Balanceamento de Chamadas E.164 VoIP com Controles Centralizado e Distribuído

Anderson A. de Albuquerque, Paulo H. de Aguiar Rodrigues, Thiago M. Resende, Claudio M. de Farias

Núcleo de Computação Eletrônica - Departamento de Ciência da Computação/IM
Universidade Federal do Rio de Janeiro (NCE/UFRJ – DCC/IM/UFRJ) *
Caixa Postal 2324 – 20001-970 – Rio de Janeiro – RJ – Brasil

Laboratório de Voz Sobre IP – Núcleo de Computação Eletrônica.

andersonaa@posgrad.nce.ufrj.br, aguiar@nce.ufrj.br,
thiagoresende@nce.ufrj.br, claudiofarias@nce.ufrj.br

Abstract. *Centralized control and distributed algorithms for balancing E.164 VoIP calls destined to PSTN are discussed and compared. Distributed solutions, exploring fuzzy logic and analytic functions, adjust delays in generating call confirmation messages to reach load balancing. ENUM queries to private DNS are integrated with SIP signaling to achieve centralized control. The efficacy of these different algorithms is verified through simulation in various scenarios and implementations issues are discussed.*

Resumo. *Propostas de algoritmos distribuídos e de controle centralizado para balanceamento de chamadas E.164 VoIP destinadas à rede pública de telefonia são discutidas e comparadas. Para a solução distribuída, o controle do atraso do envio de confirmação é usado para obter o balanceamento, explorando soluções baseadas em lógica nebulosa e em funções analíticas. Para o controle centralizado, é explorada a integração com a sinalização SIP de consultas a DNS com ENUM. A eficácia dos diferentes algoritmos é verificada através de simulação de diversos cenários e questões relacionadas à implementação das soluções são discutidas.*

1. Introdução

Atualmente, o uso da tecnologia de transmissão de voz em redes IP (VoIP) está se tornando lugar comum, com várias iniciativas desta tecnologia surgindo nos meios acadêmicos e empresariais a cada dia. Todavia, a garantia de qualidade e desempenho da interoperação entre o ambiente VoIP e a rede de telefonia fixa comutada (RTFC)

* Suporte parcial do projeto VoIP4ALL da RNP. Paulo Rodrigues é analista do NCE e professor do DCC/UFRJ. Anderson Albuquerque é mestrando do Programa de Pós-Graduação em Informática do NCE/DCC/UFRJ. Thiago Resende e Claudio Farias são graduandos do DCC/UFRJ.

exige que o balanceamento de carga entre os diversos *gateways* que completam chamadas para um mesmo código de área seja equacionado.

Em VoIP são normalmente utilizados o H.323 [ITU-T H.323 2003] e o SIP (*Session Initiation Protocol*) [RFC 3261 2002]. Nesses ambientes, *gateways* de voz, conhecidos como *gateways* VoIP/PBX, são usados para interconectar as centrais telefônicas PBX (*Private Branch eXchange*) à rede de comunicação, através de um número limitado de canais de voz. Com isso, um *gateway* VoIP/PBX pode se tornar um gargalo em horários de pico de demanda.

Dentro do padrão H.323 existe o servidor *gatekeeper* (GK) que recebe chamadas de terminais H.323 e as encaminham para um destino; no ambiente SIP existe o servidor *Proxy SIP* com funções semelhantes. Tanto os clientes H.323/SIP como os *gateways* de voz precisam estar registrados em um servidor VoIP, para que o encaminhamento de chamada possa acontecer. É pressuposto que os servidores atuem como *proxies* de mídia e sinalização, forçando a mídia e sinalização passar por eles. Dessa forma, uma chamada VoIP entre duas instituições é encaminhada primeiro pelos servidores VoIP antes de seguir para o *gateway* de voz ou terminais/clientes. Esta é uma configuração típica que facilita o controle, e a implantação de mecanismos de segurança e QoS na rede. Quando for indiferente detalhar um servidor VoIP, será utilizada a sigla SV (Servidor VoIP).

Quando vários SVs receberem uma mesma solicitação de chamada para a RTFC, o SV que enviar a confirmação primeiro ao servidor que originou a chamada (*chamador*) será o escolhido para o completamento. Esse procedimento simples da sinalização VoIP pode levar a um uso ineficiente dos *gateways*, repassando as chamadas sempre para a instituição que responder primeiro, provocando a saturação de seu *gateway,* e, como conseqüência, o não completamento de chamadas, ainda que existam canais livres para comunicação em outras instituições da mesma cidade. Além disso, a instituição que receber mais chamadas terá um gasto financeiro maior. A saturação do *gateway* pode ser contornada com o aumento do número de canais de voz, mas o compartilhamento dos *gateways* de uma mesma cidade é uma solução mais adequada. O balanceamento de chamadas tem por objetivo atingir o compartilhamento de recursos, seguindo alguma métrica de otimização pré-definida. Com isso, uma diminuição da ocorrência de chamadas não completadas pode ser alcançada.

O balanceamento, tanto em H.323 como em SIP, pode ser implementado com uso de controle centralizado ou algoritmo distribuído. No controle centralizado [Ghanem 2004], existe um ponto central de decisão que recebe as informações sobre as métricas dos *gateways* de voz das instituições envolvidas. Este ponto é sempre consultado antes do encaminhamento de uma chamada, devendo ser tomadas medidas de redundância para aumento da confiabilidade e da disponibilidade [Ghanem 2004]. Espera-se que o atraso na tomada de decisão possa ser feito em concordância com o ambiente de sinalização implantado, evitando retardos adicionais indesejáveis [Rhee 2004]. Para coerência em decisões centralizadas, é preciso garantir que as métricas no ponto central estejam sempre atualizadas. Caso a taxa de estabelecimento de chamadas seja alta, a freqüência de alteração das métricas também será elevada, provocando um aumento no tráfego de controle enviado ao ponto central [Rhee 2004] para atualização das informações. Todavia, com um *backbone* de alta performance e a localização adequada do ponto central de decisão, o impacto do tráfego de controle pode ser

marginal e sem conseqüências práticas. No controle centralizado, a eficiência e a escalabilidade do processo de decisão pode ser o ponto mais importante, principalmente em serviços com utilização crescente [Ghanem 2004 e Floyd 1993], como acontece no VoIP. [Chang 2004] apresenta soluções de balanceamento centralizadas específicas para H.323, com implementação de GK própria e é uma das poucas referências nessa área de nosso conhecimento.

Em artigo anterior [Albuquerque 2006], um algoritmo distribuído para balanceamento local dinâmico foi apresentado. Uma nova proposta mais otimizada e com menor impacto na demora na admissão de chamadas é apresentada e comparada com a proposta anterior. Uma alternativa de implementação de balanceamento distribuído baseada em lógica *fuzzy* é também desenvolvida. Finalmente, uma implementação de controle centralizado explorando a integração da sinalização SIP padrão com consultas ENUM a DNS privado é proposta e sua implementação em produção analisada.

O artigo está divido em 5 seções. Na seção 2, as alternativas de arquiteturas para encaminhamento E.164 e viabilidade de implementação de balanceamento tanto em H.323 como em SIP são discutidas. Na seção 3, os algoritmos distribuídos propostos neste artigo são explicados e o resultado de simulações é mostrado. Finalmente, a implementação com controle centralizado é mostrada na seção 4 e são apresentadas as conclusões e trabalhos futuros na seção 5.

2. Balanceamento em arquiteturas para encaminhamento E.164

O balanceamento de chamadas VoIP pode ser alcançado de várias formas, mas independentemente de como ocorra será utilizada uma métrica que reflita a ocupação dos canais de voz dos *gateways*. As formas de alcançar o balanceamento podem ser: através de atrasos para geração da confirmação de chamada ou enviando confirmação de chamadas segundo uma ordenação pré-definida.

Chamadas que se completam por um *gateway* VoIP/PBX podem ser classificadas como internas ou externas, dependendo de sua origem ser na própria instituição (SV ou PBX) ou em outra instituição, respectivamente. Chamadas estabelecidas sem uso do *gateway*, independentemente de ter uma origem interna ou externa, não afetarão o balanceamento, que somente leva em consideração as chamadas 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 cidade não poderão ser completadas e serão descartadas. Além disso, quando uma instituição opera com vários *gateways* para desempenho e redundância, apenas o total dos canais de voz importa para o balanceamento.

A métrica escolhida é dada por $U_i = E_i / (C_i - I_i)$ (1) e representa a razão de chamadas externas em andamento sobre o número de canais de voz não ocupados com chamadas internas. O índice *i* representa uma instituição; C_i é a capacidade do *gateway* da instituição; E_i é o número de chamadas externas em andamento; I_i é o número de chamadas internas em andamento. Razões para a escolha dessa fórmula podem ser obtidas em [Albuquerque 2006]. Na referência a elemento de uma figura, será usado o modelo (F-N), onde F é o número da figura e N o número da sinalização/elemento.

A Figura 1a mostra como o atraso da primitiva de confirmação pode influenciar no balanceamento de chamadas. Nela, o servidor VoIP chamador (SV_0) inicia uma chamada cujo prefixo destino permite que 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 (1a-1 e 1a-3) para cada um destes SVs.

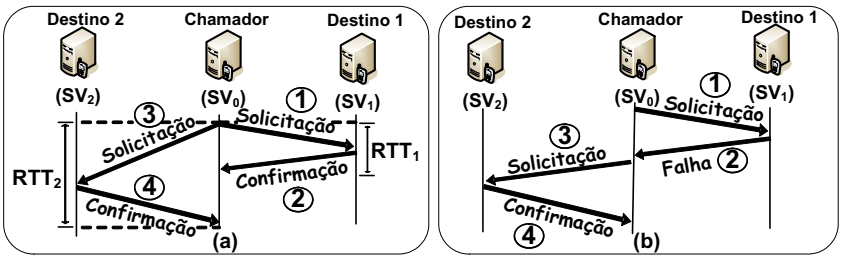

Figura 1. Mecanismos de encaminhamento de chamadas VoIP

Assim que a solicitação é recebida pelos SVs de destino, eles enviam as respostas de confirmação (1a-2 e 1a-4). Porém, os retardos das sinalizações contribuem para que (1a-2) chegue primeiro a SV_0, indicando que SV1 receberá a chamada. Essa seqüência de eventos é típica tanto de H.323 como de 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. 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 extra no encaminhamento da chamada ou na geração da confirmação pode ser utilizado para alterar a seleção do SV de destino.

A Figura 1b mostra a situação em que o SV_0, com base na consulta das métricas em modo centralizado, já conhece a ordem preferencial que os SVs de destino devem ser contatados. Nesse exemplo, o primeiro a ser contatado é o SV_1 (1b-1), e por algum motivo é retornada uma indicação de falha (1b-2). Assim, que a indicação de falha é recebida, o chamador contata o segundo SV da lista (1b-3), nesse exemplo SV_2, e envia a confirmação (1b-4), fazendo com que a chamada seja completada por ele.

2.1. Encaminhamento E.164 no H.323

Quando a sinalização H.323 é utilizada para VoIP, pode-se ter uma arquitetura completamente entrelaçada, onde todos os GKs podem realizar chamadas entre si (são considerados vizinhos) e é possível associar prefixos E.164 a cada GK. Para escalabilidade, o mais comum é utilizar uma arquitetura hierárquica, onde existe um *Directory Gatekeeper* (DGK) que armazena os prefixos E.164 dos GKs e apenas o DGK é vizinho de cada GK. No caso do DGK, este pode ser usado como *proxy* de sinalização ou não, resultando em três arquiteturas possíveis. Estas arquiteturas são mostradas na Figura 2, onde se detalha as sinalizações envolvidas em uma chamada. Na figura, o GK_0 é quem inicia a chamada e os GKs 1 e 2 são aqueles que podem encaminhá-la. As numerações nas primitivas representam os instantes em que elas foram geradas.

Na Figura 2a existe o servidor de localização (DGK) e sua base de dados com os prefixos dos vizinhos. O DGK apenas repassa a sinalização (2a-1) para GK_1 e GK_2, sem agir como *proxy*. Nesse cenário, caso fosse utilizado um algoritmo distribuído para balanceamento, às confirmações (2a-3 e 2a-4) seriam atrasadas em cada GK. Haveria a necessidade de alteração do código fonte nos GKs para gerar o atraso extra. Nesse cenário, caso as métricas das instituições fossem mantidas no DGK, um controle

centralizado poderia ser implementado, atrasando o envio do LRQ (2a-2) ao GK com menor prioridade, dando chance ao GK mais prioritário de confirmar a chamada. Observe que o DGK não participa da sinalização (só encaminha) e não recebe confirmação de falha, caso algum GK não aceite a chamada. Por isso, nesse cenário a alternativa é enviar a todos os GKs, com defasagem de tempo. Em termos de implementação, a solução centralizada com modificação apenas no DGK seria mais desejável do que alterar os códigos fonte de todos os GKs, principalmente porque nem sempre se tem acesso ao fonte, haja vista que eles podem ser proprietários ou podem ser *softwares* heterogêneos, que dificultaria a manutenção e atualização.

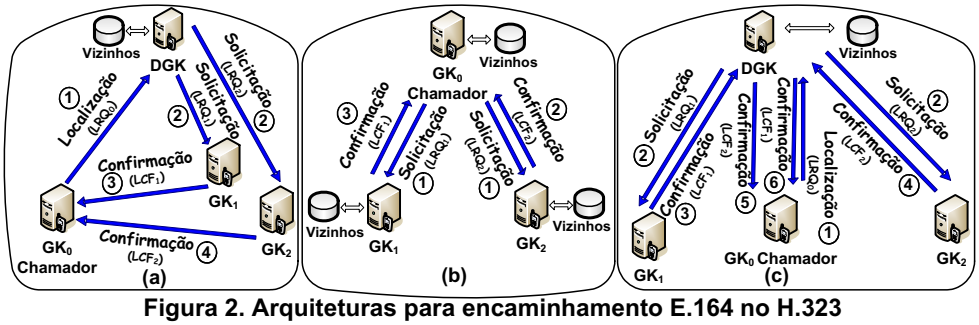

Figura 2. Arquiteturas para encaminhamento E.164 no H.323

Na Figura 2b não é utilizada uma entidade central, e todos os GKs são vizinhos entre si. Para balanceamento, a métrica atualizada de cada GK precisa estar disponível a todos os outros, bem como devem ser conhecidos os prefixos E.164 e endereços IP associados a cada GK. Nessa arquitetura, para uma solução centralizada ou distribuída, seria necessário que todos os GKs sofressem alteração de código, o que impactaria a escalabilidade.

A arquitetura da Figura 2c é semelhante à da Figura 3a, porém o DGK atua em modo *proxy* de sinalização. Balanceamentos centralizado e distribuído seriam ambos viáveis. No controle centralizado haveria a possibilidade de encaminhar as solicitações sequencialmente a cada GK destino em ordem de prioridade, pois o DGK participa da sinalização e recebe as confirmações. Opcionalmente, poderia enviar as solicitações em paralelo com atraso, como descrito no cenário 2a. A solução centralizada tem a vantagem de centralizar o desenvolvimento e alterações de *software* em um único ponto.

2.2. Encaminhamento E.164 no SIP

SIP é um protocolo VoIP que normalmente usa o DNS para resolver o destino de uma chamada especificado como um domínio (p.ex., fulano@voip.ufrj.br). Quando o destino é apenas um número E.164, é preciso montar uma arquitetura para esta finalidade. Na Figura 3 são apresentadas duas possibilidades (3a e 3b).

Na arquitetura da Figura 3a, o *proxy* SIP chamador (SV_0) precisa saber quem são os destinos de uma chamada consultando um *proxy* SIP, denominado DSER (servidor de localização) (3a-1). Para armazenamento dos prefixos E.164 é utilizado um DNS privado com registros NAPTR, assim o DSER envia uma requisição ENUM (*Eletronic NUmber Mapping*) para esse DNS (3a-2) e, com base no prefixo E.164 destino da chamada, ele informa ao DSER quem são os SVs que podem encaminhá-las. Após (3a-2), a listagem com os possíveis destinos é retornada para o chamador (3a-3) através da primitiva padrão *REDIRECT* do SIP, possivelmente com prioridades

dependentes de métricas de balanceamento. 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*), e o protocolo OSP (*Open Settlement Protocol*), mas o DNS com ENUM mostrou-se mais escalável, robusto, simples e de fácil integração [Farias 2006]. Caso seja utilizado um algoritmo centralizado, apenas um módulo seria implementado no DSER. Este módulo iria atribuir pesos para cada SV destino em função das métricas disponíveis, definindo a ordem de prioridade de encaminhamento. O DSER precisaria receber as métricas de todos os servidores, mas somente o código do DSER necessitaria de modificação, facilitando a escalabilidade. Com algoritmo distribuído, cada SV de destino precisaria de modificações de código para atrasar as primitivas de solicitação (3a-4).

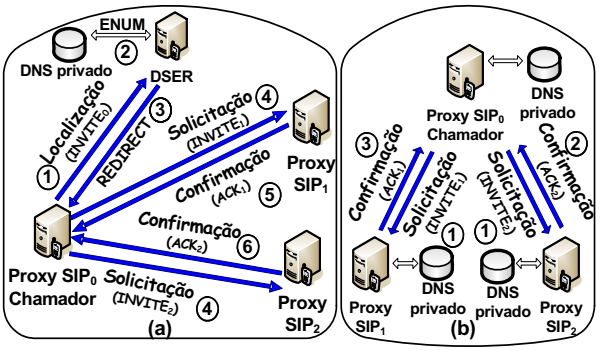

Figura 3. Encaminhamento E.164 no SIP

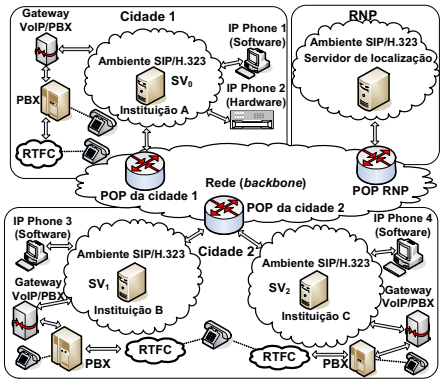

Figura 4. Cenário de implementação

Na Figura 3b as informações contidas no DNS privado são compartilhadas com todos SVs, e não existe um DSER. Além disso, para um balanceamento centralizado ou distribuído, todos os SVs precisariam ser contemplados com um módulo extra, o que pode ser de execução inviável em códigos proprietários.

3. Algoritmos distribuídos para balanceamento

Nesse item são apresentadas duas propostas de algoritmos distribuídos, sendo que na primeira é utilizado um servidor de referência com uma função de atraso e na segunda utiliza-se lógica nebulosa. A idéia geral desses algoritmos é gerar atrasos no envio das comfirmações. Esses algoritmos utilizam os retardos de rede entre os SVs e os POPs, e as formas de medi-los encontram-se em [Albuquerque 2006]. É utilizado o fone@RNP (veja a Figura 4) como cenário de implementação. Nesse serviço, o POP é um ponto comum (*convergência*) de passagem das primitivas de sinalização para os SVs de destino.

3.1. Algoritmo de balanceamento utilizando ponto de referência

Um algoritmo de balanceamento distribuído foi apresentado no SBRC2006 [Albuquerque 2006] e sua operação é resumida abaixo, já que um algoritmo aprimorado e sua análise comparativa com a versão anterior serão apresentados.

Considere um cenário como na Figura 4 em que uma cidade hospeda um POP com n servidores VoIP conectados, onde, em termos de retardo, SV_1 é o mais próximo e SV_n o mais distante. O POP é o ponto comum (*convergência*) de passagem das primitivas de sinalização para a cidade, sejam elas provenientes de um servidor local ou

de um SV chamador (SV_0) conectado a outro POP. Por simplicidade, será assumido que os *gateways* possuem um mesmo número C de canais de voz com o PBX.

Define-se: RTT_i, o retardo de rede ida e volta entre POP e o SV_i; $RTT(j,i) = |RTT_j - RTT_i|$, o retardo entre SV_j e SV_i; $ConfSol_i$, a confirmação de solicitação emitida por SV_i; U_i a utilização do *gateway* de SV_i dada por (1); e $D_i(U_i)$, o atraso para envio da $ConfSol_i$ em função de U_i. Sem perda de generalidade e para ter um mesmo cenário de [Albuquerque 2006], será assumido $U_i = E_i / C$, onde E_i é o número de chamadas externas em andamento e o número de chamadas internas foi considerado zero. 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 será definida em função da máxima variação entre pares de servidores e dada por

$$\Delta_{rede} = máximo\ \{RTT(j,i)_{\max} - RTT(j,i)_{\min}\}, \forall i, j \in \{1, \cdots, n\}\} \quad (2).$$

Quando o servidor SV_i alcança uma utilização $U_i = A < 1$, neste ponto $ConfSol_i$ é atrasada de um valor $D_i(A) = RTT(i,n)$, forçando o alinhamento entre SV_i e SV_n, isto é fazendo com que as confirmações desses dois servidores cheguem ao POP no mesmo instante. Em [Albuquerque 2006], a intenção inicial de usar a fórmula $D_i(U_i) = \left(RTT(i,n).U_i\right)/A$ para o atraso esbarrava no fato que os atrasos gerados por SV_n seriam sempre zero e este servidor sempre receberia todas as chamadas a partir do alinhamento, até a saturação de seu *gateway*. A solução encontrada foi utilizar um servidor VoIP virtual (SV_v), distante do POP de RTT_v, $RTT_v > RTT_n$, permitindo que o cálculo do atraso usasse SV_v como referência mais distante. Dessa forma, o retardo adotado usa a fórmula $D_i(U_i) = \left(RTT(i,v).U_i\right)/A$ (3), fazendo o atraso variar linearmente com a utilização, como exemplificado no gráfico da Figura 5a. Enquanto $U_i < A$, $\forall i \neq n$, SV_n não recebe chamadas, porque elas são enviadas a outro SV com menor retardo ao POP. Quando as utilizações dos SVs alcançam o valor A, eles se alinham ao SV_n. A partir daí, a geração de atraso é que comandará para quem a próxima chamada será entregue.

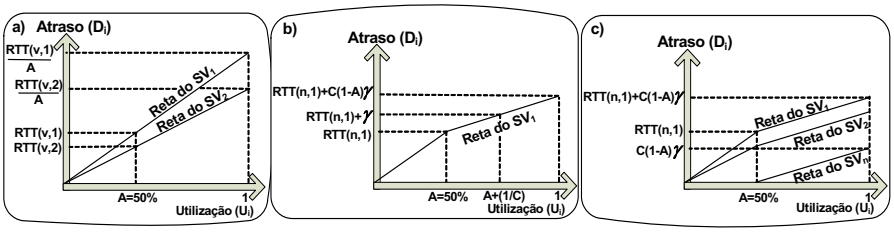

Figura 5. Gráficos do atraso D_i versus utilização U_i

O posicionamento do servidor virtual não pode ser arbitrário e, para garantir o balanceamento, é preciso que $RTT(n,v)_{\min} > C(1-A)RTT(1,n)_{\max}$ (4). Esta fórmula é mais precisa que a fórmula (5) apresentada em [Albuquerque 2006]. Quando um servidor alcança $U = 1$, todos os seus canais de voz estão ocupados e ele não mais envia confirmações. Quando $U_i = 1 - 1/C$, indicando que resta apenas um canal livre, o

maior atraso no envio de ConfSol$_i$ será $D_{i,\max} = \big(RTT(i,v).(C-1)\big)/(CA)$. O maior valor será $D_{1,\max}$. Assumindo para $RTT(n,v)$ o limite inferior dado por (4), pode ser deduzido o atraso máximo $D_{\max}$ do algoritmo que será dado por $D_{\max} = (C-1)(1-A)RTT(1,n)_{\max} / A$. Como $RTT(1,n)_{\max} < RTT(1,n) + \Delta_{rede}$, e aproximando o atraso pelo seu limite superior, chega-se a

$$D_{\max} \approx \frac{(C-1)(1-A)(RTT(1,n) + \Delta_{rede})}{A} \quad (5).$$

Esta fórmula corrige a fórmula (6) de [Albuquerque 2006], que não considera corretamente o impacto de Δ_{rede}. O pior valor para este atraso é obtido para $A = 1/C$, que é o menor valor possível de A. Substituindo em (5), obtém-se $D_{\max} = (C-1)^2 (RTT(1,n) + \Delta_{rede})$ e, como em geral $C >> 1$, chega-se a $D_{\max} \approx C^2 RTT(1,n) + C^2 \Delta_{rede}$ (6), no pior cenário.

Embora o cenário para gerar este atraso máximo requeira a saturação de todos os outros servidores o que deverá ser uma situação pouco freqüente num serviço bem administrado, este valor é uma das parcelas do tempo para a admissão de uma chamada. As outras parcelas dependem da sinalização, da arquitetura implantada, da rede, e do tempo de resposta dos elementos envolvidos no estabelecimento da chamada como *proxies* e *gateways*. No caso de SIP, estes tempos extras são da ordem do tempo de propagação ida e volta entre origem e destino.

NOVO ALGORITMO

Para resolver o problema do alinhamento do SV$_n$, é proposta uma alteração na geração do atraso da confirmação para permitir atuar *sem SV$_v$*. Para isso, o SV$_n$, servidor mais distante do POP, é usado como referência no lugar de SV$_v$. A Figura 5c reproduz a nova dependência do atraso com a utilização para os servidores SV$_1$, SV$_2$ e SV$_n$. Como no algoritmo original, usa-se $U_i = A < 1$ para definir o ponto de alinhamento. Assim, $D_i(A) = RTT(i,n)$, para $1 \le i \le n$. Deve ser observado que, enquanto as outras instituições vão recebendo chamadas e incrementando sua utilização em direção a A, o servidor mais distante SV$_n$ estará com utilização zero, sem receber nenhuma chamada externa, pois o atraso extra dos outros servidores não compensa o maior retardo de SV$_n$ ao POP. Assim como no algoritmo anterior, somente após as utilizações dos outros servidores alcançarem A, SV$_n$ poderá começar a receber chamadas externas. Trata-se de uma assimetria, mas o ponto positivo é que as chamadas externas durante este tempo são admitidas com o menor atraso possível.

Quando uma instituição já alinhada ($U_i = E_i/(C_i - I_i) \ge A$) recebe mais uma chamada externa, a utilização vai para $E_i/(C_i - I_i) + 1/(C_i - I_i)$, com um incremento de $1/(C_i - I_i)$. A este incremento corresponderá um atraso na geração da confirmação, seguindo a reta do atraso versus utilização, que cresce linearmente com a mesma inclinação para todos os servidores. Para a análise de sensibilidade à variação de retardo, é crítico o número de canais disponível nos *gateways*. O menor incremento na utilização acontecerá para o servidor com o maior número de canais de voz. Seguindo o

que foi feito para a análise do algoritmo anterior, será assumido $U_i = E_i / C$, onde E_i é o número de chamadas externas em andamento e C representa o maior número de canais disponibilizados em um *gateway* de voz. O número de chamadas internas foi considerado zero e não interfere na utilização. Para um incremento da utilização de $1/C$, o atraso mínimo será definido como γ. Este atraso γ deve garantir que, quando todas as estações estiverem alinhadas com utilização $U = E/C > A$ e somente uma estação estiver com utilização $U = (E-1)/C$, a próxima chamada será garantidamente encaminhada para esta estação com menor utilização. Para que o balanceamento seja robusto às flutuações de retardo, é necessário que $\gamma > \Delta_{rede}$, com Δ_{rede} calculado em (2).

O gráfico da Figura 5b mostra que, quando $U_i = 1 - 1/C$, indicando que resta apenas um canal livre, o maior atraso no envio de ConfSol$_i$ será $D_{i,\max} = RTT(i,N)_{\max} + (C(1-A)-1)\gamma$. O maior valor será $D_{1,\max}$. Assumindo $\gamma = \Delta_{rede}$ e observando que $RTT(1,N)_{\max} < RTT(1,N) + \Delta_{rede}$, pode-se aproximar o atraso pelo seu limite superior e obtém-se o atraso máximo para a nova proposta

$$D^*_{\max} \approx RTT(1,N) + C(1-A)\Delta_{rede} \quad (7).$$

O pior caso para este atraso ocorre para $A = 1/C$, o menor valor possível de A. Substituindo em (7), obtém-se $D^*_{\max} < RTT(1,N) + (C-1)\Delta_{rede}$ e, como em geral $C >> 1$, chega-se a $D^*_{\max} \approx RTT(1,N) + C\Delta_{rede}$ (8), em pior cenário.

Para termos uma idéia da otimização obtida com este novo método, usando $C = 30$, $A = 0{,}40$, $RTT(1,N) = 10\ ms$ e $\Delta_{rede} = 10\ ms$, obtém-se para o algoritmo descrito em [Albuquerque 2006] $D_{\max} = 870\ ms$ e para a nova proposta $D^*_{\max} = 190\ ms$. Ainda que se usem valores mais conservadores para γ para garantir robustez, como $\gamma = 2\Delta_{rede}$, ainda assim obteríamos um atraso máximo de $370\ ms$, bastante melhor que o obtido em [Albuquerque 2006]. Manuseando as expressões (7) e (5), pode-se mostrar que

$$D_{\max} = \frac{[(C(1-A)-1)RTT(1,n) + D^*_{\max} - (1-A)\Delta_{rede}]}{A} \quad (9).$$

Observa-se que $D_{\max} > D^*_{\max}$ para qualquer escolha de parâmetros, ou seja, a nova proposta é sempre melhor que a anterior [Albuquerque 2006].

Como exemplo de uma seqüência de eventos seguindo o esquema da Figura 5c, para esta nova proposta, imagine um cenário hipotético com dois SVs, com C=4, A=50%, $RTT_1 = 100\ ms$ e $RTT_2 = 200\ ms$. Como SV$_1$ está mais próximo ao POP, ele recebe as duas primeiras chamadas, fazendo sua utilização atingir 50% e ele se alinha com SV$_2$. A terceira chamada poderia ser encaminhada para qualquer um deles, mas assume-se que SV$_1$ a receba, passando a ter U$_1$=75% e ainda U$_2$=0%. Em seguida, a quarta chamada é obrigatoriamente encaminhada para o SV$_2$, e passamos a ter U$_1$=75% e U$_2$=25%, sendo que o atraso para geração da ConfSol$_2$ ainda é zero, pois usa utilização está abaixo de 50%. A quinta chamada é encaminhada para SV$_2$, e passamos a ter U$_1$=75% e U$_2$=50%. Nesse instante, como o atraso gerado por SV$_2$ ainda é zero e

SV_1 gera um atraso adicional além de $RTT(1,2)$, a sexta chamada será obrigatoriamente entregue a SV_2, então U_1=75% e U_2=75%. A sétima chama-da é entregue a um deles, e a última é enviada para o SV que ainda estiver com um canal livre.

3.2. Algoritmo de balanceamento utilizando Lógica *Fuzzy* (nebulosa)

A lógica *fuzzy* é utilizada no balanceamento de carga em diversas áreas [Borzemski 2003 e Zahirazami 2003], e na literatura encontram-se aplicações para CAC (Controle de admissão) e balanceamento de carga de processadores. Ela será utilizada em nosso contexto para gerar os atrasos no envio das confirmações de solicitação de chamadas, utilizando a métrica em (1), como feito pelo algoritmo da seção 3.1. A lógica nebulosa tentará fazer o alinhamento natural dos servidores, sem que haja nenhuma referência a uma utilização a partir da qual se faz o alinhamento entre os servidores e inicia-se o balanceamento, como no caso do algoritmo anterior. Optou-se por este comportamento, embora pudesse ser possível somente forçar o início do balanceamento a partir de uma determinada utilização.

Na simulação no MatLab duas variáveis de entrada e uma de saída foram utilizadas, com universo de discurso [0, 1]. Uma das variáveis de entrada é a **utilização**, vide Figura 6, calculada pela métrica (1). A segunda variável de entrada é a **razão** $\eta = RTT_i / RTT_n$ (10), vide Figura 7. A variável utilização tem os seus valores mapeados para a variável saída, de forma que um determinado intervalo de utilização corresponda a um intervalo de valor de atraso. Em topologias com $RTT_n >> RTT_i$ o tempo para o balanceamento será muito elevado e somente ocorrerá para valores altos de utilização, e 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.

Observe nas Figuras 6 e 8 que os conjuntos das variáveis utilização e saída possuem formato triangular. 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. Caso os conjuntos adjacentes estivessem bem afastados uns dos outros isso não ocorreria, pois a variação do atraso seria pequena para alguns intervalos de valor na entrada e o balanceamento estaria prejudicado para alguns intervalos de valores de entrada.

Através das regras de inferência, quanto menor η maior o atraso para geração de uma confirmação, e vice-versa. Sem esta variável, 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%. O atraso para geração da confirmação de solicitação será dado por $D_i(U_i, \eta)=\Theta.T$ (11), onde: Θ é a variável de saída do processo de "desfuzificação"; T representa um tempo utilizado para estimar o atraso da geração da confirmação de solicitação (D_i). Pelos valores que Θ pode assumir, o valor de D_i varia entre zero e T. Nas simulações foi usado $T > RTT(n,i)$ (12), que é suficiente para um SV_i se alinhar com o SV_n.

Quanto maior o valor de T menos suscetível o algoritmo será à variação de retardo, mas ele não pode ser da ordem de grandeza de minutos, pois isto causaria um atraso excessivo na admissão da chamada ou o chamador pode até desistir da chamada.

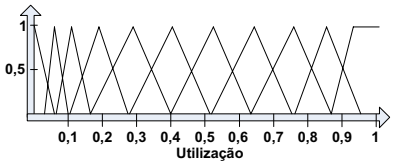

Figura 6. Entrada Utilização

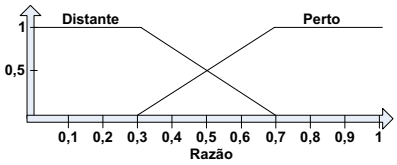

Figura 7. Entrada Razão

Figura 8. Saída (Θ)

3.3. Resultados das simulações

Foram realizadas simulações utilizando o *MatLab*, sendo necessário apenas a sua linguagem de *script* e o *Fuzzy Inference System Editor* (*FIS-Editor*) para o algoritmo com lógica *fuzzy*. Vale ressaltar que as simulações foram realizadas para os algoritmos descritos nessas seções 3.1 e 3.2. Nessas simulações, utilizando retardos determinísticos, o funcionamento dos algoritmos foi observado.

Nas simulações foram disparadas 38 chamadas com duração infinita e o cenário, com três SVs, possui parâmetros $RTT_{1=}20ms$, $RTT_{2=}30ms$, $RTT_{3=}100ms$, $\Delta_{rede=}20ms$, $A=25\%$ e $C=12$. Nas Figuras 9, 10 e 11 são mostradas as distribuições das chamadas do algoritmo utilizando SV_3 como ponto de referência. Pode-se observar que o balanceamento se inicia com $A=25\%$ e todos os servidores alcançam a utilização máxima com o disparo de 36 chamadas. Nenhuma chamada deixou de ser completada.

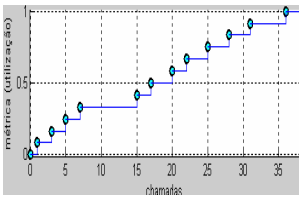

Figura 9. SV_1

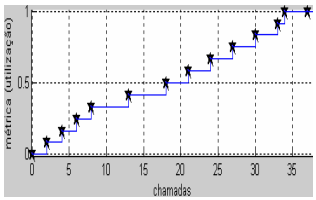

Figura 10. SV_2

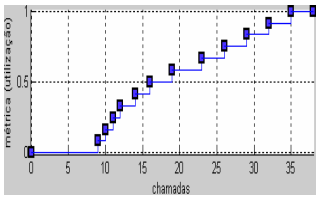

Figura 11. SV_3

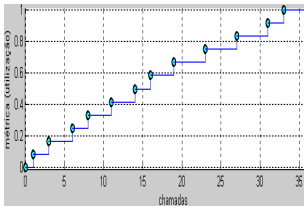

Figura 12. SV_1

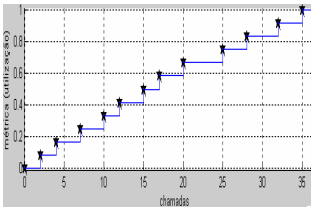

Figura 13. SV_2

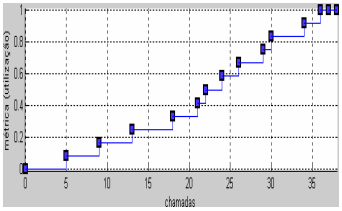

Figura 14. SV_3

As Figuras 12, 13 e 14 mostram as distribuições das chamadas para o algoritmo com lógica fuzzy, onde T= $RTT(3,1)+2\,RTT_3$, e pode-se observar que o balanceamento

ocorre de maneira similar ao do algoritmo com ponto de referência, mas o alinhamento dos SV_is com o SV_n ocorre quando eles estão com Ui≈20%. Os atrasos máximos obtidos situaram em torno de $0{,}25s$, coincidindo com o valor limite de $0{,}26s$ obtido de (7) para o algoritmo com ponto de referência. Em cenários onde o algoritmo com lógica fuzzy apresentou atrasos comparativamente maiores, sempre foi possível ajustar T para o balanceamento ocorrer com atrasos da mesma ordem de grandeza do algoritmo com ponto de referência. Nas Figuras de 9 a 14, as chamadas não são completadas apenas quando os gateways já estão saturados.

4. Mecanismo centralizado para o balanceamento de chamadas VoIP

No mecanismo centralizado, a instituição com menor utilização (1) do seu *gateway* é priorizada para o encaminhamento de chamadas. Esse mecanismo centralizado foi concebido para atender à arquitetura da Figura 3a. Nessa arquitetura, para iniciar uma chamada, um pedido de localização (3a-1) é enviado pelo chamador ao DSER. É retornada uma primitiva de redirecionamento (3a-3) que possui uma listagem com os SVs de destino que podem encaminhar a chamada. Nesta listagem existe um campo definindo a ordem com que cada um destes SVs devem ser contatado, priorizando o SV com menor utilização do seu *gateway* VoIP/PBX. Como a ordenação dos SVs de destino é feita em função da utilização, não são utilizados atrasos para envio de qualquer primitiva, o que acarreta em uma seleção ótima. O SV menos prioritário somente é contatado quando a tentativa de encaminhamento de chamada para o mais prioritário indicar uma falha. Além disso, caso existam SVs de destino igualmente prioritário, serão enviadas solicitações de chamadas simultâneas para eles (*fork* paralelo).

Para o balanceamento centralizado foi analisada a viabilidade de utilizar o *gateway* de voz *Asterisk* [Digium 2006]. Ele possui um recurso chamado AGI (*Asterisk Gateway Interface*), que possibilita executar programas externos, agregando novas funcionalidades. Quando uma chamada é iniciada ou terminada, o *Asterisk* das instituições interpreta as linhas de configuração dos seus planos de numeração, e elas ativam o AGI que executa um programa (15-1). Ele irá classificar a chamada em externa ou interna, e inserir um registro referente a ela em uma tabela (tabela chamadas em andamento) (15-1); ao seu término este registro é removido da tabela.

A cada operação na tabela chamadas em andamento é ativado um *trigger* (gatilho) (15-2), que faz o cálculo da métrica (1) consultando os registros na tabela de chamadas em andamento (15-3). O resultado é inserido na tabela utilização (15-4). Por fim, cada instituição transmite as informações para um ponto central através de um mecanismo de replicação de tabelas (15-5) [Borzemski 2003]. Assim, quando o DSER for definir em qual ordem cada SV deve ser contatado, basta consultar este ponto central.

Agora, imagine uma chamada sendo iniciada; para isso um pedido de localização é enviado para o DSER (3-1) pelo chamador. Os possíveis destinos são consultados no DNS através de uma solicitação NAPTR (ENUM) (3-2), o que é feito pela função *enum_query()* do *proxy SIP OpenSER*. Ela foi modificada para poder consultar a tabela utilização (15-6), e definir na primitiva *REDIRECT* (3-2) a ordem com que cada SV de destino deve ser contatado, adicionando pesos a estes SVs. Assim, ao receber esta primitiva com os pesos o chamador sabe que um determinado SV_i é mais prioritário em relação aos outros. O servidor mais prioritário é contatado primeiro pelo chamador (3-4) e, apenas se ele não puder encaminhar a chamada, outro com prioridade

inferior é contatado. Isso é chamado de *fork* seqüencial. Com base na veracidade dos dados da tabela utilização, garante-se que o servidor com menor utilização seja contatado primeiro. Mas a acuidade das informações depende da freqüência das suas atualizações pelo mecanismo de replicação. Mas como as atualizações estão sendo executadas a cada início e fim de chamada, a utilização instantânea está sendo mantida corrente. Um diferencial relevante deste mecanismo de balanceamento de chamadas é que o tempo para a admissão da chamada passa a ser o menor possível.

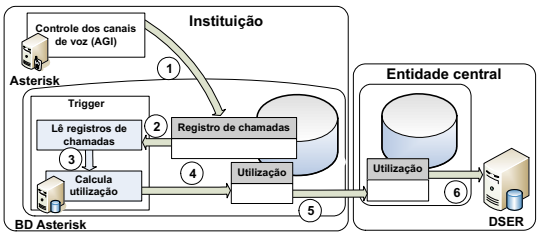

Figura 15. Balanceamento centralizado

Apesar de ter sido feito o estudo da viabilidade desse mecanismo para o SIP, ele poderia ser utilizado pelo H.323. Para isso, considerando que o DGK está configurado em modo *proxy* de sinalização (Figura 2c), ele precisaria sofrer uma modificação de código para poder consultar a tabela de utilização e enviar LRQ primeiro para o GKs mais prioritário. Outro menos prioritário seria consultado quando não houvesse contato do GK de destino ou se este GK informasse a impossibilidade da chamada ser encaminhada através dele.Utilizando esse mecanismo centralizado o balanceamento de chamadas é obtido, e nele o encaminhamento sempre contempla o SV com menor utilização dos canais de voz do *gateway* VoIP/PBX.

5. Conclusões e trabalhos futuros

A média dos atrasos e os atrasos máximos para admissão de chamada foram observados, assim concluiu-se que a nova proposta é bem melhor que [Albuquerque 2006]. O algoritmo com lógica *fuzzy* pode ter um atraso máximo próximo à solução com ponto de referência, se o parâmetro T for ajustado, para evitar atrasos excessivos. Além disso, vale ressaltar que o mecanismo centralizado não possui atrasos, e sua performance é ótima, mas depende da rapidez e acuidade com que as informações de utilização são atualizadas na entidade central, que constitui um ponto crítico de falha.

Um ponto positivo do mecanismo centralizado é que a modificação de código no *OpenSER* (proxy SIP de código aberto usado no serviço *fone@RNP*) é mínima, e o mecanismo centralizado pode ser implementado sem problemas. Além disso, para viabilizar o seu funcionamento, exceto a modificação de código citada, são utilizadas apenas soluções padrão, como por exemplo: DNS com ENUM, recursos do banco de dados e replicação de tabelas de banco de dados. Em breve, a implementação será testada em ambiente de produção.

O algoritmo com ponto de referência é sensível à variação de retardo, que deve ser estimado com freqüência ou usado um valor bem conservador. A subestimação da variação de retardo pode causar erros no balanceamento, porém, quando o percurso entre servidores VoIP compreende apenas *backbones* de alta performance, que é uma realidade atualmente, o efeito da variação de retardo é bastante minimizado.

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 pequena 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. Como trabalho futuro, no algoritmo com lógica *fuzzy* o valor de T em (11) poderia ser alterado dinamicamente e realimentar no algoritmo, ou seja, um aprendizado sobre a topologia para ajustar T. Em trabalhos futuros, simulações estocásticas podem ser realizadas.

6. Bibliografia

Albuquerque, A. A.; Rodrigues, P.H.A, " Balanceamento de Chamadas VoIP H.323 para a Rede de Telefonia Pública", 24° SBRC, Curitiba, PR – Brasil, Junho, 2006.

Borzemski, Leszek; Zatwarnicki, Krzysztof, "A Fuzzy Adaptive Request Distribution Al-gorithm for Cluster-based Web Systems", IEEE, 2003.

Chang, Cheng-yue; CHEN, Ming-syan; HUANG, Pai-han, "An H.323 Gatekeeper Prototype: Design, Implementation, and Performance Analysis", IEEE Transactions On Multimídia, p. 936-946. Dezembro, 2004.

Digium. Disponível em: <http://www.asterisk.org>. Último acesso em maio 2006.

Farias, Claudio Miceli. Encaminhamento de Chamadas VoIP SIP com Uso de Endereçamento Telefônico E.164. Rio de Janeiro: IM/CCMN/UFRJ, 2006. Disponível em: <http://www.im.ufrj.br/jicccmn/jicim/sessoes.html>. Último acesso em novembro 2006.

Floyd, S.; Jacobson V, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on networking, Agosto 1993.

Ghanem, Jean, "Implementation of Load Balancing Policies in Distributed Systems", 2004. 111f Tese (Mestrado) – Curso de Mestrado, The University of New Mexico, New Mexico, New Mexico, 2004. Disponível em: <http://www.eece.unm.edu/lb/papers/jeanthesis.pdf>. Último acesso em 02/2006.

Harchol-Balter, M. at al, "Exploiting Process Lifetime Distributions for Dynamic Load Balan-cing", University of California, Berkeley, ACM Transactions on Computer Systems, Agosto 1997.

ITU-T H.323 TELECOMUNICATION STANDARDIZATION SECTOR OF ITU, "SERIES H: AU-DIOVISUAL AND MULTIMIDIA", Julho. 2003.

RFC 3261. SIP, "Session Initiation Protocol", IETF, Junho, 2002.

Rhee, W.-S at al, "Scalable Quasi-Dynamic-Provisioning-Based Admission Control Mechanism in Differentiated Service Networks", ETRI Jornal, Vol. 26, No. 1, Fevereiro, 2004.

Riedl, R.; Richter, L, "Classification of Load Distribution Algorithms", Department of Computer Science University of Zurich, Zurich, Switzerland, 1996 IEEE.

Zahirazami, B. Seyed at al,"Load Balancing and Call Admission Control in U-MTS-RNC, using Fuzzy Logic", ICCT2003. 2003, IEEE.