quando o cliente que não está atrás do NAT deseja se conectar, ele primeiro estabelece uma conexão com o servidor, solicitando que o usuário atrás do NAT seja avisado. Dessa forma, NAT permite a conexão, porque ela é originada de dentro da rede.

A desvantagem é que essa técnica só é útil quando um dos lados estiver atrás de NAT e o outro não, o que nem sempre verdade.

Conexão reversa

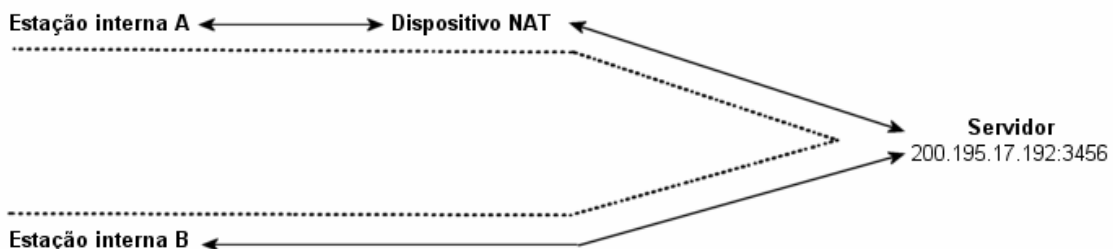


Figura 10 – Conexão reversa

4.2.3. UDP Hole Punching

Essa é considerada a melhor técnica para implementar comunicação ponto-a-ponto entre estações que estão em uma rede controlada por NATs. O *UDP Hole Punching* aproveita as propriedades comuns de *firewalls* e NATs e promove a estabilização da conexão direta entre as estações que estão sob o controle desses dispositivos. Essa conexão ponto-a-ponto é possível até mesmo se os dispositivos de controle esconderem informações da rede das estações envolvidas.

Essa técnica foi brevemente mencionada na seção 5.1 da RFC3027 [NAT-PROT] e como o nome sugere pode ser utilizada apenas sobre os pacotes UDP.

4.2.3.1. Cenários

4.2.3.1.1. Estações atrás de diferentes NATs

Suponha que os clientes A e B possuem endereços IP privados e estão sob o controle de diferentes NATs e um servidor S que usa a porta UDP 2345. A e B possuem uma conexão UDP com o servidor S. Isso faz com que o NAT A associe a porta 50100 à sessão do servidor e o NAT B associe a porta 62100 à sessão do servidor respectivamente.

UDP Hole Punching: Diferentes NATs

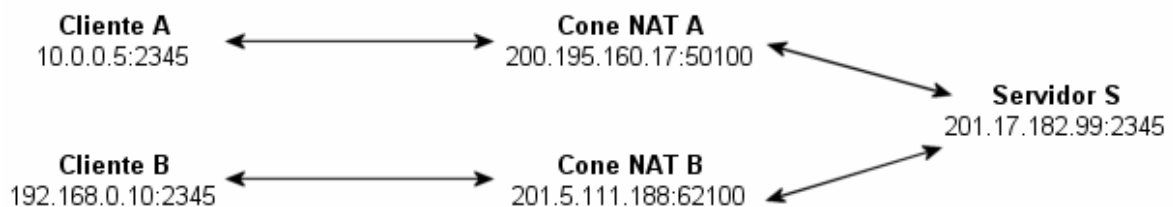


Figura 11 – UDP Hole Punching em diferentes NATs

Agora o cliente A quer estabelecer uma conexão com o cliente B. Para isso, ele envia um pacote para 201.5.111.188, porta 62100, informado pelo servidor S. Esse pacote será descartado pelo NAT (caso não seja um *Full Cone NAT*) porque a origem não combina com o endereço do servidor S que foi o endereço de destino no pacote do cliente B para estabelecer a conexão com S. O mesmo acontecerá se o cliente B tentar conexão com o A.

Imagine agora que o cliente A queira enviar pacotes UDP para o endereço público de B. Ele pede ao servidor que solicite que B faça o mesmo em relação ao endereço público de A. Os pacotes de saída de A fazem com que o NAT A crie uma sessão entre o IP privado de A e o IP público de B. Ao mesmo tempo, os pacotes de saída de B fazem o NAT B criar uma sessão entre o IP privado de B e o IP público de A. Como foram criadas sessões UDP em cada direção, os clientes A e B podem se comunicar diretamente sem a interferência do servidor S.

4.2.3.1.2. Estações atrás do mesmo NAT

Nesse cenário dois clientes estão atrás de um mesmo NAT e dentro da mesma faixa de endereços privados. O cliente A tem uma conexão com S associada à porta 62000 e o cliente B possui uma conexão com S que o NAT associou à porta 62001.

UDP Hole Punching: Mesmo NAT

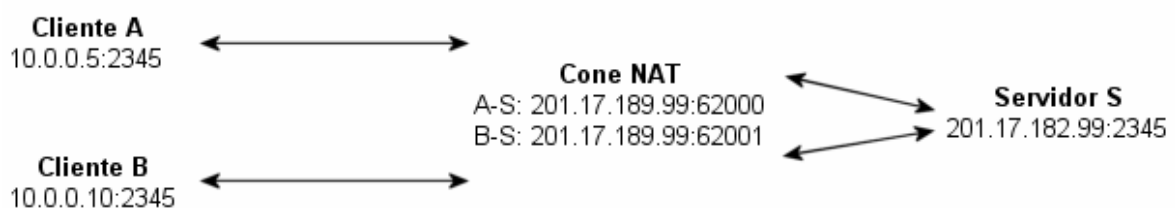


Figura 12 – *UDP Hole Punching* em um mesmo NAT

Suponha que os clientes utilizem a técnica de *UDP Hole Punching* para estabelecer uma conexão ponto-a-ponto com ajuda do servidor S. Os clientes receberão o IP e a porta pública do cliente com o qual deseja estabelecer a comunicação. Então cada um deles enviará pacotes UDP para seu próprio endereço público.

Deste modo, os dois clientes poderão estabelecer a conexão, pois o NAT permite que sejam criadas sessões com estações internas e não apenas com estações externas. Isso é chamado de “*loopback translation*”, porque pacotes desse tipo que chegam ao NAT são traduzidos e então retornam à rede interna ao invés de serem enviados para a rede pública.

Por exemplo, quando o cliente A envia um pacote UDP para o IP público do cliente B, o pacote inicialmente tem IP e porta de origem 10.0.0.5:2345 e IP e porta de destino 201.17.189.99:62001. O NAT recebe o pacote e altera a origem para 201.17.189.99:62000, que é o endereço público de A, e o destino para 10.0.0.10:2345. Então, repassa o pacote para B.

Mesmo com o processo de “*loopback translation*” disponível no NAT, não é interessante utilizá-lo, porque a latência aumenta entre os clientes envolvidos, já que a cada pacote enviado, a tradução é executada.

A solução para evitar o aumento do atraso nos pacotes é enviar, na conexão com o servidor S, o IP e a porta verificada pela própria estação. Então, os clientes enviarão os pacotes primeiramente para o endereço alternativo e, se estiverem atrás do mesmo NAT, a conexão será estabelecida sem a interferência do dispositivo NAT. Caso os clientes não estejam atrás do mesmo NAT, essa primeira tentativa não funcionará e então o processo de *UDP Hole Punching* se repetirá com o IP público de cada um dos clientes.

4.2.3.1.3. Estações separadas por múltiplos NATs

Nesse cenário de múltiplos NATs, não é possível dois cliente estabilizarem uma conexão otimizada ponto-a-ponto sem conhecer a topologia da rede que os separam.

UDP Hole Punching: Múltiplos NATs

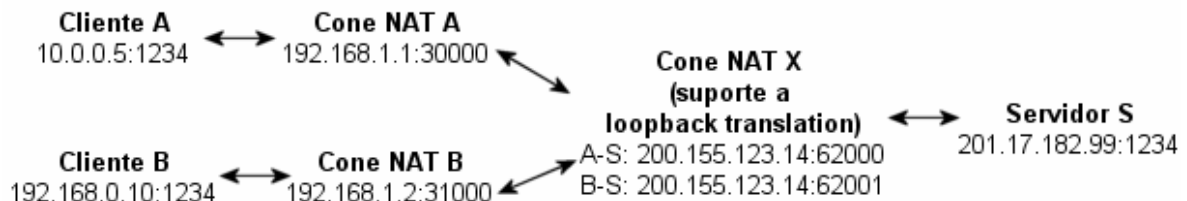


Figura 13 – UDP Hole Punching em múltiplos NATs

Imagine que X é um grande NAT utilizado por um provedor de serviços de Internet (ISP) para multiplexar seus usuários em poucos IPs públicos. A e B são pequenos NATs desenvolvidos independentemente por dois usuários dos serviços ISP para gerenciar suas redes internas. Somente o servidor S e o NAT X possuem endereços IPs públicos. Os IPs dos NATs A e B são IP privados providos pelo NAT X e os IPs dos clientes A e B também são privados, porém providos pelos NATs A e B.

Os clientes A e B, como anteriormente feito, conectam-se previamente ao servidor S. Essa conexão faz com que os NATs A e B criem sessões para seus respectivos clientes. Por sua vez o NAT X cria sessões para as requisições dos NATs A e B.

Agora suponha que os clientes A e B tentem estabelecer uma conexão ponto-a-ponto. A melhor opção seria o cliente A enviar pacotes ao IP público de B (192.168.1.2:31000) e o cliente B para o IP público de A (192.168.1.1:30000). Mas os clientes não podem ter conhecimento desses IPs, porque o servidor S somente recebe como informação o IP público do NAT X, 200.155.123.14:62000 e 200.155.123.14:62001, para os clientes A e B respectivamente.

Mesmo que os clientes A e B tenham, por algum meio, conhecimento desses IPs públicos, não seria garantido a utilização deles já que os IPs privados do NAT X poderiam entrar em conflito com os IPs privados dos NATs A e B.

Desta forma os clientes A e B não possuem outra alternativa senão usar o IP público do NAT X para executar a técnica de *UDP Hole Punching* e confiar que o NAT X possua “*loopback translation*”.

4.2.3.2. Consistência das reservas de portas

A técnica de *UDP Hole Punching* somente funciona se ambos os NATs são Cone NATs, nos quais se mantém a porta reservada entre o par IP/UDP privado e o IP/UDP público, até que a conexão UDP seja encerrada. Associar uma nova porta para cada sessão, como o *Symmetric NAT*, torna impossível que a aplicação UDP reuses uma associação já estabelecida para se comunicar com diferentes destinos.

Como os *Cone NATs* são maioria, a técnica é bastante utilizada, mas falha quando aplicada aos *Symmetric NATs* que, mesmo em minoria, são utilizados por alguns usuários.

4.2.3.3. Predição da porta UDP

Existe uma variante da técnica de *UDP Hole Punching* que permite a conexão ponto-a-ponto na existência de *Symmetric NATs*. Esse método é chamado de técnica “N+1” [BIDIR] e é explorada em detalhes por Takeda [SYM-STUN].

O método trabalha com a análise do comportamento do NAT e tenta prever qual será a porta associada para futuras sessões. Considere a situação em que os clientes A e B estão em NATs diferentes e ambos estabelecem previamente uma conexão com o servidor S.

UDP Hole Punching: Predição porta UDP

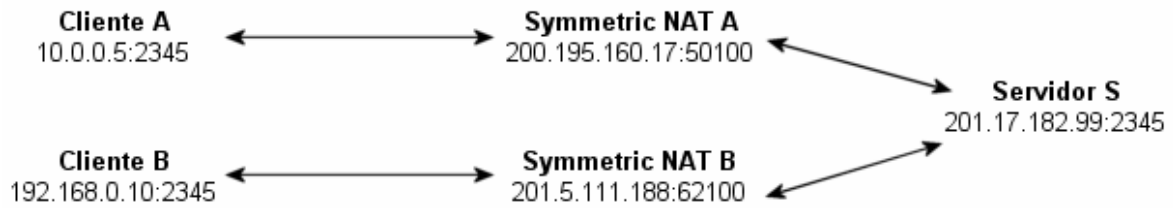


Figura 14 – UDP Hole Punching: Predição de porta UDP

O NAT A associou sua porta UDP 50100 à sessão entre o cliente A e S. Por sua vez a porta UDP 62100 foi associada pelo NAT B à sessão entre o cliente A e o servidor S. Comunicando-se com o servidor S, os clientes A e B conhecem o IP público e a porta um do outro.

Agora o cliente A envia pacotes UDP para o IP 201.5.111.188:62101 (perceba o incremento da porta) e o cliente B manda pacotes para a porta 50101 e o IP 200.195.160.17. Se os NATs A e B associarem novas portas seqüenciais para as atuais sessões e não houver muito tempo da sessão com o servidor S, então, a conexão ponto-a-ponto entre A e B será estabilizada com sucesso. Isso acontece porque os pacotes enviados de A para B fazem com que o NAT A associe uma nova porta à sessão que, observando a seqüência, será a porta 50101. O mesmo acontece com o NAT B que associa a próxima porta (62101) para os pacotes enviados de B para A.

UDP Hole Punching: Técnica "N+1"

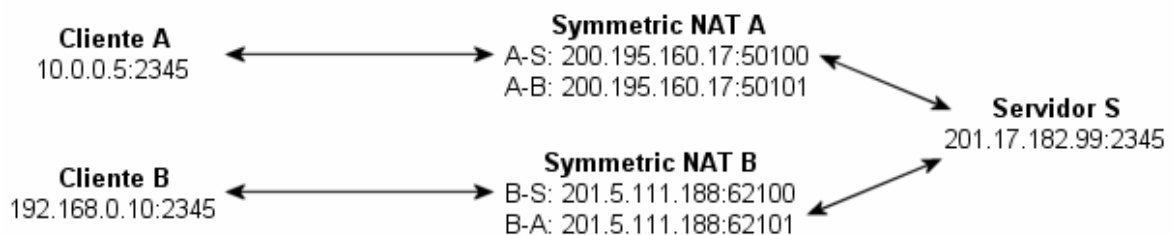


Figura 15 – UDP Hole Punching: Técnica "N+1"

Fica claro que vários fatores podem atrapalhar esse método:

- a) A porta a qual foi feita a predição pode estar sendo utilizada por uma sessão anterior a todo o processo;
- b) Uma conexão foi estabelecida no intervalo entre a sessão com o servidor S e a tentativa de conexão entre os clientes A e B;
- c) O NAT pode associar portas aleatórias para as novas sessões e não seqüencialmente como previsto;

Caso as aplicações ponto-a-ponto implementem essa técnica, elas precisam saber se os NATs envolvidos são *Cone NATs* ou se um deles é *Cone NAT* e o outro é *Symmetric NAT*. Essa identificação pode ser feita por STUN (*Simple Traversal of UDP Through NAT*), que é uma técnica utilizada para determinar o tipo do NAT. A aplicação precisa alterar seu comportamento de acordo com o resultado da detecção.

Finalmente, a técnica de predição de portas UDP não funciona quando os clientes estão separados por dois ou mais níveis de NATs e o NAT mais próximo aos cliente é do tipo *Symmetric*.

5 Incompatibilidade do protocolo H.323 com os servidores NAT

O H.323 é um protocolo complexo e possui uma série de incompatibilidades com os tradutores de endereços IP. Neste capítulo serão apresentados os dois principais problemas entre o padrão H.323 e os servidores NAT.

Esses problemas inviabilizam a criação de uma sessão H.323 entre usuários que estejam em ambientes que possuam servidores NAT.

O problema das portas dinâmicas está relacionado com a solução de mapeamento de portas no NAT para o estabelecimento de sessões H.323. Já o problema da área de dados está diretamente ligado à solução proposta neste trabalho.

5.1. Portas dinâmicas

A primeira dificuldade encontrada ao se fazer uso de aplicações H.323 em redes separadas por NAT é que o H.323 define de forma dinâmica quais portas de entrada ele irá utilizar para o estabelecimento da sessão. Para a realização de uma única chamada o H.323 utiliza uma porta de entrada para cada protocolo: H.245, H.225, RTP, RTCP.

Isso significa que, exceto pela porta de entrada 1720, os números das demais portas abertas não são previsíveis, o que impossibilita soluções do tipo mapeamento de portas no dispositivo NAT.

O mapeamento de portas consiste em definir no dispositivo NAT uma tabela de quais portas ele deve aceitar conexões de entrada e para qual ponto interno da rede os pacotes de entrada direcionados a essas portas devem ser encaminhados.

No caso do H.323, para se permitir conexões externas a um único ponto atrás do NAT, seria necessário configurar no dispositivo um mapeamento de todas as portas de entrada em todas as portas acima de 1024. Esse mapeamento acaba gerando um grave problema de segurança para o ponto de rede em questão, o que acaba inviabilizando essa solução.

Tabela 5 - Portas de entrada utilizadas pelo protocolo H.323

| Protocolo | Porta |
|---------------|--|
| Q.931/H.225.0 | Fixa 1720 (TCP) |
| H.245 | Dinâmica 1024-65535 (TCP) |
| RTP | Dinâmica 1024-65535 (UDP) |
| RTCP | Dinâmica 1024-65535 (UDP), igual a porta RTP da conexão mais 1 |

5.2. Área de dados

A outra incompatibilidade do protocolo H.323 está relacionada à solução proposta neste trabalho e diz respeito ao envio de endereços IP dentro da área de dados dos pacotes. Protocolos que utilizam esse tipo de procedimento acabam impossibilitando a negociação de conexões através de dispositivos NAT/PAT [RFC2663].

Como descrito anteriormente, esses dispositivos alteram o endereço IP nos cabeçalhos dos pacotes que trafegam por meio deles. O problema nesse caso é que o H.323 também manda o endereço IP das entidades envolvidas na parte de dados do pacote. Dessa forma, o destinatário da chamada irá usar o endereço IP recebido nos dados do pacote para criar a conexão que dará continuidade ao protocolo. Tratando-se de redes separadas por NAT, o IP recebido será um endereço privado da outra rede e não terá validade.

Geralmente os dispositivos NAT/PAT não processam a área de dados dos pacotes IP, a menos que implementem alguma função específica para tal, chamada de *Application Level Gateway* – ALG. Esse tipo de função permite ao dispositivo determinar que tipo de pacote está sendo examinado e se o mesmo precisa de ajustes. Dessa forma, existem NATs que executam a função ALG que trata os pacotes H.323, corrigindo os endereços enviados nos pacotes - do endereço privado do ponto pelo endereço público da rede.

Entretanto esse tipo de funcionalidade não é comum na maioria dos NATs por reduzirem a performance da rede e, especificamente no caso do H.323, pelo motivo das

mensagens trocadas nos protocolos H.225 e H.245 serem codificadas em *Abstract Syntax Notation* (ASN) e não em formato binário ou texto, o que dificulta ainda mais o processamento dos pacotes, como descrito em [NAT-PROT].

Dessa maneira, para que a sessão entre os pontos H.323 separados por NATs possa ser realizada, é necessário traduzir os endereços enviados dentro das mensagens H.225 e H.245 para os endereços públicos das redes.

6 Funcionamento da ferramenta

Este capítulo tem como objetivo mostrar o funcionamento da ferramenta que foi desenvolvida e como a mesma permite solucionar o problema de estabelecimento de uma sessão H.323 entre dois pontos de rede separados por servidores NAT.

A ferramenta implementada baseia-se na técnica *UDP Hole Punching* para criar o canal de comunicação que será usado para estabelecer a sessão H.323. Como descrito anteriormente, é necessário que haja um servidor acessível por ambos os pontos de rede para que os mesmos possam descobrir qual porta UDP deverá ser utilizada em cada ponto para a transmissão dos dados entre eles.

Este canal de transmissão UDP que foi criado será o único meio de comunicação entre os dois pontos, e é através dele que a ferramenta desenvolvida irá transmitir todos os pacotes de rede gerados pelo aplicativo H.323 na forma de um túnel de pacotes.

6.1. Tunelamento de pacotes

O tunelamento de pacotes consiste em transmitir os pacotes de rede gerados por uma determinada aplicação através de um outro canal, podendo ser esse uma outra porta ou protocolo na mesma rede, ou até mesmo uma outra rede.

A idéia das redes VPN (*Virtual Private Network*), exemplos típicos de tunelamento, é criar um túnel seguro entre dois pontos de rede. Para tal, estabelece-se uma conexão TCP/IP criptografada entre dois pontos de rede a partir da qual são transmitidos os dados gerados por outras aplicações.

A técnica de tunelamento implica em encapsular os pacotes que transitam entre os dois pontos de rede na área de dados dos pacotes de uma outra conexão. A mesma pode ser implementada entre entidades intermediárias da rede (como o próprio NAT) ou entre os próprios pontos de rede.

Existem inúmeras aplicações de tunelamento, que permitem desde a implementação de segurança em redes até a transmissão de dados de uma determinada rede sobre outra, como por exemplo, o envio de dados IPX em pacotes IP.

O tunelamento de pacotes realizado pela ferramenta desenvolvida nada mais é do que o envio de todos os pacotes IP que transitarão normalmente em uma sessão H.323 pelo canal de comunicação UDP que foi aberto entre os dois pontos de rede.

Cada pacote de rede gerado pela aplicação H.323 é inserido na área de dados de um novo pacote UDP, o qual é transmitido pelo canal UDP que foi criado. No lado de destino do pacote, o mesmo é retirado da área de dados do pacote UDP que foi utilizado e inserido na camada de rede do sistema operacional. O aplicativo H.323 irá recebê-lo de forma transparente, como se o mesmo tivesse sido transmitido normalmente.

6.2. Detalhes do funcionamento

A ferramenta proposta nesse trabalho utiliza as técnicas de *UDP Hole Punching*, tunelamento de pacotes sobre UDP (*UDP Tunneling*) e de filtragem de pacotes (*Application Level Gateway*) para estabelecer uma conexão H.323 entre dois pontos de rede separados por tradutores de endereço.

A técnica *UDP Hole Punching* é vital para a criação do canal UDP por onde os pacotes H.323 serão transmitidos. Como descrito anteriormente, não é possível criar conexões TCP entre pontos separados por NAT, mas o uso do tunelamento de pacotes permite que as conexões TCP (H.225, H.245), necessárias para a sinalização da chamada, sejam originadas de forma transparente para o aplicativo H.323, bem como o envio dos pacotes UDP de áudio (RTP).

A ferramenta desenvolvida deve estar em funcionamento em ambos os pontos da rede, juntamente com o aplicativo H.323 que será utilizado. A mesma executa duas atividades principais para permitir o estabelecimento da sessão H.323:

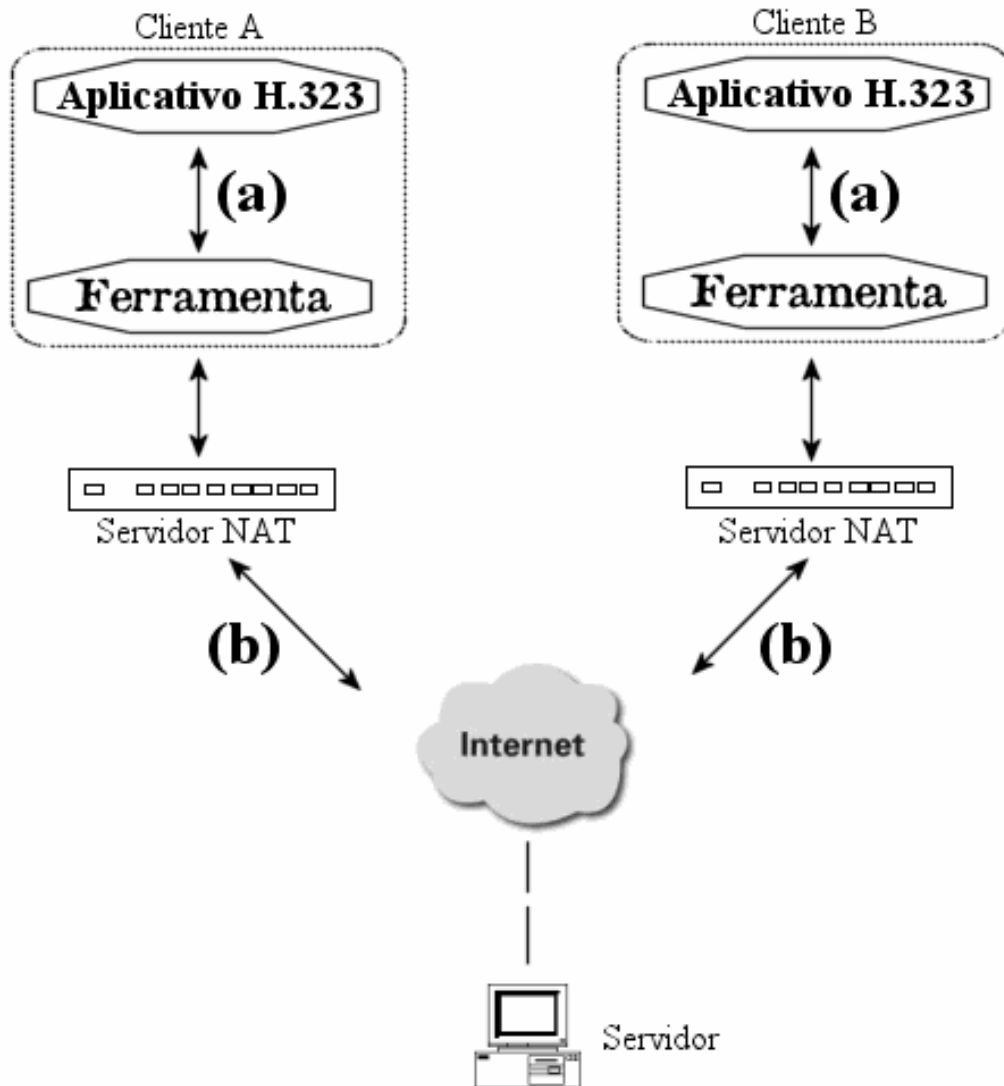


Figura 16 – Funcionamento da ferramenta

(a) Monitoramento, filtragem e injeção dos pacotes H.323 que são recebidos e enviados pelo ponto de rede.

(b) Estabelecimento do canal UDP com o destino da conexão (com auxílio do servidor) por meio da técnica de *UDP Hole Punching*.

O procedimento inicia-se quando o usuário através do aplicativo H.323 solicita uma chamada de voz com o IP público do ponto de rede com o qual ele deseja se comunicar. A partir daí os seguintes passos são executados:

6.2.1. Estabelecimento do canal UDP

O aplicativo H.323 gera o primeiro pacote de rede com o intuito de abrir a conexão TCP que inicia o protocolo - conforme descrito no capítulo 3. Esse pacote é interceptado e bloqueado localmente pela ferramenta antes de chegar ao dispositivo de rede. Do cabeçalho IP do pacote interceptado, a ferramenta retira o endereço IP do ponto de destino com o qual o usuário deseja se comunicar, o que é utilizado para a criação do túnel UDP.

Em seguida é estabelecido o canal UDP com este ponto de destino fazendo uso da técnica de *UDP Hole Punching*. Para que essa etapa possa ser concluída deverá estar previamente configurado na ferramenta qual o endereço IP do servidor a ser usado para o auxílio no estabelecimento do canal. É por meio dele que será enviado o pacote interceptado no passo acima e os subsequentes.

6.2.2. Descoberta dos endereços IP públicos

Deste processo de estabelecimento do canal UDP também são descobertos os endereços públicos de ambas as redes, os quais serão utilizados nos passos seguintes. Os endereços são visíveis ao servidor que auxilia na criação do canal UDP, e esta informação é transmitida para a ferramenta em ambos os pontos.

6.2.3. Alteração do endereço IP de origem no cabeçalho dos pacotes

Tendo o canal UDP criado, a ferramenta poderá transmitir por meio dele os pacotes de rede que serão gerados pelo aplicativo H.323 para o outro ponto da rede. Para que os pacotes possam ser respondidos corretamente pelo aplicativo H.323 do outro ponto, o cabeçalho IP do pacote deve ter o campo de endereço IP origem (*source*) corrigido. Nesse campo se encontra o IP interno desse ponto de rede, o qual é substituído pelo endereço público da rede. Após essa correção o campo de verificação de erros (*checksum*) do cabeçalho IP e do cabeçalho TCP são recalculados e substituídos pela ferramenta, caso contrário o mesmo seria rejeitado no destino.

6.2.4. Tunelamento dos pacotes

O pacote interceptado é então inserido em um novo pacote UDP o qual é transmitido pelo túnel. No ponto destino, o pacote gerado pelo aplicativo H.323 é retirado da área de dados do pacote UDP recebido.

6.2.5. Alteração do endereço IP de destino no cabeçalho dos pacotes

Antes de ser passado para a camada de rede do sistema operacional no ponto destino, o pacote precisa ter o campo de endereço destino (*destination*) do cabeçalho IP corrigido.

O cabeçalho do pacote contém nesse campo o IP público da rede deste ponto, o qual é trocado para o endereço IP privado. Novamente o campo de verificação de erros (*checksum*) do cabeçalho IP e do cabeçalho TCP é corrigido para que o pacote seja aceito.

6.2.6. Injeção do pacote recebido

Nesse passo o pacote é enfim inserido na camada de rede do sistema operacional. O aplicativo H.323 destino irá recebê-lo de forma transparente, como se tivesse sido enviado normalmente por um ponto dentro da mesma rede.

Este procedimento é realizado de forma idêntica para todos os pacotes de sinalização e de resposta enviados por ambas as partes, exceto para os pacotes H.323 que contém endereços IP de origem e destino dentro da área de dados do pacote - descritos no capítulo 5 – que requerem um passo adicional:

6.2.7. Alteração dos endereços IP dentro dos dados do pacote

Conforme descrito anteriormente, certos pacotes do protocolo H.323 contém os endereços IP de origem e destino da sessão também dentro da sua área de dados. O aplicativo H.323 ao qual o pacote se destina utiliza o endereço recebido dentro do pacote para criar as conexões subseqüentes.

Dessa forma, o endereço IP contido dentro do pacote H.323 precisa ser alterado da mesma forma que o endereço IP do cabeçalho. Para isso, a ferramenta procura pelo IP privado do ponto de rede dentro da área de dados do pacote e o substitui pelo IP público da rede. Já no ponto de destino a ferramenta substitui o endereço IP público da rede pelo IP privado do ponto.

Nesse ponto a sessão H.323 foi estabelecida, e o áudio pode começar a ser transmitido. O envio dos pacotes de áudio também segue o mesmo procedimento, pois mesmo os pacotes de áudio sendo UDP, esses também precisam ser enviados pelo túnel, visto que o *UDP Hole Punching* não permite a transmissão de pacotes UDP por uma porta específica – no caso a porta que foi definida pela conexão H.323 – apenas pela porta estabelecida pelo NAT.

6.3. Implementação da ferramenta

A ferramenta desenvolvida para monitorar os tráfego da rede utiliza o *WinpkFilter*, um *framework* de alta eficácia na filtragem de pacotes no sistema operacional Windows (9x/ME/NT/2000/XP/2003), que permite ver, modificar e descartar os pacotes de rede que são enviados e recebidos pela máquina. O *WinpkFilter* instala-se entre o adaptador de rede e a camada TCP/IP, permitindo assim, o desenvolvimento de aplicações como *Firewalls*, NAT's, VPN's e analisadores de pacotes, entre outros.

É através da funcionalidade de interceptação de pacotes do *WinpkFilter* – chamada *ReadPacket* - que a ferramenta têm acesso ao tráfego da rede. A mesma processa apenas os pacotes de rede do protocolo H.323, sendo que os demais são repassados automaticamente para a camada de rede seguinte.

Os pacotes H.323 interceptados não são repassados para a camada de rede seguinte, e sim transmitidos pelo canal UDP para o ponto destino. No recebimento dos mesmos, o seu conteúdo (o pacote embutido) é retirado e passado para a camada de rede através da chamada *SendPacketToMstcp*.

A ferramenta foi implementada no ambiente de desenvolvimento Borland Delphi 5 e consiste basicamente no uso dos métodos do *framework WinpkFilter* e na transmissão de pacotes UDP para a criação do túnel e envio de pacotes.

O aplicativo H.323 utilizado para testes da ferramenta foi o Openphone versão 1.8.1 que é baseado no OpenH323 (www.openh323.org) – implementação *open source* do protocolo H.323.

6.4. Testes realizados

Os testes foram realizados em um ambiente com as seguintes características:

Tabela 6 – Ambiente de testes

| Ponto | NAT | IP Privado | IP Público | Softwares |
|---------------------------------|----------|--------------|----------------|--|
| H.323 A | Cone NAT | 192.168.0.23 | 201.10.2.14 | Ferramenta desenvolvida Openphone v.1.8.1 |
| H.323 B | Cone NAT | 10.1.1.16 | 200.100.5.136 | Ferramenta desenvolvida Openphone v.1.8.1 |
| Servidor <i>HolePunching</i> | Não | | 150.162.160.30 | Ferramenta para auxílio <i>HolePunching</i> |

Foram realizadas com sucesso chamadas entre os pontos A e B em ambas as direções. O procedimento de estabelecimento da sessão levou em torno de 2 segundos a mais do que uma conexão H.323 normal, tempo esse gasto na negociação do túnel UDP.

A qualidade das chamadas foi considerada aceitável e semelhante a uma conexão H.323 normal, considerando que há um atraso adicional gerado pela ferramenta no processamento dos pacotes.

7 Conclusão

O principal objetivo deste trabalho foi analisar o estado da arte da comunicação de voz sobre IP na Internet atualmente e apresentar uma solução para um problema específico gerado pelas topologias de rede que vem se tornando comuns no mercado: a conexão ponto a ponto entre clientes separados por dispositivos de tradução de endereço.

A solução proposta para esta dificuldade foi obtida através do estudo dos protocolos que compõe a arquitetura VOIP típica desses casos – especialmente o NAT e o H.323 – e da busca por técnicas que viabilizassem contornar as incompatibilidades dos mesmos e oferecessem a possibilidade de comunicação entre usuários que se encontram nesse tipo de ambiente.

A implementação das técnicas estudadas e a integração das mesmas nos forneceram uma ferramenta que permite auxiliar a conexão entre aplicativos de voz sobre IP localizados em redes com essas características, alcançando assim o objetivo proposto.

Pode-se constatar também que a ferramenta gerada neste trabalho é passível de ser utilizada em conjunto com outros aplicativos além da comunicação voz sobre IP e de multimídia, ou seja, exceto pela camada que trata especificamente do protocolo H.323 trata-se uma ferramenta genérica para conexão ponto a ponto entre estações de rede.

Alguns trabalhos futuros propostos incluem o projeto e a proposta de inclusão das técnicas aqui analisadas diretamente no protocolo H.323, ou seja, torná-lo capaz de realizar a conexão ponto a ponto nesse tipo de rede sem o auxílio da ferramenta desenvolvida; bem como a aplicação destas mesmas técnicas de conexão em outros protocolos tanto de multimídia como de outras áreas.

8 Referências Bibliográficas

Comer, Douglas E. (2000) “Internetworking with TCP/IP Vol.1, 4th Edition”, Prendice-Hall.

CRAIG, Southern. Open H.323 project, [S.l.], ago. 2001. Disponível em: <<http://www.openh323.org>>. Acesso em: 12 fev. 2004.

DAVIDSON, J.; PETERS, J.; GRACEY, B. Voice over IP Fundamentals

DOMINGUES, Miriam Lúcia Campos Serra. Protocolos de Dados para Conferências Multimídia. 2000. Dissertação de Mestrado (Ciências da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre.

G.107 ITU-T Recommendation, The E-Model, a Computational Model for Use in Transmission Planning, 2002.

G.711, ITU-T Recommendation, Pulse Code Modulation (PCM) of Voice Frequencies, 1988.

G.723, ITU-T Recommendation, Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s, 1996.

G.729, ITU-T Recommendation, Coding of Speech at 8 kbit/s Using Conjugate Structure-algebraic-code-excited Linear Prediction (CSs-ACELP), 1996.

MATSUDA, Davin. Nyquist Theorem. Disponível em: <http://whatis.techtarget.com/definition/0,,sid9_gci812005,00.html>. Acesso em: 04 out. 2004

MONTEIRO, Rafael Flister. Implementação de Transporte Robusto de Voz em Redes Baseadas em Protocolos IP. 2000. 85 f. Dissertação de Mestrado (Engenharia Elétrica) - Universidade Federal de Minas Gerais, Belo Horizonte.

[NAT-TERM] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, Agosto 1999.

[NAT-TRAD] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, Janeiro 2001.

[NAT-PROT] M. Holdrege and P. Srisuresh, "Protocol Complications with the IP Network Address Translator", RFC 3027, Janeiro 2001.

[NAT-PT] G. Tsirtsis and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, Fevereiro 2000.

NÓBREGA, Obionor. Um Algoritmo adaptativo de transporte para serviços de Voz sobre IP, Simpósio Brasileiro de Redes de Computadores, 2001 Curitiba Anais... Curitiba UFPR, 2001.

NTKernel WinPkFilter. Disponível em: <<http://www.ntkernel.com>>. Acesso em: 04 out. 2004.

OLIVEIRA, Sérgio. Telefonia IP para ambientes móveis usáveis: Simpósio Brasileiro de Redes de Computadores, 2001

[RFC 1597] Address Allocation for Private Internets

[RFC2663] IP Network Address Translator (NAT) Terminology and Considerations

SCHULZRINNE, H.; CASNER S.; FREDERICK, R., RTP: A Transport Protocol for Real-Time Applications, 1996.

9 Anexos

9.1. Artigo

FERRAMENTA PARA COMUNICAÇÃO VOIP USANDO O PADRÃO H.323 EM REDES COM SERVIDORES NAT

Rodrigo Jean de Souza, Wagner Cunha

*UFSC - Universidade Federal de Santa Catarina
INE - Departamento de Informática e Estatística
Curso de Graduação em Sistemas de Informação
{rjsouza, wcunha}@inf.ufsc.br*

Resumo. Com o aumento contínuo do número de usuários na Internet, os endereços IP públicos tornaram-se insuficientes. Para resolver esse problema, passaram a ser usados servidores NAT, os quais acabaram dificultando a operação do padrão H.323 que é complexo e exige comunicação ponto-a-ponto. Nos NATs não é possível fazer esse tipo de comunicação, por isso, o H.323 não funciona para os usuários em redes com dispositivos NAT. O presente trabalho propõe uma solução eficiente para transpor os dispositivos NAT e estabelecer comunicação entre dois pontos de rede separados por tradutores de endereços, o que é necessário para o funcionamento do H.323.

Palavras-chave: Voz sobre IP, Protocolo H.323, NAT, Dispositivos de Tradução de Endereço.

1. Introdução

O crescimento da Internet aliado ao aumento constante na velocidade dos enlaces viabilizou a criação de tecnologias para aplicações de transmissão de áudio digital em tempo real. Esta tecnologia chama-se Voz sobre IP ou VoIP

O padrão H.323, proposto pela International Telecom Union (ITU-T), é o mais difundido no mercado para transmitir voz. Esse modelo é focado na conexão e no controle da chamada para transmissão de conteúdo (voz) entre os computadores.

O crescimento da Internet gerou o problema da falta de endereços IP, contornado com os servidores NAT (*Network Address Translation*), que possibilitam o acesso à Internet para diversas máquinas em uma rede local com apenas um endereço IP válido.

Os servidores NAT, no entanto, causaram problemas na comunicação do padrão H.323 pois, em função de seus procedimentos de funcionamento, não oferecem comunicação direta entre o usuário da rede interna e a Internet, o que inviabiliza a utilização do padrão H.323.

O objetivo deste trabalho é analisar a comunicação de voz através da Internet e propor uma ferramenta para que os usuários do padrão H.323 possam se comunicar mesmo que façam uso de servidores NAT. A implementação dessa ferramenta baseia-se em técnicas de tunelamento e filtragem de pacotes e *UDP Hole Punching* para criar o canal de comunicação que será usado para estabelecer a sessão H.323.

2. Transmissão de Voz sobre IP

Transformar voz em mais uma aplicação IP dentro de uma rede de dados que utilize IP como protocolo de nível de rede. Esse é o conceito da tecnologia Voz sobre IP, que permite a digitalização e a codificação de voz e o empacotamento de dados IP para a transmissão em uma rede IP. As aplicações VoIP são mais facilmente encontradas em redes corporativas privadas. Na Internet essas aplicações não são aconselháveis para fins profissionais, já que o protocolo IP não oferece padrões de Qualidade de Serviço (QoS). Isso compromete a clareza da voz, que é dependente do tráfego de dados existentes no momento da conversa.

Há alguns anos já é esperado, independente da tecnologia adotada, o movimento de integração entre voz e dados na mesma infra-estrutura de rede. As vantagens são claras, pois há redução custos com a integração. Contribuíram para Voz sobre IP (VoIP) tornar-se uma realidade o aumento do leque de novas aplicações, da disseminação de microcomputadores pessoais e da banda de transmissão disponível para o usuário.

2.1 Protocolo H.323

O padrão H.323 descreve como áudio, vídeo, dados e informações de controle podem ser gerenciados em uma rede baseada em pacotes para disponibilizar serviços de conversação. O padrão H.323 provê uma arquitetura de ação de dados multimídia, para

redes baseadas no protocolo IP. O H.323 permite ainda a operação conjunta de produtos de multimídia e aplicações de fabricantes diferentes.

A International Telecom Union (ITU-T), organismo que define padrões para redes de computadores e telecomunicações, recomenda o H.323. A primeira versão da especificação do H.323 foi aprovada em 1996 pelo Grupo de estudos 16 do ITU e a segunda em janeiro de 1998. O H.323 faz parte de uma série de padrões de comunicações que permitem vídeo conferência e VoIP através de redes.

A flexibilidade é uma das principais características do H.323, que permite a aplicação de voz, vídeo conferência e multimídia

3. NAT (Network Address Translation)

Cada computador que acessa a Internet deve ter o protocolo TCP/IP configurado. Para isso, cada computador da rede interna precisa de um endereço IP válido na Internet. E não há endereços IPs suficientes.

A solução para essa questão veio com a criação do NAT. Com ele a empresa fornece acesso à Internet para um grande número de computadores da rede interna com um número bem menor de endereços IP, válidos na Internet.

Traduzir os endereços privados, que não são válidos na Internet, para o endereço válido da interface pública do servidor é a principal função do NAT.

3.1 UDP Hole Punching

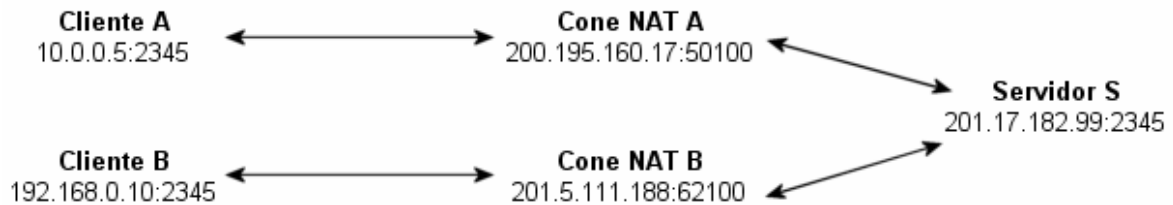
Essa é considerada a melhor técnica para implementar comunicação ponto-a-ponto entre estações que estão em uma rede controlada por NATs. O *UDP Hole Punching* aproveita as propriedades comuns de *firewalls* e NATs e promove a estabilização da conexão direta entre as estações que estão sob o controle desses dispositivos.

3.1.1 Cenário: Estações atrás de diferentes NATs

Suponha que os clientes A e B possuem endereços IP privados e estão sob o controle de diferentes NATs e um servidor S que usa a porta UDP 2345. A e B possuem uma conexão UDP com o servidor S. Isso faz com que o NAT A associe a porta 50100

à sessão do servidor e o NAT B associe a porta 62100 à sessão do servidor respectivamente.

UDP Hole Punching: Diferentes NATs



Agora o cliente A quer estabelecer uma conexão com o cliente B. Para isso, ele envia um pacote para 201.5.111.188, porta 62100, informado pelo servidor S. Esse pacote será descartado pelo NAT (caso não seja um *Full Cone NAT*) porque a origem não combina com o endereço do servidor S que foi o endereço de destino no pacote do cliente B para estabelecer a conexão com S. O mesmo acontecerá se o cliente B tentar conexão com o A.

Imagine agora que o cliente A queira enviar pacotes UDP para o endereço público de B. Ele pede ao servidor que solicite que B faça o mesmo em relação ao endereço público de A. Os pacotes de saída de A fazem com que o NAT A crie uma sessão entre o IP privado de A e o IP público de B. Ao mesmo tempo, os pacotes de saída de B fazem o NAT B criar uma sessão entre o IP privado de B e o IP público de A. Como foram criadas sessões UDP em cada direção, os clientes A e B podem se comunicar diretamente sem a interferência do servidor S.

4. Incompatibilidade do protocolo H.323 com os servidores NAT

O H.323 é um protocolo complexo e possui uma série de incompatibilidades com os tradutores de endereços IP. Esses problemas inviabilizam a criação de uma sessão H.323 entre usuários que estejam em ambientes que possuam servidores NAT.

O problema das portas dinâmicas está relacionado com a solução de mapeamento de portas no NAT para o estabelecimento de sessões H.323. Já o problema da área de dados está diretamente ligado à solução proposta neste trabalho.

5. Funcionamento da ferramenta

A ferramenta proposta utiliza as técnicas de *UDP Hole Punching*, tunelamento de pacotes sobre UDP (*UDP Tunneling*) e de filtragem de pacotes (*Application Level Gateway*) para estabelecer uma sessão H.323 entre estações separadas por NATs.

A técnica *UDP Hole Punching* é vital para a criação do canal UDP por onde os pacotes H.323 serão transmitidos.

A ferramenta desenvolvida deve estar em funcionamento em ambos os pontos da rede, juntamente com o aplicativo H.323 que será utilizado. A mesma executa duas atividades principais para permitir o estabelecimento da sessão H.323:

- Monitoramento, filtragem e injeção dos pacotes H.323 que são recebidos e enviados pelo ponto de rede.
- Estabelecimento do canal UDP com o destino da conexão (com auxílio do servidor) por meio da técnica de *UDP Hole Punching*.

5.1 Estabelecimento do canal UDP

O aplicativo H.323 gera o primeiro pacote de rede com o intuito de abrir a conexão TCP que inicia o protocolo. Esse pacote é interceptado e bloqueado localmente pela ferramenta antes de chegar ao dispositivo de rede. Do cabeçalho IP do pacote interceptado, a ferramenta retira o endereço IP do ponto de destino com o qual o usuário deseja se comunicar, o que é utilizado para a criação do túnel UDP.

Em seguida é estabelecido o canal UDP com este ponto de destino fazendo uso da técnica de *UDP Hole Punching*. É por meio desse canal que será enviado o pacote interceptado no passo acima e os subsequentes.

5.2 Descoberta dos endereços IP públicos

Deste processo de estabelecimento do canal UDP também são descobertos os endereços públicos de ambas as redes, os quais serão utilizados nos passos seguintes. Os endereços são visíveis ao servidor que auxilia na criação do canal UDP, e esta informação é transmitida para a ferramenta em ambos os pontos.

5.3 Alteração do endereço IP de origem no cabeçalho dos pacotes

Tendo o canal UDP criado, a ferramenta poderá transmitir por meio dele os pacotes de rede que serão gerados pelo aplicativo H.323 para o outro ponto da rede.

Para que os pacotes possam ser respondidos corretamente pelo aplicativo H.323 do outro ponto, o cabeçalho IP do pacote deve ter o campo de endereço IP origem (*source*) corrigido. Nesse campo se encontra o IP interno desse ponto de rede, o qual é substituído pelo endereço público da rede. Após essa correção o campo de verificação de erros (*checksum*) do cabeçalho IP e do cabeçalho TCP são recalculados e substituídos pela ferramenta, caso contrário o mesmo seria rejeitado no destino.

5.4 Tunelamento dos pacotes

O pacote interceptado é então inserido em um novo pacote UDP o qual é transmitido pelo túnel. No ponto destino, o pacote gerado pelo aplicativo H.323 é retirado da área de dados do pacote UDP recebido.

5.5 Alteração do endereço IP de destino no cabeçalho dos pacotes

Antes de ser passado para a camada de rede do sistema operacional no ponto destino, o pacote precisa ter o campo de endereço destino do cabeçalho IP corrigido.

O cabeçalho do pacote contém nesse campo o IP público da rede deste ponto, o qual é trocado para o endereço IP privado. Novamente o campo de verificação de erros (*checksum*) do cabeçalho IP e TCP é corrigido para que o pacote seja aceito.

5.6 Injeção do pacote recebido

Nesse passo o pacote é enfim inserido na camada de rede do sistema operacional. O aplicativo H.323 destino irá recebê-lo de forma transparente, como se tivesse sido enviado normalmente por um ponto dentro da mesma rede.

Este procedimento é realizado de forma idêntica para todos os pacotes de sinalização e de resposta enviados por ambas as partes, exceto para os pacotes H.323 que contém endereços IP de origem e destino dentro da área de dados do pacote que requerem um passo adicional:

5.7 Alteração dos endereços IP dentro dos dados do pacote

Certos pacotes do protocolo H.323 contém os endereços IP de origem e destino da sessão também dentro da sua área de dados. O aplicativo H.323 ao qual o pacote se destina utiliza o endereço recebido dentro do pacote para criar as conexões subsequentes.

Dessa forma, o endereço IP contido dentro do pacote H.323 precisa ser alterado da mesma forma que o endereço IP do cabeçalho. Para isso, a ferramenta procura pelo IP privado do ponto de rede dentro da área de dados do pacote e o substitui pelo IP público da rede. Já no ponto de destino a ferramenta substitui o endereço IP público da rede pelo IP privado do ponto.

Nesse ponto a sessão H.323 foi estabelecida, e o áudio pode começar a ser transmitido. O envio dos pacotes de áudio também segue o mesmo procedimento, pois mesmo os pacotes de áudio sendo UDP, esses também precisam ser enviados pelo túnel, visto que o *UDP Hole Punching* não permite a transmissão de pacotes UDP por uma porta específica – no caso a porta que foi definida pela conexão H.323 – apenas pela porta estabelecida pelo NAT.

9.2. Código fonte

```

unit uTunnelUDP;

interface

uses
  Windows, SysUtils, Snoop, Winsock, iphlp, winpkf, uudp;

type
  TTunnelUDP = class
  private
    FUDP: TUDP;
    FPorta: Cardinal;

    FIPLocal: String;
    FIPNatOrigem: String;
    FIPNatDestino: String;

    FFilt: THANDLE;
    FAdapterMode: ADAPTER_MODE;
    FEvent: THANDLE;
    FAdapts: TCP_AdapterList;
    FBufferA: INTERMEDIATE_BUFFER;
    FReadRequestA: ETH_REQUEST;
    FBufferB: INTERMEDIATE_BUFFER;
    FReadRequestB: ETH_REQUEST;
    FAdapter: THANDLE;
    FUltimaPosicaoPct: Integer;

    procedure UDPSocketRead(Sender: TObject);
  public
    constructor Create;
    destructor Destroy; override;

    procedure Conectar;
    procedure Desconectar;
    procedure Capturar;

    property IPLocal: String read FIPLocal write FIPLocal;
    property IPNatOrigem: String read FIPNatOrigem
      write FIPNatOrigem;
    property IPNatDestino: String read FIPNatDestino
      write FIPNatDestino;
    property Porta: Cardinal read FPorta write FPorta;
  end;

implementation

{ TTunnelUDP }

constructor TTunnelUDP.Create;
begin
  FUDP := TUDP.Create;
  FUDP.PortaRemota := FPorta;
  FUDP.PortaLocal := FPorta;
  FUDP.OnSocketRead := UDPSocketRead;

```

```

FFilt := OpenFilterDriver('NDISRD');

if IsDriverLoaded(FFilt) then
begin
  GetTcpipBoundAdaptersInfo (FFilt, @FAdapts);

  FAdapter :=
    FAdapts.m_nAdapterHandle[FAdapts.m_nAdapterCount];
  FReadRequestB.hAdapterHandle := FAdapter;
  FReadRequestB.EthPacket.Buffer := @FBufferB;
end;

FUltimaPosicaoPct := 1;
end;

destructor TTunelUDP.Destroy;
begin
  FAdapterMode.dwFlags := 0;
  FAdapterMode.hAdapterHandle := FAdapter;
  SetAdapterMode(FFilt, @FAdapterMode);

  SetPacketEvent(FFilt, FAdapter, 0);

  if FEvent <> 0 then CloseHandle(FEvent);

  CloseFilterDriver (FFilt);

  inherited Destroy;
end;

procedure TTunelUDP.Conectar;
begin
  FUDP.Ativo := False;
  FUDP.IPLocal := FIPLocal;
  FUDP.IPRemoto := FIPNatDestino;
  FUDP.Ativo := True;
end;

procedure TTunelUDP.Desconectar;
begin
  FUDP.Ativo := False;
end;

procedure TTunelUDP.UDPSocketRead(Sender: TObject);
var
  DataLen: Integer;
  pEtherHeader: TEtherHeaderPtr;
  pIPHeader: TIPHeaderPtr;
  pTCPHDR: pTCP_HDR;
  pUDPHDR: pUDP_HDR;
  pTCPData: Pointer;
  IPStr: String;
  TCPDataLength, position: Integer;
  IPLocal: DWord;
begin
  DataLen := FUDP.ReceiveBuf(
    FBufferB.m_IBuffer[FUltimaPosicaoPct],
    MAX_ETHER_FRAME);

```

```

if DataLen <= 0 then
  Exit;

FUltimaPosicaoPct := FUltimaPosicaoPct + DataLen;

if ((FBufferB.m_IBuffer[FUltimaPosicaoPct-1]
  <> Ord('&')) and
  (FBufferB.m_IBuffer[FUltimaPosicaoPct-2]
  <> Ord('&'))) then
  Exit;

FReadRequestB.EthPacket.Buffer.m_Length :=
  FUltimaPosicaoPct;
FReadRequestB.EthPacket.Buffer.m_dwDeviceFlags :=
  PACKET_FLAG_ON_RECEIVE;

pEtherHeader := TEtherHeaderPtr(@FBufferB.m_IBuffer[1]);
if ntohs(pEtherHeader.h_proto) = ETH_P_IP then
begin
  pIPHeader := Pointer(Integer(pEtherHeader) +
    SizeOf(TEtherHeader));

  IPStr := snoopIP2Str(ntohl(pIPHeader.DestIp));
  if (pIPHeader.Protocol = PROTO_TCP) then
  begin
    pIPHeader.DestIp := htonl(snoopStr2IP(FIPLocal));
    pIPHeader.CheckSum :=
      htons(snoopIPChecksum(PIP_HDR(pIPHeader)));
    pTCPHDR := Pointer(Integer(pIPHeader) +
      SizeOf(IP_HDR));

    pTCPData := Pointer(Integer(pTCPHDR) +
      SizeOf(TCP_HDR));
    TCPDataLength := ntohs(pIPHeader.TotalLen) -
      SizeOf(TCP_HDR);
    IPLocal := ntohl(snoopStr2IP(FIPNatOrigem));

    for position := 0 to TCPDataLength -1 do
      if (PChar(Integer(pTCPData) + position)^ = #0) and
        (PDWord(Integer(pTCPData) + position+1)^ =
          IPLocal) then
        begin
          PDWord(Integer(pTCPData) + Position+ 1)^ :=
            ntohl(snoopStr2IP(FIPLocal));
          Break;
        end;

    pTCPHDR.CheckSum :=
      htons(snoopTCPChecksum(PIP_HDR(pIPHeader),
        PTCP_HDR(pTCPHDR)));
  end else
  if (pIPHeader.Protocol = PROTO_UDP) then
  begin
    pIPHeader.DestIp := htonl(snoopStr2IP(FIPLocal));
    pIPHeader.CheckSum :=
      htons(snoopIPChecksum(PIP_HDR(pIPHeader)));
    pUDPHDR := Pointer(Integer(pIPHeader) +
      SizeOf(IP_HDR));
  end;
end;

```



```

        pUDPHDR.Checksum :=
            htons(snoopUDPChecksum(PIP_HDR(pIPHeader),
                PUDP_HDR(pUDPHDR)));
    end;

    SendPacketToMstcp(FFilt, @FReadRequestB);
end;

    FUltimaPosicaoPct := 1;
end;

procedure TTunelUDP.Capturar;
var
    pEtherHeader: TEtherHeaderPtr;
    pIPHeader: TIPHeaderPtr;
    pUDPHdr: pUDP_HDR;
    pTCPHDR: pTCP_HDR;
    pTCPData: Pointer;
    TCPDataLength: Integer;
    IPStr: String;
    IPLocal: DWord;
    position: Integer;
begin
    FAdapterMode.hAdapterHandle := FAdapter;
    FAdapterMode.dwFlags := MSTCP_FLAG_SENT_TUNNEL or
        MSTCP_FLAG_RECV_TUNNEL;

    FEvent := CreateEvent(Nil, TRUE, FALSE, Nil);

    if (FEvent <> 0) and (SetPacketEvent(FFilt, FAdapter,
        FEvent) <> 0) then
    begin
        FReadRequestA.EthPacket.Buffer := @FBufferA;
        FReadRequestA.hAdapterHandle := FAdapter;

        SetAdapterMode(FFilt, @FAdapterMode);
    end;

    while ReadPacket(FFilt, @FReadRequestA) <> 0 do
    begin
        pEtherHeader := TEtherHeaderPtr (@FBufferA.m_IBuffer[1]);

        if ntohs(pEtherHeader.h_proto) = ETH_P_IP then
        begin
            pIPHeader := Pointer(Integer(pEtherHeader) +
                SizeOf(TEtherHeader));

            if ((pIPHeader.Protocol) = PROTO_TCP) or
                ((pIPHeader.Protocol) = PROTO_UDP) then
            begin
                pUDPHdr := Pointer(Integer(pIPHeader) +
                    SizeOf(IP_HDR));

                if FBufferA.m_dwDeviceFlags =
                    PACKET_FLAG_ON_SEND then
                begin
                    if ((ntohs(pUDPHdr.Source) <> FPorta) and
                        (ntohs(pUDPHdr.Destination) <> FPorta)) and
                        (snoopIP2Str(ntohl(pIPHeader.DestIp)) =

```

```

        FIPNatDestino) then
begin
    IPStr :=
        snoopIP2Str(ntohl(pIPHeader.SourceIp));
    pIPHeader.SourceIp :=
        htonl(snoopStr2IP(FIPNatOrigem));
    pIPHeader.CheckSum :=
        htons(snoopIPChecksum(PIP_HDR(pIPHeader)));

    if (pIPHeader.Protocol = PROTO_TCP) then
    begin
        pTCPHDR := Pointer(Integer(pIPHeader) +
            SizeOf(IP_HDR));
        pTCPData := Pointer(Integer(pTCPHDR) +
            SizeOf(TCP_HDR));
        TCPDataLength := ntohs(pIPHeader.TotalLen) -
            sizeof(TCP_HDR);
        IPLocal := ntohl(snoopStr2IP(FIPLocal));
        for position := 0 to TCPDataLength -1 do
            if (PChar(Integer(pTCPData) +
                position)^ = #0) and
                (PWord(Integer(pTCPData) +
                    position+1)^ = IPLocal) then
            begin
                PWord(Integer(pTCPData)+Position+1)^ :=
                    ntohl(snoopStr2IP(FIPNatOrigem));
                Break;
            end;
        pTCPHDR.CheckSum :=
            htons(snoopTCPChecksum(PIP_HDR(pIPHeader),
                P_TCP_HDR(pTCPHDR)));
    end else
    if (pIPHeader.Protocol = PROTO_UDP) then
    begin
        pUDPHDR := Pointer(Integer(pIPHeader) +
            SizeOf(IP_HDR));
        pUDPHDR.Checksum :=
            htons(snoopUDPChecksum(PIP_HDR(pIPHeader),
                PUDP_HDR(pUDPHDR)));
    end;

    FBufferA.m_IBuffer[FBufferA.m_Length+1] :=
        Ord('&');
    FBufferA.m_IBuffer[FBufferA.m_Length+2] :=
        Ord('&');
    FBufferA.m_IBuffer[FBufferA.m_Length+3] :=
        Ord('&');
    FBufferA.m_IBuffer[FBufferA.m_Length+4] :=
        Ord('&');

    FUDP.IPRemoto := FIPNatDestino;
    FUDP.PortaRemota := FPorta;
    FUDP.SendBuf(pEtherHeader^,
        FBufferA.m_Length+4);

    Continue;
end;
end;
end;
end;

```

```

    end;

    if FBufferA.m_dwDeviceFlags = PACKET_FLAG_ON_SEND then
        SendPacketToAdapter(FFilt, @FReadRequestA)
    else
        SendPacketToMstcp(FFilt, @FReadRequestA);
    end;
end;

end.

unit UUDP;

interface
Uses Windows, Classes, Winsock, Messages;

Type
    TUDP = Class
    Protected
        fHandle : Integer;
        fAtivo : Boolean;
        fBuffer : Array[0..4095] Of Char;

        fIPRemoto : String;
        fIPRemotoBinario : Integer;
        fPortaRemota : Word;

        fIPLocal : String;
        fIPLocalBinario : Integer;
        fPortaLocal : Word;

        fSocketAddressLocal : TSocketAddrIn;
        fSocketAddressRemoto : TSocketAddrIn;
        fSocket : TSocket;
        fPHostName : PHostEnt;
        fSession : TWSAData;
        fLookupHandle : THandle;
        fLookupBuffer : Array[0..MAXGETHOSTSTRUCT -1] Of Char;

        fOnSocketRead : TNotifyEvent;
        fOnSocketWrite : TNotifyEvent;

    Procedure SetAtivo(pAtivo : Boolean);

    Procedure SetIPRemoto(Const pIPRemoto : String);
    Procedure SetPortaRemota(pPortaRemota : Word);

    Procedure SetIPLocal(Const pIPLocal : String);
    Procedure SetPortaLocal(pPortaLocal : Word);

    Procedure WndProc(Var pMsg : TMessage);
Public
    Constructor Create;
    Destructor Destroy; Override;

    Function SendBuf(Var pBuffer; pBufferLength : Integer) : Integer;
    Function ReceiveBuf(Var pBuffer; pBufferLength : Integer) : Integer;
    Function SendString(pString : String) : Integer;
    Function ReceiveString(Var pString : String) : Integer;

```

```

Property Ativo : Boolean Read fAtivo Write SetAtivo;

Property IPRemoto : String Read fIPRemoto Write SetIPRemoto;
Property PortaRemota : Word Read fPortaRemota Write
SetPortaRemota;

Property IPLocal : String Read fIPLocal Write SetIPLocal;
Property PortaLocal : Word Read fPortaLocal Write SetPortaLocal;

Property OnSocketRead : TNotifyEvent Read fOnSocketRead Write
fOnSocketRead;
Property OnSocketWrite : TNotifyEvent Read fOnSocketWrite Write
fOnSocketWrite;
End;

implementation
Uses SysUtils;

Const
  WSA_VERSION_REQUIRED = $101;
  WM_LOOKUPCOMPLETE = WM_USER + 1982;
  WM_SOCKETIO = WM_USER + 1983;

Type
  TCMLookupComplete = Record
    Msg : Cardinal;
    LookupHandle : THandle;
    AsyncBufLen : Word;
    AsyncError : Word;
    Result : Longint;
  End;

Procedure TUDP.SetAtivo(pAtivo : Boolean);
Var
  vErro : Integer;
Begin
  If pAtivo <> fAtivo Then
  Begin
    fAtivo := pAtivo;

    If fAtivo Then
    Begin
      vErro := WSASStartUp( WSA_VERSION_REQUIRED, fSession );

      If vErro <> 0 Then
      Begin
        Raise Exception.Create( 'TUDP.SetIPRemoto - Erro em
WSAstartup!' );
        Exit;
      End;

      fSocket := Socket( AF_INET, SOCK_DGRAM, IPPROTO_IP );

      If fSocket = INVALID_SOCKET Then
        Raise Exception.Create( 'TUDP.SetIPRemoto - Erro ao criar
socket! Código do erro: '+ IntToStr( WSAGetLastError ) );
    End;
  End;
End;

```

```

        If Bind( fSocket, fSocketAddressLocal, SizeOf(
fSocketAddressLocal )) <> 0 Then
            Raise Exception.Create( 'TUDP.SetAtivo - Erro ao executar
bind! Código do erro: '+ IntToStr( WSAGetLastError ));

        If WSAAsyncSelect( fSocket, fHandle, WM_SOCKETIO, FD_READ Or
FD_WRITE ) <> 0 Then
            Raise Exception.Create( 'TUDP.SetAtivo - Erro ao executar
WSAAsyncSelect! Código do erro: '+ IntToStr( WSAGetLastError ));
        End
    Else
        Begin
            If WSAAsyncSelect( fSocket, fHandle, 0, 0 ) <> 0 Then
                Raise Exception.Create( 'TUDP.SetAtivo - Erro ao executar
WSAAsyncSelect! Código do erro: '+ IntToStr( WSAGetLastError ));

                CloseSocket( fSocket ); //Fecha socket

            If WSACleanUp = SOCKET_ERROR Then
                Raise Exception.Create( 'TUDP.SetAtivo - Erro ao executar
WSACleanUp! Código do erro: '+ IntToStr( WSAGetLastError ));
            End;
        End;
    End;

Procedure TUDP.SetIPRemoto(Const pIPRemoto : String);
Begin
    fIPRemoto := pIPRemoto;
    fIPRemotoBinario := Inet_Addr( PChar( pIPRemoto )); //Endereço no
formato x.y.z.w ?

    If fIPRemotoBinario <> INADDR_NONE Then
        Move( fIPRemotoBinario, fSocketAddressRemoto.sin_addr, SizeOf(
fSocketAddressRemoto.sin_addr ))
    Else
        fLookupHandle := WSAAsyncGetHostByName( fHandle,
WM_LOOKUPCOMPLETE, PChar( pIPRemoto ), fLookupBuffer, MAXGETHOSTSTRUCT
);
    End;

Procedure TUDP.SetPortaRemota(pPortaRemota : Word);
Begin
    fPortaRemota := pPortaRemota;
    fSocketAddressRemoto.sin_port := htons( fPortaRemota );
End;

Procedure TUDP.SetIPLocal(Const pIPLocal : String);
Begin
    fIPLocal := pIPLocal;
End;

Procedure TUDP.SetPortaLocal(pPortaLocal : Word);
Begin
    fPortaLocal := pPortaLocal;
    fSocketAddressLocal.sin_port := htons( pPortaLocal );
End;

Procedure TUDP.WndProc(Var pMsg : TMessage);
Begin

```

```

If ( pMsg.Msg = WM_SOCKETIO ) And ( fAtivo ) Then
Begin
  Case WSAGETSELEVENT( pMsg.lParam ) Of
    FD_READ :
      If Assigned( fOnSocketRead ) Then
        fOnSocketRead( Self );

    FD_WRITE :
      If Assigned( fOnSocketWrite ) Then
        fOnSocketWrite( Self );
  End;
End
Else
  If ( pMsg.Msg = WM_LOOKUPCOMPLETE ) And ( TCMLookupComplete( pMsg
).LookupHandle = fLookupHandle ) Then
    Begin
      fLookupHandle := 0;

      If TCMLookupComplete( pMsg ).AsyncError <> 0 Then
        fSocketAddressRemoto.sin_addr.S_addr := 0
      Else
        fSocketAddressRemoto.sin_addr.S_addr := Integer( Pointer(
PHostEnt( @fLookupBuffer ).h_addr^ )^ );
      End
    End
  Else
    pMsg.Msg := DefWindowProc( fHandle, pMsg.Msg, pMsg.WParam,
pMsg.LParam );
  End;

Constructor TUDP.Create;
Begin
  Inherited Create;
  fAtivo := False;
  fHandle := AllocateHWND( WndProc );

  fSocketAddressLocal.sin_family := AF_INET;
  fSocketAddressRemoto.sin_family := AF_INET;
End;

Destructor TUDP.Destroy;
Begin
  SetAtivo( False );
  DeallocateHWND( fHandle );
  Inherited;
End;

Function TUDP.SendBuf(Var pBuffer; pBufferLength : Integer) : Integer;
Begin
  If fAtivo Then
    Begin
      Result := SendTo( fSocket, pBuffer, pBufferLength, 0,
fSocketAddressRemoto, SizeOf( fSocketAddressRemoto ));

      If Result = SOCKET_ERROR Then
        Raise Exception.Create( 'TUDP.SendBuf - Erro ao EnviarBuffer!
Código do erro: '+ IntToStr( WSAGetLastError ));
      End
    End
  Else
    Result := -1;
  End;

```

```

End;

Function TUDP.ReceiveBuf(Var pBuffer; pBufferLength : Integer) : Integer;
Var
  vFromLen : Integer;
Begin
  If fAtivo Then
  Begin
    vFromLen := SizeOf( fSocketAddressRemoto );
    Result := RecvFrom( fSocket, pBuffer, pBufferLength, 0,
fSocketAddressRemoto, vFromLen );
    fIpRemoto := inet_ntoa( fSocketAddressRemoto.sin_addr );
    fPortaRemota := ntohs( fSocketAddressRemoto.sin_port );

    //If Result = SOCKET_ERROR Then
    //fPrincipal.lbMensagens.Items.Insert( 0, IntToStr(
WSAGetLastError ));
    //Raise Exception.Create( 'TUDP.ReceiveBuf - Erro ao
ReceberBuffer! Código do erro: '+ IntToStr( WSAGetLastError ));
  End
  Else
    Result := -1;
  End;

Function TUDP.SendString(pString : String) : Integer;
Begin
  If fAtivo Then
  Begin
    Result := SendTo( fSocket, pString[1], Length( pString ), 0,
fSocketAddressRemoto, SizeOf( fSocketAddressRemoto ));

    If Result = SOCKET_ERROR Then
      Raise Exception.Create( 'TUDP.SendBuf - Erro ao EnviarBuffer!
Código do erro: '+ IntToStr( WSAGetLastError ));
    End
  Else
    Result := -1;
  End;

Function TUDP.ReceiveString(Var pString : String) : Integer;
Begin
  If fAtivo Then
  Begin
    Result := ReceiveBuf( pBuffer, SizeOf( pBuffer ));
    SetLength( pString, Result );
    Move( pBuffer, pString[1], Result );
  End
  Else
    Result := -1;
  End;
End;

End.

```