Framework de Serviços em Ambiente Heterogêneo de Telefonia IP

Cesar A. C. Marcondes[1], **Paulo H. de Aguiar Rodrigues**[2]

Departamento de Ciência da Computação/IM e NCE/UFRJ
Caixa Postal 2324, CEP 20001-970
Rio de Janeiro – RJ – Brasil

Resumo

Atualmente, estão sendo desenvolvidos vários frameworks de serviços de telefonia, principalmente as propostas Parlay/JAIN e DFC, que, com o uso de componentes de software, tornam os serviços transparentes às camadas inferiores da rede. Estas soluções exigem a implantação de clientes complexos, suportando todas as facilidades desejadas. Apresentamos uma abordagem diferente que implementa a programação de serviços como extensão associada aos servidores de telefonia, com uso de uma linguagem de programação de chamadas bastante simples e genérica, chamada CPL, apropriada para permitir que serviços de telefonia possam interoperar entre arquiteturas heterogêneas. Para suportar este ambiente novo de programação de chamadas, desenvolvemos um editor de *script* CPL com uma interface gráfica, e implementamos um *gatekeeper* H.323 com extensão CPL, o que possibilita suportar serviços de telefonia em ambiente H.323, sem uso de serviços suplementares, que requerem sofisticação na sinalização e complexidade na construção de gateways para a integração de arquiteturas heterogêneas.

Palavras-chave: protocolos, serviços e aplicações Internet; Telefonia IP; Linguagem CPL; Arquitetura Heterogênea de Telefonia IP; H.323, SIP.

Abstract

Presently, a couple of IP telephony service frameworks are being developed, providing abstract service creation transparent to undergoing network layers. Two of these frameworks are Parlay/Jain and DFC. Instead of integrating software components to achieve service creation as suggested by these two framework proposals, we extend the H.323 gatekeeper to support the script processing of a call oriented programming language, called CPL, quite simple and generic. CPL was devised as a means to allow service interoperation among heterogeneous IP telephony architectures. To support this new framework of programmable services, we developed a graphic interface CPL editor, and implemented an H.323 gatekeeper with CPL extensions. In this way, we are able to support H.323 telephony services without using supplementary services. Supplementary services require a more complex signaling and make VOIP gateway implementation very difficult.

Keywords: Internet protocols, services and applications; IP Telephony Services; CPL Language; Heterogeneous IP Telephony Architecture, H.323, SIP.

[1] Mestrando do Programa de Pós-Graduação em Informática do IM/NCE, e-mail: cesar@posgrad.nce.ufrj.br.
[2] Professor do Departamento de Ciência da Computação/IM e analista do Núcleo de Computação Eletrônica/UFRJ, e-mail: aguiar@nce.ufrj.br.

1. INTRODUÇÃO

Hoje, no contexto internacional, as telecomunicações vêm experimentando uma enorme mudança estrutural, onde as vantagens da tecnologia de comutação de pacotes têm despertado grande interesse tanto nas comunidades da Internet como no ambiente corporativo das operadoras telefônicas convencionais. Dentre as razões para este movimento de convergência, podemos citar a possibilidade de uma melhor utilização dos meios físicos devido à multiplexação estatística e à viabilidade do transporte integrado de dados, vídeo e voz. Outra razão recai sobre os benefícios que a indústria de telecomunicações vêm obtendo da sofisticação de seus *softwares*. Estão sendo construídos diversos frameworks de serviços com o propósito de definir uma rede virtual de telecomunicações que abranja a comutação de pacotes, as tecnologias de rede sem fio, e até a própria rede convencional de telefonia (PSTN) [1]. Esta abstração da rede de telefonia permite criar o que é chamado de ambiente heterogêneo de telefonia, e favorece o surgimento de novos serviços abstratos, que transcendem as arquiteturas.

Além das empresas de telecomunicações, também na Internet temos acompanhado o desenvolvimento de trabalhos de suporte a este ambiente heterogêneo. A universalidade de interoperação entre várias arquiteturas de telefonia, como a hoje existente entre a telefonia fixa e a celular, é uma necessidade. Na telefonia IP (VOIP), além da interoperação transparente com a telefonia pública tradicional (fixa e celular), teremos que interoperar plataformas VOIP distintas, baseadas em SIP [2], H.323 [3], ou em outros protocolos como MGCP e SS7.

Na área de programação de serviços e roteamento de chamadas para ambiente heterogêneo, as tendências mais importantes estão contidas no trabalho do Parlay Group [4] e na arquitetura DFC [5]. O Parlay Group definiu, recentemente, a Parlay API, uma interface de programação aberta, que constitui um *framework* de programação de serviços onde tanto parceiros (empresas fora do domínio da operadora da rede) quanto a própria operadora de rede podem construir novas aplicações (como serviços de *Call Waiting*), sem precisar conhecer profundamente a tecnologia usada pelos elementos da rede. Assim sendo os elementos da rede poderiam estar usando SIP, H.323, MGCP, ou ISUP, e estas tecnologias estariam sendo controladas transparentemente por esta implementação Parlay. A Sun participante do Parlay Group, desenvolveu sua versão da Parlay API em Java, chamada de JAIN.

A outra proposta para a próxima geração de *frameworks* de programação de serviço de telecomunicações está sendo desenvolvida pela AT&T Labs, e usa uma forma modular de programação chamada *Distributed Feature Composition* (DFC) [5]. Na DFC, os serviços abstratos sob o ambiente heterogêneo são compostos por componentes independentes, chamados de *feature boxes*, que permitem rotear a chamada e executar funcionalidades de diversos protocolos. Tanto Parlay como DFC exigem a implantação de clientes complexos.

Apesar da existência destes *frameworks* sofisticados apresentados acima, sempre é possível prover serviços avançados de telefonia (como *Call Forward*) pela cooperação simples entre clientes e servidores de uma certa arquitetura. A chave da programação de serviços para a telefonia IP está no entendimento profundo dos protocolos de estabelecimento e finalização de chamadas da telefonia IP, como eles localizam usuários, negociam capacidades e invocam serviços suplementares. Desenvolvemos, recentemente, uma implementação de *gateway* de telefonia IP baseado em SIP/H.323 [6], utilizando uma

interface simples chamada BSM (*Basic Signaling Messages*) para o suporte à sinalização no ambiente heterogêneo. A filosofia deste *gateway* é traduzir somente as mensagens básicas de sinalização, perdendo um pouco em abrangência por não tratar as mensagens de serviço avançadas do SIP e H.323 (como os serviços suplementares). Entretanto, a premissa principal da proposta era suportar ambientes com uso de clientes SIP/H.323 básicos e aumentar a capacidade e desempenho do *gateway*, restringindo o uso a mensagens de sinalização simples.

A programação de serviço em um ambiente heterogêneo usando apenas *gateway* SIP/H.323 de telefonia IP foi abordada por [7] (sem o uso dos *frameworks* apresentados acima), onde os serviços suplementares H.323 foram convertidos em serviços avançados SIP. Entretanto, não foi alcançada nesta abordagem toda a generalidade e flexibilidade que o ambiente heterogêneo deveria ter. Outra proposta sendo defendida é a que os servidores de rede SIP e H.323 possam abrigar os serviços avançados (como o roteamento de chamadas, serviço de *follow me*, etc) e reprogramar, dessa forma, o próprio comportamento do servidor. Isso poderia ser feito de duas maneiras: uma irrestrita, com enfoque em reprogramação do servidor, usando o mesmo conceito de CGI da Web para o ambiente de telefonia; outra, bem mais restritiva em termos de programação, sendo chamada de CPL (*Call Processing Language*) [8], e dirigida a programação de serviços pelo próprio usuário final.

A CPL ou linguagem de programação de chamada é um documento XML que contém um grafo de decisões a serem tomadas, de acordo com o estado de uma ligação. Assim, um módulo estendido CPL contendo uma lógica de serviço poderia ser adicionado a um servidor de telefonia IP, operando independentemente da sinalização, e, baseado nesta lógica, seria ditado como as requisições de chamada deveriam ser redirecionadas. Não só o redirecionamento deveria ser contemplado pelo módulo CPL, como quaisquer serviços como *Call Waiting*. Atualmente, a definição da CPL limita seu uso apenas ao roteamento passivo de chamadas. Para suportar todos os serviços suplementares é necessário criar novos *tags* e estabelecer uma operação interativa. Estas extensões para a CPL estão sendo pesquisadas e serão objeto de trabalho futuro.

Fig. 1 mostra os três tipos básicos de *framework* de telecomunicações apresentados até aqui. O diagrama (A) representa um servidor que contém uma lógica de roteamento inserida (usando CPL) para criar o serviço de *Call Forward On Busy* para SIP. O diagrama (B) mostra este mesmo serviço suplementar desenvolvido somente como a comunicação entre clientes, e em (C) temos a arquitetura DFC em funcionamento, trabalhando com seus componentes de modo a combiná-los e criar um serviço de *Call Waiting*.

Neste artigo implementamos a interoperação da programação de serviços entre várias arquiteturas VOIP pelo compartilhamento de um esquema básico de CPL, e apresentamos um *framework* de desenvolvimento de serviços para rede heterogênea de telefonia IP. Esta abordagem difere das propostas Parlay/JAIN e DFC, pois não precisamos cooperar vários componentes de software para atingir o resultado de serviço avançado esperado. Cada serviço é criado como um script CPL em sua própria rede e roda no próprio servidor de telefonia. Não existindo, portanto, necessidade da criação de uma camada adicional de lógica (*multi-tier*), deixando os *scripts* interagirem com a sinalização básica, enquanto ela puder ser traduzida por nosso *gateway*. O script CPL escrito em XML é bem genérico e mantém o significado semântico para qualquer arquitetura SIP ou H.323, permitindo que serviços que ultrapassem os limites de uma única arquitetura sejam viáveis.

O artigo está organizado em cinco seções. Na Seção 2, descrevemos os serviços avançados de telefonia baseado em H.323, detalhando um pouco mais a linguagem CPL e mapeando suas estruturas de *tags* em cabeçalhos de mensagens H.323. Na Seção 3, mostramos uma ferramenta desenvolvida para dar suporte à programação de *scripts* CPL, gerando, a partir de uma representação gráfica, um *script* correspondente. Na Seção 4, discutimos as opções de implementação de um servidor *gatekeeper* com extensões para serviços de redirecionamento CPL, e mostramos exemplo do seu funcionamento. As conclusões são apresentadas na Seção 5.

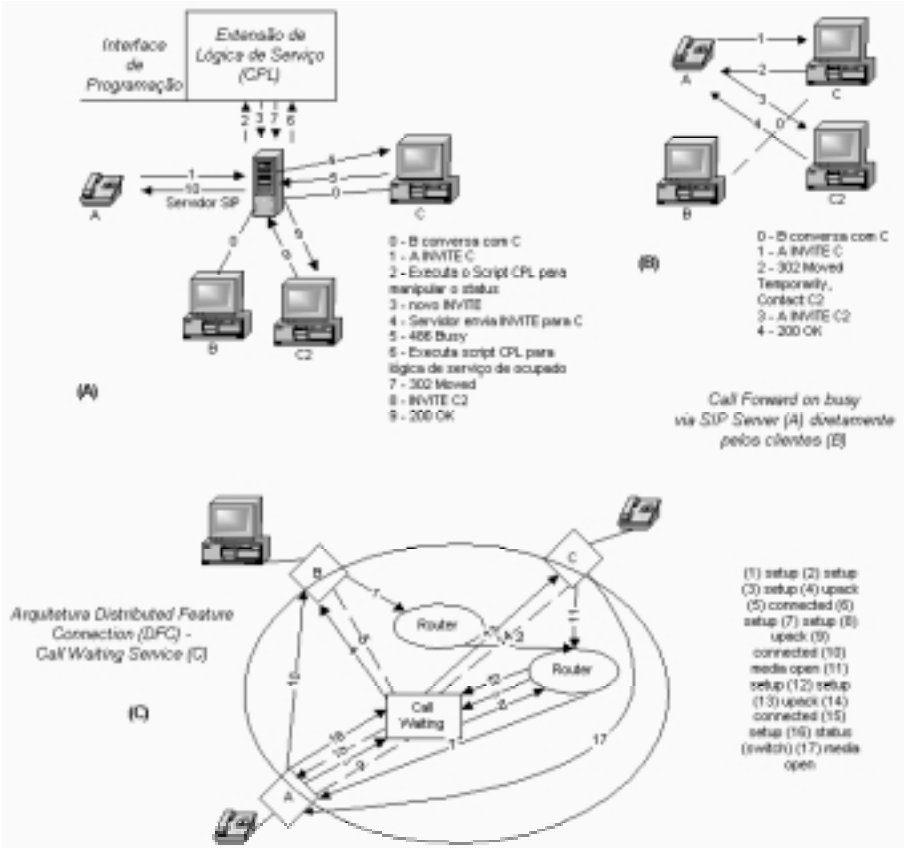

Fig. 1 – Arquiteturas de Programação de Serviço

2. SERVIÇOS AVANÇADOS H.323 DE TELEFONIA IP

Existem duas abordagens para a programação de serviços telefônicos num ambiente formado por terminais, *gateways* e servidores de registro H.323: a primeira é baseada no uso de serviços suplementares; a segunda é baseada na reprogramação do servidor H.323 (Anexo O do H.323, ainda em estudo pela ITU-T).

Os serviços suplementares descritos nos padrões H.450.X (ITU-T, 1997) são executados através de uma seqüência bem particular de mensagens H.225 [9] do tipo *Facility*, trocadas entre dois terminais. Esta comunicação é caracterizada por um terminal solicitando que seja executado algum procedimento em outro terminal. Estas mensagens *Facility* podem transportar informações sobre qual roteamento que a sinalização deve seguir ou a ativação de algum programa externo.

Cada mensagem H.225 *Facility* que ativa um serviço suplementar deve transportar as chamadas APDUs (*Application PDUs*), onde ficam os parâmetros de redirecionamento. Estas mensagens devem ser codificadas, decodificadas e interpretadas pela máquina remota. Para cada tipo de APDU, a ITU-T tem formalizado o processo de criação, configuração e execução deste serviço suplementar. Esta formalização vai desde a especificação da gramática ASN.1 da mensagem contendo o serviço até o procedimento exato de como isto deve ser processado, descrito por diagramas formais da ITU-T, chamados SDLs (*Specification and Description Language Diagrams*).

Nos últimos anos, o ITU-T vem padronizando muito destes serviços suplementares, contando atualmente com onze novos documentos H.450 (do H.450.1 até H.450.12), atendendo a uma demanda crescente da comunidade de telefonia IP pela programação de serviços. Também foi padronizada uma especificação para a criação genérica de serviços suplementares [10], de modo que extensões livres pudessem ser criadas.

Entre os exemplos mais comuns de serviços suplementares, temos o H.450.2 *Call Transfer Supplementary Service* [11]. Este serviço permite que um usuário A (iniciador da transferência) transforme uma chamada existente com o usuário B (sua chamada primária) em uma nova chamada entre o usuário B e um outro usuário C (usuário transferido), selecionado pelo usuário A.

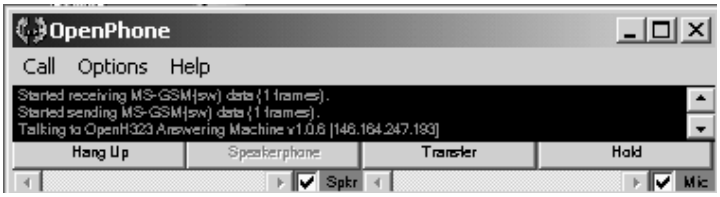

Fig. 2 – Serviços Suplementares H.323 (Call Transfer e Call Hold)

O terminal *openphone* (Fig. 2), diferentemente do *Netmeeting* da Microsoft, implementa alguns destes serviços suplementares. Logo que a ligação de telefonia IP for completada ficam habilitados dois botões (Fig. 2) para acionar estes serviços: os botões de *Transfer* e o de *Hold* (à direita). O botão *Call Transfer* aciona o serviço descrito acima. Nesta implementação, também pode ser configurado como será o transporte da APDU, se via *Facility* (visto anteriormente) ou canal lógico H.245, sendo esta outra abordagem para iniciar o serviço.

O SIP, por sua vez, também tem a sua especificação genérica para a criação de serviços avançados [12], mas o mapeamento entre os serviços suplementares H.323 e estes serviços avançados SIP não é uma boa prática para convergência num ambiente heterogêneo de telefonia IP, pelas diferenças de concepção entre eles.

Além desta abordagem de serviços suplementares, temos uma outra forma de prover programação de serviços para a telefonia IP. Esta outra forma é baseada na reprogramação do servidor (Anexo O do H.323), o que exige do administrador do sistema um conhecimento abrangente do funcionamento do H.323 e da interface de programação (API) associada ao servidor gatekeeper (GK). Um exemplo deste tipo de reprogramação do servidor seria a criação de um serviço de cobrança (*billing*), onde, sempre que a sinalização Q.931 fosse utilizada, seriam contabilizados os tempos de início e fim da chamada. Esse tipo de

reprogramação no servidor é bastante específico, e totalmente dependente da API do GK sendo utilizada.

Rosenberg *et al* propuseram em [13] dois outros modelos de reprogramação de servidor: um baseado no uso da programação CGI (para a telefonia IP) em conjunto com o servidor *gatekeeper*; e outro usando CPL (*Call Processing Language*) [8]. Estes dois modelos liberam o administrador da dependência da API do GK.

A CPL (*Call Processing Language*) [8] é descrita como uma forma de controlar serviços de telefonia IP e não está teoricamente associada a nenhuma arquitetura de sinalização, apesar da recomendação CPL prover muitos exemplos usando o SIP. Isso a torna mais adequada para o ambiente heterogêneo. Existem algumas implementações de servidores SIP com extensões CPL, como o servidor SIP da Universidade de Columbia [14], mas para o ambiente H.323 está área ainda está em estudo, com poucos protótipos.

Os serviços criados pela CPL têm uma variedade muito maior do que os serviços suplementares criados no H.323. Podemos com a CPL mapear diretamente alguns serviços acionados na fase de estabelecimento de chamada, como H.450.3, H.450.6, e H.450.9. Os outros serviços suplementares H.323, que ocorrem durante uma chamada já estabelecida, dependem de uma extensão para que o CPL atue de forma interativa.

2.1 Definição da Linguagem de Programação de Chamadas (CPL)

Baseado nos requisitos rígidos da criação de serviços CPL (pois eles vão ser instalados por quaisquer usuários e o funcionamento tem que ser garantido), a linguagem CPL foi projetada como um grafo de decisões. Estas decisões e ações são dispostas em grafo orientado acíclico, que define como será o tratamento para o serviço. Um exemplo deste fluxo de decisões seria o seguinte: imagine que precise ser criado um redirecionamento por horário para um cliente, onde, no período do expediente, todas as chamadas IP que estiverem sendo direcionadas para este cliente sejam redirecionadas para a empresa onde ele está trabalhando, mas, à noite, que estas chamadas IP sejam redirecionadas para a casa do mesmo. Isso é possível, com a ajuda da árvore de decisão que o CPL propõe.

O uso de documentos XML é perfeito para a representação estruturada de dados, particularmente de estruturas de árvores, que é a forma necessária para representar grafos orientados acíclicos, que definem este tipo de programação de serviço. Por isso, o CPL foi definido como um conjunto de *tags* em XML. Uma vantagem importante do uso de XML para representar esta programação de usuário é que a semântica e a sintaxe de um *script* podem ser verificadas de forma automática com o uso do documento de validação XML contido na especificação CPL, chamado *Document Type Definition* (*DTDs*). Isso é crucial pelo fato de que todo serviço CPL deveria estar perfeito (bem formado), para poder garantir ao cliente que este irá executar com sucesso.

Na CPL temos quatro primitivas (*tags*) básicas, que são: os *nós switch*, que representam as decisões que um *script* deveria tomar; os *nós location*, que indicam onde os usuários poderão ser encontrados, tanto diretamente ou por referência; as *ações de sinalização* que são o núcleo da linguagem de programação de serviços, pois elas atuam diretamente sobre o comportamento do protocolo de sinalização; e, finalmente, as *ações não dependentes da*

sinalização, que permitem que ações fora do contexto da chamada possam ser tomadas, como gravar alguma informação relevante em um arquivo de *log*, útil para tarifação.

O exemplo, em Fig. 3, foi extraído da recomendação CPL [8] e mostra o uso dos *tags* de roteamento por uma regra de tempo. O *tag* <*incoming*> representa uma chamada direcionada para o autor do script. A *tag* <*time-switch*> é uma *tag* dependente do sistema (mas não da sinalização) e representa uma decisão a ser tomada com relação à hora local. Ela está configurada com os parâmetros de *time-zone* para usar o horário de New York. Cada *tag* do tipo *switch* tem sempre nós filhos que representam as possíveis opções de escolha; no caso, temos duas opções: <*time*> e <*otherwise*>.

A opção <*time*> descreve o intervalo de tempo em que uma chamada entrante será processada segundo aquela sub–árvore (<*lookup*>), enquanto a opção <*otherwise*> roteia as chamadas que não chegaram no intervalo especificado acima. Seria possível usar vários intervalos de *tag* <*time*> para fazer um roteamento complexo ao longo das horas do dia. O processamento do nó <*lookup*> é feito pelo servidor de telefonia, que pesquisa o registro do usuário destinatário, autor do script. Caso esta pesquisa obtenha sucesso (<*success*>), então a chamada será roteada (<*proxy*>) para o usuário destinatário. Por outro lado, quando a requisição chegar fora do período de tempo estipulado pelo <*time*>, o processamento recairá sobre o <*otherwise*> que, a partir de uma URL descrita no *tag* <*location*>, redirecionará a chamada (<*proxy*>) para o usuário contido na *tag* <*location*>, ou seja, mandando a requisição para o *voicemail* (*jones@voicemail*...).

```
<cpl>
   <incoming>
        <time-switch tzid="America/New_York" tzurl="http://zones.com/tz/America/New_York">
            <time dtstart="20000703T090000" freq="weekly" byday="MO,TU,WE,TH,FR">
                <lookup source="registration">
                    <success>
                        <proxy />
                    </success>
                </lookup>
            </time>
            <otherwise>
                <location url="sip:jones@voicemail.example.com">
                    <proxy />
                </location>
            </otherwise>
        </time-switch>
   </incoming>
</cpl>
```

Fig. 3 – Script CPL de Roteamento pela Hora do Dia

2.1.1 Mapeamento CPL entre SIP e H.323

Apesar da recomendação CPL [8], descrever a linguagem de programação de chamada como independente de protocolo de telefonia IP, neste documento quase todos os exemplos descritos usam o protocolo SIP. A parte que explica como implementar nós CPL para H.323 fica reservada ao Anexo B da Recomendação CPL.

A ITU-T também iniciou um estudo sobre a viabilidade da implementação do CPL para sua pilha de protocolo através do documento Anexo O da Recomendação H.323 [15],

entretanto ambos os documentos apresentam certas divergências no tratamento das *tags*. O próprio documento da recomendação CPL considera o uso sugerido para H.323 como apenas normativo, e, portanto, sujeito a quaisquer normas a serem criadas pela ITU.

Em Quadro 1, descrevemos algumas destas divergências, que enfocam como os nós CPL atuam e sua relação com os campos de cabeçalho dos protocolos, usando tanto a abordagem do Apêndice B do CPL, quanto do Anexo O do H.323.

Nó CPL	Significado	Apêndice B do CPL	Anexo O do H.323
Incoming e Outgoing	Indica se o script CPL será executado quando a chamada foi encaminhada para o dono do script ou partindo do dono do script.	Segundo o Apêndice B do CPL não há diferença entre estes nós nos ambientes SIP e H.323.	Quando o nó for incoming, e nó address–switch estiver sendo utilizado com parâmetro *destination*, usar o H.323 *destCallSignalAddress.*
Address-Switch: parâmetro *field*	Decisões baseadas em endereços presentes na requisição original. Parâmetro *field* (*origin*, *destination*)	No H.323, o parâmetro *origin* corresponde ao H323 *SourceAddress* contendo vários *alias addresses*. Deve-se adotar uma política de prioridade, caso múltiplos *aliases* estiverem presentes.	Embora vários Aliases possam existir em uma mensagem H.323, a opção obrigatória da escolha do tipo "*alias"* dentro de um número de possibilidades limita os sistemas H.323
Address-Switch: parâmetro *subfield*	Decisões baseadas em endereços presentes na requisição original. Parâmetro *subfield* (*alias-type*)	O mapeamento dos endereços H.323 em *sub-fields* depende do alias-type. Os possíveis valores de alias-type seriam: *dialedDigits*, *h323-id*, *uri-id*, *transportID*, *mobileUIM*, *emailID*, *partyNumber* e *Q931IE*.	Segundo o Anexo O, a seção "*Address-Switch Mapping for H.323*" da Especificação CPL deveria ser usada apenas como "ilustração" de como os campos CPL poderiam ser mapeados em campos H.323. Mas o Anexo O não define o mapeamento em nenhum local.
String-Switch	Decisões baseadas em quaisquer informações textuais presentes na requisição da chamada. Parâmetro *field*: (*subject*, *organization*, *user-agent*, *language* e *display*)	O nó *string-switch* contendo parâmetro *language* pode ser mapeado no campo *H.323 UUIE language*, traduzindo-se o formato especificado para aquele campo. O campo CPL *display* corresponde ao campo display Q.931 de mesmo nome.	Nada é definido com relação a isso.
Location	Especifica localização a ser usada para o redirecionamento. Parâmetro *url*	O *Location* do CPL em H.323 deve ser mapeado somente para o campo H.323 *url-id*. Caso o CPL esteja usando extensões de alias-type (<*address switch*>), o mapeamento ocorre para outros tipos de endereços H.323.	Na montagem da mensagem de resposta CPL no ambiente H.323, certas informações podem não ser disponibilizadas como ocorre no SIP. P. ex., a causa da rejeição de uma chamada pode estar incluída na URL SIP do nó *location*, o que não seria possível em H.323.
Lookup	Especificam meios externos para buscar e adicionar *locations* para o redirecionamento.	O nó de *lookup* faz uma busca do autor do script no servidor GK da Zona Administrativa, usando mensagens RAS. Esta busca permite adicionar dinamicamente um novo *location* para redireciona-mento.	O nó *lookup* é um conceito independente de sinalização. Entretanto, no caso do SIP, o *lookup* pode ser feito com outros parâmetros de filtragem (por exemplo, por capacidades de mídia ou língua), algo não suportado pelo H.323.

Quadro 1 – Diferenças entre as abordagens CPL do Apêndice B do CPL e Anexo O do H.323

Os outros nós CPL, como o *time-switch*, *priority-switch*, os nós específicos da sinalização como *redirect*, *proxy*, *reject* e os nós *mail* e *log* não apresentam divergências sobre o seu significado para o ambiente H.323. Esse estudo preliminar foi necessário para tomarmos decisões de quais campos e mensagens tratar na execução de um script CPL para o ambiente H.323.

No caso da divergência da interpretação, optamos por usar o Apêndice B da CPL, por ter maior detalhamento de quais operações e campos devem ser checados. O Anexo O do H.323 é considerado ainda muito incipiente na área de CPL.

3. IMPLEMENTAÇÃO DE EDITOR CPL

Para construir os scripts CPL-XML, desenvolvemos um editor com interface gráfica, argumento defendido como um requisito básico do *framework* CPL [16], para que os usuários com pouca experiência em criar e editar scripts CPL (e conseqüentemente com código XML) e possam montar esquemas de como desejam que o fluxo da chamada seja redirecionado de uma maneira gráfica, extremamente fácil e prática. Dessa forma, estamos ajudando no desenvolvimento de novos serviços de programação com CPL e também contribuindo com a sua popularização.

Nossa implementação deste editor foi desenvolvida em Java usando a tecnologia de *applets* com *Java Plugin* [17]. Trata-se de uma ferramenta com a filosofia "**What you see is what you get**", onde o usuário não interage diretamente com as ***tags*** XML, e sim com botões coloridos que podem ser conectados para formar o grafo do serviço desejado em CPL.

Os conjuntos de objetos desenvolvidos para este editor têm duas visões do sistema: uma visão da parte gráfica e de gerenciamento de painéis; e outra, relacionada com a interpretação, geração do XML correspondente ao grafo e validação do mesmo. Usamos a biblioteca do projeto Apache, Xerces em Java [18], que manipula e trata validação de XML.

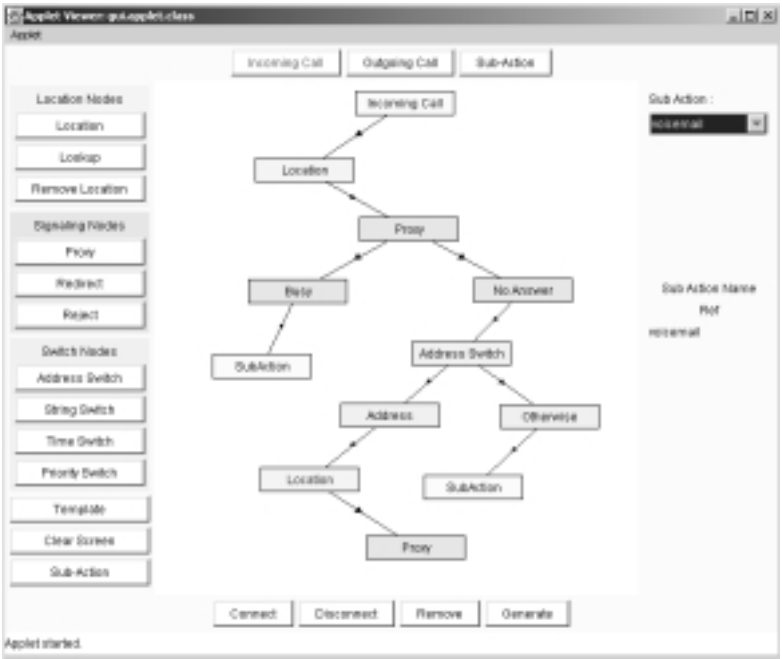

Fig. 4 –Editor Gráfico de CPL e Gerador do XML correspondente ao grafo CPL

Em Fig. 4, temos uma amostra do “script complexo”, um exemplo referenciado na especificação CPL [8]. Os três tipos básicos de nós são agrupados por cores representando os grupos de nós. Cada nó é criado (menu à esquerda) e é gerada sua própria exibição, em formato de botão, no centro da tela, sendo que cada botão possui seu próprio painel de configuração de parâmetros (menu à direita).

O usuário pode conectar os nós quando quiser criando setas direcionadas entre um nó pai e outro filho (menu inferior). Após ter montado o grafo, o usuário deve clicar no botão **Generate**, gerando automaticamente o XML correspondente ao CPL desenhado. Antes de exibir o resultado é feita uma avaliação da consistência do modelo construído em relação ao DTD CPL, avaliando se o script CPL gerado segue rigorosamente a estrutura hierárquica das *tags* CPL/XML.

3.1 Utilizando Serviço Web para gatekeeper

O passo seguinte é submeter o script CPL ao servidor H.323 com esta extensão, para instalar o serviço de redirecionamento de chamada criado por certo usuário. Para este fim, modificamos um ***uploader*** Web com autenticação, desenvolvido em *Perl*, e obtido em [19].

As mudanças neste script *Perl* foram para que o nome do arquivo a ser enviado para o servidor Web fosse modificado com o próprio nome de *login* do usuário, acrescido da extensão .cpl. Na parte do código onde o arquivo é gravado no servidor, fizemos uma alteração para verificar, via segmentação XML, se o documento enviado está em conformidade com o DTD CPL (validando este documento XML), e, então, permitir salvar o arquivo no diretório de trabalho do *gatekeeper*. Uma outra maneira de implementar este instalador CPL baseado na *Web* seria adicionar ao *gatekeeper* um módulo de manipulação HTTP (algo possível no atual estado de desenvolvimento da biblioteca *Pwlib* [20]).

4. REDIRECIONADOR CPL PARA *GATEKEEPER*

Estudamos três possíveis servidores *gatekeeper* para implementar a extensão de programação de serviços via CPL, a saber:

(1) Servidor *opengatekeeper* [21], implementado pela empresa Egoboo, que trabalha tanto nos modos ***Direct Call*** e ***GK Routed***, e com suporte para utilização de *gatekeeper*s vizinhos e uso de prefixos para gateways H.323;
(2) Servidor opengk [20], desenvolvido pela Equivalence ,tendo suporte a serviços HTTP e autenticação H.235, embora apenas trabalhando no modo ***Direct Call***, e sem suporte a utilização de *gatekeeper*s vizinhos ou prefixos;
(3) Servidor *openh323proxy*, baseado em uma versão antiga do *opengatekeeper*, com a capacidade de rotear especialmente o fluxo de mídia entre dois terminais, algo apropriado quando usamos ***firewalls*** ou ***NAT*****.** Infelizmente, o *openh323proxy*, está hoje com suas bibliotecas desatualizadas e sem suporte institucional [22].

Optamos pelo *opengatekeeper*, por suportar o modo ***GK Routed***, necessário para o tratamento do CPL, e também pelo seu prévio usao na implementação do *Signaling Gateway* SIP/H.323 [6] e em testes com o Grupo VOIP da Internet2.

Para entender como um fluxo H.323 é roteado por um script CPL, imaginemos o cenário onde o usuário ***cesar*** deseja realizar o redirecionamento incondicional. O tipo de serviço onde a ligação é transferida incondicionalmente para outro telefone sem saber quem está ligando (Exemplo 13.1 da recomendação CPL [8] Call Redirect Unconditional).

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd">
<cpl>
  <incoming>
    <location url="h323:55212598****@gw.voip.nce.ufrj.br">
     <redirect />
    </location>
  </incoming>
</cpl>
```

Fig. 5 – Redirecionamento Incondicional

Fig. 5 mostra como fica o script CPL-XML. O *tag <?xml version="1.0" ?>* diz respeito à versão do XML usada. Em seguida, *<!DOCTYPE ...>* define qual DTD é usado para validar a estrutura deste *script*. O *script* define o início do processamento no *tag <cpl>*, sendo este o nó raiz da árvore de decisões. Tudo que estiver entre *<cpl>* e *</cpl>* está associado a esta raiz. O nó *<incoming>* significa que a execução deste *script* estará vinculada a chamadas que estiverem sendo direcionadas para este usuário. Assim, se um usuário ***joão*** tentar realizar uma chamada para ***cesar*** via servidor *gatekeeper* (com extensão CPL), onde está depositado este *script*, o script será executado.

O nó *<incoming>* é a próxima raiz da árvore. Abaixo dele temos o nó *<location>* que altera o endereço de destino da chamada H.323 para o valor que estiver neste campo e não mais para o endereço registrado pelo usuário. Por fim, temos o nó *<redirect />* (representação XML equivalente a *<redirect></redirect>*, quando este é um nó folha), indicando ao servidor *gatekeeper* que o usuário ***joão*** deverá ser redirecionado usando o modo *Direct Call* de sinalização para a *url h.323* descrita em *<location>*.

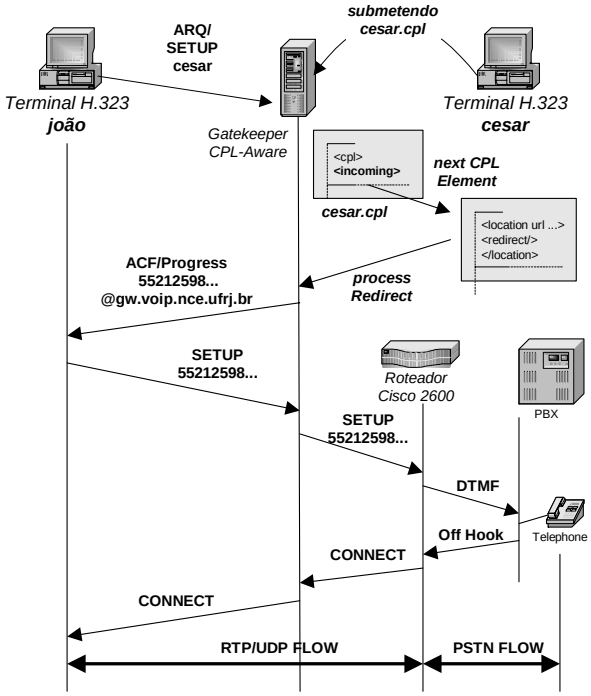

Fig. 6 – Fluxo Básico do Redirecionamento Incondicional

Fig. 6 descreve o esquema de redirecionamento incondicional, onde uma chamada vinda de ***joão*** é automaticamente redirecionada para o Gateway Cisco 2600 (com capacidade

de Gateway H.323/PSTN). Neste fluxo podemos verificar que a mensagem ACF (Admission Confirm) foi modificada segundo a lógica de serviço descrita por ***cesar***.

Esta implementação de extensão CPL para o Gatekeeper foi desenvolvida em C++, com as bibliotecas do projeto OpenH323 [20] e a biblioteca Xerces do Projeto Apache [18] na sua versão para C++. O código executa tanto nos sistemas Windows e Linux.

4.1 Módulo de Manipulação CPL

Desenvolvemos um módulo para realizar a manipulação do CPL e modificar as mensagens de sinalização. Este módulo foi baseado numa implementação em Java [23] chamada *cplParser*. Ele é formado por quatro classes principais que foram desenvolvidas em C++ (Fig. 7). A classe *CPLScript* representa o script como um todo e é onde se realiza a segmentação (*parsing*) do arquivo CPL; a classe *CPLElement* representa a investigação de cada elemento (*tag*) do documento CPL; a classe *CPLOperation* aciona as sinalizações (das classes de sinalização do GK, como a *RASThread* e a *CallThread*) fazendo o redirecionamento das chamadas do GK, e por fim, a classe *CallState* mantém as estruturas de dados relativas ao estado da ligação sendo processada, armazenando os cabeçalhos a serem verificados.

O GK pode, dependendo da fase da sinalização (durante o RAS, ou já na negociação Q.931), acionar o *CPLScript*. Atualmente, estamos processando somente o redirecionamento das mensagens vindas do protocolo RAS.

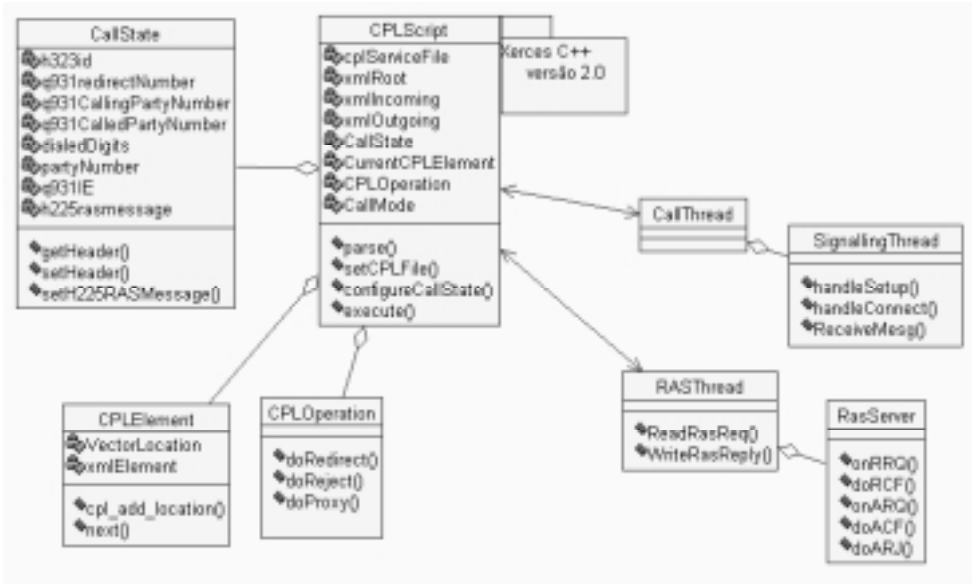

Fig. 7 – Diagrama de Classes do GK com Extensão CPL

Cada vez que uma mensagem ARQ chega, o método OnARQ da classe *RasServer* é processado. Este método verifica se o usuário sendo chamado possui *script* CPL para ser executado. Optamos por primeiro processar os *scripts* que contenham a sub-árvore *Incoming*. Para isso, é feito o acesso à tabela de *Endpoints* (*EndpointTabl*) do GK, procurando pelo Alias do usuário chamado, e, ao encontrá-lo, é verificado se o campo hasCPL está com o valor verdadeiro. (Esta variável estará verdadeira se o usuário estiver registrado, e possuir o arquivo CPL no diretório de trabalho do servidor).

Caso o valor de hasCPL esteja verdadeiro, então é instanciado um objeto para *CPLScript* e configurados os atributos do objeto *CallState*. O passo seguinte é segmentar o arquivo CPL associado aquele usuário e começar a executar a árvore a partir do nó <incoming>. Utilizamos a abordagem DOM na manipulação do XML, em detrimento à abordagem SAX, pois a DOM armazena toda a árvore de objetos na memória, tornando mais rápido o seu acesso.

Cada elemento CPL obtido é então escalonado para execução na classe *CPLElement*, onde são feitas as checagens entre os campos de protocolo e os *tags* CPL para tomar as decisões.

Quando os nós CPL de manipulação da sinalização (como por exemplo, nó CPL *redirect*) aparecerem após os nós de decisão, estes acionam os métodos da classe CPLOperation, como o *doRedirect,* o qual, por sua vez, está manipulando diretamente o GK via classe *RasServer*, mandando como parâmetro uma mensagem *H225_RasMessage* modificada, contendo o endereço de redirecionamento descrito no CPL.

Em nossa implementação, optamos por não usar a recomendação "*alias-type*" para definir qual tipo *alias* será usado. Esta decisão simplifica a lógica do programa, pois estamos trabalhando apenas com o *alias h323-id*.

Este servidor não tem suporte para todos os nós CPL possíveis. No momento, estamos usando um subconjunto do DTD CPL para validar apenas os *scripts* cujos nós estão implementados. Futuramente, pretendemos trabalhar com outras *tags*, como *Time-Switch*, que precisam decidir o roteamento da ligação pela hora do dia. Lennox [24] já desenvolveu um software em Java chamado Cal-Code para auxiliar na determinação de períodos de tempo. Também pretendemos trabalhar com o *tag* <proxy>, que requer uma grande mudança da estrutura do servidor, pois ele é acionado após a chamada ter sido redirecionada.

4.2 Impacto do CPL para o Número Máximo de Chamadas Roteadas via GK

O uso de serviços CPL com o servidor GK H.323 traz sérias conseqüências ao desempenho do servidor. Quanto mais processamento tem que ser feito para concluir a chamada, ou seja, quanto maior o número de nós de decisão do *script*, menor será a escalabilidade do servidor. Por isso, o uso desta programação de serviços para o ambiente H.323 deve ser mantido em servidores com abrangência departamental. Para o núcleo da rede convém utilizar gatekeeper com menor inteligência em termos de serviços, mas maior capacidade de roteamento das chamadas (o uso do modo Call Direct, ajuda a aumentar esta capacidade, pois não é necessário manter o estado de cada ligação nem rotear os canais de sinalização H.225). Este tipo de GK seria mais adequado para fazer o *peering* dos gatekeepers departamentais.

5. CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento de serviços de programação de chamadas tem tido uma demanda grande por parte da comunidade internacional. Muitas arquiteturas têm sido propostas e implementadas para o ambiente heterogêneo de telefonia. Sabemos que a programação de serviços pode ser implementada tanto ao nível de comunicação entre clientes, como através de

uma lógica especial no servidor ou com o uso de componentes de *software* cooperando mutuamente.

Com base em nossa experiência na implementação de *gateway* H.323/SIP [6] e em trabalhos relacionados [1,5], podemos afirmar que a conversão entre sinalizações avançadas para a operacionalização de programação de serviços em ambiente VOIP heterogêneo baseado em interação de componentes de software, como sugerido nas arquiteturas Parlay/Jain e DFC, é muito complexa.

Este trabalho apresenta uma abordagem diferente destas propostas, fazendo uso de uma linguagem de programação de chamadas bastante simples e genérica, chamada CPL, apropriada para permitir que serviços de telefonia possam interoperar entre arquiteturas heterogêneas. Para suportar este ambiente novo de programação de chamadas, desenvolvemos um editor de *script* CPL com uma interface gráfica, e implementamos um *gatekeeper* H.323 com extensão CPL. Esta solução possibilita suportar serviços de telefonia em ambiente H.323 sem uso de serviços suplementares, que requerem sofisticação na sinalização e complexidade na integração entre arquiteturas heterogêneas.

Sabemos como os serviços suplementares são importantes, e que, para implementá-los na nossa arquitetura heterogênea de telefonia IP, precisamos estender o paradigma da programação CPL, usando o que chamamos de CPL interativo. CPL interativo é a execução de um *script* CPL durante uma chamada telefônica IP. Este *script*, contendo a mesma lógica do serviço suplementar, deve ter o mesmo significado tanto nos ambiente H.323 quanto SIP, ou outros. Para realizar estes objetivos, estamos desenvolvendo um estudo sobre a sêmantica SDL (Specification and Description Language Diagrams) como candidata a ser a base para nossa extensão de CPL interativo, pois ela descreve formalmente no protocolo H.323 qual é o grafo de execução de certo serviço suplementar. Outro ponto fundamental para a base do CPL interativo é que ele deve ser implementado dentro dos servidores *gatekeeper*s (usando o modo GK Routed), não provocando qualquer ônus ou reimplementação por parte dos clientes H.323.

Com esta arquitetura estendida, além dos serviços suplementares H.323 mapeados em CPL interativo, poderemos adaptar a comunicação entre clientes que não possuam a implementação de serviços suplementares (como o caso do Netmeeting) e clientes mais sofisticados.

Uma questão crucial nesta arquitetura será decidir como ativar o CPL interativo, já que, no caso do H.323, não é possível usar *mime-types* para transportar um payload XML com um serviço. Como usamos uma interface Web para instalar o serviço CPL convencional, também em nossa proposta de extensão de CPL interativo, pretendemos usar páginas Web para ativar os serviços, sendo que estas páginas terão que ser abertas durante a comunicação de telefonia IP.

6. BIBLIOGRAFIA

[1] G. W. Bond, et al. **ECLIPSE Feature Logic Analysis**, Proceedings of IP Telephony Workshop, New York, fev. 2001.

[2] H. Schulzrinne, J. Rosenberg, et al. **IETF RFC 2543 – SIP: Session Initiation Protocol,** outubro 2001.

[3] ITU-T Recommendation H.323. **Packet-Based Multimedia Communications Systems**, setembro, 1999.
[4] S. Beddus, G. Bruce e S. Davis. **Opening Up Networks with JAIN Parlay**. IEEE Communications Magazine, abril 2000.
[5] M. Jackson and P. Zave. **Distributed Feature Composition: A Virtual Architecture for Telecommunications Services**. IEEE Transaction on Software Engineering, vol. 24, n. 10, pp. 831-847, outubro 1998.
[6] B. Ribeiro, P. Rodrigues, C. Marcondes. **Implementação de Gateway de Sinalização entre Protocolos de Telefonia IP SIP/H.323**. SBRC 2001, Florianópolis, maio 2001.
[7] R. Ackermann, et al. **An Open Source H.323-SIP Gateway as Basis for Supplementary Service Interworking**. 2º - IP Telephony Workshop, New York, 2001.
[8] J. Lennox, H. Schulzrinne. **CPL: A Language for User Control of Internet Telephony Services**. draft-ietf-iptel-cpl-04.txt, maio 2001. Trabalho em Andamento.
[9] ITU-T Recommendation H.225.0. **Media Stream Packetization And Synchronization On Non-Guaranteed Quality Of Service LANs**. Setor de Telecomunicações, Genebra, Suiça. nov. 1996.
[10] **ITU-T Recommendation H.450.1.** Generic functional protocol for the support of supplementary services in H.323**, setembro 1997**.
[11] ITU-T Recommendation H.450.2. **Call Transfer Supplementary Service For H.323**. Setor de Telecomunicações, Genebra, Suiça. set. 1997.
[12] R. Sparks. **IETF Draft: SIP Call Control**. http://www.cs.columbia.edu/~hgs/sip/ drafts/ draft-ietf-sip-cc-transfer-04.txt. ago. 2001.
[13] J. Rosenberg; J. Lennox; H. Schulzrinne. **Programming Internet Telephony Services**. Revista IEEE Network, vol. 13, pp. 42–49, maio/junho 1999.
[14] Columbia University. **SIPD – Servidor Proxy e Redirect SIP com Extensões CPL**. http://www.cs.columbia.edu/~lennox/sipd. dez 1999.
[15] ITU-T Recommendation H.323 – Anexo O. **Use of Complementary Internet Protocols with H.323 Systems**, março 2001.
[16] J. Lennox, H. Schulzrinne. **IETF RFC 2824 - Call Processing Language Framework and Requirements**. maio 2000.
[17] J. H. Melman, V. Brasileiro. **Editor Gráfico de CPL**. Relatório de Iniciação Científica IM/NCE/UFRJ, 2002.
[18] **Xerces para Java e C++ - Apache Project** (http://xml.apache.org/).
[19] M. Muquit. **Código Fonte do Programa em Perl para Upload de Arquivos**. http://www.muquit.com/muquit/software/upload_pl/upload_pl.html. Acesso set. 2000.
[20] Equivalence Inc. **Código Fonte do Projeto OpenH323 e OpenGk.** (http://www.openh323.org) Acesso em dez. 2000.
[21] ***Opengatekeeper* Source Code** –(http://opengatekeeper.sourceforge.net)
[22] **OpenH323 Proxy *Gatekeeper*** – (http://openh323proxy.sourceforge.net)
[23] Viena University of Technology and Wien University. **Interpretador CPL em Java: cplParser**. http://www.ikn.tuwien.ac.at/ftw-a1/. Acesso dez. 2001.
[24] J. Lennox. **Cal-Code: Java code for CPL time-switches**. Disponível em: http://www.cs.columbia.edu/~lennox/Cal-Code/. Acesso dez. 2001.