

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

Mauro Wilkens Cavalcante

Um Modelo para a Distribuição de Tráfego
Internet a um Cluster de Servidores WEB.

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. João Bosco da Mota Alves
Orientador

Belém, Outubro / 2001

Um Modelo para a Distribuição de Tráfego Internet a um Cluster de Servidores WEB.

Mauro Wilkens Cavalcante

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação na área de concentração de Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Banca Examinadora

Fernando Ostuni Gauthier, Dr.
Coordenador do Curso

João Bosco da Mota Alves, Dr.
Orientador

Paulo José Freitas Filho, Dr.
Co- Orientador

Silvia Modesto Nassar, Dra.

Gustavo Augusto Lima de Campos, M.Eng.

"Deus é um artista.
Ele inventou a girafa, o elefante e a formiga.
Na verdade,
Ele nunca procurou seguir um estilo –
Simplesmente foi fazendo
Tudo o que tinha vontade de fazer”
(Pablo Picasso)

Dedico este trabalho especialmente:
a meus pais: Abner (em memória) e Therezinha;
a minha esposa Leticia;
a meus filhos: Rodrigo e Mariana.

AGRADECIMENTOS:

Agradeço aos Professores do Mestrado da UFSC (Universidade Federal de Santa Catarina), Prof. Dr. Luiz Fernando Jacintho Maia, Prof. Dr. Rosvelter Coelho da Costa, Profa. Dra. Elisabeth Sueli Specialski, Prof. Dr. Paulo Sérgio da Silva Borges, Profa. Dra. Silvia Modesto Nassar, Prof. Dr. Mauro Roisenberg, que estiveram em Belém, transmitindo seus conhecimentos, experiências e especialmente ao Prof. Dr. Paulo José Freitas Filho pela colaboração das idéias fundamentais no planejamento e diretriz deste trabalho e ao Prof. Gustavo Lima de Campos do CESUPA-PA (Centro de Ensino Superior do Pará) - consideração e compreensão - sempre dedicado à coordenação da Turma de Mestrado e também a minha pessoa.

A Diórgenes Carneiro e Eugênio Pessoa, pois formamos uma equipe, durante este período do mestrado, trabalhando juntos com um mesmo objetivo, desde a primeira disciplina até a defesa da dissertação, fruto de nossos esforços, disciplina e dedicação.

Ao meu orientador, Prof. Dr. João Bosco da Mota Alves, que desde o primeiro instante acreditou em mim, nas minhas idéias, no meu trabalho, sendo firme e decisivo nos momentos de especiais decisões para norteá-lo, sendo flexível nos momentos de ouvir, discutir e contribuir para este trabalho.

Aos colegas da turma MCC1 do CESUPA e aos colegas da UFSC pela consideração sempre dispensada a mim, apoiando e colaborando direta ou indiretamente na realização deste.

A todos os meus familiares a quem dedico inteiramente este trabalho, quero ressaltar a importância do apoio fundamental, proporcionando-me as devidas condições para prosseguir nesta caminhada, enfrentando desafios, apoiando-me para continuar seguindo em frente.

SUMÁRIO

Figuras e Tabelas	IX
Resumo	X
Abstract	XI
Introdução	12
1. Introdução Geral	12
1.2 Problema a ser Tratado	15
1.3 Justificativa.	17
1.4 Objetivos.	18
1.4 Estrutura do Trabalho.	20
2. Modelo de Referência de Conexão de Sistemas Abertos (OSI)	21
2.1 Introdução.	21
2.2 Comunicação entre Sistemas.	22
2.3 Camadas do Modelo.	23
2.3.1 Camada física do modelo OSI.	23
2.3.2 Camada de conexão de dados do modelo OSI.	23
2.3.3 Camada de rede do modelo OSI.	24
2.3.4 Camada de transporte do modelo OSI.	24
2.3.5 Camada de sessão do modelo OSI.	24
2.3.6 Camada de apresentação do modelo OSI.	24
2.3.7 Camada de aplicação do modelo OSI.	25
3. Protocolos de <i>Internet</i>	26
3.1 Introdução.	26
3.2 Protocolos de <i>Internet</i> .	27
3.3 O Protocolo IP	28
3.4 Protocolo de Controle de Transmissão (TCP).	30
3.5 Protocolo de Aplicação HTTP.	32
4. Balanceamento Dinâmico de Carga em Servidores <i>Internet</i> (DLBWS).	33
4.1 Introdução.	33
4.2 Acessos com base no DNS.	34
4.2.1 Algoritmos com TTL constante.	36
4.2.1.1 Algoritmos SSA.	36

4.2.1.2 Algoritmos baseados no estado do Servidor.	37
4.2.1.3 Algoritmos baseados no estado do Cliente.	37
4.2.1.4 Algoritmos baseados no estado do Cliente e do Servidor.	37
4.2.2 Algoritmos com TTL variados dinamicamente.	38
4.3 Distribuidor Concentrado (Dispatcher).	39
4.3.1 <i>Dispatcher</i> em um único caminho.	39
4.3.2 <i>Dispatcher</i> em mão dupla.	40
4.3.3 <i>Dispatcher</i> de pacotes pelo endereço físico.	41
4.4 Redirecionamento HTTP.	43
4.5 Distribuição com base no Servidor.	44
4.6 Distribuição com base no Cliente	45
5. Método de Distribuição na Arquitetura <i>Dispatcher</i>	46
5. Método de Distribuição na Arquitetura <i>Dispatcher</i>	46
5.1 Introdução.	46
5.2. Arquitetura <i>Dispatcher</i> .	47
6. Modelo Proposto.	50
6.1 Introdução.	50
6.2 Método de Distribuição - BC.	52
6.3 Característica Principal.	53
6.3.1 Algoritmo de Gerência.	53
6.3.2 Algoritmo Distribuição.	54
6.4 Restrições ao Modelo.	56
7. Análise de Desempenho.	58
7.1 Introdução.	58
7.2 Modelo de Desempenho.	59
7.3 Modelo Lógico.	60
7.4 O Modelo Físico.	61
7.5 Algoritmo BC .	64
7.6 Algoritmo RR.	67
8. Projeto de Experimentos.	69
8.1 Introdução.	69
8.2 Experimentos Realizados como o Modelo BC.	71

8.3 Experimentos Realizados como o Modelo RR.	72
8.4 Análise Comparativa.	73
8. Conclusão e Trabalhos Futuros	75
Referências Bibliográficas	79
GLOSSÁRIO	82

Figuras e Tabelas

Figura 1 – Universal Resource Location	15
Figura 2 – Arquitetura <i>Dispatcher</i>	19
Figura 3 - Camadas do Modelo OSI	22
Figura 4 – Protocolos <i>Internet</i>	27
Figura 5 – Balanceamento com base no DNS	34
Figura 6 – Requisição <i>Internet</i>	35
Figura 7 - <i>Dispatcher</i> em caminho único	39
Figura 8 – <i>Dispatcher</i> em mão dupla.	41
Figura 9 – Redirecionamento HTTP	43
Figura 10 – Modelo <i>Dispatcher</i> BC	52
Figura 11 - Diagrama do Modelo Simulado	59
Figura 12 - Modelo Lógico Simulado	60
Figura 13 - Modelo Físico Simulado	61
Tabela 1– Tempo do Processo dos Algoritmos de Distribuição	62
Tabela 2 – Fatores e Níveis	69
Tabela 3 – Experimentos para o Algoritmo BC	70
Tabela 4 – Experimentos para o Algoritmo RR	70
Tabela 5– Variáveis de Resposta do algoritmo BC	71
Tabela 6 – Variáveis de Resposta do algoritmo RR	72
Tabela 7– Tempo do processamento do <i>switch</i>	73

Resumo

Os provedores de informações da Rede Mundial *Internet* (“*Web sites*”) que apresentam uma alta taxa de requisições de páginas de informação simultâneas necessitam dispor de qualidade no serviço prestado aos seus clientes, por exemplo, o tempo máximo de resposta para apresentar uma página requisitada do *site*.

Um único servidor de páginas *internet* tem o desempenho limitado a uma configuração física máxima o que não lhe permite escalabilidade para atender a grandes demandas de tráfego nos *web sites*.

O Sistema de Balanceamento ou Distribuição de Carga (LBWS) são arquiteturas que consistem no atendimento de requisições de páginas de informação através do uso de vários servidores *internet* (“*Web Server*”), utilizando-se de uma política para a distribuição das requisições aos vários servidores de forma a proporcionar a escalabilidade necessária para atender a demanda estimada de tráfego no *site*.

Estas arquiteturas vêm sendo adotadas ao longo dos últimos anos como as melhores alternativas para manter a qualidade no tempo de resposta dos serviços oferecidos pelos provedores de informação da *internet* (*site*). O uso crescente dos LBWS para atender aos provedores de informação e a diversidades dos serviços implementadas pelos mesmos (som, vídeo, transações, criptografia, etc) permite ao mercado dispor de diferentes tipos LBWS direcionados a situações específicas conforme a necessidade de cada sites.

Neste trabalho, propõe-se um modelo de LBWS, denominado - rajada controlada (*Burst Control* - BC), que pode ser implementado na camada de nível quatro (4) ou sete (7) do padrão de Conexão de Sistemas Aberto OSI.

O Algoritmo BC executa a racionalização da distribuição de requisições *internet* dos clientes a uma família de servidores heterogêneos de páginas estáticas. A principal característica do modelo é estabelecer uma carga máxima de processamento a cada um dos servidores *WEB*. Uma análise de desempenho e uma comparação entre o modelo proposto e o modelo *Round-Robin* também é apresentada.

Abstract

The Popular Web site grows more sophisticated for keeping up with the processing demands becomes increasingly difficult, because neither single multiprocessor-based server nor support the ever-increasing request load. Web site that have to guarantee an acceptable response time, can be provided by load-balancing or load-sharing in a distributed Web Server architectures that schedule client request among the multiple server nodes in a user-transparent way. As the demand for load-balancing or load-sharing functions continues to grow, the market is expanding very on fast last years, with variety products of Load-Balancing or Load-Sharing covered both the switch and router markets.

In this paper we propose a scheduling police Load-Sharing-Dispatcher, namely *Burst Control* (BC), for Web switches operating at layer four or seven of the OSI protocol stack. Its goal is to improve rationalize load sharing in Web Clusters that provide static information. We demonstrate, for layer four, through a simulation experiments that dispatching polices aiming to improve control in sending request that not exceed the server performance give best results for traditional Web publishing sites providing static information and some simple database searches. The proposed algorithm has the additional benefit of guaranteeing a maximum Web server response time.

Introdução

1. Introdução Geral

Como se sabe, o ano de 2000 foi marcado pela euforia com a *internet*, em que todas as empresas corriam para marcar sua presença na Web. Além disso, aquele foi o momento da explosão das pontocom. Garantir seu espaço no mundo eletrônico era primordial para as empresas [1].

A presença corporativa na rede internet pode ser realizada de várias formas e com vários objetivos, mas é fato determinante a rapidez de seu acesso e a disponibilidade dos serviços oferecidos pela entidade provedora (*site*) [2].

O crescimento explosivo do tráfego na *World Wide Web* (WWW) traz em consequência o rápido crescimento da taxa de requisições sobre os com conhecidos e visitados diariamente na internet - *site* populares. Os *site* populares podem sofrer grandes congestionamentos, quando são solicitadas milhões de requisições por segundo, principalmente em ocasiões especiais de algum acontecimento ou evento.

Como exemplo, pode citar-se a Empresa Victoria's Secret que, na realização do seu segundo desfile na internet, usou toda tecnologia possível para preparar a transmissão do desfile em áudio e vídeo (*Webcast*) de 55 minutos, assistida por cerca de dois milhões de espectadores, cujo tempo médio para acessar o *site* durante a *Webcast* foi de cinco segundos, com 97,9% de disponibilidade. Desempenho medido pela Empresa Keynote Systems [2].

Ter um *Web site* rápido para algumas companhias é uma necessidade básica, assim como deve suportar quaisquer picos de acessos que surjam na rede, como exemplo a corretora “online” Charles Schwab [3], que para suportar as significativas flutuações nas bolsas de valores, momentos estes em que centenas de milhares de clientes tentam comprar ou vender ações, criou uma rede de computadores unindo aproximadamente 1,2 mil servidores e dez computadores de grande porte (*mainframes*), atingindo valores de 95 mil acessos simultâneos e mais de 600 mil pedidos [2].

Outro exemplo, no Governo do Estado de São Paulo, o patrocinador da solução de e-business foi ninguém menos que o então governador Mário Covas [1]. Apesar de ser um órgão público, a experiência adquirida com o projeto de governo eletrônico

apresentam muitas lições para o mundo corporativo. Com 21 secretarias estaduais, o governo estava financeiramente quebrado e não havia ferramentas que oferecessem um controle centralizado. "Até 1995 o mais próximo que chegávamos de uma forma eletrônica de comunicação era a utilização do fax", conta Roberto Meizi Agune, responsável pelo SEI – Sistema Estratégico de Informações da secretaria do governo e gestão estratégica. Com mais de 600 órgãos ligados à administração estadual, fica impossível que o governo tenha um projeto único e centralizado de governo eletrônico. "A secretaria realizou alguns projetos primordiais, a rede foi o primeiro deles, e criamos diretrizes para o trabalho, que fica a cargo de cada órgão", conta Dalmo Nogueira Filho, secretário adjunto da secretaria do governo e gestão estratégica. Assim, contar com o comprometimento de todas as partes é crucial para o sucesso. "A internet nasceu de uma forma anárquica, pois tudo converge para um sistema que respeita padrões. O mesmo acontece aqui", filosofa Nogueira Filho [1].

Uma das soluções de grande destaque e visibilidade é o Controle de Contratos de Serviços Terceirizados, que é responsável pelo gerenciamento de seis mil contratos de 507 diferentes tipos de serviço. A solução consiste de um grande banco de dados com todos os contratos celebrados no estado, de onde são retiradas informações usadas na definição de padrões, como valor pago e validade dos contratos. Entre janeiro de 1995 e janeiro de 2001 esse sistema permitiu uma redução de gastos de mais de 28%, gerando uma economia de 4,41 bilhões de reais [1].

O peso das ferramentas e aplicativos baseados em internet tornou premente a necessidade de servidores muito mais robustos para suportar a infra-estrutura das empresas que querem se conservar competitivas em seu segmento. "Para mantermos nossas 5,5 mil posições de atendimento em todo o mundo com a mesma performance, em qualquer hora do dia ou da noite, mantemos um ambiente que usa normalmente cerca de 50% de sua capacidade. Essa sobra existe para evitarmos surpresas", exemplifica Tony Facó, gerente de suporte técnico da TAM – Transportes Aéreos Regionais [4].

Há pouco mais de um ano no ar, o *site* do Banco do Brasil registrou um total de 85 milhões de transações em seu *office banking*. Somente em junho/2000, o número de transações pela *Web* atingiu 13,1 milhões. De acordo com dados da instituição, de

janeiro a junho, o número de transações *online* aumentou de 25% para 65% do total. O banco espera que 220 mil clientes se cadastrem no *site* até o final do ano [4].

Com a evolução da velocidade e tráfego da estrutura da rede internet no mercado mundial a tendência é o tráfego das redes remotas (WAN) ser igual ou superior às redes locais (LAN), conforme previsão do Gartner Group [5].

Como pode ser observado nos exemplos citados, o grande problema dos administradores do ambiente *internet* das empresas é dimensionar o tráfego e sua expectativa de evolução, para projetar e manter a arquitetura e a escalabilidade dos recursos computacionais para atender a demanda de forma eficiente.

1.2 Problema a ser Tratado

Todos os pontos de presença corporativa são definidos por um endereço (*link*) definido por um *Universal Recurse Location* (URL) que apontam para o ponto de entrada do servidor de presença corporativa, como pode ser visto na figura 1.

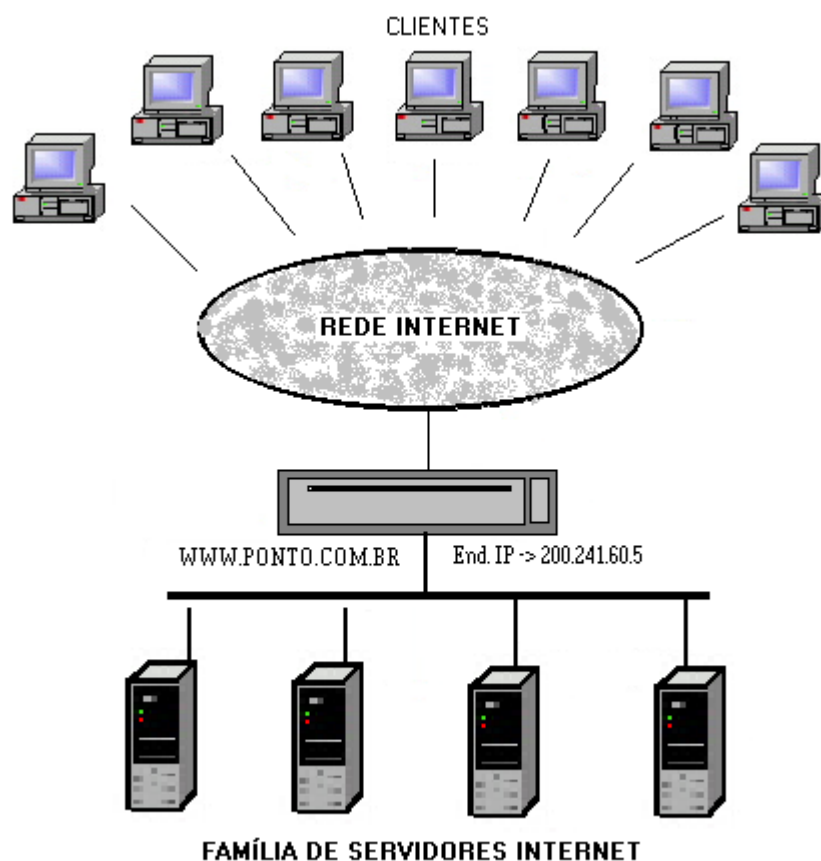


FIGURA 1 – UNIVERSAL RECURSE LOCATION

O *Domain Name Server* (DNS) [6] é como uma grande lista de endereços eletrônicos que permite relacionar os endereços URL a nomes de domínio pelos quais são de conhecimento dos clientes. Essas informações são armazenadas na *Internet*, com os servidores designados (ou principais) responsáveis por grupos de URL de um determinado domínio.

Quando um cliente solicita acesso a uma página *Web* ou envia um correio eletrônico (*e-mail*), essa ação gera uma consulta ao seu servidor DNS local, que por sua vez, consulta seus pares na rede a fim de encontrar o servidor DNS autorizado para o

site. Quando o servidor principal é localizado, ele informa o endereço URL (número de *Internet Protocol* - IP) correto para o servidor DNS que o contactou, e então o passa adiante até que chegue ao cliente.

O fluxo de requisições a esse endereço de acesso nos *site* populares mantém constantemente seus administradores em busca de soluções e alternativas que permitam aos servidores de serviços *internet* (*web server*), a capacidade de processamento de requisições para atender a demanda requisitada pelos usuários. O caminho mais óbvio nesta solução é realizando o crescimento dos recursos computacionais deste único ponto de acesso, contudo, às vezes, esta solução ainda não é suficiente, pois o número máximo de requisições que um único servidor pode suportar é limitado e pode não ser grande o suficiente para atender a crescente demanda de requisições.

Para possibilitar uma escalabilidade no tráfego requisitado, existe a necessidade de soluções para que mais de um servidor possa estar habilitado a responder as requisições *internet* [7]. A esse conjunto de servidores em um mesmo local denomina-se família servidores provedores de serviço *internet* (*web servers*).

Para alguns *site*, a solução localizada em um único ponto global de acesso, ainda não permite que tenha condições necessárias para operar conforme os requisitos de tráfego e segurança. Para garantir a presença corporativa na rede, de modo eficaz na *Internet*, isso significa, direcionar as solicitações da *Web* para vários *site* geograficamente distribuídos, de modo que se permita um conjunto de servidores de presença corporativa ativos que proporcionem às informações, um fluxo contínuo e com a máxima independência de falhas nos servidores e respectivos meios de comunicação.

A implementação da distribuição das requisições dos usuários para uma família de servidores necessita de uma arquitetura ao *site* e uma técnica de distribuição de requisições para os servidores, para que os usuários tenham suas requisições atendidas sempre pelo mesmo servidor, garantindo o processo de conexão.

Não há nada que impeça os administradores de redes de utilizar os próprios serviços oferecidos por um servidor DNS para construir redes de servidores *Web* distribuídos. Mas o DNS não oferece muito mais do que um balanceamento de carga básico. O DNS também pode ser configurado com servidores distribuídos geograficamente, hospedando o mesmo conteúdo. Nesta configuração, há vários endereços de IP para o mesmo domínio. Quando o servidor DNS é consultado, ele

utiliza um esquema de rodízio (*round-robin* - RR) para selecionar um endereço de IP, apresentando o primeiro endereço para uma consulta; o segundo endereço, para a consulta seguinte, e assim por diante. O problema é que essa abordagem não direciona os usuários para a central de dados mais próxima ou com desempenho mais rápido.

É fundamental a distribuição otimizada das requisições na internet, pois os produtos para balanceamento de cargas são a melhor solução existente no mercado, no momento, para soluções de distribuição eficazes. Todos eles agem como servidores DNS, mas acrescentam inteligência — selecionando *site* com base em parâmetros como: proximidade, tempo de resposta/latência, perda de pacotes (avaliam o desempenho de link), carga do servidor local e condições deste servidor (analisam o desempenho e a situação atividade dos servidores individuais).

O balanceamento pode ser visto em duas situações diferentes, um localmente, que visa à distribuição da carga entre os servidores locais; e outra globalmente, que visa à distribuição geograficamente e globalmente equilibrada, possibilitando maior segurança, tais como ataques de *hacker*, incêndios, enchentes, falhas de equipamentos de comunicação e servidores de serviços. As corporações que hospedam aplicativos de comércio eletrônico ou destinado a informações importantes em tempo real, pois lidar com problemas desse tipo não só faz parte do dia-a-dia como também, do plano de contingência da Organização.

1.3 Justificativa.

Os produtos de balanceamento de carga são fundamentais e com importância crescente para as Organizações que disponibilizam seus negócios através da *internet*, pois os computadores servidores *internet* escaláveis e multiprocessados não são suficientes para suportar o número de acessos existentes em *site* populares ou em eventos especiais.

Para suportar esta demanda de tráfego, existem, no mercado, diversas soluções de equipamentos (hardware) e também de software, que permitem aos *sites* o controle eficaz de demanda de tráfego.

Os métodos empregados na busca de soluções à distribuição inteligente de tráfego (carga), implementados nos roteadores ou *switches* de balanceamento de carga, acarretam uma latência devido ao processamento das informações obtidas nas mensagens provenientes dos servidores *WEB*, como por exemplo: CPU, sessões ativas, filas.

A pesquisa e o teste de alguns algoritmos que proporcionem a tomada de decisão adequada, sem a necessidade de uma grande latência nos equipamentos, por conta da busca de informações da rede, do cliente e dos servidores, é a principal motivação deste trabalho.

Os principais benefícios atingidos com o balanceamento de carga são a distribuição da carga entre servidores, conforme regras, balanceamento geográfico, aceleradores de criptografia *Secure Socket Layer* (SSL) e tolerância a erro em virtude de não existir a dependência em um único servidor.

1.4 Objetivos.

Denomina-se de *Dispatcher* a uma política que permita a seleção das requisições dos clientes com base em um distribuidor para uma família de servidores possibilitando um processo de crescimento gradual ao desempenho do *site internet*. O *Dispatcher*, também denominado de *Web switch* ou *switch*, é o responsável por receber todo o tráfego oriundo dos clientes e distribuí-los aos servidores *internet*, como mostra a figura 2.

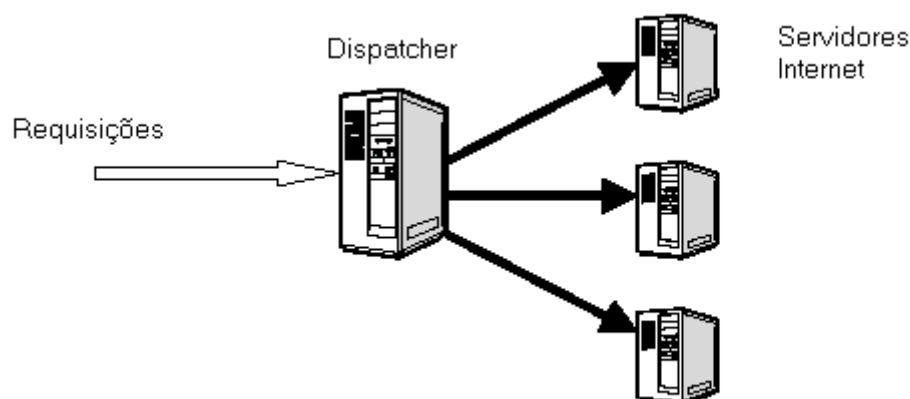


FIGURA 2 – ARQUITETURA *DISPATCHER*

Propõe-se neste estudo um método de distribuição de tráfego para um *Dispatcher*, na camada 4 do modelo de referência de conexão de sistemas abertos (OSI), com um procedimento de distribuição que não ultrapasse o limite estabelecido de desempenho dos servidores *web* (requisições por segundo), através de um algoritmo de distribuição das requisições dos clientes aos servidores *Web* aplicadas seqüencialmente na forma de rajada de requisições, isto é, em lotes, limitando-se ao desempenho especificado a cada servidor.

As principais características do modelo são:

- 1) Permitir que os servidores jamais ultrapassem seus limites de desempenho, permitindo um tempo máximo de resposta estabelecido a uma requisição ao *site*;
- 2) Permitir um aproveitamento do recurso do Sistema Operacional do Servidor, durante o procedimento de abertura de conexão e manipulação de fila de entrada, motivado pelo procedimento de rajada de requisições do mesmo tipo;
- 3) Permitir que a distribuição seja realizada sem requerer informações externas ao *Dispatcher*, com objetivo de simplificar a arquitetura, a funcionalidade, não necessitar de tráfego na rede interna do *site* e processamento adicional no servidor;
- 4) Permitir que os parâmetros de distribuição possam ser ajustados conforme a demanda de tráfego;
- 5) Permitir um algoritmo simples e rápido, reduzindo a possibilidade do *Dispatcher* vir a ser um ponto de estrangulamento de tráfego do *site*.

1.4 Estrutura do Trabalho.

Os capítulos 2º e 3º introduzem ao modelo OSI e alguns protocolos da *internet*.

O capítulo 4º apresenta as diversas formas de implementação de arquiteturas para o balanceamento de requisições na *internet*.

O capítulo 5º descreve o método de distribuição na arquitetura *Dispatcher*.

O Capítulo 6º apresenta o algoritmo do modelo proposto e descreve as características e funcionalidades.

O capítulo 7º apresenta o procedimento de modelagem e resultados obtidos.

O capítulo 8º apresenta a conclusão e recomendações de trabalhos futuros.

2. Modelo de Referência de Conexão de Sistemas Abertos (OSI)

2.1 Introdução.

O modelo de referência de conexão de sistemas abertos, OSI, especifica um conjunto de regras de um programa de computador que são transferidas pela rede até um programa em outro computador. O modelo foi desenvolvido pela *International Organization for Standardization* (ISO) em 1984 e atualmente é a principal referência de modelo de arquitetura para comunicações entre computadores. É um modelo conceitual composto de sete camadas, cada uma especificando funções de rede particulares. Todas as camadas são razoavelmente auto-suficientes, de forma que as funções atribuídas a cada uma delas possam ser implementadas de maneira independente.

2.2 Comunicação entre Sistemas.

As informações transferidas de uma aplicação de um computador para outro computador, utilizando-se do modelo OSI, necessitam realizar um fluxo de procedimentos que iniciará a partir de uma camada do modelo à camada subseqüentemente inferior até sua transferência ao outro computador que receberá pela camada inferior e passará para a subseqüentemente superior, procedendo sempre desta forma o fluxo de conversação entre os mesmos, conforme exemplificado na figura 3.

Uma determinada camada OSI, geralmente pode se comunicar com a camada imediatamente acima, ou imediatamente abaixo, ou a camada na mesma hierarquia de um computador, e sempre com a camada correspondente em outro computador.

O procedimento de comunicação entre camadas é realizado através de serviços oferecidos especificados na camada, onde três elementos básicos estão relacionados aos serviços das camadas: o usuário do serviço, o provedor e o ponto de acesso do serviço (SAP).

As sete camadas OSI utilizam um controle de informações para realizar a comunicação entre as camadas correspondentes existentes entre os sistemas de computadores, que consiste em solicitações e instruções específicas, implementadas em duas formas: cabeçalho (*header*) e rodapé (*trailer*) [8]. Os cabeçalhos são anexados ao início dos dados recebidos das camadas superiores. Os rodapés são anexados ao final dos dados recebidos das camadas superiores. Não é obrigatório o acréscimo de um cabeçalho ou de um rodapé aos dados recebidos das camadas superiores.

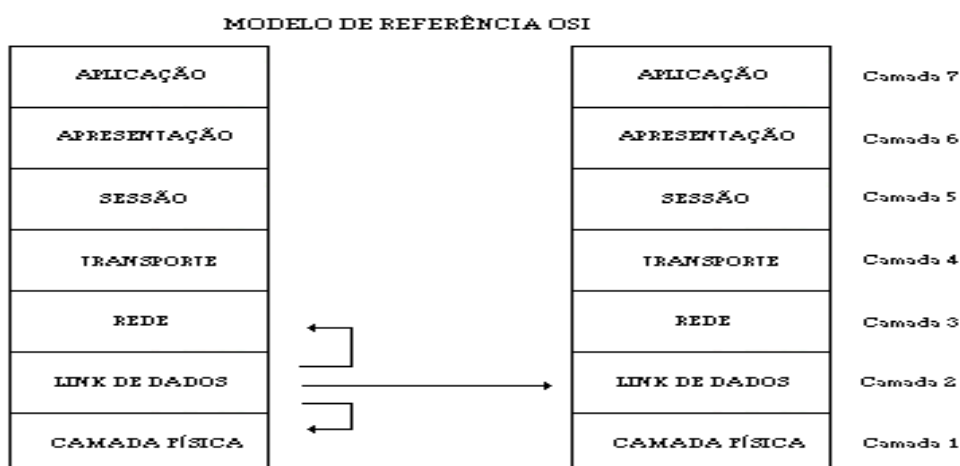


FIGURA 3 - CAMADAS DO MODELO OSI

2.3 Camadas do Modelo.

As sete camadas do modelo de referência OSI [8] são:

Camada 7 – Camada de Aplicação;

Camada 6 – Camada de apresentação;

Camada 5 – Camada de sessão;

Camada 4 – Camada de transporte;

Camada 3 – Camada de rede;

Camada 2 – Camada de conexão (*link*) de dados;

Camada 1 – Camada física.

2.3.1 Camada física do modelo OSI.

A camada física define as especificações elétricas, mecânicas, procedimentos e funcionalidades, destinadas a ativar, manter e desativar a conexão (*link*) físico entre os sistemas de comunicação de rede.

2.3.2 Camada de conexão de dados do modelo OSI.

A camada de conexão de dados proporciona o trânsito confiável de dados por uma conexão física da rede. Diferentes especificações da camada de conexão de dados definem diferentes características de rede e protocolo. O endereçamento físico define como os dispositivos são enviados a uma camada de conexão de dados.

O Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) subdividiu a camada de conexão de dados em duas subcamadas: Controle de *Link* Lógico (LLC) e Controle de Acesso de Mídia (MAC). O LLC administra a comunicação entre dispositivos em uma única conexão de uma rede. O MAC administra o acesso de protocolos ao meio físico da rede, onde é definido o endereço MAC de um dispositivo na rede, permitindo, assim, que sejam identificados de maneira única na camada de conexão de dados.

2.3.3 Camada de rede do modelo OSI.

A camada de rede proporciona o roteamento e funções relacionadas, permitindo que várias conexões de dados sejam combinadas em uma coleção de redes individuais, conectadas por meio de dispositivos que funcionam como uma única grande rede. Isto é realizado pelo endereçamento lógico de dispositivos.

2.3.4 Camada de transporte do modelo OSI.

A camada de transporte implementa serviços confiáveis de transporte de dados de uma coleção de redes individuais, conectadas por meio de dispositivos, que funcionam como uma única grande rede. As funções tipicamente realizadas nesta camada, incluem o controle de fluxo, multiplexação, administração de circuito virtual, verificação e recuperação de erros.

2.3.5 Camada de sessão do modelo OSI.

A camada de sessão estabelece, gerência e finaliza as sessões de comunicação entre entidades da camada de apresentação.

2.3.6 Camada de apresentação do modelo OSI.

A camada de apresentação oferece uma variedade de funções de codificação e de conversão destinadas aos dados da camada de aplicação. Alguns exemplos de esquemas de conversão e codificação da camada de apresentação incluem formatos comuns de representação de dados, conversão de formatos de representação de caracteres, esquemas comuns de compactação de dados e esquemas comuns de criptografia. As

implementações da camada de apresentação não costumam estar associadas a uma determinada pilha de protocolos.

2.3.7 Camada de aplicação do modelo OSI.

A camada de aplicação é a camada onde o programa do computador e os usuários interagem de forma direta, implementando um componente de comunicação. As funções da camada de aplicação costumam incluir a identificação de parceiros de comunicação, a determinação de disponibilidade de recursos e a sincronia da comunicação.

3. Protocolos de *Internet*

3.1 Introdução.

O modelo OSI proporciona uma estrutura conceitual para a comunicação entre computadores, mas o modelo em si não é um método de comunicação. A comunicação entre entidades é realizada através de protocolos. No contexto de rede de dados, um protocolo é um conjunto formal de regras e convenções que controlam como os computadores realizam o intercâmbio de informações em redes.

Neste capítulo, apresentam-se alguns protocolos mais utilizados na *internet*, com objetivo introdutório e básico para os demais capítulos desta dissertação conforme definidos em [6], [8] e [9].

3.2 Protocolos de *Internet*.

Os protocolos de *internet* consistem em um conjunto de protocolos de comunicação, dentre os quais os mais conhecidos são o protocolo de controle de transmissão (TCP) e o protocolo de *internet* (IP).

Os protocolos de *internet* começaram a ser desenvolvidos na primeira metade dos anos 70, quando a *Defense Advanced Research Projects Agency* (DARPA) estava interessada em estabelecer uma rede de comutação de pacotes para facilitar a comunicação entre os diferentes sistemas de computadores das instituições de pesquisa. Posteriormente o TCP/IP foi incluído na versão *Berkeley Software Distribution* (BSD) do UNIX e se tornou, desde então, a fundação em que a *internet* e a *World Wide Web* (WWW) estão baseadas.

A documentação dos protocolos de *internet* (incluindo novos ou revisados) e as políticas são especificadas em relatórios técnicos, chamados de Solicitações para Comentários (Request for Comments - RFC), publicados e depois revisados e analisados pela comunidade de *internet*.



FIGURA 4 – PROTOCOLOS *INTERNET*

3.3 O Protocolo IP

O protocolo *internet* (IP) é um protocolo da camada de rede, que contém as informações de endereçamento da informação e algumas informações de controle que permitem o roteamento de pacotes.

O protocolo IP está documentado na RFC 791 e tem duas responsabilidades primárias: fornecer a entrega com menor esforço e independente de conexão, de pacotes de informações (chamados de datagramas) na rede; e providenciar a fragmentação e a remontagem de datagramas para suportar *link* de dados com diferentes tamanhos de unidades de transmissão máxima (MTU).

Um pacote IP contém vários tipos de informações em quatorze campos:

Versão – indica a versão do produto.

Comprimento do cabeçalho IP (IHL) – indica o comprimento do cabeçalho do datagrama em palavras de 32 *bits*.

Tipo de serviço – especifica como um protocolo da camada superior, solicita a manipulação do datagrama atual e atribui vários níveis de importância aos datagramas.

Comprimento total – especifica o comprimento em *bytes* do pacote IP inteiro, inclusive os dados e o cabeçalho.

Identificação – contém um inteiro que identifica o datagrama atual. Este campo é utilizado para ajudar a reunir os fragmentos do datagrama.

Flags – consiste em um campo de três *bits*, dos quais os dois *bits* de baixa ordem controlam a fragmentação, e o intermediário especifica se o pacote é o último fragmento.

Offset de fragmento – indica a posição dos dados do fragmento em relação ao início dos dados no datagrama original, usado na reconstrução do datagrama original.

Tempo de vida – mantém um contador que é decrementado gradualmente até zero, quando o datagrama será descartado.

Protocolo – indica qual o protocolo da camada superior recebe os pacotes de entrada depois do processamento do protocolo IP estiver concluído.

Checksum do cabeçalho – usado para assegurar a integridade do cabeçalho.

Endereço de origem – especifica o nó remetente.

Endereço de destino – especifica o nó destinatário

Opções – permite ao protocolo suportar várias opções, como segurança.

Dados – Contém informações da camada superior.

Cada computador da rede recebe um único endereço lógico com tamanho de 32 bits, dividido em duas partes: o número da rede e o número do computador. O número da rede é atribuído pelo Centro de informações de redes da Internet (InterNIC). O número do computador é atribuído pelo administrador da rede local.

O endereço IP é agrupado em oito *bits* por vez, separados por pontos, e é representado no formato decimal. Cada *bit* no octeto tem um peso binário (128, 64, 32, 16, 8, 4, 2, 1), sendo o valor mínimo zero e o valor máximo 255.

O endereçamento IP está dividido em 5 classes diferentes, chamadas A, B, C, D e E, classificados conforme o estado do *bit* de mais alta ordem respectivamente (0, 10, 110, 1110, 1111), sendo que somente as classes A, B e C estão disponíveis para o uso comercial.

As redes IP podem ser divididas em redes menores chamadas sub-redes, que proporcionam uma utilização mais eficiente dos endereços de rede e a possibilidade de divisão de tráfego. As sub-redes estão sobre uma administração interna da organização.

3.4 Protocolo de Controle de Transmissão (TCP).

O protocolo da camada de transporte TCP proporciona uma transmissão confiável de dados em um ambiente IP, oferecendo os seguintes serviços: transferência de dados contínua, confiabilidade (orientado a conexão), controle de fluxo eficiente, operação simultânea de enviar e receber e multiplexação de várias conversações simultâneas da camada superior em uma única conexão.

Sendo um protocolo orientado à conexão, o estabelecimento da mesma é executado através do mecanismo de “*handshake* de três vias”, isto é, sincroniza as duas extremidades de uma conexão com a concordância da seqüência inicial de números, cujo procedimento ocorre da seguinte maneira:

O primeiro computador (CP1) inicia uma conexão, enviando um pacote com um número seqüencial inicial (X_{cp1}) e o *bit* SYN definido para indicar um pedido de conexão. O segundo computador (CP2) recebe o SYN, registra o número de seqüência X_{cp1} e envia uma resposta $ACK = X_{cp1} + 1$, e registra o número de seqüência X_{cp2} . O ACK com a seqüência incrementada de um significa que o mesmo recebeu até o *byte* X_{cp1} e está aguardando o *byte* $X_{cp1} + 1$, técnica esta chamada de reconhecimento antecipado. O computador CP1 reconhece a transmissão do CP2 enviando uma resposta com o $ACK = X_{cp2} + 1$, e então inicia a transferência de dados.

O protocolo TCP utiliza o conceito de janela móvel TCP a fim de que os computadores possam enviar vários *bytes* ou pacotes antes de aguardar por um reconhecimento do mesmo, sendo que o tamanho da janela é informado em todos os pacotes, expressa em *bytes*. O tamanho inicial da janela é indicado na configuração da conexão, podendo variar ao longo da transferência, podendo inclusive ser zero, que significará não enviar dados.

As descrições a seguir resumem os campos do pacote TCP:

Porta de origem e porta de destino – identifica os pontos em que os processos de origem e destino da camada superior recebem os serviços TCP.

Número seqüencial – normalmente especifica o número atribuído ao primeiro *byte* de dados na mensagem atual. Na fase de estabelecimento da conexão, esse campo identifica o número seqüencial inicial a ser utilizado.

Número de reconhecimento – contém o número seqüencial do próximo *byte* de dados que o remetente do pacote espera receber.

Offset de dados – indica o número de palavras de 32 *bits* encontradas no cabeçalho do protocolo.

Reservado – uso futuro.

Flags – contém informações de controle, como SYN, ACK e FIN (finaliza conexão).

Janela – define o tamanho da janela de recebimento de remetente.

Checksum – controle para verificação de integridade do cabeçalho.

Ponteiro urgente – informa o primeiro *byte* de dados urgente do pacote.

Opções – especifica várias opções do protocolo.

Dados – contém os dados da camada superior.

3.5 Protocolo de Aplicação HTTP.

O protocolo *Hypertext Transfer Protocol* (HTTP) é um protocolo da camada de aplicação para sistemas de informação de *Hypermídia*, em uso pelo *Web* Mundial desde 1990. As mensagens de HTTP consistem em requisições do cliente para um servidor e respostas deste servidor ao cliente.

A comunicação através do protocolo de HTTP normalmente acontece em cima de conexões de TCP/IP. Isto não impede do HTTP ser implementado em cima de qualquer outro protocolo na *Internet*, ou em outras redes.

O protocolo HTTP/1.0, definida pela RFC 1945 [10], e a versão HTTP/1.1, definida pela RFC 2616 [11], e tem como principal característica permitir a construção de sistemas que sejam independentes dos dados que são transferidos.

Em HTTP/1.0, a maioria das implementações usa uma conexão nova para cada conjunto de requisição/resposta (*request/response*). Em HTTP/1.1, uma conexão pode ser usada para um ou mais requisições/respostas, utilizando o recurso de uma conexão permanente.

Antes de estabelecer uma conexão HTTP, uma conexão de TCP foi estabelecida para ir buscar cada URL. Conexões de HTTP permanentes têm várias vantagens tais como: abrindo e fechando menos conexões de TCP, economizado tempo de CPU nos clientes e servidores, e otimizando a memória usada pelo protocolo TCP no estabelecimento de cada conexão.

O uso de uma conexão permanente permite, além dos benefícios anteriores, a possibilidade do cliente de realizar múltiplos pedidos sem esperar por cada resposta, otimizando o uso de uma única conexão de TCP, permitindo um menor tempo de resposta ao cliente, reduzindo o tráfego na rede e o número de pacotes TCP do procedimento de conexão e desconexão. Conseqüentemente a latência nas requisições subseqüentes será reduzida, visto que não há perda de tempo no estabelecimento da conexão TCP, além de que podem ser informados erros sem a penalidade de fechar a conexão de TCP.

4. Balanceamento Dinâmico de Carga em Servidores *Internet* (DLBWS).

4.1 Introdução.

A necessidade de ampliação de capacidade nos servidores de conteúdo *internet* (*Web Server*) conforme a demanda dos usuários teve início como a instalação de *sites* espelhos (*mirrored-server architecture*) que possibilitaram aos usuários selecionar manualmente um *site* de uma lista de URL independentes [12].

Esta solução não contempla uma arquitetura transparente ao usuário e não permite o controle da distribuição através dos *sites* espelhos, sendo totalmente estática e não balanceada.

A arquitetura de distribuição realizada através de um roteamento das requisições tornou-se mais promissora e com uma transparência aos usuários. Esta arquitetura permite um ganho de capacidade de processamento e permite uma grande escalabilidade e disponibilidade aos sistemas de serviço *internet*.

Neste capítulo descrevem-se as arquiteturas utilizadas no balanceamento dinâmico de carga. (classificadas conforme [13]).

4.2 Acessos com base no DNS.

Esta arquitetura, sintetizada na figura 5, é realizada através de uma interface única virtual com a *internet* no nível da URL, durante sua conversão do nome de domínio para o endereço do Protocolo *Internet (Internet Protocol -IP)* do servidor *internet*.

Uma requisição *internet*, cujo diagrama é apresentado na figura 6, é normalmente realizada através do endereço URL do *site*, por exemplo: *www.site.com*. Esta solicitação necessitará ter seu endereço resolvido para o respectivo endereço IP que constará da requisição. Esta fase de pesquisa inicial poderá ter uma solução em nível local, através do servidor de DNS da instalação, e caso não seja encontrado, será resolvido pela hierarquia da resolução de DNS até obter do servidor de DNS do *site* procurado o respectivo endereço IP. Todo este procedimento entre o nome simbólico (URL) e o endereço IP é realizado através de um protocolo de procedimentos específicos de resolução até o servidor apropriado que responderá o endereço IP do nome simbólico [14]. Neste procedimento de busca entre o cliente e o *site* de DNS em questão, a requisição poderá passar através de vários outros servidores de Domínio que poderão resolver o endereço através da busca deste mapeamento IP do conteúdo de seu *cache (network-level address caching)* para reduzir o tráfego na rede, podendo em alguns casos, inclusive ser resolvido pelo navegador do cliente (*client-level address caching*).

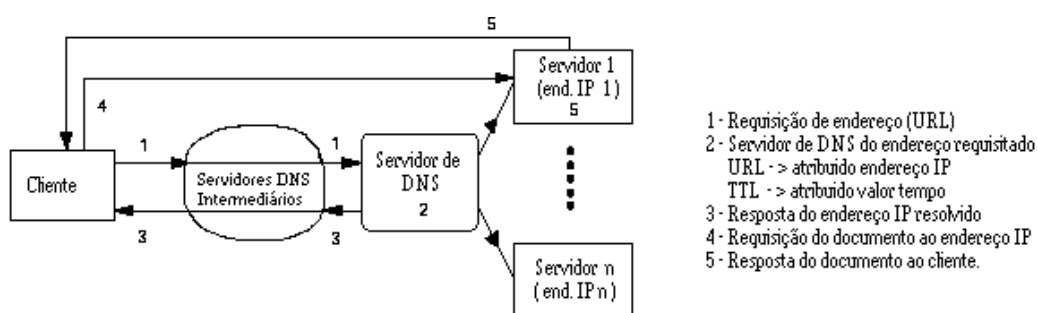


FIGURA 5 – BALANCEAMENTO COM BASE NO DNS

O servidor de DNS ao informar o endereço IP, também é especificado um tempo de validade para esta informação, tempo este conhecido por *Time-To-Live (TTL)*, para ser usado no armazenamento do mapeamento IP. Quando o TTL expira no *cache* dos

servidores de DNS, as solicitações recebidas são encaminhadas em frente para que possam achar um DNS válido possibilitando, assim, que o servidor de DNS forneça outro endereço IP de um servidor espelhado, possibilitando desta forma uma divisão de tráfego entre servidores Web do *site*.

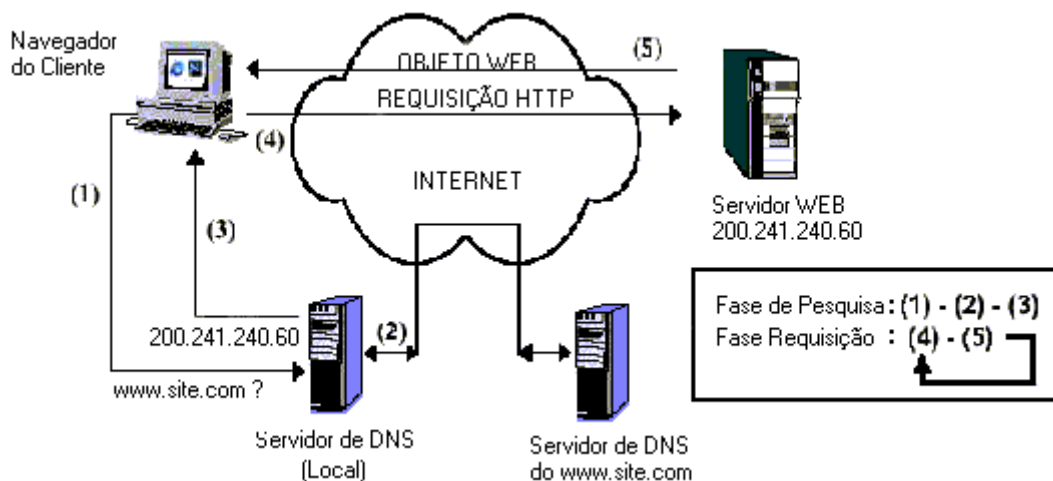


FIGURA 6 – REQUISIÇÃO INTERNET

Existem duas alternativas para a resolução do endereço de URL como pode ser visto na figura 6. Se algum servidor de Domínio entre o cliente e o servidor de Domínio do *site* a ser requisitado dispuser de uma válida resolução para aquele determinado endereço, ele assim será resolvido e informado ao cliente, e a outra solicitação irá até o servidor de DNS do *site* solicitado (*cluster* DNS) que selecionará conforme um algoritmo um endereço IP entre a família de servidores disponíveis e estipulará um tempo TTL para esta resolução retornando ao cliente, sendo que todos os servidores de DNS intermediários neste procedimento de retorno ao cliente irão armazenar no respectivo *cache* esta associação e o tempo de vida da mesma.

Como podemos observar o controle de endereçamento realizado através do DNS é limitado aos seguintes fatores: ao tempo de vida da associação (TTL), armazenada nos servidores de DNS intermediários e no *caching* do navegador do cliente (neste caso o tempo de vida é desconsiderado); o TTL não pode ser utilizado com valor próximo de zero, devido os servidores de DNS não cooperativos ignorarem valores menores que 120ms; não existe um controle efetivo do número de requisições que estão sendo realizados em um determinado instante por um servidor *internet*, tornando, assim, um ponto de gargalo de tráfego do servidor.

Considerando que o TTL da arquitetura DNS pode assumir duas situações distintas, isto é, trabalhar com um TTL constante, ou com um TTL variado de forma dinamicamente pelo servidor de DNS, isto possibilita implementar algoritmos diferentes para trabalhar em cada caso.

4.2.1 Algoritmos com TTL constante.

Na utilização de um TTL constante, a escolha de um servidor *internet* poderá utilizar um algoritmo com base em: nenhuma informação (*System stateless algorithms* (SSA)); cliente, servidor *internet*; combinação de mais de um desses fatores.

4.2.1.1 Algoritmos SSA.

O mais importante algoritmo para este caso é chamado de *Round Robin* DNS (DNS-RR) implementado pela *National Center for Supercomputing Applications* [15]. Esta foi a primeira implementação de distribuição homogênea entre servidores *internet* e representa um passo básico para as novas tecnologias.

O método DNS-RR realiza uma distribuição das requisições para uma lista de servidores disponíveis cadastradas no servidor de DNS, de forma seqüencial na lista, sendo esta uma lista circular, isto é, após o último elemento retorna a distribuição ao primeiro elemento da lista.

A distribuição de carga (DC) utilizando o DNS-RR não realiza um balanceamento efetivo, devido aos diversos dispositivos na rede *internet* guardarem a informação de endereçamento (*cache* e servidores de DNS), tornando-se sem efeito quando o acesso é realizado por clientes de um mesmo domínio, que possivelmente serão destinados a um mesmo servidor devido à resolução de endereçamento realizada em função do *cache* do servidor de DNS deste domínio. Estas restrições podem causar um desequilíbrio e uma sobrecarga dos servidores devido: ao método não considerar a disponibilidade e capacidade dos servidores; a obrigatoriedade de todos os servidores terem o mesmo

conteúdo ou oferecerem o mesmo serviço, isto é, assume-se que o ambiente de distribuição das requisições é homogêneo.

4.2.1.2 Algoritmos baseados no estado do Servidor.

A distribuição DNS com controle implementada pelo trabalho da SunSCALR [16] permite combinar a técnica DNS-RR com mecanismos de alerta enviado pelos servidores, possibilitando um controle de distribuição, tentando evitar a sobrecarga dos servidores.

4.2.1.3 Algoritmos baseados no estado do Cliente.

Dois tipos de informação podem ser obtidos do Cliente: localização geográfica e a carga típica de um determinado domínio com base na média das requisições realizadas por este domínio em um determinado tempo.

O *Cisco Distributed Director* [17] em um de seus modos considera a localização do cliente/domínio para distribuir. Neste modo age como um DNS primário, considerando a relação de proximidade entre o cliente e o servidor, e, conseqüentemente, uma menor latência na rede.

4.2.1.4 Algoritmos baseados no estado do Cliente e do Servidor.

Uma estimativa de carga de um determinado domínio não permite prever a situação de carga dos servidores, que disparando um alarme de sobrecarga, permite que o distribuidor de DNS exclua o servidor da distribuição durante este período. Um avançado sistema de balanceamento utiliza vários tipos de informação dos servidores que lhes permite um controle eficiente do estado dos mesmos, como por exemplo, o *Cisco Distributed Director*.

4.2.2 Algoritmos com TTL variados dinamicamente.

Inicialmente os algoritmos de controle variados estabeleciam a um ajuste mais eficiente na distribuição DNS através de uma redução no TTL para poder obter um controle supostamente mais efetivo da sobrecarga dos servidores, contudo este método não se mostrou tão eficiente na distribuição de cargas aos clientes de domínios populares e com o uso de servidores de desempenhos diferentes. A razão deste acontecimento é que o fator de peso de um domínio cresce com o valor do TTL e com o correto ajuste do TTL irá permitir o controle subsequente das requisições de cargas previstas de um domínio, então esta situação é a principal causa de sobrecargas dos servidores [13].

Os algoritmos dinâmicos realizam o processo de decisão em duas etapas. A primeira seleciona o servidor com base na carga estimada pela variação média do tempo entre requisições, e na segunda etapa, o DNS arbitrará o valor apropriado do TTL para o próximo período. Para o caso de domínios populares o TTL é inversamente proporcional à taxa de requisição do domínio, ou ajustá-los conforme o desempenho dos servidores, atribuindo valores mais altos a servidores de maior desempenho.

4.3 Distribuidor Concentrado (Dispatcher).

O distribuidor de requisições, ao contrario do DNS, possui total controle das requisições dos clientes e disfarça para o cliente como se fosse o servidor *internet*, de fato o endereço IP do *dispatcher* é o único endereço conhecido dos clientes de um determinado URL.

Desta forma o *Dispatcher* age como um concentrador e distribuidor de requisições pela família de servidores através de algoritmos simples (Round-robin, carga de servidores) ou complexos (SITA-V algoritmos).

Existem duas formas distintas de implementação da distribuição, cuja requisição ao ser atendida pelo servidor *internet* retorna diretamente ao cliente e a outra forma retorne ao *dispatcher* para ele retornar ao cliente.

4.3.1 *Dispatcher* em um único caminho.

Neste caso como mostra a figura 7, o *Dispatcher* direciona a requisição do cliente para o servidor, alterando a requisição do cliente, o endereço de destino com o IP (e IP checksums) do servidor.

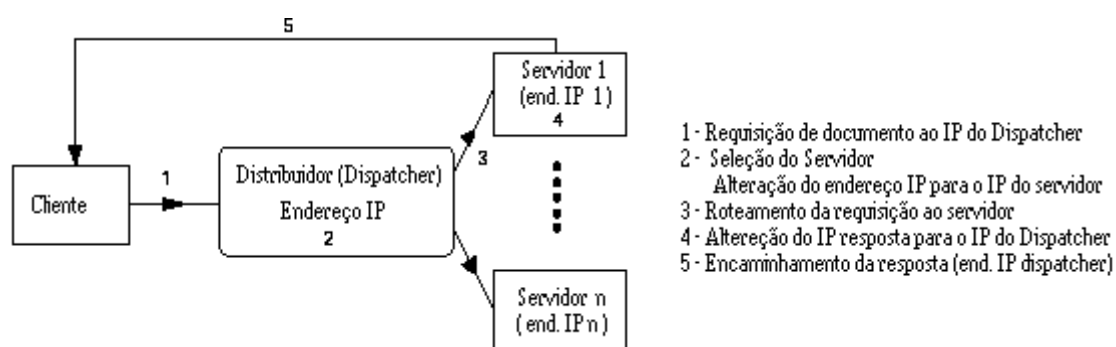


FIGURA 7 - DISPATCHER EM CAMINHO ÚNICO

Este procedimento é executado em todas as requisições HTTP do cliente, pois o Distribuidor é o único endereço conhecido pela rede e pelo cliente do servidor destino,

assumindo, assim, como concentrador e redirecionador de todas as requisições enviadas dos clientes ao servidor.

Para manter o estado de conexão do cliente com o servidor, torna-se necessário manter um registro de todos os assinalamentos realizados entre cliente/servidor, permitindo, assim, que as novas requisições de um determinado cliente sejam enviadas para o mesmo servidor das requisições anteriores.

Como o servidor irá retornar a solicitação diretamente para o cliente, isto é, sem passar pelo *dispatcher*, o servidor deverá alterar o cabeçalho IP com o endereço IP do *dispatcher*, assim sendo, o cliente não tem conhecimento do endereço do servidor.

Esta é uma solução totalmente transparente para o cliente e sem nenhuma necessidade de alteração profunda nos programas dos servidores, visto que o procedimento de substituição do endereço IP ocorre somente na camada do protocolo TCP/IP.

A arquitetura também permite que seja estabelecido um alto nível de disponibilidade do sistema, pois em caso de alguma falha em qualquer servidor, o endereço poderá ser removido da tabela de roteamento, sendo o *dispatcher* o ponto vital da rede, por ser um ponto único de acesso aos servidores, e seu endereço IP cadastrado em todos os servidores, sendo o mesmo o ponto de risco da estabilidade.

4.3.2 *Dispatcher* em mão dupla.

Neste caso como mostra a figura 8, o *Dispatcher* direciona a requisição do cliente para o servidor, escrevendo na requisição o endereço IP do servidor, e o endereço IP do cliente é substituído pelo endereço IP do *dispatcher*, que somente no retorno da requisição do servidor, será alterado novamente o endereçamento IP, agora com o endereço IP do Cliente.

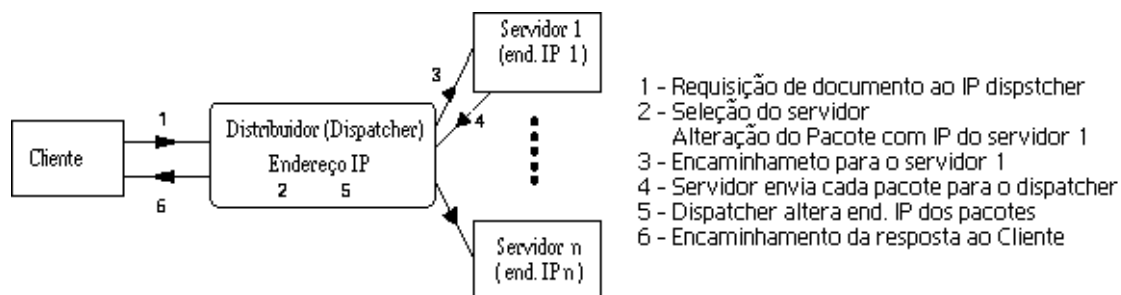


FIGURA 8 – DISPATCHER EM MÃO DUPLA.

Esta metodologia de alterar o pacote IP duas vezes, na ida e volta do servidor, é a base do *Network Address Translation* (NAT) definido em [18].

A Arquitetura *Magicrouter*[19] executada na camada de aplicação, intercepta e altera o endereçamento e o *checksum* dos pacotes nos dois sentidos, e possibilita o balanceamento através de três opções de algoritmos para a escolha do servidor : *round-robin*, randômico e carga proporcional (seleciona o servidor com a menor carga corrente estimada).

Outra arquitetura é o *LocalDirector* [17] da Cisco, executada na camada de rede IP alterando as informações do cabeçalho dos pacotes e proporciona o balanceamento do servidor através de um algoritmo baseado no menor número de conexão ativas.

Esta solução de alteração dos pacotes nos dois sentidos tem duas grandes desvantagens: não funciona com protocolos que manipulam o endereçamento IP dentro da aplicação [18] e requer o próprio *dispatcher* altere o pacote nos dois sentidos, sendo que no sentido do servidor para o cliente existe muito mais tráfego.

4.3.3 Dispatcher de pacotes pelo endereço físico.

O *Dispatcher* pode encaminhar os pacotes dos clientes ao servidor selecionado, usando o endereçamento físico do servidor quando eles estão em uma mesma rede local, não modificando o cabeçalho IP.

O *IBM Network Dispatcher* [20] implementa esta solução, escolhendo o servidor de destino através de algoritmos baseados de forma dinâmica na carga e disponibilidade

dos servidores. A implementação desta solução também foi estendida a redes não locais, procedendo neste caso a alteração do cabeçalho IP para o servidor remoto, o qual altera o endereçamento IP ao endereço inicial e encaminha pelo endereço físico ao servidor local.

A implementação *ONE-IP Address* [21] utiliza a opção do UNIX *ifconfig alias* para configurar os servidores com o mesmo IP secundário do *dispatcher*, fundamentada nesta arquitetura duas técnicas [21] : *routing-based* e *broadcast-based dispatcher*.

As duas técnicas apresentadas utilizam algoritmos estáticos que não consideram a carga e a heterogeneidade dos servidores, além de poder gerar uma sobrecarga no tráfego da rede, pois os servidores devem estar em um mesmo segmento de rede. Na técnica *routing-based* utiliza uma função de busca sobre o endereço IP do cliente enviando o pacote ao servidor. Na técnica *broadcast-based* o pacote do cliente é encaminhado por um broadcast aos servidores, e cada servidor avalia através de uma função de busca sobre o endereço IP do cliente verificando se o resultado identifica que o pacote deve ser tratado por ele.

4.4 Redirecionamento HTTP.

O protocolo HTTP permite ao distribuidor encaminhar as requisições de acordo com o conteúdo existente no cabeçalho e destinar ao servidor cujo documento deve ser acessado, desta forma a distribuição é realizada em função do conteúdo da requisição enviada pelo cliente que assim a informa em função da resposta recebida anteriormente pelo servidor web, o que pode proporcionar uma distribuição fundamentada na localização do cliente e na situação do conteúdo requisitado.

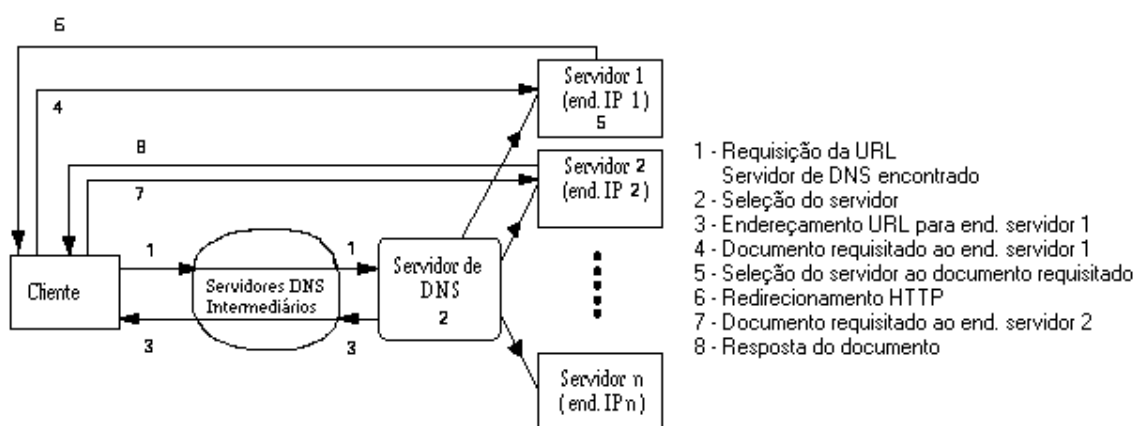


FIGURA 9 – REDIRECIONAMENTO HTTP

Esta técnica, mostrada na figura 9, é executada na camada de aplicação por um *switch* nível – 7 apresenta um tempo de avaliação bem superior e com algoritmos mais complexos, pois a decisão é tomada fundamentada no conteúdo da mensagem, o que pode ser mais um ponto de sobrecarga de tráfego a ser considerado.

4.5 Distribuição com base no Servidor.

A arquitetura de distribuição com base no servidor realiza o encaminhamento das mensagens com base em dois níveis, primeiro as requisições são distribuídas pelo serviço de DNS para os servidores, então em um segundo nível o servidor assinalado poderá ou não redistribuir a requisição para um outro servidor. Esta combinação complementa a técnica de distribuição por DNS, tais como a distribuição não uniforme em um determinado domínio e uma limitação de requisições a um servidor. Os servidores redirecionam a requisição sobrescrevendo o endereçamento ou as informações de cabeçalho do protocolo HTTP.

A técnica SWEB [22] usa o re-direcionamento de mensagens do protocolo HTTP pelo servidor para outro servidor em busca de minimizar o tempo de resposta. Este re-direcionamento pode causar um acréscimo de tráfego entre os servidores, inclusive porque estes necessitam manter os demais informados do seu percentual de utilização.

A técnica DPR [23] redireciona os pacotes alterando o IP das requisições, aplicando uma função de busca sobre o endereço e a porta do serviço IP do cliente e, então, também é considerada a carga do servidor para o encaminhamento da requisição.

Esta arquitetura de distribuição com base no servidor garante uma escalabilidade do sistema, contudo apresenta um crescimento no tempo de latência das mensagens.

4.6 Distribuição com base no Cliente

A arquitetura de distribuição com base no cliente realiza o encaminhamento das mensagens com base no mecanismo de espalhar a requisição entre vários servidores, redirecionando a requisição inicial do cliente. Este método, o próprio navegador do cliente realiza a redistribuição, realizando uma nova requisição ao *site* redirecionado, como por exemplo, a solicitação ao endereço `www.site.com.br`, o *site* encaminha ao navegador o novo endereço do *site* e então o navegador redireciona para receber a página do novo endereço, o endereço `wwwi.site.com.br`, onde *i* normalmente é um número que varia conforme o número de servidores de distribuição.

Esta arquitetura não permite escalabilidade, e depende do histórico armazenado do navegador para ser alterada sua distribuição.

Uma outra solução pode ser realizada com a utilização de clientes inteligentes, através de aplicações na linguagem Java, carregadas no cliente com a finalidade de obter informações do estado dos servidores dos *sites* e das respectivas redes, permitindo informar ao navegador a seleção do servidor apropriado.

Esta solução aumenta processamento no navegador do cliente, assim como o tráfego da rede pelo monitoramento dos servidores e retardos de rede, mas permite a escalabilidade e disponibilidade dos serviços.

5. Método de Distribuição na Arquitetura *Dispatcher*

5.1 Introdução.

A concentração das requisições em um único ponto (URL) para posterior distribuição a um servidor do conjunto de servidores *internet (Web Cluster)* de uma rede necessita que seja estabelecido um método que proporcione a distribuição das requisições aos servidores. Esta política pode ser implementada de diversas maneiras e com diversas finalidades, tendo em vista os diversos protocolos e tipos de dados manipulados pela rede *internet*.

A arquitetura de balanceamento pode ser dividida em duas etapas distintas: a entidade (Cliente, DNS, *Dispatcher*, Servidor) que executa a tarefa de balanceamento, e o algoritmo que toma a decisão de distribuição das requisições HTTP entre os servidores.

5.2. Arquitetura *Dispatcher*.

Neste trabalho evidencia-se a arquitetura *dispatcher*. O distribuidor (*web switch*) é o elemento chave de qualquer *Web Cluster*, responsável por receber todo o tráfego entrante ao site e distribuir aos servidores.

Nos últimos anos, diversos trabalhos acadêmicos e produtos de mercado são com base nesta arquitetura: NCSA[15], Cisco Local Director [17], NAT[18], Magirouter[19], IBM Network Dispatcher [20], ONE-IP[21], Linux Virtual Server Project [24], LARD[25], HydraWeb [26], ShockAbsorber[27], HACC[28], CAP [29].

As empresas Alteon Web Systems[30], Cisco[31], F5 Labs[32], Radware[33] e IBM [34] tem se destacado no mercado comercial com produtos classificados de acordo com a camada do protocolo OSI: nível 4 ou camada 4 ; e nível 7 ou camada 7.

Os *web switches* nível 4 (WSn4) realizam o procedimento de distribuição no momento da conexão do protocolo TCP/IP, camada 4 modelo OSI. Os pacotes pertencentes à mesma conexão TCP devem ser enviados ao mesmo servidor, procedimento que se realiza mantendo uma tabela de associação de cada cliente com o respectivo servidor.

O WSn4 examina o cabeçalho de cada requisição e com base no campo *flag* determina quando o pacote pertence a uma nova conexão ou a uma já existente.

Os WSn4 são compostos de algoritmos que processaram a distribuição sem nenhum conhecimento sobre o tipo de requisição que será posteriormente solicitada pelo cliente, portanto é realizada uma distribuição cega (*content information blind*). Este método permite maior rapidez no processo, mas em contrapartida a possibilidade de um equilíbrio de carga de transação entre os servidores, não é possível de ser realizado. Também é um ponto negativo, a impossibilidade de dispor de servidores com conteúdos e finalidades diferentes.

Os *web switch* nível 7 (WSn7) podem estabelecer uma conexão completa TCP com o cliente, e somente ao receber uma requisição HTTP, será examinado o cabeçalho e com as informações analisadas será realizado o processo de distribuição aos servidores. Esta forma de distribuição tem um retardado obrigatório (*delayed binding*) no WSn7 antes de chegar ao servidor e a sua distribuição pode estar fundamentada no serviço solicitado, conteúdo das URL da requisição HTTP, identificadores segurança

(SSL) e informações de navegação (*cookies*), permitindo um melhor equilíbrio de carga, assim como de servidores para serviços específicos.

Um superficial comparativo entre a vantagem do SWn7 sobre o Swn4 é a distribuição a servidores heterogêneos, a alta taxa de utilização de *cache* e as desvantagens são o alto processamento realizado no *switch*, que pode vir a ser o gargalo do sistema, e a complexidade de configuração do ambiente de distribuição [29].

Os *switchs* podem ser classificados quanto à forma de distribuição em:

- Algoritmos estáticos ou dinâmicos.
- Centralizados ou distribuídos.

Os algoritmos estáticos (como: randômico, *round-robin*) são implementados com o procedimento de distribuição imutável, não permitindo a alteração de comportamento no modo de distribuição.

Os algoritmos dinâmicos são distribuídos conforme a análise das informações base do algoritmo de distribuição, que podem ser do cliente (IP, porta TCP), do servidor (número conexões ativas, servidor de menor carga) ou de uma combinação cliente/servidor.

Os *web switch* precisam ter como principal diretriz o uso de algoritmos simples, ágeis e de pouco processamento, devido à necessidade de tomar decisões rápidas de centenas de requisições por segundo, e sobre este ponto de vista, os algoritmos estáticos têm grande vantagem em relação aos algoritmos dinâmicos, da mesma forma os SWn4 sobre os SWn7.

Os *switchs* centralizados executam a distribuição a uma LAN e os distribuídos a uma WAN. Este trabalho evidencia exclusivamente a estrutura centralizada através de algoritmos estáticos.

Na arquitetura com base em família de servidores, (*cluster-based web server*) pode-se utilizar a política de distribuição com base no algoritmo *round-robin* [12], devido sua simplicidade de implementação. Este método possui as seguintes variações:

Simple rodízio - *roun-round* (RR)

Servidor de menor carga - *least load server round-robin* (LL)

Servidor de menor número de conexões – *least conection round-robin* (LC)

O algoritmo estático RR executa a distribuição por diferentes servidores através de uma lista circular, obedecendo à seqüencialidade de cada item da lista.

O algoritmo dinâmico LL, o distribuidor define a próxima requisição para o servidor que possuir a menor carga (a carga de um servidor é definida como a soma dos tempos de todas as requisições pendentes no servidor). Para realizar esta distribuição é necessário que o algoritmo receba a informação de carga de cada servidor, o que requer uma aplicação de monitoramento e informação em execução nos servidores. Devido ao tráfego entre servidores e distribuidor, esta arquitetura gera, também, um acréscimo ao tráfego da rede interna.

O algoritmo dinâmico LC encaminha a próxima requisição ao servidor que tiver o menor número de conexões HTTP ativas, necessitando desta forma manter uma tabela de número de conexões estabelecidas por servidor em memória do *dispatcher*.

6. Modelo Proposto.

6.1 Introdução.

Propõe-se o algoritmo Rajada Controlada (*Burst Control* - BC), que atua em nível de camada 4 da OSI, durante o processo estabelecimento de conexão, estabelecendo um processo de distribuição sequencial, sendo que cada servidor *internet* recebe “*n*” requisições seguidas (rajada) desde que, neste determinado instante, as distribuições realizadas ao servidor *internet* não ultrapasse o limite máximo de requisições que o servidor pode receber a cada segundo (controle de distribuição).

O principal objetivo deste modelo é possibilitar a limitação de carga máxima dos servidores *internet*, permitindo que os mesmos operem dentro de limites operacionais estáveis e com a possibilidade de dispor de um tempo de resposta máximo garantido ao usuário.

Como objetivo secundário, a distribuição em rajada possibilita o aproveitamento das rotinas operacionais de estabelecimento de conexão serem instanciadas e permanecerem em memória durante o tratamento de solicitações de conexões seguidas, permitido um ganho no processo operacional do mesmo.

Considerando-se que a demanda de tráfego apresenta um comportamento sazonal, com oscilações significativas durante o período de atividade dos *site* como visto no capítulo 1º, o estabelecimento de um processo de inteligência ao método, permitirá a variação de parâmetros ao algoritmo, procedimentos ou até algoritmos diferenciados conforme a demanda solicitada, dependendo do nível de abrangência desejado. O método estabelece a existência de uma camada de gerência ao distribuidor que lhe permita alterar parâmetros que influencie no procedimento do algoritmo, tais como os servidores ativos, valor de “*n*” da rajada, valor de desempenho dos servidores.

A principal característica do modelo de gerência é possibilitar a alteração dinâmica do comportamento de distribuição, atuando através de gatilho, estabelecido pela fila do número de requisições entrantes no distribuidor, operando com dois níveis de gatilhos, um máximo e outro mínimo, conforme a carga, sendo que os gatilhos não

são de valores fixos, os valores são gerenciados pelo próprio gerente, estabelecendo o próximo momento que atuará sobre o sistema de distribuição.

O gerente também poderá ser chamado através de uma seqüência de conexão específica, ou através de uma porta de configuração, porta esta que permitirá atuar no módulo de gerência mesmo nos momentos de alta demanda de carga entrante.

6.2 Método de Distribuição - BC.

O modelo de distribuição em rajada controlada (BC), conforme a figura 10, trabalha com algumas informações associadas às requisições dos clientes obtidas na camada 4 do OSI (nível 4).

A idéia básica é a distribuição limitada do número de requisições por segundo encaminhadas ao servidor, possibilitando a família de servidores ser constituída de equipamentos heterogêneos.

As informações e os controles realizados são obtidos tanto no processo de estabelecimento de conexão (fase da distribuição) quanto nas requisições de informações trocadas posteriormente.

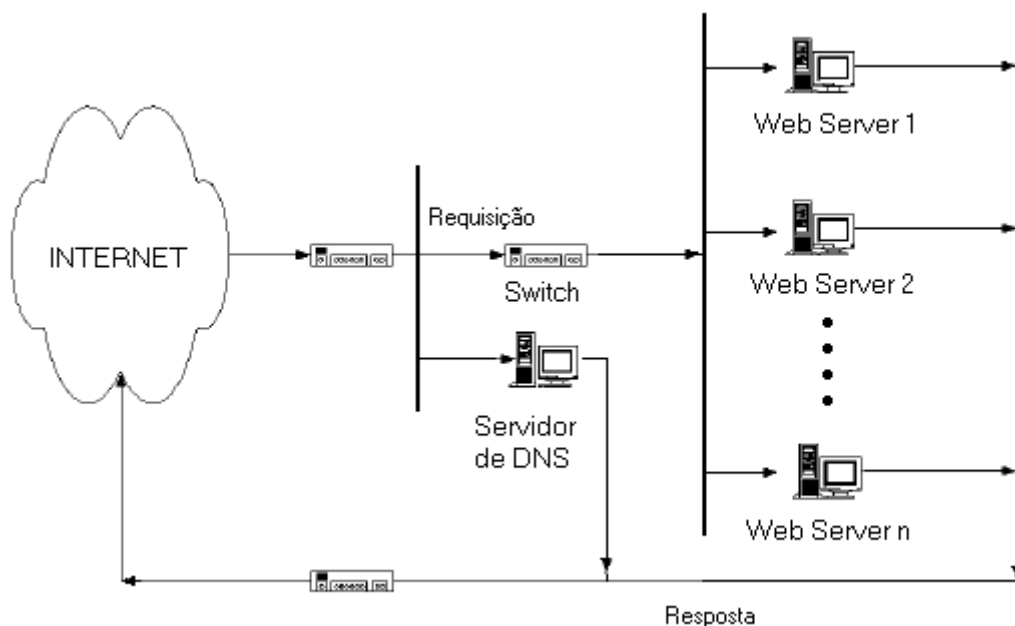


FIGURA 10 – MODELO DISPATCHER BC

6.3 Característica Principal.

O processo de distribuição realiza-se através de uma lista circular de endereços de servidores, e com base no conteúdo das mensagens do protocolo TCP proporciona a escolha do servidor ao qual a mesma será enviada.

A base de informações é tratada a dois níveis de algoritmos, o básico responsável pela distribuição (AD) e o gerencial (AG) responsável pela coleta e gerência da distribuição.

6.3.1 Algoritmo de Gerência.

O algoritmo de gerência realiza o procedimento de coleta de informações, determina a lista de servidores e implementa a política de distribuição.

É responsável pela análise das requisições enviadas pelos clientes para a coleta informações necessárias a escolha do servidor. As informações básicas tipo o serviço de porta TCP, tamanho da janela, rede de origem, número de requisições por servidor, e número de conexões ativas por servidor possibilitam uma avaliação estimada de carga.

A quantidade de tipo de informação obtida neste processo é decorrente das necessidades de distribuição específica de cada tipo de *site*, podendo este módulo gerencial alterar a política de distribuição dinamicamente conforme as características do *site*, realizando uma maior coleta de informações nos momentos menos críticos, e executando o mínimo de coleta com os dados mais críticos ao processo durante os instantes de alta demanda de tráfego.

Observa-se que o método pode implementar com mais recursos e controle nos *switch* de nível 7, pois a vicissitude da análise das requisições e resultados na qualidade da distribuição serão muito mais precisas.

Cabe a este módulo proceder a supervisão e gerenciamento dos parâmetros de distribuição, podendo até, ter a função de ligar e desligar servidores, conforme a necessidade específica do *site*.

6.3.2 Algoritmo Distribuição.

O algoritmo básico realiza o procedimento de distribuição com base na lista de distribuição que o algoritmo de gerência disponibiliza, controla número de requisições que podem ser enviadas ao servidor por segundo e controla a rajada de conexão TCP que podem ser enviado ao mesmo servidor que é responsável pelo procedimento de distribuição das solicitações de conexão e controle das requisições.

Os pedidos de conexões são atendidos através de lista circular de distribuição ordenada pelo algoritmo de gerência e encaminhados em forma de rajada aos servidores, isto é, cada servidor recebe “n” solicitações de conexões seguidas, para que possa ser escolhido outro servidor para atender as requisições de conexões. O número “n” é definido para cada servidor, e pode ser no mínimo um.

O encaminhamento das requisições é realizado pela relação cliente x servidor estabelecida na conexão. Ao receber um pedido de finalização de conexão, esta relação será desfeita e a cada novo pedido de conexão, a nova situação de relação será estabelecida, possibilitando, assim, que o algoritmo de gerência realize o tratamento deste novo estado, em cada momento de tempo e estado dos servidores para elaborar a relação da tabela de distribuição. Para obter um melhor desempenho na gerencia desta fila de relações, a manutenção da fila será realizada ciclicamente ao processo, exceto quando a fila atingir o limite máximo de elementos na fila. Ao manter esta relação durante um determinado tempo, a distribuição de um novo pedido de conexão deste mesmo cliente será estabelecida pela tabela de relação se existir, evitando, assim, todo o procedimento de avaliação e inserção de elemento na lista de relacionamentos, senão será estabelecido um novo relacionamento: a lista de servidores de distribuição.

Todas as requisições, de conexão ou não, deverão ser submetidas ao controle de limitação de desempenho que cada servidor pode receber, esta limitação é contabilizada no algoritmo pela quantidade de mensagens por segundo que podem ser enviadas a cada servidor. Para cada segundo será mantido o número de requisições enviadas e encaminhamento da requisição deverá ser submetido ao controle desta limitação, se neste exato momento o número de requisições enviadas no último segundo, não ultrapassar o limite estabelecido ao servidor, a requisição será encaminhada, senão este

controle poderá submeter á requisição a um retardo de $1/60$ segundos e posicionar novamente a fila de entrada, para novo procedimento de distribuição.

No caso de pedido de uma nova conexão, não pertencente à lista de relacionamento cliente x servidor, será escolhido o próximo servidor da fila de distribuição. Se o servidor escolhido estiver com o número máximo de mensagens saturado, é automaticamente escolhido outro servidor, e somente se nenhum servidor da fila estiver apto a receber este pedido de conexão, isto é, todos estão saturados, então a mensagem será submetida a um retardo de $1/60$ segundos, e encaminhada para nova distribuição.

O crescimento da fila de requisições submetidas ao retardo de $1/60$ segundos é o indicativo que o distribuidor está entrando em forma de saturação, assim como os servidores estão no limite máximo de processamento especificado. Este limite de requisições em fila de retardo é o gatilho de rejeição de novas requisições de conexões, realizadas também pelo tempo de $1/60$ segundos, assim toda vez que a fila atingir o limite máximo de elementos da fila de retardo, o algoritmo recusará todas as solicitações de novas conexões no período de $1/60$ segundos.

6.4 Restrições ao Modelo.

Para permitir o dimensionamento de memória e desempenho do *switch*, é necessário estabelecer ao algoritmo métodos de controle que lhe garantam o desempenho necessário para o objetivo desejado.

Primeiramente, o tempo de processamento realizado pelos algoritmos que implementados diretamente em componentes físicos do equipamento, pode ser dimensionado em função do tamanho máximo das filas de armazenagem e do método de recuperação de tabelas, dados e características físicas. O número máximo de elementos na fila e tabela, são especificados através da capacidade máxima de memória definida ao equipamento.

É conclusivo que quanto maior forem as filas e tabelas, os métodos de acesso e manipulação dos dados requerem um tempo maior de processamento, portanto limitando o número máximo de distribuições realizadas em virtude da limitação tecnológica imposta. O método de distribuição proposto permite então que seja especificado com grande margem de segurança o limite de tráfego máximo a ser distribuído.

Para garantir a limitação das filas, os seguintes controles são necessários:

- Números de requisições de conexão recebidas a cada 1/60 de segundos.
- Número de requisições no *switch* aguardando a disponibilidade do servidor. (atender ao limite máximo do número total de requisições em 60 segundos).

O controle nas requisições de conexões permite estabelecer dois controles de filas internas. O primeiro e mais importante controle ocorre durante o processo de distribuição aos servidores *web*, que o recurso de processamento é maior sendo então o ponto de impacto a saturação do sistema de encaminhamento. O sinalizador de atuação ao início do procedimento de controle de entradas de conexões é justamente a outra fila crítica do sistema que, ao ultrapassar o limite de segurança estabelecido, gera durante 1/60 segundos um alerta, e todas as requisições recebidas durante este alerta são ignoradas para o processamento pelo *switch*.

O controle de disponibilidade dos servidores é realizado por um procedimento de retardo na requisição recebida, tanto de conexão, finalização e requisições intermediárias entre a conexão e finalização. O retardo é solicitado sempre que no

momento de envio da requisição ao servidor, o mesmo estiver com a sua carga máxima de requisições enviadas nos últimos 60 segundos. O retardo mínimo submetido é $1/60$ segundos e o máximo depende do limite máximo definido para a fila de retardo no projeto, pelo contexto de memória e processamento.

7. Análise de Desempenho.

7.1 Introdução.

A análise de desempenho tem por objetivo verificar o desempenho do algoritmo proposto na distribuição de requisições de clientes a um *cluster* de servidores *Web*, cujos servidores possuem o mesmo conteúdo e são provedores de páginas estáticas.

A caracterização da carga atende o perfil de duas situações distintas: a primeira, a taxa de requisições é inferior a 85% da capacidade de processamento do sistema. No outro teste, utiliza-se a taxa de requisições superior a 90% da capacidade de processamento do sistema.

A análise propõe a comparação do desempenho entre o algoritmo do modelo proposto e o algoritmo *Round Robin*.

7.2 Modelo de Desempenho.

O Servidor de DNS primário informa na conversão do endereço do *site* para o respectivo endereço IP, o endereço Ip do *switch*. Os endereços dos servidores *web* são privados e invisíveis aos clientes. O *switch* recebe todo o tráfego de requisições dos clientes.

Na figura 11 apresentam-se os componentes utilizados na modelagem.

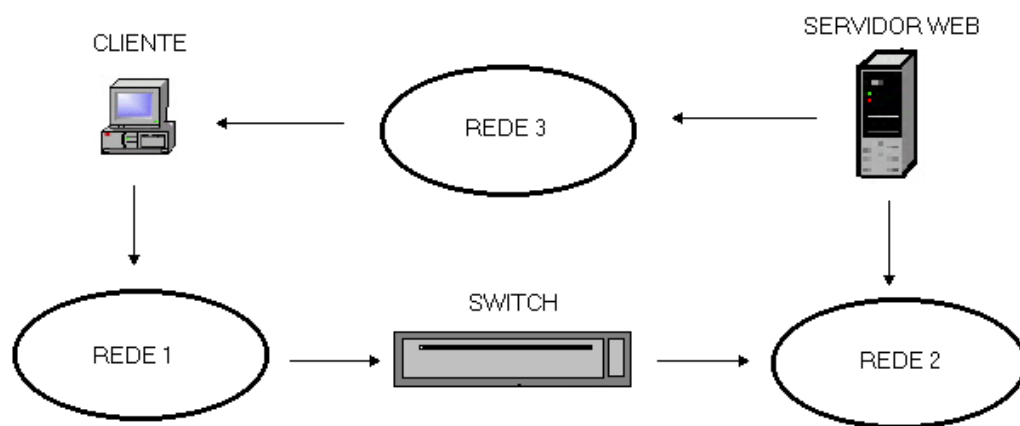


FIGURA 11 - DIAGRAMA DO MODELO SIMULADO

7.3 Modelo Lógico.

O modelo lógico apresentado na figura 12, com os respectivos componentes do modelo: cliente, rede 1, rede 2, *switch*, *web server* (n) e rede 3.

Para o modelo considera-se o seguinte:

O cliente realiza requisições completas, isto é, estabelece a conexão, realiza aleatoriamente “n” requisições por sessão e finaliza a sessão.

Não foram considerados os tempos de processamento no computador do cliente e o tempo de pensamento do operador entre uma solicitação e outra, devido na análise do modelo do tráfego entrante no distribuidor não importar a respectiva origem, assim sendo, estabeleceu-se um tempo zero de processamento no cliente.

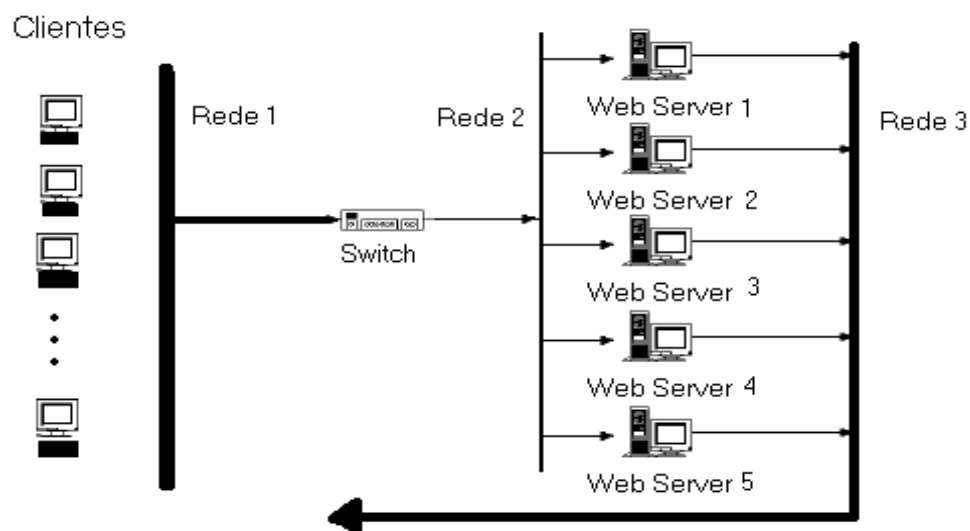


FIGURA 12 - MODELO LÓGICO SIMULADO

7.4 O Modelo Físico.

O modelo físico é basicamente um modelo de filas e processos cujo diagrama está representado na figura 13.

Para a modelagem do algoritmo utiliza-se a técnica de solução de um problema pela análise de um modelo que descreve o comportamento do algoritmo, usando um computador digital, isto é, a simulação através do programa ARENA de uso específico para simulações de sistemas em computadores digitais.

O ARENA é um produto da empresa americana *Systems Modeling Corporation* e comercializado desde 1993.

O processo de modelagem contemplou dois algoritmos:

- a) Modelo - Rajada Controlada (BC)
- b) Modelo - Round-Robin (RR).

A nível deste trabalho, a modelagem não contemplou o módulo de gerência para o modelo BC.

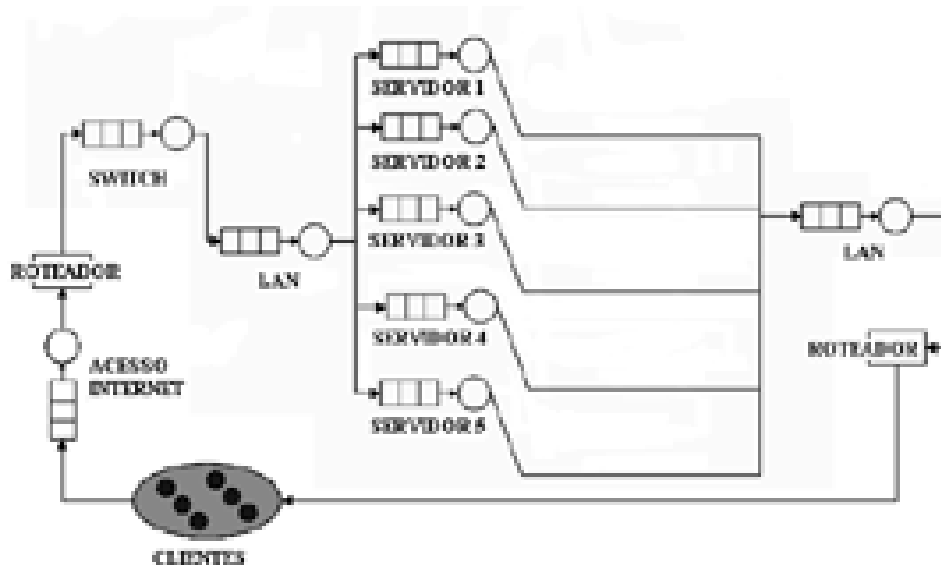


FIGURA 13 - MODELO FÍSICO SIMULADO

O número de requisições entrantes de tráfego gerado é determinado através do modelo de uma função exponencial, que tem um comportamento similar ao ambiente *internet* com picos aleatórios de rajadas de tráfego.

A rede é dividida em três segmentos: Rede 1; Rede 2 e Rede 3. Para os objetivos da simulação, o tempo despendido no tráfego da informação na rede não interfere significativamente no resultado, no sentido do tráfego entrante de requisições ao *switch*, Já para o tráfego de retorno ao cliente, processo de tráfego originado do ciclo de requisições enviadas ao cliente, necessários para completar a informação da página *web* no cliente, este tempo foi modelado segundo uma distribuição função Triangular, com base nos tempos e características do modelo de rede utilizado por Cardellini [35], contemplando na função triangular as bandas de tráfego pequena, média e larga.

O modelo não contempla a variação do tamanho de bytes das solicitações de requisições. O algoritmo comporta o modelo de servidores heterogêneos, mas no modelo simulado considera-se que os servidores possuem o mesmo comportamento. Seus tempos de serviço foram também modelados na distribuição Triangular. Seus parâmetros englobam os tempos de todo o conjunto de recursos utilizados no processamento das requisições. A função triangular oblíqua representa adequadamente o modelo de processamento para servidores *web* que fornecem páginas estáticas de informação, tais como servidores de busca de conteúdo, servidores institucionais e portais *internet*, que representam um número considerável de *sites internet*.

O valor estabelecido para o desempenho estabelecido para cada servidor é de 180 requisições por segundo. Todos os servidores têm o mesmo desempenho e o mesmo conteúdo.

O tempo de processamento do algoritmo foi definido no modelo conforme a tabela 1.

Procedimento	Modelo BC	Modelo RR
Conexão	0.005	0.003
Requisições	0.001	0.0005
Finalização	0.001	0.0005

TABELA 1– TEMPO DO PROCESSO DOS ALGORITMOS DE DISTRIBUIÇÃO

A simplificação proposta nos modelos dos componentes não invalida os resultados obtidos e a principal funcionalidade testada no modelo é a distribuição controlada de requisições ativas processadas.

7.5 Algoritmo BC .

A descrição do algoritmo implementado no ARENA do modelo BC.

Algoritmo :

constante $s \leftarrow 1/60$ seg

Cte.Número.servidores = n° de servidores ativos em distribuição

Limite.contador.rajada.servidor = número máximo de requisições tipo SYN pode

Receber de modo seguido.

Servidor.rajada $\leftarrow 1$ (inicia com o primeiro servidor)

faça enquanto for (verdadeiro)

 receber e analisar a próxima requisição r

Se Fila no procedimento de aguardo de servidor (tempo de $1/60$) > 25

 E é uma requisição de conexão

Então requisição \leftarrow rejeita requisição.

 Se Flag = FIN

 Então

$n \leftarrow$ servidor destino (IP cliente)

 Número.Conexão.Ativas (n) \leftarrow Número.Conexão.Ativas (n) - 1

 Número.hits.ativos (n, s) \leftarrow Número.hits.ativos (n, s) + 1

 Se Número.hits.ativos (n, s) = $<$ Limite.hits.ativos (n)

 Então

 Envia request

 Senão

 Número.hits.ativos (n, s) \leftarrow Número.hits.ativos (n, s) - 1

 Processo.assíncrono.retardo.tempo (s) \rightarrow retorno em início

 Fim de se

 Fim de Se

Se Flag \diamond SYN ou FIM

 Então

$n \leftarrow$ servidor destino (IP cliente)

 Número.hits.ativos (n, s) \leftarrow Número.hits.ativos (n, s) + 1

 Se Número.hits.ativos (n, s) = $<$ Limite.hits.ativos (n)


```

        Então
    Envia request
        Senão
            Número.hits.ativos (n, s) ← Número.hits.ativos (n,s) - 1
            Processo.assíncrono.retardo.tempo ( s) → retorno em início
        Fim de se
    Fim de se
    Se Flag = SYN
        Então
        Rotulo - Inicio
            Conta.rajada ← conta.rajada + 1
    Se Conta.rajada = < limite.número.rajada.Servidor (servidor,rajada)
    então
        Conta.rajada ← conta,rajada + 1
        Senão
            Vá para o rótulo Meio
    Fim de se
    n ← servidor.rajada (conta.rajada)
    Número.hits.ativos (n, s) ← Número.hits.ativos (n,s) + 1
    Se Número.hits.ativos (n, s) = < Limite.hits.ativos (n)
        Então
            Número.Conexão.Ativas ( n) ← Número.Conexão.Ativas (n) – 1
            Número.hits.ativos (n, s) ← Número.hits.ativos (n,s) + 1
            Encaminha requisição os servidor n
    Inicia nova requisição
        Senão
            Número.hits.ativos (n, s) ← Número.hits.ativos (n,s) - 1
        Fim de se
    Rótulo Meio
        Servidor.rajada ← servidor,rajada + 1
        Conta.rajada ← 0
        Se servidor.rajada > cte.número.servidores

```

```
Então
Servidor.rajada ← 1
Fim de se
Se ip,cliente ( r ) = IP.cliente.anterior
Então
Processo.assíncrono.retardo.tempo ( s ) → retorno em início
Fim de se
Ip.cliente.anterior ← Ip.cliente ( r )
Vá para rótulo início
Fim de se
```

7.6 Algoritmo RR.

O algoritmo implementado no ARENA do modelo RR.

Os acumuladores usados na simulação são usados exclusivamente para a avaliação dos resultados.

Cte.Número.servidores = n° de servidores ativos em distribuição

faça enquanto for (verdadeiro)

receber e analisar a próxima requisição r

Se Flag = FIN

Então

$n \leftarrow$ servidor destino (IP cliente)

Número.Conexão.Ativas (n) \leftarrow Número.Conexão.Ativas (n) – 1

Número.hits.ativos (n, s) \leftarrow Número.hits.ativos (n,s) + 1

Envia request

Fim de Se

Se Flag \diamond SYN ou FIM

Então

$n \leftarrow$ servidor destino (IP cliente)

Número.Conexão.Ativas (n) \leftarrow Número.Conexão.Ativas (n) – 1

Número.hits.ativos (n, s) \leftarrow Número.hits.ativos (n,s) + 1

Envia request

Fim de Se

Se Flag = SYN

Então

Conta \leftarrow conta + 1

Se Conta. \leq Cte.Número.servidores

então

$n \leftarrow$ servidor (conta)

Senão

Conta \leftarrow 1

$n \leftarrow$ servidor (1)

Fim de se

Envia request .

Fim de se

8. Projeto de Experimentos.

8.1 Introdução.

Dentre os diversos tipos de projeto de experimentos, considerou-se neste trabalho o projeto fatorial completo 2^k , que utiliza todas as combinações possíveis de todos os níveis de todos os fatores, conforme tabela 3.

A vantagem deste projeto é que todas as combinações possíveis de configuração e carga são examinadas, possibilitando encontrar o efeito de todos os fatores, incluindo fatores secundários e suas interações. A Tabela 2 apresenta os fatores e níveis empregados nos experimentos realizados.

O processo de modelagem analisou-se os efeitos (fatores individuais e combinados) sobre uma variável de resposta através dos seguintes fatores:

		Max	Min
Fator A	Tráfego (EXPONENCIAL (x))	0.01	0.02
Fator B	Num. Interações com o Cliente (POISON (x))	10	5
Fator C	Rajada de Distribuição por Servidor	4	2
Fator D	Fila de <i>Delay</i> (Disparo de Rejeição)	40	20

TABELA 2 – FATORES E NÍVEIS

Analisados sobre as seguintes variáveis de resposta:

- Tempo de Resposta
- Taxa de Ocupação do Servidor
- Número de Requisições Rejeitadas
- Número de Requisições Atendidas

O estudo de desempenho para o algoritmo BC, Tabela 3, com 4 fatores (A,B,C e D), com cada fator tendo 2 níveis, requerendo 2^4 (16) experimentos.

Simulação	Fator A	Fator B	Fator C	Fator D
01	0.01	5	2	20
02	0.01	5	2	40
03	0.01	5	4	20
04	0.01	5	4	40
05	0.01	10	2	20
06	0.01	10	2	40
07	0.01	10	4	20
08	0.01	10	4	40
09	0.02	5	2	20
10	0.02	5	2	40
11	0.02	5	4	20
12	0.02	5	4	40
13	0.02	10	2	20
14	0.02	10	2	40
15	0.02	10	4	20
16	0.02	10	4	40

TABELA 3 – EXPERIMENTOS PARA O ALGORITMO BC

O estudo de desempenho para o algoritmo RR, Tabela 4, com 2 fatores (A e B), com cada fator tendo dois níveis, requerendo 2^2 (4) experimentos.

Para a análise dos fatores no modelo RR, os fatores C e D não são analisados devido o algoritmo não tratar estes dois fatores.

Simulação	Fator A	Fator B
1	0.01	5
2	0.01	10
3	0.02	5
4	0.02	10

TABELA 4 – EXPERIMENTOS PARA O ALGORITMO RR

8.2 Experimentos Realizados como o Modelo BC.

Para analisar o comportamento do sistema sob estudo, foram eleitas quatro variáveis de resposta: Tempo de Resposta, Carga do Servidor, Requisições Rejeitadas e Requisições Atendidas.

Os efeitos dos fatores (Tabela 3) sobre as variáveis sob estudo estão demonstrados na Tabela 5.

Fator	Tempo de Resposta	Carga no Servidor	Requisições Rejeitadas	Requisições Atendidas
A	40,70%	1,18%	87,72%	17,17%
B	51,79%	2,38%	11,22%	29,98%
C	0,00%	0,52%	0,04%	6,36%
D	0,00%	0,01%	0,01%	0,18%
AB	7,48%	0,71%	0,00%	9,35%
AC	0,01%	1,36%	0,59%	18,95%
AD	0,00%	0,03%	0,10%	0,27%
BC	0,02%	0,86%	0,21%	12,15%
BD	0,00%	0,36%	0,05%	4,78%
CD	0,00%	0,06%	0,05%	0,80%
Erros	0,00%	92,52%	0,00%	0,00%

TABELA 5– VARIÁVEIS DE RESPOSTA DO ALGORITMO BC

8.3 Experimentos Realizados como o Modelo RR.

Para analisar o comportamento do sistema sob estudo, foram eleitas quatro variáveis de resposta: Tempo de Resposta, Carga do Servidor e Requisições Atendidas.

Os efeitos dos fatores (Tabela 4) sobre as variáveis sob estudo estão demonstrados na Tabela 6.

Fator	Tempo de Resposta	Carga no Servidor	Requisições Atendidas
A	40,15539	84,88462	85,28645
B	51,01634	14,76317	14,2452
AB	8,500897	0,09009	0,177467
Erros	0,327375	0,262126	0,290877

TABELA 6 – VARIÁVEIS DE RESPOSTA DO ALGORITMO RR

8.4 Análise Comparativa.

Considerando o tempo de processamento definido no modelo, foi conforme a tabela constante da tabela 7, observa-se que:

Procedimento	Modelo RR	Modelo BC
Conexão	0.005	0.003
Requisições	0.001	0.0005
Finalização	0.001	0.0005

TABELA 7– TEMPO DO PROCESSAMENTO DO SWITCH

O tempo de resposta mostrou-se que no modelo RR é mais rápido que o BC, quando a carga de requisições ao *site* está dentro do projetado, o que era esperado, visto que o método de distribuição RR é mais rápido no procedimento de distribuição, contudo, quando o *site* é saturado de requisições comprometendo a estabilidade de processamento dos servidores, elevando o tempo de resposta para o modelo RR. O modelo BC sofre a sobrecarga de requisições no distribuidor, elevando, também, o tempo de processamento das requisições, mas o processo de rejeição de requisições possibilita estabilizar o tempo de resposta. A rajada utilizada não foi significativamente grande para influenciar no resultado, visto que o modelo do processamento do servidor não possibilitou ajustes de tempo de processamento a reutilização de código compartilhado na memória.

A carga de processamento no servidor mostrou-se muito significativa em função do tráfego de requisições para o modelo RR, o que demonstra que o tráfego atua diretamente no desempenho dos servidores e que o controle de desempenho estabelecido pelo modelo BC foi eficaz mantendo a carga do processamento dos servidores estável em relação ao tráfego.

Para estabelecer o controle de carga em relação ao tráfego, as requisições rejeitadas sinalizaram claramente o custo do controle de tráfego - altas taxas de rejeição para altas taxas de tráfego.

As requisições atendidas – *throughput* do sistema, para o modelo RR tem sua maior representatividade no tráfego, enquanto para o modelo BC está no número de

interações realizadas entre o cliente e o servidor. O modelo ratificou garantia de um tráfego estável dentro da capacidade previamente estabelecida aos servidores. Mostrou também que uma página *web* com muitas interações com o servidor, gera a degradação do tempo de resposta e o *throughput* do sistema.

8. Conclusão e Trabalhos Futuros

O trabalho de pesquisa de distribuição do tráfego *internet* foi realizado através de pesquisa bibliográfica de trabalhos técnicos publicados em congressos e revistas especializadas, disponíveis para consulta na internet até a data de julho de 2001, sobre os métodos de controle de tráfego de distribuidores *internet* e respectivos algoritmos. Não foi encontrado nenhum livro publicado sobre o assunto, assim como a literatura técnica de documentação dos produtos comercializados não documenta de forma detalhada o método de distribuição utilizado nos respectivos produtos.

Verificou-se que na pesquisa bibliográfica realizada que os métodos publicados têm como objetivo a otimização da performance do distribuidor, desconsiderando na análise, a capacidade dos servidores *web* em atender as requisições encaminhadas, possibilitando desta forma o saturamento da capacidade de processamento do servidor.

Outro fato é que os distribuidores não buscam coletar informação sobre o tráfego que continuamente passa pelo mesmo, utilizando-se de mecanismos de coleta de informações nos equipamentos externos, mantendo assim o foco na performance do distribuidor.

Observamos, também, que a eficiência da distribuição é a tônica dos distribuidores com o direcionamento dos produtos para os algoritmos de distribuição para a camada 7 do modelo OSI, utilizando-se de servidores heterogêneos e com serviços específicos.

Com base neste conhecimento, propôs-se desenvolver um método de distribuidor que atuasse com a preocupação da não saturação do processamento dos servidores, isto é, limitação da distribuição à capacidade de atendimento do conjunto de servidores disponíveis para tal, e extrair o máximo possível de informações para nortear o processo de distribuição do conteúdo das mensagens passantes no distribuidor.

No contexto modelado, o fator da rajada não caracterizou uma variável importante nos resultados em relação ao ambiente simulado. No modelo o tempo de processamento de uma requisição no servidor é constante, portanto não contempla um valor diferenciado para quando ocorre uma seqüência de procedimento similar executado no processador, motivo pelo qual a variável rajada não apresentou resultados significativos. A modelagem detalhada do comportamento do servidor permitirá conclusões mais

definitivas sobre as potencialidades da utilização da rajada na otimização da distribuição do Modelo BC.

Assim sendo, o distribuidor concentrou-se no controle do tráfego enviado aos servidores com o intuito de evitar que cada servidor da família não receba mais mensagens do que é capaz de processar a cada unidade de tempo, que foi definido neste trabalho em requisições por minuto.

O controle do limite de requisições processadas por minuto nos servidores foi de encontro ao paradigma de atendimento de requisições, percebeu-se alguns problemas relacionados à instabilidade do tráfego, pois, como poderá ser eficiente se existe a possibilidade de que pedidos de conexões sejam rejeitados no distribuidor (*site*), como garantir um tempo de resposta a um cliente se o servidor pode ter o processador do servidor completamente saturado. Esta situação de instabilidade de tráfego é devida exclusivamente ao ambiente *internet* já que existe uma total independência no controle de usuários quando estes, em um determinado momento acessam a telas de um determinado *site*. Desta forma a rede *internet* sempre possibilita que a expectativa de tráfego pode ultrapassar a capacidade planejada do *site*, tornando os servidores com cargas de processamento superiores ao limite máximo que os processadores suportam, causando resultados inesperados ao comportamento dos mesmos.

Concluiu-se então que o método mostrou ser eficaz na distribuição de requisições *internet* a uma família de servidores, mesmo em situações atípicas, conforme os resultados obtidos na simulação.

Como principais benefícios do método podemos citar:

- Algoritmo de fácil implementação;
- Algoritmo independente de dados externos ao *switch*;
- Garantia de um tempo de resposta máximo ao cliente;
- Garantia de comportamento estável aos servidores.

Pode ser considerado como ponto positivo à rejeição de requisições utilizadas para manter o desempenho dos servidores no valor máximo estabelecido. A rejeição é fundamental para a estabilidade do sistema.

O número de interações entre o cliente e o servidor para a completa apresentação da página *internet* solicitada é o fator que mais influencia na performance do tempo de resposta do processo, assim como na limitação de requisições atendidas pelo servidor,

portanto recomenda-se que para a melhoria do desempenho a construção de páginas com o mínimo possível interações, como exemplo a utilização de muitas figuras, visto que a carga de cada uma delas é realizado individualmente pelo navegador.

A especificação de desempenho dos servidores em requisições por minuto, não deve ser utilizada para altas demandas de tráfego, na simulação realizada o controle efetivo de distribuição foi realizado dentro do limite de 1 segundo, assim para cada segundo o controle de requisições avaliava a capacidade de distribuição, esta unidade de avaliação deve ser estudada em milésimos de segundo, e os servidores terem seus desempenhos medidos em segundos para que melhor caracterize a demanda para grande oscilação no tráfego *internet*.

A rejeição de novas conexões, quando os servidores estão operando no limite máximo determinado, mostrou-se eficaz para o controle do tempo máximo de resposta ao *site*, possibilitando que sejam definidos padrões de qualidade de serviço aos usuários, como exemplo, o tempo máximo de resposta.

Como alternativa para a rejeição e o aproveitamento total das requisições recebidas dos clientes, pode ser implementado um algoritmo adicional específico para o encaminhamento do tráfego rejeitado a um outro *switch*, de ação de escape, durante uma sobrecarga de tráfego. Tal solução tem um encadeamento máximo limitando a memória, o tempo de processamento de busca e operação à mesma a ser suportado pelo primeiro *switch*.

Cabe finalmente ressaltar que esta é uma pesquisa em andamento. Estudos ampliados dos experimentos aqui apresentados estão sendo realizados. Tais estudos levarão a conclusões mais definitivas sobre as potencialidades do Modelo BC.

O trabalho permite pesquisa de trabalhos futuros em :

1 – Extensão do algoritmo proposto para trabalhar na camada de nível 7.

Trabalhar na camada 7, torna-se necessário que o *switch* controle o processo de conexão das requisições dos clientes, e mantenha por sua vez uma ou mais com os servidores. Este ambiente cria a possibilidade de trabalhar com conjuntos de servidores que oferecem serviços específicos resultando o controle de requisições por grupo, e clientes utilizando mais de um grupo em um único processo de conexão.

2 – Metodologias para distribuição em vários *switch*, isto é paralelismo.

Proceder a distribuição desta forma permite uma maior segurança ao provedor de serviços de que sua organização não está dependendo única e exclusivamente de um único equipamento para oferecer os serviços aos cliente, possibilitando a redundância e uma menor possibilidade de falha no serviço oferecido, com total transparência ao cliente.

3 – Extensão para metodologias de *sites* distribuídos.

Para altas demandas, a concentração dos acessos a uma única região causa o congestionamento das rotas e caminhos de redes para aquele determinado ponto. O processo de distribuição geográfica dos serviços de forma transparente envolve além dos distribuidores, a construção de processos de roteamento de tráfego distribuído, que possibilitem configurar rotas dinamicamente.

4 – Métrica de desempenho para servidores *internet*.

A avaliação de capacidade de servidores internet deve dispor de métrica de desempenho que retratem a característica do tráfego e cada tipo de transação característica, já que o ambiente é altamente diversificado, com som, imagens, textos, transações, criptografias, etc.

Referências Bibliográficas

- [1] A. Scaglia, “**Lições de quem enxergou o futuro**”, Information Week Brasil, 20/07/2001. <http://www.informationweek.com.br/decapa/artigo.asp?id=14109>.
- [2] InformationWeek Brasil – ITWEB, ”**Ser rápido já não é suficiente**”, 19-07-2000. <http://www.informationweek.com.br/noticias/artigo.asp?id=5429>
- [3] The Charles Schwab Corporation. <http://www.CharlesSchwab.com>
- [4] Information Week Brasil, “**Universo em expansão**“, 20/07/2001. <http://www.informationweek.com.br/techreport/artigo.asp?id=14181>.
- [5] M. Fabbi, J. Pultz, “**Look out WAN – The Ethernet Roadkill Machine Is Coming**”, Gartner Group Publication, COM-12-9201, 06/02/2001.
- [6] D. Comer .”**Interligação em Rede com TCP/IP**”, Volume I, “Princípios, protocolos e arquitetura“. Editora Campus. BR RJ 1998.
- [7] J. Saraiva. Os switches do futuro 12-06-2001 ou “**Alta performance e flexibilidade**” Cisco (ver revistas)
- [8] K. Downes et AL, “**Internet Working Technologies Handbook**”. Editora Campus. BR RJ. 2ª Edição 2000.
- [9] D. Comer, D. Stevens. “**Interligação em Rede com TCP/IP**”, Volume II, “Projeto, implementação e detalhes internos”. Editora Campus. BR RJ 1999.
- [10] Berners-Lee, T., Fielding, R. and H. Frystyk, “**Hypertext Transfer Protocol -- HTTP/1.0**”, RFC 1945, Maio 1996. <http://www.ietf.org/rfc/rfc1945.txt>.
- [11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, “**Hypertext Transfer Protocol -- HTTP/1.1**”, RFC2616, Junho 1999. Esta RFC substitui a RFC 2068.
- [12] D. Dias, W. Kish, R. Muhkerjee, and R. Tewari. “**A scalable and highly available web server**”. In Proceedings of the IEEE Computer Conference (COMPCON), Santa Clara, Março 1996. <http://citeseer.nj.nec.com/dias96scalable.html>
- [13] V. Cardellini, M. Colajanni, P. Yu; “**Dynamic Load Balancing on Web-server System**”. IEEE Internet Computing, vol 3,nº 3, pp 28-39, May-June 1999.
- [14] P.V. Mockapetris, K. Dunlap.”**Development of the Domain Name System**”. In Proceeding of SGCOMM 88, Stanford, CA. Agosto, 1988.

- [15] T. Kwan, R. McGrath, D. Reed, “**NCSA’s World Wide Web server: Design and Performance**”, IEEE Computer, n° 11, pp. 68-74, Novembro 1995.
<http://citeseer.nj.nec.com/kwan95ncsas.html>
- [16] A.Singhai, S. Lim, S. Radia, “**The SunsSCALR framework for internet servers**”. Proc. of IEEE Fault Tolerant Computing Systems, Munich, Germany, Junho, 1998.
- [17] Cisco’s Distributed and Local Director. <http://www.cisco.com>.
- [18] K. Egevang, P.Francis, “**The IP Network Address Translator (NAT)**”, RFC 1631, Maio, 1994. <http://www.ietf.org/rfc/rfc1631.txt>
- [19] E. Anderson, D.Patterson, E Brewer, “**The Magicrouter, an application of fast packet interposing**”, University of California, Berkeley, Maio, 1996.
- [20] G. Hunt, “**Network Dispatcher: a connection router for scalable Internet service**”, Computer Networks and ISDN Systems, n.30, pp. 347-357, 1998.
- [21] O.P. Damani, P.Y. Ching, Y. Hiang, C. Kintala, Y. Wang, “**ONE-IP : Techniques for hosting a service on a cluster of machines**“, Jornal Computer Networks and ISDN Systems, v.29, pp-1019-1027, Setembro, 1997.
- [22]D.Andresen, T.Yang, V.Holmedahl, O.Ibarra, "**SWEB: Towards a Scalable World Wide Web Server on Multicomputers**", Proc. of 10th IEEE International Symp. on Parallel Processing (IPPS'96), pp. 850-856. April, 1996.
<http://citeseer.nj.nec.com/andresen96sweb.html>
- [23] A. Bestavros, M. E. Crovella, J. Liu, D. Martin, “**Distributed Packet Rewriting and its application to scalable server architectures**”, T.R. BUCS-TR-98-003, Universidade de Boston, Dezembro, 1997.
- [24] Linux Virtual Server Project, <http://www.linuxvirtualserver.org>.
- [25] V. Pai et Al., “**Locality-Aware Request Distribution in Cluster-based Network Servers**”, Proceeding of ACM 8ª International Conference Architectural Support for Prog. Language and Op. Systems, Outubro, 1998.
- [26] HydraWeb. HTTP load manager. <http://www.hydraWEB.com>, 1996.
- [27] Extend Ed, "**ShockAbsorber: A TCP Connection Router**",
<http://citeseer.nj.nec.com/75805.html>.

- [28] X. Zhang, M. Barrientos, J. B. Chen, and M. Seltzer. HACC: “**An Architecture for Cluster-Based Web Servers**”. In Proceedings of the 3rd USENIX Windows NT Symposium, Seattle, WA, Julho 1999. <http://citeseer.nj.nec.com/zhang99hacc.html>
- [29] E. Casalicchio, M. Colajanni, “**A client-aware dispatching algorithm for Web clusters providing multiple services**”, Proc. of 10th Int'l World Wide Web Conference, Hong Kong, Maio 2001.
<http://www.ce.uniroma2.it/SelectedPublications/www10-434.ps>
- [30] Alteon Web Systems. <http://www.alteonwebsites.com>
- [31] Cisco Systems Inc, <http://www.cisco.com>.
- [32] F5 Network Inc. <http://www.f5labs.com>.
- [33] At Radware, <http://www.radware.com>
- [34] International Business Machines Corporation – IBM. <http://www.ibm.com>
- [35] Cardellini, V., Colajanni, M., and Yu, P. S.(2000) “**Geographic load balancing for scalable distributed Web systems**”. In Proc. IEEE Mascots 2000, São Francisco, CA, Agosto/Setembro 2000.

GLOSSÁRIO

- link** – Canal de comunicação em rede, formado por um circuito ou caminho de transmissão e todo o equipamento relacionado entre o remetente e o destinatário. Costuma fazer referência a uma conexão WAN. Às vezes é chamado de linha ou link de transmissão.
- WAN** – Rede de Longa Distância. Rede de comunicação de dados que serve os usuários em uma ampla área geográfica e freqüentemente usa dispositivos de transmissão fornecidos por portadoras comuns.
- LAN** – Rede Local. Rede de dados de alta velocidade e baixa ocorrência de erros, cobrindo uma área geográfica relativamente pequena. As LAN conectam estações de trabalho, periféricos, terminais e outros dispositivos existentes em uma área geograficamente limitada. Os padrões de LAN especificam o cabeamento e a sinalização na camada física e na camada de link de dados no modelo OSI.
- OSI** – Conexão de sistema aberto. Programa internacional de padronização criado pela ISO e pelo ITU-T para o desenvolvimento de padrões de redes de dados que facilitem a interoperabilidade de equipamentos de vários fabricantes.
- ISO** - Organização Internacional para padronização. Organização internacional responsável por uma ampla variedade de padrões, incluindo os padrões relevantes a redes, como o modelo de referência OSI.
- ITU-T** – Setor de padronização de telecomunicação da União de Telecomunicações Internacionais. Organização internacional que desenvolve padrões mundiais para tecnologias de telecomunicações.