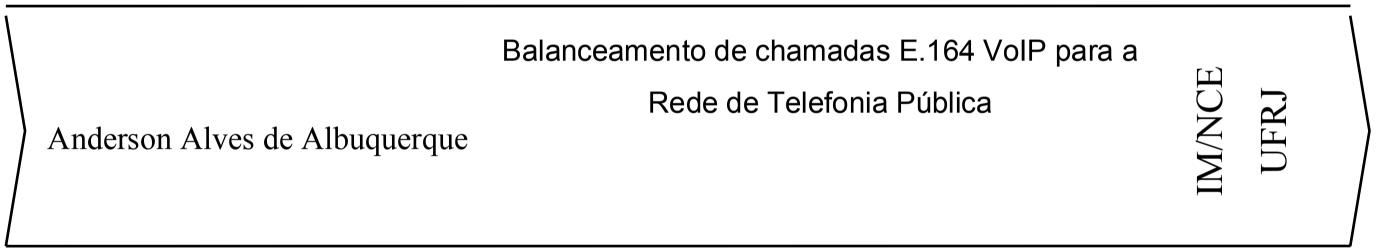
ANDERSON ALVES DE ALBUQUERQUE

Balanceamento de chamadas E.164 VoIP para a Rede de Telefonia Pública

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

RIO DE JANEIRO

2007


Balanceamento de chamadas E.164 VoIP para a
Rede de Telefonia Pública
Anderson Alves de Albuquerque
IM/NCE
UFRJ

ANDERSON ALVES DE ALBUQUERQUE

BALANCEAMENTO DE CHAMADAS E.164 VOIP PARA A REDE DE TELEFONIA PÚBLICA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DO INSTITUTO DE MATEMÁTICA / NÚCLEO DE COMPUTAÇÃO ELETRÔNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO – UFRJ, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM INFORMÁTICA.

Orientador: Paulo Henrique de Aguiar Rodrigues

RIO DE JANEIRO

2007

A345 Albuquerque, Anderson Alves de Albuquerque.

Balanceamento de chamadas E.164 VoIP para a Rede de Telefonia Pública /Anderson Alves de Albuquerque. – Rio de Janeiro, 2007.
139f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro, Instituto de Matemática, Núcleo de Computação Eletrônica, 2007.

Orientador: Paulo Henrique de Aguiar Rodrigues

1. Redes de computadores – Teses. 2. VoIP – Teses. 3. Algoritmo – Teses. 4. Protocolos – Teses. I. Paulo Henrique de Aguiar Rodrigues (Orient.). II. Universidade Federal do Rio de Janeiro. Instituto de Matemática. Núcleo de Computação Eletrônica. III. Título.

CDD:

ANDERSON ALVES DE ALBUQUERQUE

BALANCEAMENTO DE CHAMADAS E.164 VOIP PARA A REDE DE TELEFONIA PÚBLICA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DO INSTITUTO DE MATEMÁTICA / NÚCLEO DE COMPUTAÇÃO ELETRÔNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO – UFRJ, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM INFORMÁTICA.

Aprovada por

_____

Prof. Paulo Henrique de Aguiar Rodrigues - Orientador
Ph.D, UCLA (Univ. of California Los Angeles), EUA.

_____

Prof. Antônio Tadeu Azevedo Gomes
D.Sc., PUC-Rio (Pontifícia Universidade Católica), Brasil.

_____

Prof. Célio Vinicius Neves de Albuquerque
Ph.D, UCI (University of Califórnia, Irvine), EUA.

_____

Prof. Luiz Fernando Rust da Costa Carmo
DR., UPS (Université Paul Sabatier/LAAS), França.

AGRADECIMENTOS

Aos companheiros de trabalho no LabVoIP, que através da convivência contribuíram para o meu crescimento no percurso durante o mestrado: Leandro Caetano Gonçalves Lustosa, Fábio David, João Carlos Peixoto de Almeida da Costa, Carlos Mauricio Nunes, Vitor Lourenço Brasileiro, Rodrigo Moreira Silveira, Thiago Maluf Resende, Claudio Miceli de Farias, Mariam dos Passos Afonso da Conceição, André de Abrantes Davim Pereira e Souza, Paulo Victor Barion Heckmaier, Marcelo Arza Lobo da Costa e Douglas Gielo Quinellato.

Não podia esquecer dos amigos de mestrado, que foram verdadeiros companheiros nos momentos de dificuldades; em especial agradeço a Gustavo Alberto Alves (LabNET), Nilson Rocha Vianna (LabNET), Frederico Mendonça Motta, Ana Lúcia Maia da Silva Mostardinha (LabNET), Eugênio Silva (LabIC), Rinaldo Ribeiro e Erica Esteves Cunha.

A RNP (Rede Nacional de Ensino e Pesquisa) que estava presente através de projetos, como fone@RNP e VoIP4ALL, sem os quais não teríamos um ambiente em produção para o desenvolvimento e implantação de novas idéias.

Ao professor Adriano Joaquim de Oliveira Cruz, que teve importante participação quando existiam dúvidas referentes à sua área de conhecimento.

Em especial, ao meu orientador Paulo Henrique Rodrigues de Aguiar, pela confiança nos momentos difíceis e oportunidade de desenvolver este trabalho. Principalmente, por ter ele disponibilizado toda uma estrutura de pessoal e material no LabVoIP nesses anos.

Ao IM/UFRJ e NCE/UFRJ, pela infra-estrutura disponível para o desenvolvimento dos trabalhos, e principalmente aos funcionários, prestadores de serviço e professores. Em especial para aqueles que estiveram mais próximos: Adriana Furtado Lima, Lina Marchese Olivieiro dos Santos, Deise Lobo Cavalcante e a Professora. Maria Luiza Machado Campos.

A Deus, pela bênção da vida, saúde, luz e oportunidades.

Aos meus pais, José Carvalho de Albuquerque e Meire Sebastian Alves de Albuquerque, pela educação, apoio, amor, estímulo e entusiasmo, em todas as fases da minha vida.

RESUMO

ALBUQUERQUE, A. A. **Balanceamento de Chamadas E.164 VoIP para a rede de telefonia pública.** 2007. 139f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.

Com uso crescente do VoIP para estabelecer chamadas com a rede de telefonia fixa comutada através de *gateways* VoIP/PBX conectados a servidores VoIP instituicionais, surge a necessidade de efetuar o balanceamento de chamadas, evitando que ligações sejam direcionadas sempre para uma mesma instituição, o que favorece o congestionamento de recursos do *gateway* local e até um eventual não completamento da ligação.

O recurso de atrasar a confirmação de aceitação de uma chamada com base em uma métrica de utilização dos *gateways* institucionais é explorada para a implementação de algoritmos distribuídos para balanceamento das chamadas. Nos cenários em que a variabilidade de retardos na rede é pequena e as oscilações podem ser medidas com alguma precisão, é explorado um algoritmo com uso de funções analíticas. Nos cenários onde há muita variabilidade de retardos ou impossibilidade de medir atrasos com precisão, uma proposta de algoritmo distribuído baseada em lógica nebulosa é apresentada. Observamos que este algoritmo com lógica nebulosa é robusto e seu desempenho se aproxima daquele obtido com a otimização do algoritmo distribuído com uso de função analítica. As duas variantes de algoritmos distribuídos são estudadas via simulação em MatLab.

Uma solução de balanceamento centralizado, que evita a adição de qualquer atraso extra na confirmação da chamada, é implementada através da integração da sinalização SIP com consulta a DNS com ENUM. Detalhes de implementação dessa solução para a arquitetura do serviço fone@RNP são apresentados.

A dissertação também discute a adequação do uso do balanceamento centralizado e distribuído nas várias arquiteturas H.323 e SIP.

ABSTRACT

ALBUQUERQUE, A. A. **Balanceamento de Chamadas E.164 VoIP para a rede de telefonia pública.** 2007. 139f. Dissertação (Mestrado em Informática) – Instituto de Matemática, Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.

The increasing use of VoIP to estabblish PSTN calls through gateways VoIP/PBX connected to instutional VoIP servers makes it necessary to deploy call balancing. Call balancing prevents calls from being always directed to the same institution and avoids congestioning local voice gateway resources, which can cause call non-completion.

The ability to delay the confirmation of call acceptance, based on a metric which reflects gateway utilization, is explored for the implementation of distributed call balancing algorithms. For scenarios where the variability of delays in the network is small and jitter can be measured with a certain precision, an algorithm based on analytical functions is described. For scenarios where there is great jitter or impossibility to measure delays with precision, a distributed algorithm based in fuzzy logic is proposed. It is shown that this algorithm is robust and its performance is close to the optimized solution for the distributed algorithm with analytical function.

The variants of distributed algorithm are discussed through simulation in MatLab.

The implementation of centralized balancing, which avoids the addition of any extra delay in the confirmation of the call, is obtained through the integration of SIP signalization using DNS queries with ENUM. Details of this solution for fone@RNP service architecture are presented.

The dissertation also discusses the adequacy of the use of centralized and distributed balancing with various H.323 and SIP architectures.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ACK	*ACKnowledge*
AGI	*Asterisk Gateway Interface*
BD	*Banco de dados*
CAC	*Call Admission Control*
DDD	*Discagem Direta a Distância*
DDI	*Discagem Direta Internacional*
DGK	*Directory Gatekeeper*
DSER	*Directoy OpenSER*
DNS	*Domain Name Service*
ENUM	*E.164 NUmber Mapping*
E.164	*Recomendação da ITU para plano de numeração*
GK	*Gatekeeper*
GNUGK	*Gatekeeper GNUGK*
GPS	*Global Positioning System*
GW	*Gateway*
H.323	*Padrão ITU-T*
HTTP	*HyperText Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IP	*Internet Protocol*
ISDN	*Integrated Services Digital Network*
ISP	*Internet Service Provider*
ITU-T	*International Telecommunication Union - Telecommunication*
LCF	*Location Confirm*
LCR	*Least Cost Routing module*
LRQ	*Location Request*
NAPTR	*Naming Authority Pointer*
NTP	*Network Time Protocol*
OSP	*Open Settlement Protocol*
PBX	*Private Branch eXchange*
PC	*Personal Computer*
PHP	*PHP: Hypertext Preprocessor*
POP	*Ponto de presença*
PSTN	*Public Switched Telephone Network*
QoS	*Qualidade de Serviço*
RIP	*Request in Progress*
RFC	*Request for Comment*
RNP	*Rede Nacional de Enisno e Pesquisa*
RTCP	*Real-Time Transport Control Protocol*
RTFC	*Rede de Telefonia Fixa Comutada*
RTT	*Round Trip Time*
SER	*SIP Espress Router*
SIP	*Session Initiation Protocol*
SRV	*Registro de serviço*
SV	*Servidor VoIP*
SV_V	*Servidor VoIP Virtual*

TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
URI	*Uniform Resource Identifiers*
VoIP	*Voice Over IP*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Na atualidade, o uso da tecnologia de transmissão de voz em redes IP (*Internet Protocol*) está se tornando comum a cada dia, com várias iniciativas dessa tecnologia surgindo em várias frentes. Todavia, para garantir a qualidade e desempenho da interoperação entre o ambiente VoIP (*Voice Over IP*) e a RTFC (Rede de Telefonia Fixa Comutada), é imprescindível que haja uma eficiente distribuição de carga entre os *gateways* de voz que encaminham as chamadas provenientes da rede VoIP para a RTFC.

Com o crescimento da utilização do VoIP, as chamadas destinadas à RTFC ocorrem com mais freqüência, o que pode levar a um aumento na probabilidade de elas ocorrerem simultaneamente. Como os equipamentos não podem encaminhar infinitas chamadas, assim algumas podem não ser completadas, o que é um problema. Desafios como estes precisam ser enfrentados para que um serviço VoIP se torne escalável.

Assim, para resolver essa questão, são propostos nesta dissertação algoritmos e mecanismos de balanceamento de chamadas, baseados em atrasos para o envio de primitivas ou priorização dos *gateways* segundo uma métrica. Antes de iniciar a discussão sobre balanceamento, é feita a seguir uma introdução sobre o problema a ser enfrentado.

Quando se utiliza o VoIP, é possível estabelecer comunicação de voz entre os usuários diretamente conectados à rede de dados (*Intranet* ou *Internet*), e, num outro cenário, onde pelo menos um deles pertence à rede de telefonia comutada. Em cada um desses cenários o

usuário encontra-se conectado a uma rede de comunicação e isto caracteriza o ambiente ao qual ele pertence; ou seja, basicamente, existem usuários VoIP ou da telefonia. Assim, para uma chamada que envolva mais de um ambiente, aplica-se uma conversão entre os protocolos envolvidos, o que é feito por um *gateway*. Esse *gateway* que permite a comunicação entre os ambientes VoIP e de telefonia é chamado de *gateway* VoIP/PBX, porque é assumido que ele se conecta ao PBX (*Private Branch eXchange*).

Uma instituição pode ser entendida como se fosse um provedor VoIP, que, ao implementar um serviço de telefonia sobre IP (*Internet Protocol*) para seus associados, viabiliza a realização de chamadas entre eles, para outras instituições ou para a rede de telefonia. Além disso, essas instituições possuem servidores VoIP, responsáveis pela autenticação de usuários e controle de chamada, e *gateways* de voz.

Nos ambientes VoIP, dependendo da configuração dos servidores, as chamadas encaminhadas entre as instituições, passam pelos seus servidores VoIP; assim, dois usuários finais ou *gateways* nunca se comunicam diretamente. Esses usuários precisam se registrar no servidor VoIP da sua instituição e este, sim, realizará as chamadas para o destino pretendido, solicitando-as ao seu servidor.

No caso de uma chamada envolvendo o ambiente VoIP e um usuário na RTFC, na configuração utilizada neste trabalho, ela passa por um servidor VoIP conectado a um *gateway*, responsável por encaminhá-la a um PBX (*gateway* VoIP/PBX), todavia a chamada poderia ir diretamente para o *gateway* que faz a comunicação com a RTFC; seja como for, a chamada pode ser comutada para a RTFC. Essa interligação é feita por meio de uma quantidade limitada de canais de voz, sendo que cada um deles permite apenas uma chamada simultânea, ou seja, a quantidade de canais determina a capacidade de um *gateway* VoIP/PBX

para encaminhar chamadas simultâneas via PBX, o que é uma limitação. Caso todos os canais estejam ocupados, diz-se que o *gateway* está saturado, ou com ocupação total dos canais.

Observe que, no cenário apresentado, é necessário utilizar um *gateway*, o qual, como foi mencionado, encaminha para a RTFC uma quantidade limitada de chamadas simultâneas. Desta forma, um número crescente de chamadas utilizando o *gateway* VoIP/PBX pode provocar a ocupação total dos seus canais de voz. Há duas maneiras para contornar essa limitação na quantidade de canais de um *gateway*: aumento da quantidade de canais de voz (linhas telefônicas) para comunicação com a RTFC, ou utilização de mais de um *gateway*. A primeira opção não é muito escalável porque fica limitada à quantidade máxima de canais que um *gateway* pode suportar, seja ela relacionada aos *softwares* ou aos equipamentos envolvidos. A outra é mais atraente, primeiro porque é mais escalável e por outro lado, porque é redundante a falhas.

Em um ambiente de rede não comercial, onde os serviços são proporcionados de forma cooperativa pelas várias instituições pertencentes à rede, a conectividade com a RTFC pode ser realizada através dos *gateways* das próprias instituições. Assim, mais de um *gateway* é utilizado para o encaminhamento à RTFC.

Na cidade onde a chamada é encaminhada para a RTFC, é desejável que exista mais de uma opção para tal, o que pode ser alcançado através de vários pontos de escoamento com *gateways* VoIP/PBX. Não basta, entretanto, que esses pontos estejam em uma única instituição, pois a rede de uma instituição pode estar inatingível por algum motivo. Uma alternativa mais redundante, então, é ter a possibilidade de enviar chamadas por instituições distintas, ou seja, redes de dados distintas. Dessa forma, caso o(s) *gateway*(s) localizado(s) em uma instituição não possa(m) ser alcançado(s), a chamada será encaminhada através de outra.

No VoIP, quando há vários destinos que podem encaminhar chamadas para a RTFC de uma cidade (empresas/instituições colocalizadas), um pedido de encaminhamento (solicitação de chamada) é enviado para cada um dos seus servidores VoIP. Esse pedido possui o número telefônico (número E.164 [1]) com o qual se deseja estabelecer uma comunicação de voz e, ao recebê-lo, cada servidor de destino informa se está apto a encaminhar a chamada, enviando uma confirmação. Antes de enviar essa confirmação, o servidor contatado (chamado) apenas verifica se é possível encaminhar a chamada ao número E.164 de destino informado no pedido de encaminhamento, ou seja, não são realizados em algumas implementações de servidores VoIP, por exemplo, verificação de disponibilidade de canais nos *gateways* e balanceamento de chamadas com outros destinos.

É desejável, então, que a decisão de um algoritmo de balanceamento aponte a instituição com maior chance de sucesso para o encaminhamento de uma chamada, porque apenas uma tem a oportunidade de recebê-la, como foi descrito.

Como a instituição que recebe as chamadas é aquela cuja confirmação chega primeiro ao chamador, uma delas tende a receber mais chamadas do que as outras, ou seja, quem encaminha a chamada é influenciado pela topologia da rede. A situação de saturação dos canais de um *gateway* VoIP/PBX se torna crítica, principalmente em uma instituição onde a utilização do serviço VoIP é crescente, o que pode acontecer com o aumento de usuários locais, ou com o aumento de tráfego originado em outras instituições pertencentes ao serviço. Uma instituição, ao receber mais chamadas para a sua cidade, acaba tendo um gasto com telefonia mais elevado, o que pode não ser desejável.

Utilizando um algoritmo ou mecanismo de balanceamento de carga, as chamadas direcionadas à RTFC serão entregues, preferencialmente, ao *gateway* que apresente menor utilização instantânea dos seus canais de voz, evitando-se o acontecimento de chamadas não

completadas, o que pode ocorrer quando um *gateway* já está com todos os seus canais de voz ocupados.

Em uma situação em que um *gateway* está saturado, é desejável que as chamadas sejam encaminhadas para outro *gateway* com canais de voz disponíveis, evitando a subutilização de recursos de um serviço VoIP.

No VoIP há protocolos, como o SIP (*Session Initiation Protocol*) e o H.323 (*Padrão ITU-T*), que definem os ambientes VoIP abordados nessa dissertação. Eles são muito comuns atualmente e, em função disso, o balanceamento deve ser aderente a estes dois protocolos de sinalização.

1.1 Balanceamento distribuído e balanceamento centralizado

Para o balanceamento existem algumas opções de projeto de acordo com a literatura [3, 4, 5 e 6] e, entre elas: **(i) balanceamento distribuído de chamadas**, não havendo, na tomada de decisão, necessidade de consultar uma entidade centralizada, o que faz com que o balanceamento não fique suscetível a falhas; **(ii) balanceamento de chamadas centralizado**, no qual uma entidade central, para onde as informações são enviadas, toma as decisões de encaminhamento.

No algoritmo centralizado existe uma entidade central [3], que precisa receber atualizações das métricas a serem utilizadas pelo algoritmo de decisão e, além disso, ser consultada antes do encaminhamento de uma chamada, o que provoca um atraso adicional para a chamada ser completada. As informações precisam ser sincronizadas com freqüência na entidade central [4], evitando erros na tomada de decisão do algoritmo. A complexidade de uma arquitetura centralizada [4] é um ponto negativo.

Com o aumento do número de clientes da entidade centralizadora, as consultas tendem a crescer [4], assim um aumento de tráfego na rede é provocado [4] em torno desse ponto de decisão; além disso, existe a necessidade de um aumento no poder computacional nesta entidade, a fim de realizar a tomada de decisão em tempo hábil. Nesse tipo de algoritmo a entidade central é um ponto de falha [3], que pode ter problemas no seu funcionamento ou na comunicação com a rede. Neste tipo de algoritmo, cuidados devem ser tomados para que a confiabilidade e eficácia na tomada de decisão não sejam comprometidas.

No algoritmo distribuído, cada elemento gerencia os seus próprios recursos [5], e pode tomar decisões baseando-se apenas em informações locais. Quando são utilizadas informações de outros elementos para viabilizar o funcionamento do algoritmo, ele é classificado como global [3]. Neste tipo de algoritmo não existe um elemento central a ser consultado e o ponto de falha única é eliminado. Um ponto que deve ser levado em consideração é a escalabilidade do algoritmo distribuído [3, 5], o que é importante em serviços onde a sua utilização é crescente.

Na literatura em [6], existe uma proposta de balanceamento para VoIP que é centralizada, porém ela é específica ao ambiente H.323. Nesta proposta da literatura, um *gatekeeper* próprio é desenvolvido, alguns elementos utilizados em [6], como os servidores e clientes VoIP, têm comportamentos diferentes daqueles utilizados pelos *software* livres encontrados atualmente, o que é um entrave quando a proposta precisa ser aplicada num ambiente em produção.

Na literatura não são conhecidos outros trabalhos de balanceamento de chamadas VoIP publicados, embora certamente existam de alguma forma implementações de soluções comerciais.

Levando em consideração o que foi exposto, a implementação de um balanceamento centralizado ou distribuído, que considere as limitações e versatilidades dos elementos da arquitetura H.323 e SIP, é uma necessidade.

Algumas formas de encaminhamento de chamadas são possíveis de acordo com a literatura, aqui isto é descrito como os mecanismos de encaminhamento de chamadas. Basicamente, essas formas se dividem em duas: uma primitiva de sinalização VoIP é enviada simultaneamente para cada um dos servidores de destino da chamada, ou a tentativa de contato destes servidores segue uma prioridade pré-determinada. Além disso, as arquiteturas dos ambientes VoIP precisam ser consideradas, sendo elas determinadas pelas configurações dos equipamentos e servidores VoIP.

1.2 Requesitos para balanceamento

O balanceamento atua quando chamadas originadas em determinada cidade são destinadas à RTFC [2] de uma outra cidade (*chamadas externas*) e, mais precisamente, quando a RTFC pode ser alcançada por pelo menos dois *gateways* pertencentes a instituições diferentes. Nesta dissertação são apresentadas três propostas para balanceamento distribuído de chamadas VoIP e, adicionamente, uma proposta de um mecanismo de balanceamento centralizado.

Pretende-se que o balanceamento atue, levando em consideração a utilização instantânea dos *gateways* envolvidos na chamada a partir de uma métrica. Assim, aqueles *gateways* que possuirem as melhores condições, segundo tal métrica, podem encaminhar a chamada para a RTFC. Essa métrica precisa apenas indicar o quanto uma instituição está inclinada a receber uma chamada, sendo que a forma como ela é obtida ou calculada não deve interferir no funcionamento do balanceamento. Isso significa que um algoritmo ou mecanismo

de balanceamento deve ser transparente à métrica utilizada, porque, com a evolução do serviço VoIP, outros fatores podem influenciar a métrica a ser utilizada.

Os algoritmos utilizados não devem atrasar demasiadamente a admissão das chamadas, o que causaria insatisfação nos usuários e, até mesmo, desistência da chamada por expiração de tempo (*timeout*). Assim, é desejável que o algoritmo atrase a admissão o mínimo possível (minimizar atrasos).

Na implementação de algum tipo de balanceamento, seu impacto frente às possíveis arquiteturas VoIP deve ser avaliado, preferencialmente utilizando soluções padrão. Em um ambiente em produção, uma proposta de balanceamento de chamada precisa ser escalável, pois o aumento de instituições e usuários utilizando o serviço VoIP não deve alterar o funcionamento do algoritmo. Além disso, na tomada de decisão o balanceamento deve ser justo sem que haja favorecimento de alguma instituição, ou provocar erros na distribuição de chamadas.

É desejável que o algoritmo de balanceamento seja extensível para os ambientes H.323 e SIP sem muitas modificações, ou seja, que tenha pouco impacto nas arquiteturas e nos ambientes VoIP (independência do ambiente VoIP).

Outro ponto importante é a robustez, o algoritmo deve convergir para uma decisão quase correta quando houver falhas em equipamento ou na comunicação, ou seja, é preciso evitar grandes discrepâncias entre uma decisão adotada e a esperada.

Por fim, as propostas de algoritmos devem apontar as arquiteturas para as quais elas são mais adequadas e devem-se comparar os resultados obtidos entre os algoritmos segundo alguns critérios, como: justeza, distribuição de chamadas, valores próprios e médios dos

atrasos gerados pelo algoritmo para admissão da chamada, impacto para implementação do algoritmo e escalabilidade.

Simulações podem ajudar a determinar aspectos positivos ou negativos das propostas frente a certas situações. Áreas como sistemas distribuídos e lógica nebulosa podem oferecer subsídios para alguma proposta e, além disso, recursos existentes nos servidores VoIP precisam ser analisados, porque podem apontar uma possível solução.

1.3 Ambiente de implementação

A RNP (Rede Nacional de Pesquisa) oferece, para as instituições integrantes da sua rede o serviço fone@RNP, sendo utilizadas as sinalizações H.323 e SIP para a geração de chamadas interinstitucionais. Os elementos (servidores VoIP) que compõem o fone@RNP se comunicam com uso da arquitetura TCP/IP, podendo estar conectados em qualquer ponto da sua rede local, que é normalmente conectada ao *backbone* da RNP em algum POP (Ponto de Presença) na cidade.

O cenário de aplicação das propostas de balanceamento é uma rede acadêmica, ou seja, não comercial. Pretende-se que algumas das propostas de balanceamento de chamadas, aqui apresentadas, sejam implementadas no fone@RNP.

1.4 Objetivos e organização

Nesta dissertação, as arquiteturas dos ambientes H.323 e SIP são analisadas quando ocorre o processo de localização de um destino e o encaminhamento de chamada; em um segundo passo, as propostas para balanceamento de chamadas VoIP são desenvolvidas.

Duas soluções distribuídas fazem uso de funções analíticas, o que permite forçar atrasos a uma primitiva de confirmação. Estas soluções podem ser empregadas em redes onde

os retardos da rede são conhecidos e suas oscilações podem ser medidas com precisão; assim, conhecer o *jitter* é decisivo para uma distribuição de chamadas justa. O valor dos atrasos para envio destas primitivas ocorre em função da utilização dos *gateways* dos servidores VoIP, quanto maior essa utilização dos canais de voz dos *gateway*, menor deve ser a chance do servidor conectado a ele de receber a chamada, o que é alcançado atrasando o envio de uma das primitivas de sinalização VoIP. Além disso, as funções analíticas são determinadas com a ajuda de pontos de referência e as utilizações citadas são apontadas por uma métrica.

Outra solução distribuída para o balanceamento de chamadas utilizando lógica nebulosa (*Fuzzy Logic*) é proposta, sendo mostrado que esta proposta é insensível a variabilidade do *jitter* da rede.

Outra proposta foi concebida, onde uma entidade central toma as decisões de balanceamento, sem adicionar atrasos extras às primitivas envolvidas no processo de admissão da chamada. Esta entidade determina prioridade para quem vai receber a chamada, de acordo com a utilização de canais dos *gateways*. Assim, as entidades que possuem *gateways* com menor ocupação de canais, de acordo com uma métrica, são contatadas primeiro para uma tentativa de encaminhamento das chamadas.

Esta dissertação está estruturada em oito capítulos, como descrita a seguir. No Capítulo 2 são analisadas formas de encaminhar as chamadas E.164 em algumas arquiteturas, analisando como a aceitação da chamada é realizada. No Capítulo 3 um algoritmo de balanceamento distribuído de chamadas VoIP baseado na geração de atrasos de primitivas é proposto e analisado; no Capítulo 4 esse algoritmo é aprimorado, levando a uma diminuição significativa dos atrasos na admissão das chamadas. O Capítulo 5 descreve um algoritmo de balanceamento de chamadas utilizando *lógica nebulosa*, também com características distribuídas e geração de atrasos para tomada de decisão. No Capítulo 6 são mostrados e

comparados os resultados obtidos na simulação desses algoritmos, que utilizam atrasos determinísticos para comprovar o funcionamento dos algoritmos. No Capítulo 7 um mecanismo centralizado para balanceamento de chamadas será mostrado, discutindo-se uma avaliação de sua implementação. Por fim, no Capítulo 8 são apresentadas as conclusões e os trabalhos futuros.

Capítulo 2

ENCAMINHAMENTO E.164 EM H.323 E SIP

Neste capítulo é analisado como o encaminhamento de chamadas E.164 no ambiente H.323 e SIP é realizado; e a métrica proposta para calcular a utilização dos GWs (*gateways*) é apresentada.

Para entender o funcionamento dos algoritmos, é necessário analisar como a sinalização VoIP utilizada na localização envolvida no encaminhamento de chamadas ocorre, frente as possíveis arquiteturas de um serviço VoIP. As sinalizações abordadas nessa dissertação são a H.323 e a SIP. Além disso, serão apresentados os elementos que compõem um serviço de telefonia sobre IP e seu funcionamento no encaminhamento E.164, formando, assim uma base necessária de conhecimento.

Este capítulo foi organizado como descrito a seguir: Na seção 2.1 são apresentados detalhes sobre o processo de localização de destino de uma chamada, aspectos sobre as limitações dos *gateways* e considerações sobre o balanceamento de chamadas envolvendo instituição com múltiplos *gateways* VoIP/PBX. Na seção 2.2 é apresentada e justificada a métrica proposta para os algoritmos de balanceamento. Na seção 2.3 são apresentadas as arquiteturas para o encaminhamento E.164 nos ambientes SIP e H.323; e na seção 2.4 são propostas algumas formas para determinação de atrasos, que são necessários para o funcionamento do algoritmo.

2.1 Localização de destino e aceitação de chamadas VoIP

A transmissão de voz sobre IP (VoIP) permite a comunicação de voz sobre a estrutura TCP/IP, sendo possível integrá-la a outros serviços disponíveis em uma rede. Há vários protocolos que podem ser utilizados no VoIP, mas, nesta dissertação, apenas o H.323 [1, 7 e 8] e o SIP [4, 9 e 10] estão envolvidos, caracterizando os ambientes VoIP.

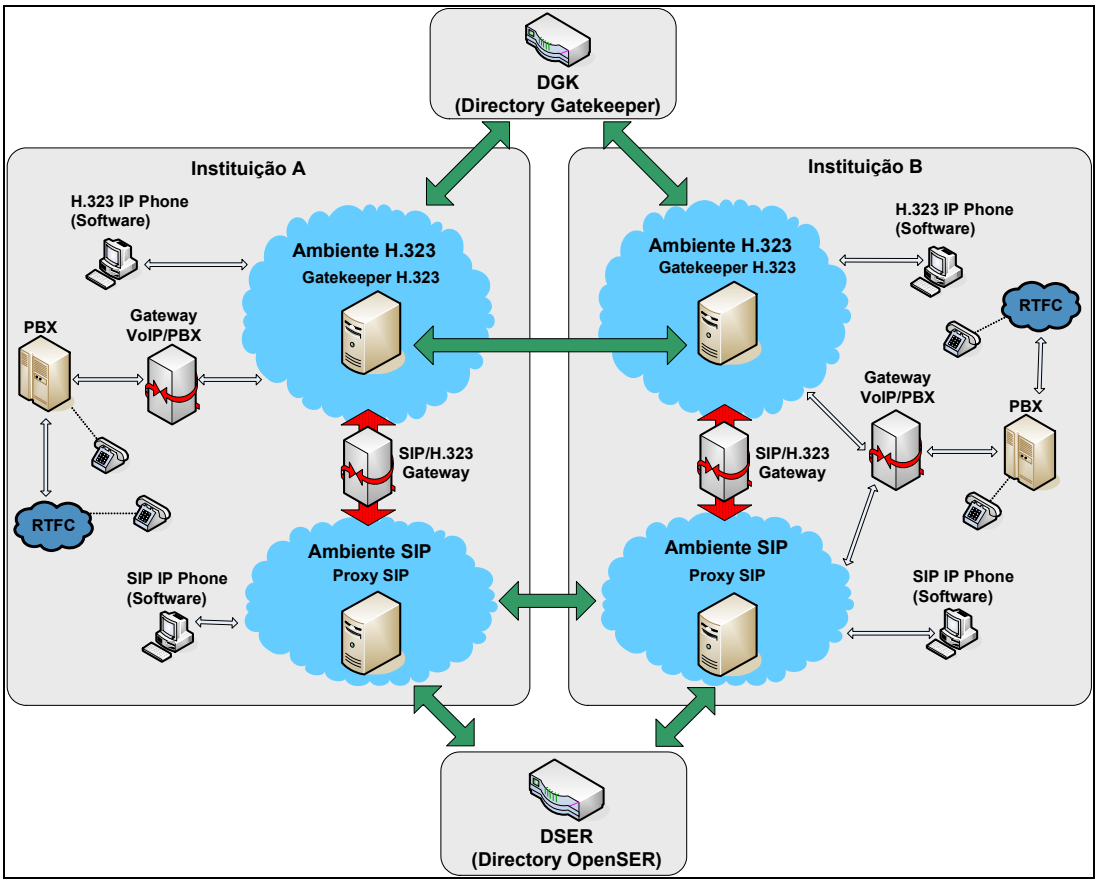


Figura 1. Exemplo da arquitetura de um serviço VoIP

O padrão H.323 é parte da família de recomendações da ITU-T (*International Telecommunication Union Telecommunication Standardization sector*) e tem como objetivo especificar sistemas multimídia para redes baseadas em pacotes sem tratamento de QoS (*Qualidade de Serviço*); já o SIP é um protocolo de aplicação similar ao HTTP (*HyperText Transfer Protocol*), baseado em texto e utilizado para iniciar sessões interativas entre

usuários. Considera-se para esta dissertação que o H.225 do H.323 e o SIP operam sobre o protocolo UDP (*User Datagram Protocol*), o que é a situação mais comum para o VoIP.

A Figura 1 mostra um cenário típico do serviço fone@RNP, composto por duas instituições. Os usuários de cada uma delas podem encaminhar chamadas entre si, ou para os da outra instituição e, conforme a figura, usuários SIP podem realizar chamadas para usuários H.323, através de um *gateway* (GW) chamado de *gateway* SIP/H.323 e vice-versa. Na instituição *A*, a RTFC só pode ser alcançada pelo ambiente H.323, através de um *gateway* conectado ao PBX (*gateway* VoIP/PBX), ao contrário da outra instituição, onde qualquer usuário H.323 ou SIP é capaz de alcançá-la. As estruturas de autenticação dos usuários e contabilidade de chamadas foram suprimidas nas figuras, pois, para o entendimento dos algoritmos de balanceamento propostos, não é importante entrar em muitos detalhes de como isto ocorre.

Em cada um dos ambientes, H.323 e SIP, existem servidores VoIP que são responsáveis pela autenticação dos usuários, controle das chamadas, autorização para os terminais a encaminharem chamadas e realização de comunicação com uso da arquitetura TCP/IP. No H.323, o servidor é chamado de *gatekeeper* (GK) e, no SIP, o *proxy SIP* é quem realiza este papel. Para que um usuário pertença a um destes ambientes, ele precisa se autenticar no servidor correspondente, o que é realizado através de um nome de usuário e senha, ou seja, usuários do ambiente SIP se registram em um *proxy SIP* e os do ambiente H.323 estão registrados no *gatekeeper*. Por questão de simplicidade, nesta dissertação será utilizado o nome Servidor VoIP (SV), quando for indiferente detalhar a qual servidor se refere o texto.

Na Figura 1, os *gatekeepers* da instituição A e B permitem que um envie chamadas pelo outro e vice-versa, mantendo assim uma relação de confiança entre ambos. Comumente,

quando os GKs possuem esta relação de confiança, eles são chamados de vizinhos. Esse termo também pode ser utilizado no ambiente SIP, embora haja um outro muito empregado neste ambiente: *peer* confiável.

Cada um desses ambientes pode encaminhar chamadas para a RTFC ou para a rede privada da instituição, o que é factível com uso do *gateway VoIP/PBX* interligado ao PBX dessa instituição. Porém, antes de a chamada passar através desse *gateway*, ela precisa passar pelo servidor VoIP da instituição.

Em uma instituição pode haver um ou mais SV's (H.323/SIP), mas pressupõe-se que sempre existe um servidor principal que recebe as chamadas originadas em um SV de outra instituição e, caso um ambiente tenha mais de um SV, eles são configurados de forma que fiquem hierarquicamente abaixo do principal.

Um usuário pode realizar uma chamada utilizando um terminal, que tem funções semelhantes às de um telefone. Estes terminais podem ser *software* (*softphone*) instalados em computadores ou *hardware* dedicados, ambos têm de possuir alguma conectividade com a rede de dados. Qualquer um deles precisa se registrar em um SV, pois somente assim suas chamadas podem alcançar os demais usuários registrados no mesmo SV ou num outro, e eles podem ser chamados de telefones IP, ou simplesmente de telefones convencionais.

Para uma chamada chegar ao usuário destino, é necessário conhecer seu identificador, que é o número E.164, dividido em prefixo E.164 e identificador do usuário na instituição. Através dele, um usuário pode ser localizado e receber chamadas dentro de uma arquitetura VoIP.

Antes do número E.164, pode haver os códigos de área e do país, que identificam uma cidade e um país, respectivamente. Assim, para alcançar um usuário em outro país, por exemplo, seriam informados o código do país, o código de área da cidade e o número E.164.

O prefixo E.164, que identifica uma instituição, é formado pelos *N* primeiros dígitos, da esquerda para a direita, do número E.164; os dígitos restantes identificam um usuário dentro da própria instituição. Além disso, um número E.164 pode ser dividido em duas categorias: a primeira, chamada de *prefixo IP*, reúne os telefones (*softphone* ou *hardware*) que se registram em algum servidor VoIP, ou seja, aqueles pertencentes ao mundo IP, diferentemente disso, a segunda categoria, o prefixo PBX, engloba os telefones alcançáveis através do PBX, seja um prefixo da RTFC ou da rede de telefonia privada.

O usuário ou servidor VoIP que inicia uma chamada é designado chamador; em contrapartida, aqueles que recebem a chamada são definidos como chamados, usuários destino ou usuários finais. Essas definições também podem ser aplicadas à instituição, bem como a qualquer elemento VoIP ou de telefonia. Por exemplo, o SV onde está registrado o usuário que inicia uma chamada seria o servidor VoIP chamador (SV chamador).

Pressupõe-se que os servidores VoIP atuam como *proxy* de sinalização e mídia, ou seja, as sinalizações e as mídias envolvidas na chamada são encaminhadas primeiro para eles, antes de alcançarem os usuários finais, ou *gateways*. Isso significa que estes usuários não se comunicam diretamente, configuração tal que facilita o controle e a implantação de mecanismos de QoS e segurança em uma rede. No decorrer desta dissertação, é possível que um servidor atue somente como *proxy* de sinalização e, ao utilizar esta configuração, a sinalização VoIP envolvida na chamada passa por ele, antes de algum usuário recebê-la.

Quando um usuário SIP ou H.323 inicia uma chamada (chamador), é necessário enviar uma sinalização para o servidor onde ele está registrado. Ela possui o número E.164, para o

qual a chamada se destina (número E.164 destino), além, é claro, o endereço IP do próprio terminal chamador. Após recebê-la, o servidor VoIP verifica se o número E.164 destino encontra-se registrado e, caso esteja, a chamada é encaminhada ao usuário correspondente. Se o número E.164 destino não for encontrado, é enviado um **pedido de localização** para um servidor de localização (sinalização Figura 2b-1 ou Figura 2a-1), que possui registros com os endereços IPs dos SVs existentes em um serviço VoIP e quais são os prefixos E.164 encaminháveis por estes servidores. A função desse servidor de localização é encontrar quem são os possíveis destinos de uma chamada.

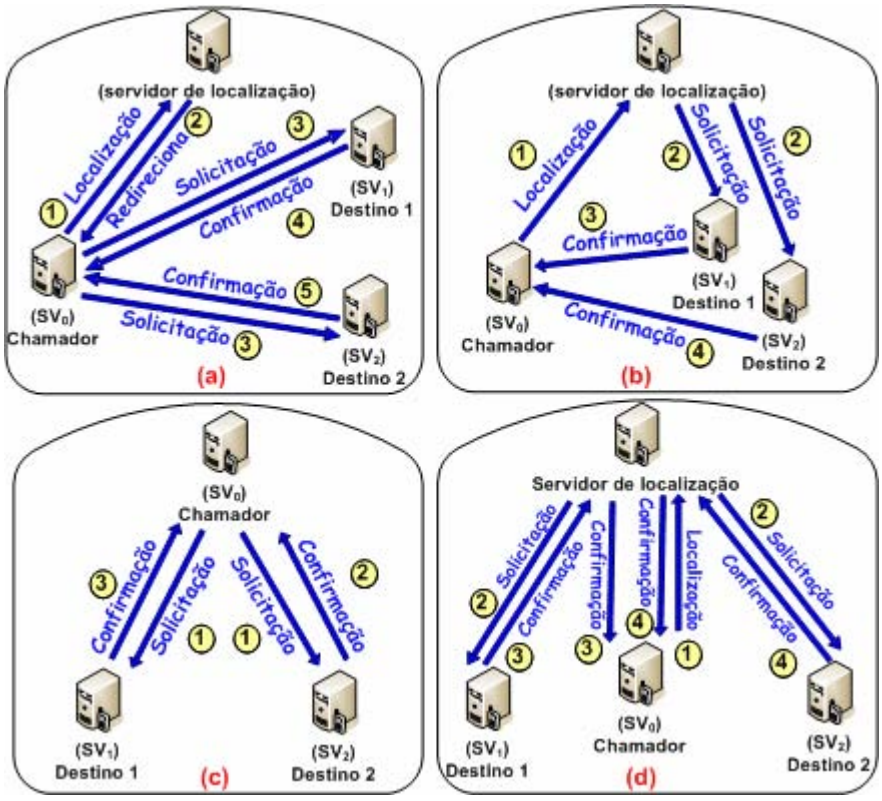


Figura 2. Sinalização para localização de um SV destino

Nesse processo, tais prefixos são comparados com o prefixo do número E.164 do chamado e, considerando os dígitos da esquerda para a direita, aqueles que possuírem a maior quantidade de números consecutivos idênticos ao prefixo chamado são ditos **prefixos mais significativos**. Assim, na configuração interna do servidor de localização, se existir algum

prefixo classificado como prefixo mais significativo, o seu servidor é considerado como um servidor de destino da chamada.

No ambiente H.323, o servidor de localização é chamado de DGK (*Directory Gatekeeper*) e, no ambiente SIP, foi adotado o nome DSER (*Directory OpenSER*). A palavra OpenSER, na sigla DSER, foi originada do nome do *software* OpenSER [11], que é a implementação de um *proxy* SIP. Em algumas arquiteturas, os dois ambientes coexistem, podendo seus usuários, assim, realizar chamadas entre si. Nesse caso, existe um *gateway* responsável pela conversão da sinalização entre os mundos H.323 e SIP, chamado de *gateway* SIP/H.323, ou *gateway* H.323/SIP. No texto, o sufixo H.323/SIP será utilizado quando a chamada for originada no H.323 e destinada ao SIP, enquanto que o outro, quando ela seguir o percurso inverso.

Os *gateways* mencionados até o momento podem ser implementados usando *software* aberto como o *Asterisk* [12 e 13], instalados em um PC, ou através de equipamentos proprietários, mas estes detalhes serão mencionados apenas quando isto for relevante.

Após o servidor de localização receber um pedido e encontrar os endereços IP dos SVs de destino, esses últimos são contatados para encaminhar a chamada ao usuário final, mas um só deles é escolhido para isto. Esse contato pode se dar de duas formas: o servidor de localização se comunica diretamente com os SVs de destino (Figura 2b), ou o próprio chamador faz o contato, após receber do servidor de localização uma listagem dos SVs de destino (Figura 2a).

Tomando como exemplo a Figura 2b, onde o servidor de localização contata os SVs de destino, cada um deles recebe uma **solicitação de chamada** (Figura 2b-2) e, aqueles que puderem encaminhá-la, enviam uma **confirmação de solicitação** (Figura 2b-3 e Figura 2b-4) para o chamador (ConfSol). Por questões de implementação, a primeira confirmação que

chegar ao servidor chamador será a única a ter a chance de receber e encaminhar a chamada; isto é importante e será utilizado pelos algoritmos de balanceamento que utilizam atrasos para envio de confirmações. Todas as confirmações que chegarem depois não serão consideradas, assim os seus servidores não terão oportunidade de encaminhar a chamada.

Um servidor de localização pode atuar como *proxy* das sinalizações envolvidas no processo de localização (Figura 2d) mas, se nada for mencionado, considera-se que ele não atua desta maneira (Figura 2a e Figura 2b).

O processo de localização descrito centraliza, em um servidor de localização, informações como endereços IP dos SVs de um serviço VoIP e seus prefixos E.164 (Figura 2a, Figura 2b e Figura 2d) e, apesar disto não ser tolerante a falhas, é muito utilizado. Existe, entretanto, outra possibilidade: em vez de ter essas informações em um servidor central, elas podem estar presentes em cada uma das instituições (Figura 2c). Assim, o processo de localização é realizado pelos SVs institucionais, tornando a decisão de localização descentralizada, tornando tal processo menos susceptível a falhas.

Um pedido de localização pode resultar em retornos de mais de um servidor VoIP capaz de encaminhar a chamada, sendo um exemplo típico disto uma chamada destinada à RTFC. Existindo mais de uma instituição numa cidade que possa encaminhar a chamada até o usuário final (chamado) na RTFC, uma solicitação (Figura 2b-2) é enviada para cada uma delas, mas apenas uma instituição encaminha a chamada através do seu *gateway*. Esse envio simultâneo de sinalização (solicitação) referente a uma mesma chamada é dito *fork* paralelo (encaminhamento paralelo). Além desse, existe o encaminhamento sequencial (*fork* sequencial), no qual determinada instituição tem prioridade para estabelecer uma comunicação e, caso isto não seja possível, por qualquer motivo, outra é contatada. Esta

prioridade da instituição é determinada segundo algum critério, o que pode ser diferente para cada serviço VoIP.

No *fork* paralelo, apesar de todos os possíveis SVs destino receberem uma solicitação de chamada, apenas um é escolhido para encaminhá-la; no SIP os servidores que não irão encaminhar uma chamada são informados sobre isto, enquanto que no H.323 apenas o servidor que é escolhido para encaminhar a chamada recebe alguma informação. Tanto no SIP, como no H.323, o SV escolhido para encaminhar a chamada é aquele cuja confirmação de solicitação chegar primeiro ao chamador, confirmando a solicitação de chamada e realizando o encaminhamento da chamada. As outras confirmações não são consideradas, então estes servidores não têm oportunidade de encaminhar a chamada.

No caso de haver canais de voz disponíveis nos *gateways* a chamada será completada com sucesso; caso contrário ela não será completada e retornará erro ao chamador.

2.1.1 Limitações dos gateways VoIP/PBX do SV chamado

O PBX é um equipamento que comuta chamadas para a RTFC ou para a rede de telefonia privada. Além disso, o *gateway* de voz provê uma interface com o PBX com a rede de dados, possibilitando o estabelecimento de comunicação entre os usuários dos ambientes VoIP e os da RTFC, ou de ramais da instituição.

Quando a chamada tiver como destino a RTFC, ela será encaminhada para o *gateway VoIP/PBX,* seja pelo ambiente H.323, ou SIP. Todavia, este *gateway* possui um número limitado de chamadas simultâneas que pode encaminhar, porque ele se interliga ao PBX da instituição com um número restrito de linhas telefônicas (ramais), sendo que cada linha permite a ocorrência de apenas uma chamada por vez.

Um exemplo da topologia de um serviço VoIP é mostrado na Figura 1, na qual as instituições *A* e *B* estão em uma mesma cidade, sendo capazes de encaminhar chamadas VoIP para a RTFC através dos seus *gateways* interligados aos seus PBXs. Uma chamada H.323 originada em outra cidade, destinada ao código de área da RTFC da cidade em questão, é encaminhada a um dos *gatekeepers* existentes, que a repassa ao seu *gateway* e esse, por sua vez, a encaminha para o seu PBX. Assim, a chamada chega ao usuário final na RTFC. Quando um usuário na RTFC, ou em um ramal da instituição, realiza uma chamada destinada a um terminal VoIP, o percurso inverso é seguido, ou seja, primeiro a chamada passa pelo PBX, depois vai para o *gateway* VoIP/PBX, somente então chegando ao SV da instituição, para ser roteada até o seu destino no ambiente VoIP. Caso esta chamada envolva o ambiente SIP e os ambientes conectados ao PBX, a mesma seqüência de passos do H.323 é seguida.

Os *gateways* VoIP/PBX possuem conexões com o PBX chamadas de canais de voz e o tipo de interface utilizada por estes *gateways* determina a quantidade de canais e a limitação de chamadas simultâneas que eles podem suportar. Aqueles com interface E1 ISDN (*Integrated Services Digital Network*), por exemplo, conectam-se a troncos digitais do PBX e são capazes de suportar 30 conexões de voz simultâneas, enquanto que a interface FXO (*Foreign eXchange Office*), usada para conexão a um ramal analógico do PBX, possibilita o estabelecimento de uma única chamada telefônica.

Como a quantidade de canais dos *gateways* de voz é limitada, o aumento do número de usuários e a expansão de um serviço VoIP podem gerar uma demanda de tráfego excessiva para ser atendida pelos *gateways*. Conforme dito, o SV que tiver a sua primitiva de confirmação chegando primeiro ao chamador é aquele que recebe a chamada com destino à RTFC. Esse procedimento simples para eleger o SV chamado tende a um uso ineficiente dos *gateways* VoIP/PBX, porque existe uma dependência da topologia da rede para a escolha do

servidor chamador. As chamadas destinadas à RTFC de uma cidade diferente daquela do servidor chamador, sempre são repassadas para um conjunto de instituições, provocando o congestionamento de seu *gateway* e, como conseqüência, o não completamento da chamada, ainda que existam canais livres em outras instituições da mesma cidade.

Quando um *gateway* está com todos os canais ocupados, naquele momento, o seu PBX é inalcançável. É importante salientar que, quando todos os *gateways* de uma cidade estiverem saturados, sem canais livres, chamadas que passam pelo *gateway* VoIP/PBX para a mesma não são completadas e são descartadas. Além dessa, outras situações podem contribuir para que uma chamada não seja completada, dentre elas a ocorrência de falhas na rede local de uma das instituições envolvidas na chamada, problemas com elementos VoIP do serviço VoIP ou das instituições envolvidas na chamada.

A saturação numa instituição pode ser contornada com o aumento do número de canais de voz com o PBX, pressupondo-se que ela possa absorver o volume de tráfego VoIP pretendido. Todavia, o compartilhamento dos recursos das várias instituições de uma mesma cidade seria uma solução melhor. O balanceamento de chamadas tem por objetivo distribuí-las entre os vários *gateways*, segundo alguma métrica. Com isso, uma diminuição na probabilidade de chamadas não completadas por saturação pode ser alcançada.

2.1.2 Múltiplos gateways em uma instituição

Nesta seção são tratadas as possibilidades de encaminhar chamadas para uma instituição que possui vários *gateways*. Assim, quando uma chamada chega a uma instituição com mais de um *gateway*, potencialmente, vários *gateways* podem encaminhá-la para a RTFC. Isto será tratado à parte do algoritmo de balanceamento, pois o balanceamento leva em consideração apenas o total de canais de voz do(s) *gateway*(s) de uma instituição. Quando, então, houver

mais de um *gateway* na instituição, a distribuição de chamadas entre eles será feita pelo seu servidor VoIP e não pelo algoritmo de balanceamento. Este último apenas decidirá qual instituição de destino será contemplada com a chamada e o servidor VoIP de tal instituição escolhe qual dos *gateways* receberá a chamada, segundo critérios locais. Este é um comportamento encontrado nos servidores; determinado pela sinalização VoIP, pela configuração e pela implementação do SV.

Foram pesquisados recursos existentes na implementação do *software* GNU *Gatekeeper*, ou simplesmente GNUGK [14], um servidor *gatekeeper* utilizado no ambiente H.323. Para o ambiente SIP, foram pesquisados recursos para o encaminhamento de chamadas envolvendo múltiplos *gateways* no *proxy* SIP OpenSER [11]. Esses servidores VoIP são instalados em um computador pessoal (PC) com o sistema operacional Linux. Ambos os servidores são recomendados para as instituições do serviço fone@RNP.

É importante observar que o balanceamento de chamadas ocorre em função da disponibilidade das instituições receberem chamadas, e apenas isso, o que é feito com base em uma métrica. A decisão de para qual *gateway* encaminhar a chamada pertence ao servidor VoIP da instituição.

Nos primeiros servidores GNUGK utilizados existia o recurso de operação *round-robin* para encaminhamento de chamadas, através do qual o *gatekeeper* não tinha conhecimento da utilização dos *gateways* e das chamadas em andamento em um dado momento. Assim, uma chamada enviada para um *gateway* podia ser descartada, por exemplo, quando ele já estivesse saturado. Isso acontecia porque, mesmo que houvesse recurso disponível em outro *gateway* da instituição, a chamada não lhe era encaminhada. A partir disso, surgiu a necessidade de utilizar esses recursos de forma mais racional, evitando perdas desnecessárias, um subcaso do balanceamento de chamadas que será tratado a seguir.

Quando uma instituição opera com mais de um *gateway* ligado ao PBX, o *gatekeeper* precisa controlar o encaminhamento das chamadas entre os *gateways*, evitando repassar chamadas para os que estejam sem canais livres. No GNUGK, o balanceamento entre *gateways* pode ser realizado a partir da versão 2.2.X. Nestas versões, é possível definir quantidade de canais de voz para os *gateways* registrados, de forma que eles não retornem confirmações enquanto estiverem saturados. Isso é possível, em tais versões, porque ele funciona em modo *proxy* de sinalização e mídia, tendo conhecimento do estado das suas chamadas.

Além disso, existe a possibilidade de associar prioridades aos *gateways*, o que propicia, por exemplo, operação em *round-robin* quando se utilizam prioridades iguais. Estas prioridades também permitem encaminhar chamadas para um *gateway* até o seu saturamento; somente quando todos os canais deste estiverem ocupados é que outro começa a recebê-las.

No OpenSER do ambiente SIP o módulo *dispatcher* permite encaminhar chamadas para múltiplos *gateways* pertencentes a uma instituição. Esse módulo opera em modo *round-robin* e, quando o encaminhamento da chamada para um *gateway* falha, outro é contatado em seguida, sendo que a chamada só não é encaminhada para o PBX quando não existirem canais livres nos *gateways* VoIP/PBX.

2.2 Métrica utilizada no balanceamento

O balanceamento de chamadas VoIP pode ser alcançado de várias formas mas, independentemente de como ele ocorra, é utilizada uma métrica que reflete a ocupação dos *gateways* VoIP/PBX.

Em relação a um estabelecimento, as chamadas que se completam por um *gateway*, podem ser classificadas como internas ou externas, dependendo de sua origem ser na própria

instituição ou em outra instituição, respectivamente; porém, para classificar a chamada em interna ou externa, consideram-se apenas chamadas que utilizam recursos (canais de voz) do *gateway* de voz, na interface com PBX ou com a RTFC.

É desejável que os algoritmos que serão propostos utilizem alguma métrica, porque sua função será apenas a de indicar o quanto um destino (*gateway*) está disposto a receber chamada. Um balanceamento de chamadas somente precisa atuar frente a chamadas externas destinadas à RTFC, tendo como finalidade distribuí-las entre as diversas instituições conectadas à RTFC de uma cidade. Dessa forma, como cada instituição paga pela chamada encaminhada à RTFC pelo seu *gateway*, o custo envolvido com as chamadas entregues a rede de telefonia fixa de uma cidade não recai sobre um conjunto de instituições apenas.

A métrica apenas precisa refletir o quanto uma empresa está disposta a receber chamadas em um determinado instante. Isto é, os fatores envolvidos no cálculo de uma métrica podem ser modificados dependendo das necessidades que venham a surgir.

A métrica escolhida representa a utilização de um *gateway* (GW_i), que é dada pela razão de chamadas externas em andamento sobre o número de canais de voz não ocupados com chamadas internas, considerando apenas chamadas que passam pelo PBX ou que são destinadas a RTFC. Esta métrica tem similaridade com as que são apresentadas em [6 e 15] e matematicamente ela é dada por

$$U_i = U(GW_i) = \frac{E_i}{(C_i - I_i)} \quad (1)$$

onde o índice i representa uma instituição e o seu serviço VoIP; C_i é a capacidade total dos *gateways* VoIP/PBX da instituição i; E_i é o número instantâneo de chamadas externas

em andamento na instituição i; I_i é o número instantâneo de chamadas internas em andamento na instituição i e U_i é a utilização do *gateway* VoIP/PBX (GW_i) da instituição i.

Na Equação (1) sempre $E_i \leq (C_i - I_i)$ quando todos os canais de voz de um *gateway* VoIP/PBX estiverem ocupados, $C_i = E_i + I_i$ e, neste momento, o seu SV não pode mais receber chamadas. No caso de $C_i = I_i$ e, consequentemente, $E_i = 0$ (que ocorre quando todas as chamadas em andamento são internas), a utilização fica indeterminada e é pressuposto o valor 1 (um).

A Fórmula (1) foi escolhida por representar simples e adequadamente o compromisso voluntário de cada instituição de completar chamadas externas. Caso a utilização fosse calculada por outra fórmula, isso não afetaria os algoritmos em si, mas apenas a prioridade, ou a ordem de recepção das chamadas. Assim, o funcionamento dos algoritmos é transparente à métrica adotada.

2.3 Balanceamento em arquiteturas para encaminhamento E.164

Os mecanismos para alcançar o balanceamento podem se dar através de atrasos para geração da confirmação de chamada (Figura 3a), ou através de envio de confirmação de chamadas, segundo uma ordenação pré-definida (Figura 3b). No segundo mecanismo, a ordenação pode ser determinada por um peso que define a prioridade de um SV e, por simplicidade, pode ser utilizado o termo prioridade para expressar que um SV tem maiores ou menores chances de completar a chamada com sucesso.

Chamadas estabelecidas sem uso do *gateway*, independentemente de terem origem interna ou externa, não afetam o balanceamento, que somente leva em consideração aquelas que fazem uso de recursos do *gateway*. É importante ressaltar que, quando todos os *gateways*

de uma cidade estiverem saturados, sem canais livres, chamadas para a RTFC da mesma não são completadas. Além disso, quando uma instituição opera com vários *gateways* para melhorar o desempenho e atingir uma redundância do serviço VoIP, apenas o total dos canais de voz importa para o balanceamento.

Quando as métricas de utilização dos *gateways* VoIP/PBX não estão centralizadas em uma entidade, o balanceamento é classificado como distribuído. São utilizadas informações pela métrica que mede a ocupação dos canais de voz dos *gateways* e, caso essas informações sejam locais, sua obtenção é mais fácil.

Na Figura 3a existem sinalizações chamadas solicitação, que são enviadas para o SV destino quando o chamador pretende iniciar uma chamada. Então, este servidor, ao receber essa sinalização, envia uma confirmação, também mostrada nesta figura, indicando ao originador da solicitação que ele está apto a encaminhar a chamada para o destino. Por fim, o chamador somente envia a chamada após receber a confirmação da solicitação de chamada.

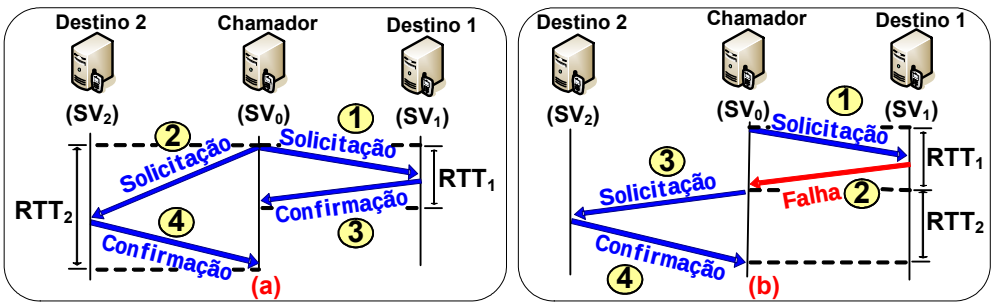


Figura 3. Mecanismos para encaminhamento de chamadas

A Figura 3a mostra como o retardo de rede da primitiva de confirmação pode influenciar no balanceamento de chamadas. Nesta figura, o servidor VoIP chamador inicia uma chamada cujo prefixo destino permite que ela seja encaminhada para o destino final pelos servidores SV_1 ou SV_2, que já são conhecidos de SV_0 e são considerados seus vizinhos. Assim, é enviada uma solicitação de chamada (Figura 3a-1 e Figura 3a-2) para cada um destes

SVs e, logo após esta solicitação ser recebida pelos SVs de destino, eles enviam as respostas de confirmação (Figura 3a-3 e Figura 3a-4).

Os retardos das sinalizações Figura 3a-3 e Figura 3a-1 contribuem para que a confirmação Figura 3a-3 chegue primeiro ao SV_0, indicando que SV_1 receba a chamada. Essa seqüência de eventos é típica, tanto em H.323 como em SIP. Os retardos de ida e volta entre o chamador e os servidores SV_1 e SV_2 são RTT_1 e RTT_2, respectivamente, sendo $RTT_1 < RTT_2$. Dessa forma, se for conhecido o tempo de retardo na rede entre o chamador e os SVs de destino da chamada, o acréscimo de um atraso no encaminhamento da chamada, ou na geração da confirmação, pode ser utilizado para alterar a seleção do SV de destino. Na Figura 3a, se a confirmação do SV_1 (Figura 3a-3) sofrer um atraso maior que ($RTT_1 - RTT_2$), a confirmação (Figura 3a-4) chega antes ao chamador e SV_2 recebe a chamada.

Os atrasos para o recebimento das confirmações no SV chamador não podem ser da ordem de grandeza de minutos, por um motivo muito simples: isso causaria a desistência da chamada por *timeout* da sinalização. Existe a recomendação E.721 da ITU-T, em [16], que sugere alguns atrasos máximos na admissão das chamadas e, apesar de serem aplicados a ISDN, servem como referência para o serviço VoIP.

A Figura 3b mostra a situação em que o SV_0, realiza a consulta das métricas (1) dos servidores de destino da chamada para determinar a ordem preferencial em que estes SVs devem ser contatados. Nesse exemplo, o primeiro a ser contatado é o SV_1 (Figura 3b-1), mas, por algum motivo, é retornada uma indicação de falha (Figura 3b-2). Assim que ela é recebida, o chamador contata o SV de prioridade imediatamente inferior (Figura 3b-3), que nesse exemplo é o SV_2. Esse, ao receber a solicitação (Figura 3b-3), envia uma confirmação, fazendo com que a chamada seja completada por ele. Caso houvesse alguma indicação de falha, em vez da confirmação (Figura 3b-4), um outro SV seria contatado, até que não

existissem mais SVs, ou que um enviasse uma confirmação para o SV_0. Observe que o responsável pela decisão de balanceamento é o chamador, ou um servidor de localização que esteja em modo *proxy* de sinalização. Assim, considera-se como centralizado um algoritmo de balanceamento operando com o mecanismo da Figura 3b, pois um dos servidores mencionados tem a visão dos estados de utilização dos GW_is.

Uma grande diferença entre os dois mecanismos é o atraso para estabelecimento, pois, na Figura 3b, ele pode ser mínimo, quando a chamada é encaminhada logo na primeira tentativa, mas, caso isto não ocorra, pode existir uma demora para o início da chamada. No exemplo dessa figura, a demora foi de pelo menos $RTT_1 + RTT_2$; já no mecanismo da Figura 3a, o atraso sempre é de RTT_i.

Estes dois mecanismos para balanceamento de chamadas mostrados na Figura 3 podem atuar nos ambientes H.323 e SIP, os quais possuem algumas arquiteturas caracterizadas pela configuração dos seus SVs e por suas sinalizações no encaminhamento E.164.

2.3.1 Arquiteturas e encaminhamento E.164 no ambiente H.323

No ambiente H.323 existem algumas arquiteturas que podem ser utilizadas em um serviço VoIP e elas são mostradas nos cenários da Figura 4.

Observe, pela numeração das sinalizações dos cenários da Figura 4, que todas as solicitações referentes a uma mesma chamada chegando aos SVs chamados são enviadas em paralelo; isto é importante para o balanceamento de chamadas baseado no atraso das confirmações, utilizando o mecanismo da Figura 3a. Quando for comentado algo sobre balanceamento baseado em priorização de destinos, o mecanismo da Figura 3b é o utilizado, independentemente das numerações das sinalizações de solicitação.

Na Figura 4b, os GKs (SVs) formam uma arquitetura completamente entrelaçada, onde todos eles são vizinhos, ou seja, em suas bases de dados existem as informações dos IPs e prefixos E.164 de todos os GKs do serviço VoIP. Desta forma, eles podem encaminhar as chamadas diretamente entre si, sem que haja uma entidade central para coordenar isso. Um ponto positivo dessa arquitetura é a ausência de um ponto de falha central.

Na Figura 4a é mostrada uma estrutura hierárquica com um DGK, que centraliza os prefixos E.164 e endereços IP dos GKs integrantes do serviço VoIP. Nessa arquitetura, apenas o DGK é vizinho de todos os GKs e estes, por sua vez, confiam que qualquer vizinho dele é seu vizinho também. Assim, qualquer GK desse serviço VoIP pode encaminhar chamadas para outro GK vizinho do DGK. Apesar de o DGK ser um ponto central de falhas, essa arquitetura é muito comum por utilizar o ponto central apenas para o processo de localização.

Por fim, a Figura 4c também mostra uma arquitetura hierárquica como a Figura 4a, porém, neste caso, o DGK atua como *proxy* de sinalização (apenas para o processo de localização de vizinhos).

Nessas arquiteturas da Figura 4, as sinalizações possuem uma numeração, cujo valor indica a ordem em que elas foram geradas, ou simplesmente os instantes de geração. Essas sinalizações possuem um nome em português, que representa a função realizada por elas nos cenários apresentados e, além disso, entre parênteses, as siglas LRQ (*Location Request*) e LCF (*Location Confirm*) representam respectivamente as sinalizações de solicitação e confirmação no H.323 [7 e 8].

Quando um GK recebe um LRQ, existem três passos para encaminhar a chamada: primeiro, o SV verifica se o usuário chamado está registrado nele e se não estiver, ele verifica se a chamada pode ser encaminhada para um *gateway* e, por fim, o terceiro passo é repassar o

LRQ para um vizinho. Somente se nenhum dos vizinhos puder receber a chamada é que uma resposta de rejeição ao seu estabelecimento é enviada para o chamador.

Nas arquiteturas da Figura 4, quando aparece a imagem de um banco de dados ligado a um SV por uma seta, significa que o SV em questão tem os registros dos prefixos E.164 e os endereços IPs dos SVs que fazem parte do serviço VoIP. Assim, estes SVs são considerados vizinhos entre si.

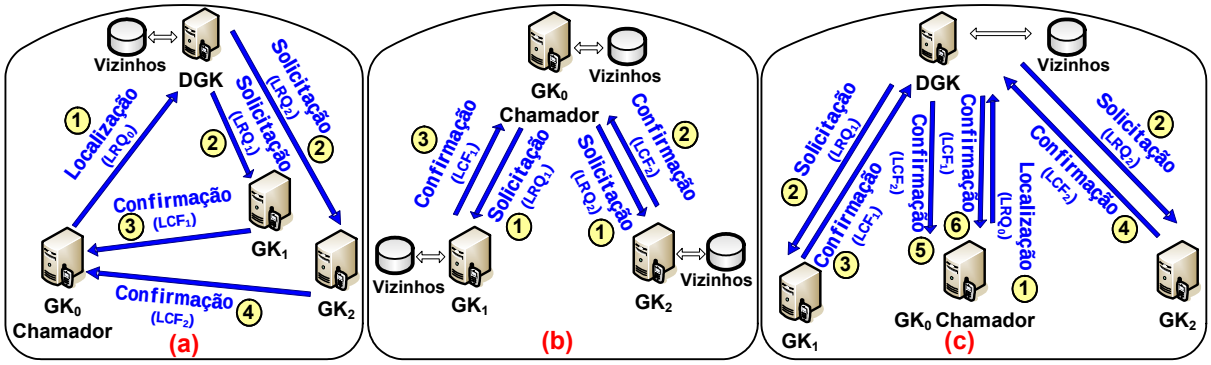


Figura 4. Arquiteturas para encaminhamento E.164 no ambiente H.323

Na Figura 4a existe um DGK com o banco de dados contendo os endereços IP de vizinhos e os seus prefixos E.164. Quando um chamador (GK_0) realiza uma chamada para um usuário de outro GK, uma solicitação de chamada é enviada a esse DGK (Figura 4a-1), o qual localiza dois GKs para encaminhamento dessa chamada. Em seguida, o servidor de localização repassa a solicitação de chamada para os GKs 1 e 2 (Figura 4a-2). Como o DGK não atua como *proxy* de sinalização e mídia, as confirmações Figura 4a-3 e Figura 4a-4 são enviadas diretamente para o chamador (GK_0), o que indica que a chamada pode ser encaminhada pelos GK_1 ou GK_2. O chamador pode iniciar a chamada, a princípio, por qualquer um deles, mas o eleito é aquele cuja primitiva LCF chega primeiro ao chamador.

Nesse cenário, utilizando um algoritmo distribuído, atrasos às primitivas LCF seriam aplicados, como no mecanismo mostrado na Figura 3a. Assim, em termos de implementação, os GKs das instituições precisariam sofrer modificação no código fonte, o que não é muito

desejável, porque o *software* pode ser solução proprietária ou heterogênea. Quando a solução é proprietária, o código fonte é inacessível, inviabilizando alterações; na possibilidade dos GKs terem *software* heterogêneos, a manutenção e atualização são dificultadas.

Por outro lado, caso um controle centralizado utilizando o mecanismo da Figura 3b fosse implementado no cenário da Figura 4a, as métricas (1) seriam mantidas no DGK e, assim, o mecanismo mostrado na Figura 3b estaria em operação. Ao utilizar o controle centralizado, as primitivas LRQs (Figura 4a-2) de um GK pouco prioritário seriam retidas, sendo enviadas somente quando fossem retornadas falhas referentes às primitivas LRQ dos SVs mais prioritários, ou seja, o GK com menor utilização (1) dos seus canais de voz seria contatado primeiro. Observe que o DGK não participa da sinalização, ele só a encaminha; assim, o DGK não receberia as sinalizações de confirmação (Figura 4a-3 e Figura 4a-4) que são enviadas diretamente para o chamador. Por causa disso, nesse cenário, a solução viável seria a de gerar atrasos (Figura 3a).

Já na Figura 4b não existe um DGK, o que implica que todos os GKs precisam ter uma base de dados com os prefixos e endereços IP de todos os vizinhos. Nesse caso, uma chamada é enviada do chamador diretamente para o chamado, sem intermediários. Por outro lado, se um GK fizer parte de outro serviço VoIP, dependendo do plano de numeração, os dois serviços podem encaminhar algumas chamadas entre si, sem que seus administradores desejem.

Nesse cenário, tanto soluções distribuídas como centralizadas necessitariam que todos os GKs sofressem alteração de código. Isso impactaria a escalabilidade, porque podem existir soluções de GKs proprietárias ou heterogêneas.

Para finalizar a discussão sobre encaminhamento E.164 no H.323, falta mencionar o cenário da Figura 4c. Este é semelhante ao da Figura 4a, mas o DGK atua como *proxy* de

sinalização no processo de localização de vizinhos. Assim, uma solução centralizada seria viável, ao contrário da Figura 4a. Apesar de um número maior de primitivas passarem pelo DGK, se comparado à Figura 4a, não há problemas de escalabilidade porque o processo de localização de vizinhos envolve poucas primitivas, que são enviadas apenas antes da inicialização da chamada.

Apesar de tanto soluções centralizadas como distribuídas poderem ser utilizadas, as centralizadas implicariam apenas em alteração de código em um ponto único (DGK). Como as soluções distribídas exigiriam alterações de código em outros pontos além do DGK, as centralizadas seriam mais viáveis. Os mecanismos utilizando atrasos (Figura 3a), ou prioridades dos SVs (Figura 3b), poderiam ser viáveis, porque o DGK como *proxy* de sinalização envia a solicitação (LRQ) e pode receber as suas confirmações (LCF); em suma, o servidor de localização tem o controle sobre a sinalização, em vez de ser apenas um coadjuvante, como na Figura 3a.

2.3.2 Arquiteturas e encaminhamento E.164 no ambiente SIP

O SIP é um protocolo VoIP e, com ele, os destinos são resolvidos através do DNS (*Domain Name Service*), utilizando domínios (por ex. fulano@dominio.com.br). Quando o destino é caracterizado pelo número E.164, é necessário montar uma estrutura para mapear o endereço E.164 em número IP ou nome de domínio e, para isto, diversas alternativas de implementar encaminhamento E.164 além do DNS foram analisadas, como uso de tabela estendida em um banco de dados, o módulo LCR (*Least Cost Routing*) [17, 18 e 19] e o protocolo OSP (*Open Settlement Protocol*) [20, 21, 22 e 23]; mas o DNS privado com ENUM (*E.164 NUmber Mapping*) [24, 25, 26 e 27] mostrou-se mais escalável, robusto, simples e de fácil integração.

O ENUM permite que, através do prefixo E.164, um ou mais servidores de destino possam ser encontrados, assim no ambiente SIP passa a existir uma entidade chamada DSER, similar ao DGK no H.323. O DSER consulta o DNS privado, para que os destinos de uma chamada sejam localizados pelo processo de localização, o qual aponta os SVs que possuem entre os seus prefixos um classificado como mais significativo.

Apenas o DSER tem acesso ao DNS privado, outra entidade conectada na rede (Internet) não tem permissão para consultá-lo diretamente.

Na Figura 5, as sinalizações são apresentadas com seu nome de acordo com a função desempenhada; já os nomes que estão entre parênteses, seguem a especificação do protocolo SIP [4].

Nessa figura há dois cenários para o encaminhamnto E.164 no SIP, sendo que na Figura 5a existe uma estrutura hierárquica com o DSER e, na outra, todos os *proxies* SIP são vizinhos entre si, como uma rede do tipo *mesh*.

Inicialmente, o DSER recebe um pedido de localização (Figura 5a-1), e para saber quais são os SVs de destino, ele realiza uma consulta de DNS aos registros NAPTR (*Naming Authority Pointer*) (Figura 5b-2), os quais possuem informações como prefixos E.164 e endereços IP dos servidores que pertencem a um serviço. Para cada prefixo pode existir a associação de um ou mais servidores registrados na base de dados do DNS e, quando uma requisição chega ao DNS, ele localiza quem pode encaminhá-la e repassa isto ao DSER. Assim, em seguida, o DSER pode contactar os SVs de destino, ou repassar esta informação para o chamador (Figura 5a-3).

A opção do DSER contactar diretamente os SVs destino, atuando como *proxy* de sinalização, não é considerada. Isso porque, ao contrário do H.323, o *proxy* SIP OpenSER não

permite que apenas as sinalizações do processo de localização passem por ele. Assim, se o DSER fosse um *proxy* de sinalização, haveria um ponto de falha central durante todo o andamento da chamada.

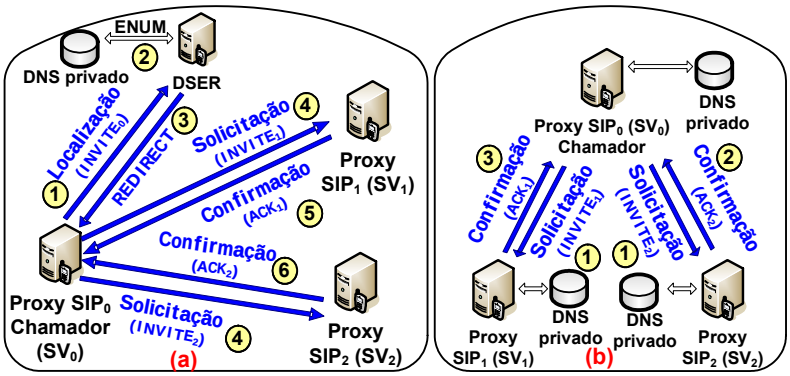


Figura 5. Arquiteturas para encaminhamento E.164 no ambiente SIP

No cenário da Figura 5a, o *proxy* SIP chamador (SV_0) inicia uma chamada e precisa saber quem são os SVs destino. Para isso, um pedido de localização é enviado ao DSER (Figura 5a-1) que, ao recebê-lo faz uma consulta ENUM ao DNS privado (Figura 5a-2), passando a conhecer quem são os possíveis destinos da chamada pretendida. Como ele não atua como intermediário entre o chamador e os chamados, uma sinalização de redirecionamento é retornada ao SV chamador (Figura 5a-3) que, ao recebê-la, pode iniciar a chamada com as primitivas de solicitação de chamada (Figura 5a-4). Essa primitiva de redirecionamento informa a ordem em que cada SV deve ser contatado, que é definida por um campo com as prioridades dos SVs das instituições, conforme Figura 3b. Finalizando, uma confirmação à solicitação de chamada é enviada ao chamador por cada um dos SVs destino (Figura 5a-5 e Figura 5a-6).

As prioridades citadas na sinalização de redirecionamento podem ser definidas de várias formas, por exemplo, pela métrica (1). Para que isso ocorra, o DSER precisa ter acesso ao grau de utilização dos GWs conectados aos SVs, o que pode ser viabilizado de duas formas: o DSER poderia consultar os *gateways* das instituições, ou receber esta informação

das instituições com atualizações freqüentes. Em termos de implementação no cenário da Figura 5a, para que as prioridades sejam modificadas em função da métrica, um módulo para o *proxy* SIP (OpenSER) foi desenvolvido. Assim, os SVs com menor utilização são contatados primeiro.

Nesse caso, uma solução centralizada seria viável; ela utilizaria o mecanismo da Figura 3b, e apenas o DSER sofreria a modificação de código. Como os SVs não seriam modificados, essa solução seria escalável, ao contrário de uma solução distribuída. Nesta última, todos os SVs precisariam sofrer modificações no código fonte, o que é necessário para definir prioridades aos SVs, em função de uma métrica. Isso é inviável quando os SVs são proprietários, ou de difícil manutenção (vários *software* diferentes utilizados como *proxy* SIP em um serviço).

No cenário da Figura 5b, as informações contidas no DNS privado seriam compartilhadas com os *proxies* SIP, não havendo um DSER para centralizar o processo de localização de vizinhos. Tanto em um balanceamento centralizado como em um distribuído, todos os SVs precisariam ser contemplados com um módulo extra, o qual seria de difícil manutenção e atualização, em um cenário com SVs de fabricantes diferentes, além, é claro, deste módulo não poder ser utilizado por SVs proprietários.

2.4 Determinação de atrasos

Na arquitetura de um *backbone* de abrangência nacional é comum que, em cada cidade, exista um POP, onde as redes das instituições desta localidade se conectam. Assim, podemos dizer que as redes das instituições da cidade *A* estão conectadas ao POP desta cidade. Em resumo, como o POP é um ponto de convergência de tráfego, quando uma sinalização é enviada para

qualquer servidor em uma cidade diferente daquela do chamador, ela, obrigatoriamente, passa pelo POP da cidade do servidor destino.

Para alguns dos algoritmos de balanceamento propostos, será necessário conhecer os retardos de propagação das primitivas VoIP entre um servidor e o POP, o que pode ser obtido através de medições. A seguir, são apresentadas formas de medir estes retardos no ambiente VoIP, sendo consideradas apenas medidas ativas [28].

Essas medições podem ser realizadas por sondas, que são o envio de ICMP (*Internet Control Message Protocol*) para aferir o retardo de ida e volta entre dois pontos. Esta estimativa do retardo de ida e volta, RTT (*Round Trip Time*), é uma informação necessária para alguns dos algoritmos aqui propostos.

Caso os erros das medições sejam na faixa de unidades de milissegundos, os algoritmos podem apresentar erros admissíveis, porém uma análise mais detalhada precisa ser realizada para erros de uma ordem de grandeza maior.

Caso fosse utilizada medição unidirecional, uma sincronização dos relógios dos equipamentos seria necessário, o que pode ser viabilizado com o uso de GPS (*Global Positioning System*) ou NTP (*Network Time Protocol*) [29 e 30]. Além disso, alguns problemas como *offset*, atualizações e *skew* [29 e 31] proporcionam erros às medidas unidirecionais e precisam ser tratados. Apesar de essas medidas serem utilizadas em várias aplicações, elas são mais difíceis de serem obtidas [28] do que as medidas de retardo, considerando o percurso de ida e volta, as quais realmente importam para o balanceamento.

Os medidores para envio das sondas podem estar em três locais diferentes: no servidor de localização, no POP, ou nos servidores chamados (servidores das instituições).

A primeira forma é utilizar um medidor no servidor de localização, que realizaria as medidas de retardo até os SVs, utilizando PING e, num segundo passo, enviaria para estes SVs as medidas obtidas em campos de extensão do usuário nas mensagens de solicitação de chamadas. A vantagem deste procedimento é que o servidor de localização tem conhecimento completo dos IPs e prefixos de todos os servidores, além de processar as mensagens de solicitação, tornando simples a implementação desta solução. A desvantagem é que esse servidor pode estar muito distante, em termos de retardo, do POP correspondente e as medidas podem ter uma ordem de grandeza bem maior que a diferença entre os atrasos dos SVs aos POPs, induzindo ao erro, se a variância de retardo na rede for grande.

Algumas das arquiteturas mostradas na seção anterior não podem utilizar essa forma de medir os retardos, porque elas não possuem um servidor de localização. Estas arquiteturas são mostradas nas Figura 4b e Figura 5b.

A segunda forma é utilizar um medidor nos POPs, que poderia obter o endereço IP e os prefixos dos SVs de uma região no servidor de localização. Depois seria necessário realizar as medições, utilizando PING e, num segundo passo, enviar para os SVs as medidas obtidas. A vantagem desta solução seria a maior precisão na medida do atraso do POP aos SVs, quando comparada com a anterior. A desvantagem é que o repasse das informações teria que ser feito fora da sinalização H.323/SIP, possivelmente usando a porta de gerência do SV, ou uma aplicação sendo executada no SV destino, o que tem implicações de segurança para acesso privilegiado. Essa segunda forma permite que qualquer uma das arquiteturas da Figura 4 e Figura 5 possa ser utilizada.

Uma terceira forma seria tentar realizar as medidas a partir de cada SV_i. Inicialmente, a relação dos SV_i da região é obtida no servidor de localização. A partir desse momento, cada SV_i de uma região utilizaria o PING, para medir o seu atraso até os outros SV_i da mesma,

assumindo que a rota entre SVs passe pelo POP. As medidas deveriam ser obtidas nos vários SVs em um mesmo intervalo de tempo, para evitar erros grosseiros, o que é uma desvantagem e um complicador. Esta forma pode ser utilizada por qualquer uma das arquiteturas da Figura 4 e Figura 5.

Nas três formas de realizar as medições o ICMP foi citado como uma sonda; porém, na sinalização VoIP, o RTCP (*Real-Time Transport Control Protocol*) poderia realizar este papel. Para isto, servidores VoIP localizados próximos ao DGK/DSER ou próximos ao POP da cidade seriam utilizados, para manter chamadas ativas com os SVs das instituições e, nesta forma de medição, *software* simulariam usuários registrados nos servidores VoIP envolvidos nas medições.

Existe uma quarta forma, que é viável apenas no H.323 e, por causa disso, não foi incluída entre as três mencionadas anteriormente. Ela é uma medida passiva [28], envolvendo a própria sinalização entre o DGK e o *gatekeeper* que pretende iniciar uma chamada destinada à RTFC. Assim que um GK_i envia um pedido de localização para o DGK cuja chamada é destinada a RTFC da sua cidade, uma mensagem chamada RIP (*Request in Progress*) é enviada do DGK para o *gatekeeper* de onde partem o LRQ. Dessa forma, esse GK_i conhece o RTT entre ele e o DGK, o qual será subtraído do tempo que o LCF de cada um dos outros GK_i levar para chegar, obtendo, assim, o tempo de ida e volta de cada GK_i até o POP. Esse procedimento seria falho se houvesse rotas fortemente assimétricas no *backbone*, mas isso não é o caso hoje em dia. Os retardos da arquitetura da Figura 4b não podem ser medidos utilizando a sinalização VoIP, pois nela não existe o DGK, ao contrário das outras duas da Figura 4.

As medições de RTT entre o POP e os SVs precisam ser realizadas ao longo do tempo, se a variância do atraso da rede for grande. Todavia, esta medida para as instituições do

fone@RNP mostra uma variância pequena, dada a alta disponibilidade de banda e baixa utilização dos enlaces da conexão das instituições ao POP. Com esta alta velocidade e baixa utilização, a formação de filas é bastante pequena nos equipamentos de redes, causando baixa variação de retardo neste trecho. Além disso, dentro das instituições, os servidores VoIP devem estar em locais privilegiados, onde a saturação e congestionamento são baixos.

Capítulo 3

ALGORITMO DE BALANCEAMENTO DISTRIBUÍDO UTILIZANDO PONTO DE REFERÊNCIA VIRTUAL

Nesta seção é apresentado um algoritmo de balanceamento de chamadas VoIP distribuído, que é baseado em atrasos aplicados às primitivas de sinalização, conforme mostrado na Figura 3a. A métrica a ser utilizada é a da Equação (1), que calcula a ocupação dos canais de voz de um *gateway* (utilização de um *gateway*).

Com base na ocupação desses canais, calculada pela Equação (1), as primitivas são atrasadas, para que um balanceamento de chamadas seja alcançado. Quando as chamadas externas destinadas à RTFC são enviadas para uma cidade, vários servidores de destinos podem encaminhá-la, porém um conjunto de servidores tende a receber a maioria ou todas elas. Dessa forma, o balanceamento atua para que as chamadas sejam distribuídas entre todos os servidores e seus *gateways*, sendo desejável que isto seja realizado com justeza e não exista chamada não completada, enquanto existir algum canal livre, em algum destes *gateways* VoIP/PBX. Além disso, é desejável que o balanceamento seja aderente aos ambientes SIP e H.323.

Neste algoritmo, será utilizado um ponto de referência, através do qual obtêm-se os atrasos necessários para o balanceamento, explorando funções analíticas. Por simplicidade, no texto o algoritmo é chamado de algoritmo com Servidor VoIP Virtual (SV_v).

Foi desenvolvido um protótipo (Apêndice A) para realizar simulações executadas no *software* Matlab, assim o funcionamento do algoritmo pôde ser analisado.

Na seção 3.1, o algoritmo é apresentado. Na seção 3.2, um servidor de referência (servidor VoIP virtual) é utilizado para o balanceamento. Na seção 3.3, apresenta-se o cálculo de posicionamento deste servidor, a condição mais restritiva para sua posição e o atraso máximo gerado pelo algoritmo com este servidor. Por fim, na seção 3.4, um exemplo do funcionamento deste algoritmo para um cenário é descrito.

3.1 Introdução ao algoritmo utilizando atrasos

No balanceamento de chamadas externas serão utilizados atrasos na sinalização VoIP (**atraso adicional**), sendo, para isso, utilizada a métrica calculada por (1). Ela leva em consideração a quantidade dos canais livres e ocupados de um *gateway VoIP/PBX* nos seus cálculos.

O balanceamento de chamadas que será proposto pode ser alcançado adicionando atrasos, que podem ser utilizados para o envio das confirmações da Figura 4, para o ambiente H.323 e nas confirmações da Figura 5, referente ao ambiente SIP.

Esse atraso é necessário porque a primeira confirmação que chega ao chamador pertence ao SV de destino escolhido para encaminhá-la, como mostrado no mecanismo para encaminhamento de chamadas da Figura 3a. Assim, um atraso na confirmação de chamada pode dar oportunidade de um outro servidor encaminhar a chamada. Outras confirmações que cheguem após a primeira não são consideradas, então os SVs delas não têm chance de encaminhar a chamada.

A idéia geral do algoritmo é que, ao diminuir a quantidade de canais livres de um *gateway* associado a um SV, este deve atrasar um pouco mais o envio das confirmações, para

atenuar a sua probabilidade de ser escolhido para receber a chamada, evitando assim congestionar ainda mais o seu *gateway VoIP/PBX*.

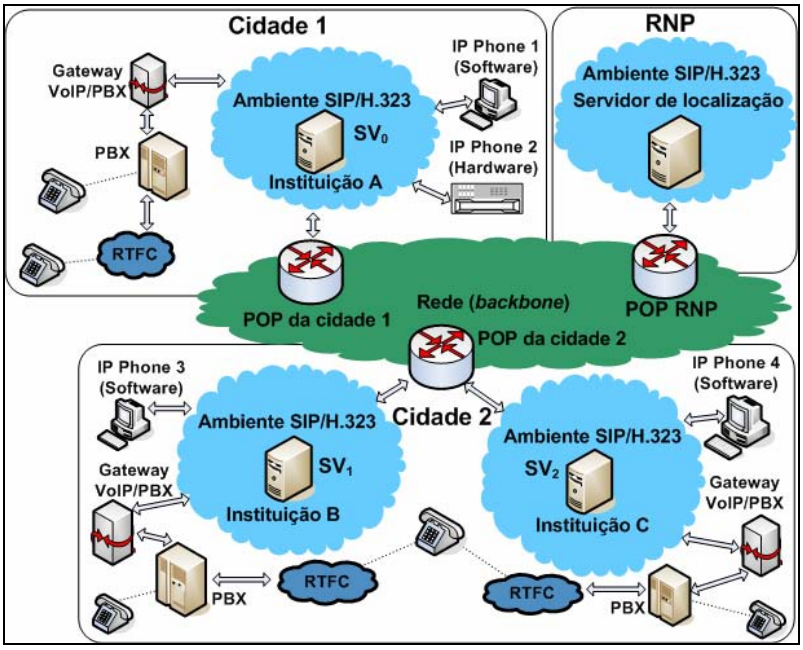


Figura 6. Exemplo de um cenário VoIP com H.323/SIP

Esse algoritmo, além de tomar decisões descentralizadas, utiliza apenas informações locais, portanto, quando uma instituição toma uma decisão, o estado instantâneo das outras não é consultado.

Um exemplo da topologia de um serviço VoIP é mostrado na Figura 6, onde as instituições conectadas ao POP são capazes de encaminhar chamadas VoIP para a RTFC de uma cidade, via PBX. Uma chamada originada na cidade 1 e destinada ao código de área da cidade 2 é roteada até o POP da cidade 2 e depois encaminhada a um dos servidores VoIP existentes, o qual irá repassá-la ao *gateway* e este, por sua vez, ao PBX.

Esses POPs são pontos de convergência do tráfego, onde todas as redes das instituições de uma cidade se conectam. Em função disso, o tráfego envolvendo estas instituições obrigatoriamente passa por este ponto. Essa é uma configuração típica em *backbones*; por exemplo, as instituições do serviço VoIP da RNP, que é chamado fone@RNP,

possuem esta disposição. Além disso, as instituições da Figura 6 podem ter tempos de propagação (retardos) entre elas e seus POPs diferentes.

A Figura 6 mostra um cenário onde esse algoritmo proposto pode atuar, onde existem os serviços VoIP de três instituições e um servidor de localização. Os SVs mostrados podem ser GKs ou *proxies* SIP, pois para entender o funcionamento do algoritmo não é relevante distinguí-los.

É definido que nesta figura todas as instituições têm uma mesma capacidade C, o SV_0 é o chamador e qualquer SV com um índice subscrito maior que zero é considerado um servidor chamado. Além disso, considera-se que na cidade 2 existem *n* instituições que podem receber chamadas oriundas de alguma instituição da cidade 1; onde o *i* subscrito define uma instituição, tal que $0 \leq i \leq n$. O $i = 0$ refere-se ao chamador e $n = 1$ significa que apenas um SV pode encaminhá-la e, por outro lado, $1 < i \leq n$ diz que vários SVs podem encaminhar uma chamada.

O incremento do índice *i* ocorre em função do retardo de propagação de ida e volta de uma primitiva entre um SV_i e o POP da sua cidade, sendo este retardo definido pela sigla RTT e quanto maior o índice *i* maior o RTT medido entre um SV e o POP da cidade; em outras palavras, quanto maior este índice mais afastado um SV está do seu POP. Desta forma, as siglas SVs e RTTs recebem os índices apropriados. Sendo assim, os SV_i e SV_{i+1} possuem seus RTT_is ,tal que, $RTT_i \leq RTT_{i+1}$. Pelo que foi exposto, na Figura 6, teriamos SV_1 e SV_2 na cidade 2 e pressupõe-se, para o funcionamento do algoritmo, que $RTT_1 < RTT_2$. Além do índice *i*, pode ser utilizado o *j*, pressupondo-se que $j > i, \forall\ 2 \leq j \leq n$.

Um SV_i, ao receber uma solicitação de chamada do SV_0, envia uma confirmação que será definida como $ConfSol_i$ (Figura 3, Figura 4 e Figura 5).

A diferença de retardo de ida e volta (RTT) do SV_i e do SV_j medido em relação ao ponto de convergência (POP), é dada por $RTT(j,i) = | RTT_j - RTT_i |$, onde $j > i$; $ConfSol_i$ é a confirmação de solicitação emitida por SV_i; U_i é a utilização do *gateway* do SV_i dada por (1) e $D_i(U_i)$ é o atraso para envio da $ConfSol_i$ em função de U_i.

A equação (1) será simplificada com a quantidade de chamadas internas igual a zero ($I_i = 0$); assim ela se restringe a

$$U_i = \frac{E_i}{C} \quad (2)$$

observe que a capacidade dos *gateway* é considerada nesta fórmula igual para todos os SV_is e, por isso, C no denominador é utilizado em vez de C_i.

Dado os servidores chamados SV_i e SV_j, $\forall\, i > 1 \text{ e } j > i$, que estão em uma mesma cidade e o servidor chamador (SV_0) em outra, existe um ponto de convergência para o tráfego envolvendo SV_i e SV_j. O $POP_{i,j}$ é este ponto de convergência de tráfego da cidade de SV_i e SV_j e é dito que, caso as confirmações referentes a uma mesma chamada sejam enviadas por SV_i e SV_j passarem pelo $POP_{i,j}$ ao mesmo instante, elas tendem a chegar juntas ao chamador, então é considerado que estes SVs estão alinhados. Como existem retardos causados pelo tratamento de pacotes nos roteadores, elas não chegam ao mesmo instante no servidor chamador.

3.2 Algoritmo utilizando um ponto de referência virtual (com SV_v)

Para descrever o funcionamento do algoritmo na Figura 6, pressupõe-se que na cidade 2 existem vários SVs, apesar de ela mostrar apenas dois. Estes SVs possuem condições iniciais: a mesma capacidade *C*, como já foi mencionado; todos os canais de voz dos *gateways* estão

desocupados, ou seja, $U_i = 0$ e $U_j = 0$; SV_i e SV_j com $RTT_j - RTT_i > 0$. A métrica utilizada será a utilização dos canais de voz de um *gateway*, calculada pela Fórmula (2); as chamadas são iniciadas pelo SV_0 (chamador) com destino à cidade 2 para os SV_i e SV_j (chamados), sendo que apenas um deles poderá encaminhá-la.

Quando o SV_0 enviar a primeira solicitação, os SVs da cidade 2 possuem ocupação de canais igual a zero, assim a utilização calculada através de (2) é zero, o que implica em um atraso de zero segundos para o envio da $ConfSol_i$. Isto define a primeira condição de funcionamento do algoritmo; portanto, um *gateway* sem canais de voz ocupados não gera atrasos para envio da confirmação ($ConfSol_i$).

O segundo ponto é definido quando o SV_i atinge $U_i = A < 1$ e nele a sinalização $ConfSol_i$ é atrasada de um valor $D_i(A) = RTT(i,n)$, o que faz o SV_i se alinhar com o SV_n, ou seja, o SV_n é utilizado como ponto de referência. Este valor *A* deve ser o mesmo para todos os SVs, a princípio. Todavia, caso uma instituição resolva utilizar um valor acima deste especificado, ela poderá fazê-lo, mostrando estar propensa a receber uma porcentagem maior de ligações externas.

O valor de D_i para alinhamento com um SV_n foi apresentado, podendo ser entendido com ajuda da Figura 7, que é um diagrama de sinalização do cenário da Figura 6. Nela observa-se que os SVs da cidade 2 são contatados através de uma solicitação de chamada (Figura 7-1 e Figura 7-2); contudo, apenas o tempo de propagação das solicitações (Figura 7-2) é considerado, justamente por serem enviadas simultaneamente pelo servidor de localização (SL).

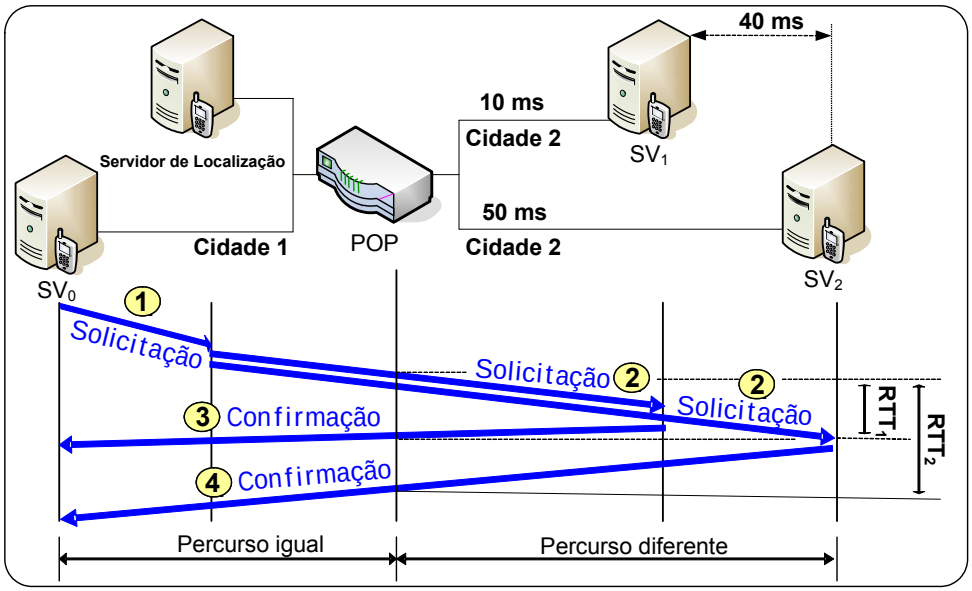


Figura 7. Sinalizações e seus atrasos

Existe na Figura 7 o trecho chamado "percurso igual", onde qualquer sinalização envolvendo o chamador e os servidores em outra cidade possuem o mesmo retardo de propagação, porque passa pelo mesmo caminho na rede. Este trecho não precisa ser considerado e o ponto de referência para as medições passa a ser o POP_{12}; assim, a medida RTT_1 e RTT_2 corresponde a $RTT_{(i,POP)} = RTT_i$, ou seja, o tempo de ida e volta de uma sinalização entre os SV_is e o POP.

Utilizando os dois pontos apresentados, é possível construir um gráfico, o qual faz a correlação de atrasos impingidos à primitiva de confirmação e a utilização dos canais de voz de um *gateway*. Para obter esses atrasos no cenário da Figura 7, é utilizado o gráfico mostrado na Figura 8.

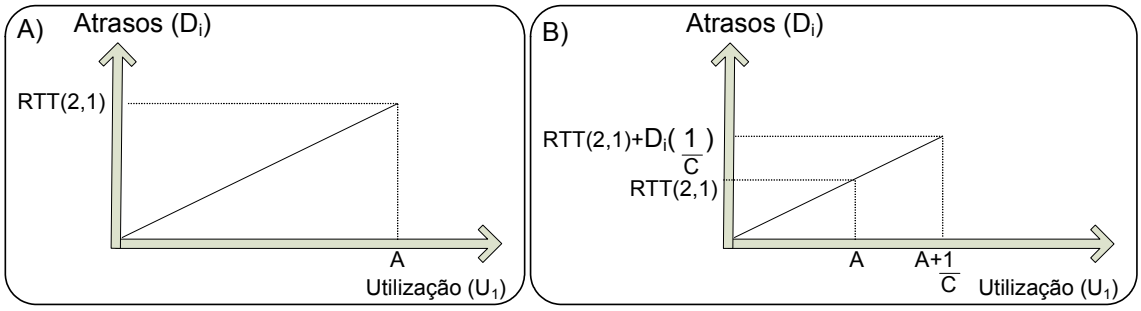


Figura 8.Gráficos D_i versus utilização para obter os atrasos até U_i=A

Na Figura 8a, quando o SV_1 não tem chamada em andamento, sua utilização é zero (2). O segundo ponto desse gráfico é a condição de alinhamento já descrita e através destes dois pontos, uma reta linear é traçada; outras curvas poderiam ser usadas, mas optamos pela variação linear, por simplicidade. Assim, o algoritmo calcula D_i, segundo a fórmula

$$D_i(u_i) = \frac{RTT(n,i).U_i}{A} \quad (3)$$

fazendo com que o atraso varie linearmente com a utilização. Esta dependência linear é suficiente e adequada, como será mostrado. Pressupõe-se inicialmente que a utilização do SV_1 e SV_2 é igual a zero na Figura 8a e, em seguida o SV_1 começa a receber chamadas até ele se alinhar com o segundo, que é quando o SV_1 atinge a utilização *A*. Logo após este momento, o SV_2 pode começar a receber chamada e não gera atrasos para as suas primitivas de confirmação, porque ele é o último SV e não possui um servidor como referência, para utilizar a Fórmula (3). Assim, quando o SV_1 receber mais uma chamada, como na Figura 8b em $U_1 = A + 1/C$, ele terá um atraso maior do que o utilizado para alinhá-lo com o último (SV_2 ou SV_n) e, como conseqüência, este SV irá receber todas as chamadas restantes. Na Figura 8b, o ponto $U_1 = A + 1/C$ significa que o SV_1 recebeu uma chamada depois de ter se alinhado ao último, pois, pela Fórmula (2), ao receber uma chamada externa, a utilização é incrementada de $U_i = 1/C$.

A solução encontrada para impedir que o SV_n receba todas as chamadas após alinhamento e saturação, foi conceber um SV_v (Servidor VoIP virtual), que está distante do POP de RTT_v, tal que $RTT_v > RTT_n$. Assim, um SV_v é utilizado como a referência mais distante, em vez do SV_n; por causa disso, esse algoritmo passa a ser chamado de **algoritmo com SV_v**. A fórmula (3) foi reescrita, substituindo o índice *n* (último SV) por *v* de SV_v , resultando em

$$D_i(u_i) = \frac{RTT(v,i).U_i}{A} \quad (4)$$

fazendo com que o atraso varie linearmente, como mostrado na Figura 8, onde, somente agora o SV_v é utilizado como referência.

Quando a utilização obtida por (3) for 1, o atraso sofrido será de $RTT(n,i)/A$ de acordo com (2), como indicado no gráfico da Figura 9. Isto significa que os gráficos da Figura 8 continuam além do ponto $U_i = A$, indo até a utilização alcançar o valor máximo $U_i = 1$.

O valor de *A* não deve ser demasiadamente pequeno, a ponto de provocar um valor exagerado para o atraso, quando U=1, que é dado por $RTT(i,n)/A$ (Figura 8 e Figura 9). Todavia, como nos cenários reais, como por exemplo, o serviço VoIP da RNP, a diferença de retardo entre servidores é da ordem de milissegundos, a escolha de *A* não deve ser crítica. Empiricamente, espera-se que *A* seja escolhido entre 20% e 50%. Caso o valor de *A* seja pequeno, o atraso obtido pela Equação (4) pode ser execessivo; já um valor de *A* grande implica em um alinhamento quando a utilização encontra-se próximo ao valor de saturamento.

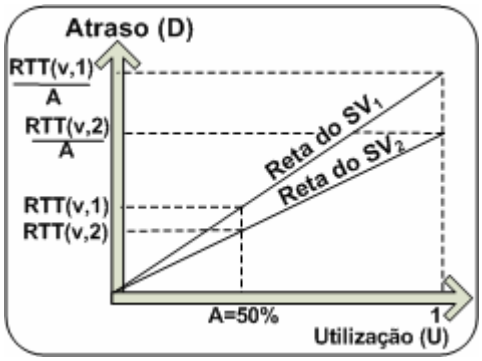


Figura 9. Gráficos do atraso D_i versus utilização

A Figura 9 mostra os gráficos da Figura 8 utilizando o SV_v como referência e, além disso, observe que o valor de *A* adotado arbitrariamente é de 50%, que é o ponto para o

alinhamento dos SV_1 e SV_2 com o SV_v. O maior atraso obtido nesse gráfico é alcançado com a ocupação total dos canais de voz de um *gateway* (GW_i), ou seja, $U_i = 1$.

Na Figura 9, enquanto $U_i < A$, os servidores não estão alinhados com o virtual. Quando as utilizações dos SVs alcançam o valor A, eles se alinham ao SV_v.

3.3 Posicionamento do Servidor VoIP Virtual (SV_v)

O posicionamento do servidor virtual não pode ser arbitrário e, para garantir o balanceamento, é preciso determinar a sua posição mínima, que será mostrada a seguir, através de desenvolvimentos matemáticos.

No cenário da Figura 7a, a equação de alinhamento do SV_1 e SV_2 precisa considerar o atraso sofrido pelas suas sinalizações, constituído pelas parcelas: atraso gerado pelo algoritmo, calculado através de (4), e o retardo, provenientes da propagação da sinalização na rede. Para que dois SVs *i* e *j* estejam alinhados, é necessário que

$$\begin{array}{c} \text{Atraso do } SV_i \;=\; \text{retardo entre o } SV_i \text{ e } SV_j \;+\; \text{Atraso do } SV_j \,,\; \forall\; j > i \\ D_{iv}(u_i) = RTT(i,j) + D_{jv}(u_j) \end{array} \quad (5),$$

os índices *i* e *j*, onde $j > i$, são genéricos e podem ser aplicados a qualquer um dos *N* SVs, porém, no cenário da Figura 7, eles seriam, respectivamente, os SVs 1 e 2. Nessa equação, $D_{iv}(u_i)$ é o atraso obtido através da fórmula (4), onde o SV_i tem o SV_v como referência; o $D_{jv}(u_j)$ é calculado através de (4), onde o SV_j tem o SV_v como referência; por fim, $RTT(i,j)$ é a diferença de retardo do SV_i e SV_j, tomando o POP como referência.

Se, na equação de alinhamento (5), o atraso do SV_i for menor do que o segundo membro, o SV_i recebe a chamada. Isso, a partir de (5), é escrito matematicamente como

$$\text{Atraso do } SV_i < \text{retardo entre o } SV_i \text{ e } SV_j + \text{Atrasos do } SV_j \text{ , } \forall\, j > i$$
$$D_{iv}(u_i) < RTT(i,j) + D_{jv}(u_j) \quad (6).$$

Caso o SV_j receba a chamada, a situação oposta acontece, então

$$\text{Atraso do } SV_i > \text{retardo entre o } SV_i \text{ e } SV_j + \text{Atrasos do } SV_j \text{ , } \forall\, j > i$$
$$D_{iv}(u_i) > RTT(i,j) + D_{jv}(u_j) \quad (7).$$

As equações (6) e (7) são chamadas de equações de alinhamento e são semanticamente iguais a (5).

3.3.1 Condições para o posicionamento do SV_v

Nesta seção, a fim de deduzir o posicionamento do SV_v é utilizada a condição mais restritiva para o balanceamento. Para isso, o posicionamento do SV_v deve permitir que os *n* SVs alcancem a ocupação máxima ($U = 1$).

Como o retardo na rede oscila devido ao tráfego, existem os valores limites $RTT(i,j)_{min}$ e $RTT(i,j)_{max}$ para o retardo entre dois servidores. A variação de retardo da rede (*jitter*) é definida em função da máxima variação entre pares de servidores e dada por

$$\Delta_{rede} = \text{máximo } \{RTT(j,i)_{max} - RTT(j,i)_{min}\}, \forall i,j \in \{1, \ldots, n\} \text{ , } j > i\} \quad (8),$$

nesta fórmula, quando é subscrito *max* e *min* significa que o valor utilizado pela variável são os máximo e mínimo, respectivamente, dentre os conhecidos.

Dependendo do posicionamento do SV_v, quando alguns SVs chegarem à utilização máxima ($U = 1$), outros com canais livres podem não receber mais chamadas, porque as confirmações dos SVs saturados chegam antes ao chamador. Isso precisa ser evitado, com o correto posicionamento do SV_v.

Para determinar o posicionamento do SV_v é necessário que todos os SVs ultrapassem a utilização *A* e apenas um SV possua uma chamada a menos do que os outros. A situação que foi descrita permite a existência de duas condições iniciais para dedução das fórmulas que regem o posicionamento do SV_v. Na primeira, o SV_i é aquele que tem uma chamada a menos, e na segunda, o SV_j é o que tem uma chamada a menos. Para essas duas condições, deseja-se que, sempre o SV com uma chamada a menos receba a próxima chamada, o que é definido como a condição de maior restrição.

Quando pretende-se considerar o pior retardo de rede em um seguimento, temos:

$$RTT(i,j)_{max} \leq RTT(i,j) + \Delta_{rede} \quad (9)$$

- **Condição inicial um:**

As condições iniciais são dadas por:

$$U_i(t_0) = U_j(t_0)\ , \qquad U_i(t_1) = U_i(t_0) \qquad e \qquad U_j(t_1) = U_j(t_0) + \frac{1}{C_j}$$

Nesta situação, U_i e U_j possuem a mesma ocupação em um tempo t_0 e apenas SV_j ganha uma chamada em um instante t_1.

Em (6), temos $D_{iv}(u_i) < RTT(i,j) + D_{jv}(u_j)$, em seguida as condições iniciais e a fórmula (4) são aplicadas:

$$\frac{U_i(t_0)RTT(i,v)}{A} < RTT(i,j) + \frac{(U_i(t_0) + 1/C_j)}{A} RTT(j,v)$$

Pressupondo $C = C_i = C_j$, temos:

$$U_i(t_0) \times RTT(i,v) < A \times RTT(i,j) + U_i(t_0) \times RTT(j,v) + \frac{RTT(j,v)}{C}$$

$$C \times U_i(t_0) \times RTT(i,v) < C \times A \times RTT(i,j) + C \times U_i(t_0) \times RTT(j,v) + RTT(j,v)$$

$$C \times U_i(t_0) \times \left[RTT(i,v) - RTT(j,v)\right] < C \times A \times RTT(i,j) + RTT(j,v)$$

$$C \times U_i(t_0) \times RTT(i,j) < C \times A \times RTT(i,j) + RTT(j,v)$$

$$C \times \left[U_i(t_0) - A\right] \times RTT(i,j) < RTT(j,v)$$

Esta inequação pode alcançar a sua maior restrição, se e somente se, $i = 1$ e $j = n$ em função da variável $RTT(1,n)$, assim ela pode ser modificada e reescrita como

$$RTT(n,v) > C \times \left[U_1(t_0) - A\right] \times RTT(1,n) \quad (10).$$

Em (10), a condição mais crítica ocorre quando $U_1(t_0) = 1 - \frac{1}{C}$, assim o SV_1 está com uma chamada a menos do que os outros. Ou seja, com exceção do primeiro, todos os SVs estão com seus *gateways* saturados. Após os SVs de um serviço VoIP estarem com esta ocupação, é desejável que o SV_1 receba a próxima chamada.

Aplicando $U_1(t_0) = 1 - \frac{L}{C}$ em (10), onde *L* é o número de canais de voz não ocupados, temos:

$$RTT(n,v) > C \times \left[1 - \frac{L}{C} - A\right] \times RTT(1,n)$$

$$RTT(n,v) > \left[C(1 - A) - L\right] \times RTT(1,n) \Rightarrow \quad L = 1 \text{ (situação mais crítica, apenas um canal desocupado)}$$

Aplicando (9) para considerar o pior caso:

$$RTT(n, v) > \left[C(1 - A) - 1\right] \times RTT(1, n)_{max} \quad (11)$$

A razão para haver a restrição (11), relacionada ao posicionamento do SV_v, pode ser entendida, ao observar a curva da Figura 9, após o alinhamento com o SV_v, que sempre beneficia o SV mais próximo do POP, ou seja, em outras palavras, o mais afastado do servidor virtual. Para cada chamada aceita nas curvas dos SVs, Figura 9, observa-se que o SV_1 tende a aplicar um atraso proporcionalmente maior a sua primitiva de confirmação em comparação aos outros servidores com igual utilização, evitando o recebimento de chamada.

Caso (11) não seja satisfeita em uma topologia, os servidores mais distantes do POP, a partir do alinhamento, receberão preferencialmente as chamadas, em detrimento dos servidores mais próximos. Uma saturação crítica ocorreria, se o servidor virtual estivesse tão próximo que, embora o servidor mais distante estivesse com utilização $U_n(t_0) = \frac{C-1}{C}$ e o servidor mais próximo estivesse com $U_1(t_0) < U_n(t_0)$, ainda assim a chamada seria encaminhada para o servidor *n*, no instante t_0, porque o atraso gerado pelo servidor 1 é maior que o gerado pelo servidor *n*. A inequação (10) previne que isto aconteça, forçando um posicionamento mínimo para o SV_v.

Antes de obter (11), *L* poderia ser zero, isto retrataria o caso de todos os canais de um *gateway* estarem ocupados ao receber uma solicitação de chamada externa.

- **Condição inicial dois:**

$$U_i(t_0) = U_j(t_0)\ , \qquad U_j(t_1) = U_j(t_0) \qquad e \qquad U_i(t_1) = U_i(t_0) + \frac{1}{C_i}$$

Nesta situação, U_i e U_j possuem a mesma ocupação em um tempo t_0 e depois no instante seguinte t_1, SV_i ganha uma chamada adicional. Agora queremos estabelecer a condição para que o SV_j receba a próxima chamada. Esta condição pode ser expressa como:

$$D_{iv}(t_1) > RTT(i,j) + D_{jv}(t_1)$$

Em (7), temos $D_{iv}(t_1) > RTT(i,j) + D_{jv}(t_1)$, em seguida, as condições iniciais e a fórmula (4) são aplicadas:

$$\frac{U_i(t_0)RTT(i,v)}{A} > RTT(i,j) + \frac{U_j(t_0)RTT(j,v)}{A}$$

$$\frac{\left[U_i(t_0) + \frac{1}{C_i}\right] \times RTT(i,v)}{A} > RTT(i,j) + \frac{U_j(t_0) \times RTT(j,v)}{A}$$

Pressupondo $C = C_i$, temos:

$$C \times U_i(t_0) \times RTT(i,v) + RTT(i,v) > A \times C \times RTT(i,j) + C \times U_j(t_0) \times RTT(j,v) \quad \Rightarrow temos: U_i(t_0) = U_j(t_0)$$

$$C \times U_i(t_0) \times [RTT(i,v) - RTT(j,v)] + RTT(i,v) > A \times C \times RTT(i,j)$$

$$C \times U_i(t_0) \times [RTT(i,j)] + RTT(i,v) > A \times C \times RTT(i,j)$$

$$RTT(i,v) > [A - U_i(t_0)] \times C \times RTT(i,j) \quad (12)$$

A inequação (12) apresenta uma restrição menos crítica do que a (11), porque, quando os SVs estiverem próximos da saturação, com certeza $U_i(t_0) > A$, e o segundo membro sempre será negativo, assim o $RTT(i,v)$ sempre atenderá a inequação, para todo $RTT(i,v) > RTT(j,v)$.

3.3.2 Atraso adicional

O maior atraso adicional gerado pelo algoritmo será ocasionado pelo servidor 1 e será dado quando $U_1 = 1 - 1/C$, isto é, a última chamada será aceita pela estação 1, estando todos os outros servidores com todos os canais já ocupados. Para calcular este atraso, é utilizada a fórmula (4).

$$D_1(1-\frac{1}{C}) = \frac{\left(1-\frac{1}{C}\right)\times \mathrm{RTT}(1,v)}{A} = \frac{(C-1)\times \mathrm{RTT}(1,v)}{C\times A} \quad (13)$$

Este valor será máximo para o menor valor de A, que é $1/C$. Ele pode não ser utilizado na prática, pois geraria um atraso máximo elevadíssimo.

O $\mathrm{RTT}(1,v)$ pode ser escrito como $\mathrm{RTT}(1,v) = \mathrm{RTT}(1,n) + \mathrm{RTT}(n,v)$, substituindo (11) neste $\mathrm{RTT}(n,v)$ de (11), temos

$$\mathrm{RTT}(1,v) = \mathrm{RTT}(1,n) + \left[C(1-A)-1\right]\times \mathrm{RTT}(1,n)$$

$$\mathrm{RTT}(1,v) = \left[C(1-A)\right]\times \mathrm{RTT}(1,n)$$

Utilizando o valor mínimo de A, que é $A = \frac{1}{C}$, assim

$$\mathrm{RTT}(1,v) = \left[C(1-\frac{1}{C})\right]\times \mathrm{RTT}(1,n) = (C-1)\times \mathrm{RTT}(1,n) \quad (14)$$

Substituindo isto em (13):

$$D_1(1-\frac{1}{C}) = \frac{(C-1)\times(C-1)\times \mathrm{RTT}(1,n)}{C\times A} = \frac{(C-1)\times(C-1)\times \mathrm{RTT}(1,n)}{C\times\left(\frac{1}{C}\right)}$$

$$D_1(1-\frac{1}{C}) = (C-1)^2 \times RTT(1,n) \quad (15)$$

Em (15) é apresentado maior atraso adicional, mas fazendo $C >> 1$, e $RTT(1,n)_{max} = RTT(1,n) + \Delta_{rede}$ obtido a partir de (9), ele tende a

$$\text{Atraso adicional máximo} = D_{1,max} = C^2 \times RTT(1,n)_{max} \quad (16).$$

Num cenário onde a capacidade é igual a 30 canais e $RTT(1,n) = 10ms$ o D_{max} seria de 9 segundos. Observe que este algoritmo pode produzir alguns atrasos adicionais máximos excessivos em alguns cenários.

3.4 Exemplo do funcionamento do algoritmo em um cenário

A Figura 10a apresenta dois SVs com atrasos bem variados ao POP e esse cenário é utilizado para explicar a operação do algoritmo. Esses SVs têm capacidade de 4 canais de voz, sendo inicialmente o número de chamadas externas e internas iguais a zero, ou seja, sem atraso para envio das confirmações de solicitação de chamadas. Neste exemplo, será pressuposto A = 50%, para todos os SVs. O posicionamento do servidor VoIP virtual é 50 ms de retardo do POP e, ele foi escolhido para simplificar os cálculos no decorrer da explicação, em função dos outros valores adotados no cenário da Figura 10.

Na Figura 10b existe o gráfico com as retas de atraso versus utilização dos SVs. No texto que se segue, a utilização de um SV é designada como U_i; por exemplo, a utilização do SV_1 é U_1. Nos gráficos existem indicações de pontos (valores) nas retas, que são apontados pelo cruzamento das linhas pontilhadas perpendiculares aos eixos vertical e horizontal.

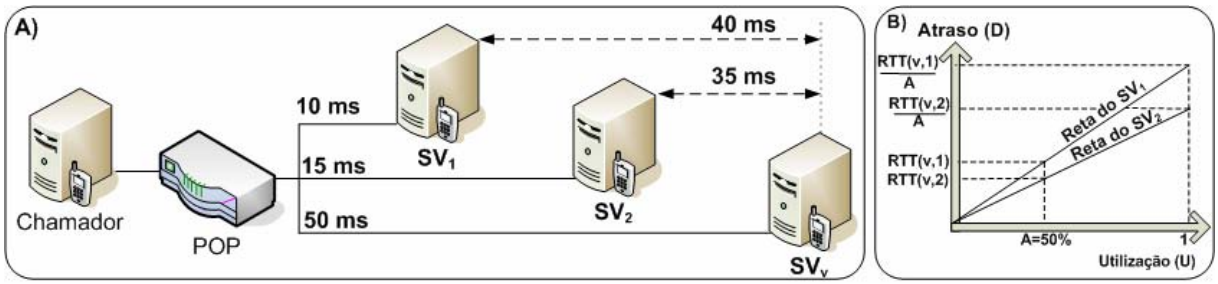


Figura 10. Cenário

Quando se tenta estabelecer a primeira chamada, uma solicitação de chamada é recebida por cada um dos dois SVs e aquela destinada ao SV_1 chega primeiro, pois o outro SV está mais distante do POP e do SV chamador. Como conseqüência, a chamada é enviada para o SV_1, pois sua confirmação ($ConfSol_1$) foi a primeira a chegar ao chamador.

Após esta primeira chamada, U_1 será de ¼ e a sua próxima confirmação será atrasada de 40 ms; isto é mostrado no gráfico da Figura 11a como $\frac{RTT(v,1)}{2}$. Quando ocorre uma segunda chamada externa, a $ConfSol_2$ será enviada primeiro e chegará ao POP antes que a outra, fazendo com que a chamada seja encaminhada para SV_2 e não para SV_1, como ocorreria na ausência do algoritmo. A utilização do SV_2, que era de zero, passará a ser de ¼, a próxima confirmação do SV_2 enviada por ele será atrasada de 35 ms. Isso é mostrado no gráfico da Figura 11b como $\frac{RTT(v,2)}{2}$.

A solicitação de uma terceira chamada externa chega primeiro no SV_1, que atrasa o envio da confirmação em 40 ms, sendo que existem duas parcelas de 10 ms (retardo entre o POP e o SV) que são consideradas, totalizando 60 ms. Uma parcela referente ao percurso da solicitação até o SV_1 e a outra referente a confirmação que é enviada por este servidor. O envio da $ConfSol_2$ sofrerá um atraso de 35 ms adicionais as duas parcelas de 15 ms (retardo entre o POP e o SV), totalizando 65 ms. Uma parcela é pertinente a solicitação direcionada ao SV_2 e outra ligada a confirmação enviada pelo segundo servidor.

Isso fará com que a confirmação do segundo SV seja enviada ao mesmo tempo daquela pertencente ao primeiro; porém, como SV_1 está mais próximo do POP, a sua confirmação chegará 5 ms antes ao chamador, fazendo com que o SV_1 receba sua segunda chamada. Com ela, a utilização deste SV pela fórmula (1) será de ½ (Figura 11c), assim o envio da confirmação de outra chamada será atrasada de 80 ms, conforme mostrado na Figura 11c, e neste momento o SV_1 e SV_v estarão alinhados (A=50%).

A solicitação de uma quarta chamada externa chegará antes no SV_1 e o envio da sua ConfSol será atrasada de 80 ms. Quando o SV_2 receber a solicitação desta quarta chamada, 5 ms depois do SV_1, o envio da sua confirmação será atrasado de apenas 35 ms e chegará ao chamador com 35 ms de folga em relação a $ConfSol_1$. Assim, esta quarta chamada será enviada para o SV_2, cuja utilização passa a ser de ½, com um atraso para envio da próxima confirmação de 70 ms, como mostrado na Figura 11d.

Quando a quinta chamada chegar, as confirmações do SV_1 serão atrasadas de 80 ms e a do SV_2 de 70 ms; assim, todos os SVs estão alinhados agora, ou seja, $U_i = A$. As confirmações dessa chamada chegam ao mesmo instante ao SV chamador. Mas, existem retardos ocasionados pela propagação na rede e pelo tratamento de pacotes em equipamentos; assim, na prática, elas chegarão aproximadamente ao mesmo tempo no chamador, mas uma delas tem que chegar primeiro, resultando que o seu servidor recebe a chamada.

Se SV_1 receber a chamada, ele passa a ter uma utilização de ¾, o que força o atraso para envio da confirmação a ser de 120 ms. No caso de SV_2 receber a chamada, o atraso será de 105 ms. Será arbitrado que o SV_1 receba essa chamada (Figura 12a). Assim, a sexta confirmação enviada pelo SV_2 chega ao chamador primeiro e ele recebe esta chamada. O atraso desta sinalização será de 105 ms, como mostrado na Figura 12b.

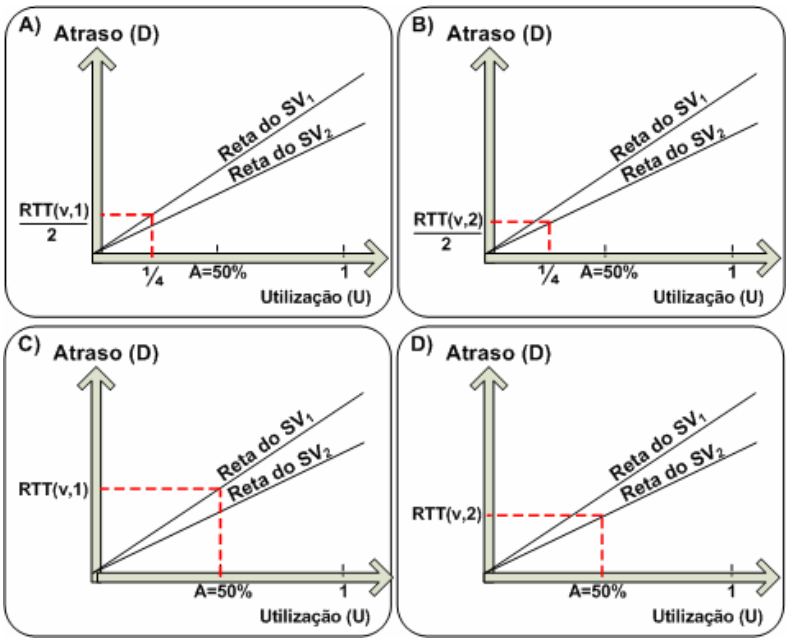


Figura 11. Gráficos até os SVs se alinharem

Ambos SVs estão com $U_i = 3/4$, e a sétima chamada é encaminhada para o SV_2, que aplica um atraso de 105 ms à confirmação de solicitação na Figura 12c; assim o seu *gateway* chega à ocupação máxima dos canais de voz.

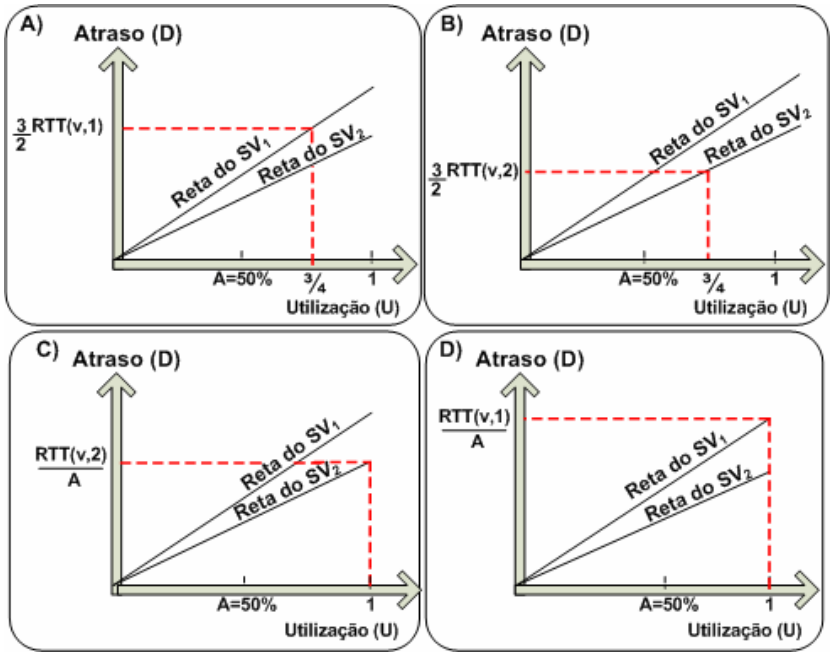


Figura 12. Gráficos após os SVs se alinharem

A confirmação da oitava chamada sofre um atraso de 120 ms do SV_1 e a enviada para o SV_2 tem um atraso de 140 ms; assim, a chamada é encaminhada para o SV_1 (Figura 12d). Após encaminhar esta chamada, os dois SVs estão com todos os canais de voz ocupados.

Como os *gateways* dos SVs estão saturados, qualquer chamada externa enviada para a RTFC da cidade deles não será completada. Observe que, ao utilizar o algoritmo as chamadas são encaminhadas para a RTFC, todos os canais em todos os *gateways* chegam a ser ocupados, o que não é verdade na operação sem o algoritmo de balanceamento de chamadas. Caso o algoritmo não fosse utilizado, as chamadas seriam entregues para o primeiro servidor, porque ele está mais próximo do POP. Observe que ele somente poderia receber quatro chamadas; em função disto, os outros ficariam com os seus canais desocupados.

3.5 Comentários

Como foi visto, neste algoritmo utilizando ponto de referência virtual, há uma dependência de conhecer o RTT entre os servidores VoIP de uma cidade e seus POPs que pode variar. Em função do panorama atual dos *backbones* e dos enlaces, possíveis erros nas medições do RTT em funções das variações de retardo não prejudicam muito o algoritmo por causa do posicionamento do SV_v.

Um ponto a ser considerado é o posicionamento do servidor virtual, que em alguns cenários pode levar a um atraso adicional máximo elevado. Em virtude disto, pequenos valores de *jitter* afetam pouco o balanceamento.

Capítulo 4

ALGORITMO DE BALANCEAMENTO DISTRIBUÍDO OTIMIZADO

O algoritmo descrito no capítulo anterior fazia uso de um SV_v, a fim de balancear as chamadas externas destinadas à RTFC; mas, através de cálculos, verifica-se que o atraso gerado com a utilização do SV_v é excessivo na subseção 3.3.2. Assim, será mostrado que, após a utilização dos SVs atingir o valor *A*, existe um atraso mínimo para o balanceamento acontecer.

Quando o SV virtual é utilizado, a reta que determina o atraso do envio da confirmação não se altera, após a utilização *A*, como na Figura 9. Isto causa um atraso adicional excessivo, o que foi mostrado no capítulo anterior, quando a utilização do *gateway* do primeiro SV chega a (C-1) chamadas. Uma outra abordagem é explorada neste capítulo, onde, após os servidores se alinharem, basta compensar os *jitters* da rede, no percurso entre o POP e o servidor VoIP; assim, após o balanceamento dos SVs, esse algoritmo gera atrasos para envio de confirmações menores do que o algoritmo com ponto de referência virtual.

A abordagem concebida para o algoritmo otimizado (sem servidor virtual) a partir do momento que um SV alcance a utilização *A*, é mostrada a seguir. O algoritmo continua com as mesmas características, que são: ser distribuído, atuar de forma descentralizada para tomada de decisão, baseado apenas em informações locais para tomar decisões, utilizar o mecanismo Figura 3a e operar no cenário da Figura 6, onde existem POPs.

Para validar o funcionamento do algoritmo, foi desenvolvido um protótipo (Apêndice A) para realizar simulações executadas no *software* Matlab.

Na seção 4.1 este algoritmo otimizado, sem um uso do SV_v, é apresentado; na segunda seção, o seu atraso máximo é deduzido. Na seção 4.3, o atraso máximo gerado por este algoritmo e pelo algoritmo com SV_v são comparados. A última seção apresenta o funcionamento deste algoritmo, utilizando um cenário.

4.1 Algoritmo otimizado

Esse algoritmo tem seu funcionamento semelhante ao algoritmo com SV_v, até a utilização de um servidor atingir A, sendo que o ponto de referência é o SV_n e não o virtual. Como o SV_n é o servidor de referência, os atrasos são gerados em função de $RTT(i,n)$; exceto este detalhe, o algoritmo não possui alterações até o ponto em que os SVs estão alinhados com o SV_n. Nesse momento, os SVs, exceto o SV_n, estarão com as suas utilizações igual a A.

Observe que, ao utilizar ao utilizar o último servidor como referência, em vez do virtual, tendo em vista preliminarmente apenas o intervalo $0 \leq U_i < A$, os atrasos gerados pelo algoritmo otimizado já são menores do que o anterior. Tendo em vista isso, a admissão da chamada sofre um atraso menor.

Após os SVs se alinharem com o último, o SV_n envia as confirmações com atraso zero, o que determina que ele acaba recebendo as chamadas. Apenas depois do SV_n atingir a utilização de alinhamento ($U=A$), é que ele pode impingir atrasos às suas confirmações e, somente então, os outros servidores retornam a disputa pela preferência do encaminhamento de chamada, utilizando atrasos para envio de confirmações.

Será deduzido o atraso mínimo para garantir o balanceamento, que todos os servidores devem observar após alinhamento com o último ocorrido em função da utilização.

Para encontrar o atraso mínimo, uma condição inicial precisa ser examinada, onde os *gateways* dos SVs estão com chamadas em andamento suficientes para atingir a utilização igual a A e, além destas chamadas, o SV_i possui $(L-1)$ chamadas externas e os outros SVs estão com L chamadas. Assim, quando os *gateways* estão com esta quantidade de chamadas externas em andamento, a utilização do SV_i é de $A+\frac{L-1}{C}$ e os outros servidores VoIP possuem utilização igual a $A+\frac{L}{C}$.

$$U_i = A+\left(\frac{L-1}{C}\right) = A+(L-1)\gamma \quad ,onde\ \gamma = \frac{1}{C} é\ o\ incremento\ da\ utilização$$

$$U_j = A+\left(\frac{L}{C_j}\right) = A+L\gamma \quad ,onde\ \gamma\ é\ o\ incremento\ da\ utilização$$

Utilizando estes U_i e U_j, é possível deduzir o atraso mínimo para que o balanceamento aconteça, dado que ele é aplicado após um SV atingir o valor A de utilização. Na Figura 13, o algoritmo mostrado no capítulo anterior funciona até o ponto P para dois SVs, o qual define que eles estão alinhados com um terceiro SV_n. Este último servidor, SV_n, pode receber chamadas apenas quando os outros atingirem o ponto P. Neste momento, atrasos, D_n, não são impingidos para o envio da confirmação até o *gateway* do SV_n atingir a utilização A.

Em resumo, as chamadas são balanceadas até os *gateways* dos SVs 1 a $(n-1)$ alcançarem $U = A$, em seguida as chamadas são enviadas para o SV_n. Quando o *gateway* deste SV também estiver com $U = A$, ele começa a balancear chamadas com os

outros e, a partir deste momento, os *n* SVs utilizam apenas um atraso mínimo (γ) por chamada para o balanceamento, o que é mostrado na Figura 14.

Pressupondo que todas as chamadas já encaminhadas continuam ativas, quando o *gateway* do SV_n estiver com essa utilização, $U = A$, todos os SVs utilizarão um atraso mínimo (γ - incremento mínimo para o atraso) para balancear as chamadas; assim, cada confirmação gerada recebe o atraso para alinhamento com o último SV, mais um atraso γ por chamadas pertencentes ao intervalo $A < U \leq 1$.

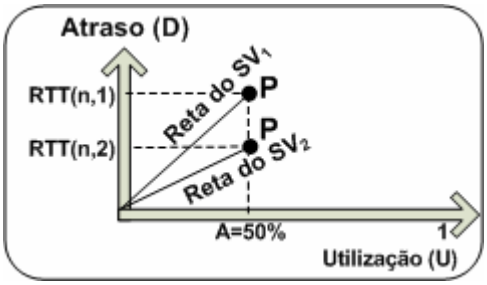


Figura 13. Balanceamento até o ponto *P*, envolvendo SV_1 e SV_2

Para encontrar o atraso mínimo, é considerado um cenário com o SV_i e SV_j, onde os seus *gateways* estão com chamadas suficientes para estarem alinhados. Além destas chamadas, o SV_i possui (*L*-1) chamadas e o SV_j *L* chamadas. Como o SV_i tem uma chamada a menos, deseja-se que ele receba a próxima chamada, para ficar também com *L* chamadas. Assim, o somatório dos atrasos e retardos da solicitação e da confirmação do SV_i precisam ser menores do que as enviadas por um SV_j. A pior situação para o balanceamento tem que ser mensurada; para isto, o *jitter* da rede calculado pela Equação (8), que envolve os caminhos entre os SVs chamados (SV_i) e o POP, precisa ser considerado. Sendo assim, caso o *jitter* máximo no percurso SV_i-POP, mais o atraso gerado pelo algoritmo em SV_i, não sejam menores do que o atraso gerado pelo algoritmo em SV_j, a chamada não é encaminhada via GW_i. O que foi mencionado neste parágrafo é matematicamente representado por (17):

$$RTT(i,n) + (L-1)\gamma + \Delta_{rede} < \quad RTT(j,n) + (L)\gamma + RTT(i,j) \quad (17)$$

$$\mathrm{RTT(i,n)} - \mathrm{RTT(j,n)} + (\mathrm{L}-1)\gamma + \Delta_{\mathbf{rede}} < (\mathrm{L})\gamma + \mathrm{RTT(i,j)}$$

$$\mathrm{RTT(i,j)} + (L-1)\gamma + \Delta_{rede} < (L)\gamma + \mathrm{RTT(i,j)}$$

$$(L-1)\gamma + \Delta_{rede} < (L)\gamma$$

$$\gamma < -\Delta_{rede}$$

$$\gamma > \Delta_{rede} \quad (18).$$

Em (18), verifica-se a condição necessária para o balanceamento acontecer, que é o atraso da confirmação compensar o maior *jitter* da rede no percurso SV_i-POP. Assim, na Figura 14, tanto o SV_1 como o SV_2, a partir do ponto *P* para cada chamada já aceita, a confirmação recebe um atraso γ, além é claro, do atraso para o alinhamento destes SVs. Após os SV_1 e SV_2 se alinharem ao SV_n, as chamadas são encaminhadas para o SV_n, até seu *gateway* chegar à utilização igual a *A*, mostrado pelo ponto *P* da Figura 14. Em seguida, quando isto acontece, todos os SVs começam a balancear chamadas entre si, até os seus *gateways* chegarem à saturação.

A Figura 14 é semelhante à Figura 13, porém ela tem a reta que mostra o atraso imposto pelo algoritmo, quando a utilização compreende o intervalo $A \leq \mathrm{U} \leq 1$.

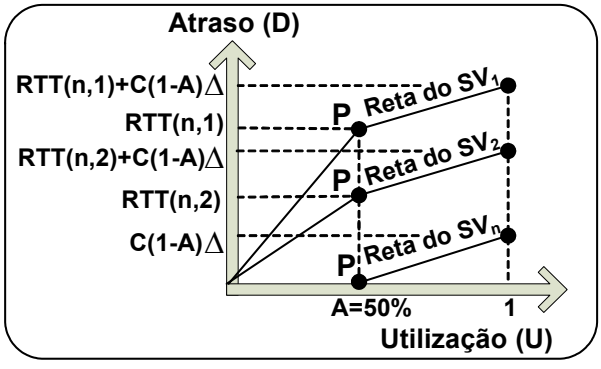


Figura 14. Balanceamento além do ponto *P*, envolvendo SV_1, SV_2 e SV_n

4.2 Atraso adicional

O atraso adicional ($D_{i,max}$), que será mostrado a seguir, acontece quando SV_i está com todos os canais de voz do seu *gateway* ocupados. Além disso, lembre que o SV_1 gera o maior atraso, porque $RTT_{(1,n)} > RTT_{(i,n)}$ para $i > 1$. Nas fórmulas matemáticas obtidas a seguir, (9) é utilizado, assim o maior retardo num seguimento é considerado.

$$D_{i,max} = (\text{parcela até } U_i = A) + (\text{parcela para } U_i > A)$$

$$D_{i,max} = RTT_{(i,n)_{max}} + C(1-A)\Delta_{rede} \quad (19)$$

Em (19), temos o maior atraso adicional para um dado valor de *A*, quando todos os canais de voz estão ocupados, mas pode ser utilizado $A = 1/C$ e *i*=1, para obter o maior atraso adicional possível.

Com $C >> 1$, o valor do atraso adicional em (19) pode ser levado ao seu limite máximo

$$D_{1,max} = RTT_{(1,n)_{max}} + (C-1)\Delta \approx RTT_{(1,n)_{max}} + C\Delta_{rede} \quad (20)$$

Na Figura 14, o maior atraso de um SV é mostrado como $RTT_{(n,i)} + C(1-A)\Delta_{rede}$, mas a última sinalização de confirmação é enviada quando a ocupação do *gateway* é de $(C-1)$ chamadas; assim, o maior atraso gerado pelo algoritmo será de $RTT_{(n,i)} + [C(1-A)-1]\Delta_{rede}$. Utilizando valor de *i*=1 para que $RTT_{(n,i)}$ seja máximo, temos o atraso adicional máximo ($D^*_{1,max}$), dado por

$$D^*_{1,max} = RTT_{(1,n)_{max}} + [C(1-A)-1]\Delta_{rede} \quad (21)$$

Utilizando o menor valor possível para *A*, que é $A = 1/C$ e em seguida $C >> 2$, considerando o atraso adicional máximo se aproximar do seu maior valor dado por

$$D^{*}_{1,max} = RTT(1,n)_{max} + [C-2]\Delta \approx RTT(1,n) + C\Delta_{rede} \quad (22)$$

Em (21), temos o atraso adicional máximo para ocupação de (*C*-1) chamadas, porém um valor de *A* qualquer pode ser utilizado. Já em (22), o atraso adicional máximo é calculado para o menor valor de *A*.

Quando for necessário calcular o atraso do SV_n, o valor de RTT(i,n) será zero, nas fórmulas apresentadas nesta subseção, para o cálculo de $D_{n,max}$ e $D^{*}_{n,max}$.

4.3 Comparação dos atrasos

O $D^{*}_{1,max}$ é mostrado em(22), quando o algoritmo não utiliza SV_v, podendo ser comparado com o atraso do algoritmo utilizando SV_v em (16).

Em(22):

$$\text{Atraso adicional máximo} \approx RTT(1,n)_{max} + C\Delta_{rede}$$

Em (16):

$$\text{Atraso adicional máximo} \approx C^{2} \times RTT(1,n)_{max}$$

É possível observar que o atraso adicional máximo em (22), quando comparado com o algoritmo utilizando SV, é menor. Isso é sempre verdade em um ambiente real, porque os *jitters* encontrados atualmente não possuem uma ordem de grandeza suficiente para **(16)** ser maior.

4.4 Exemplo do funcionamento do algoritmo em um cenário

Este tópico tem como objetivo explicar o funcionamento do algoritmo proposto neste capítulo, que não utiliza servidor virtual (SV_v). Para isso, utiliza-se o mesmo cenário do capítulo anterior, mostrado na Figura 15a. Nesta figura, o SV_v aparece riscado com um " x", porque ele não é utilizado; ao lado dela, a Figura 15b mostra o gráfico atraso versus utilização, para o algoritmo otimizado, com *A* igual a ½.

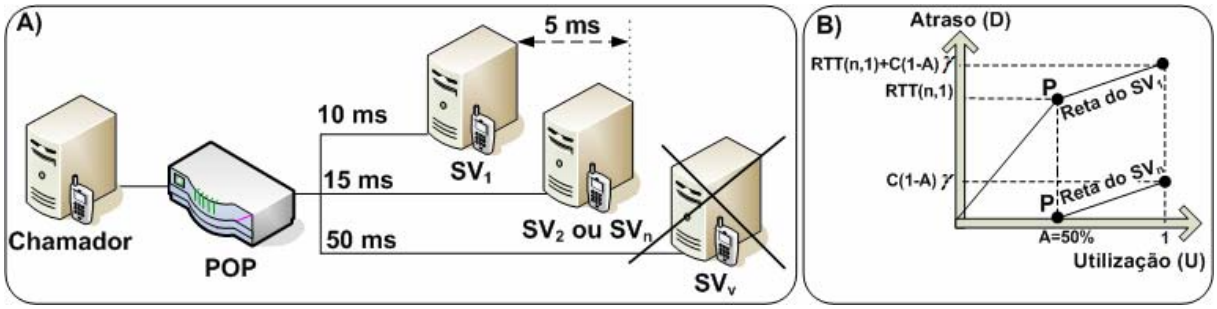


Figura 15. Cenário para o algoritmo otimizado

Inicialmente os *gateways* dos SVs não possuem chamadas em andamento; assim, as confirmações ($ConfSol_i$) da primeira chamada não sofrem atraso. Nos gráficos seguintes, são retratadas as situações descritas no texto, que são marcados pelo cruzamento das retas pontilhadas, cruzando perpendicularmente os eixos vertical e horizontal.

As solicitações referentes à primeira chamada são enviadas, uma para cada SV. Como o SV_1 é o mais próximo do POP, sua confirmação chega primeiro ao chamador e ele recebe a primeira chamada (Figura 16a), alcançando $U_1 = 1/4$. A segunda chamada também é encaminhada para ele; apesar de $ConfSol_1$ ser atrasada de 5 ms, ela chega primeiro ao chamador (Figura 16b), que passa a possuir $U_1 = 1/2$. Após recebê-la, o primeiro SV encontra-se alinhado com o segundo; assim, as suas ($ConfSol_i$) chegam no mesmo instante ao SV chamador.

Até este momento duas chamadas foram encaminhadas e, por causa disto, a utilização do primeiro SV atingiu o valor A de ½ (Figura 16c). Cada chamada encaminhada por um SV, a partir desta taxa de ocupação, acarreta um incremento do atraso adicional maior que Δ_{rede}.

Em um caso hipotético, assumindo que K chamadas são encaminhadas para configurar a situação $U_i = A$; n é o número de chamadas em andamento após o alinhamento, então o atraso mínimo aplicado para assegurar a robustez do balanceamento neste SV é de $\gamma > (n - K)\Delta$, mais o atraso de alinhamento com o último servidor.

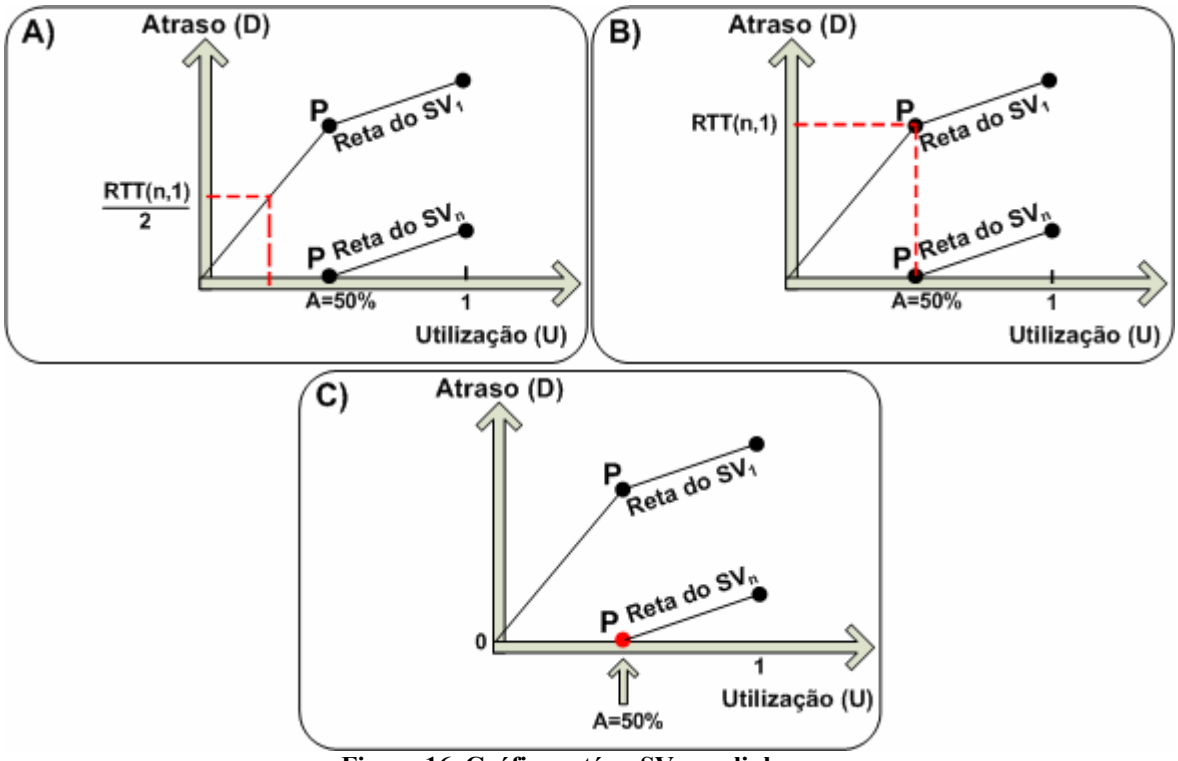


Figura 16. Gráficos até os SVs se alinharem

Será definido que a $ConfSol_1$ da próxima chamada passa pelo POP primeiro; assim, o primeiro SV encaminha a terceira chamada externa (Figura 17a) e sua utilização passa a ser de $U_1 = 3/4$. O SV_1 passa a gerar um atraso nas suas confirmações de $RTT(n,1) + \Delta$ (Figura 17a); assim, a partir de agora, o segundo servidor passa a encaminhar as chamadas.

A quarta e quinta chamadas são encaminhadas pelo segundo servidor (Figura 16c), porque as ConfSol`s dele não sofreram atrasos, já que sua utilização era $U_2 < A$. O SV_n encaminhando a sua segunda chamada, tem sua utilização alcançando o valor *A*.

A sexta chamada também é encaminhada pelo SV_2 e sua utilização chega a $U_2 = 3/4$ (Figura 17b); então, neste momento, ele inicia a forçar atrasos nas confirmações, o que propicia o balanceamento das chamadas com o SV_1.

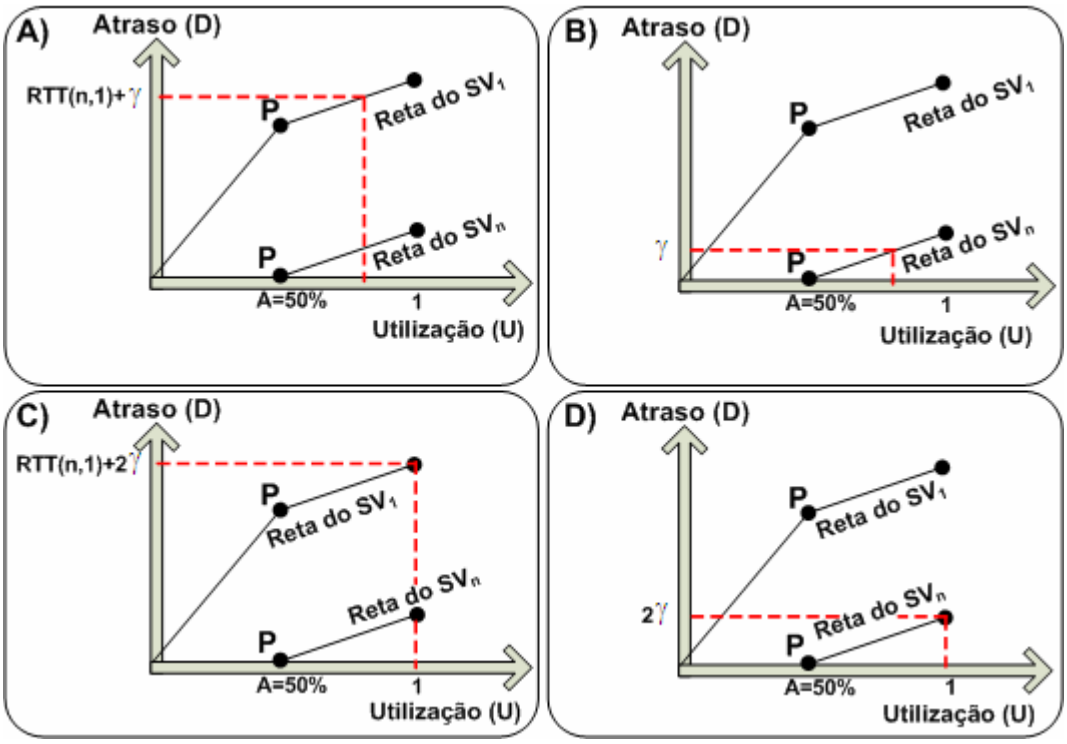


Figura 17. Gráficos após os SVs se alinharem

Quando a solicitação de uma sétima chamada é enviada para cada SV, qualquer um dos dois pode recebê-la, mas é arbitrado que a $ConfSol_1$ chega primeiro ao chamador ($U_1 = 1$). Assim, esse servidor impõe um atraso de $RTT(n,1) + 2\Delta$ para emitir próxima confirmação (Figura 17c).

A oitava chamada é encaminhada para o SV_2 ($U_2 = 1$), e suas ConfSol`s passam a receber um atraso de 2Δ (Figura 17d), ficando alinhado novamente ao primeiro SV.

Neste momento os dois servidores estão com todos os seus canais ocupados, ou seja, utilização de 100%.

Em seguida, qualquer chamada destinada à RTFC da cidade dos SVs 1 e 2 não será completada, pois não existem mais canais livres nesses SVs. Se o algoritmo não fosse utilizado, apenas o SV_1 receberia as chamadas e o outro permaneceria sem a ocupação dos seus canais de voz. Assim, sem o algoritmo de balanceamento, em vez de serem encaminhadas oito chamadas externas para RTFC, apenas quatro seriam encaminhadas. Utilizando este algoritmo otimizado o atraso para admissão de chamada será menor do que o algoritmo com SV_v, dado que eles foram comparados para um mesmo cenário.

4.5 Comentários

O algoritmo otimizado apresenta, conforme dedução matemática, um atraso adicional máximo bem menor do que o algoritmo com ponto de referência virtual. Porém, o *jitter* máximo da rede entre instituição e POP precisa ser conhecido. Qualquer avaliação errada desse *jitter* pode comprometer, em muito, o balanceamento de chamadas externas.

O SV_n não imprime atrasos para enviar a solicitação enquanto U_n não é maior que A, fazendo com que a admissão da chamada aconteça sem atrasos adicionais.

Capítulo 5

BALANCEAMENTO DISTRIBUÍDO DE CHAMADAS UTILIZANDO LÓGICA NEBULOSA

Nesta seção é mostrado como realizar um balanceamento de chamadas VoIP utilizando lógica nebulosa (*fuzzy*), que atua quando estas chamadas externas são destinadas à RTFC. Esse algoritmo utiliza atrasos no envio das confirmações para realizar um balanceamento de chamadas, que é distribuído e atua apenas com informações locais dos *gateways VoIP/PBX*. O mecanismo para encaminhamento de chamadas previsto para esse algoritmo é o da Figura 3a; além disso, as redes das instituições estão conectadas em POPs, conforme a Figura 6.

Na prospecção realizada na literatura, observou-se que a lógica nebulosa pode ser utilizada em diversas aplicações, como, por exemplo: balanceamento de carga entre processadores [32], segurança de redes [33 e 34], ajuste de desempenho de aplicações distribuídas [35] e CAC (*controle de admissão*) [36, 37 e 38]. No CAC não é realizado balanceamento de carga entre elementos de uma rede, mas assemelha-se com o balanceamento de chamadas, pelo fato de tomar decisões com base na utilização de equipamentos, ou da rede. Após essa pesquisa na literatura, procurou-se aplicar a lógica nebulosa na solução do problema de balanceamento de chamadas VoIP, sendo obtido sucesso após simulações com diversas máquinas de inferência.

A implementação da simulação desse algoritmo foi realizada no MatLab, pois este *software* possui ferramentas que permitem criar protótipos (Apêndice A), utilizando sistemas

de inferência nebuloso e, depois, submetê-los a diversas simulações. Foi possível realizar testes e validar diversos cenários de um serviço VoIP.

5.1 Balanceamento utilizando lógica nebulosa

A lógica nebulosa é utilizada no balanceamento de carga em diversas áreas, como constatado na literatura [32 e 38]. Ela será utilizada em nosso contexto para gerar os atrasos no envio das confirmações de solicitação de chamadas, utilizando a métrica em (1), como foi realizado pelos algoritmos anteriores.

A lógica nebulosa faz o alinhamento natural dos servidores VoIP (SV), sem que haja nenhuma referência a uma utilização a partir da qual se faz o alinhamento entre os servidores VoIP, como no caso do algoritmo do capítulo anterior. Um atraso que deve ser fornecido, mas não precisa ser rígido, ou seja, ele pode ser arbitrado para valores maiores que $RTT_{(1,n)}$ em cada SV_i. Optou-se por este comportamento, embora fosse possível somente forçar o início do balanceamento a partir de certa utilização.

Para este algoritmo, foi utilizado um protótipo para realizar simulações, que foi desenvolvido para funcionar no *software* MatLab. Também foram utilizadas ferramentas existentes no MatLab, que permitiram, além de criar as máquinas inferência para este algoritmo, acompanhar os seus valores de saída, frente a um conjunto de valores de entrada.

O protótipo que foi desenvolvido é mostrado no Apêndice A com mais detalhes, mostrando também as variáveis nebulosas e as interfaces para visualização do resultado final de uma simulação.

Algumas características dos algoritmos anteriores foram mantidas, como a de atuar de forma descentralizada e utilizar apenas informações locais dos *gateways VoIP/PBX*. As variáveis e nomenclaturas definidas nas outras seções serão mantidas para esta seção.

Utilizando a lógica nebulosa para o envio das confirmações, atrasos são utilizados para diminuir a probabilidade de um servidor receber a chamada. Este atraso depende do valor da utilização do *gateway* do servidor que aplica o atraso. Assim, quando um dado servidor gera um atraso, a sinalização de confirmação de outro SV tem maior probabilidade de chegar primeiro ao chamador.

5.2 Algoritmo utilizando lógica Nebulosa (Fuzzy)

A máquina de inferência do algoritmo proposto é apresentada, que é composta das variáveis nebulosas e das regras de inferência nebulosa. O cálculo realizado para esse algoritmo determinar o atraso da primitiva de confirmação também é mostrado.

Inicialmente pensou-se em utilizar diversas variáveis de entrada na máquina de inferência nebulosa, porém, isto tornaria a proposta difícil de ser implementada. Por causa disso, várias máquinas de inferências foram utilizadas na fase de testes, a fim de obter, empiricamente, uma forma de balancear as chamadas com variáveis de entrada viáveis de serem utilizadas em uma implementação. O processo de "*desnebulização*" foi investigado e, através de testes, foram verificados qual seria a melhor abordagem para o balanceamento de chamadas VoIP.

No protótipo final, duas variáveis de entrada e uma de variável de saída foram utilizadas e os testes abrangeram diversos cenários, a fim de serem construídas as semânticas dos conjuntos nebulosos destas variáveis.

Uma das variáveis de entrada é a **utilização** calculada por (1), com universo de discurso [0,1] e continua com o mesmo comportamento, definindo que, quanto maior a utilização, maior será o atraso do algoritmo (Figura 18).

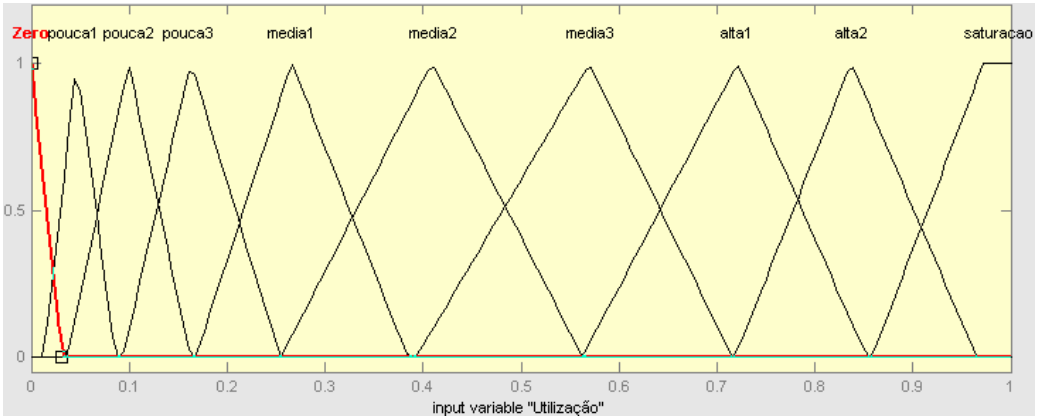


Figura 18. Variável de entrada "utilização" da máquina de inferência

Em topologias com $RTT_n >> RTT_i$, o tempo para o balanceamento será muito elevado e somente ocorrerá para valores altos de utilização; para contornar o problema, foi utilizada outra variável de entrada, chamada **razão**. Ela torna o atraso maior ou menor em função de $RTT(n,i)$, garantindo que os SV_is se alinhem ao SV_n, mesmo quando $RTT_n >> RTT_i$. A razão também garante que, quando os SV_is estão com uma utilização baixa, eles se alinham ao SV_n. A variável de entrada razão é dada por

$$\eta_i = \frac{RTT_i}{RTT_n} \quad (23),$$

ela pode ser vista na Figura 19 e, assim como a utilização, possui universo de discurso [0,1]. Essa variável permite que, para uma dada utilização, quanto maior o η, maior o atraso.

A variável utilização tem os seus valores mapeados para a variável saída, de forma que um intervalo de utilização corresponda a um intervalo de valor de atraso. Na Figura 18 há vários conjuntos nebulosos com o formato triangular, que definem os intervalos mencionados. Quanto menor a quantidade de conjuntos, menos preciso é o mapeamento para a saída e este raciocínio é utilizado para a outra variável.

Através das regras de inferência, quanto menor η, maior o atraso para geração de uma confirmação e vice-versa. Dependendo do cenário, se o SV_i estiver próximo do POP e o SV_n

muito distante, o alinhamento deles pode acontecer quando os valores de U_i são altos, p.ex. maiores que 50%. Com uso da variável η, existe a tendência dos SVs se alinharem com valores menores de utilização, comparando-os com os valores obtidos sem o uso desta variável; porém, em alguns cenários, o efeito do uso do η pode não ser tão relevante.

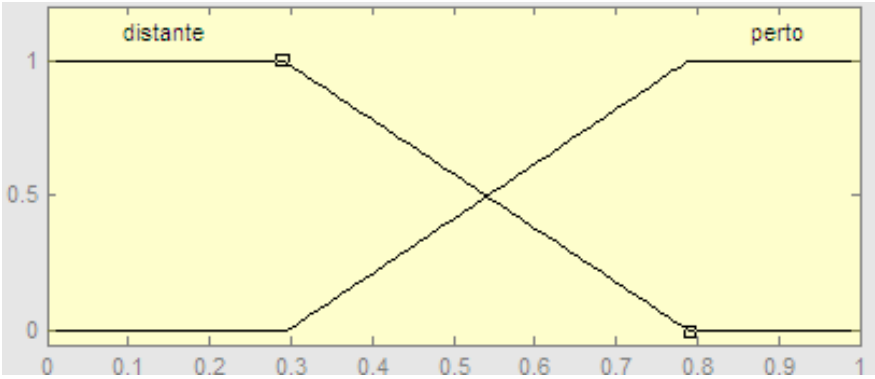


Figura 19. Variável de entrada "razão" da máquina de inferência

O atraso para geração da confirmação de solicitação será dado por

$$D_i(U_i, \eta)=\Theta.T \quad (24),$$

onde: D_i é o atraso que a sinalização de confirmação sofre; Θ é a variável de saída do processo de "desnebulização" com universo de discurso de [0, 1] (Figura 20); T representa um tempo utilizado para estimar o atraso D_i. Pelos valores que Θ pode assumir, o valor de D_i varia entre zero e T.

Em algumas situações, o η impede que algum servidor VoIP, que esteja com todos os canais dos seus *gateways* ocupados, receba chamadas, enquanto outro ainda possua canais de voz disponíveis, ou seja, evita-se subutilização dos recursos.

Nas simulações, foi utilizado $T > RTT(n,1)$ (25), que é suficiente para SV_1 pelo menos se alinhar com o SV_n. A subsessão 5.3.1 mostra os resultados de simulações, onde o valor de T sofre variações. Através destas simulações, constatou-se que o valor de $RTT_{(1,n)}$ deve ser

próximo de 20% do valor de T; assim o alinhamento do primeiro SV com o último não acontece para valores de utilização altos.

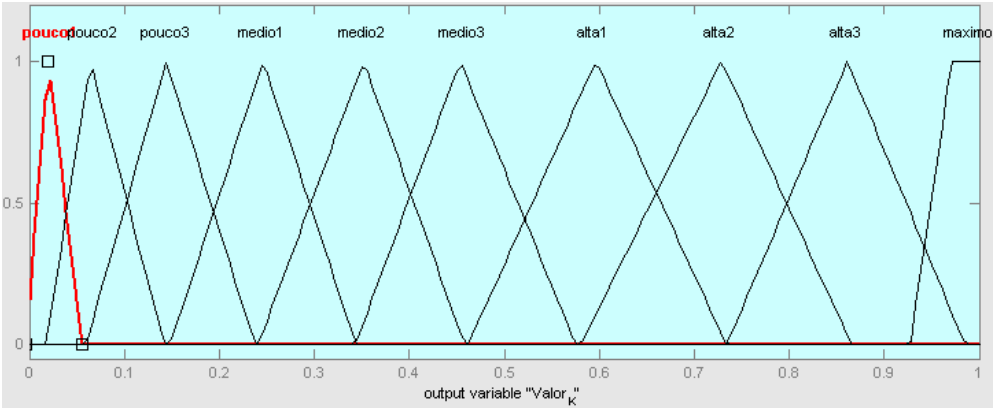


Figura 20. Variável de saída da máquina de inferência

Observe na Figura 18 e na Figura 20 que os conjuntos da variável de entrada utilização e a variável de saída possuem formato triangular, com exceção de um conjunto em cada variável. Com relação aos seus pontos sobre o eixo das abscissas, podemos notar que tais conjuntos são construídos de forma que seus limites mínimo e máximo encontram esse eixo onde os conjuntos adjacentes terminam ou começam, respectivamente. Seguindo essa forma de construção, a variação de valores na entrada mantém-se proporcional aos valores de saída. Por outro lado, caso os conjuntos adjacentes estivessem bem afastados uns dos outros, isso não ocorreria, porque a variação do atraso seria pequeno para alguns intervalos de valores de entrada, o que prejudicaria o balanceamento para alguns intervalos de valores de entrada.

As regras de inferência determinam as implicações entre os conjuntos nebulosos das variáveis de entrada e saída, fazendo com que o balanceamento aconteça.

O valor de T não deve ser maior do que a ordem de grandeza da unidade de segundos; pois isto causaria um atraso excessivo na admissão da chamada, podendo até ocasionar a desistência da chamada pelo chamador.

Quanto maior for o valor de T, menos suscetível ao *jitter* o algoritmo será; mas ele não deve ser demasiadamente grande, a fim de não provocar muito atraso para admissão da chamada.

O valor de T tem uma relação com a variável de entrada η e, de acordo com as simulações, quanto maior o valor de T menor é a dependência desta variável para o algoritmo. Todavia, em cenários onde RTT_n é elevado, o valor de T não pode ser grande, para que a admissão da chamada não seja prejudicada, por exemplo, na subsessão 5.3.2, o valor de T foi arbitrado para não alcançar unidades de segundos e, nesta situação, η permite que apenas existam chamadas não completadas quando os *gateways* estiverem com ocupação elevada de canais.

Além do que foi citado, o valor de T influi no alinhamento do SV_n com os outros servidores; quanto maior T menor é a utilização dos *gateway* para os seus servidores se alinharem com o último.

5.2.1 Balanceamento utilizando lógica nebulosa

O Quadro 1 mostrado a seguir, contém as regras utilizadas para a inferência dos conjuntos nebulosos das Figuras 18, 19 e 20.

Essas regras foram testadas para situações com $0{,}1 \leq \eta \leq 1$, através de medidas RTT em algumas instituições do serviço fone@RNP (serviço VoIP da RNP) e, através delas, constatou-se que valores dentro deste intervalo correspondem às topologias de redes utilizadas atualmente.

Quadro 1. Regras de inferência nebulosa

Regra	*Entrada*		*Saída*	*Regra*	*Entrada*		*Saída*
#	**Utilização**	η	**Θ**	#	**Utilização**	η	**Θ**
1	Zero	Perto	Pouco1	**11**	Zero	Distante	Pouco2
2	Pouca1	Perto	Pouco2	**12**	Pouca1	Distante	Pouco3
3	Pouca2	Perto	Pouco3	**13**	Pouca2	Distante	Médio1
4	Pouca3	Perto	Médio1	**14**	Pouca3	Distante	Médio2
5	Média1	Perto	Médio2	**15**	Média1	Distante	Médio3
6	Média2	Perto	Médio3	**16**	Média2	Distante	Alta1
7	Média3	Perto	Alta1	**17**	Média3	Distante	Alta2
8	Alta1	Perto	Alta2	**18**	Alta1	Distante	Alta3
9	Alta2	Perto	Alta3	**19**	Alta2	Distante	Alta3
10	Saturação	Perto	Máximo	**20**	Saturação	Distante	Máximo

5.3 Sensibilidade do parâmetro T no algoritmo utilizando lógica nebulosa

5.3.1 Variando o valor de T

O cenário proposto tem RTT_1 = 10 ms, RTT_2 = 15 ms e RTT_3 = 50 ms e apenas o parâmetro T varia. A capacidade dos *gateways* dos SVs é de 12 canais de voz, sendo considerado que , chamadas internas não ocorrem e que uma chamada externa encaminhada à RTFC tem duração infinita. Observe que os rótulos colocados embaixo das figuras deste subitem mostram o valor de T utilizado nas simulações.

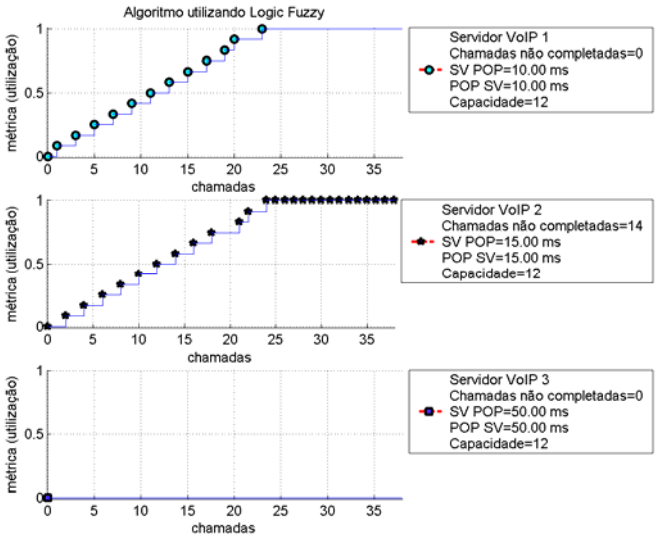


Figura 21. Valor T=$RTT_{(i,n)}$

Na Figura 21 o valor de T é suficiente apenas para os dois primeiros servidores se alinharem com o último, ou seja, T é igual a $RTT_{(i,n)}$. Como era esperado o SV_3 não encaminha chamadas.

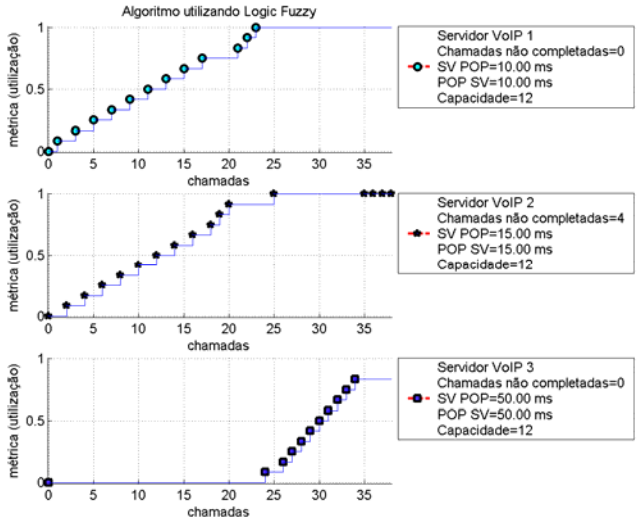


Figura 22. Valor T=$RTT_{(i,n)}$+RTT_1

Na Figura 22, o valor do T para é suficiente para o alinhamento do SVs mais 10 ms e, neste cenário, o SV_3 recebe chamadas. Observe que o SV_3 só recebe alguma chamada quando os outros estão com uma utilização altíssima.

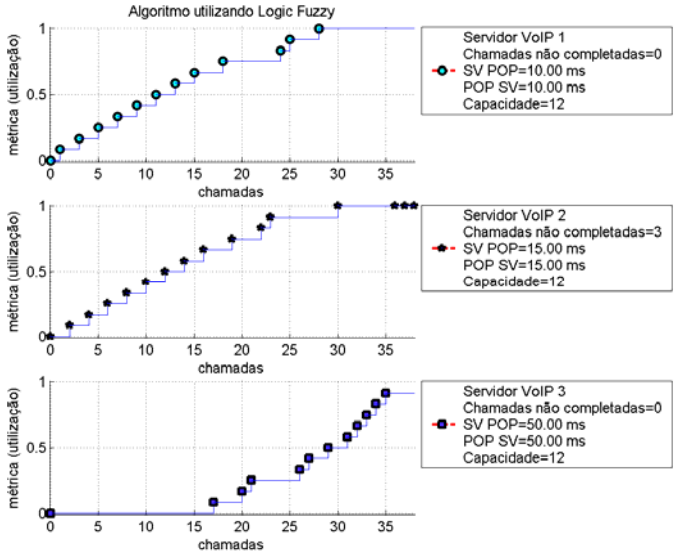


Figura 23. Valor $T=RTT_{(i,n)}+2RTT_1$

Na Figura 23 o valor de T aumenta mais 10 ms e, assim o último servidor começa a receber chamadas mais cedo. O mesmo se observa na Figura 24, mas o alinhamento dos SVs começa quando a utilização dos dois primeiros é de aproximadamente 58%.

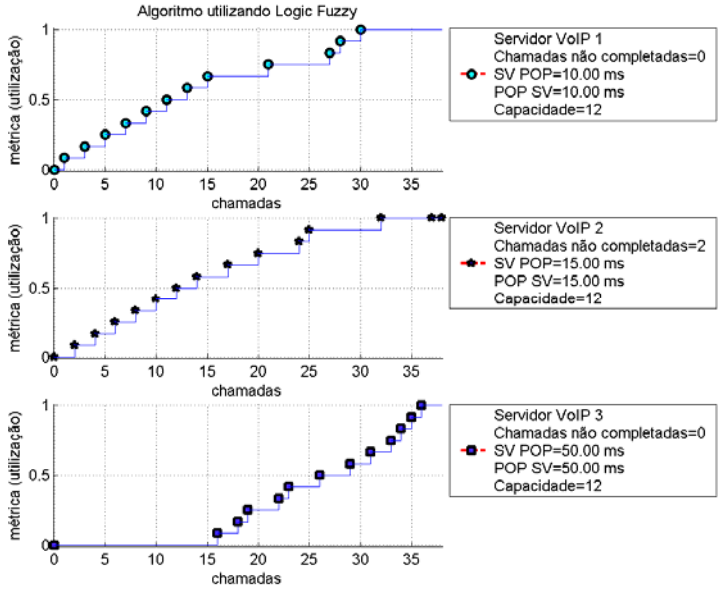


Figura 24. Valor $T=RTT_{(i,n)}+3RTT_1$

Na Figura 25 o valor de T é igual a $RTT_{(i,n)}$ mais o retardo do SV_n ao POP, assim o alinhamento dos SVs ocorrem quando os dois SVs mais próximos do POP estão com 50%,

que é um menor do que o anterior. O mesmo comportamento das figuras anteriores se apresenta, aumentando T o valor da utilização para o alinhamento dos SVs diminui.

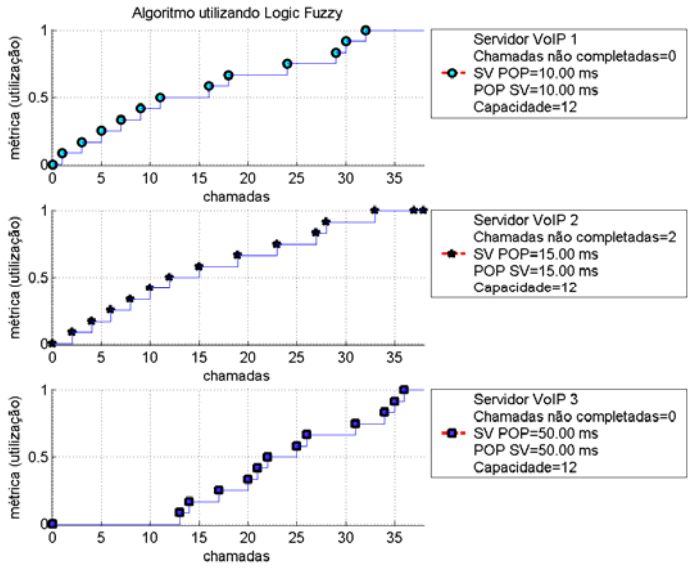


Figura 25. Valor $T=RTT_{(i,n)}+RTT_n$

Na Figura 26 o alinhamento dos SVs ocorre para valores de utilização um pouco maiores que 30%, mas para o alinhamento acontecer com valores menores do que este T tem que ser maior.

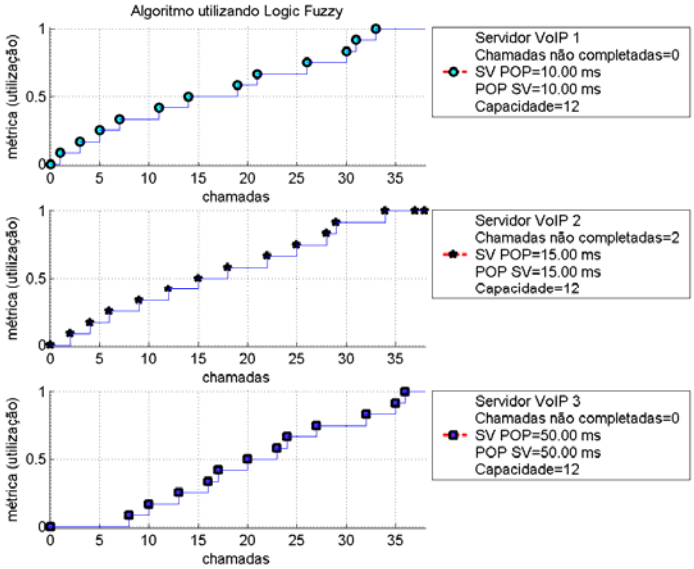


Figura 26. Valor $T=RTT_{(i,n)}+2RTT_n$

Na Figura 27, T aumenta e o alinhamento ocorre para valores menores de utilização, mas em alguns cenários esta utilização pode ser próxima de 30%. Por causa disso, procurou-se aumentar T.

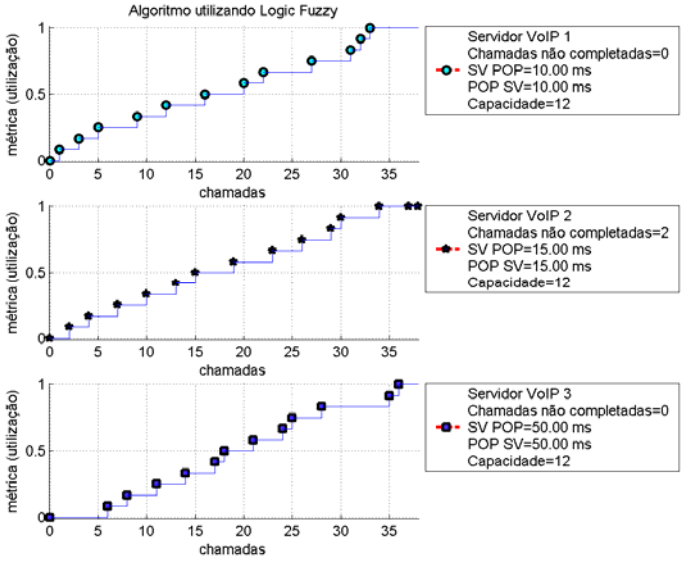


Figura 27. Valor $T=RTT_{(i,n)}+3RTT_n$

Na Figura 28, o alinhamento dos SVs ocorre com uma utilização em torno de 20% do SV_1 e do SV_2. Isso foi comprovado em outros cenários.

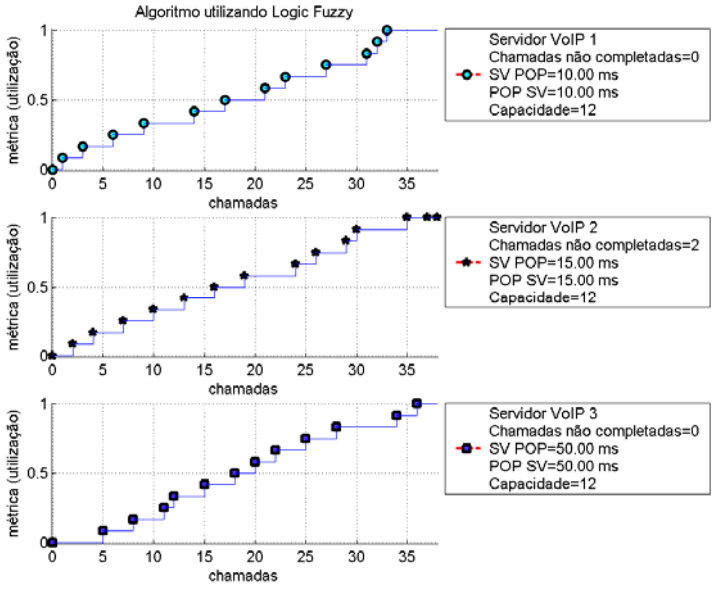


Figura 28. Valor $T=RTT_{(i,n)}+4RTT_n$

Na Figura 29, aumentando o valor de T ainda mais, o percentual da utilização para alinhamento diminuiu muito pouco, o mesmo ocorre em outros cenários. Por causa disso, o valor de T da Figura 28 foi utilizado nas simulações.

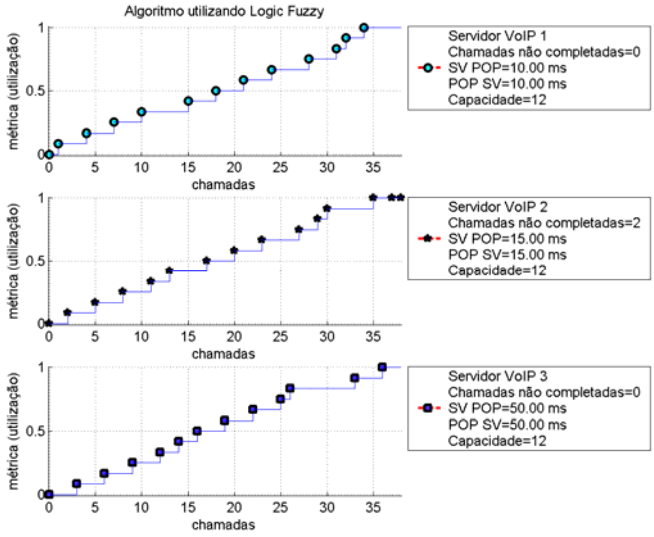


Figura 29. Valor $T=RTT_{(i,n)}+6RTT_n$

Em cenários onde existe algum η muito pequeno e um RTT_n grande, observa-se que o balanceamento é prejudicado e, isto será tratado no subiten 2. Veja que, para RTT_n grande $T= RTT_{(i,n)}+4RTT_n$ (Figura 28) geraria um atraso excessivo para admissão de chamadas.

5.3.2 Uso do η

Para verificar a sensibilidade da variável de entrada η, a Figura 30 e a Figura 31 mostram um cenário em que ela influe decisivamente no balanceamento. Nestas figuras, o cenário simulado é o mesmo, onde RTT_1 = 10 ms, RTT_2 = 50 ms e RTT_3 = 550 ms. Como RTT_3 é alto, T não pode ser elevado, para não prejudicar o balanceamento de chamadas e, de acordo com a Figura 28 T seria $RTT_{(i,n)}+4RTT_n$, assim o SV_3 geraria um atraso para U=1 igual a 2,2 segundos.

Nos cenários simulados, a capacidade dos *gateways* é de 15 canais de voz e as únicas chamadas utilizadas na simulação são as externas, que possuem duração infinita quando são encaminhadas para a RTFC.

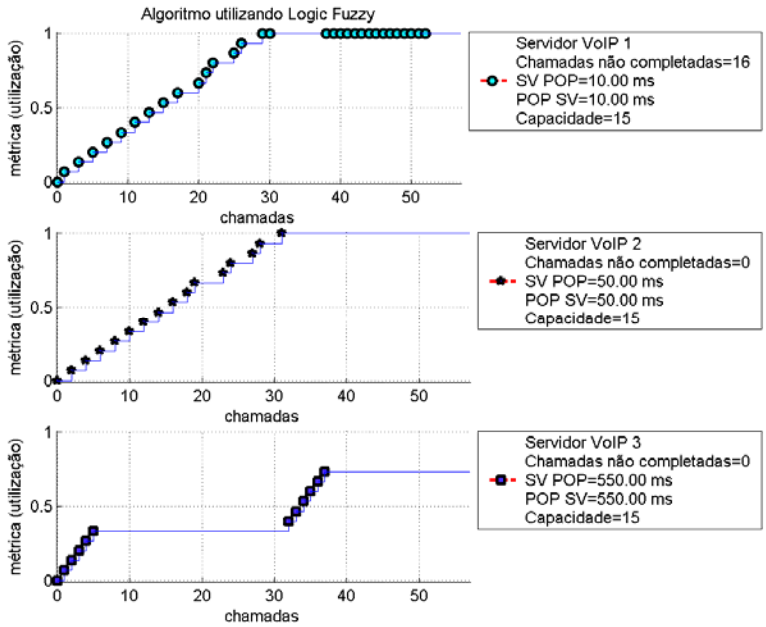


Figura 30. Simulação sem uso da variável η

Na Figura 30, η não foi utilizado, fazendo com que chamadas não completadas aconteçam quando o *gateway* do último SV ainda tinha canais livres.

O valor de $T=RTT_{(i,n)}+200ms$ foi escolhido para não atrasar muito a admissão da chamada. Como o RTT_n é de 550 ms, um atraso maior do que 200 ms, após o alinhamento dos SVs com o último, levaria a admissão da chamada a um atraso próximo de um segundo.

Em uma situação onde existe pelo menos um servidor com η muito pequeno e uma das instituições tem um retardo de propagação elevado, recursos de alguns *gateways* podem ser subutilizados (Figura 30). No exemplo deste item, o SV_3 acabaria prejudicando a admissão de chamada dos servidores em funçã do valor de RTT, como mostrado na Figura 30.

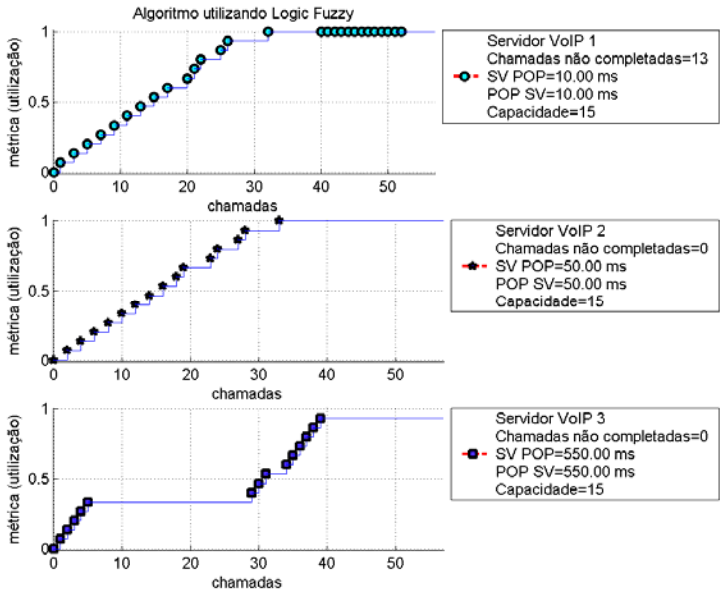


Figura 31. Simulação usando a variável η

Nesse cenário, a partir de um $T=RTT_{(i,n)}+300$ ms, as chamadas não completadas somente ocorrem quando todos os *gateways* estão com utilização igual a 100%; porém, este é uma valor expressivo para o atraso, e para diminuí-lo η será utilizado.

Nesse exemplo, valor da variável η para os servidores serviria apenas para inferir o quanto eles estão distantes do SV_n; em função disso, basta o valor de η ser próximo do verdadeiro.

Em contrapartida, η foi utilizado na Figura 31 com $T=RTT_{(i,n)}+200$ ms, permitindo uma ocupação de canais elevada dos *gateways VoIP/PBX*, antes da ocorrência de chamadas não completadas.

É importante observar que as regras de inferência influem no comportamento do balanceamento, podendo assim, sofrer alterações, resultando no uso de um valor de T menor para o cenário apresentado aqui.

5.4 Comentário

Como foi visto, a precisão de alguns parâmetros é dependente do cenário; outro fator importante é o tempo para admissão da chamada, que precisa ser considerado para valores grandes de T.

Em termos de implementação, tomando como base o ambiente do fone@RNP, a implementação deste algoritmo no *gatekeeper* GNUGK precisaria ser feita alterando o código; o que pode ser um complicador.

No ambiente SIP, o OpenSER pode executar um programa externo, para ele operar com este algoritmo. Assim, não há necessidade de alteração de código desses servidores.

Uma vantagem deste algoritmo é que o valor de η não precisa ser medido com freqüência e T pode ser arbitrado, desde que não prejudique muito a admissão da chamada com atrasos excessivos. Dependendo do valor de T e Δ_{rede}, o algoritmo pode apresentar erros aceitáveis, sem prejuízo para o balanceamento de chamadas.

Capítulo 6

SIMULAÇÃO E COMPARAÇÃO DOS ALGORITMOS DISTRIBUÍDOS

Foi desenvolvido um protótipo para realizar a simulação dos algoritmos de controle distribuído (capítulos 3 e 4) e lógica nebulosa (capítulo 5), que funciona através do *software* MatLab [39 e 40]. Os detalhes sobre esse protótipo encontram-se no Apêndice A.

Nesta seção, os resultados obtidos na simulação são apresentados para três cenários diferentes, os quais possuem $RTT_{(i,n)}$ e capacidade dos *gateways* diferentes. Gráficos são utilizados para mostrar a distribuição de chamadas e atrasos gerados pelo algoritmo, sendo possível visualizar o comportamento dos algoritmos frente a diversos cenários. Esses resultados serão importantes para as conclusões finais.

6.1 Considerações sobre as simulações

Três cenários são apresentados, possuindo retardos diferentes de propagação na rede entre o POP e os SVs. Os valores das capacidades também são modificados e o valor de *A* é mantido para todas as simulações. Outro detalhe importante é que uma chamada encaminhada por um *gateway* tem duração infinita.

No início de cada seção, um cenário é apresentado, com os valores adotados para a simulação. Em cada subseção, um algoritmo é simulado para o cenário e os resultados obtidos são mostrados em gráficos. Qualquer dúvida para ler os gráficos de distribuição de chamadas e atrasos impingidos pelos algoritmos, basta ler o Apêndice A.

Quando as simulações com o algoritmo otimizado forem mostradas, considera-se que o Δ_{rede} é de 20 ms.

Através do gráfico de distribuição de chamadas, pode ser verificado, por exemplo, se um *gateway* recebe mais de uma chamada consecutiva, o que não é desejável. Os gráficos que mostram os atrasos possuem o atraso instantâneo gerado pelo algoritmo e o atraso médio; e este atraso médio foi calculado, utilizando a média aritmética dos atrasos gerados pelas chamadas admitidas.

Para o algoritmo utilizando lógica nebulosa, o valor de T foi arbitrado como $RTT_{(i,n)}$+4*RTT_n. Caso o valor de $RTT_{(1,n)}$ seja maior do que 20% do valor de T, o alinhamento do primeiro SV com o último acontece para valores de utilização altos e, como conseqüências, podem existir muitas chamadas consecutivas para um mesmo SV.

6.2 Primeiro cenário

No primeiro cenário existem três SVs, os quais têm um retardo de propagação até o POP de 100 ms, ou seja, $RTT = 100\,ms$. Esses SVs possuem $C = 4$ e o valor de A é 25%. As chamadas, ao serem encaminhadas por um servidor ao seu *gateway*, não são terminadas. Na simulação foi utilizado um Δ_{rede} com o valor de 20ms.

6.2.1 Algoritmo com ponto de referência virtual

A Figura 32 e a Figura 33 mostram os resultados obtidos na simulação do algoritmo utilizando ponto de referência, sendo que a primeira figura mostra a distribuição de chamadas e a outra os atrasos impostos pelo algoritmo à primitiva de sinalização.

Na Figura 32, o balanceamento não concentrou muitas chamadas consecutivas em um servidor apenas. Para isso não acontecer, um *jitter* elevado precisaria ocorrer.

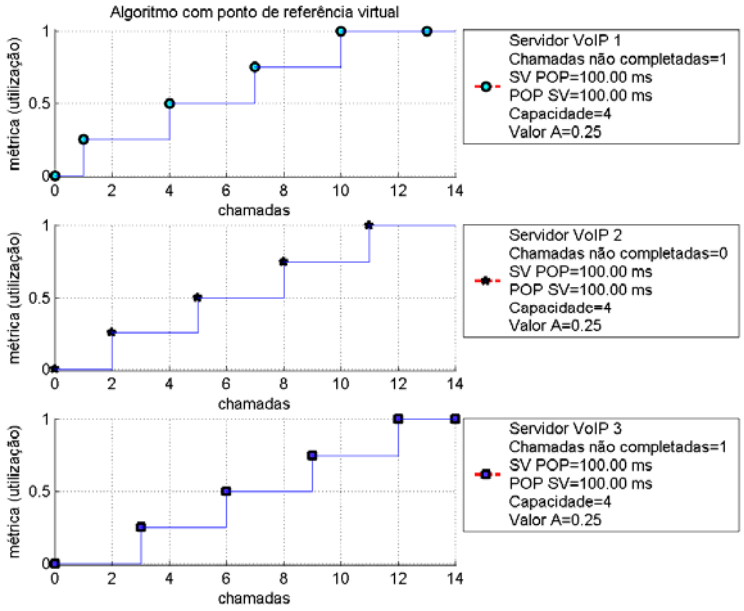


Figura 32. Distribuição de chamadas – simulação 1

Para este algoritmo, a fórmula (11) para o posicionamento do SV_v não foi utilizada, pois, como, $RTT(1,n) = RTT(2,n) = 0$, o servidor virtual teria uma posição definida de RTT_v=300 ms e, caso ele fosse reduzido a $1/3$ deste valor, o atraso máximo ainda assim ficaria acima dos apresentados pelos outros algoritmos (Figura 33).

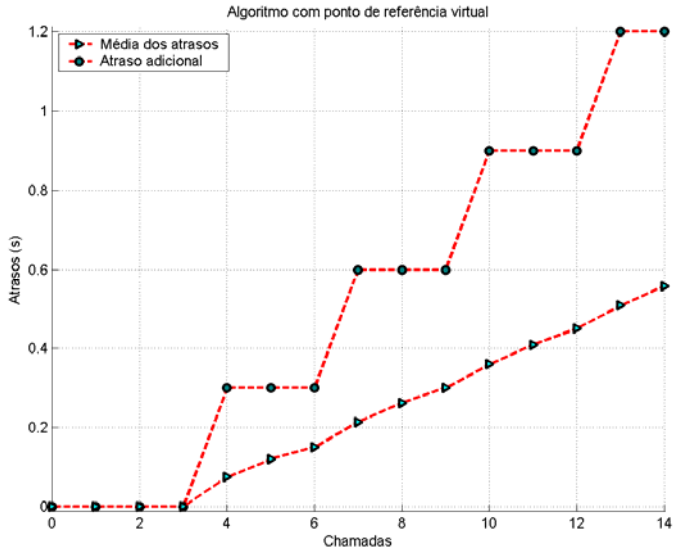


Figura 33. Atrasos gerados – simulação 1

6.2.2 Algoritmo otimizado

A Figura 34 mostra a distribuição de chamadas para o algoritmo otimizado, já a Figura 35 mostra os atrasos para este algoritmo.

Neste algoritmo, como temos os $RTT(i,n)=0$, no início da simulação os SVs estão alinhados. O Δ_{rede} na simulação é de 20 ms. Na Figura 34, a distribuição de chamadas é muito parecida com a anterior.

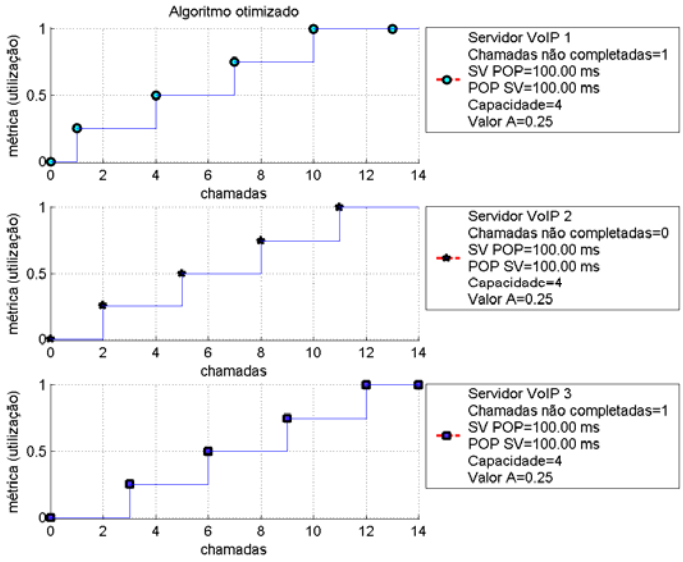


Figura 34. Distribuição de chamadas – simulação 2

Na Figura 35, o atraso adicional máximo é bem menor do que o algoritmo com ponto de referência e, mesmo que o *jitter* da rede aumentasse, dificilmente teríamos um quadro apontando para a situação inversa.

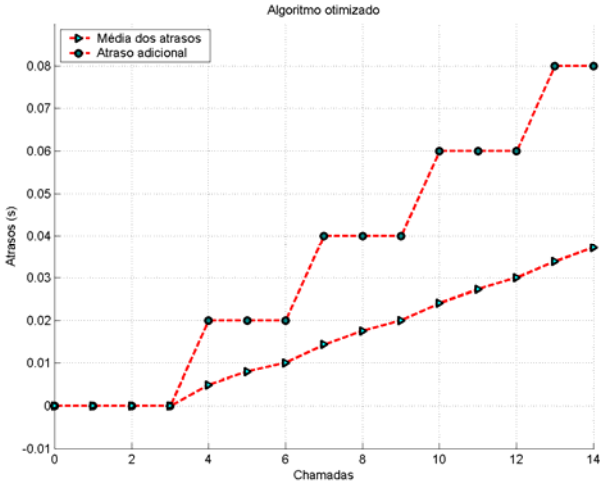


Figura 35. Atrasos gerados – simulação 2

6.2.3 Algoritmo utilizando lógica nebulosa

Na Figura 36, temos a distribuição de chamadas para o algoritmo utilizando lógica nebulosa e, na Figura 37, os atrasos gerados por este algoritmo.

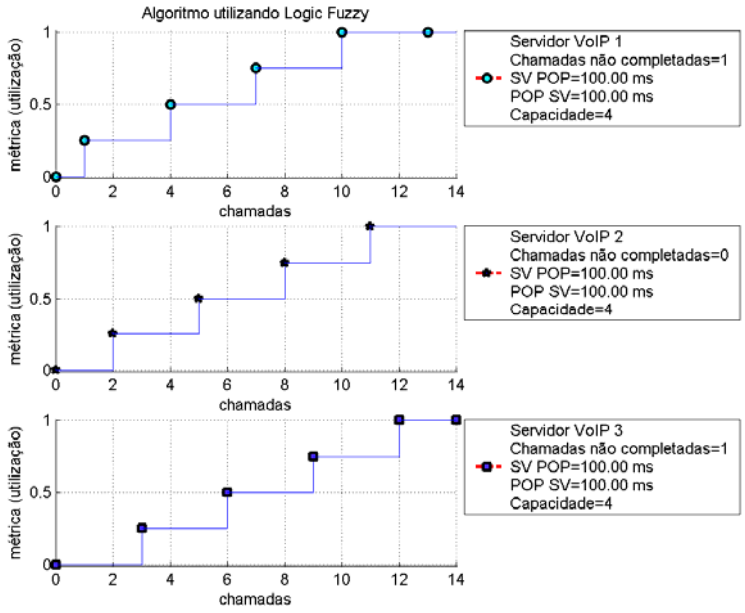


Figura 36. Distribuição de chamadas – simulação 3

Na Figura 36, a distribuição de chamadas é muito parecida com aquelas distribuições dos algoritmos anteriores, o que ocorre em função dos SVs terem o mesmo RTT. Na Figura 37, o T utilizado para cada SV foi de $RTT_{(i,n)}$+400 ms.

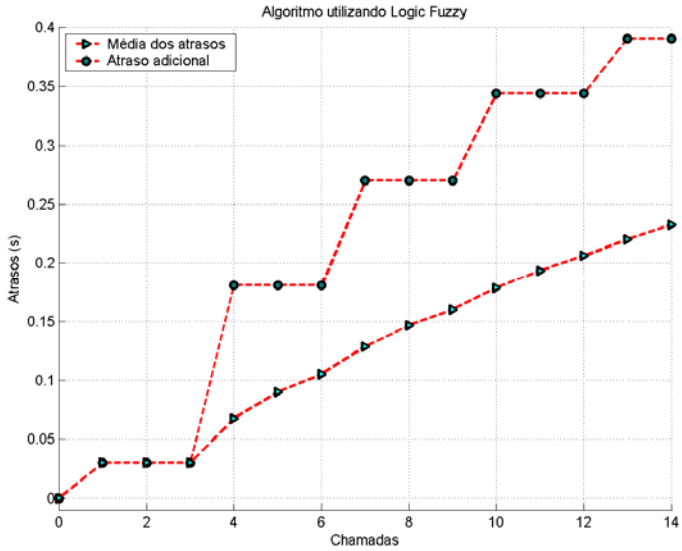


Figura 37. Atrasos gerados – simulação 3

6.2.4 Comparação dos atrasos gerados pelos algoritmos no segundo cenário

Na figura abaixo, os atrasos obtidos pelos algoritmos na simulação do primeiro cenário são mostrados. Constata-se que os atrasos do algoritmo utilizando ponto de refência virtual são os maiores, já o algoritmo otimizado é o que gera os menores atrasos.

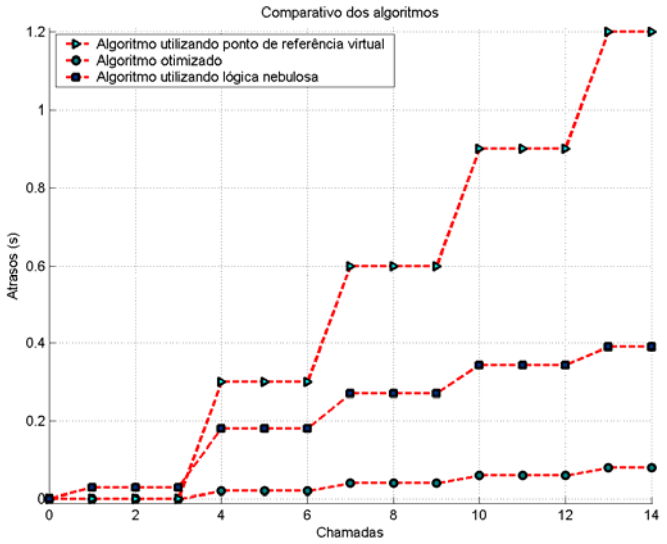


Figura 38. Comparação dos atrasos – simulação 1 a 3

6.2.5 Comentários

Pelos resultados obtidos, que são mostrados nas três subseções anteriores, observa-se que a distribuição de chamadas é idêntica para qualquer algoritmo. Isto acontece porque os RTT_is são iguais, ou seja, os GW_is estão alinhados, quando todos os seus canais estão desocupados.

Nesse cenário, descarte de chamadas apenas ocorrem quando todos os canais já se encontram ocupados, assim recursos dos *gateways* não são desperdiçados.

O atraso adicional máximo impingido pelo algoritmo é menor para o algoritmo otimizado, o que é dependente do valor do *jitter* utilizado. O algoritmo com lógica nebulosa e o algoritmo otimizado podem ter o atraso adicional máximo na mesma ordem de grandeza, dependendo dos valores do *jitter* e T. Se o *jitter* tivesse um valor de pelo menos 100 ms, os algoritmos otimizado e com lógica nebulosa teriam atrasos adicionais máximos bem próximos.

Como já era esperado, em função das deduções matemáticas, o algoritmo com SV_v possui o maior atraso adicional.

6.3 Segundo cenário

Nesse cenário existem três SVs, que têm um retardo de propagação do POP: $RTT_1 = 90\,\text{ms}$, $RTT_2 = 170\,\text{ms}$ e $RTT_3 = 180\,\text{ms}$. Esses SVs possuem $C = 12$, sendo o valor de *A* igual a 25%. Assim como no cenário anterior, as chamadas encaminhadas através de um *gateway* possuem duração infinita. Na simulação foi utilizado um Δ_{rede}=20ms.

6.3.1 Algoritmo com ponto de referência Virtual

A Figura 39 mostra a distribuição de chamadas desta simulação e observa-se que os recursos dos *gateways* não foram subutilizados. Existe um grande número de chamadas não completadas, mas isto ocorre apenas quando todos os canais de voz já estavam ocupados.

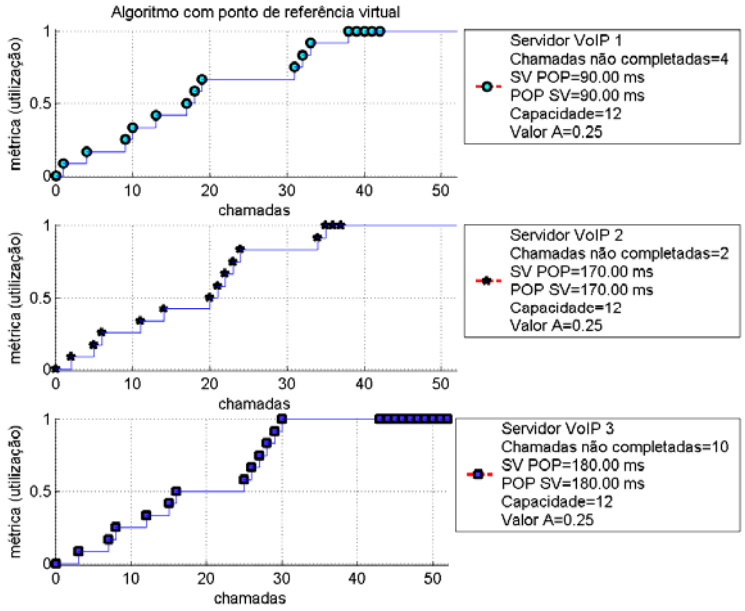


Figura 39. Distribuição de chamadas – simulação 4

Pelos RTTs utilizados espera-se um valor enorme para o atraso adicional máximo, o que é mostrado pela Figura 40. O RTT_v utilizado nessa simulação foi 1980 ms, o que gera um atraso máximo de 7920 ms.

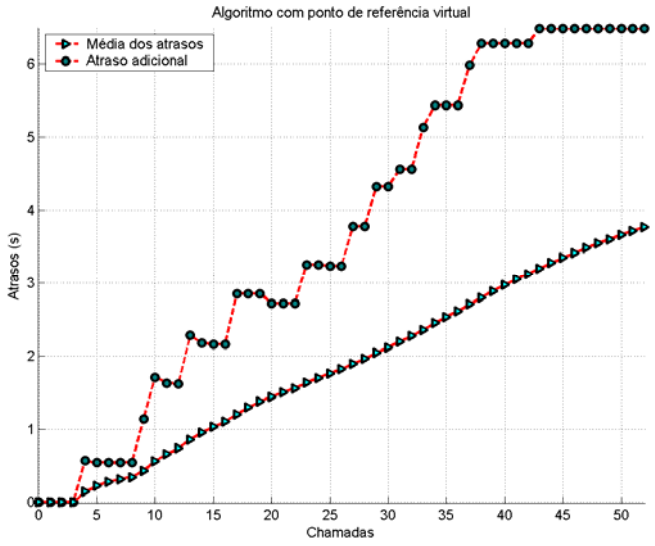


Figura 40. Atrasos gerados – simulação 4

6.3.2 Algoritmo Otimizado

O Δ_{rede} na simulação da Figura 41 e Figura 42 é de 20 ms, e como era de se esperar, o atraso médio deste algoritmo é bem menor quando comparado ao algoritmo com ponto de referência virtual.

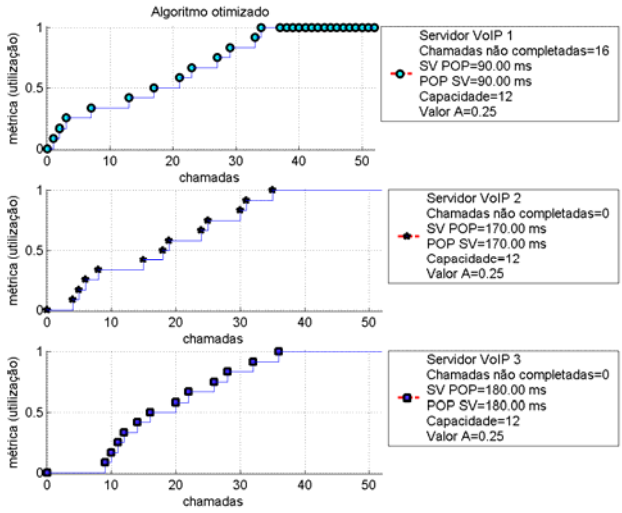


Figura 41. Distribuição de chamadas – simulação 5

Em situações reais, comparando os atrasos da Figura 42 com aqueles obtidos com o algoritmo anterior, sempre existiria um atraso menor para a admissão da chamada, porque os enlaces utilizados atualmente nas redes possuem pequenos retardos de propagação.

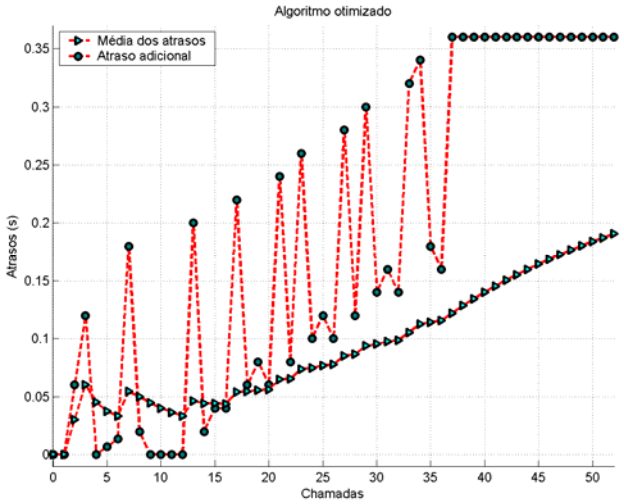


Figura 42. Atrasos gerados – simulação 5

6.3.3 Algoritmo utilizando lógica nebulosa

A Figura 43 mostra a distribuição de chamadas utilizando lógica nebulosa e, assim, como nos algoritmos anteriores, não há desperdício de recursos dos *gateways*.

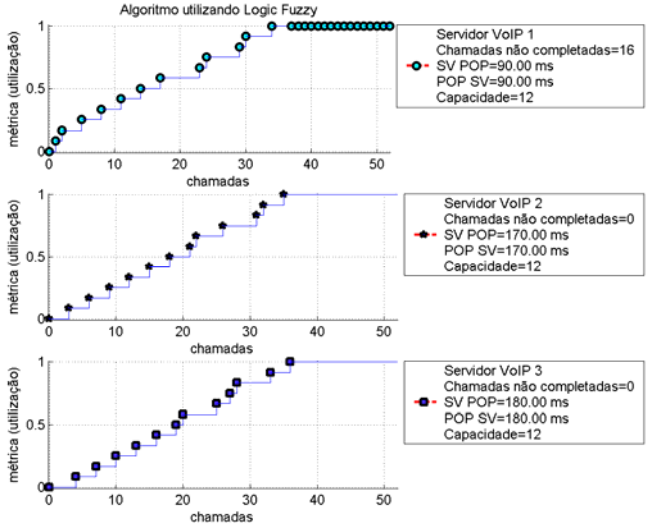


Figura 43. Distribuição de chamadas – simulação 6

Na Figura 44, o T utilizado para cada SV foi de $RTT_{(i,n)}$+360 ms ($2RTT_n$=360 ms); neste cenario não foi utilizado T= $RTT_{(i,n)}$+$4RTT_n$, porque este é um valor alto e, através de simulação, consegue-se um balanceamento de chamadas.

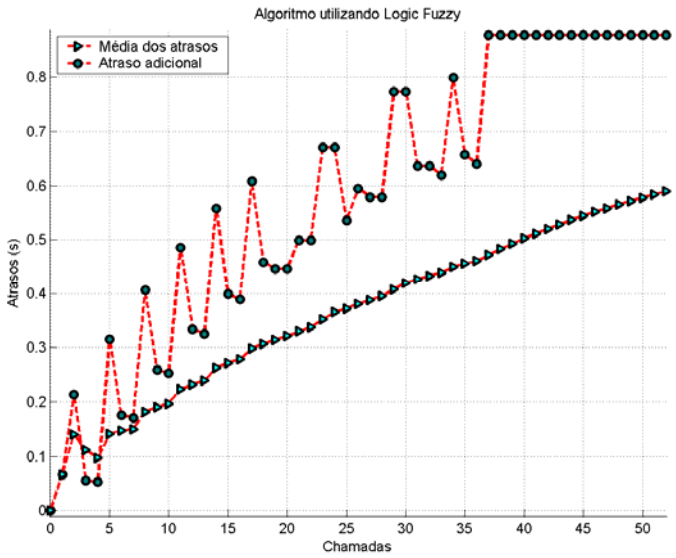


Figura 44. Atrasos gerados – simulação 6

Considera-se que o Δ_{rede} não seria de uma ordem de grandeza suficiente para prejudicar o balanceamento. Utilizando este T, a admissão de chamadas não é prejudicada com grandes atrasos.

6.3.4 Comparação dos atrasos gerados pelos algoritmos no segundo cenário

Na figura abaixo, os atrasos obtidos pelos algoritmos na simulação do segundo cenário são mostrados. Os atrasos do algoritmo utilizando ponto de refência virtual continuam sendo maiores do que os outros.

Além disso, verifica-se que com o aumento da utilização dos canais de voz do *gateway*, os atrasos gerados por este algoritmo possuem uma grande tendência de aumento, enquanto que a mesma tendência nos outros algoritmos se mantém baixa. Isto ocorre em função do posicionamento do servidor virtual.

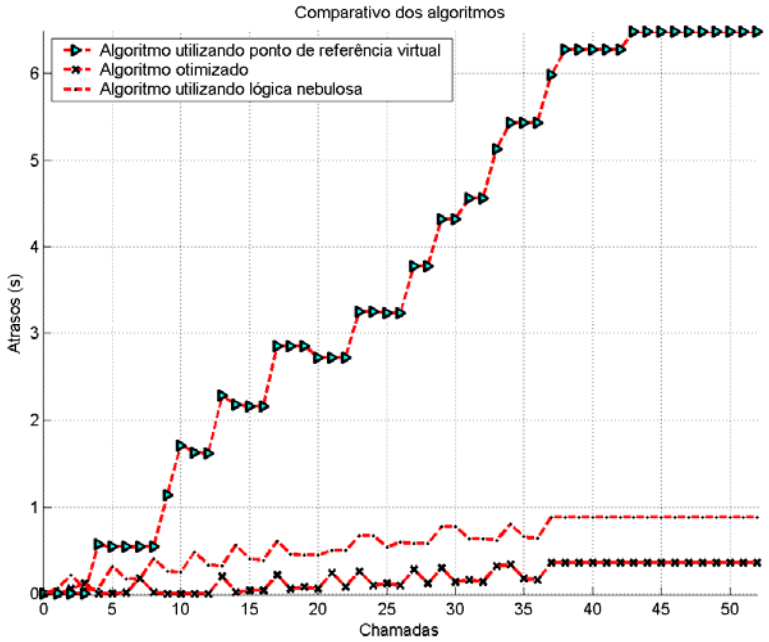


Figura 45. Comparação dos atrasos – simulação 4 a 6

6.3.5 Comentários

O algoritmo com ponto de referência virtual apresenta algumas aglomeração de chamadas consecutivas, o que pode ser evitado, aumentando o posicionamento do SV_v. Ao contrário disso, os outros não apresentam aglomeração de chamadas consecutivas.

Os atrasos utilizando o algoritmo otimizado continuam sendo os menores, como é esperado pelas fórmulas que foram apresentadas. Assim, a admissão de chamadas é favorecida neste algoritmo. As chamadas não completadas, assim como no cenário anterior, apenas existem quando todos os SVs estão com canais ocupados.

Para este cenário, o algoritmo utilizando lógica nebulosa apresenta o segundo menor atraso na admissão das chamadas, quando comparado com os outros dois. Apesar do T utilizado ser um pouco maior de 360 ms, ele poderia ser menor neste cenário, caso Δ_{rede} não prejudique o balanceamaneto.

Várias chamadas consecutivas são encaminhadas para o SV_n utilizando o algoritmo otimizado, enquanto $U_n \leq A$ e os outros SVs estão com utilização de pelo menos A, o que é previsto por esse algoritmo torna-se mais evidente aqui, onde as capacidades dos GWs são maiores.

6.4 Terceiro cenário

Nesse último cenário existem três SVs, os quais têm diferentes retardos entre eles e o POP. O SV_1 possui um retardo de propagação do POP de 10 ms, SV_2 tem um retardo de 15 ms e o SV_3 tem um retardo de 50ms, ou seja, $RTT_1 = 10\,ms$, $RTT_2 = 15\,ms$ e $RTT_3 = 50\,ms$. Esses SVs possuem $C = 12$, sendo o valor de A é 25%. Na simulação o Δ_{rede} adotado foi de 20ms.

6.4.1 Algoritmo com ponto de referência virtual

Nesta simulação, a Figura 46 mostra a distribuição de chamadas sem subutilização dos canais de voz dos *gateways*. Para esta simulação RTT_V=720 ms foi utilizado, podendo gerar um atraso máximo de 2880 ms.

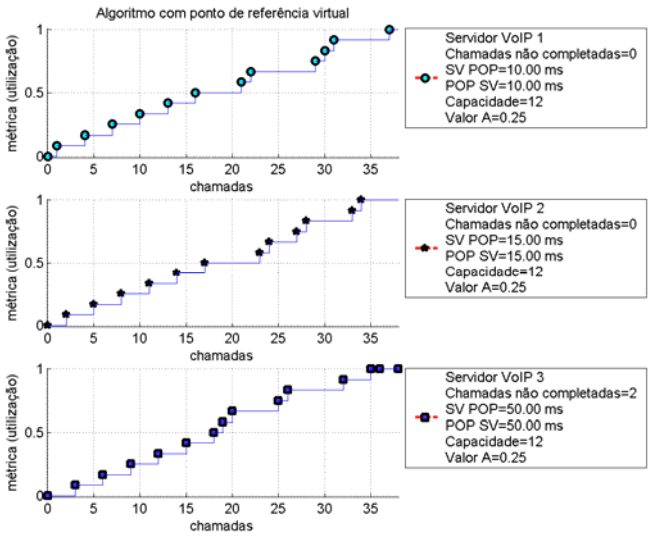


Figura 46. Distribuição de chamadas – simulação 7

Na Figura 47, o atraso impingido por este algoritmo é excessivo, como previsto pelas deduções matemáticas. Em função disso, nas redes utilizadas atualmente, dificilmente o *jitter* da rede provocaria problemas no balanceamento.

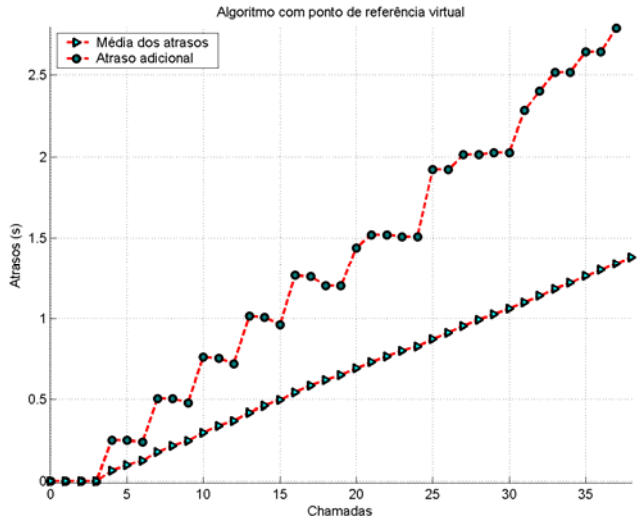


Figura 47. Atrasos gerados – simulação 7

6.4.2 Algoritmo otimizado

Neste algoritmo, o Δ_{rede} foi de 20 ms; a Figura 48 mostra a distribuição de chamadas e a Figura 49 os atrasos impingidos pelo algoritmo. Observe que a partir da nona chamada os dois primeiros SVs estão com uma utilização de pelo menos *A*, fazendo com que o SV_3 receba chamadas até ele atingir a utilização de 25%.

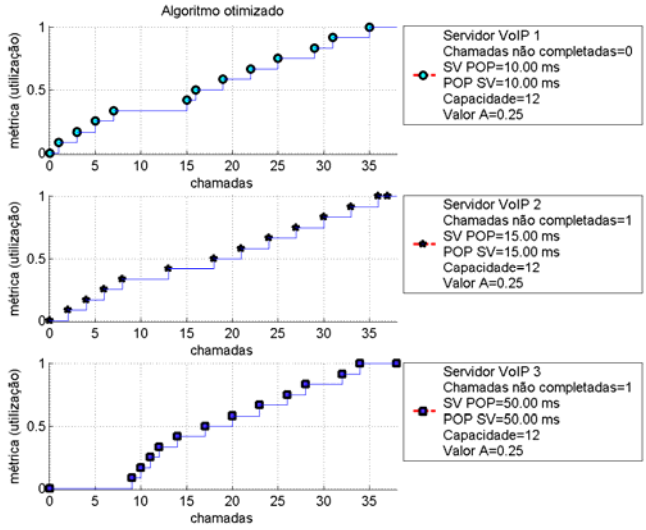


Figura 48. Distribuição de chamadas – simulação 8

Na Figura 49, o atraso é menor do que aquele apresentado pelo algoritmo anterior, como era de se esperar.

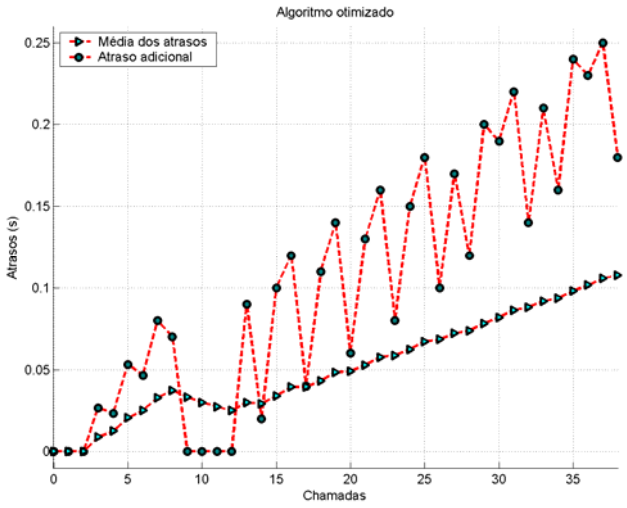


Figura 49. Atrasos gerados – simulação 8

6.4.3 Algoritmo utilizando lógica nebulosa

A Figura 50 mostra a distribuição de chamadas e, pelo algoritmo, o balanceamento ocorre sem chamadas não completadas, enquanto existem canais disponíveis nos GWs.

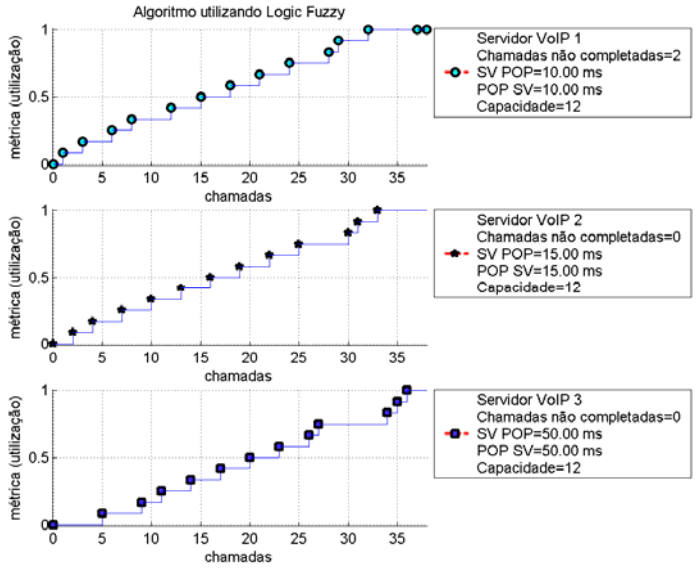


Figura 50. Distribuição de chamadas – simulação 9

Na Figura 51 utilizou-se T= $RTT_{(i,n)}$+200 ms, provocando atrasos próximos aos do algoritmo otimizado.

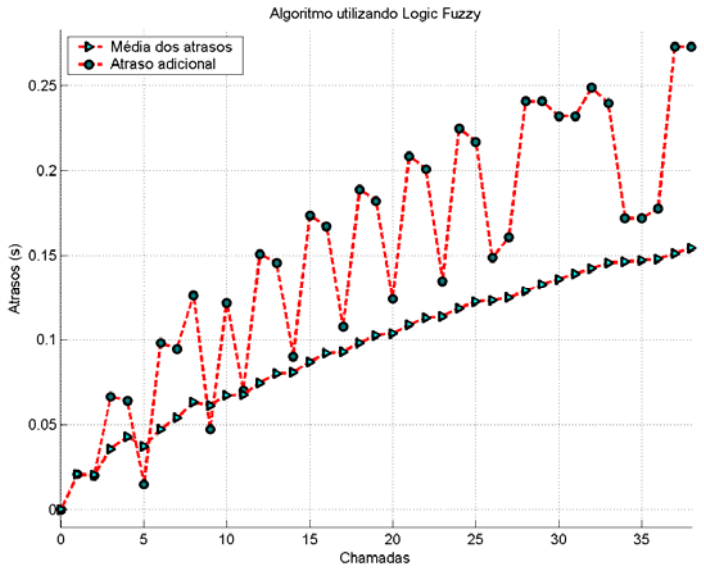


Figura 51. Atrasos gerados – simulação 9

Observe que o atraso médio deste algoritmo é menor, porque segundo o algoritmo anterior o SV_3 reduz a média quando ele atinge U_3=A e algumas chamadas são enviadas com atraso zero. Em um cenário com 30 canais, por exemplo, um *gateway* com placa E1, a diferença entre a média desses algoritmos seria maior ainda.

6.4.4 Comparação dos atrasos gerados pelos algoritmos no segundo cenário

Na figura abaixo os atrasos gerados pelos três algoritmos são mostrados, quando estes atuam no terceiro cenário. Observa-se os valores de atrasos próximos dos algoritmos utilizando lógica nebulosa e o algoritmo otimizado. Já os atrasos gerados pelo algoritmo utilizando ponto de referência virtual, são bem maiores do que os obtidos pelos outros algoritmos.

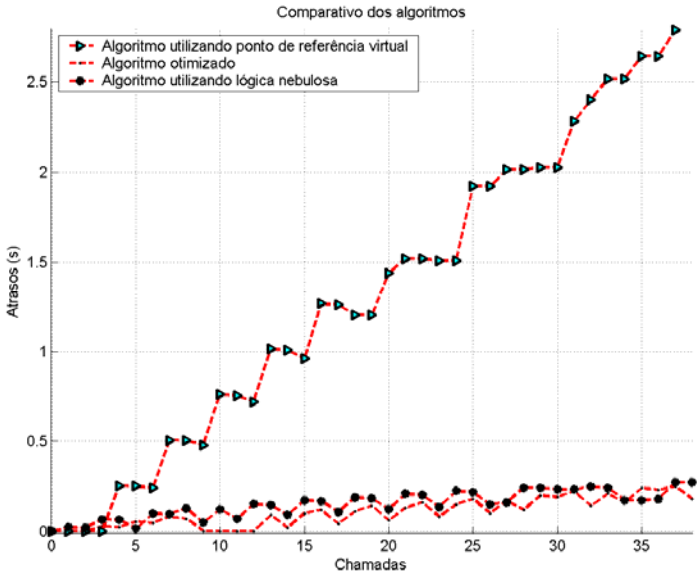


Figura 52. Comparação dos atrasos – simulação 7 a 9

6.4.5 Comentário

A distribuição de chamadas em todos os algoritmos não mostra aglomerações de chamadas consecutivas em um mesmo SV. O maior atraso para admissão de chamadas é ocasionado pelo algoritmo com ponto de referência, o que já é esperado.

Diversas chamadas consecutivas são encaminhadas para o SV_n com o algoritmo otimizado, enquanto $U_n \leq A$ e os outros SVs estão com utilização de pelo menos A. Durante este momento, o atraso gerado pelo SV_n no envio da solicitação é de zero.

Os atrasos obtidos com o algoritmo otimizado e lógica nebulosa são da mesma ordem de grandeza. Assim, dependendo da quantidade de canais e do valor do *jitter*, no algoritmo com lógica nebulosa o atraso adicional máximo pode ser próximo ao do algoritmo otimizado; mas sua média tende a ser menor.

Capítulo 7

MECANISMO DE BALANCEAMENTO CENTRALIZADO PARA CHAMADAS VOIP

A tomada de decisão do balanceamento pode ser realizada por uma entidade central, ao contrário do que foi exposto até o momento. No encaminhamento das chamadas externas, esta entidade poderá ser o DGK ou o DSER e, para isso ocorrer, ela precisa ter acesso à utilização dos *gateways VoIP/PBX*, das instituições pertencentes ao serviço VoIP que utilizam esse balanceamento centralizado.

Assim, será necessário que as informações sobre utilização dos *gateways* das instituições sejam transmitidas para um BD (banco de dados) [41], que está localizado próximo ao servidor de localização. Com isto, o DSER/DGK consulta rapidamente essas informações e o mecanismo para encaminhamento de chamadas da Figura 3b (página 41) pode ser aplicado.

Nesse balanceamento, dois parâmetros precisam ser analisados em uma implementação: o tempo de atualização das informações que são replicadas e o impacto destas atualizações na rede.

Na seção 7.1, o balanceamento centralizado é apresentado e, na seção seguinte, detalhes a respeito de sua implementação são descritos.

7.1 Balanceamento centralizado

Em um serviço VoIP, várias instituições em uma mesma cidade podem encaminhar chamadas destinadas à RTFC, e o balanceamento centralizado tem que contemplar a instituição com menor utilização no seu *gateway* de voz. Elas podem ser destinadas, seguindo uma ordem pré-estabelecida (prioridades), para uma instituição com maior utilização no seu *gateway* somente ter chances de encaminhar chamada quando aquelas com menor utilização não puderem fazer isto.

Como a preferência (prioridade) para o encaminhamento da chamada é dada ao servidor que possui o *gateway* com menor utilização, atrasos não são utilizados, assim a decisão é ótima.

No balanceamento centralizado, as informações sobre utilização dos *gateways* associados aos servidores das instituições são replicadas para a entidade central, tarefa esta realizada pelo *software* Slony-l [41]. Então, para realizar o balanceamento com base na utilização dos *gateways*, um servidor de localização (DSER/DGK) consulta essas informações e determina as prioridades aos servidores das instituições.

O mecanismo de encaminhamento de chamadas utilizado é o da Figura 3b na página 41, assim, caso esse balanceamento seja viabilizado no ambiente H.323, o DGK precisa atuar em modo *proxy* de sinalização, como mostrado na arquitetura Figura 4c. Outra opção é implementar esse mecanismo no ambiente SIP e, para isto, a arquitetura da Figura 5a é a mais indicada.

A viabilidade para implantação de um balanceamento centralizado foi analisada no ambiente SIP, pois nele é possível utilizar apenas soluções padrão, que causam pouco impacto a arquitetura VoIP. Uma implementação desse balanceamento no ambiente H.323 necessitaria

de modificações no código fonte do DGK para operar com o mecanismo da Figura 3b, porque este servidor de localização não pode atuar desta forma por padrão, ao contrário do DSER no ambiente SIP. Em vista do que foi exposta, a implementação desse balanceamento é feita somente no ambiente SIP, utilizando o mecanismo Figura 3b e a arquitetura Figura 5a.

Para efetivar a implementação do balanceamento centralizado, são utilizados recursos existentes no OpenSER e no *gateway* Asterisk, o que não retrata um algoritmo e, por causa disso, esta solução de balanceamento será tratada como um mecanismo de balanceamento centralizado e não como um algoritmo.

A Figura 53 mostra a arquitetura VoIP, onde o mecanismo de balanceamento centralizado atua. O DSER nesta figura é o servidor de localização, além disso, existe o DNS privado, que guarda as informações dos IP dos servidores VoIP e os prefixos que eles podem encaminhar.

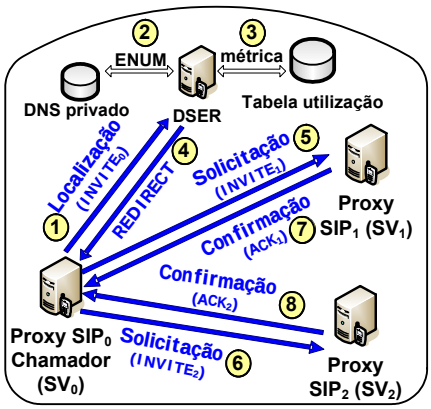


Figura 53. Arquitetura para o mecanismo centralizado

De uma forma geral, o DSER conhece a utilização dos canais de voz dos *gateways* associados aos SVs das instituições, podendo então tomar decisões de balanceamento. Estas decisões são realizadas da seguinte maneira: a lista de servidores que podem encaminhar uma determinada chamada é transmitida pelo DSER para o chamador e ela possui, além dos endereços IPs dos servidores, os pesos associados a eles, os quais definem o quanto um

servidor é mais propenso a receber chamadas do que os outros (prioridades), ou seja, quem estiver com uma utilização menor deve receber preferencialmente as chamadas.

Ao receber essa lista, o chamador contata o servidor de destino com maior prioridade e, somente se não houver possibilidade de encaminhar a chamada por ele, um outro, com menor prioridade, é contatado. Observe que, somente se os destinos com menor utilização de canais de voz dos seus *gateways*, não puderem encaminhar uma chamada, é que os SVs com menor prioridade têm chance de receber a chamada.

Quando dois ou mais *gateways* estão com a mesma utilização, o SV chamador envia simultaneamente para cada um dos seus servidores uma solicitação de chamada. Isso é chamado na literatura de *fork* paralelo (encaminhamento paralelo). Assim, estes SVs possuem, em parte, chances iguais de receber a chamada. Mas, lembre que, ao utilizar o *fork* paralelo para esses destinos com o mesmo grau de utilização dos seus GWs, os retardos ocasionados pela topologia determinam o servidor de destino.

Assim, para iniciar uma chamada, aqueles SVs com menor ocupação dos canais de voz dos seus *gateways* (1) são contatados primeiro, e somente em caso de falha nesta tentativa de início de chamada é que os SVs com maior utilização (1) recebem alguma sinalização. Isso se chama *fork* seqüencial (encaminhamento seqüencial).

Em resumo, nesse mecanismo, as instituições precisam enviar para um BD próximo ao DSER as utilizações dos seus *gateways* VoIP/PBX segundo (1), que precisam ser atualizadas com freqüência pelas instituições. Caso o tempo médio de recebimento de chamadas externas destinadas à RTFC seja conhecido, uma instituição pode ajustar a freqüência de atualização da sua métrica usada para realizar o balanceamento.

Quando um servidor pretende encaminhar uma chamada para outra instituição, uma solicitação de chamada é enviada ao DSER (Figura 53-1) e, ao recebê-la, ele consulta o DNS privado (Figura 53-2) para obter quem pode encaminhar esta chamada (servidores de destino). Em seguida, de posse dos endereços IP destes SVs, o DSER pode consultar o banco de dados, a fim de extrair o valor da utilização dos *gateways* pertencentes aos servidores das instituições (Figura 53-3). Então, a partir desse momento, o chamador pode ser informado na sinalização de redirecionamento (Figura 53-4) quais são os servidores mais propensos a receber uma chamada. Quando o servidor chamador receber essa sinalização, um encaminhamento seqüencial é realizado, para que um dos servidores de destino encaminhe a chamada (Figura 53-5 e Figura 53-7).

7.2 Implementação do controle de chamadas

Nesta seção é descrito, com mais detalhes, o mecanismo de balanceamento centralizado da Figura 53, utilizando a Figura 54. Na primeira subseção, é mostrada o que a estrutura VoIP de uma instituição precisa para usufruir do balanceamento centralizado; a subseção seguinte aborda as alterações necessárias no DSER para este balanceamento operar.

Uma chamada destinada à RTFC é encaminhada para o PBX pelo *gateway* de voz, que faz a conversão da sinalização entre o ambiente VoIP e o PBX. Isso implica que esse *gateway* tem controle total sobre as chamadas em andamento. O *gateway* utilizado no estudo de viabilidade desse balanceamento é o Asterisk, que, por exemplo, possui suporte para placas do tipo FXO, FXS e E1 [13]. Nesse *gateway* existem funcionalidades que não foram implementadas ou previstas e, para contornar isto, um recurso chamado AGI (*Asterisk Gateway Interface*) [12 e 41] pode ser utilizado. Através desse recurso, ganha-se mais flexibilidade para operar novas funcionalidades.

No Asterisk, o andamento de uma chamada pode ser dividido em duas etapas bem distintas, que são: início da chamada e fim da chamada. Quando qualquer uma destas etapas está sendo iniciada, é possível executar um programa externo ao Asterisk, através do AGI, que pode ser desenvolvido em qualquer linguagem de programação, seja ela compilada ou interpretada. Além disso, nesse capítulo, os programas executados pelo AGI serão chamados de programas de balanceamento; os códigos desses programas e as configurações do Asterisk para executá-los encontram-se no Apêndice B.

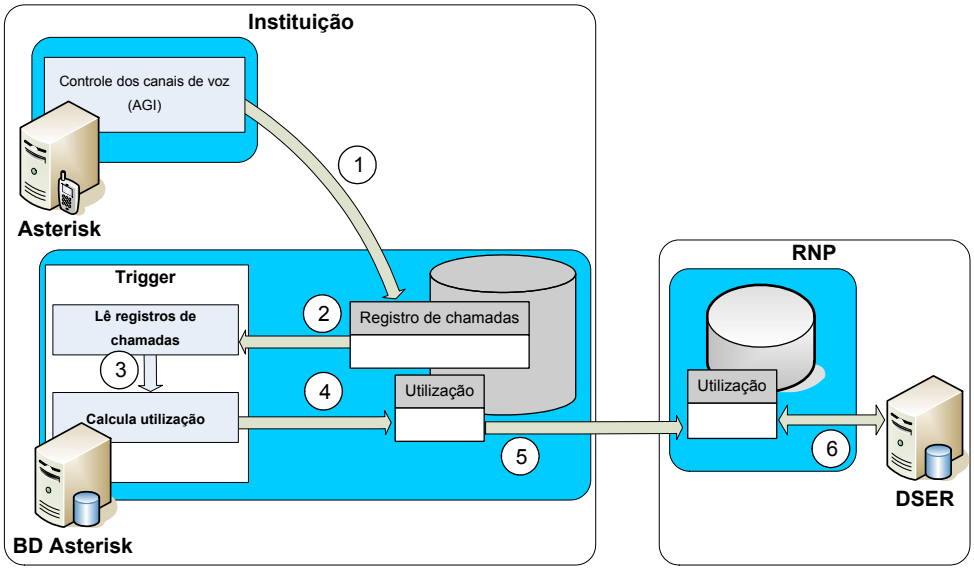


Figura 54. Arquitetura do controle centralizado

No momento em que o *gateway* estiver encaminhando uma chamada à RTFC (etapa início da chamada), a funcionalidade AGI executa um programa de balanceamento, que insere uma linha em uma tabela localizada em um BD local, chamada **tabela de registro de chamadas** (Figura 54-1), fazendo o registro da chamada que está sendo iniciada. Nesta linha existem informações identificando a chamada, que permitem defini-las como chamadas internas e externas. Além disso, quando uma chamada for terminada, o AGI executa um outro programa de balanceamento, que apaga da tabela o registro da chamada que está sendo finalizada, refletindo com fidelidade a quantidade chamada em andamento. Extraindo as

informações da tabela de registro de chamadas, que estão discriminadas em chamadas internas e externas, a utilização pode ser calculada pela Fórmula (1).

Vale observar que a tabela de registro de chamadas apenas possui dados sobre as chamadas destinadas à RTFC em andamento, sejam elas internas ou externas. Assim, uma chamada envolvendo apenas usuários VoIP não ativa o AGI e como conseqüência, seus dados não são enviados para essa tabela.

O cálculo da utilização é realizado por um *trigger* (gatilho), que é ativado quando um registro de chamadas é inserido ou removido da tabela de registro de chamadas (Figura 54-3). Por fim, o *trigger* armazena o resultado deste cálculo na **tabela utilização** (Figura 54-4), que posteriormente é transmitido para um banco de dados localizado próximo ao DSER. O código fonte deste gatilho e a estrutura das tabelas mencionadas estão disponíveis no apêndice B.

A informação sobre utilização dos canais de voz com o PBX é replicada [41] para um banco de dados central próximo ao DSER. Por fim, o DSER, com base nas utilizações dos *gateways* que foram replicadas, pode tomar as decisões de balanceamento.

Nas instituições, em cada linha da tabela de registro de chamadas existem informações como a quantidade de canais de voz (capacidade), identificação do canal de voz utilizado pela chamada, identificação do processo do Asterisk, identificação se a chamada é externa ou interna e uma identificação do *gateway*. Este nível de detalhamento de uma chamada é necessário para o cálculo da métrica de utilização.

A identificação do canal de voz é importante no término da chamada, pois permite que apenas a linha referente à chamada finalizada seja removida, o que pode ser feito na etapa de fim da chamada.

Quando um *gateway* Asterisk é interrompido e depois iniciado, o AGI não é executado, e os registros das chamadas em andamento não podem ser removidos da tabela de registro de chamadas, resultando que o cálculo com estas informações não reflita a real utilização dos canais de voz. Assim, identificar os registros das chamadas realizados antes da última inicialização do Asterisk é necessário. Isto pode ser realizado graças ao identificador do processo do Asterisk, que é modificado pelo sistema operacional a cada execução de um programa, e por causa disto ele é armazenado em uma das colunas da tabela de registro de chamadas. No momento em que o AGI executar o programa de balanceamento, seja no início ou no término da chamada, captura-se o identificador corrente do processo do Asterisk, e qualquer chamada na tabela de registro de chamadas que contenha um identificador diferente do atual é apagado.

Com esse procedimento simples, os registros contidos na tabela de registro de chamadas retrata, com total fidelidade, a quantidade de chamadas em andamento nos *gateways* VoIP/PBX de uma instituição, e o cálculo da utilização pode ser realizado, sem levar em considerações informações erradas.

A identificação do *gateway* serve apenas para discriminar mais de um *gateway* VoIP/PBX, que é importante para não remover um registro referente à chamada em andamento de outros *gateways*.

A última informação da tabela de registro de chamadas é a identificação das chamadas em internas e externas, que, juntamente com a capacidade do *gateway*, permite o cálculo da métrica de utilização mostrada na Fórmula (1).

Como o *gateway* tem controle total sobre a chamada, no seu término, o AGI executa um programa para debitar o registro relacionado a uma chamada na tabela de registro de chamadas, o que é feito removendo desta tabela uma linha de registro apenas.

7.2.1 Detalhamento do mecanismo de balanceamento nas instituições

Nessa seção, detalhes da arquitetura que envolve a infra-estrutura das instituições no balanceamento centralizado são colocados. A Figura 54 ajuda no entendimento do texto, mostrando esquematicamente o mecanismo de balanceamento de chamadas centralizado, onde existe o DSER, tomando as decisões de balanceamento, e uma instituição que possui um serviço VoIP.

A analise da viabilidade desse mecanismo de balanceamento ocorreu para o ambiente VoIP da RNP (fone@RNP), assim, quando for relevante no texto, as distribuições e versões dos *softwares* serão mencionadas.

A Figura 54 mostra uma instituição com um *gateway* Asterisk e um BD (Banco de Dados), no qual as informações sobre as chamadas em andamento são registradas. Este banco de dados também pode ser designado de BD Asterisk, conforme essa figura.

Cada chamada encaminhada ou terminada pelo *gateway VoIP/PBX* (Asterisk) provoca, respectivamente, uma inserção ou remoção de registro (Figura 54-1) na tabela "registro de chamadas" do BD Asterisk, que será realizada pelo programa executado por meio do recurso AGI. Quando o Asterisk encaminha a chamada à RTFC, existe a inserção de um registro identificando-a, e ele contém uma classificação, se a chamada é interna ou externa. No término, o registro desta chamada é apagado, para que a tabela expresse a quantidade de chamadas em andamento.

Após qualquer modificação na tabela, através de uma inserção ou remoção, um gatilho é acionado (Figura 54-2 e Figura 54-3) e sua função é extrair da tabela informações das chamadas em andamento e contabilizar quantas chamadas internas e externas existem e, depois realiza-se o cálculo da utilização (Figura 54-3) pela Fórmula (1). O resultado deste

cálculo substitui o valor corrente da tabela de utilização (Figura 54-4), garantindo que o valor da utilização disponível nesta tabela não esteja desatualizado.

O resultado do cálculo da utilização é único para uma instituição, ou seja, mesmo que ela tenha vários *gateways*, a Fórmula (1) apenas considera o total de chamadas internas, chamadas externas e o somatório das capacidades dos *gateways*. Em função disso, é pressuposto que a tabela utilização contém a utilização da instituição, que é replicada para um BD próximo ao DSER (Figura 54-5), permitindo que este servidor de localização possa obter rapidamente o valor da utilização instantânea dos *gateways* VoIP/PBX das instituições.

O mecanismo de replicação citado é muito simples: em intervalos de tempo configuráveis, o conteúdo de uma tabela pertencente a uma instituição pode ser atualizado em um banco de dados em outro ponto da rede.

Agora, imagine que uma solicitação de localização seja encaminhada ao DSER (Figura 53-1), a fim de obter os destinos que podem encaminhar a chamada. Então, este servidor envia uma solicitação ENUM para o DNS privado (Figura 53-2), para obter quais são os SVs que podem encaminhar a chamada; em seguida, se a chamada for destinada à RTFC, o DSER aplica alguma decisão de balanceamento, após consultar o estado da utilização dos *gateways* na tabela utilização (Figura 53-3 ou Figura 54-6). Esta consulta é realizada pelo módulo *enum_query()* do OpenSER, que teve o seu código fonte modificado.

Inicialmente, o módulo *enum_query()* realizava apenas consultas ENUM ao DNS privado, para obter quem são os SVs destinos. Como foi mencionado, é desejável que a utilização dos *gateways* VoIP/PBX das instituições sejam conhecidas, para o DSER definir prioridades no encaminhamento das chamadas. Em função disto, o módulo *enum_query()* foi alterado, possibilitando, então, ao DSER consultar a utilização armazenada em uma tabela em

um BD (Figura 53-3) e definir prioridades para o encaminhamento das chamadas (Figura 53-3).

Em seguida, o chamador recebe a primitiva REDIRECT (Figura 53-4) enviada pelo DSER, com a relação dos SVs que podem encaminhar a chamada e as suas prioridades, possibilitando que o servidor chamador realize um encaminhamento seqüencial.

7.2.2 Detalhes sobre o DSER

O funcionamento do módulo *enum_query()* que realiza consultas ENUM no DNS privado e, depois obtém junto ao banco de dados a utilização do GWs associados aos servidores de destino (instituições), encontra-se na Figura 55, em um diagrama de blocos, composto pelos blocos **ENUM** e **Função de Balanceamento**. O terceiro bloco, chamado **REDIRECT**, pertence a outro módulo, que não sofreu alteração no código fonte.

De uma forma geral, o módulo *enum_query()* permite o DSER realizar a consulta de DNS (Figura 55-3 e Figura 55-4), para conseguir a relação dos SVs que podem encaminhar uma chamada. Este módulo não sofreu alteração e na Figura 55 ele é chamado de bloco **ENUM**.

O bloco **Função de Balanceamento** foi incorporado, com a modificação de código fonte do módulo *enum_query()*; a sua tarefa é consultar uma tabela em um banco de dados, para obter a utilização dos *gateways* das instituições de destino da chamada (Figura 55-6 e Figura 55-7). Por último, com base nos valores de utilização, prioridades são definidas e na primitiva *redirect* (Figura 55-8), elas são correlacionandas aos seus respectivos servidores de destino, que é enviada para o chamador no bloco **REDIRECT**. Quando o SV chamador receber esta primitiva, inicia-se a tentativa de *fork* seqüencial da chamada.

Maiores prioridades são dadas às instituições com menores utilizações dos *gateways*, dando a elas maiores chances de encaminhar as chamadas, pois o fork seqüencial é utilizado para isso. Aquelas instituições com menores prioridades indicam que os seus *gateways* estão com alto grau de utilização e, precisam ter chances reduzidas de receber a chamada.

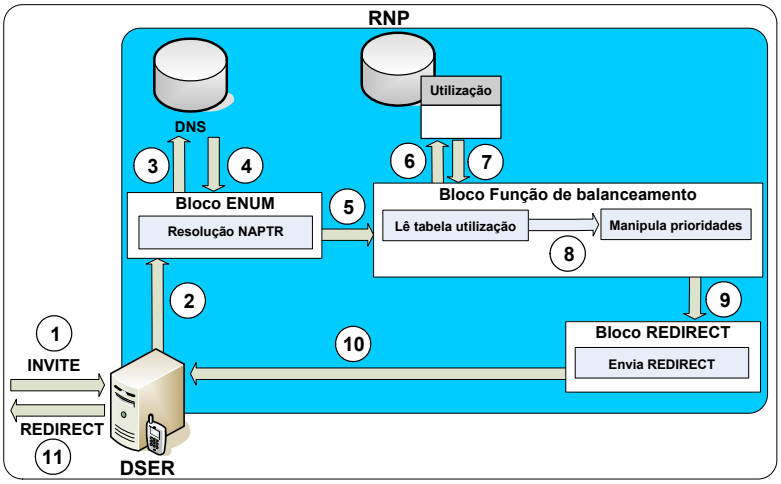


Figura 55. Função balanceamento no DSER

7.3 Comentários

Olhando pelo lado de uma futura implementação no fone@RNP, neste mecanismo de balanceamento pouca modificação de código no ambiente SIP com OpenSER é necessária, facilitando a sua implementação. Já no ambiente H.323 com GNUGK, muitas das funcionalidades necessárias para este algoritmo funcionar precisariam ser implementadas.

Como a decisão de balanceamento define prioridades para os servidores VoIP, não existem atrasos extras , ao contrário do que vinha acontecendo nos algoritmos dos capítulos 3, 4 e 5.

Capítulo 8

CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação apresentou propostas para balanceamento de chamadas externas VoIP destinadas à RTFC de uma cidade, sendo isto possível de forma distribuída ou centralizada. As instituições que atuam como provedores VoIP encaminham essas chamadas para a rede de telefonia das suas cidades, que podem utilizar tanto os protocolos SIP como H.323 e, por causa disso, os algoritmos precisam atender aos dois ambientes.

A motivação para o balanceamento de chamadas existe, porque uma chamada externa tende a ser enviada apenas para um conjunto de instituições, pois, como foi exposto, no VoIP, o servidor que tiver sua confirmação a uma solicitação de chamada chegando primeiro ao iniciador da chamada (chamador) é o escolhido para o seu encaminhamento, o que é função da topologia da rede.

Esses provedores VoIP possuem recursos limitados de canais de voz, o que fixa a quantidade de chamadas simultâneas encaminhadas pelos seus *gateways* VoIP/PBX. Em momentos com elevado número de chamadas externas, os *gateways* que tendem a recebê-las acabam chegando à saturação e, a partir deste instante, qualquer chamada que chegue não é completada. Sempre um conjunto de *gateways* tende a receber as chamadas, porque o mecanismo de escolha do servidor chamado favorece aquele que tiver a sua primitiva de confirmação chegando primeiro ao chamador, o que é dependente da topologia.

As chamadas não completadas por uma instituição (servidor VoIP) jamais são encaminhadas para outras, até mesmo que os seus *gateways* tenham canais de voz disponível, o que leva a uma subutilização destes recursos. Para que isso não aconteça, uma alternativa é utilizar balanceamento de chamadas.

Esta seção está organizada da seguinte forma: na seção 8.1 são apresentadas as principais contribuições da dissertação; na seção 8.2 pontos positivos e negativos dos algoritmos são apresentados; por fim, na última seção, possíveis trabalhos futuros são apresentados.

8.1 Principais contribuições

A maior contribuição é oferecer uma gama de opções para o balanceamento de chamadas VoIP, sendo assim possível selecionar o melhor algoritmo para cada arquitetura.

Os algoritmos mostrados neste documento podem atender os mecanismos e as arquiteturas apresentados no capítulo 2 mas, antes, a escolha do algoritmo precisa ser avaliada, com o propósito de verificar sua aderência.

- **Balanceamento distribuído para redes com *jitter* conhecido:** em redes onde o *jitter* máximo é facilmente estimado ou determinado, o algoritmo otimizado pode ser utilizado. Com adoção desse algoritmo, ganha-se pelo fato de o algoritmo não gerar atrasos elevados para o balanceamento; consequentemente, a admissão de chamadas não é muito prejudicada.

- **Balanceamento distribuído independente do *jitter* da rede:** O algoritmo utilizando lógica nebulosa não depende do *jitter* máximo da rede, além disso, a variável razão pode ter um valor aproximado. Esse tipo de algoritmo atende a redes em que o *jitter* varia bastante ou sua estimação pode não ser precisa.

- **Balanceamento distribuído minimizando o atraso:** O algoritmo otimizado, utilizando o servidor virtual, possibilita o atraso mínimo para admissão da chamada quando a utilização atinge o valor *A*. Todavia, o *jitter* máximo da rede no trecho entre os servidores chamados e POP precisa ser conhecido.

- **Uso de encaminhamento E.164 no SIP:** no decorrer do desenvolvimento da proposta do mecanismo centralizado, a configuração do *proxy* SIP OpenSER foi modificada, para viabilizar encaminhamento E.164, que utiliza DNS com ENUM.

- **Mecanismo de balanceamento centralizado:** foi apresentada uma possibilidade para realização de balanceamento centralizado, que pode ser opção, contanto que o ponto de central não seja suscetível a falha. Num cenário de localização centralizada, o desempenho deste algoritmo é ótimo; assim, quando o atraso na admissão da chamada não é evitável, ele é uma ótima opção. Também, quando a topologia da rede não permite valer-se de um ponto de referência, o uso do mecanismo centralizado seria indicado. Foi mostrada a viabilidade de implementação do mecanismo centralizado integrado com a sinalização SIP.

8.2 Características das formas de balanceamento

A média dos atrasos e os atrasos máximos para admissão de chamada foram observados para o algoritmo otimizado e, como já era esperado, com base nas deduções matemáticas, neste o atraso para admissão da chamada é menor do que o algoritmo com SV_v, mas o *jitter* máximo precisa ser conhecido ou estimado com freqüência.

Um aspecto negativo do algoritmo sem ponto de referência é a sensibilidade à variação de retardo, que deve ser estimado com periodicidade, ou usado um valor bem conservador. Por outro lado, o algoritmo com ponto de referência produz um atraso excessivo.

Apesar de uma estimação do *jitter* ser possível, como descrito, deve-se considerar que a subestimação da variação de retardo pode causar erros no balanceamento. Quando o percurso entre servidores VoIP compreende apenas *backbones* de alta performance, que é uma realidade atualmente, o *jitter* tende a ser pequeno.

O algoritmo com lógica nebulosa pode ter um atraso máximo próximo às soluções com ponto e sem ponto de referência, o que depende muito do parâmetro *T*.

Em alguns cenários onde a capacidade dos *gateways* é pequena, um valor de *T* próximo ao atraso adicional máximo do algoritmo otimizado não permite uma boa distribuição de chamadas. O valor de *T* deve permitir que os SV_1 e SV_n se alinhem quando a utilização do primeiro atinge valores menores que 30%, fazendo com que o último servidor participe do balanceamento, quando os outros somam poucas chamadas externas em andamento.

A grande vantagem do algoritmo utilizando lógica nebulosa é que não existe nenhuma dependência em relação à variação de retardo Δ_{rede} e a variável de entrada razão pode ter um valor aproximado, não necessitando de medidas exatas. A desvantagem da lógica nebulosa é que, em alguns cenários, as semânticas dos conjuntos nebulosos podem precisar de ajustes. Além disso, o comportamento do balanceamento pode não ser previsível.

Através de cenários com poucos canais nos *gateways*, verifica-se que, utilizando lógica nebulosa, o atraso é maior do que o algoritmo otimizado, já, quando os números de canais aumentam, dependendo do *jitter* máximo, estes algoritmos geram atrasos da mesma ordem de grandeza.

Um ponto positivo do mecanismo centralizado é que a modificação de código no OpenSER é mínima, e ele pode ser implementado sem encontrar adversidades. Além disso,

para viabilizar o seu funcionamento, com exceção da modificação de código citada, são utilizadas apenas soluções padrão, como o servidor DNS com ENUM, recursos do banco de dados e replicação de tabelas de banco de dados.

No mecanismo centralizado não existem atrasos quando a primeira tentativa de encaminhamento não apresenta falhas, fazendo a sua performance ser ótima, mas o balanceamento depende da rapidez e acuidade da atualização de informações na entidade central, o que constitui um ponto crítico.

8.3 Trabalhos futuros

Para o algoritmo que utiliza lógica nebulosa, como obter, sem intervenção humana, o valor T merece ser investigado. Seria interessante que o próprio algoritmo pudesse aperfeiçoar as semânticas (alteração dinâmica) dos conjuntos nebulosos das variáveis de entrada e saída, assim haveria um aprendizado baseado na topologia. Para estes dois focos levantados, uma realimentação do algoritmo com lógica nebulosa seria necessária.

As simulações realizadas nesta dissertação utilizaram retardos determinísticos dos enlaces, verificando apenas o funcionamento dos algoritmos. É importante que simulações estocásticas sejam consideradas para verificr o comportamento do algoritmo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ITU-T E.164 Telecomunication Standardization Sector of ITU (05/1997), series E: The international public telecommunication numbering plan. http://www.vsi.ru/library/ITU-T/. Site acessado a última vez em janeiro de 2007.

[2] ALBUQUERQUE, A. A., Rodrigues, P.H.A., "Balanceamento de Chamadas VoIP H.323 para a Rede de Telefonia Pública", 24° Simpósio Brasileiro de Redes de Computadores (SBRC'2006), Curitiba, PR – Brasil, Maio/Junho 2006.

[3] GHANEM, J., "Implementation of Load Balancing Policies in Distributed Systems", The University of New Mexico, New Mexico, dissertação de mestrado. June, 2004. Site http://www.eece.unm.edu/lb/papers/jeanthesis.pdf consultado em fevereiro de 2006.

[4] RFC 3261, "SIP: Session Initiation Protocol". IETF, Jun. 2002.

[5] FLOYD, S. e Jacobson V., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on networking. vol I . No 1. August 1993.

[6] CHANG, Cheng-yue; CHEN, Ming-syan; HUANG, Pai-han. An H.323 Gatekeeper Prototype: Design, Implementation, and Performance Analysis. IEEE Transactions On Multimidia, p. 936-946. dez. 2004.

[7] ITU-T H.323 Telecomunication Standardization Sector of ITU (07/2003), series H: AudioVisual and Multimidia Systems.

[8] ITU-T H.225 Telecomunication Standardization Sector of ITU (11/1996), series E: Media stream packetization and synchronization on non-guaranteed quality of service LANs. http://www.vsi.ru/library/ITU-T/. Site acessado a última vez em janeiro de 2007.

[9] RFC 3665. Session Initiation Protocol (SIP) Basic Call Flow Examples. IETF, Dez. 2003.

[10] RFC 3666. Session Initiation Protocol (SIP) Public Switched Telephone Network (PSTN) Call Flows. IETF, Dez. 2003.

[11] Proxy SIP OpenSER, http://www.openser.org/, site acessado em outubro 2006.

[12] DIGIUM. Site do ASTERISK. Disponível em: <http://www.asterisk.org>. Acesso em: maio 2006.

[13] MAHLER, Paul. "VoIP Telephony with Asterisk". Editora Signate. ISBN: 0975999222. 2005.

[14] Gatekeeper GNUGK, http://www.gnugk.org/, site acessado em outubro 2006.

[15] GOGFREY, Brighten, Surana, Sonesh, et al. "Load Balancing in Dynamic Structures P2P Systems", IEEE INFOCOM. 2004.

[16] ITU-T E.721 Telecomunication Standardization Sector of ITU (1991), series E: Network grade of service parameters and target values for circuit-switched services in the evolving ISDN. http://www.vsi.ru/library/ITU-T/. Site acessado a última vez em janeiro de 2007.

[17] DURKIN, James F."Voice Enabling tha data Network: H.323, MGCP, SIP, QoS, SLAS, and Security". Cisco Press. 2002.

[18] GUIZANI, Mohsen, et al. "Wirelesss Communications System and Networks". Kluwer Academic. 2004.

[19] Módulo LCR do OpenSER, http://www.openser.org/docs/modules/1.0.x/lcr.html, site acessado a última vez em setembro de 2006.

[20] JEAN-PIERRE, Petit ."IP Telephony Deploying Voice-Over-IP Protocols". JOHN WILEY PROFESSIO. 2005.

[21] OpenOSP Product Overview, http://www.vovida.org/applications/downloads/openosp/, site acessado a última vez em julho de 2006.

[22] OpenOSP Installation & Configuration Guide (guia de instalação e configuração do software OpenOSP), http://www.vovida.org/applications/downloads/openosp/, site acessado a última vez em julho de 2006.

[23] OpenOSP Interfaces, http://www.vovida.org/applications/downloads/openosp/, site acessado a última vez em julho de 2006.

[24] RFC 3764. Enumservice registration for Session Initiation Protocol (SIP) Addresses-of-Record. IETF, Abr. 2004.

[25] RFC 2915. The Naming Authority Pointer (NAPTR) DNS Resource Record. IETF, Set. 2000.

[26] Módulo ENUM do OpenSER, http://www.openser.org/docs/modules/1.0.x/enum.html, site acessado a última vez em setembro de 2006.

[27] HERSENT, Oliver, Guide, David, Petit, Jean-Pierre."Telefonia IP: Comunicação Multimídia Baseada em Pacotes". Editora Makron Books. ISBN 8588639025. 2001.

[28] LEÃO, Rosa M. M, SILVA, Edmundo S., et al. Estimando a média e variância do atraso em um sentido utilizando o IPID da máquina remota. 24° Simpósio Brasileiro de Redes de Computadores (SBRC'2006), Curitiba, PR – Brasil. maio/junho. 2006.

[29] MILLS, David L. “Precision synchronizaion of computer network clocks” ACM Compur. Commun. Rev., vol. 24, no. 2, 16 pp., Abril. 1994.

[30] MILLS, David L. "A Brief History of NTP Time: Memoirs of an Internet Timekeeper". ACM SIGCOMM Computer Communications Review. Páginas: 9-21. Volume 33, Issue 2 ISSN:0146-4833. 2003.

[31] BI, Jingping, Wu, Qi, Li, Zhongcheng. "On estimating clock skew for one-way measurements". Computer Communications - www.sciencedirect.com, Ago. 2005.

[32] Borzemski, Leszek, Zatwarnicki, Krzysztof, “A Fuzzy Adaptive Request Distribution Algorithm for Cluster-based Web Systems”. IEEE, 2003.

[33] VIANNA, Nilson Rocha, et al. “EWIDS: Uma Extensão para Arquiteturas de Sistemas de Detecção de Intrusos para Redes Sem Fio Metropolitanas”, 24° Simpósio Brasileiro de Redes de Computadores (SBRC'2006), Curitiba, PR – Brasil, Maio/Junho 2006.

[34] LAGES, Alexandre Gomes, et al.”Um sistema de Reputação Fuzzy para Segurança Orientada a Serviços em Redes de Banda Larga sem Fio”, 24° Simpósio Brasileiro de Redes de Computadores (SBRC'2006), Curitiba, PR – Brasil, Maio/Junho 2006.

[35] JUNIOR. Oliveira, Aguiar, Hime. “Lógica difusa: aspectos práticos e aplicações”. Rio de Janeiro. Editora Interciência, páginas 127-133. 1999.

[36] RESENDE, R. A., Rossi, S. M.”Traffic Engineering with MPLS Using Fuzzy Logic for Application in IP Networks”, The IEEE International Conference on Fuzzy Systems. 2003.

[37] KO, You-Chang, Park, Sun-Chun, et al.”An adaptive QoS Provisioning Distributed Call Admission Control Using Fuzzy Logic Control”, IEEE. 2001.

[38] ZAHIRAZAMI, B.Seyed, Yekrangian, Golnaz, Spencer, Mendel, “Load Balancing and Call Admission Control in UMTS-RNC, using Fuzzy Logic”. ICCT2003, IEEE, 2003.

[39] Cruz, A. Lógica Nebulosa. Disponível em http://equipe.nce.ufrj.br/adriano/fuzzy/, site acessado em outubro 2005.

[40] MATSUMOTO, Élia Yathie. “Matlab 6.5 - Fundamentos de Programação”. Editora Érica. ISBN 8571949204. 2003.

[41] The slony1 Project - Slony - A replication system for PostgreSQL, http://www.slony.org, site acessado em outubro de 2006.

Apêndice A
Implementação dos algoritmos distribuídos com o MatLab

Este apêndice descreve um protótipo no MatLab desenvolvido para realização de simulações dos algoritmos propostos.

Foi desenvolvido no MatLab um protótipo para realizar as simulações dos algoritmos e, na sua concepção utilizou-se a linguagem de *script* do MatLab, a ferramenta *Guide* (*MATLAB Graphical User Interface development environment*) e o FIS-Editor (*Fuzzy Inference System Editor*).

Esse FIS-Editor é uma ferramenta existente no MatLab, que permite ao usuário, através de uma interface gráfica, criar um sistema nebuloso. Outra facilidade do FIS-Editor é a de acompanhar graficamente o funcionamento do sistema nebuloso, através das variáveis de entrada fornecidas e dos valores de saída observados.

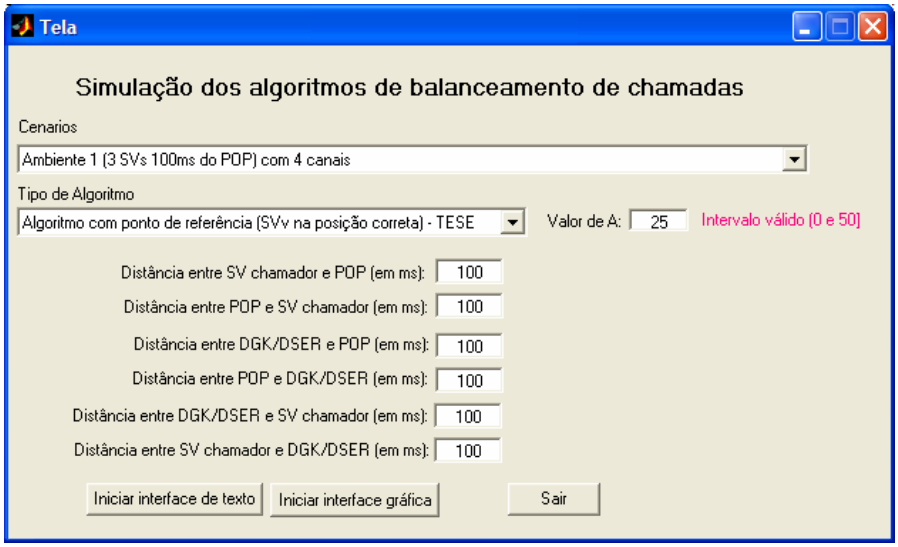


Figura 56. Interface gráfica

Na simulação o usuário utiliza uma interface gráfica desenvolvida com a ferramenta *Guide* do MatLab, Figura 56, para fornecer algumas informações, que são: os retardos dos enlaces entre os elementos da rede (SV chamador, POP e DSER/DGK), o cenário para

simulação (retardo entre os SVs chamados ao POP) e o algoritmo utilizado na simulação.

Na simulação são utilizados eventos que registram ações a serem executadas e, neles um retardo de rede ou atraso impingido pelo algoritmo são registrados. Assim, o próximo instante de execução do evento é conhecido.

No desenvolvimento, para facilitar a depuração de erros, algumas funções responsáveis por tarefas bem definidas foram criadas, como:

- Função principal: são carregadas as variáveis e matrizes necessárias para o funcionamento do programa de simulação;
- Interface gráfica: utilizada para o usuário informar a função principal, quais são retardos dos enlaces, algoritmo utilizado, valor de *A* (não é utilizado pelo algoritmo com lógica nebulosa) e cenário para simulação;
- Dispara chamada: uma solicitação para o servidor de localização referente a cada chamada destinada à RTFC é enviada;
- Calcula utilização: esta função calcula o atraso gerado pelo algoritmo. Dentro desta função existe um trecho de código para cada algoritmo de balanceamento;
- Geração de gráficos: no final da simulação os seus resultados são mostrados em gráficos, existe um com a distribuição das chamadas, para cada SV, e outro com os atrasos gerados pelo algoritmo, para cada chamada encaminhada;
- Geração de relatórios (*logs*): caso seja habilitada, a geração de registros sobre o que aconteceu durante a simulação é colocada em um arquivo;

- Localização do próximo evento: como os eventos registram em qual momento eles precisam ser ativados, através de uma busca na matriz de eventos, é possível localizar o próximo evento;

- Controlar alocação dos canais: o cálculo da utilização dos canais é necessária, assim cada chamada aceita é registrada. Além disso, caso alguma chamada chegue quando o *gateway* já está saturado, é feito um registro dela como chamada não completada;

- Registro de chamadas aceita ou não completadas: esta função armazena em variáveis o registro das chamadas encaminhadas para um *gateway*;

- Carrega cenário: um cenário é escolhido pelo usuário através da interface gráfica e nesta função, as variáveis e matrizes necessárias para a simulação ser realizada no cenário escolhido são carregadas.

Na configuração do *FIS-Editor*, é possível definir as condições para a "**desfuzificação**", que são os métodos utilizados: mínimo como "e" e máximo como "ou". Em contrapartida, para a implicação e agregação, foram utilizados o mínimo e máximo, respectivamente. Na "***desfuzificação***", o modelo Mandami foi utilizado e a abordagem com bissetores ou centróides pode ser configurada.

Os resultados obtidos com a simulação são mostrados em gráficos e eles podem ser de dois tipos: o primeiro gráfico mostra a distribuição de chamadas e o outro os atrasos gerados pelo algoritmo.

Cada um dos tipos de gráfico está em uma figura diferente e, em qualquer uma delas, existe um título indicando qual foi o algoritmo utilizado na simulação e, um exemplo disto é a Figura 57, onde é utilizado o algoritmo de balanceamento com lógica nebulosa, conforme descrito pelo seu título.

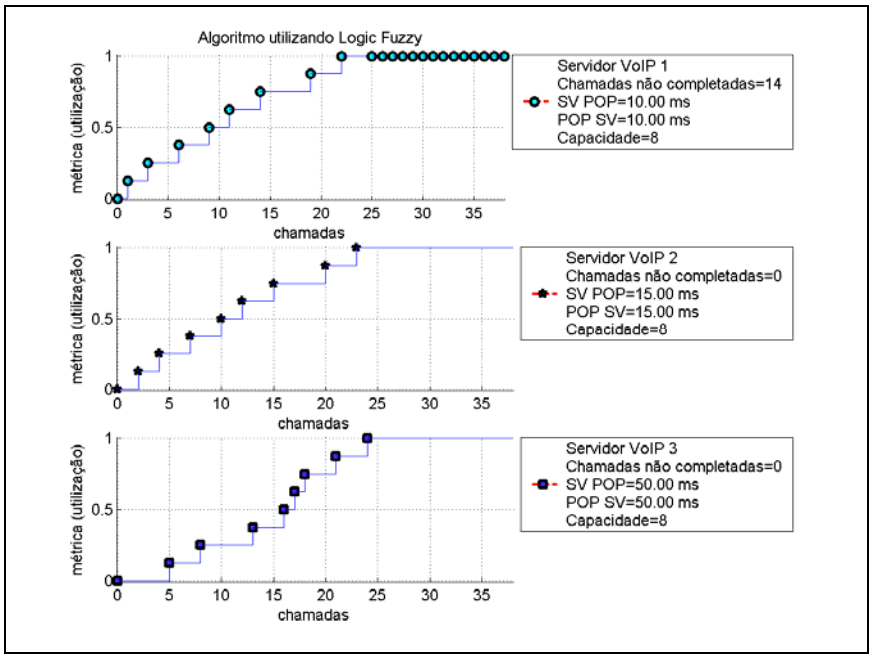


Figura 57. Utilização versus chamadas

Cada SV possui um gráfico com a sua distribuição de chamadas, que são mostrados juntos em uma única figura, como na Figura 57 onde existem três SVs. O eixo das abscissas mostra a ordenação das chamadas encaminhadas (primeira, segunda, terceira, etc); na ordenada estão os valores da métrica (2), que pode variar de zero a um. Assim, quando a primeira chamada externa for encaminhada para a RTFC, ela é marcada com um ponto no gráfico do SV que a encaminhou. Por exemplo, na Figura 57, quem encaminha a quinta chamada é o SV_3, então, quando a abscissa tem valor cinco, apenas o gráfico do SV_3 possui a marcação de um ponto.

Cada um dos gráficos da Figura 57 tem uma legenda no seu lado direito, que mostra a quantidade de chamadas não completadas, capacidade do *gateway*, identificação do servidor VoIP e retardo de rede entre o SV e o POP.

No gráfico da Figura 57, quando um *gateway* encontra-se saturado, a marcação da chamada encaminhada é feita em $U_i = 1$. Como uma chamada já aceita tem duração infinita, qualquer uma encaminhada após esta é considerada como não completada. Nessa figura, o

SV_1 encontra-se com todos os seus oito canais ocupados ao receber a vigésima primeira chamada; assim, qualquer outra que chegue depois é considerada como não completada.

Na Figura 58, é mostrado um gráfico com os atrasos adicionais utilizados pelo algoritmo de balanceamento (ordenada) e, a quantidade de chamadas encaminhadas está no eixo das abscissas. No topo desta figura existe um título, que indica o algoritmo aplicado para a realização do balanceamento de chamadas.

Duas informações existem neste gráfico, uma é o atraso gerado pelo algoritmo para atrasar a primitiva. A segunda informação é a média aritmética de todos os atrasos gerados pelo algoritmo para a chamada sendo encaminhada, ou para as que foram encaminhadas. Cada ponto de referência do gráfico mostra uma chamada encaminhada e seu atraso, mas para saber qual SV gerou o seu atraso, é necessário localizá-la na figura dos gráficos de distribuição de chamadas.

Por exemplo, no gráfico da Figura 58 a quarta chamada sofre um atraso de, aproximadamente, 0.08 segundos e as médias dos atrasos até esta chamada são de, 0.05 segundos.

Outra forma de acompanhar o resultado é através de relatório (*log*), que pode ou não ser gerado. A opção de gerá-lo existe, mas como padrão esse relatório não é gerado porque as operações de escrita envolvidas aumentam excessivamente o tempo gasto na simulação.

Os cenários utilizados na simulação procuram retratar ambientes reais e fictícios, assim é possível observar o comportamento dos algoritmos na distribuição das chamadas e no atraso para admissão das chamadas. Isso é importante para constatar a eficácia de um algoritmo.

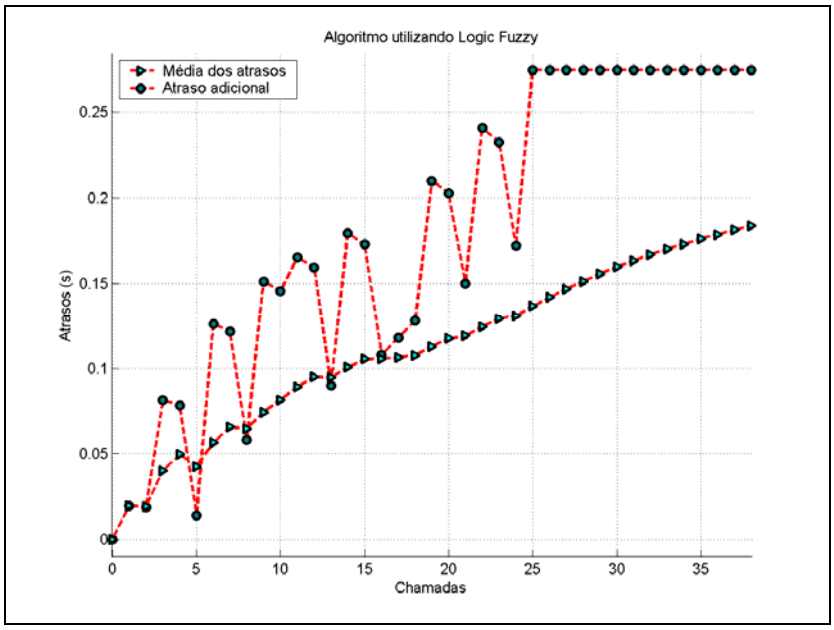


Figura 58. Atrasos versus chamadas

Nas simulações, as chamadas possuem duração infinita a partir do seu encaminhamento através de um *gateway*. Se uma chamada fosse terminada em vez de ter duração infinita, pelo cálculo da métrica em (1), a utilização de um *gateway* apenas retornaria a um estado anterior. Assim, simplesmente aumentaria a probabilidade deste *gateway* de encaminhar chamadas.

Apêndice B
Implementação no Asterisk do mecanismo centralizado

No balanceamento centralizado, o Asterisk pode executar algum programa, o que é feito através de um recurso chamado AGI (*Asterisk Gateway Interface*). Foram desenvolvidos dois programas, um para registrar informações em um banco de dados quando as chamadas são iniciadas e outro quando elas são terminadas. Além disso, o banco de dados precisa ter uma tabela para que esse programa registre as chamadas, além de um gatilho que calcula a utilização, inserindo-as em outra tabela. Abaixo são mostrados os códigos e as configurações necessárias para viabilizar o que foi descrito neste parágrafo.

- ✓ Este tópico tem a configuração do arquivo *extensions.conf* do Asterisk. É definido nesse arquivo que, quando uma chamada é encaminhada, o programa *balancer-start.agi* é executado, realizando um registro referente a esta chamada na tabela *active_calls*. No término de uma chamada o arquivo *balancer-stop.agi* é executado, removendo, na tabela *active_calls* o registro referente à chamada finalizada. Ambos os programas foram desenvolvidos na linguagem PHP (*PHP: Hypertext Preprocessor*).

```
exten => _XXXX,1,agi,balancer-start.agi
exten => _XXXX,n,Dial()
exten => _XXXX,n,Deadagi,balancer-stop.agi
exten => _XXXX,n,Hangup()
exten => h,1,DeadAgi,balancer-stop.agi
exten => h,2,Hangup()
```

- ✓ Abaixo é mostrado o arquivo *balancer-start.agi*. Ele registra na tabela *active_calls* informações sobre a chamada externa encaminhada.

```
#!/usr/bin/php4 -q
<?php
        $MAQ1 = "146.164.247.240"; // endereço da máquina 1
        $MAQ2 = "146.164.247.235";// endereço da máquina 2
        $canais=30; // Quantidade de canais dos gateways VoIP/PBX
        $gw = "0"; // Identificação deste gateway
```

```
//na tabela de balanceamento do sql?
     if ($fpid = fopen("/var/run/asterisk.pid", "r")){//utilizado para verificar se o GW foi reiniciado
                $a = fscanf($fpid, "%s");
                $pid = $a[0];//armazena o PID que será armazenado na tabela
                fclose($fpid);
          }
          else $pid = 0;
//se o PID receber 0 o que acontece?
          $sql_start = "DELETE from active_calls where pid != '".$pid."' AND gw = '".$gw."'";//caso existam registros de outra inicialização do GW
          $rdel_start = select($sql_start);//executa o delete
                function select($sql){ // seção para conexão no BD
                $bda = pg_connect("dbname=openser user=openser password=voipix");
                        $sel = pg_query($bda, $sql);
                        return $sel;
                }
                function numrows($sqlr){// seção para conexão no BD
                $bdr = pg_connect("dbname=openser user=openser password=voipix");
                        $selt = pg_query($bdr, $sqlr);
                        $qnt  = pg_num_rows($selt);
                        return $qnt;
                }
          // O Sistema
          set_time_limit(0);
          require('phpagi.php');
          $agi = new AGI();
          $channel = str_replace("\n", "", $agi->request['agi_channel']);
          $tech = str_replace("\n", "", $agi->request['agi_tipo']);
          $ip = substr($agi->request['agi_accountcode'], 1);
          $ip = str_replace("\n", "", $ip);
          $sql_location = "SELECT * from location where contact ~* '".$ip."'";//prepara select
          $location = select($sql_location);//faz select
          $qnt  = pg_num_rows($location);
          //$agi->verbose("Quant: ".$qnt);
          if ($qnt != 0) $tipo_origem = 0;
          else {
                if ($ip == $MAQ1 || $ip == $MAQ2 || $tech != "SIP") $tipo_origem = 0;
                else $tipo_origem = 1;
          }
          $sql = "INSERT into active_calls (id, tipo_origem, pid, gw, capacidade) values ('".$channel."','".$tipo_origem."','".$pid."','".$gw."','".$canais."')"; //prepara p/ inserir
          $rstInsere = select($sql);// faz insert
 ?>
```

- ✓ Abaixo é mostrado o arquivo *balancer-stop.agi*, fazendo com que, na tabela *active_calls*, o registro referente à chamada externa terminada seja removido.

```
#!/usr/bin/php4 -q
 <?php
function select($sql){ //infos de acesso ao banco
        $bda = pg_connect("dbname=openser user=openser password=voipix");
        $sel = pg_query($bda, $sql);
        return $sel;
}// O Sistema
set_time_limit(0);
require('phpagi.php');
$agi = new AGI();
$channel = str_replace("\n", "", $agi->request['agi_channel']);
$sql = "DELETE FROM active_calls WHERE id = '".$channel."'"; //prepara delete
$rstInsere = select($sql);//faz delete.
// FAZER calculo da utilização
?>
```

- ✓ A tabela *active_calls* do banco de dados é utilizada pelos programas *balancer-start.agi* e *balancer-stop.agi*, e a especificação desta tabela é descrita abaixo.

Coluna	Tipo	Valor
id	inteiro	acc_code
tipo_origem	inteiro	0 - propria instituição (chamada interna) 1 - outra instituição (chamada externa para a RTFC)
time	hora	now()
pid	inteiro	num_pid_asterisk
gw	inteiro	num_id_gateway
capacidade	inteiro	-

Essa tabela pode ser criada pelo comando:

```
CREATE OR REPLACE TABLE "public"."active_calls" (
        "id" VARCHAR(80) NOT NULL,
        "tipo_origem" INTEGER DEFAULT '-1' NOT NULL
        "time" TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
        "pid" INTEGER DEFAULT '-1' NOT NULL,
        "gw" INTEGER DEFAULT '-1' NOT NULL,
        "capacidade" INTEGER DEFAULT '-1' NOT NULL,
        CONSTRAINT "active_calls_pkey" PRIMARY KEY("id")
) WITH OIDS;
```

- ✓ As informações registradas na tabela *active_calls* são utilizadas para calcular a utilização, o que é realizado por um gatilho (*trigger*) mostrado abaixo, e, além dele, os comandos

necessários para criar a tabela utilização, necessária para armazenar o resultado do cálculo da métrica de utilização, são mostrados.

```
CREATE OR REPLACE FUNCTION "public"."uso" () RETURNS trigger AS'
        BEGIN
                UPDATE utilizacao
                SET grau = (
                CASE
                 WHEN (SELECT count(*) from active_calls) = "0" then "0"
                 WHEN (SELECT count(*) from active_calls where
        tipo_origem = "0")>= (SELECT capacidade from active_calls limit 1) then "1"
        ELSE
            (SELECT  CAST((SELECT count(*) from active_calls where tipo_origem = "1") as float)/((SELECT capacidade from active_calls limit 1) - (SELECT count(*) from active_calls where tipo_origem = "0")))
        END)
              WHERE grau is not null;
         RETURN NEW;
        END;
        'LANGUAGE 'plpgsql' STABLE CALLED ON NULL INPUT SECURITY INVOKER;
        CREATE TRIGGER "calcuso" AFTER INSERT OR DELETE
        ON "public"."active_calls" FOR EACH STATEMENT
        EXECUTE PROCEDURE "public"."uso"();
```

Os comandos para criar a tabela “utilizacao” e a sua visão vêm em seguida. Um detalhe importante, é que o nome dessa visão (<nome da inst>) é o domínio da instituição, que foi declarado no parâmetro *alias* da configuração do OpenSER, com os pontos existentes no nome deste domínio substituídos por “_”.

```
CREATE TABLE utilizacao {
      grau double;
};
CREATE VIEW <nome da inst> AS SELECT * FROM utilizacao;
```