Central telefônica para voz sobre IP

Luís André Gomes de Abreu
Indústria de Material Bélico
Rua Monsenhor Manoel Gomes, 520, Caju
Rio de Janeiro - RJ CEP: 20931-670
lagabreu@yahoo.com.br

Roberto Miranda Gomes
Centro de Desenvolvimento de Sistemas
Esplanada dos Ministérios, bloco O, 9º andar.
Brasília - DF, CEP: 70052-900.
gomes@cdf.eb.mil.br

Talles Marcelo G. de A. Barbosa
Dep. de Engenharia de Sistemas e Computação
Instituto Militar de Engenharia
Praça General Tibúrcio, 80, Praia Vermelha,
Rio de Janeiro - RJ, CEP: 22290-970.
talles@de9.ime.eb.br

RESUMO

O crescente uso da tecnologia de VoIP(*voice over IP*- voz sobre IP)tem se mostrado bastante atraente principalmente para redes onde o tráfego pode ser controlado. Contudo, a maioria dos produtos disponível comercialmente tem um custo elevado.

O presente trabalho versa sobre como realizar esta tarefa com custo reduzido utilizando a tecnologia Java e HTTP. Para tal, exporemos sucintamente os fundamentos da tecnologia de voz sobre IP, analisando aspectos relevantes de forma resumida. Em seguida, apresentaremos a modelagem do sistema implementado. O sistema é constituído de uma interface para realização das comunicações (chamadas), que se baseia na troca de mensagens entre terminais (clientes), via programação *socket*, e de um ambiente de gerência, baseado em programação Web.

ABSTRACT

The increasing use of VoIP technology (VoIP – voice over IP), show us a very powerful solution for Computer Networks where the traffic may be controlled. But, the big part of the avaliable products is too expensive.

The present work turns on as to accomplish this task with low costs by using Java and HTTP technologies. For such we expose shortly the foundations of the voice technology on IP, analyzing its advantages and disadvantages. Soon after we presented the design of the implemented system. The system is constituted of an interface for accomplishment of the communications that bases on the change of messages among clients, through socket programming, and of a management environment based on Web programming.

Palavras-chave: VoIP, sockets, Java Sound, programação web.

1 - Introdução

O presente trabalho visa à construção de uma central telefônica que coordene a comunicação entre os diversos *hosts* de uma rede local IP. Tanto ela quanto os terminais de comunicação(clientes) foram implementados totalmente via software, sem necessidade da criação de um hardware específico, apenas utilizando recursos disponíveis em um computador pessoal multimídia genérico.

Muito se evoluiu desde a primeira transmissão de áudio em pacotes IP [1], mas, o campo de estudo de voz sobre IP teve o seu grande desenvolvimento ao longo dos anos 90. Este trabalho se propõe a iniciar os estudos a partir de um escopo menor, que são as redes locais. Cabe ressaltar que um estudo mais profundo sobre o assunto, com utilização de novos protocolos e outras tecnologias, poderia estender as funcionalidades do sistema para a grande rede.

Soluções já existentes, como o *framework* H.323 do ITU-T(*Telecommunication Standardization Sector of International Telecommunication Union*) e o pacote de recomendações baseados no SIP(*Session Initiation Protocol*) do IETF(*Internet Engineering Task Force)* não serão abordados por motivo de simplificação. Não se discute a magnitude dos trabalhos do ITU-T tanto quanto do IETF. Para maiores detalhes acerca desses trabalhos, sugere-se [2], [3], [4], [5].

Como trabalho multidisciplinar dentro da Computação, este projeto envolve as áreas de programação cliente-servidor, Banco de dados, Redes de Computadores e Sistemas Multimídia. Dito isso, pois, muito se pode acrescentar ao trabalho desenvolvido.

A seção 2 faz uma apresentação dos conceitos gerais de voz sobre o protocolo de rede IP e comenta os aspectos positivos e negativos de se investir nessa tecnologia. A seção 3 analisa a organização do projeto. A seção 4 descreve requisitos de hardware e software básicos necessários para suporte ao sistema. Por fim, a seção 5 encerra o trabalho com um resumo do que já foi desenvolvido e com propostas para trabalhos futuros.

2 - Voz sobre IP

2.1 Conceito

Voz sobre IP é uma tecnologia que permite a digitalização e codificação da voz e seu particionamento em pacotes de dados IP para a transmissão em uma rede que utilize TCP/IP [6], ou seja, Voz sobre IP é um conceito relativamente simples: transformar sinais de voz em datagramas dentro de uma rede de dados que utilize IP como protocolo de nível de rede.

2.2 TCP/IP e UDP/IP para aplicações de voz

Devido à imprevisibilidade do atraso de uma rede IP convencional, os protocolos da camada de transporte da Internet, o TCP e o UDP, não são adequados para aplicações de voz em tempo real. A utilização do TCP (*Transport Control Protocol*) implica na execução de um algoritmo de recuperação dos dados perdidos por retransmissão, assim o fornecimento dos dados deve esperar por todas as retransmissões, gerando grandes atrasos. O UDP (*User Datagrama Protocol*) evita o problema da retransmissão, fornecendo um serviço sem conexão orientado a datagrama, com a desvantagem que este protocolo não é confiável, ou seja, não tem qualquer responsabilidade quanto a perdas. Aplicações de voz precisam de certas

garantias de qualidade de serviço (*QoS - Quality of Service*), que é definido como sendo o efeito coletivo de performance que determina o grau de satisfação do usuário deste serviço específico. As redes IP convencionais não oferecem nenhum tipo de QoS, impondo o principal problema para as aplicações VoIP.

2.3 Vantagens das aplicações de voz sobre IP

Primeiramente, podemos citar o baixo custo de sua implementação, diretamente ligado à simplicidade da API (*Application Programming Interface)* de programação e do reuso da infra-estrutura de rede já instalada principalmente para o ambiente de chamada entre poucos computadores. A digitalização da voz é feita pelos computadores que executam o cliente VoIP e a transmissão realizada através da rede IP, bastando para isto uma interface de programação baseada em *sockets*. Outro aspecto inserido ao baixo custo de implementação refere-se ao meio de transmissão, que é mais barato se comparado ao sistema telefônico[7]. O uso do protocolo IP se justifica pelo de ele já ser um "protocolo comum" à vasta estrutura de redes instalada, sendo economicamente difícil modificar todo este quadro para uma nova alternativa.

Podemos citar ainda a possibilidade de oferecer outros serviços como correio de voz (implementado no software gerado pelo trabalho através da classe *voicemai*l), *call centers* via Internet, segunda linha virtual, além dos serviços já prestados numa rede IP convencional como *email*, fax, web entre outros. Como estes últimos são compartilhados, podem ser encarados como desvantagens, o que será mais bem descrito na próxima seção. A possibilidade de novos serviços implementados por software vem se tornando cada vez mais o fator decisivo na adoção desta tecnologia.

Em comparação aos sistemas telefônicos, destacamos também a melhor utilização da largura de banda, com possível uso de compactação e supressão do silêncio implementado pelos codecs. [8].Também gera uma economia na infra-estrutura e manutenção, unificando as redes de transporte, sinalização e gerência (o que também foi realizado aqui, possibilitando que esta última feita de maneira remota, a partir de qualquer ponto da rede). O tratamento digital do sinal de áudio tem demonstrado soluções boas com custos viáveis, contribuindo para a convergência da transmissão de voz em canais de dados.

Em resumo, como já foi dito, a maior vantagem da utilização de Telefonia IP não é somente baixo custo. Enquanto ligações de longa distância baratas estão incentivando o uso, as razões pelas quais as companhias são atraídas para ela são a facilidade de criação de serviços e a consolidação de suas redes, unificando voz e dados.[9]

2.4 Desvantagens e problemas das aplicações de voz sobre IP

As adversidades surgem em virtude de as redes baseadas em IP serem comutadas por pacotes, em contraponto às redes telefônicas, que são redes comutadas por circuitos.

Nas redes de comutação de circuitos, um caminho fixo é estabelecido ao se efetuar a conexão entre as entidades comunicantes. Além disto, após ser feita a alocação de banda e demais recursos da rede, esta fica dedicada somente àquela conexão, sem compartilhamento. Ganha-se, portanto, uma garantia de Qualidade de Serviço, conforme desejado. Entretanto, perde-se em eficiência no uso dos recursos da rede, já que mesmo que não se transmita nada na banda alocada, ela se encontra inutilizada.

As redes de comutação de pacotes, por sua vez, compartilham os recursos entre diversos usuários que desejam transmitir. Além disso, estas redes não alocam um caminho dedicado a uma conexão. Com isto, pacotes diferentes de um mesmo usuário podem ser transmitidos por rotas diversas. Desta forma, pode-se perder a ordenação dos pacotes transmitidos. Neste tipo de rede, também é significativa a necessidade de processamento por parte dos nós intermediários de uma conexão.

Outra dificuldade que aparece na transmissão de mídias que exigem cadência, como voz, é que o serviço prestado pelas redes IP é do tipo "melhor esforço" (*best effort*). Assim, todos os pacotes são tratados de forma igual, sem nenhuma discriminação entre os diversos tipos de tráfegos, nem atribuição de prioridades aos pacotes. O escalonamento é feito baseado em filas FIFO (*First In First Out*), onde, se houver espaço nos buffers dos roteadores, o pacote é armazenado para transmissão e, caso contrário, ele é descartado.

Mais um fator crítico para a transmissão de voz é o problema dos atrasos[10]. Estes podem ser agrupados em dois tipos: os fixos e os variáveis. As variações de atraso também recebem a denominação de *jitter*. Os dois tipos possuem naturezas e causas diferentes. Os atrasos fixos causam desconforto na conversação e os variáveis, atrapalham a ritmo da transmissão da voz.

Os atrasos fixos são ocasionados principalmente pelos fatores citados abaixo:

- **Compressão:** tempo gasto na codificação e compressão da voz em pacotes.
- **Entre processos:** atraso que ocorre em função dos *handoffs* entre os roteadores da rede.
- **Transmissão:** devido às limitações de velocidade dos enlaces.
- **Rede:** uma função das capacidades da rede.
- **Buffer**: relacionado ao tamanho do *buffer* de transmissão/recepção.
- **Descompressão:** em função do tempo de processamento dos *codecs* de decodificação e desempacotamento das informações.

Os atrasos variáveis são decorrentes do tráfego e do congestionamento da rede. Estes são causados principalmente pelo enfileiramento dos pacotes nos roteadores. [11]

Além dos atrasos, é fundamental considerar as perdas existentes na rede. Para o tráfego de voz codificado sem compressão (utilizado no presente trabalho), elas não são tão importantes. Contudo, ao comprimirmos a voz, estaremos aumentando a sensibilidade em relação às perdas e introduzindo mais uma forma de atraso fixo, o tempo de processamento do *codecs* [12]. Apesar do protocolo TCP tentar garantir a recuperação contra perdas. Existem trabalhos que retratam a recuperação de perdas pela inserção de informações redundantes[13], porém introduzem uma quantidade de tráfego desnecessária. Note que a transmissão de voz é muito mais sensível a retardos e suas variações do que a pequenas perdas e uma garantia maior só é obtida através de uma banda maior disponível ou com a adição de mecanismos que possam garantir requisitos mínimos de operação, que serão discutidos adiante.

Deve-se, também, considerar a escassez de banda. A conversação normal possui intervalos de silêncio, que pode gerar um desperdício de recursos em se tratando de reservar uma possível banda fixa, como ocorre com a telefonia convencional. A supressão do silêncio é realizada pelos VAD (*Voice Activity Detection*). Uma boa discussão pode ser encontrada em [14]

Por fim, podemos mencionar como mais uma dificuldade que se afigura o uso do protocolo UDP como transporte para aplicações de voz. Este protocolo, que não efetua a reordenação, nem a recuperação por retransmissão, tem a vantagem de ser o mais adequado para se manter a cadência da conversação. Além do mais, há alguma tolerância a perdas

quando se trata da transmissão de voz. Contudo, o UDP é do tipo datagrama, e não possui controle de congestionamento algum. Por isso, ele pode ser um emissor agressivo para a rede, elevando demais o tráfego. Existe solução para monitoramento do serviço UDP, com a inserção de cabeçalhos ao pacote da aplicação de forma a permitir identificação do tempo e conteúdo dos pacotes de voz transmitidos. Isto com a utilização dos Protocolos RTP/RTCP[15].

2.5 QoS sobre IP

O IETF regulamentou vários tipos de protocolos destinados a oferecer QoS(*Quality of Service*), de forma que esses protocolos possam contornar as limitações impostas por redes baseadas no protocolo IP. Destacamos aqui dois desses protocolos: o RTP (*Real-time Transfer Protocol*) e o RSVP (*Resource ReSerVation Protocol*), os mais usuais para VOIP.

O protocolo RTP foi definido pela RFC 1889, sua função principal é agir como uma interface melhorada entre as aplicações de tempo real e os protocolos das camadas já existentes. O RTP não garante o fornecimento de pacotes no tempo desejado ou de qualidade de serviço. Não previne que os pacotes sejam entregues fora de ordem, e também não assume que a rede na qual ele esteja sendo executado seja confiável ou que forneça a entrega de pacotes em seqüência, porém apresenta indiscutível importância para monitoração das chamadas de VoIP, apesar de em alguns casos o seu desempenho seja questionável, conforme apresenta [16]. O RTP trabalha em conjunto com outro protocolo IETF RTCP(*Real Time Control Protocol*), responsável pelo retorno das informações providas pelos cabeçalhos RTP.

Definido pela RFC 2205, o protocolo RSVP não é um protocolo de roteamento, mas sim trabalha em conjunto com ele, desempenhado a função de sinalizar aos nós a necessidade de reserva de recurso de rede, permitindo que as aplicações possam usufruir a Qualidade de Serviço requerido para seus fluxos de dados, num ambiente de serviços integrados de redes IPv4 ou Ipv6 e hoje é o principal mecanismo discutido com o intuito de prover QoS para VoIP.

Delongar sobre esse assunto foge ao escopo deste trabalho, pois muito sem tem pesquisado a respeito e além dessas propostas acima existem outras também eficientes. Duas boas referências ao tocante encontram-se em [17],[18].

2.6 Gerência baseada na Web

Gerência baseada na Web ou Gerenciamento através da web é um enfoque promissor que pode fornecer uma solução de gerenciamento verdadeiramente integrada, uniforme e simples para o gerenciamento de redes, sistemas, aplicações e serviços [19]. Calcado neste preceito, podemos afirmar que a utilização de redes IP para transmissão de voz traz de forma endêmica a vantagem de se ter a interface Web como solução inerente para interface de gerência da rede.

De acordo com [20], as ferramentas de gerenciamento de dispositivos baseada na Web podem ser classificadas em:

- Modelada em Três camadas, o software de gerenciamento roda como uma aplicação sobre o sistema operacional coletando e disseminando as informações reunidas dos dispositivos da rede para serem apresentadas pelo *browser*;
- Modelada em Duas camadas, o software de gerenciamento está integrado aos dispositivos da rede, não havendo tradução de protocolos como, por exemplo,

SNMP para HTTP (*Hyper Text Transfer Protocol*). O próprio HTTP é o protocolo de gerência.

No mundo da Telefonia IP, o modelo de três camadas é o mais usual, com a utilização do RTCP e HTTP como protocolos de gerência.

As informações de gerência são usualmente apresentadas através de uma consulta ao Banco de Dados e a geração de uma página HTML dinâmica através de um programa executado no Servidor Web, um CGI(*Commun Gatway Interface)* [21]. A utilização de Servlets Java também poderiam ser utilizadas de modo bastante eficiente[22].

O sistema apresentado a seguir utiliza uma aproximação ao modelo de três camadas, com a utilização de uma CGI.

3 - Desenvolvimento

O sistema desenvolvido neste projeto é constituído de quatro partes correlacionadas:

- Um programa cliente desenvolvido totalmente em Java (JSDK1. 3). Ele dispõe de toda um interface gráfica para a solicitação de serviços ao servidor, alteração da configuração local e gravação ou reprodução de mensagens.
- Um programa servidor também criado em Java, que gerência as diversas solicitações dos programas clientes espalhados pela rede. Ele se subdivide em duas unidades menores, representadas por objetos diferentes. A primeira é uma interface *socket* que realiza a identificação dos serviços e repassa as mensagens de resposta. Caso algumas destas requisições necessitem consultar ou alterar o banco de dados, um método do manipulador de banco de dados é acionado.
- A interface em CGI-Perl, pela qual o administrador do servidor pode interagir com o Banco de Dados.
- Um Banco de Dados através do qual o histórico do sistema e sua situação atual podem ser recuperados. Este projeto ele foi implementado na plataforma Oracle 8.

O diagrama de blocos a seguir ilustra o relacionamento entre ao diversos componentes:

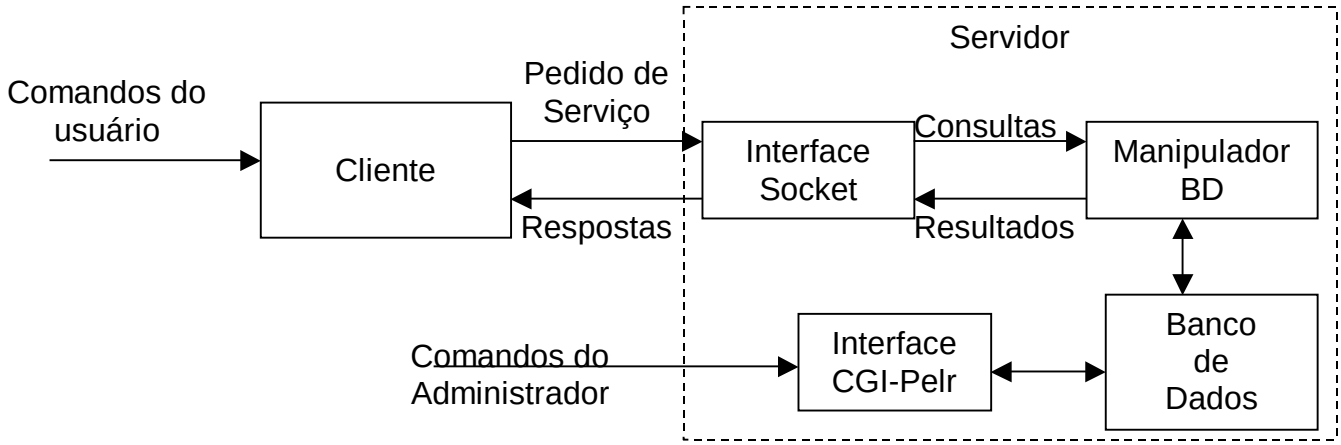

FIG. 3.1 - Diagrama de Blocos do Servidor.

Os serviços a serem prestados pelo servidor são:

- Autorização e registro de entradas de assinantes;
- Envio de mensagens de áudio para assinantes *off-line*;
- Recebimento de mensagens armazenadas;
- Listagem de assinantes *on-line* não ocupados;
- Ligação entre assinantes.

3.1 Aspectos Construtivos

Toda a comunicação via *sockets* entre o cliente e o servidor foi realizada por conexão (TCP), visto que os cabeçalhos das mensagens criadas e dos arquivos enviados não podem sofrer alteração. O tráfego do sinal de áudio foi realizado em datagrama, porém sem nenhum tratamento de qualidade de serviço. Apesar de, em uma rede local, ser pouco provável a corrupção de pacotes, é mais fácil e principalmente confiável tratar uma possível exceção (queda do cliente ou do servidor durante uma transação, por exemplo) monitorando uma conexão TCP. A escolha de datagramas para os sinais de voz foi dada pelos motivos expressos na seção anterior.

As API's de áudio digital Java utilizadas neste projeto são fornecidas no pacote javax.sound.sampled API que está disponível a partir da versão 1.3 da linguagem Java. Java Sound é a API de baixo nível para a realização e controle de entrada e saída de áudio, focando na reprodução e captura de dados de áudio formatados. Ela está baseada na modelagem, em software, de um sistema físico de áudio. Recursos de um *mixer* comum, como controle de volume, estão disponíveis para cada linha de entrada ou saída. Ela não assume nenhuma configuração específica de hardware, estando projetada para permitir a instalação de diferentes componentes no sistema e acessá-los. Para maiores detalhes consulte[23].

Outras linguagens possuem aproximações mais simples que as apresentadas aqui (como o C UNIX, que trata todos os dispositivos como arquivos)[24]. Porém, mesmo tendo uma sintaxe mais complexa, a API Java Sound possui a vantagem de fornecer grande funcionalidade. Convém lembrar que a total utilização dos recursos da API requer a correta instalação de todos os dispositivos utilizados, bem como só se exigir recursos compatíveis com o hardware utilizado.

A divisão da parte Java da central em duas unidades vem da opção em se separar a lógica da aplicação da que é utilizada no banco de dados. A vantagem deste modelo é que ele permite a reutilização da interface *socket* (lógica da aplicação) em caso de mudança do banco de dados. Uma alteração na aplicação, por sua vez, não afetará a lógica dos dados. O diagrama de seqüência da criação de uma ligação entre dois clientes ilustra bem este fato:

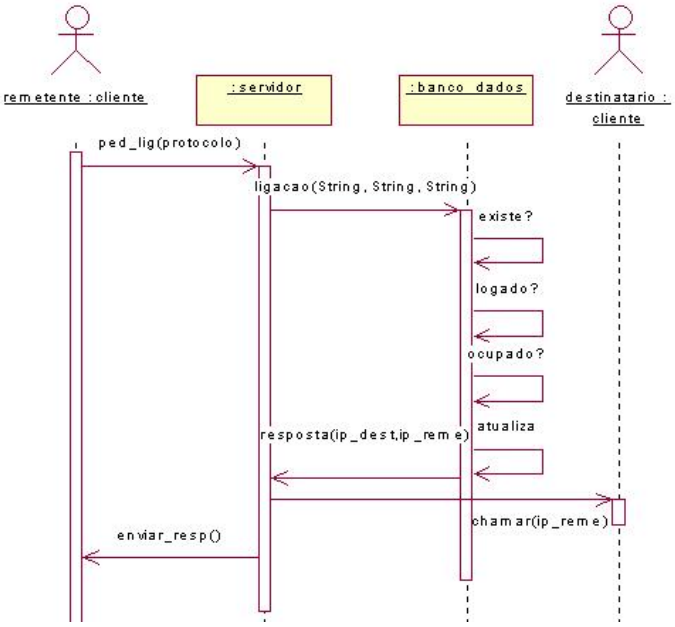

FIG. 3.2 - Pedido de Ligação: Diagrama de Seqüência

Durante o estabelecimento da ligação entre dois clientes, a central toma parte apenas na verificam da disponibilidade do destinatário e no envio de mensagens de chamada para este e o remetente. Cada uma contém o endereço IP daquele com que se irá falar. A parir daí a central sai e cada cliente cria dois *sockets* de datagrama (entrada e saída) e se interconectam.

A figura a seguir exibe a interface gráfica do programa cliente, junto com o quadro de diálogo responsável pelo início das ligações entre os assinantes.

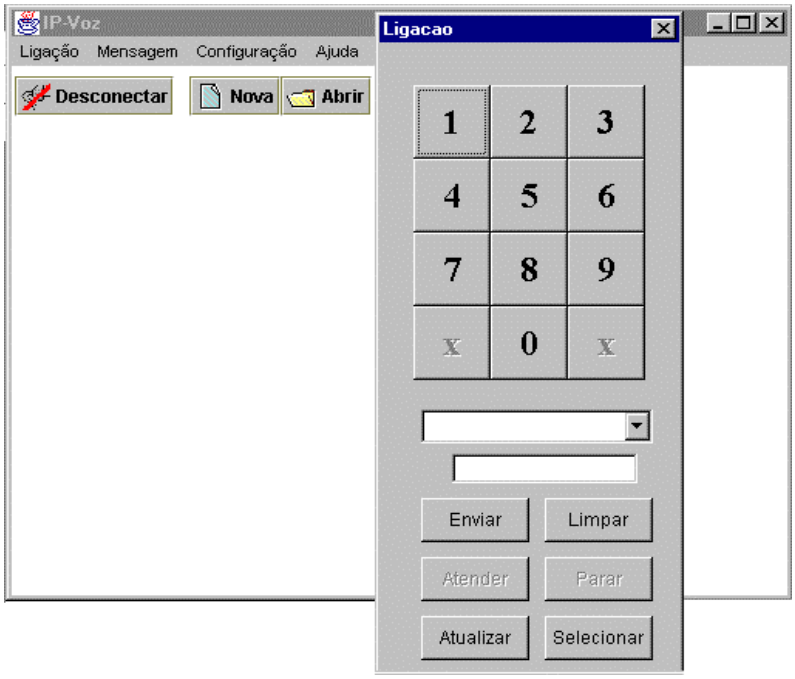

FIG. 3.3 - Aspecto da Interface Gráfica do Cliente

4 - Especificações do Protótipo

Na seção anterior, foram apresentadas as plataformas onde foram desenvolvidas as aplicações do sistema. É necessário, contudo, alguns requisitos mínimos de software e hardware que permitam o uso delas com certa qualidade.

4.1 Requisitos de software

- Para o desenvolvimento do programa cliente foi utilizado o JSDK1. 3 , não sendo possível nenhuma versão anterior, visto que a API Java Sound só está disponível a partir dessa versão. Para executá-lo, basta apenas o *runtime* da mesma versão. A parte em Java da central pode ser suportada a partir do JDK1. 1 e seu respectivo *runtime*.
- Na máquina do cliente deve estar o *driver* ODBC (*Open DataBase Conectivity*) da Microsoft. Muitos servidores de banco de dados usam protocolos específicos do fabricante. Isto faz com que tenha de saber uma linguagem diferente para cada servidor. A tarefa do ODBC é abstrair os protocolos nativos de cada tipo em uma interface comum para os clientes de banco de dados. Em verdade, o software apresentado não utiliza diretamente o ODBC, mas sim a ponte JDBC-ODBC. Ela foi a primeira solução encontrada para a época em que não existiam *drivers* JDBC puramente Java de eficiência comparável à do ODBC [25]. Por sua disponibilidade, ela foi adotada aqui, apesar da desvantagem de atrelar o projeto ao ambiente Windows.

FIG. 4.1 Papel do JDBC-ODBC

- Um servidor Web Perl precisa ser instalado no sistema de redes. Foi utilizado o ActivePerl para Windows 32 bits (existe em versões Solaris e Linux). Para o Perl 5.004 existe duas versões para sistemas Win32: uma desenvolvida pela ActiveState Tool (atual ActivePerl) e outra da distribuição padrão do Perl. Elas são bem compatíveis, existindo algumas diferenças. Com o Perl 5.005 as duas versões foram mescladas, ou seja, agora se baseiam no mesmo código fonte. O ActivePerl é útil para instalar a partir de uma versão binária e a distribuição padrão para construir o Perl a partir do código fonte.

4.2 Requisitos de hardware:

Foram os requisitos mínimos dos fabricantes para a instalação dos softwares básicos que executam os aplicativos desenvolvidos. Para aqueles em Java, é necessário, no mínimo, um computador Pentium 166 MHz. Ele deve dispor de 32 MB de memória RAM, para executar a GUI (*Graphical User Interface*) utilizada no cliente e no servidor. A placa de som é o ponto crítico do hardware. O tipo mínimo exigido é uma SoundBlaster 16, com suporte *full-duplex*. Os problemas surgem quando ela estiver mal configurada, impossibilitando o software cliente de abri-la para leitura e/ou escrita, de forma permanente ou intermitente.

Pode ser visualizada no diagrama de implantação da figura 4.2. O *host* com o programa servidor de Java e CGI-Perl ocupa posição central a qual os terminais dos assinantes devem se conectar antes de obter algum serviço. O servidor do banco dados pode estar ou não no mesmo *host* do servidor Java, sendo que no diagrama foi colocado em separado. Os terminais se conectam entre si no instante de uma ligação, dois a dois. Lembrando que a natureza servidor-cliente está nos programas e não nas máquinas, esta distribuição é uma sugestão, podendo ser alterada conforme algum impositivo.

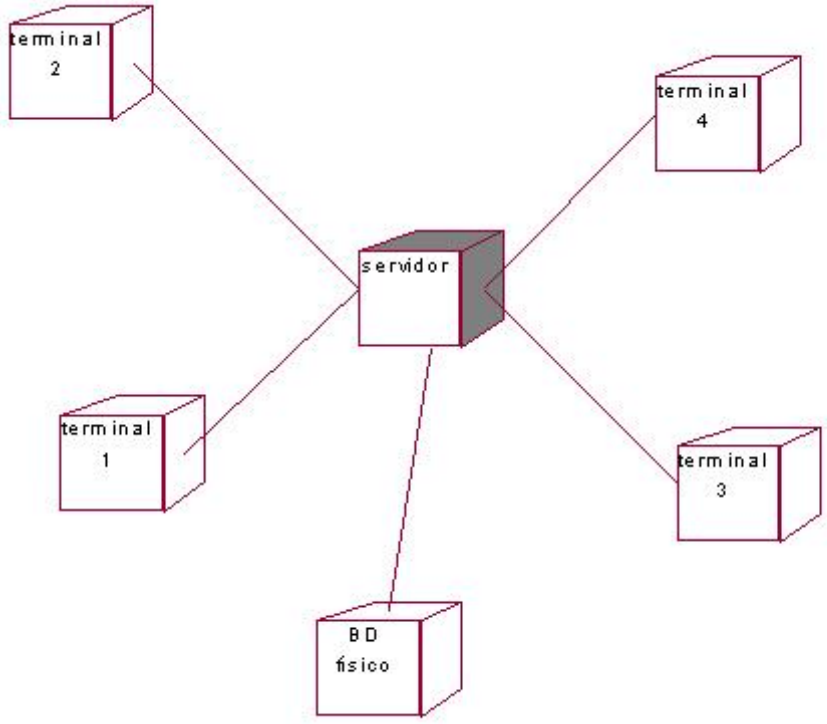

FIG. 4.2 – Diagrama de Implantação.

4.3 Requisitos de Rede

A rede utilizada nos testes realizados foi uma LAN Ethernet 10Base-T. Apesar deste tipo de rede local ser bastante popular pela usa facilidade de manutenção e custo associado, ele não é o adequado para VoIP. Os 10 Mbps de largura de banda deste modelo de rede deveria ser mais do que suficientes para transporta os 64 Kbps da codificação PCM de vários usuários, porém a utilização completa deste espaço sofre restrições. A família Ethernet utiliza o protocolo de múltiplo acesso CSMA/CD [26], que, por sua própria natureza, não permite uma reserva exclusiva de parte da banda para um usuário. Em caso de uma rede de tráfego mais intenso, a vazão de transmissão dos pacotes será bastante reduzida, devido ao próprio tratamento das colisões originadas do compartilhamento de um mesmo meio.

Enlaces que não executassem o mesmo algoritmo de contenção para tratamento de colisões proporcionariam um desempenho melhor.

5 - Conclusão

O sistema em seu estado atual cumpriu a tarefa de transmitir o sinal de voz de um ponto a outro e de realizar transações simples de gerência, servindo de protótipo a ser finalizado. Faltam ainda itens para um funcionamento de qualidade ideal. Segue-se relação dos principais melhoramentos a serem feitos:

- Melhoria nos tratamentos de exceções.
- Adição de métodos criptográficos.
- Adição de recursos de QoS.

Em teste feitos em laboratório, constatou-se a presença de um retardo da ordem de 500ms dentro do mesmo domínio de uma rede, ou seja, ainda não tolerável de acordo com as normas do ITU. Isto devido, em parte, às condições de teste submetidas, período de sobrecarga da rede e a utilização de um enlace associado a um protocolo probabilístico, CSMA/CD(*Carrier Sense Multiple Acess with Collision Detection*). Mecanismos de qualidade de serviço, não foram implementados, devido ao tempo de decorrência do projeto.

Além da API Java Sound que foi estudada anteriormente, a Java Media Framework (JMF), a API de multimídia para alto nível do Java, já suporta a execução de RTP através da API do Gerente de Sessão RTP (*RTP Session Manager* – JAWORS). Esta está contida no pacote *javax.medi.rtp*, também lançado recentemente.

Outra interessante inovação é a API Java Telephony (JTAPI), que provê recursos de telefonia a aplicações Java. Ela suporta desde o básico de telefonia, tal como emissão de chamada e reposta a elas, até formas avançadas, como centrais de chamada ou *streams* de mídia. Os acessos podem ser tanto diretos como indiretos, por intermédio da rede. A JTAPI consiste de 18 pacotes ao todo, mas todos a partir do pacote básico *javax.telephony*. É uma promissora via de interface do sistema aqui desenvolvido com a rede telefônica.

A passagem do atual projeto de aplicativo para applet pode ser tentada, porém deve se ressaltar que, segundo recomendações do próprio fabricante é bom elevar a memória RAM disponível para 48 Mb e utilizar o software Java Plug-in para carregar os applets. Este faz com que eles sejam executados pelo *runtime* Java e não pelo *browser*. Algumas API's utilizadas são novas e, dependendo da versão, podem não ser suportadas pelo programa de busca.

Uma forma de incrementar a segurança seria a adoção de algoritmos de criptografia ao sistema. As próprias características da voz humana, tão distintas para cada indivíduo, já funcionam como uma forma de assinatura. A questão principal seria evitar que estranhos tivessem acesso aos sinais de uma conversação. Em geral, mensagem oral tem um tempo de vida útil menor que o de documentos escritos, sendo muitas vezes informais. Isto permite que o algoritmo utilizado não tenha de ser muito sofisticado. Nem há como: dada a necessidade de iteração em tempo real, um processamento muito "pesado" seria comprometedor. Uma das formas mais simples [27] é escolhendo uma seqüência um tanto longo de bytes como chave e realizar um "ou-exclusivo" com cada byte do *buffer* do sinal capturado (cifra de Vernam). Para evitar a maior fraqueza deste método, que é o uso repetido das mesmas chaves, deve-se implementar um procedimento que gere chaves aleatórias para cada sessão de comunicação diferente (método *one time pad*). Pode-se estudar o caso de se adaptar o servidor para ser um centro de gerência de chaves secretas, se for o caso.

Olhando também para o lado de Banco de dados, muitas questões ainda ficam em aberto. Contribuições como a utilização de Banco de dados dedutivos, Banco de dados temporais ou

Data Warehousing como já existem em redes de telecomunicações[28], poderiam implicar em melhor performance no lado do gerenciamento *off-line*, com a possibilidade de realizar análises sobre os dados armazenados, com objetivos específicos em mente.

Além disso, a utilização de um *sniffer* com RTPdump para monitoração de pacotes RTP/RTCP em tempo real, poderia contribuir muito para contrução de uma ferramenta para gerência de performance, apropriada para medições de QoS.

A inovação do presente trabalho, com a utilização de ferramentas Java, promete muitas possibilidades na área de comunicações. Mesmo que ainda haja um caminho longo até o produto final vale a pena prosseguir no uso das plataformas utilizadas. Por ocasião de seu lançamento, Java era bem mais pobre em recursos multimídia, mas hoje, em sua versão JSDK1. 3, há uma ampla variedade de API's multimídia, sendo algumas citadas aqui. Isto permitirá que futuros desenvolvedores não tenham que iniciar seus trabalhos do marco zero, pois já dispõem de bons instrumentos. O primeiro passo já foi dado e o caminho está indicado, agora só falta segui-lo.

Agradecimentos

Este projeto teve o apoio do Ministério da defesa, entidade à qual está subordinado o Exército brasileiro. Aos professores: Edmundo Lopes Cecílio, Luis Gustavo Varges de Resende e Marcos Veloso Peixoto, do Instituto Militar de Engenharia o nosso muito obrigado pela colaboração e orientação.

Referencias

[1] D. Cohen, "Specification for the Network Voice Protocol", RFC 0741, Internet Engineering Task Force, Nov. 1977.

[2] " Visual Telephone Systems and Equipment for Local Area Networks ", Draft Rec.H323v2, Internacional Telecommunication Union, Fev. 1997.

[3] M. Handley, H. Schulzrinne, E. Schooler and J. Rosemberg, " SIP: Session Initiation Protocol ", RFC 2543, Internet Engineering Task Force, Mar. 1999.

[4] H. Schulzrinne, J. Rosemberg, " A Coparision of Sip and H323 for Internet Telephony " , http://www.nossdav.org/1998/papers, Jul. 1998.

[5] N. Kausar , J. Crowcroft, " An architeture of Conference Control Functions ",Proc. Of Photonics East, Boston, USA, Sep. 1999.

[6] M. Gonçalves. *Voice over IP Networks*, McGraw-Hill, 1999.

[7] J. Furst Jr. " Telefonia IP, O Novo Mundo das Telecomunicações, Cisco Systems, 1999.

[8] J.C. Bolot , A. Veja-Garcia, " Control Mechanisms for Packet Audio in the Internet ", Proc. IEEE Infocom 1996, São Francisco, USA, Abr. 1996.

[9] S. Oliveira, S. L. da Silva, P. F. de Moura Jr. e A. F. Loureiro, " Telefonia IP para Ambientes Móveis e Compactos" ,http://www.lecom.dcc.ufmg.br/~sergiool/telefonia/telefoni.htm, [online].

[10] A. Percy, " Understanding Latency in IP Telephony ", Brooktrout Technology Inc., http://www.brooktrout .com, [online].

[11] A. Watson, "Evaluating Real-Time Multimedia Audio and Video Quality" , CHI'97 Doctoral Consortion, Jan. 1997.

[12] N.L.L. Fernandes, " Voz sobre IP: Uma visão geral", http://www.gta.ufrj.br, [online].

[13] D.R. Figueiredo, B.R. Brandi e E. de Sousa e Silva, " Mecanismos para recuperação de perdas de pacotes de voz na Internet " , Anais do 16^{O} Simpósio Brasileiro de Redes de Computadores, Rio de Janeiro, Brasil, Maio 1998.

[14] H. Schulzrinne, W. Jiang, " Analysis of On-Off Patterns in VOIP and Their Effect on Voice Traffic Aggregation", http//www.cs.columbia.edu/~wenyu, 1999.

[15] J. F. Kurose, K.W. Ross, *Computer Networking A Top-Down Approach Featuring the Internet*, Addison Wesley, 2000.

[16] C.S. Perkins, J. Crocroft, " Notes on the use of RTP for shared workspace applications" , ACM Computer Communications Review, volume 30, número 2, Abr. 2000.

[17] P. Pan, H. Schulzrinne, " Yessir: A Simple Reservation Mechanism for the Internet" , http://www.cs.columbia.edu/~hgs/netlib, 2000.

[18] R. Hunt," Evoluing Technologies for new Internet Applications ", IEEE Internet computing, 1999.

[19] L.F. de Moraes, S.R. Teixeira, J.H. Teixeira Jr., " Gerência Baseada na Web", relatório técnico, http://www.ravel.ufrj.br, 2000.

[20] " A white Paper on Web-Based Management " , CyberManage Inc., http://www.cybermanage.com , 2000.

[21] J. Lenox, J. Rosemberg and H. Schulzrinne, " Common Gatway Interface for SIP" , Draft Internet Engineering Task Force, 2000.

[22] W.V. Leekwijck, D. Brouns, " Siplets: Java-Based Service programming for IP Telephony", http//www.alcatel.com, 2000.

[23] SUN MICROSYSTEMS, *Java Sound API Programmer's Guide*,http://java.sun.com/j2se/1.3/docs/guide/sound/prog_guide/title.fm.html [online], 2000.

[24] P. Gevros, A. Kapetanaki, L. Liu, E. Waizel and B. Wong , "Java Audio Tool " , Master's Thesis , DCNDS group, University College London , 1996.

[25] J.Jaworski , *Java 2 Unleashed*, Sams, 1999.

[26] A. S. Tanenbaum , *Computer Networks*, 3ª ed., Prentice Hall, 1995.

[27] R. E. Smith, *Internet Cryptography*, Addison Wesley, 1997.

[28] S. V. de Miranda, C.M. Machado e J. M. S. Nogueira ", Um Sistema de Suporte ao Gerenciamento do Nível de Serviço-SGSWeb", Anais do 18^{o} Simpósio Brasileiro de Redes de Computadores, Belo Horizonte, Brasil, 2000.