Desempenho em ambiente *Web* considerando diferenciação de serviços (*QoS*) em *cache*s, rede e servidor: modelagem e simulação

Iran Calixto Abrão

Orientador: ***Prof. Dr. Marcos José Santana***

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

USP – São Carlos
Novembro de 2008

Desempenho em ambiente *web* considerando diferenciação de serviços (*QoS*) em *cache*, rede e servidor: modelagem e simulação

Iran Calixto Abrão

Dedico este trabalho a minha família,
em especial aos meus pais José Robeto e
Vera e aos meus irmãos Iury e
Katiuscia.
À Thatia por sua presença e amor
incondicional
E à Maria Clara por sua alegria, seu
amor e sua compreensão

Muito obrigado.
AMO VOCÊS!

Agradecimentos

Registrar os agradecimentos pela finalização desta tese é uma tarefa bastante difícil, mais complicada que redigir as conclusões do trabalho, não apenas pela grande quantidade de pessoas e instituições que, em diferentes momentos, possibilitaram que esse sonho se realizasse, mas principalmente pelo significado deste processo.

A realização de qualquer trabalho na vida envolve a participação de um grande número de pessoas que direta ou indiretamente oferecem a sua contribuição. A confecção desta tese não foi diferente. Várias pessoas estiveram e estão envolvidas com seu amor, seu trabalho e seu apoio. Desejo expressar os meus sinceros agradecimentos:

À DEUS pela minha vida por ter me dado forças para vencer todas as barreiras e concretizar mais esse objetivo que sempre sonhei. Obrigado por proteger-me nesta jornada e por mais esta conquista.

Ao Prof. Marcos Santana, minha eterna gratidão pela excelente orientação, pela confiança, pela paciência, pelo incentivo e pela oportunidade de desenvolver este trabalho no LaSDPC.

À Profa. Regina Santana, por sua disponibilidade pelas sugestões e opiniões seguras que auxiliaram muito no desenvolver desse trabalho.

Aos amigos e colegas do LaSDPC que auxiliaram e opinaram durante o desenvolvimento da tese, principalmente nos Seminários do grupo. Sentirei saudades!!!!

À PUC Minas que forneceu recursos físicos essenciais para o desenvolvimento do trabalho, mas, principalmente, ofereceu apoio e incentivo.

Aos professores e amigos da PUC Minas, principalmente aos professores do Curso de Ciência da Computação.

Aos grandes e eternos amigos Márcio e Cláudio, pelo exemplo de dedicação ao trabalho, pela franqueza de suas palavras, pelo apoio nos momentos difíceis e pelas risadas nos momentos de alegria.

Aos meus familiares que estiveram sempre presentes de perto e de longe, todos de quem eu tive que me privar da companhia, mas tiveram paciência de aguardar esta tese acabar.

Aos meus pais, José Roberto e Vera, que nunca mediram esforços para que eu pudesse realizar meus sonhos e que sempre me apoiaram.

À minha irmã Katiuscia pela paciência em me escutar.

Ao meu irmão Iury, Regina e Yasmin meu eterno carinho.

Por último (mas os últimos são sempre os primeiros), à Maria Clara, pela compreensão, alegria e ternura sempre manifestadas apesar do “débito” de atenção. E, à Thatia pela sua enorme paciência, por seu incentivo nos momentos que fraquejei, por suas risadas e por nosso amor.

Resumo

Esta tese de doutorado apresenta a investigação de alternativas para melhorar o desempenho de ambientes *Web*, avaliando o impacto da utilização de mecanismos de diferenciação de serviços em todos os pontos do sistema. Foram criados e modelados no *OPNET Modeler* cenários com diferentes configurações voltadas tanto para a diferenciação de serviços, quanto para o congestionamento da rede. Foi implementado um servidor *cache* com suporte à diferenciação de serviços (*cache* CDF), que constitui uma contribuição dentro deste trabalho, complementando o cenário de diferenciação de serviços de forma positiva, assegurando que os ganhos obtidos em outras etapas do sistema não sejam perdidos no momento da utilização do *cache*. Os principais resultados obtidos mostram que a diferenciação de serviços introduzida de forma isolada em partes do sistema, pode não gerar os ganhos de desempenho desejados. Todos os equipamentos considerados nos cenários propostos possuem características reais e os modelos utilizados no OPNET foram avaliados e validados pelos seus fabricantes. Assim, os modelos que implementam os cenários considerados constituem também uma contribuição importante deste trabalho, uma vez que o estudo apresentado não se restringe a uma modelagem teórica, ao contrário, aborda aspectos bem próximos da realidade, constituindo um possível suporte de gerenciamento de sistemas *Web*.

Abstract

This PhD thesis presents the investigation of alternatives to improve the performance of Web environments by evaluating the impact of using differentiated service mechanisms in all points of the system. Several scenarios were created *and* modeled in the OPNET Modeler, with different configurations of both differentiated services *and* network overloading. A special cache server supporting differentiated services (CDF cache) was proposed *and* included in the model, comprising one of the major contributions of this work once it positively complements the differentiated service scenario, making that the gains obtained with other stages of the system do not be spoiled when using the cache. The main results obtained show that the adoption of differentiated services in isolated parts of the system cannot generate the expected performance gains. The features of all the equipments considered in the several scenarios defined in this work are very close to the reality *and* the models used in the OPNET were evaluated *and* validated by the companies that produce those equipments. Thus, the models that implement the scenarios considered in this work also comprises an important contribution of this thesis, once the study presented is not just a theoretical modeling exercise but, conversely, it approaches aspects very close to the reality, comprising a possible Web system management support.

Sumário

Lista de Figuras

Lista de Tabelas

Abreviaturas

ARPA	*Advanced Research Projects Agency*
ASDA	*Ambiente de Simulação Distribuída Automático*
ASiA	*Ambiente de Simulação Automático*
DiffServ	*Serviços Diferenciados*
DS	*Differentiated Services.*
DSCP	*Differentiated Services Codepoint*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IGMP	*Internet Group Management Protocol*
IntServ	*Serviços Integrados*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
MEF	*Máquina de Estados Finitos*
NAM	*Network Animator*
NS	*Network Simulator*
OPNET	*Optimized Network Engineering Tool*
OSI	*Open Systems Interconnection*
PHB	*Per-hop Behavior*
QoS	*Qualidade de Serviço*
RFC	*Request for Comments*
RSVP	*Resource Reservation Protocol*
SLA	*Service Level Agreement*
SWDS	*Servidor WEB com Diferenciação de Serviços*
TCP	*Transfer Control Protocol*
UDP	*User Datagram Protocol*
XML	*Extensible Markup Language*

Capítulo

1

Introdução

1.1 Contextualização

A evolução dos meios de comunicação e a interconexão de diferentes equipamentos eletrônicos permitiram que novos modelos e tecnologias de redes de comunicação surgissem, facilitando e aumentando o intercâmbio de informações. Juntamente com esses novos modelos foram introduzidos diferentes serviços, na maioria das vezes sem um planejamento adequado envolvendo estudos de avaliação de desempenho e do impacto sobre os usuários.

Originalmente, a Internet não foi projetada para dar suporte à grande carga de trabalho atual e a prover recursos para as diferentes aplicações que surgiram. As primeiras trocas de informações na Internet eram formadas somente por arquivos texto, oferecendo-lhe pouca carga. Nos últimos anos, impulsionados principalmente pelo surgimento e rápida expansão da *Web*, aliado ao crescimento dos provedores comerciais, o tráfego aumentou em algumas ordens de grandeza, numa tendência que se mantém até hoje. A mudança não ocorreu apenas na quantidade de tráfego, mas também na natureza do mesmo. O uso de imagens e vídeos está disseminado na *Web* e aplicações multimídia já são utilizadas largamente. A *Web* também se transformou há algum tempo em uma plataforma para a realização de transações comerciais, o chamado comércio eletrônico (XIAO *and* NI 1999).

O serviço oferecido pela Internet baseia-se, ainda, em um modelo de melhor esforço (*best-effort*) (GEVROS, *et al.* 2001). Cada usuário envia seus dados e compartilha a mesma largura de banda com todos os fluxos de dados dos outros usuários. Os dados

são transportados da melhor forma possível para chegar ao seu destino, conforme as rotas definidas e a largura de banda que estiver disponível. Quando há congestionamento, pacotes são descartados sem distinção, não havendo garantia de que o serviço será realizado com sucesso ou de desempenho. Entretanto, muitas aplicações necessitam de tais garantias.

É fato que a Internet está, atualmente, caminhando em direção a um ponto de estrangulamento e a solução não é somente adicionar à rede mais largura de banda ou equipar os servidores com mais capacidade de processamento e armazenamento. A sobrecarga na Internet pode ser percebida facilmente observando-se os altos tempos de recuperação dos objetos solicitados na *Web*. Assim, é necessário avaliarem-se os modelos existentes na *Web*, tais como os modelos de diferenciação de serviços e as políticas de utilização de *cache* na *Web* e, a partir dessa avaliação, integrar e introduzir novos modelos que sejam capazes de atender às novas exigências das aplicações na *Web* adequadamente, conferindo-lhe Qualidade de Serviço (*QoS*).

1.2 Motivação

A *Web* se tornou o maior e mais acessado serviço da Internet e esse desenvolvimento ocorreu de maneira espontânea devido, principalmente, a algumas características particulares tais como:

- Interatividade;
- Conectividade;
- Redução do custo de acesso ao sistema
- Número de serviços oferecidos pela *Web;*

Esses fatores permitiram o crescimento acelerado e a popularização da *Web*, gerando sobrecarga na Internet e nos servidores, o que acarreta maiores tempos de resposta às requisições e degradação dos sistemas e serviços. Quando são feitos pedidos para servidores em conexões lentas, existe geralmente uma demora considerável na

recuperação de objetos remotos. Além disso, a alta taxa de transferência de objetos pela rede leva a um aumento de tráfego que acaba reduzindo a largura de banda disponível e, também, introduzindo atrasos perceptíveis ao usuário.

A introdução de *QoS* na Internet (XIAO *and* NI 1999) tem sido apontada como uma necessidade e uma exigência das empresas que oferecem serviços pela *Web* há algum tempo. Provedores de serviço desejam oferecer aos seus usuários serviços com vários níveis de diferenciação em qualidade e preços. Os usuários, por outro lado, desejam utilizar aplicações multimídia a qualquer hora do dia, o que é difícil pela falta de qualidade de serviço na Internet (EL-GENDY, BOSE *and* SHIN 2003). A *QoS* é necessária nos diferentes níveis de estrutura da *Web*, tais como nível de rede e nível de aplicação, pois, soluções desconexas de *QoS* podem não gerar o efeito desejado, uma vez que existe a necessidade de que todos os elementos da rede permitam esse controle diferenciado, não existindo pontos de estrangulamento que possam degradar o sistema.

Para melhorar o desempenho e a escalabilidade da *Web* é necessária a execução de soluções conjuntas e não soluções isoladas, pois o ambiente da Internet não pode ser tratado separadamente. Fatores importantes, tais como o planejamento na implantação de serviços *Web*, a caracterização da carga do sistema, o planejamento na implantação de servidores *cache* e a utilização de diferenciação de serviços nos diferentes níveis, devem ser estudados em conjunto para possibilitar um melhor desempenho dos serviços *Web* (ZHOU, HASSANEIN *and* MARTIN 2004).

Uma premissa para melhorar o desempenho da *Web* é a compreensão do tráfego de rede, pois este é fundamental para um bom planejamento e implementação adequada de servidores *cache* na *Web*. A utilização eficiente de servidores *cache* pode proporcionar o uso eficiente da largura de banda e reduzir a latência na recuperação dos objetos. Porém, somente a utilização isolada de uma boa política de reposição de *cache* pode não representar um ganho de qualidade para o usuário final (PODLIPNIG *and* BÖSZÖRMENYI 2003).

Além disso, podem-se ter diferentes tipos de usuários e diferentes tipos de tráfego circulando pela *Web*. Um usuário que esteja simplesmente navegando por um *site* de comércio eletrônico, comparado com outro em processo de finalização de uma compra, deve receber uma prioridade de atendimento bem menor que a do segundo, o que não ocorre atualmente. A mesma situação se dá em nível de aplicação, com as solicitações que chegam aos servidores *Web*, as quais recebem um tratamento uniforme, sem distinção. Sendo assim, verifica-se claramente a necessidade de técnicas para provisão de diferentes classes de serviços na *Web*, a fim de atender aos diversos requisitos dos usuários e aplicações. A diferenciação de serviços implantada de forma isolada também pode não gerar os ganhos desejados, causando prejuízos às empresas que venderam os serviços (CARPENTER *and* NICHOLS 2002).

A integração das diferentes soluções de melhoria de desempenho para os serviços *Web* se faz necessária e este trabalho se propõe a estudar o impacto da integração de servidores *cache* na *Web* com a utilização de serviços diferenciados em nível de rede e em nível de aplicação,

1.3 Objetivos

O presente trabalho tem como objetivo geral a investigação de alternativas para melhorar o desempenho de aplicações *Web*, avaliando o impacto da utilização de servidores *cache na Web*, em ambientes com e sem diferenciação de serviços.

Dentre seus objetivos específicos destacam-se:

- Modelagem de um ambiente geral para *Web* adotando servidores de *cache* e diferenciação de serviços, em nível de rede e em nível de aplicação, visando a ganhos expressivos de desempenho e a garantias de qualidade de serviço.
- Avaliação do impacto da utilização de servidores *cache* na *Web* em ambientes com provisão de serviços diferenciados;

- Desenvolvimento de um modelo de servidor *cache* para dar suporte às aplicações que necessitem de diferentes níveis de serviço;
- Avaliação do impacto da utilização de diferenciação de serviços em nível de rede no ambiente *Web*;

Pretende-se com este trabalho avaliar o desempenho dos principais serviços *Web*. Para tal, o uso de modelagem como ferramenta básica de análise permitirá abstrair as principais características dos servidores *Web*, dos servidores *cache* e do suporte a diferenciação de serviços.

Esta tese objetiva contribuir para um melhor entendimento dos aspectos relacionados à utilização de mecanismos de diferenciação de serviços na *Web* em todos os pontos da rede, desde o usuário até os servidores *Web*. O uso de modelagem como ferramenta de análise permite abstrair as principais características do ambiente da *Web*, com ênfase na diferenciação de serviços nos diferentes elementos desse ambiente. Essa arquitetura é incrementada com diferentes configurações voltadas tanto para a diferenciação de serviços, quanto para o congestionamento da rede.

1.4 Organização dos Capítulos

O capítulo 2 desta tese apresenta uma introdução sobre a Internet, a sua evolução em relação ao número de usuários, os serviços *Web*, o protocolo *TCP (Transfer Control Protocol*) e o protocolo *HTTP (Hypertext Transfer Protocol)* que dão suporte à grande parte das aplicações da *Web*. No capítulo 3, aborda-se o assunto de diferenciação de serviços na Internet e discutem-se as limitações do seu modelo atual de atendimento a clientes. São apresentados conceitos de qualidade de serviço e detalhadas as arquiteturas de serviços integrados e diferenciados, com ênfase nessa última. O capítulo 4 apresenta

os principais conceitos envolvidos com servidores *cache* na *Web* e como sua utilização pode auxiliar para melhorar o desempenho do ambiente *Web*. Também, nesse capítulo, é descrito o servidor *cache* com suporte à diferenciação de serviços implementado, chamado *cache* CDF.

No capítulo 5 é apresentada uma visão geral do processo de avaliação de desempenho de sistemas, com ênfase na modelagem, com solução por meio de simulação. São descritos alguns ambientes de simulação utilizados para transcrição de modelos definidos pelo usuário e é apresentada a ferramenta *OPNET*, que constitui o ambiente de modelagem e de simulação utilizado para o desenvolvimento dos modelos e das simulações existentes neste trabalho. No capítulo 6 aborda-se a caracterização da carga de trabalho da *Web* com o objetivo de melhor descrever o comportamento das requisições dos usuários que os servidores atendem.

No capítulo 7 é discutida a necessidade da construção de modelos, que permitam compreender a complexidade da *Web.* A avaliação de desempenho baseada em modelos apresenta-se como uma forma adequada no processo de avaliação de desempenho de sistemas *Web*. São apresentados os modelos de ambiente *Web* descritos na literatura e é descrito o modelo construído no *OPNET Modeler*, que serve de base para todos os experimentos realizados neste trabalho.

O capítulo 8 apresenta a análise dos principais resultados obtidos nas simulações realizadas neste trabalho. Os dados são apresentados em formas de gráficos e tabelas, o que permite a comparação entre cinco cenários distintos, adotados no trabalho, bem como a comparação entre os ambientes com quatro taxas de congestionamento diferentes. Ainda, é possível a comparação entre as quatro classes de usuários adotadas que compreendem usuários padrão, bronze, prata e ouro.

Finalmente, o Capítulo 9 apresenta as principais conclusões e as contribuições oriundas desta tese, bem como aponta caminhos para a continuidade da pesquisa através de trabalhos futuros.

Capítulo

2

A Internet

2.1 Considerações Iniciais

A Internet teve sua origem no final da década de 60 no projeto *ARPANET* coordenado pela *Advanced Research Projects Agency – ARPA*, nos Estados Unidos. O objetivo inicial do projeto era obter um meio de comunicação que funcionasse mesmo se algum ponto fosse afetado em caso de um bombardeio; era o auge da Guerra Fria (LEINER *and* CERF 2003).

O termo Internet surgiu em 1983, quando a *ARPANET* foi dividida em duas redes: uma militar, a *MILNET* e uma versão reduzida da *ARPANET*. A Internet hoje é uma grande coleção de redes interligadas no mundo inteiro, formando uma grande rede de computadores.

Neste capítulo, serão abordados alguns tópicos sobre a Internet, tais como, a evolução do número de *hosts* e usuários, os principais protocolos, que compreendem o *TCP/IP* e o *HTTP*, usados na comunicação entre clientes e servidores. Ainda, são discutidos os serviços oferecidos pela Internet denominados de serviços *Web* e a dificuldade em se modelar e simular a Internet.

2.2 A Evolução da Internet

O crescimento da Internet sempre foi em escala exponencial, sendo que a quantidade de componentes conectados nos seus primeiros dias, cerca de 10 nós, transformou-se em mais de 100 milhões de nós em menos de 30 anos, e ainda continua crescendo. De acordo com dados do Comitê Gestor da Internet no Brasil (CGI.br 2008) e do Internet Systems Consortium, Inc. (ISC 2007) pode-se verificar o crescimento do numero de *hosts* no mundo. A

Tabela 2.1 apresenta os dados de 1999 a julho de 2007 e a Figura 2.1 mostra o gráfico da evolução desses números.

Tabela 2.1 - Evolução do numero de *HOSTS* no mundo

Ano	Total *Hosts* Mundo
1999	43.230.000
2000	93.047.785
2001	125.888.197
2002	162.128.493
2003	171.638.297
2004	285.139.107
2005	353.284.187
2006	439.286.364
2007	489.774.269

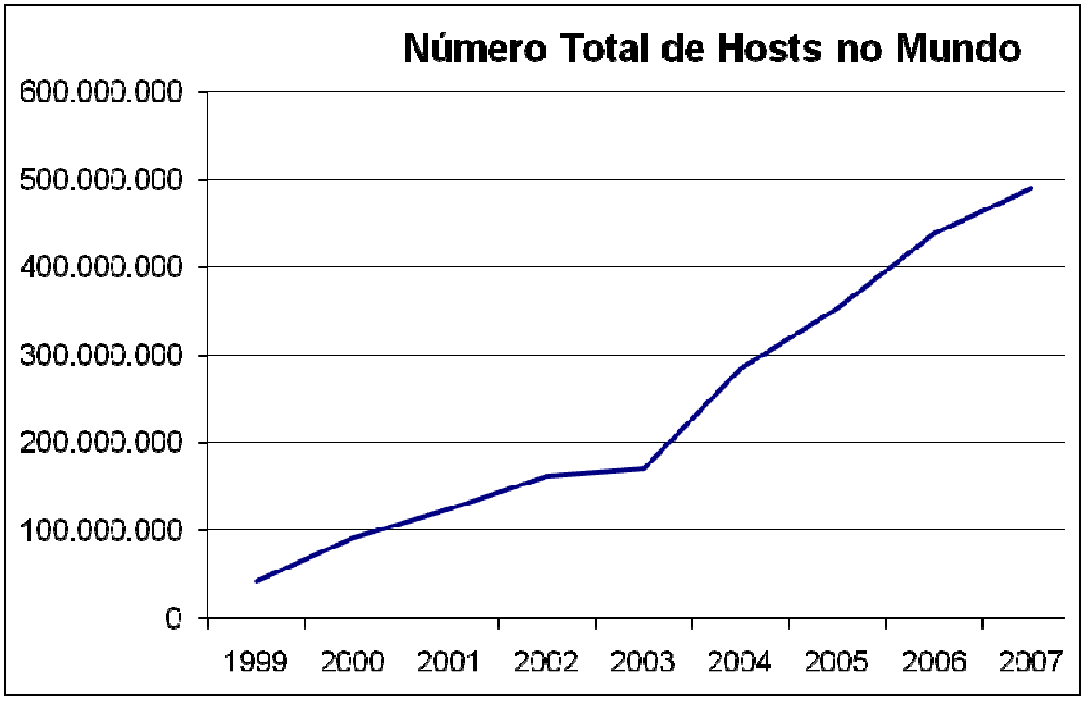


Figura 2.1 - Gráfico da evolução do número de *HOSTS* no mundo

O Brasil ocupa a nona posição em relação ao número total de domínios conforme ilustrado na Tabela 2.2 (CETIC.br 2008).

Tabela 2.2 – Países por número de *hosts*

País	Total Domínios .br
Estados Unidos	302.884.146
Japão (.jp)	36.803.719
Alemanha (.de)	20.659.105
Itália (.it)	16.730.591
França (.fr)	14.356.747
China (.cn)	13.113.985
Austrália (.au)	10.707.139
Holanda (.nl)	10.540.083
Brasil (.br)	**10.151.592**
México (.mx)	10.071.370

Ainda, de acordo com o relatório de PESQUISA SOBRE O USO DAS TECNOLOGIAS DA INFORMAÇÃO E DA COMUNICAÇÃO NO BRASIL 2007 elaborado pelo Centro de Estudos sobre as Tecnologias da Informação e da Comunicação (CETIC.br 2008) podem-se destacar alguns números:

- 24% dos domicílios brasileiros possuem computadores;
- A Internet atingiu 17% do total de domicílios brasileiros;

- Em 2007, pela primeira vez, mais da metade da população brasileira com mais de 10 anos (53%) informou já ter usado um computador e que 41% já usaram a Internet;
- 45% das pessoas que já utilizaram a Internet declararam ter realizado pesquisas de preço de produtos ou serviços pela rede em 2007, enquanto apenas 16% informaram ter realizado uma compra via *Web*.

Apesar do aumento do número de computadores nos domicílios e do acesso à Internet, mais da metade da população brasileira com mais de 10 anos ainda não usou a Internet.

2.3 Protocolos TCP/IP

Os protocolos *TCP/IP* foram criados juntamente com a Internet, como solução para a interligação dos computadores nessa rede. O sucesso da Internet fez com que o *TCP/IP* se tornasse um padrão de fato para a interligação de computadores, tanto em redes locais quanto de longa distância e demonstrou também a viabilidade do uso da Arquitetura *TCP/IP* em larga escala. As principais características do protocolo *TCP/IP* são (COMER 2000):

- **Uso de padrões abertos.**
  O *TCP/IP* está disponível livremente, não estando preso a nenhuma plataforma de *hardware* ou *software* específica. As especificações dos protocolos *TCP/IP* estão disponíveis na Internet, através de *RFCs* (*Request for Comments*), promulgadas pela *IETF - Internet Engineering Task Force*.

- **Independência da tecnologia de rede.**
  O *TCP/IP* usa a comutação de pacotes para a comunicação entre as partes e é capaz de funcionar sobre uma variedade de protocolos das camadas de enlace e física, de forma transparente para as aplicações. Por essa razão, é a solução mais escolhida para a interligação de *hardware* e *software* diversos.

- **Protocolos padronizados.**
  Os protocolos *TCP/IP* são padronizados não apenas no nível de rede e de transporte, mas também em nível de aplicação. Existem, por exemplo, padrões para *e-mail*, *login* remoto e transferência de arquivos, o que torna mais fácil a confecção de novas aplicações.

- **Interconexão total.**
  Cada dispositivo conectado a uma rede *TCP/IP* recebe um endereço único que o identifica e permite que ele se comunique e seja acessível de qualquer outro ponto da rede.

- **Confirmações fim-a-fim.**
  O *TCP/IP* fornece um mecanismo de reconhecimento (*acknowledgement)* entre a origem e o destino final de uma comunicação, em vez de simplesmente entre pares de máquinas ao longo do caminho, o que torna a entrega de dados, como um todo, mais confiável.

2.3.1 Transações com TCP

O protocolo *TCP* provê um serviço de transmissão de dados orientado à conexão, ordenado e confiável. Mas, para garantir tudo isso, ele causa também um grande consumo de recursos (*overhead*). Uma conexão *TCP* é formada por três fases: o estabelecimento de conexão, a troca de dados e a finalização da conexão.

A fase inicial de estabelecimento de conexão é realizada pela troca de três mensagens, formando o *three-way-hanshaking*. O cliente inicia a ligação enviando um pacote *TCP* com a *flag SYN* e espera que o servidor aceite a ligação, enviando de volta um pacote *SYN*. Se, durante um determinado espaço de tempo, esse pacote não for recebido ocorre um *timeout* e o pacote *SYN* é reenviado. O estabelecimento da ligação é concluído por parte do cliente, confirmando a aceitação do servidor respondendo-lhe com um pacote *ACK*.

A fase de finalização da conexão *TCP* é um processo de quatro fases, em que cada interlocutor responsabiliza-se pelo encerramento do seu lado da ligação. O cliente, após enviar seu pedido, envia um segmento *FIN* (finalização), indicando que não transmitirá mais dado. O servidor responde com um segmento *ACK* que confirma tanto o recebimento do pedido quanto o do segmento *FIN*; neste momento a conexão está semi-fechada (só no sentido cliente para servidor). O servidor, então, manda um segmento com a resposta e mais um segmento *FIN*, para finalizar a conexão no outro sentido. Finalmente, o cliente envia um segmento *ACK* para a chegada da resposta e do *FIN* do servidor. A partir desse instante, a conexão está fechada completamente. A Figura 2.2 ilustra o estabelecimento, troca de dados e o término de uma conexão *TCP*.

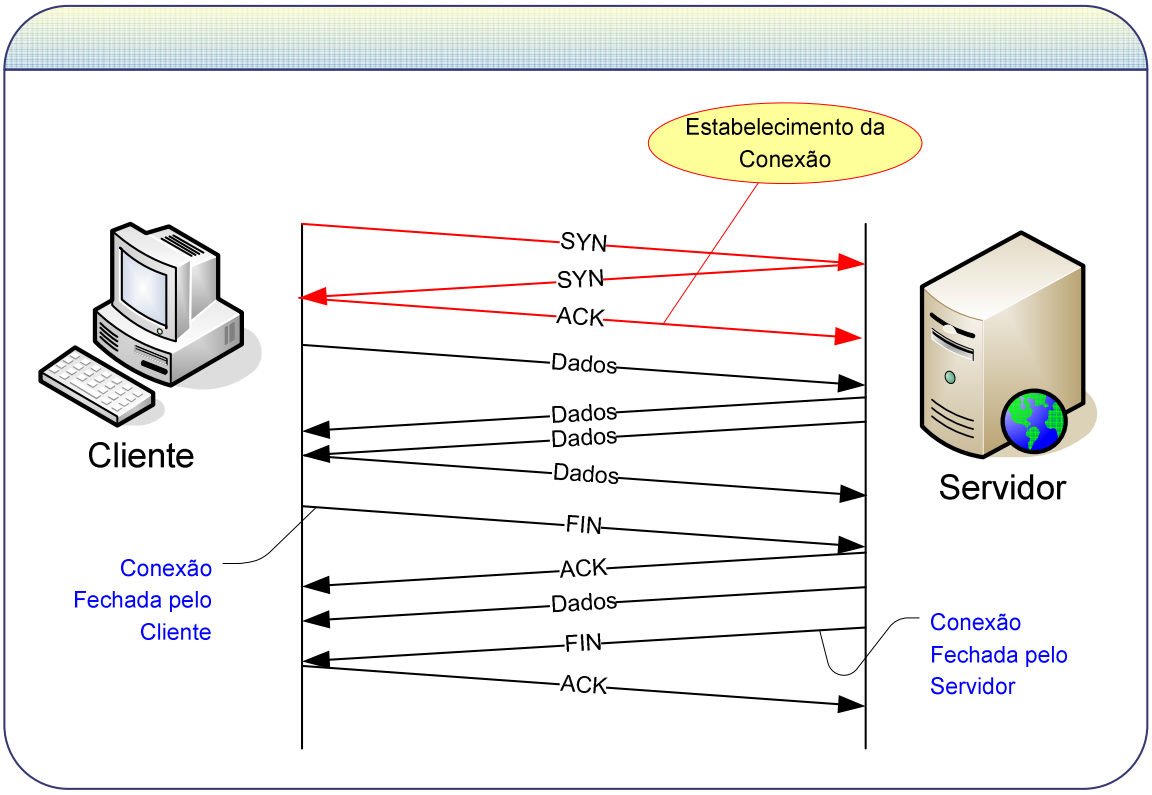


Figura 2.2 – Estabelecimento e término de uma conexão *TCP*

2.4 Protocolo HTTP

O *Hypertext Transfer Protocol – HTTP*- (BERNERS-LEE, FIELDING *and* FRYSTYK 1992) é um protocolo usado para a transferência de informações na *Web*. Os dados transferidos pelo *HTTP* podem ser hipertextos, textos não estruturados, imagens, vídeo, ou qualquer outro tipo de informação.

O *HTTP* define uma única interação de requisição/resposta, que é vista como uma "transação *Web*". Cada transação *HTTP* consiste em uma requisição enviada do cliente ao servidor, seguido de uma resposta enviada do servidor ao cliente, como ilustrado na Figura 2.3.

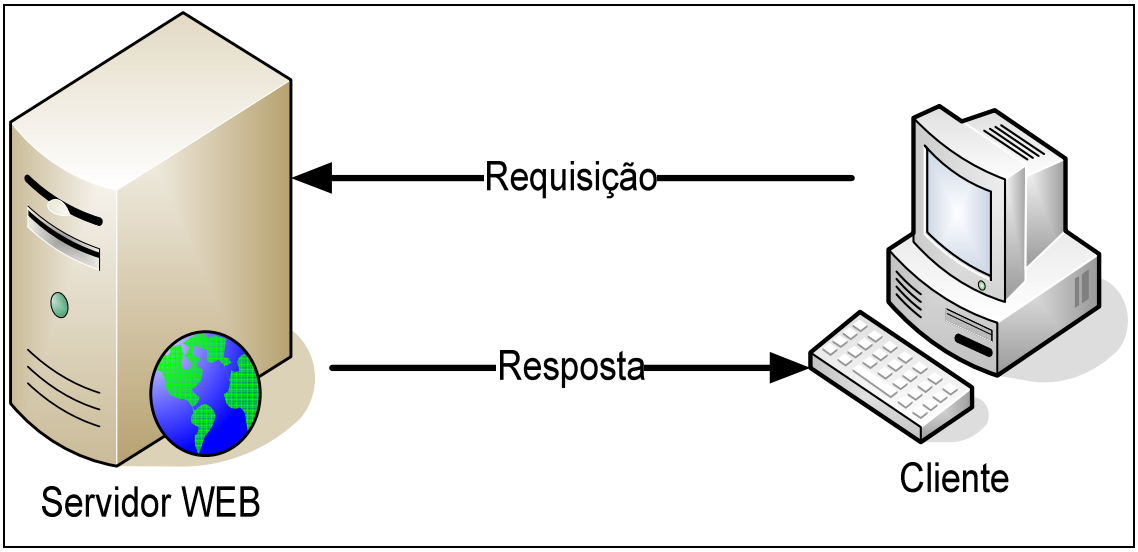


Figura 2.3 – Esquema de uma transação *Web*

Quando um servidor recebe uma requisição, ele analisa a requisição e verifica qual ação deve ser executada de acordo com o método especificado. O protocolo *HTTP* define um conjunto de métodos que o cliente pode invocar. Esses métodos funcionam como comandos enviados ao servidor.

O *HTTP* é um protocolo da camada de aplicação (TANENBAUM 2002). O *HTTP* não se preocupa com os detalhes de uma comunicação da rede, função desempenhada pelo

TCP/IP. As principais etapas envolvidas em uma transação entre o cliente e o servidor são:

- Mapear o nome do servidor para um endereço *IP*;
- Estabelecer uma conexão *TCP* com o servidor;
- Transmitir a requisição com todas as informações necessárias;
- Receber a resposta do servidor;
- Fechar a conexão *TCP*

Cada uma dessas etapas possui um custo que depende do desempenho do servidor e da rede.

O *HTTP* é um protocolo *stateless*, ou seja, o servidor não guarda nenhuma informação em relação ao estado dos clientes. Caso ocorra alguma falha na execução da tarefa, é responsabilidade do cliente fazer novamente a transação, reenviando todas as informações necessárias para refazê-la. Cada transferência é totalmente isolada da requisição anterior ou da próxima. Considerando que milhões de clientes podem ter acesso ao servidor simultaneamente, a ausência de estado do protocolo *HTTP* aumenta a eficiência do sistema, pois dados dos clientes não precisam ser armazenados e/ou gerenciados pelo servidor. Por outro lado o desempenho do protocolo *HTTP* fica prejudicado por essa característica, pois uma nova conexão *TCP* sempre tem que ser estabelecida para cada requisição. A versão 1.1 do protocolo *HTTP* (FIELDING, GETTYS, *et al.* 1999) procura minimizar o custo de múltiplas conexões utilizando-se de conexões persistentes, que permitem utilizar uma conexão *TCP* para realizar um conjunto de operações simultâneas.

É importante destacar que, embora sob o ponto de vista do usuário, uma solicitação de página se resuma, muitas vezes, a apenas um clique do mouse, na verdade ela freqüentemente dá origem a várias transações *HTTP*, que são enviadas do *browser* para o servidor *Web* (seguidas de suas respectivas respostas no sentido contrário). Isso ocorre porque, para cada objeto contido em uma página *HTML*, é gerada uma requisição independente ao servidor. No protocolo *HTTP* 1.0 (BERNERS-LEE, FIELDING *and* FRYSTYK 1992) isso implica em estabelecer uma nova conexão *TCP* para cada objeto

solicitado, o que pode levar a uma sobrecarga desnecessária no servidor e na rede. Antes de iniciar uma transação, o *browser* e o servidor *Web* geralmente negociam os tipos de dados que serão trocados, o que também contribui para a sobrecarga inerente ao *HTTP*.

2.4.1 HTTP 1.0 e 1.1

A versão 1.0 do protocolo *HTTP* (BERNERS-LEE, FIELDING *and* FRYSTYK 1992) foi introduzida juntamente com a *Web*, em 1990. Nessa versão inicial, ele era apenas um modo simples de recuperar informações através da Internet. Com o crescimento da *Web*, surgiram novos requisitos e algumas de suas limitações foram aparecendo.

A principal limitação do *HTTP* 1.0 é a necessidade de estabelecer sempre uma nova conexão *TCP* para cada objeto solicitado. Inicialmente, quando os documentos eram constituídos basicamente por texto, isso não representava um grande problema, porém, atualmente, em que uma simples página pode conter dezenas de pequenas imagens, isso tende a causar uma grande sobrecarga no tráfego da Internet, bem como nos servidores.

O protocolo *HTTP* 1.1 (FIELDING, GETTYS, *et al.* 1999), padronizado em 1999 pelo W3C (*World Wide Web Consortium*), usa como padrão um esquema de conexões persistentes, que permite que uma mesma conexão *TCP* seja usada por várias transações *HTTP*, o que o torna bem mais eficiente, quando comparado com a versão 1.0.

O *HTTP* 1.1 também permite fazer o *pipelining* de requisições. Nesse caso, várias requisições são enviadas em seqüência, sem aguardar pelas respostas. Isso é muito útil na recuperação de várias imagens de uma página em ambientes que possuam uma alta latência para o estabelecimento de conexões *TCP*, por exemplo.

2.5 A Web

A *World Wide Web*, também conhecida como *Web* ou WWW, teve início em 1989 no CERN, o centro europeu para pesquisa nuclear. A *Web* nasceu da necessidade de fazer com que grupos de cientistas de diferentes nacionalidades pudessem colaborar uns com os outros, através da troca de relatórios, imagens, desenhos, fotos e outros documentos.

Em 1994, o CERN e o MIT assinaram um acordo criando o *World Wide Web Consortium* (abreviado como W3C), uma organização voltada para o desenvolvimento da *Web*, a padronização de protocolos e para o incentivo à interoperabilidade entre os sistemas. A *Web* tornou-se rapidamente o serviço mais utilizado em toda a Internet. Por volta de 1995, tornou-se responsável pela maior parte do tráfego na Internet superando todos os outros serviços, inclusive aplicações tradicionais como FTP e *e-mail*. Para muitos usuários, principalmente os mais leigos, a Internet é a *Web*.

Do ponto de vista dos usuários, a *Web* é uma vasta coleção de documentos, geralmente chamados de páginas. Cada página pode conter *links* para outras páginas em qualquer lugar do mundo. Os usuários podem seguir um *link* que os levará até a página indicada. Esse processo pode ser repetido indefinidamente. Inicialmente, as páginas eram formadas somente por textos e imagens. Hoje, as páginas *Web* podem ter áudio, vídeo e diversos outros documentos eletrônicos. Ainda, é possível o acesso a serviços de compras *online*, consultas bancárias e outros serviços chamados de serviços *Web*.

A base da *Web* é a transferência de páginas do servidor para o cliente. As páginas da *Web* podem ser divididas em:

- **Páginas Estáticas**: são páginas já prontas que ficam armazenadas em algum servidor esperando o momento de serem recuperadas pelos usuários.

- **Páginas Dinâmicas**: são páginas que são criadas no servidor de acordo com as requisições dos usuários. As páginas dinâmicas necessitam de um

processamento nos servidores *Web* e, geralmente, demandam mais tempo para serem retornadas aos usuários.

2.5.1 Serviços Web

A Internet é um sistema em constante evolução e expansão, que incorpora novos componentes e serviços em um ritmo muito acelerado. Os serviços que são oferecidos através da Internet, normalmente chamados de serviços *Web*, vão desde aplicações de correio eletrônico a comércio eletrônico. Os serviços *Web* são implementados de forma flexível e possuem um papel importante nas interações entre empresas por meio eletrônico.

Com o aumento do número de usuários e de serviços oferecidos pela *Web*, o problema de desempenho vem aumentando cada vez mais devido à natureza imprevisível dos pedidos de informação e serviços pela *Web*. Para tornar o problema mais complicado, existe uma grande população de agentes de *software* autônomos, robôs, interagindo com os servidores *Web* e consumindo recursos dos sistemas.

De acordo com (MENASCÉ *and* ALMEIDA 2002) há seis pontos a serem examinados quando se analisa o desempenho dos serviços na *Web*:

- Conteúdo;
- *Software* do servidor;
- *Hardware;*
- Aplicação;
- Largura da banda;
- Infra-estrutura.

Um serviço na *Web* pode ser executado em um *site* ou pode ser o resultado de vários serviços executados em diferentes *sites*. Os problemas mais comuns de desempenho estão relacionados à:

- Largura da banda insuficiente;
- Sobrecarga dos servidores;
- Processamento de conteúdo dinâmico;

- Conexões insuficientes entre servidores de aplicação e banco de dados, por exemplo;
- Falha dos serviços de terceiros;
- Remessa de conteúdo multimídia.

Um dos maiores problemas para os administradores de servidores *Web* é o dimensionamento adequado da infra-estrutura de Tecnologia da Informação necessária para oferecer a qualidade de serviço exigida pelos usuários. Para oferecer um serviço de qualidade é necessário sempre monitorar a intensidade de carga de trabalho, detectar gargalos no sistema, prever expansões para o sistema como um todo (*software* e *hardware*).

A modelagem de cenários que apresentem ambientes com serviços *Web* e a posterior simulação desses para avaliar o desempenho, são técnicas que podem auxiliar os administradores de sistemas a planejar melhor seus ambientes para oferecer serviço com qualidade aos usuários (YAN, *et al.* 2000) (LI, *et al.* 2000) (KILGORE 2002) (BERNARDO 2001). Porém, a modelagem e simulação desses ambientes inseridos na Internet não é tarefa trivial (FLOYD *and* PAXSON 2001).

2.6 Dificuldades para Simular a Internet

Simular como a Internet se comporta é uma tarefa extremamente complexa devido às suas características únicas, que tornam difícil obter uma descrição precisa a seu respeito. Fatores como a sua heterogeneidade e as grandes mudanças que ocorrem diariamente na rede são pontos que dificultam a modelagem e a análise do seu comportamento (FLOYD *and* PAXSON 2001).

Dentre as dificuldades existentes em modelar e simular o comportamento da Internet podem-se destacar:

- **Dimensão da *Internet***.

  A Internet possui dimensões globais, sob qualquer parâmetro que se queira observar. Sua dimensão pode ser medida, por exemplo, em número de usuários,

estações de trabalho conectadas à rede, tráfego de dados, número de acessos por segundo.

- **Heterogeneidade.**
  Um dos fatores do grande sucesso da Internet é a interconexão de diferentes equipamentos e diferentes plataformas. Essa característica facilita o intercâmbio de informações, entretanto, a facilidade em proporcionar conectividade causa uma grande dificuldade em compreender o seu comportamento, pois cada tecnologia de rede apresenta um comportamento diferente.

- **Mudanças Constantes**.
  Outra grande dificuldade para simular a Internet são mudanças constantes que ocorrem de maneira rápida e imprevisível. Muitos estudos revelam resultados surpreendentes, como crescimento súbito do tráfego em determinada situação e depois uma diminuição também expressiva em níveis semelhantes aos do início do crescimento. Exemplo de crescimento imprevisível é a expansão em aplicações multimídia com transmissão de áudio e vídeo.

- **Número Diferente de Aplicações**.
  O número de aplicações que são executadas na Internet cresce a cada dia. Cada aplicação possui características próprias, que muitas vezes torna difícil sua modelagem e simulação. A previsão da carga gerada por essas aplicações como utilização de CPU, memória e tráfego de rede são difíceis de serem previstas e calculadas.

2.7 Considerações Finais

Este capítulo apresentou um panorama geral da evolução da Internet, que em 2008 já ultrapassou o número de mais de 500.000.000 de *hosts* no mundo. No Brasil, pode-se observar que mais da metade da população brasileira com mais de 10 anos já afirmou ter utilizado computadores e que 43% já usaram a Internet. Esses números cada vez

mais crescentes demonstram a necessidade de estudos para melhorar o desempenho dos serviços oferecidos através da Internet.

Neste capítulo também foram discutidos os principais protocolos utilizados na Internet, compreendendo o *TCP/IP* e o *HTTP*. A expansão da Internet fez com que o *TCP/IP* se tornasse um padrão de fato para a interligação de computadores, tanto em redes locais quanto de longa distância. O *HTTP* é o principal protocolo utilizado para a transferência de informações na *Web*. Foram discutidos o funcionamento e o impacto desses protocolos no comportamento da Internet.

Também foram descritas algumas características dos serviços oferecidos através da Internet, denominados de serviços *Web*, que vão desde aplicações de correio eletrônico a comércio eletrônico. Novos serviços *Web* são oferecidos a cada dia, aumentando a demanda de recursos na Internet. As requisições dos usuários devem ser atendidas com a qualidade de serviço necessária para a realização da transação. Um dos maiores problemas para a implantação, monitoração e administração dos serviços *Web* é o dimensionamento adequado da infra-estrutura necessária para oferecer a qualidade de serviço exigida pelos usuários.

Para finalizar, foram discutidas as dificuldades que são encontradas em se modelar e simular o ambiente da Internet. A modelagem de cenários que apresentem ambientes com serviços *Web* e a posterior simulação desses cenários para avaliar o desempenho, são técnicas que podem auxiliar os administradores de sistemas a planejar melhor seus ambientes visando a oferecer um serviço com melhor qualidade aos usuários.

Outro ponto importante em relação ao fornecimento de serviço aos diferentes tipos de usuários é a necessidade de priorizar determinadas requisições. O próximo capítulo abordará a necessidade de soluções que tratem os usuários ou as requisições mais prioritárias de formas diferenciadas. Serão apresentadas as duas abordagens de fornecimento de *QoS* propostas pelo IETF: a Arquitetura de Serviços Integrados (BRADEN, CLARK *and* SHENKER 1994) e a Arquitetura de Serviços Diferenciados (BLAKE, *et al.* 1998) (CARPENTER *and* NICHOLS 2002).

Capítulo

3

Serviços Diferenciados

3.1 Considerações Iniciais

O crescimento da Internet nos últimos anos possibilitou o surgimento e crescimento de um novo conjunto de aplicações na *Web*. Entretanto, a Internet, com seu modelo de serviço do tipo *best-effort* (GEVROS, *et al.* 2001), vem dando sinais de estrangulamento há algum tempo. Esse tipo de serviço baseia-se em um modelo de melhor esforço, ou seja, a rede procura transportar os dados que lhe são confiados, no menor tempo possível, de preferência sem erros ou inconsistências, contudo não são dadas às aplicações garantias de que isso realmente acontecerá. Na maior parte das vezes, considerando-se uma rede com pouco tráfego e aplicações não muito exigentes, esse modelo se mostra satisfatório. Porém, ocorrem problemas quando a rede está sobrecarregada ou as aplicações possuem requisitos para os quais ela não foi projetada, tais como aplicações multimídia e transações comerciais.

Com o modelo de melhor esforço, todo tráfego é tratado de maneira uniforme, sem nenhum tipo de diferenciação ou priorização. Essa tendência se reflete até mesmo no projeto de serviços críticos da *Web*, cujos servidores, em sua grande maioria, tratam as requisições dos clientes segundo a filosofia de que, a primeira a chegar será a primeira a ser atendida.

Atualmente, pode ser verificado que os tipos de tráfegos e transações não são equivalentes ou possuem prioridades diferentes para os usuários. Não podem ser comparadas requisições de documentos disparadas por “agentes” com as requisições

originadas por um usuário. No primeiro caso, trata-se de uma requisição especulativa, que poderia ser atendida com uma prioridade inferior, enquanto que, no segundo caso, a requisição originada é de interesse para o usuário que a solicitou. Outro exemplo é o caso de um usuário que esteja simplesmente navegando por um *site* de comércio eletrônico, comparado com outro em processo de finalização de uma compra. O segundo usuário deveria receber uma prioridade de atendimento bem maior do que a dispensada ao primeiro, porém não é o que ocorre atualmente, pois, as requisições são tratadas da mesma maneira, geralmente sem que seja considerada sua prioridade relativa aos usuários e às aplicações. A mesma situação se dá em nível de aplicação, com as solicitações que chegam aos servidores *Web*, as quais recebem um tratamento uniforme, sem distinção. Sendo assim, verifica-se claramente a necessidade de técnicas para provisão de diferentes classes de serviços na Internet, a fim de atender aos diversos requisitos dos usuários e aplicações (TEIXEIRA, SANTANA *and* SANTANA 2005).

O tráfego atual na Internet, além de intenso, se origina de aplicações que possuem requisitos computacionais variados e a infra-estrutura existente na rede não está preparada para dar suporte adequado. Aplicações multimídia, por exemplo, necessitam de uma grande largura de banda e, em geral, podem suportar pequenas perdas de pacotes em favor de uma melhor sincronização na entrega dos mesmos. Por outro lado, aplicações de tempo real, como voz sobre *IP*, coloca pouca demanda sobre a largura de banda, mas, em compensação, necessitam de requisitos de temporização rígidos que, se não forem atendidos, podem inviabilizar a comunicação entre as partes. Aplicações de transferência de arquivos necessitam de largura de banda e não podem tolerar perda de nenhum bit de informação. Há também aplicações que necessitam de um serviço *multicast* como, por exemplo, videoconferência e transmissão de rádio e TV via *Web*. Além disso, a *Web* vem sendo cada vez mais utilizada em transações eletrônicas que exigem alta confiabilidade e disponibilidade do meio de transmissão, sendo que esses usuários estão dispostos a pagar mais por um serviço mais confiável e de melhor qualidade.

Para toda a gama de requisitos de aplicações existentes não há nenhum tipo de rede capaz satisfazê-los, visto que o ponto crítico não está somente na largura de banda disponível. Ainda, tem-se que considerar que a Internet é uma rede composta por várias redes, compreendendo as mais diversas tecnologias e, portanto, não é tão simples

melhorar a capacidade dos canais de comunicação uniformemente. Outro problema é a existência de tráfego em rajadas, o que pode levar a demanda por largura de banda a níveis muito superiores aos experimentados usualmente.

3.2 Qualidade de Serviço

Qualidade de Serviço (*QoS*) pode ser definida como a capacidade de fornecer a um elemento de rede (aplicação, cliente, servidor ou roteador) algum nível de segurança de que seus requisitos de tráfego e serviço serão satisfeitos. *QoS* está relacionada com a garantia de um atraso de entrega (*delay*) e uma perda de pacotes suficientemente baixos para certos tipos de aplicações ou tráfego (ZHAO, OLSHEFSKI *and* SCHULZRINNE 2000). Os requisitos de qualidade podem ser determinados por fatores humanos, por exemplo, limites máximos para o atraso a fim de permitir um diálogo entre duas pessoas, a necessidade de concluir certa tarefa em um tempo mínimo ou, ainda, por aplicações críticas cujo funcionamento fique comprometido caso o atraso supere certos limites.

Fornecer *QoS* não é uma tarefa trivial, mesmo em sistemas pequenos e proprietários, muito menos em um sistema global como a Internet. Para que se tenha *QoS*, faz-se necessária a cooperação entre todas as camadas da rede, assim como de todo e qualquer elemento da rede, ponto-a-ponto. Qualquer garantia de *QoS* será tão forte quanto o mais frágil elemento nessa cadeia entre o cliente e o servidor.

É importante ressaltar que o emprego de *QoS* não é capaz de aumentar a largura de banda, ou seja, a rede nunca poderá fornecer aquilo que ela não tem. O que a *QoS* permite é administrar a largura de banda existente, segundo a demanda das aplicações e dentro de certos parâmetros de gerenciamento e desempenho da rede. Uma rede habilitada para fornecer *QoS* continuará dando suporte ao tráfego de melhor esforço, contudo, parte da largura de banda será reservada para as aplicações de mais alta prioridade.

Com a preocupação de tratar as necessidades de tráfego da Internet, duas abordagens distintas de fornecimento de *QoS* foram propostas pelo *IETF - Internet Engineering*

Task Force - nos últimos anos: Arquitetura de Serviços Integrados - *IntServ* (BRADEN, CLARK *and* SHENKER 1994) e a Arquitetura de Serviços Diferenciados – *DiffServ* - (BLAKE, *et al.* 1998) (CARPENTER *and* NICHOLS 2002). Ambas as propostas possuem vantagens e desvantagens. Assim, várias pesquisas realizadas mostram que a integração de *IntServ* e *DiffServ* deverá levar à solução mais apropriada para a obtenção de *QoS*. (VASILIOU e LUTFIYYA 2000). A seguir serão discutidos alguns pontos das duas arquiteturas.

3.3 Serviços Integrados - IntServ

A abordagem de Serviços Integrados (BRADEN, CLARK *and* SHENKER 1994) foi a primeira das soluções propostas para dar suporte a *QoS* na Internet. Seu objetivo inicial era atender garantias exigidas por determinados tipos de tráfego, como transmissão de áudio e vídeo. A arquitetura *IntServ* visa a fornecer, em uma rede comutada por pacotes, como a Internet, o serviço mais próximo possível da abstração de circuitos virtuais.

A idéia principal do *IntServ* é a de reserva de recursos. Antes de iniciar a transmissão dos dados, as aplicações precisam encontrar um caminho até o receptor que satisfaça suas demandas de *QoS*, reservando, ao longo do mesmo, os recursos necessários. Para tanto, as aplicações fazem uso do protocolo *RSVP - Resource Reservation Protocol-*, um protocolo de controle e sinalização que atua na camada de rede, sendo responsável por reservar caminhos e recursos na sub-rede de comunicação.

A arquitetura de serviços integrados define duas classes de serviços, além do modelo atual de melhor esforço. São elas:

- **Serviço Garantido** (SHENKER, PARTRIDGE *and* GUERIN 1997).
  Fornece um limite superior rígido para o atraso fim-a-fim, além de garantir a disponibilidade de largura da banda. Esse serviço destina-se a aplicações que têm requisitos estritos de tempo real, atingindo um alto nível de *QoS* na Internet.

- **Serviço de Carga Controlada** (WROCLAWSKI 1997).
  Fornece um serviço equivalente ao modelo de melhor esforço em uma rede pouco utilizada, com quase nenhuma perda ou atraso. Em situações de

sobrecarga, esta abordagem será capaz de compartilhar a largura de banda entre múltiplos fluxos, de uma maneira controlada, garantindo um serviço melhor que o usual. Entretanto, é um modelo que não oferece garantias de atraso máximo, apenas um limiar probabilístico, assim como também não pode assegurar que os pacotes não serão perdidos.

O Modelo de Serviços Integrados é implementado por quatro componentes (VIANA, *et al.* 2000):

- protocolo de sinalização (por exemplo *RSVP - Resource Reservation Protocol* (BRADEN, *et al.* 1997) (WHITE 1997) ;
- rotina de controle de admissão;
- classificador;
- escalonador de pacotes.

Aplicações exigindo serviço Garantido ou serviço de Carga Controlada devem configurar caminhos e reservar recursos antes de transmitir seus dados. As rotinas de controle de admissão decidirão se uma requisição por recursos pode ser garantida. Quando um roteador recebe um pacote, o classificador realizará uma classificação *Multi-Field* e colocará o pacote em uma fila específica baseada no resultado da classificação. Então, o escalonador de pacotes escalonará os pacotes de forma a satisfazer suas exigências de *QoS*.

Os principais problemas com a Arquitetura de Serviços Integrados são:

- A quantidade de informação de estado aumenta proporcionalmente com o número de fluxos. Isto exige um enorme espaço de armazenamento e gera sobrecarga de processamento nos roteadores. Por essa razão, esta arquitetura não é escalável para a Internet.
- As exigências nos roteadores são altas. Todos os roteadores devem implementar *RSVP*, controle de admissão, classificação *MF* e escalonamento de pacotes.

3.4 Serviços Diferenciados - DiffServ

Os problemas de escalabilidade da arquitetura *IntServ* e a dificuldade de sua implantação em uma rede com as proporções da Internet motivaram os esforços para o desenvolvimento da arquitetura de Serviços Diferenciados (BLAKE, *et al.* 1998) (CARPENTER *and* NICHOLS 2002).

Enquanto o *IntServ* realiza as reservas de recursos por fluxo, exigindo dos roteadores a manutenção de informações de estado para cada fluxo, fim-a-fim, a abordagem *DiffServ* baseia-se na idéia de agregação de fluxos em umas poucas classes de serviço. Dessa forma, o *DiffServ* fornece diferenciação de serviços local para grandes agregados de tráfego, enquanto o modelo *IntServ* dá garantias de desempenho fim-a-fim para fluxos individuais.

A arquitetura de serviços diferenciados usa a classificação de pacotes como mecanismo para atingir a *QoS*. Para tanto, é redefinido o layout do octeto *Type-of-Service* do cabeçalho do protocolo *IP*v4, ou o campo *Traffic Class* do *IP*v6, que passa a ser chamado de campo *DS - Differentiated Services*. Até o momento, somente os seis primeiros bits do campo *DS* são usados, recebendo a denominação de campo *DSCP - Differentiated Services CodePoint.* Seu objetivo é especificar o tratamento dado ao encaminhamento de pacotes em cada roteador, o chamado *PHB - Per-Hop Behavior*. Os *PHBs* são mecanismos de priorização que permitem a agregação de fluxos gerados por diferentes aplicações, definindo uma classe de serviço.

O modelo *DiffServ* funciona através da marcação de pacotes de forma distinta, o que permite que os mesmos sejam tratados internamente à rede de maneira diferenciada, segundo a classe de serviço à qual pertencem. A abordagem empregada pelo *DiffServ* baseia-se em um esquema de prioridades relativas, ou seja, ele garante que o tráfego gerado por uma aplicação, com um certo nível de prioridade, receberá um tratamento melhor que o gerado por qualquer outra que possua uma prioridade inferior (XIAO *and* NI 1999).

A marcação dos pacotes ocorre nos pontos de ingresso na rede, como os *hosts* finais e os roteadores de borda. Dessa forma, retém-se um dos princípios básicos do projeto da Internet, que é colocar a complexidade na fronteira da rede, ao contrário da abordagem *IntServ*, que exige complexidade fim-a-fim. O modelo *DiffServ* não necessita de um protocolo próprio, pois se utiliza um campo do próprio datagrama *IP*. Tudo que um roteador precisa fazer é examinar o campo *DSCP* de cada pacote para determinar qual o tratamento a ser dado ao mesmo. Assim, pacotes marcados da mesma forma recebem tratamento igual. Não é necessário manter nenhum tipo de informação relacionada a fluxos, pois os roteadores só precisam ser capazes de distinguir entre um certo número de classes de serviço pré-definidas.

Existem duas classes de serviços principais definidas na arquitetura de serviços diferenciados:

- **Encaminhamento Expresso** (JACOBSON, NICHOLS *and* PODURI 1999)
  Permite a adaptação do modelo de serviço garantido da arquitetura *IntServ* à arquitetura de serviços diferenciados. Oferece garantia de *QoS* absoluta, com baixos valores de perda, atraso e *jitter*, fornecendo o equivalente a uma linha privada virtual com largura de banda fixa entre dois *hosts*.

- **Encaminhamento Garantido** (HEINANEN, *et al.* 1999).
  Destina-se a aplicações que demandem da rede um serviço mais confiável que aquele de melhor esforço, mas sem todas as garantias de *QoS* dadas pelo encaminhamento expresso. Este serviço não oferece limites superiores para o atraso e *jitter*, mas garante um tratamento preferencial ao tráfego que dele se utilize.

3.5 Comparação IntServ e DiffServ

As arquiteturas de serviços integrados e diferenciados não devem ser consideradas como competidoras, ao contrário, elas são complementares e podem ser usadas em conjunto, explorando-se as melhores características de uma e de outra.

Entre as diferenças das duas arquiteturas podem-se destacar:

1. O nível de granulosidade em que cada mecanismo trabalha: o modelo *IntServ* tem seu foco em fluxos individuais fim-a-fim, entre uma origem e um destino; já o modelo *DiffServ* lida com o conceito de agregação de fluxos, os quais constituem as classes de serviço.

2. A arquitetura *IntServ* representa uma mudança com relação à abordagem tradicional empregada nas redes *IP*, pois ela leva a complexidade para o núcleo da rede; já a arquitetura *DiffServ* é mais natural, pois coloca a complexidade nos pontos externos da rede e mantém os roteadores simples, preservando as diretrizes do projeto original da Internet.

3. O *IntServ* necessita de um protocolo de sinalização como o *RSVP* para funcionar, o qual precisa incluir informações de estado nos roteadores ao longo do caminho; o *DiffServ* apenas faz uma redefinição de um campo do cabeçalho *IP*, o que agiliza sua implantação.

4. A classificação dos pacotes no *IntServ* é feita a partir de vários campos, como os endereços de origem e destino, números de portas e o protocolo utilizado, individualmente para cada fluxo; já o *DiffServ* usa apenas o campo DS do protocolo *IP* para este fim.

5. O modelo *IntServ* está mais voltado para as demandas de serviço de cada aplicação, procurando satisfazê-las fim-a-fim e, por causa disso, exige uma reserva de recursos antes do início de qualquer transmissão de dados. O modelo *DiffServ* visualiza mais as propriedades do tráfego gerado, usando um mecanismo de priorização para classificá-lo segundo suas características de demanda de *QoS*. É responsabilidade das aplicações, ou de algum roteador de borda, marcar seus pacotes adequadamente a fim de que recebam o serviço de que precisam.

3.6 Considerações Finais

O modelo original da Internet não previa a dimensão e a estrutura que é apresentada hoje em dia. O modelo de serviço do tipo *best-effort* vem apresentando sinais de estrangulamento há algum tempo, e já não é o mais indicado para determinadas aplicações que surgiram na *Web*. Considerando-se uma rede com pouco tráfego e aplicações não muito exigentes, esse modelo se mostra satisfatório. Porém, ocorrem problemas quando a rede está sobrecarregada ou as aplicações possuem requisitos para os quais ela não foi projetada.

Fornecer *QoS* não é uma tarefa trivial, mesmo em sistemas pequenos e proprietários, muito menos em um sistema global como a Internet. Para que se tenha *QoS*, faz-se necessária a cooperação entre todas as camadas da rede, assim como de todo e qualquer elemento da rede, ponto-a-ponto. Este capítulo apresentou as abordagens de fornecimento de *QoS* que foram propostas pelo IETF, descrevendo as arquiteturas de serviços integrados e diferenciados, com ênfase nesta última.

As arquiteturas de serviços integrados e diferenciados não devem ser consideradas como competidoras, ao contrário, elas são complementares e podem ser usadas em conjunto, explorando-se as melhores características de uma e de outra. Neste trabalho foi utilizada a arquitetura DiffServ explorando suas principais características de agregação de fluxos de dados em classes de serviço e deixando a complexidade de implementação nos pontos de saída da rede, preservando com isso as diretrizes do projeto original da Internet.

Adicionalmente, para melhorar o desempenho das aplicações *Web* e evitar que a rede fique sobrecarregada, outras técnicas vêm sendo utilizadas, tais como o uso de servidores *cache*. No próximo capítulo serão apresentados servidores *cache* para *Web* como uma alternativa para melhorar o desempenho dos sistemas. O objetivo principal dos servidores *cache* na *Web* é armazenar documentos utilizados com freqüência,

visando ao uso futuro, de modo que não seja necessário fazer novamente a recuperação dos dados no servidor origem, quando esses forem solicitados.

Capítulo

4

Cache para Web

4.1 Considerações Iniciais

Desde a introdução da *Web* o tráfego de informações vem crescendo a cada dia e a sobrecarga da Internet pode ser percebida pelo alto tempo gasto na recuperação de informações, bem como na degradação do desempenho do sistema em virtude do tráfego redundante. Para melhorar o desempenho dos sistemas, diferentes técnicas vêm sendo utilizadas, tais como o uso de servidores *cache*. O objetivo principal dos servidores *cache* na *Web* é armazenar documentos utilizados com freqüência para uso futuro, de modo que não seja necessário fazer novamente a recuperação dos dados no servidor origem, quando esses forem solicitados. O servidor *cache* reduz o tempo de acesso aos dados fazendo que esses estejam o mais perto possível dos clientes que os estão requisitando.

A utilização de servidores *cache* na *Web* é um dos métodos mais populares para tentar melhorar o desempenho do ambiente. Em geral, a taxa de acerto de um servidor *cache* simples não é maior que 40% (ZOU, MARTIN *and* HASSANEIN 2003) (ABRAMS, *et al.* 1995), mas representa uma melhora notável no desempenho observado pelo usuário.

Vários trabalhos vêm sendo desenvolvidos visando a permitir uma melhor integração de servidores *cache* em ambientes *Web* (ZHOU, HASSANEIN *and* MARTIN 2004) (CHEN, MARTIN and HASSANEIN 2004). Alguns estudos vêm sendo realizados com a introdução de novas metodologias, que permitem o armazenamento de conteúdos dinâmicos em servidores *cache* (ZHU e YANG 2001) (HOSANAGAR, *et al.* 2005) (CHEN, MARTIN e HASSANEIN 2003), o que pode possibilitar melhorias consideráveis no desempenho observado por um usuário.

Outros estudos destacam-se por inserir mecanismos de diferenciação de serviços nos servidores *cache* (ZHOU, HASSANEIN *and* MARTIN 2004) (YING, ABDELZAHER *and* SAXENA 2004). Servidores *cache* tradicionais tratam todas as requisições de clientes da mesma forma, independentemente de existirem requisições com maior prioridade sobre as outras. Tal fato pode afetar outros mecanismos presentes no sistema para melhorar o desempenho, como a utilização de diferenciação de serviços na rede.

Neste capítulo são descritas considerações sobre *cache* na *Web* apresentando as principais diferenças em relação aos *cache* tradicionais e a utilização de *cache* na *Web*. Ainda, é descrito o servidor *cache* com suporte à diferenciação de serviços, denominado de *cache* CDF, implementado neste trabalho com o objetivo de avaliar o impacto da diferenciação de serviços dos servidores *cache* no ambiente *Web*.

4.2 Diferenças entre Cache na Web e Caches Tradicionais

Técnicas de *cache* na *Web* são diretamente influenciadas por estudos dos *caches* tradicionais, uma vez que ambos mecanismos utilizam o mesmo princípio de funcionamento: objetos requisitados são copiados em posições mais próximas do usuário, com o propósito de diminuir o tempo do próximo acesso ao objeto. Embora os conceitos sejam similares, há diferenças fundamentais entre *cache Web* e *caches* tradicionais (PINHEIRO 2001):

- **Tamanho dos objetos.**

  Os *caches* tradicionais transferem objetos de tamanhos fixos, o número de *bytes* encontrados no *cache* em relação ao número de *bytes* requisitados é exatamente a taxa de acertos. Por outro lado, *cache* na *Web* envolvem objetos de tamanhos diversos, variando desde *bytes* até *megabytes*. Isso pode implicar em que o armazenamento de apenas um objeto grande no *cache* pode gerar a retirada de dezenas, centenas ou milhares de objetos pequenos, alterando completamente o seu estado. Em *cache Web*, os objetos são transmitidos e armazenados na sua forma integral, não havendo o conceito de bloco.

- **Latência.**

Enquanto que em *caches* tradicionais o tempo de espera por uma requisição (latência) é pequeno, os *cache Web* podem ter diferentes tempos de acesso para objetos de mesmo tamanho, cujos valores podem estar em escalas de tempo muito maiores. Essa diferença é influenciada por diversos fatores, tais como: largura de banda disponível, sobrecarga na rede, no servidor e sua localização em relação ao cliente.

- **Operação de escrita.**

Não há em *cache* na *Web* operações de escrita, como em *caches* tradicionais; o sistema de *cache* na *Web* é basicamente para leitura.

4.3 Utilização de Cache na Web

Vários são os benefícios que justificam a utilização de *cache* na *Web*, dentre eles podem-se destacar (PINHEIRO 2001):

- **Redução de tráfego.**

  Um menor número de requisições e respostas precisa trafegar na rede. Um documento é recuperado do servidor somente uma vez e armazenado em *cache*, reduzindo a banda de rede usada pelo cliente. Por meio da utilização de *cache*, pode-se aumentar a escalabilidade do sistema, reduzir a possibilidade de congestionamento e maximizar a utilização da infra-estrutura da Internet, minimizando a quantidade de tráfego redundante.

- **Redução na carga de servidores**.

  Como muitos documentos serão replicados e armazenados em *caches* haverá um menor número de requisições diretas para os servidores.

- **Redução de latência**.
  As respostas de requisições a objetos que podem ser armazenados no *cache* são feitas a partir do *cache* local, e não pelo servidor original. Dessa maneira, o acesso tende a ser bastante rápido.

- **Possibilidade de acesso *off-line***.
  Considerando-se que o servidor de um endereço especificado está inacessível, ou recebendo mais solicitações do que pode atender adequadamente, se o objeto estiver armazenado no *cache* será possível recuperá-lo, porém não é possível sua atualização.

Embora o *cache* tenha muitas vantagens ele introduz um novo conjunto de problemas, que muitas vezes são difíceis de serem resolvidos, tais como:

- **Manter a consistência dos documentos**.
  Documentos podem não estar atualizados no *cache*; não é fácil prever o tempo em que um documento permanecerá válido.

- **Documento requisitado não se encontra no *cache***.
  Caso ocorram muitas falhas, o tempo para a recuperação de um documento aumenta devido ao tempo adicional para verificar se ele encontra-se no *cache*. Para evitar esse problema, a taxa de acerto deve ser maximizada e o custo de uma falha deve ser minimizado durante o projeto de sistemas de *cache*.

- **Gargalos no sistema**.
  O *cache* pode se tornar um gargalo no sistema.

- **Armazenamento de Páginas Dinâmicas:**
  Servidores *caches* tradicionais armazenam somente páginas estáticas. Basicamente, na *Web*, disponibilizam-se dois tipos de conteúdos: estáticos e dinâmicos. Conteúdos estáticos são informações armazenadas em arquivos nos servidores; os conteúdos dinâmicos são informações geradas por programas em

tempo real nos servidores. Exemplos de conteúdos dinâmicos são resultados de pesquisas, páginas de notícias, páginas personalizadas, dentre outras. Servidores *cache* tradicionais armazenam somente conteúdos estáticos, não conseguindo armazenar conteúdos dinâmicos. Por outro lado, alguns estudos foram e ainda vêm sendo realizados com a introdução de novas metodologias, que permitem o armazenamento de conteúdos dinâmicos em servidores *cache* (CHEN, MARTIN and HASSANEIN 2004) (ZHU e YANG 2001) (HOSANAGAR, *et al.* 2005) (CHEN, MARTIN e HASSANEIN 2003), o que pode possibilitar melhorias consideráveis no desempenho observado por um usuário.

O *cache* na *Web* pode ser implementado de três maneiras:

- ***Cache* de cliente**.

  Os navegadores mantêm pequenos *cache* de páginas visitadas recentemente no disco local dos usuários. Esse recurso é muito útil, principalmente quando os usuários "clicam" no botão voltar ou retornam para uma página visitada recentemente. O *cache* de cliente possui a vantagem de evitar o acesso à rede.

- ***Cache* de servidor**.

  Implementado na memória principal dos servidores *Web*, tem como objetivo principal deixar os documentos mais acessados em memória principal, tornando o acesso mais rápido;

- ***Cache* de *Proxy***.

  Implementado em pontos estratégicos da rede, os servidores de *cache* ficam localizados entre clientes e servidores como ilustrado na Figura 4.1. Quando um documento é solicitado por um cliente, esse documento é solicitado ao *Proxy*, esse por sua vez solicita o documento ao servidor remoto, retornando o pedido ao cliente. Os próximos pedidos ao mesmo documento a partir de qualquer cliente atendido pelo *Proxy* são atendidos pela cópia armazenada em *cache*, otimizando o fluxo de tráfego e reduzindo o consumo de banda da rede.

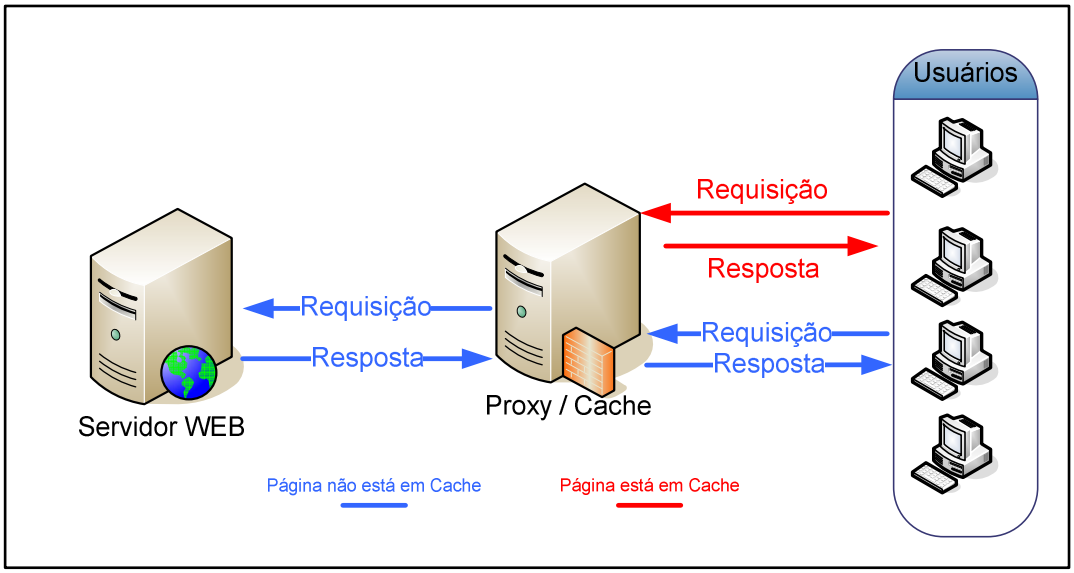


Figura 4.1 - Exemplo de um *Cache* de *Proxy*

A implementação do servidor *Cache/Proxy* pode ser realizada de várias maneiras. Neste trabalho, por exemplo, foi utilizado um servidor *cache* transparente. Esse tipo de implementação utiliza componentes da rede para redirecionar as requisições para os servidores *cache*, normalmente *switches* de camada 4 e 5. Essa técnica é chamada de transparente, pois os clientes (*Web Browsers*) não necessitam realizar configurações explicitas do ponto de acesso ao servidor *cache*. *Switches* de camada 5 (L5) permitem o redirecionamento das requisições clientes através de informações nos pacotes *TCP* e/ou nos cabeçalhos de requisições *HTTP* (LIANG, HASSANEIN *and* MARTIN 2001) (HASSANEIN, LIANG *and* MARTIN 2002) (ZOU, MARTIN *and* HASSANEIN 2003).

A eficiência do servidor *cache Web* pode ser medida por três métricas (MENASCÉ *and* ALMEIDA 2002):

- **Taxa de acerto**.
  É definida como o número de pedidos atendidos pelo *cache* sobre o número total de pedidos.

- **Taxa de acerto em *bytes***.
  Igual à taxa de acerto ponderada pelo tamanho do documento.

- **Taxa de dados transferidos**

  Representa o número total de *bytes* transferidos entre o *cache* e o mundo exterior, durante um período *T* de tempo.

Quando as políticas de substituição do *cache Web* favorecem objetos menores, a taxa de acerto é maximizada, pois, nesse caso, mais objetos poderão ser armazenados. Por outro lado, se objetos maiores forem armazenados no *cache*, aumenta a probabilidade de se ter uma economia do tempo de transferência e uma maior taxa de acertos em *bytes*.

Embora o *cache* tenha muitas vantagens ele introduz um novo conjunto de problemas que muitas vezes são difíceis de serem contornados, tais como:

- como manter um documento em *cache* e garantir que ele esteja atualizado;
- quais documentos irão ficar armazenados em *cache*;
- quanto tempo um documento ficará armazenado no *cache*;
- qual a estratégia de substituição de documentos em *cache* será adotada;
- como armazenar objetos dinâmicos em *cache*.

4.4 Servidor Cache Com Diferenciação de Serviços

Servidores *cache* tradicionais tratam todas as requisições de clientes da mesma forma, independentemente de existirem requisições com maior prioridade sobre as outras. Tal fato pode afetar outros mecanismos presentes no sistema para melhorar o desempenho, como a utilização de diferenciação de serviços na rede. Alguns trabalhos vêm sendo desenvolvidos, visando a permitir uma melhor integração de servidores *cache* em ambientes com a presença de diferenciação de serviços (ZHOU, HASSANEIN *and* MARTIN 2004) (YING, ABDELZAHER *and* SAXENA 2004).

Servidores *cache* tradicionais realizam todas as requisições ao servidor *Web* com tipo de serviço *best-effort* (classe padrão), independente da requisição inicial do usuário possuir maior prioridade (Figura 4.2). Com o objetivo de não se perder a prioridade requisitada pelos clientes, foi implementado neste trabalho um servidor *cache* com suporte à

diferenciação de serviços denominado de *cache* CDF. Toda a requisição do servidor *cache* CDF para o servidor *Web* é realizada com a mesma prioridade que o cliente solicitou originalmente, Figura 4.3.

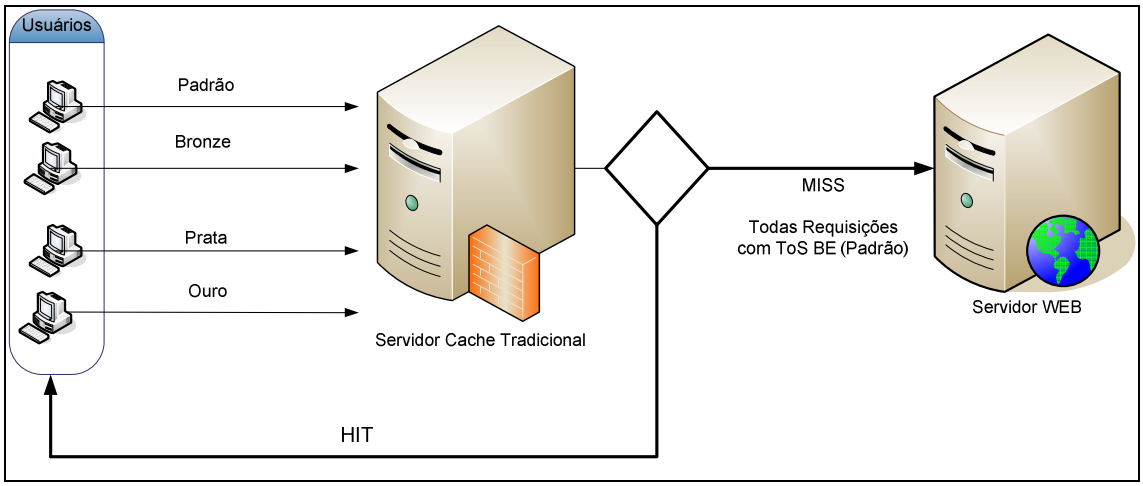


Figura 4.2 – Descrição do Funcionamento do Servidor *Cache* Tradicionais

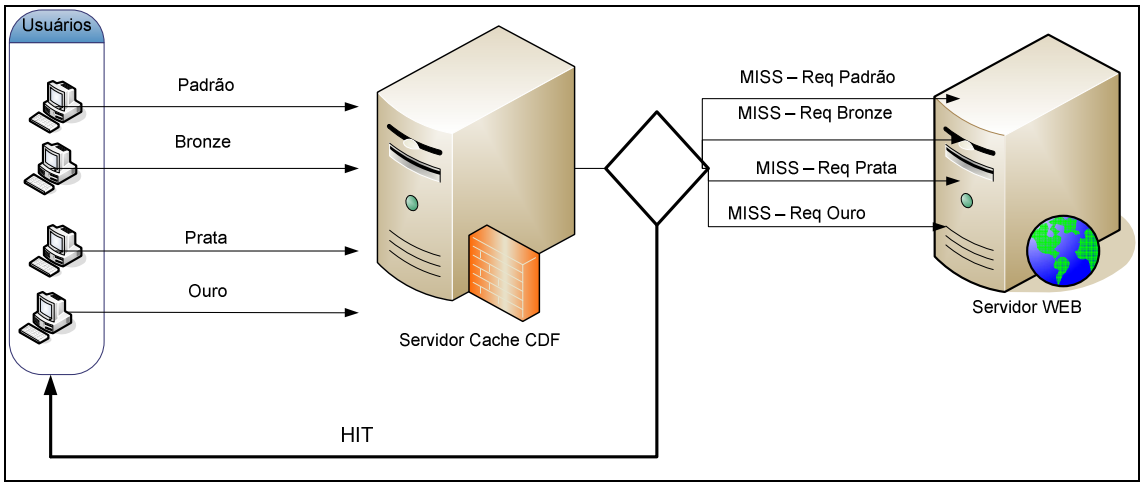


Figura 4.3 – Descrição do Funcionamento do Servidor *Cache* CDF

Para os objetivos deste trabalho, o *cache* CDF foi implementado no *OPNET Modeler* (*OPNET Modeler* 2008) a partir do modelo de um servidor *cache* tradicional disponível. Os modelos de equipamentos disponíveis no *OPNET Modeler* possuem o comportamento validado por seus fabricantes. No caso do servidor *cache* foi utilizado um modelo genérico chamado *ethernet_cache_server_adv*. Esse modelo representa um nó servidor com as aplicações sendo executadas sobre os protocolos *TCP*/IP e *UDP*/IP com suporte *cache Web*. O servidor *cache* opera de forma transparente aos usuários.

Todos os modelos de equipamentos do *OPNET Modeler* são descritos através de um diagrama de fluxo de dados entre elementos funcionais, denominados módulos. A Figura 4.4 ilustra o diagrama de fluxo de dados entre todos os elementos funcionais do servidor *cache* tradicional. Para a implementação do servidor *cache* CDF foi necessária a alteração da programação do elemento *application* do servidor *cache* tradicional. O elemento *application* é o responsável pelo processamento de todas as requisições que chegam ao servidor *cache*.

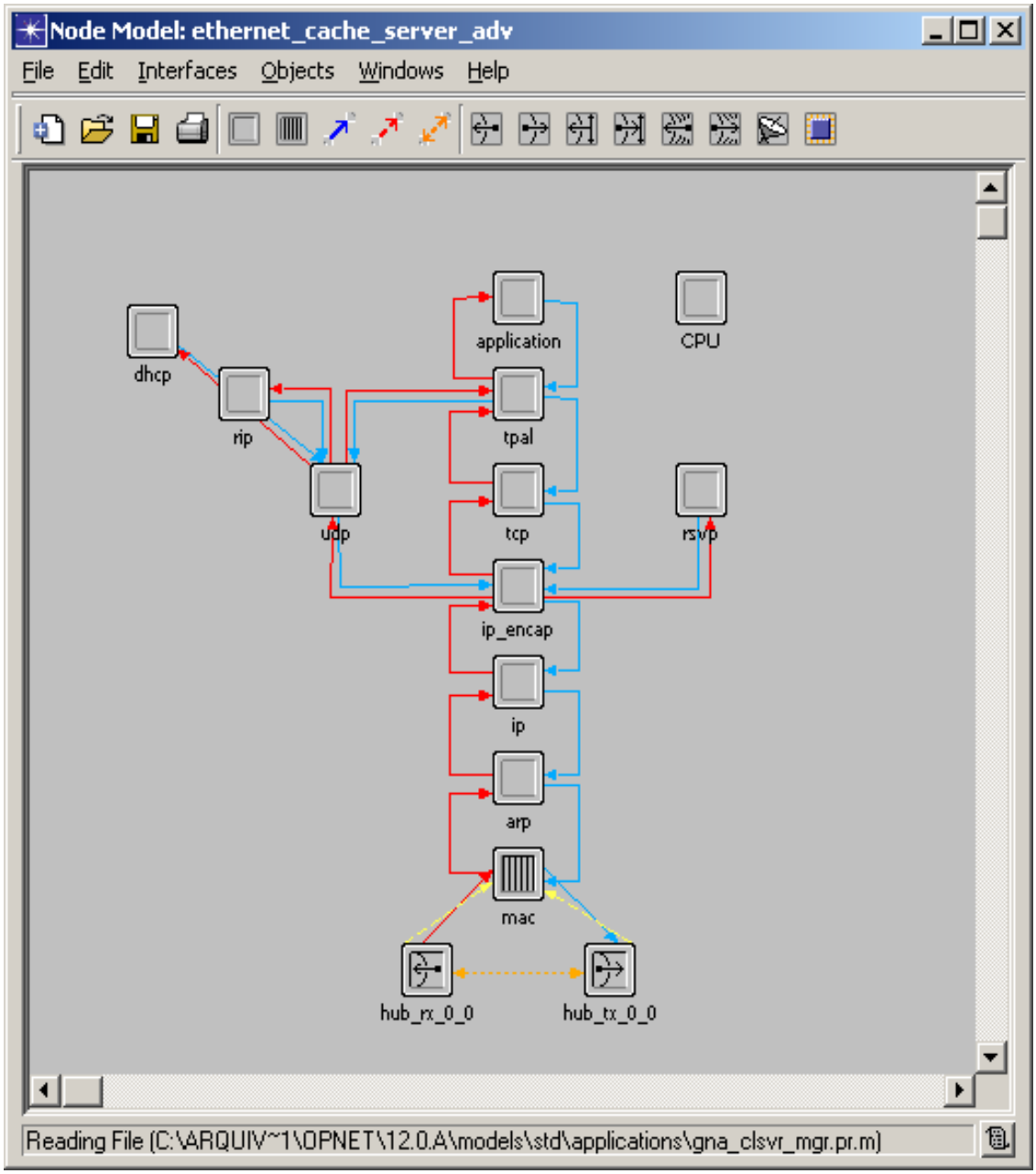


Figura 4.4 – Diagrama de fluxo de dados entre elementos do Servidor *Cache*

O elemento *application* é descrito através de uma Máquina de Estados Finitos (MEF). Essa MEF é formada por vários estados ilustrados na Figura 4.5. Cada estado da MEF possui um código em C que é executado quando o estado é ativado. Na MEF do elemento *application* foi implementado um classificador responsável por analisar qual a prioridade da requisição do usuário e realizar o pedido ao servidor *Web* com a mesma prioridade. O Classificador somente é executado se o pedido do usuário não estiver presente no *cache* CDF. O Anexo D ilustra parte do código C alterado.

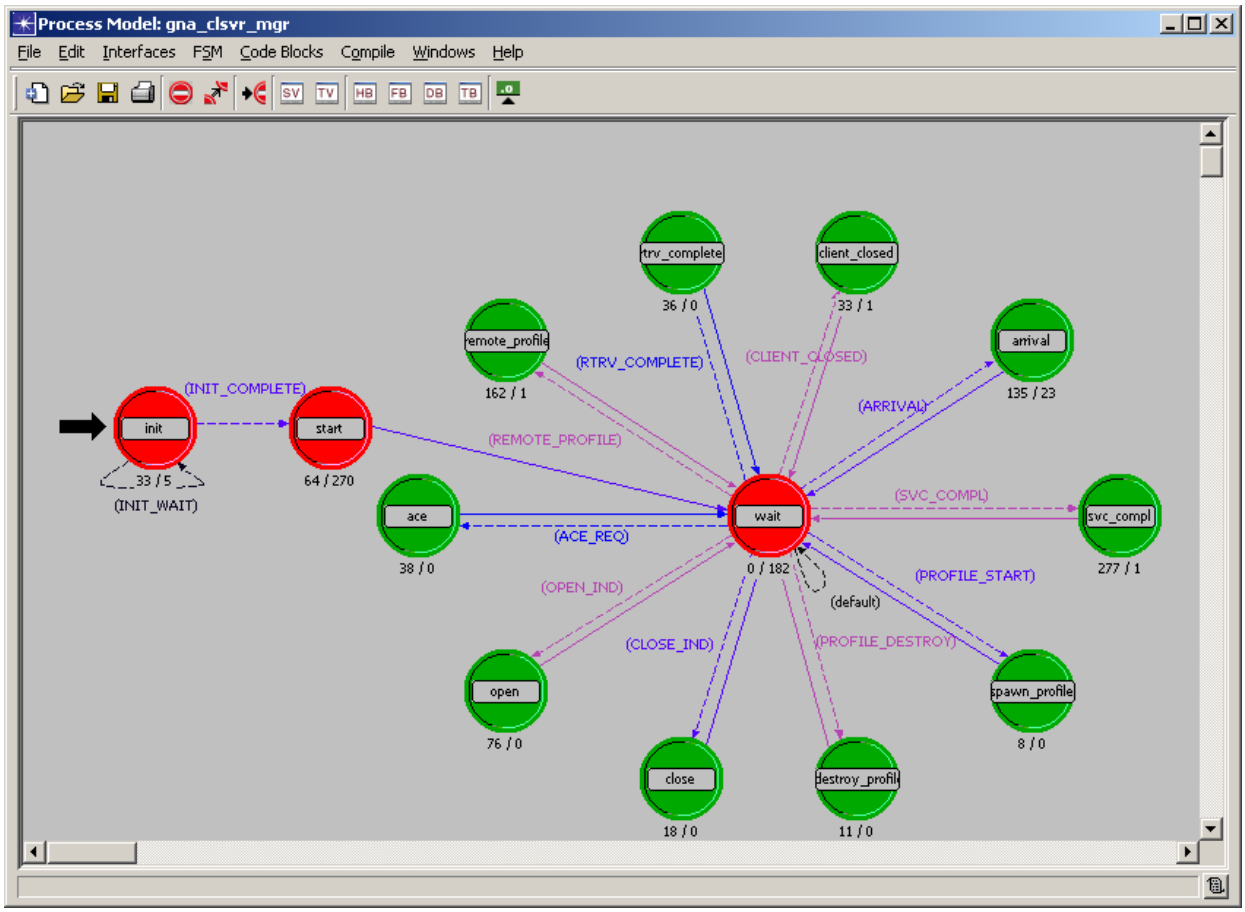


Figura 4.5 – Máquina de estado finito do módulo application do Servidor *Cache*

4.5 Considerações Finais

Neste capítulo foram apresentadas considerações sobre a utilização de *cache* na *Web*. Destacou-se que o tráfego da Internet tem crescido com vigor nestes últimos anos, gerando situações claras de congestionamento. O uso de servidores *cache* para armazenar objetos pode aliviar não só a carga nos diversos *links* da rede, mas também diminuir o tempo de acesso aos objetos remotos. Servidores *cache* tradicionais são mecanismos muito utilizados para minimizar problemas de desempenho em redes, pois sua implementação geralmente é simples, o custo não é elevado e apresentam resultados rapidamente. Porém, servidores *cache* tradicionais não são preparados para realizarem tarefas de diferenciação de serviços entre classes de usuários, prejudicando os esforços de priorização de serviços executados em outros pontos do sistema.

Para solucionar o problema do não tratamento diferenciado das requisições dos clientes pelos servidores *cache* tradicionais, foi proposto um servidor *cache* com suporte à diferenciação de serviços denominado *cache* CDF.

O *cache* CDF foi modelado no *OPNET Modeler* a partir de um servidor *cache* tradicional e toda a requisição do servidor *cache* CDF para o servidor *Web* é realizada com a mesma prioridade que o cliente solicitou originalmente. Nesse, sentido o servidor *cache* CDF continuará realizando a tarefa de diferenciação de serviços sem prejudicar os esforços de *QoS*.

No próximo capítulo serão apresentadas técnicas para avaliar o desempenho de sistemas computacionais. Serão discutidas as técnicas de aferição e de modelagem, bem como os métodos para a solução dos modelos e ambientes computacionais utilizados para simulação, com destaque para o *OPNET Modeler*, que foi o ambiente de modelagem e simulação utilizado neste trabalho.

Capítulo

5

Avaliação de Desempenho: Modelagem e Simulação

5.1 Considerações Iniciais

Quando se pretende avaliar o desempenho de um sistema, o primeiro desafio está em definir a técnica mais adequada para essa avaliação. O domínio da aplicação é um dos fatores que pode determinar a escolha de uma técnica em detrimento das demais. Os estudos do desempenho de ambientes computacionais são tarefas complexas e trabalhosas, principalmente quando se trata de ambientes distribuídos que oferecem serviços *Web*. As técnicas utilizadas podem, de modo geral, ser agrupadas em (SANTANA 1990a) (SANTANA 1990b) (SANTANA, *et al.* 2004):

- Técnicas de modelagem e
- Técnicas de aferição.

Esses dois grupos de técnicas são amplamente utilizados como suporte ao estudo de avaliação de desempenho em diversas áreas, quer seja para o levantamento de dados para a avaliação de um sistema, como também, para fornecer dados para a construção e/ou parametrização de modelos que representem determinados ambientes. As técnicas de aferição mais utilizadas são coletas de dados, construção de protótipos e utilização de *benchmarcks*. Em relação às técnicas de modelagem são utilizadas Redes de Filas, Redes de P*etri*, S*tatechart*, entre outras (FRANCÊS, *et al.* 1997). A Figura 5.1 ilustra as técnicas de avaliação de desempenho.

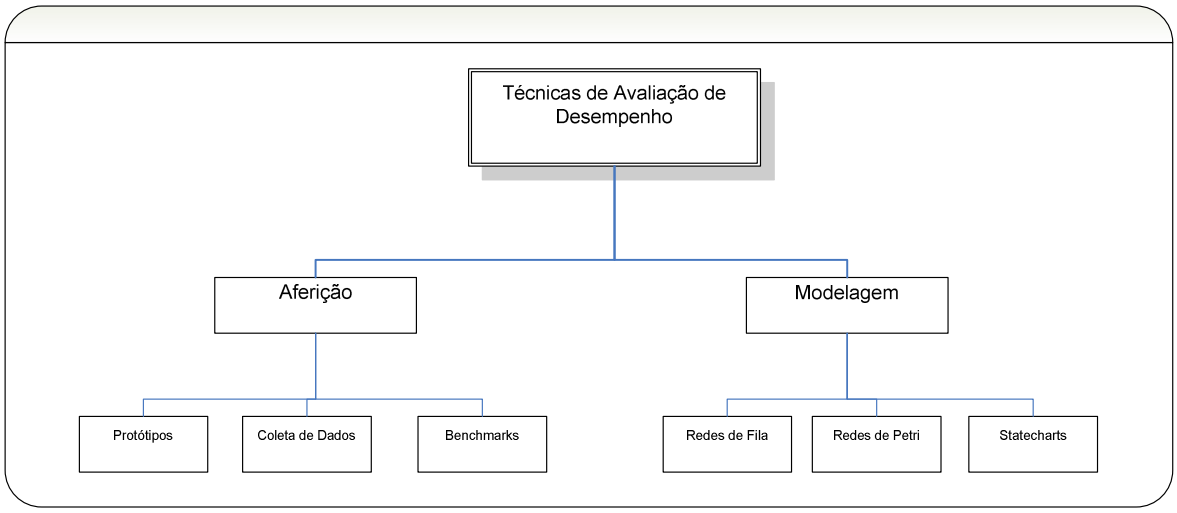


Figura 5.1 – Técnicas de Avaliação de Desempenho

Neste capítulo são apresentadas técnicas de aferição, técnicas de modelagem de sistemas, métodos para a solução de modelos e ambientes computacionais utilizados para simulação, com destaque para o *OPNET Modeler*.

5.2 Técnicas de Aferição

Técnicas de Aferição são aplicadas principalmente na avaliação de sistemas já existentes ou em fase final de desenvolvimento (FRANCÊS, *et al.* 1997). No caso da aferição de um sistema utilizam-se monitores de *hardware*, *software* ou híbridos, que permitem a coleta de dados reais a partir do funcionamento do sistema, fornecendo informações sobre o comportamento do ambiente em termos de desempenho, por exemplo.

No caso das técnicas de aferição, um dos grandes desafios é a atenção necessária para se evitar perturbações no sistema sob avaliação, de modo a minimizar as interferências no resultado observado. Técnicas de aferição são também necessárias quando se deseja monitorar um sistema, durante sua operação diária, visando, por exemplo, a melhorar os aspectos de escalonamento de processos.

As técnicas de aferição podem ser divididas em:

- ***Benchmarks*** (WEICKER 1990)

Programas representativos de um certo domínio de aplicação, usados para testar o desempenho de um *software*, *hardware* ou sistema computacional. A mesma tarefa ou programa é submetida a diferentes sistemas e seus resultados são comparados entre si. Deve-se tomar cuidado para que a própria execução do *benchmark* não venha a influenciar os resultados obtidos. O resultado dos *benchmarks* é utilizado para avaliar o desempenho do sistema e também como ferramentas de monitoração do ambiente.

- **Coleta de Dados**

As técnicas de aferição, em geral, são as que fornecem os resultados mais precisos; em particular, a coleta de dados é a mais precisa. Seu objetivo principal á a avaliação de desempenho através da obtenção direta de dados em um sistema já existente (FRANCÊS 1998). A coleta de dados pode ser efetuada através de monitores. A função principal de um monitor é coletar dados referentes à operação do sistema. A atividade de monitoração deve ser realizada de modo que não afete a operação do sistema em estudo. Os monitores podem ser divididos em monitores de *hardware, software e híbridos* (MENASCÉ *and* ALMEIDA 2002).

- **Protótipos**

A construção de Protótipos é, geralmente, utilizada no caso de sistemas ainda não existentes, mas que se encontram em fase final de desenvolvimento (FRANCÊS 1998). Essa técnica consiste na simplificação do sistema computacional, em que são representadas somente suas características essenciais. A grande desvantagem da construção de protótipos está relacionada aos altos custos para sua implantação

5.3 Técnicas de Modelagem

Modelar um sistema consiste na construção de um modelo (uma representação simbólica, literal ou numérica), segundo algumas técnicas de construção de modelos, que abstrai as características a serem analisadas no sistema real. Uma das tarefas mais complexas no processo de modelagem é o levantamento de quais elementos do sistema serão incluídos no modelo. Assim, ao modelar um sistema, deve-se procurar abstrair quais elementos são importantes na resolução de uma questão particular e quais relações entre esses elementos são necessárias à resolução do modelo. Dessa forma, pode-se ter vários modelos de um mesmo sistema, cada um com características e especificações mais adequadas à resolução de um problema específico.

Os modelos podem ser classificados de diferentes formas de acordo com algumas características específicas (PEGDEN, SHANNON *and* SADOWSKI 1995) (BRUSCHI 2002):

- **Detalhamento do Modelo**.
  Uma possível classificação pode ser com relação à quantidade de detalhamento inserido na modelagem. Existem os que são uma réplica precisa do sistema e os que podem ter suas principais características abstraídas. Nesse caso, a inclusão de muitos detalhes pode introduzir complicações desnecessárias, enquanto que a exclusão de características importantes pode invalidar o modelo.

- **Utilização ou não de variáveis aleatórias**.
  Poucos sistemas podem ser considerados totalmente livres da influência da aleatoriedade. Os modelos que representam esses sistemas são conhecidos como modelos determinísticos. Os sistemas que consideram a aleatoriedade são classificados como modelos estocásticos.

- **Classificação temporal**.

  Com relação ao tempo podem-se dividir os modelos em dinâmicos e estáticos. Os modelos dinâmicos descrevem o comportamento do sistema através do tempo, enquanto que os modelos estáticos simplesmente fornecem um retrato do sistema em um ponto específico do tempo. Os modelos dinâmicos podem ainda ser classificados como contínuos ou discretos no tempo. Os modelos discretos representam sistemas em que as mudanças nas variáveis de estados ocorrem em pontos específicos e descontínuos do tempo simulado, alterando o estado do sistema espontaneamente.

Um fator importante para o desenvolvimento de um modelo é a definição da abordagem a ser utilizada para a representação do modelo. No caso de sistemas computacionais a modelagem pode ser realizada através de técnicas de modelagem que utilizam:

- **Redes de Filas.**

  Técnica baseada na Teoria de Filas (JAIN 1991) criada para modelar sistemas nos quais existe a ocorrência de filas. Os componentes básicos de uma rede de filas são: servidores, filas e clientes. Os recursos modelados no sistema correspondem aos servidores, os quais prestam serviços aos clientes. A fila é uma área de espera para clientes que estão requisitando serviços e não são atendidos prontamente devido ao servidor estar ocupado. O conjunto de um ou mais servidores e uma ou mais filas é denominado centro de serviço. O conjunto de centros de serviço é denominado uma rede de filas.

- **Redes de Petri.**

  As Redes de *Petri* (PETRI 1966) foram propostas por Carl Adam Petri em 1962 como uma ferramenta matemática para modelar sistemas e, em particular, notações de concorrência, comunicação e sincronismo (MOORE *and* BRENNAN 1995). A partir da década de 70, expandiu-se também para a utilização na modelagem de componentes de *hardware*, linguagens de programação, controle de processos, protocolos de comunicação, dentre outros. Atualmente, tem-se utilizado Redes de *Petri* para modelagem de sistemas concorrentes, assíncronos, distribuídos, paralelos, não determinísticos e estocásticos, *software* distribuído, banco de dados distribuídos, controle

industrial, memória de multiprocessadores, sistemas *data-flow* e tolerante a falhas, programação lógica e VLSI, compiladores e sistemas operacionais, linguagens formais, redes locais, entre outros.

- ***Statecharts***

  Statecharts são uma extensão das máquinas de estado finito – MEF - que permitem a representação de hierarquia, concorrência e comunicação entre os diversos estados de um sistema. *Statecharts* têm a finalidade de representar sistemas que reagem a estímulos internos e externos.

  Os *statecharts* foram propostos com a finalidade de resolver alguns problemas que os diagramas de estado apresentavam em relação à representação de sistemas. Entre os problemas podem-se destacar:

  1. Os diagramas de estado não fornecem a noção de profundidade, hierarquia ou modularidade;
  2. Conforme o sistema a ser descrito cresce linearmente, o número de estados cresce exponencialmente;
  3. Os diagramas de estados não possuem nenhuma característica que permita a representação de estados concorrentes.

  Estes problemas são superados através da decomposição dos estados em subestados, decomposições *AND*/OR, combinado com um mecanismo eficiente de comunicação. Vários trabalhos vêm sendo realizados com propostas de extensões para os *Statecharts* como a inclusão de aspectos estocásticos, os *Stochastic-Statecharts* (FRANCÊS 1998) e aspectos temporais, visando melhor aproximá-los das Redes de Fila, como é o caso dos *Queing Statecharts* (FRANCÊS, *et al.* 2001).

5.4 Soluções para o Modelo

Após a escolha da técnica utilizada para a representação do modelo, deve-se decidir qual a solução a ser dada ao mesmo. As técnicas de solução disponíveis: são: a analítica, a simulação e a híbrida. As técnicas têm suas vantagens e desvantagens, que normalmente são os fatores determinantes na escolha de uma em detrimento da outra.

5.4.1 Solução Analítica

A solução por métodos analíticos consiste em substituir os componentes do ambiente por um modelo matemático, que possa ser resolvido analiticamente. Dessa forma, os problemas de avaliação de desempenho e dimensionamento são transformados em problemas matemáticos, que podem ser resolvidos através de técnicas matemáticas.

Uma solução analítica nem sempre pode ser utilizada em virtude da complexidade de alguns modelos. O número de variáveis que devem ser adicionadas no modelo pode ser grande. A fim de manter os modelos tratáveis, várias simplificações são feitas, produzindo assim cenários bastante limitados. Apesar de ser um método que propicie resultados expressivamente exatos, as simplificações introduzidas podem fazer com que o modelo não seja uma representação fiel do sistema real, fato que pode simplesmente tornar sem sentido a utilização do modelo.

5.4.2 Solução por Simulação

Caso o modelo proposto para o sistema envolva uma grande quantidade de informações e exija um detalhamento das operações sem que seja possível realizar simplificações no modelo, a resolução através de simulação é mais propícia que a resolução analítica.

A simulação é atualmente uma das ferramentas disponíveis para avaliação de desempenho de sistemas computacionais e se tornou atrativa e atuante nas mais diversas áreas de aplicação devido a:

- Versatilidade: pode ser utilizada em diferentes ocasiões;
- Flexibilidade: é adaptável a novas e diferentes situações;
- Baixo custo: com um mesmo programa podem-se simular diferentes situações.

A simulação se destaca no planejamento e desenvolvimento de projetos de ambientes computacionais distribuídos, devendo-se avaliar e checar vários pontos básicos visando a evitar a ocorrência de falhas e problemas futuros. Com base nesse pensamento é de clara compreensão que uma simulação que abrange todos os aspectos a serem analisados, questionados e discutidos, seja de grande ajuda no desenvolvimento do ambiente.

Evidentemente, a validade dos resultados da simulação depende da existência de um modelo que represente o sistema real de maneira correta e adequada. Para isto, a modelagem deve ser baseada no conhecimento detalhado de informações do domínio do sistema a ser modelado. O uso de ferramentas de simulação e o levantamento minucioso das características do ambiente são fatores básicos para a obtenção de resultados relevantes.

O processo de desenvolvimento de uma simulação envolve diversas etapas. Primeiro deve-se especificar o modelo, abstraindo as características mais importantes do sistema. Tendo o sistema sido especificado e modelado, é necessário transformar o modelo em um programa de simulação. Em simulações estocásticas, devido à aleatoriedade dos dados de entrada e do comportamento dos elementos do sistema, os dados gerados pelo

programa de simulação devem receber um tratamento estatístico adequado, de modo a se ter precisão adequada e significância estatística para as conclusões que serão formuladas. Uma possível técnica é ter-se o programa de simulação executado diversas vezes a fim de garantir que a influência da aleatoriedade nos resultados finais esteja dentro dos limites requeridos.

A modelagem e simulação de sistemas computacionais, em geral, podem ser divididas nas seguintes etapas (CHANG 1999) (BRUSCHI 2002):

- **Definição do Problema**.
  O levantamento de equipamentos e serviços disponíveis no sistema é realizado nesta etapa, bem como a definição de quais os problemas a serem analisados;

- **Construção dos Modelos**.
  Uma vez que o sistema a ser simulado está bem definido é necessário abstraí-lo em um modelo, o qual irá representar uma visão particular do sistema. Um modelo deve ser de fácil compreensão, sem que haja perda das características importantes e essenciais para avaliação do desempenho. A quantidade de detalhes incluídos deve variar de acordo com o propósito da construção do modelo. É recomendado que o modelo seja, a princípio, bem simples e que os detalhes sejam inseridos sistematicamente até que a complexidade desejada seja alcançada.

- **Coleta de dados**
  Ao gerar a descrição do modelo surge a necessidade de se definir quais os tipos dos dados que irão fluir no sistema. Estes serão os dados de entrada e servirão de parâmetros para o modelo, podendo ser tanto valores hipotéticos, como valores baseados em alguma análise preliminar. Essa etapa pode ser realizada paralelamente à construção do modelo.

- **Codificação**

  A escolha dos equipamentos a serem utilizados na simulação, linguagem e ferramentas depende dos seguintes fatores: o sistema a ser simulado; os recursos disponíveis (*hardware e software*); as preferências do usuário; e a descrição do comportamento dinâmico do sistema.

  Três abordagens podem ser utilizadas para descrever o comportamento dinâmico de sistemas discretos: atividades, processos e eventos. Um sistema pode ser visto dinamicamente como uma coleção de processos interativos, cada um composto por diversas atividades, com as interações controladas e coordenadas pela ocorrência de eventos. Normalmente as linguagens são orientadas a uma dessas três abordagens.

  Para a codificação do modelo, isto é, para a geração do programa de simulação propriamente dito podem-se seguir um dos seguintes enfoques (SANTANA 1990a) (SANTANA 1990b):

    - Desenvolvimento do programa de simulação em uma linguagem convencional;
    - Utilização de um ambiente de simulação;
    - Adoção de linguagens de simulação de uso geral;

- **Verificação**

  A fase de verificação consiste em verificar se o programa de simulação é uma implementação válida do modelo. Esta fase responde à seguinte pergunta: *"O modelo operacional (*programa) *representa o modelo conceitual (modelo)?"*. Para pequenos sistemas, uma inspeção bem realizada no modelo final pode ser suficiente. Para grandes modelos é sugerido que a verificação seja contínua, não esperando que todo o programa esteja pronto para iniciar a verificação. Técnicas de verificação formal podem ser adotadas, mas isso envolve um grau de complexidade maior e muitas vezes não compatível com o experimento desenvolvido.

- **Validação**

Na fase de validação deve-se demonstrar que o modelo proposto é uma representação do sistema a ser simulado respondendo a seguinte pergunta: "*O modelo conceitual pode substituir o sistema real?"*. A validação pode ser considerada em dois casos: quando o sistema simulado existe e pode ser medido, e quando o sistema modelado não existe, tendo-se apenas o projeto do sistema a ser simulado. No primeiro caso, o objetivo da análise é avaliar uma mudança proposta no sistema e a validação é baseada na comparação dos resultados do modelo com as medidas reais do sistema. No segundo caso, o objetivo da análise é estimar o desempenho do projeto ou mesmo avaliar projetos alternativos. Neste caso, o modelo pode ser validado baseando-se nos resultados esperados no projeto do sistema e cada suposição e abstração são justificadas. Alternativamente, pode-se lançar mão do conhecimento adquirido com base em sistemas semelhantes, que representem, por exemplo, um sub-conjunto do sistema em estudo (SANTANA 1990) (CHANG 1999).

- **Período experimental – Pré-Simulação**

  É nesta fase que as decisões com relação ao tamanho da simulação, o número de replicações e a maneira como a simulação será iniciada são tomadas de maneira a se obter as informações desejadas a um custo mínimo.

- **Simulação**

  Após as definições no período experimental é iniciada a simulação

- **Análise dos Dados**

  Os dados obtidos das várias execuções do programa de simulação devem ser compilados, tabulados adequadamente e, normalmente, geram gráficos que devem ser minuciosamente analisados. Nesta fase os parâmetros podem ser alterados e o programa de simulação pode ser novamente executado para que novos dados e novos gráficos sejam gerados, levando a novas análises e conclusões.

- **Resultados Finais**

  A partir da análise dos resultados obtidos nas etapas anteriores, as conclusões sobre alterações e melhorias no ambiente devem ser tomadas nessa etapa. A análise dos dados, a elaboração das conclusões sobre o desempenho do sistema, as modificações que devem ser efetuadas, tudo isso depende do nível de conhecimento do analista envolvido no processo. É importante que a análise seja feita considerando-se tanto o conhecimento técnico do sistema considerado, como das técnicas de análise, pois, sem essa conjunção de conhecimentos, pouco poderá ser concluído com segurança.

5.5 Ambientes de Simulação

Um dos maiores obstáculos na elaboração de uma simulação é a transcrição do modelo definido pelo usuário em um programa de simulação. Para auxiliar o usuário nesse processo de transcrição, foi desenvolvido o conceito de ambiente de simulação com geração automática de código. Esses ambientes são ferramentas que oferecem ao usuário uma série de recursos que visam a facilitar o desenvolvimento de uma simulação (CHUNG *and* LEE 2003) (FELTEN, *et al.* 1999). O programa de simulação pode ser gerado e executado automaticamente a partir de uma representação gráfica do modelo do sistema a ser avaliado. Para isso é necessária a inclusão de editores gráficos, interpretadores, compiladores e otimizadores de código. Dessa forma, o ambiente de simulação é referido como sendo um ambiente de simulação automático (BRUSCHI 2002).

A programação visual é uma característica importante nos ambientes de simulação automáticos. Utilizando a programação visual, o modelo de simulação é criado graficamente na tela e os resultados são observados através de gráficos, planilhas ou animações. Quando as informações são apresentadas graficamente, o usuário pode manipulá-las de modo mais eficiente, tornando a programação visual rápida e mais fácil.

Existem vários ambientes de simulação automáticos disponíveis, sendo a maioria proprietários e envolvendo custos elevados. A seguir serão descritos alguns ambientes de simulação.

5.5.1 ARENA

O *ARENA* (SWET *and* DRAKE 2001) é um ambiente de simulação comercial baseado na linguagem *SIMAN V - SIMulation ANalysis* - que modela sistemas discretos, contínuos ou híbridos. O *ARENA* deu origem ao uso de *templates* (módulos pré-definidos) de simulação. Com isto, o *software* é facilmente adaptado para uma indústria específica, uma companhia ou um projeto. Um Editor Profissional *ARENA* permite que o usuário projete seus próprios *templates*. O *ARENA* permite também a construção de modelos animados e possui uma interface gráfica *GUI*, que permite a modelagem do sistema através de módulos, uma interface para *Microsoft VBA* permitindo integração com programas que dão suporte ao *Active X*.

5.5.2 OMNeT++

O *OMNeT++* (*OMNeT++ Community* 2008) é um sistema de simulação discreta seqüencial orientada a objetos, desenvolvido na área acadêmica com o objetivo de ser uma alternativa aos altos preços dos produtos comerciais da área de simulação. Foi desenvolvido para simular, principalmente, redes de computadores, multiprocessadores e outros sistemas distribuídos.

O *OMNeT++* possui os seguintes componentes:

- Biblioteca de simulação;
- Compilador para a linguagem de descrição de topologia
- Editor gráfico de redes
- Interface gráfica para a execução da simulação
- Interface em linha de comando para a execução da simulação;
- Ferramenta para impressão dos gráficos de saída;
- Utilitários (gerador de números aleatórios, ferramenta para criação de *makefiles*).

A Figura 5.2 ilustra a simulação de uma rede de computadores *no OMNeT++* (*OMNeT++ Community* 2008).

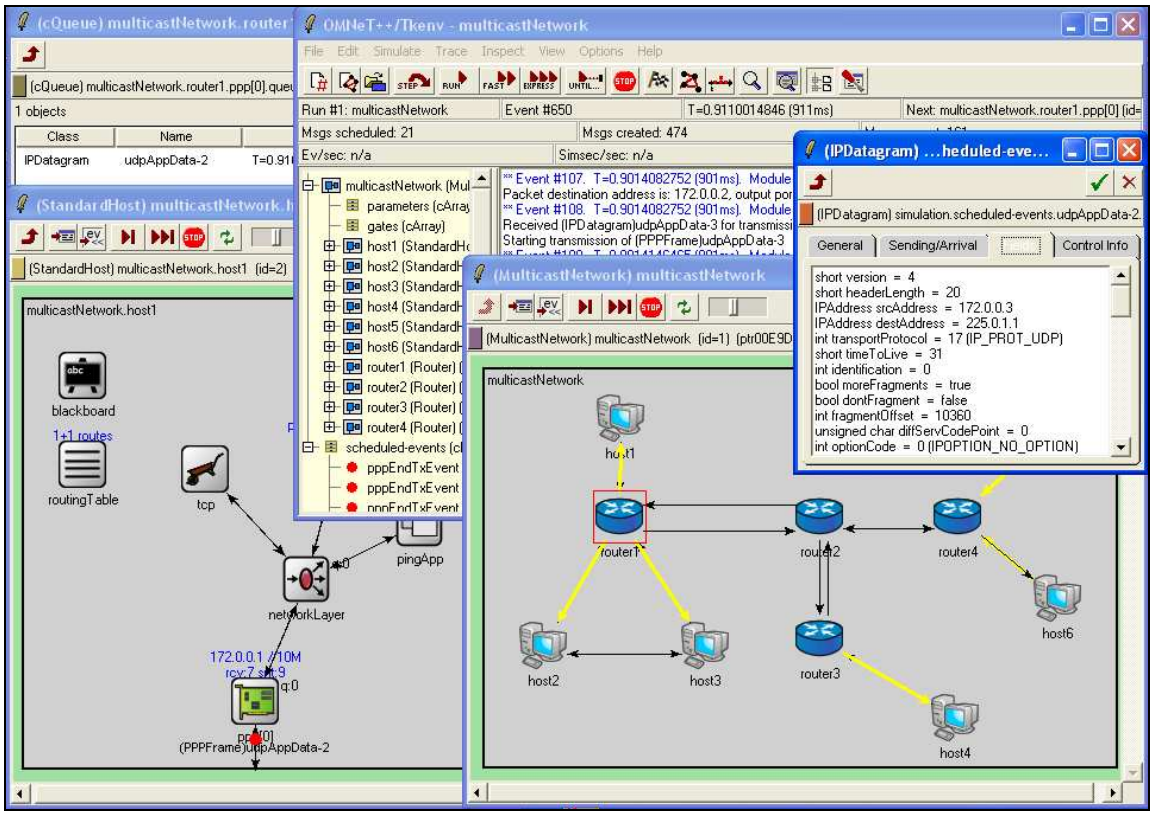


Figura 5.2 – Simulação de uma rede no *OMNet++*

5.5.3 ASiA

O *ASiA* - Ambiente de Simulação Automático - é um ambiente de simulação desenvolvido também na área acadêmica, para simulação seqüencial, no qual o programa é automaticamente gerado pelo sistema, baseado em uma especificação gráfica do modelo fornecida pelo usuário, por meio de redes de fila (SANTANA 1990a) (SANTANA 1990b)

Esse ambiente tem por objetivo afastar o usuário da tarefa de transcrição do modelo em um programa de simulação. O usuário fornece o modelo do sistema e os parâmetros necessários para a simulação através de um editor gráfico. As demais tarefas serão executadas automaticamente pelo *AsiA*, que gera um programa em C, baseado na extensão funcional SMPL (MACDOUGALL 1987).

Quatro componentes básicos são utilizados para a obtenção dos resultados no sistema de simulação automático:

- **Editor gráfico**: fornece ao usuário uma interface amigável, organizando e testando a consistência dos dados de entrada;

- **Gerador de aplicações**: responsável pela geração do programa de simulação;

- **Estágio de saída e análise dos dados**: responsável por analisar os resultados fornecidos pela simulação;

- **Saída gráfica**: responsável por apresentar os resultados em forma de gráficos.

5.5.4 ASDA

O *ASDA* - Ambiente de Simulação Distribuída Automático é um ambiente de simulação voltado à avaliação de desempenho de sistemas computacionais. São considerados nesse ambiente os modelos que melhor representam os sistemas computacionais: dinâmicos e contínuos no tempo, estocásticos e de estados discretos (BRUSCHI 2002).

O *ASDA* tem por objetivo principal fornecer aos usuários de simulação distribuída um ambiente em que a simulação possa ser desenvolvida sem a necessidade de muitos conhecimentos e experiências nas áreas de Computação Paralela, Simulação ou Simulação Distribuída. Neste caso, o usuário deve ter conhecimento do sistema a ser simulado, de como desenvolver o modelo desse sistema e dos objetivos da simulação, ou seja, quais os resultados que a simulação deve fornecer.

O *ASDA* oferece ao usuário um ambiente flexível e amigável. Flexível no sentido de permitir diversas formas de utilização e de disponibilizar diferentes abordagens de simulação. Para usuários pouco experientes, o *ASDA* oferece o recurso de geração automática do programa de simulação, escolha automática da abordagem a ser adotada dentre outras características. Por outro lado, para usuários mais experientes o *ASDA* oferece o recurso de utilização de um código próprio, de modificação do código gerado e definição da abordagem a ser utilizada.

O *ASDA* pode ser considerado um ambiente amigável por oferecer ao usuário a possibilidade de representar seu modelo, parametrizá-lo e definir todas as características desejadas para a simulação através de uma interface gráfica de fácil interação.

O *ASDA* foi desenvolvido no grupo de Sistemas Distribuídos e Programação Concorrente do Instituto de Ciências Matemáticas e de Computação (BRUSCHI 2002) como um aperfeiçoamento natural do *ASiA* (M. SANTANA 1990) (R. H. SANTANA 1990).

5.5.5 NS

O *NS* - *Network Simulator* - (*Network Simulator* - ns-2 2008) é um ambiente de simulação que tem sido utilizado com grande freqüência em pesquisas em redes de computadores. Foi desenvolvido em 1989 a partir de uma variação do *REAL Network Simulator*, um projeto da *Cornell University*, EUA. É um ambiente gratuito utilizado principalmente no meio acadêmico. O *NS* encontra-se atualmente em sua versão 2.

O *NS* simula uma rede *IP* e protocolos tais como *TCP* e *UDP*. Ainda, pode ser utilizado para implementar *multcasting* e alguns dos protocolos da camada *MAC* para simulação de LANs. Quando uma simulação é feita, o *NS* produz uma ou mais saídas baseadas em texto, que contém dados detalhados da simulação. Esses dados podem ser utilizados para análise de simulação ou como entrada para outras ferramentas, tais como *Network Animator – NAM*.

5.5.6 OPNET Modeler

O *OPNET Modeler* (*OPNET Modeler* 2008) (LUCIO, *et al.* 2003) é um ambiente utilizado para modelar e simular redes, permitindo o projeto e estudo redes de comunicação, dispositivos, protocolos, e aplicações com flexibilidade e escalabilidade. Dentre as principais características do *OPNET Modeler* podem-se destacar:

- **Modelagem Hierárquica.**

  OPNET Modeler emprega uma estrutura hierárquica de modelagem. Cada nível da hierarquia descreve aspectos diferentes do modelo completo que está sendo descrito. O modelo de rede hierárquico permite o controle e gerenciamento de topologias complexas de rede, possibilitando a criação de um número ilimitado de sub-redes. A Figura 5.3 ilustra níveis de uma modelagem hierárquica.

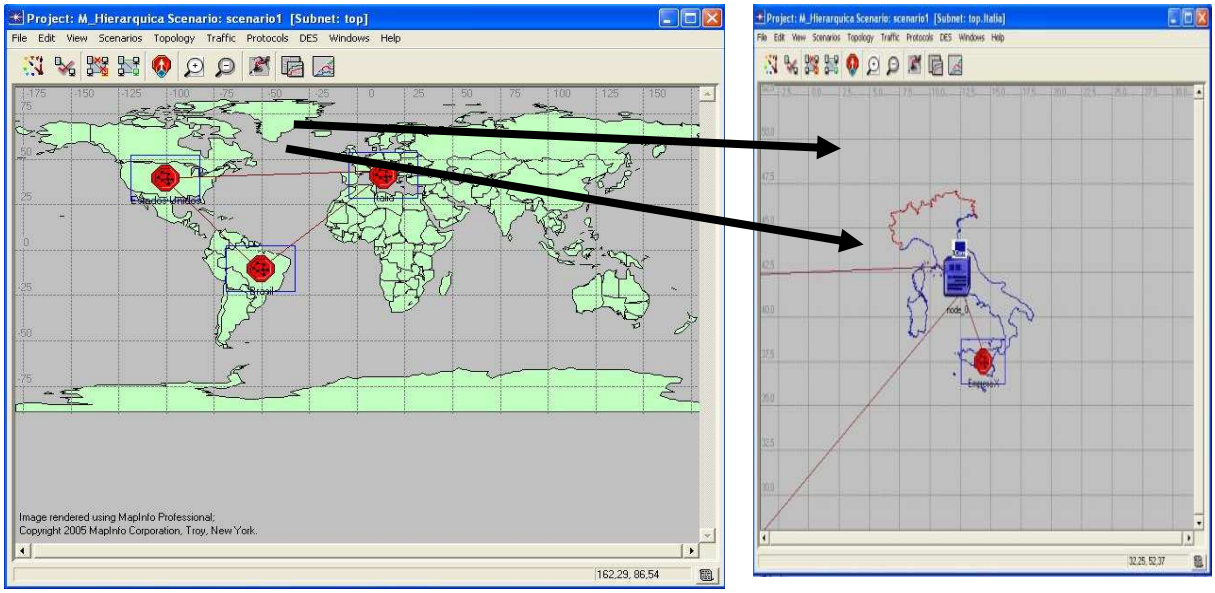


Figura 5.3 – Níveis de uma Modelagem Hierárquica

- **Paradigma Orientado a Objetos.**

  Todos os modelos utilizam o paradigma de orientação a objetos, facilitando o controle e a alteração dos mesmos. Os nós e os protocolos são representados através de classes.

- **Utilização de Máquinas de Estados Finitos – *MEF***

  Os protocolos, aplicações, algoritmos, políticas de filas e outros processos possuem sua representação através de uma máquina de estado finito. Os estados e as transições da *MEF* definem graficamente a progressão de um processo em resposta aos eventos. Isso facilita a visualização e compreensão do funcionamento dos módulos. Todos os estados e transições podem ser alterados, bem como novos estados podem ser inseridos na *MEF*. A Figura 5.4 ilustra a Máquina de Estados Finitos do protocolo *TCP*.

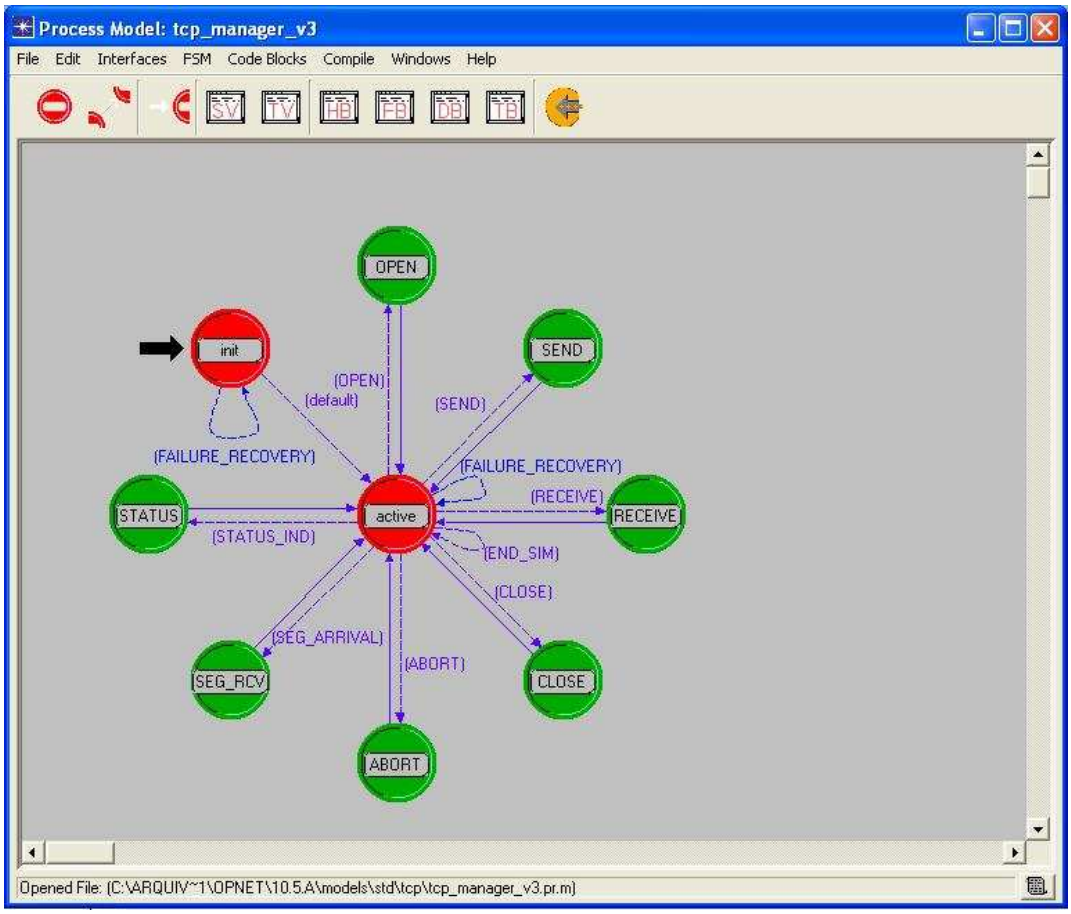


Figura 5.4 – Máquina de Estados Finitos do Protocolo *TCP*

- **Bibliotecas de Protocolos e Aplicações**

  O *Opnet Modeler* disponibiliza um vasto conjunto de bibliotecas com modelos detalhados de protocolos e aplicações como: *HTTP*, *TCP*, *IP*, *IPV6*, *OSPF*, *BGP*, *EIGRP*, *RIP*, *RSVP*, *Frame Relay*, *FDDI*, *Ethernet*, *ATM*, *802.11 Wireless LANs*, *MPLS*, *PNNI*, *DOCSIS*, *UMTS*, *IP Multicast*, *Circuit Switch*, *MANET*, *IP* móvel, *IS-IS*. Todas as *MEF* dos modelos possuem código aberto.

- **Bibliotecas de Dispositivos de Rede**

  A biblioteca de dispositivos de rede padrão possui centenas de modelos específicos e modelos genéricos de dispositivos incluindo roteadores, *switches*, estações de trabalho, servidores e geradores de trafego. É também possível, através do criador de dispositivos, construir a especificação de um novo equipamento, com as características definidas pelo próprio usuário. A Figura 5.5 ilustra a biblioteca de dispositivos de rede do fabricante *3COM*.

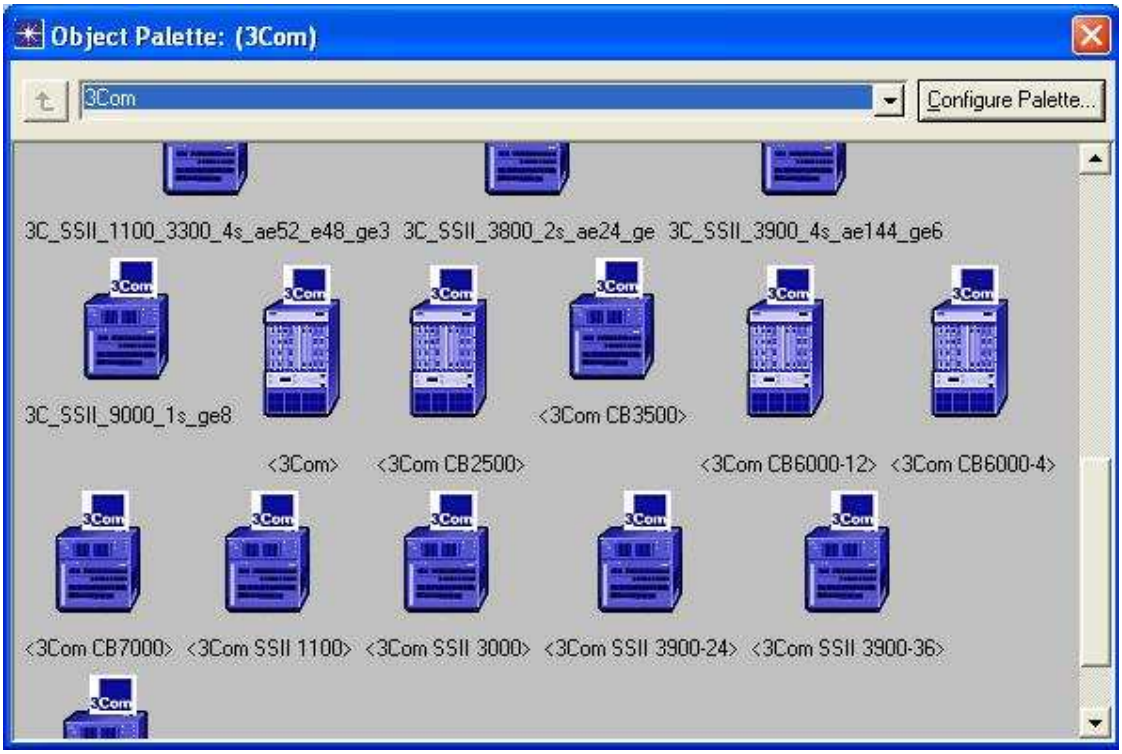


Figura 5.5– Biblioteca de Dispositivos de Redes

- **Modelagem Geográfica e móvel**

  Para redes formadas por componentes móveis, redes sem fio, redes de satélite ou redes com telefonia móvel, é possível controlar a posição de cada nó, dinamicamente, avaliando o seu comportamento e desempenho. É possível definir rotas de deslocamento para cada dispositivo e verificar a intensidade do sinal da rede durante a sua movimentação, por exemplo.

- **Plataforma de simulação e Depurador Integrado**

  A simulação dos projetos é realizada através de simulação discreta seqüencial ou paralela. O depurador integrado verifica rapidamente o comportamento da simulação ou apresenta a lista dos problemas ocorridos durante a execução.

- **Ferramentas de Análise Integradas**

  Possui um conjunto de ferramentas para visualizar os resultados da simulação. Permite traçar e analisar os histogramas, as funções da probabilidade, curvas paramétricas, e intervalos da confiança. Os dados dos resultados podem ser exportados para documentos *HTML*, *XML*, dentre outros formatos.

- **Animação**

  Através do módulo *animate*, o *OPNET* possibilita a animação durante ou após a simulação, e monitora graficamente os valores estatísticos durante a execução da simulação.

- **Importação de Dados**

  É possível importar dados de arquivos texto, *XML* e das ferramentas mais populares como *MRTG, cflowd, TCPdump, Cisco, HP, NetScout, BMC, Concord, Sniffer, Infovista,* entre outras.

O *Modeler* possui uma série de editores hierárquicos que paralelizam diretamente a estrutura de redes dos equipamentos e de protocolos. Esses editores são:

- Editor de Projeto;
- Editor de Nó;
- Editor de Processo.

5.5.6.1 ***Editor de Projeto***

O Editor de Projeto, ilustrado na Figura 5.6, é utilizado para especificar a topologia física de uma rede de comunicação, que é constituída por nós e objetos de ligação (*links*). Os nós e os *links* possuem um conjunto de parâmetros que podem ser configurados através das caixas de diálogo. Um nó pode ser fixo, móvel ou até mesmo um satélite.

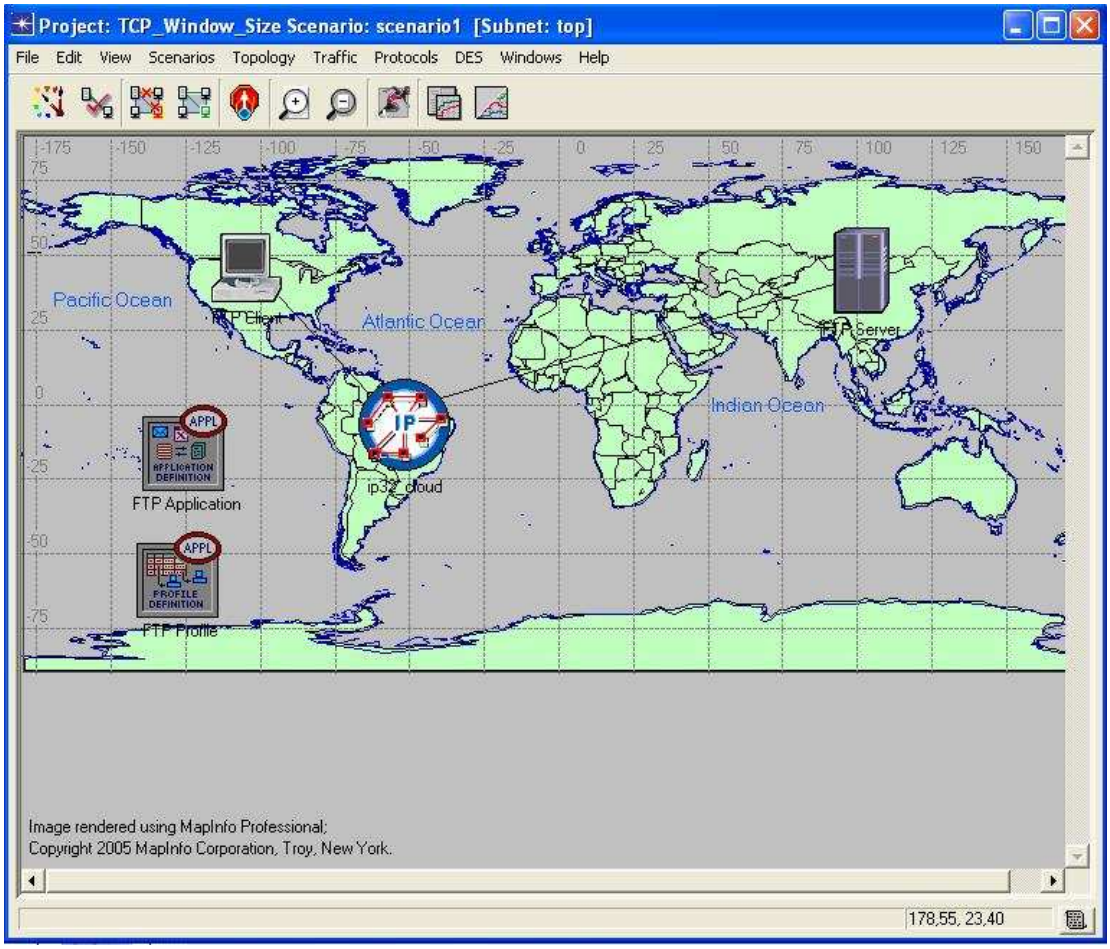


Figura 5.6– Editor de Projeto

Redes complexas podem ser modeladas abstraindo o problema maior e criando sub-redes. As sub-redes por sua vez podem conter outras sub-redes e assim sucessivamente. O editor do projeto fornece o contexto geográfico da rede modelada, com as características físicas refletidas apropriadamente na simulação.

5.5.6.2 ***Editor de Nó***

Cada nó no *OPNET Modeler* é descrito como um diagrama de fluxo de dados entre elementos funcionais, chamados módulos.

Esses módulos podem ser agrupados em duas categorias distintas:

- A primeira categoria é formada por módulos que possuem características predefinidas e um conjunto de parâmetros internos. Os exemplos são geradores de pacote, transmissores ponto a ponto e os receptores de rádio.

- A segunda categoria é formada por módulos que podem ser programados. Esses módulos são protocolos, algoritmos e filas, por exemplo.

Os módulos são interconectados por fios estáticos ou se comunicam trocando pacotes de informações. Os módulos podem gerar, emitir/receber pacotes de outros módulos, para executar sua função dentro do nó. Cada módulo programável em um modelo do nó tem sua funcionalidade definida por um modelo *process*. A Figura 5.7 ilustra o editor de nó.

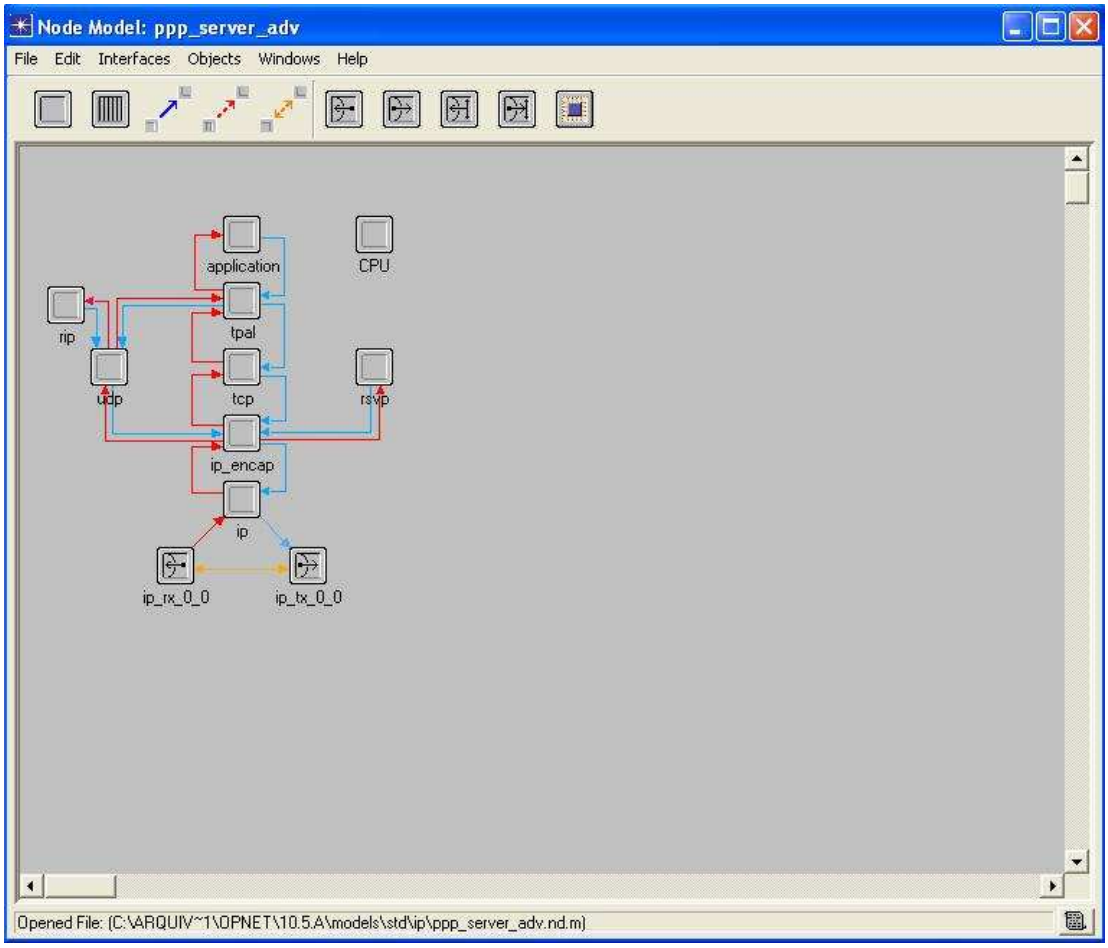


Figura 5.7– Editor de nó

5.5.6.3 ***Editor de Processo***

O Editor de Processo usa uma máquina de estado finito para especificar, em todo o nível de detalhamento, os protocolos, os recursos, as aplicações, os algoritmos e as políticas das filas. A Figura 5.8 ilustra o editor de processo.

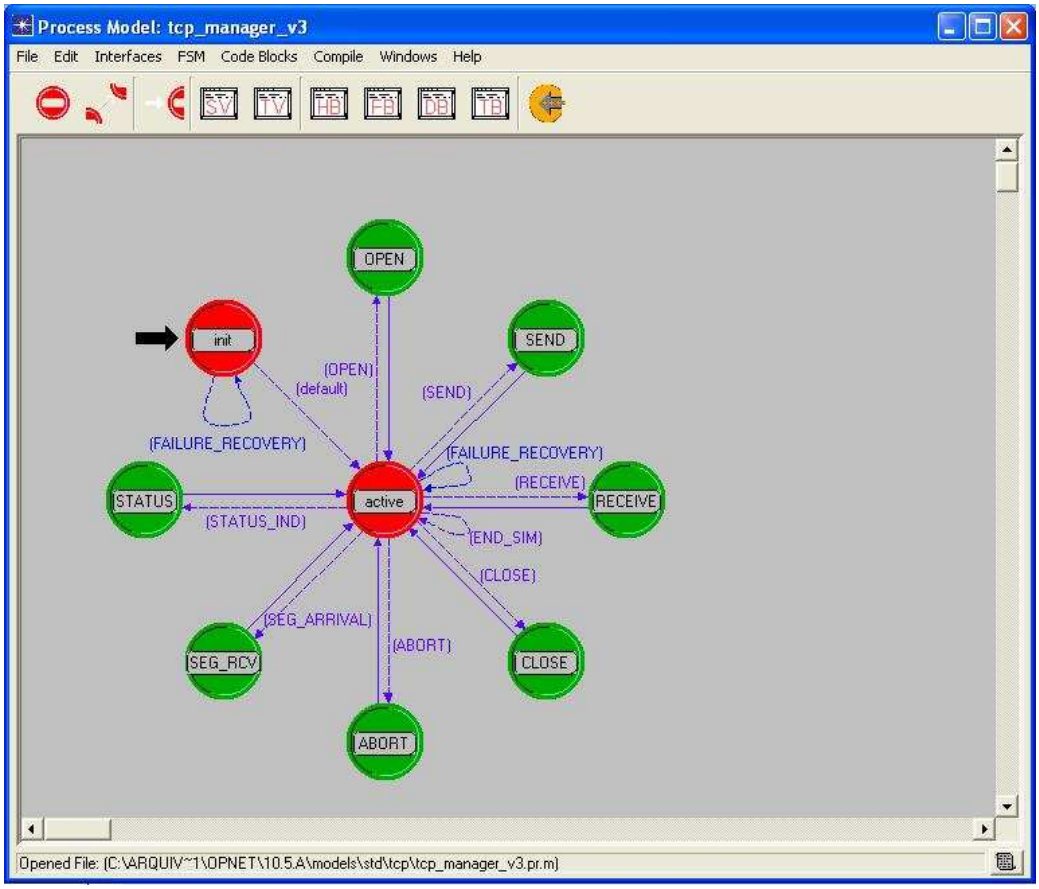


Figura 5.8– Editor Processo

Os estados e as transições da *MEF* definem graficamente a progressão de um processo em resposta aos eventos. Dentro de cada estado, a lógica da operação pode ser especificada usando uma biblioteca de funções predefinidas ou mesmo criando novas funções usando a linguagem C.

5.6 Considerações Finais

Este capítulo apresentou uma visão geral do processo de avaliação de desempenho de sistemas. Sem pretender esgotar o assunto, foram apresentadas e discutidas as técnicas de avaliação de desempenho agrupadas em técnicas de aferição e técnicas de modelagem.

A solução dos modelos desenvolvidos utilizando ambas as técnicas também foi apresentada e divida em soluções analíticas e soluções por simulação, dando ênfase à

segunda abordagem. Alguns ambientes de simulação utilizados para transcrição de modelos definidos pelo usuário em simulações foram apresentados, com destaque para o *OPNET*, que constitui o ambiente escolhido para a realização dos experimentos desenvolvidos neste trabalho devido a:

- Facilidade no processo de modelagem do sistema através de interfaces gráficas e divididas em níveis hierárquicos que permite a expansão dos modelos em sub-redes, por exemplo;

- Grande número de equipamentos de rede pré-definidos em bibliotecas padrão e bibliotecas de fabricantes como *3COM, Dell, NOVELL*, etc.;

- Equipamentos já avaliados e validados por seus fabricantes o que permite obter resultado muito próximo dos obtidos em ambientes reais;

- Possibilidade de alteração/criação de qualquer equipamento e outros componentes como: Protocolo, CPU, filas, etc.;

- Código fonte disponível de todos os equipamentos em C, facilitando alterações e novas implementações;

- Facilidade de inclusão de algoritmos nos diversos níveis do modelo, como algoritmos de escalonamento nos roteadores;

- Disponibilidade de suporte técnico na utilização dos recursos;

No próximo capítulo são discutidos aspectos sobre a caracterização de carga de trabalho da *Web*. O desempenho de um servidor *Web* está diretamente relacionado à carga de trabalho a que ele é submetido. Entender a natureza dessa carga é o primeiro passo para melhorar a qualidade de serviço oferecida

Capítulo

6

Caracterização de Carga de Trabalho

6.1 Considerações Iniciais

A *Web* é um sistema complexo e em constante crescimento motivando várias pesquisas com o objetivo de identificar características comuns dos usuários, possibilitando solucionar problemas existentes e melhorar o seu desempenho.

A caracterização da carga de trabalho é o processo de descrever com precisão a carga de trabalho global do sistema em termos de seus componentes principais (MENASCÉ *and* ALMEIDA 2002). Cada componente é decomposto em componentes básicos e esses componentes básicos são caracterizados pela intensidade da carga de trabalho, tal como a taxa de chegada de requisições.

Quando a intensidade da carga de trabalho é alta, grandes coleções de medições podem ser obtidas. O tratamento desses dados não é muito prático, especialmente se os resultados da caracterização da carga de trabalho forem utilizados para a previsão de desempenho por meio de modelos analíticos (MENASCÉ, ALMEIDA *and* DOWDY 1994). Para facilitar esse trabalho, deve-se substituir a coleção de valores medidos de todos os componentes básicos por uma representação mais compacta denominada modelo de carga de trabalho.

Ao se construir modelos são realizadas abstrações da realidade que está sendo modelada com o objetivo de simplificar e facilitar a coleta e uso de dados. As abstrações comprometem a precisão do modelo, de modo que este precisa ser validado dentro de uma margem de erro aceitável; esse processo é chamado de validação do modelo. A validação dos modelos de carga de trabalho requer a execução de uma carga de trabalho

sintética, composta de resultados do modelo de carga de trabalho, e a comparação das medidas de desempenho obtidas com aquelas da execução da carga de trabalho real. Segundo Menascé (MENASCÉ *and* ALMEIDA 2002), se os resultados forem próximos, dentro de uma margem de erro de 10% a 30%, o modelo de carga de trabalho é considerado válido.

Estudos de caracterização de carga de trabalho da *Web* possuem diversas aplicações, nas quais se podem destacar (MODESTO, *et al.* 2005):

- avaliação de arquiteturas de softwares de coleta;
- melhoria das funções de classificação de páginas em máquinas de busca;
- estudo de comportamentos sociais;
- estudos lingüísticos;

O confronto de coletas realizadas em diferentes épocas fornece uma base para estimativas sobre o futuro da *Web*.

6.2 Trabalhos Relacionados

O desempenho de um servidor *Web* está diretamente relacionado às requisições dos clientes que ele deve atender. Entender a natureza dessas requisições é o primeiro passo para melhorar a qualidade de serviço oferecida (WANG, *et al.* 2003). Calzarossa & Serazzi (CALZAROSSA *and* SERAZZI 1993) determinaram algumas diretrizes gerais no processo de análise da carga de trabalho de um sistema. Dentre essas diretrizes destaca-se a necessidade de se identificar qual o tipo de carga que se pretende analisar, já que na *Web* existem diferentes tipos e tamanhos de objetos diferentes, por exemplo.

6.2.1 Busca de Invariantes

Como a *Web* é um ambiente distribuído e complexo existem inúmeros parâmetros que devem ser levados em consideração no estudo da carga de trabalho. Um dos primeiros trabalhos na área de caracterização de carga de trabalho para *Web* foi realizado por Arlitt & Williamson (ARLITT *and* WILLIAMSON 1996).

Nesse trabalho foram estudados seis registros de *logs* de servidores *Web*, sendo três de ambientes acadêmicos, dois de instituições de pesquisa e um comercial. O objetivo principal desse trabalho era o de determinar invariantes, uma particularidade da carga de trabalho que representa uma verdade universal, aplicável a todo o sistema. Foram definidos dez invariantes, apresentadas na Tabela 6.1 abaixo.

Tabela 6.1 –Invariantes definidas por Arlitt e Williamson

Inv.	**Nome**	**Descrição**
1	Taxa de Sucesso	Taxa de sucesso de atendimento das requisições no servidor de aproximadamente 88%
2	Tipos de Arquivo	Arquivos em formato *HTML* e imagens representam de 90% a 100% das requisições
3	Tamanho médio das transferências	Média menor que 21Kbytes
4	Requisições Distintas	Menos de 3% das requisições são para arquivos distintos
5	Referências Únicas	Aproximadamente um terço dos arquivos e bytes são acessados uma única vez
6	Distribuição de Tamanhos	A distribuição do tamanho dos arquivos segue a distribuição Pareto com $0,40 < \alpha < 0,63$
7	Concentração das referências	10% dos arquivos acessados correspondem a 90% das requisições ao servidor e a 90% dos bytes transferidos.
8	Tempos entre as referências	Os tempos entre as referências dos arquivos são exponencialmente distribuídos e independentes
9	Requisições remotas	Sites remotos representam mais de 70% dos acessos ao servidor e mais de 60% dos bytes transferidos
10	Uso da WAN	10% dos domínios dão origem a mais de 75% dos acessos

De acordo com esse estudo verificou-se que aproximadamente 88% das requisições foram atendidas. Os diferentes tipos de arquivos requisitados foram agrupados nas seguintes categorias: *HTML*, imagem, áudio, vídeo, dinâmico e outros. Verificou-se que, na época, 90% das requisições eram de imagens e objetos *HTML*.

Os autores publicaram esse trabalho em 1996 utilizando como base *logs Web* coletados em 1995. Em 1999, Choi & Limb (CHOI *and* LIMB 1999) publicaram novos estudos utilizando dados coletados em 1998. Comparando os dados com os do trabalho de Arlitt & Williansom com o trabalho de Choi & Limb pode-se verificar uma queda na taxa de sucesso (classe 2xx) de 88% para 75,87%. As requisições restantes foram distribuídos entre as classes 3xx, 4xx e outras, com 21,78%, 0,87% e 1,58% respectivamente. Isso ilustra ainda a existência de poucas requisições com erros, referenciando páginas inexistentes. Ainda, segundo as pesquisas de Choi & Limb os objetos mais requisitados eram imagens e documentos *HTML*.

6.2.2 LOG da Copa do Mundo de Futebol de 1998

Em 2000, Arlitt & Jin (ARLITT *and* JIN 2000) divulgaram um relatório técnico com detalhes do estudo realizado com logs da copa do mundo de futebol de 1998. Os dados foram coletados durante um período de três meses, em trinta servidores, distribuídos em quatro localizações do planeta. Os pedidos do *site* superaram a taxa de 1 bilhão de requisições, com média de 11.000 por minuto. Os resultados mostram que a taxa de sucesso (classe 2xx) foi de 80,52% e corresponderam a 97,86% do total de bytes transferidos. A classe 304, objeto não modificado, obteve 18,75% indicando que a consistência de *cache* teve grande impacto nos servidores *Web*. A Figura 6.1 ilustra o resumo dos códigos de respostas da Copa de 1998.

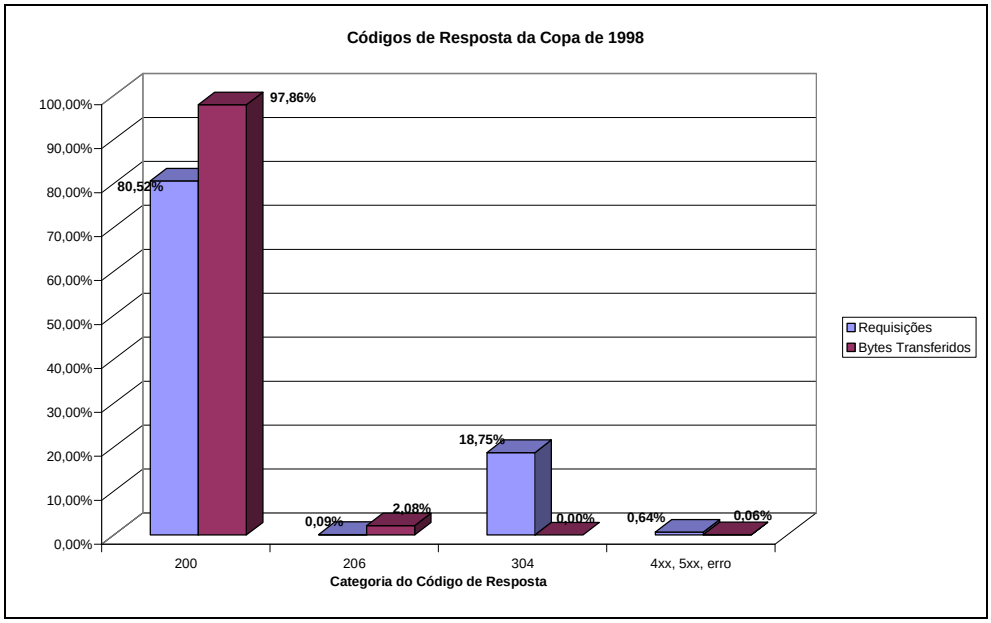


Figura 6.1 – Códigos de Resposta da Copa de 1998

Em relação aos objetos requisitados no *site* da Copa do Mundo verificou-se que as imagens representaram 88,16% das requisições e um volume de dados de 35,02% enquanto os objetos *HTML* representaram 9,85% das requisições e um volume de dados de 38,60%. A Figura 6.2 ilustra um resumo dos objetos requisitados.

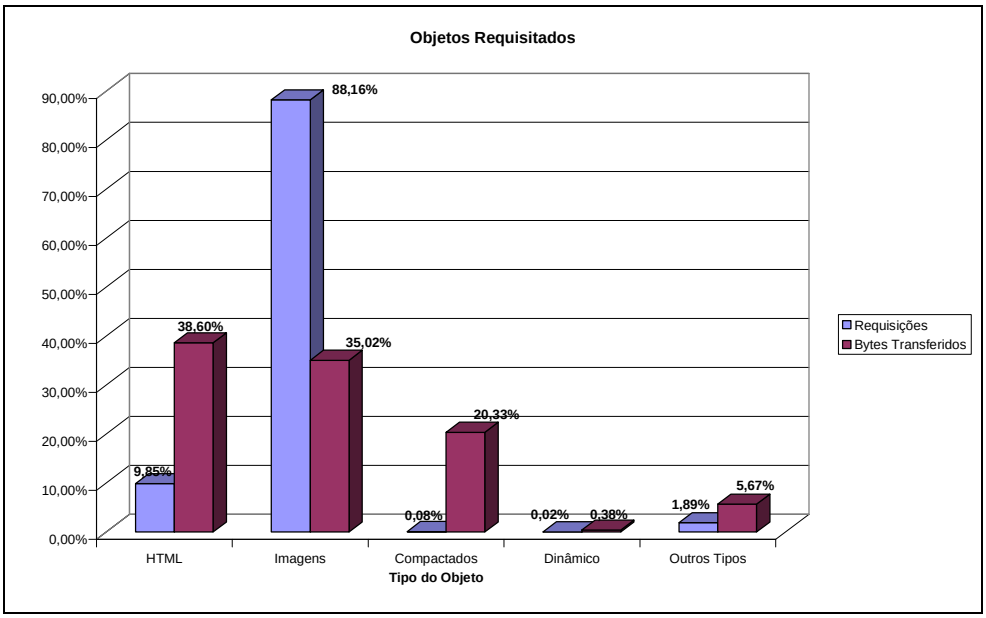


Figura 6.2 – Objetos Requisitados na Copa de 1998

6.2.3 Carga de Trabalho Multimídia

Segundo Guo (GUO, *et al.* 2005) três métodos são normalmente utilizados para a entrega de conteúdo multimídia na Internet:

- *Download*;
- *Pseudo Streaming*;
- Streaming.

O *Download* é um método baseado na utilização do protocolo *HTTP*. Os objetos multimídia são distribuídos da mesma forma que os outros objetos. O cliente requisita um vídeo para o servidor, é realizada a transferência completa do vídeo do servidor para o cliente e aí se inicia a sua apresentação. Esta abordagem é de fácil implementação e não necessita de nenhuma alteração no servidor.

O método de *Pseudo Streaming* também é baseado no protocolo *HTTP* e a técnica é semelhante à do *Download*, porém, a apresentação do objeto multimídia é iniciada mesmo sem o término da transferência completa do objeto do servidor para o cliente. O cliente pode iniciar a apresentação assim que iniciar o recebimento das informações. A maior limitação do método *Pseudo Streaming* ocorre quando as conexões de rede são muito lentas. Nesse caso a apresentação é interrompida diversas vezes esperando a chegada dos dados.

Já no método *Streaming* os protocolos mais comumente utilizados são: *RTP/ RTSP* e *MMS* utilizando o *TCP* para o controle de mídia e *UDP* para a transmissão dos dados (caso o *UDP* esteja desabilitado o *TCP* pode ser utilizado). Pode ser considerado o método mais eficiente para técnica de entrega de objetos multimídia. A apresentação de um objeto multimídia pode ser iniciada assim que o cliente começar a receber os dados do servidor. A técnica de *streaming* oferece aos clientes um controle sobre a apresentação, permitindo que ele adiante, retorne ou interrompa a apresentação.

Comparando a *streaming* com as outras duas técnicas tem-se que a quantidade de dados transferidos entre o servidor e o cliente é somente a necessária para o cliente visualizar

as aplicações que lhe são necessárias. Devido a essa vantagem, muitas aplicações da Internet estão utilizando essa técnica atualmente.

De acordo com o estudo de GUO (GUO, *et al.* 2005) verificou-se que:

- a maior quantidade de transferência de dados multimídia é através de *Download*;

- um grande número de conexões de *Download* é abortado devido ao tempo de espera da rede e a impaciência dos clientes, resultando em mais de 20% de desperdício da largura da banda da rede;

- Comparando com *Download*ing, clientes que utilizam *Pseudo Streaming* tendem a abortar um maior número de conexões.

- A diferença entre a taxa de transferência do servidor e a velocidade de apresentação do objeto no cliente utilizando *Pseudo Streaming* é comum, causando freqüentes *delays* na apresentação ou um tráfego de dados desnecessários na Internet.

6.2.4 Carga de Trabalho de E-Commerce

Em 2004 Wang (WANG, MAKAROFF *and* EDWARDS 2004) apresentou um trabalho sobre carga de trabalho em *sites* de comércio eletrônico. Os *logs* foram coletados de três *Web sites*, o primeiro um *site* internacional de aluguel de carros via Internet. O segundo um *site* de uma companhia de TI que comercializa *software*, *hardware* e serviços via *Web*. O último foi o *site* do Departamento de Ciência da Computação (DC) da Universidade de *Saskatchewan*, no Canadá; este *site* disponibiliza informações aos estudantes, professores e outros usuários. As atividades no *site* do DC são semelhantes às realizadas em um *site* de *e-commerce*, com autenticação de usuários, transações, acesso a banco de dados e utilização de conexões seguras.

A tabela 6.2 e a figura 6.3 ilustram a porcentagem do código das respostas dos três servidores analisados.

Tabela 6.2 – Códigos de Resposta do trabalho de Wang

Código de Resposta	Aluguel de Carros	TI	DC
200	73,00%	84,90%	86,40%
206	0,06%	0,00%	0,23%
301	0,00%	0,00%	2,18%
302	1,70%	6,00%	0,62%
304	25,00%	7,40%	8,22%
4xx, 5xx	0,18%	1,70%	2,35%
outros	0,06%	0,00%	0,00%

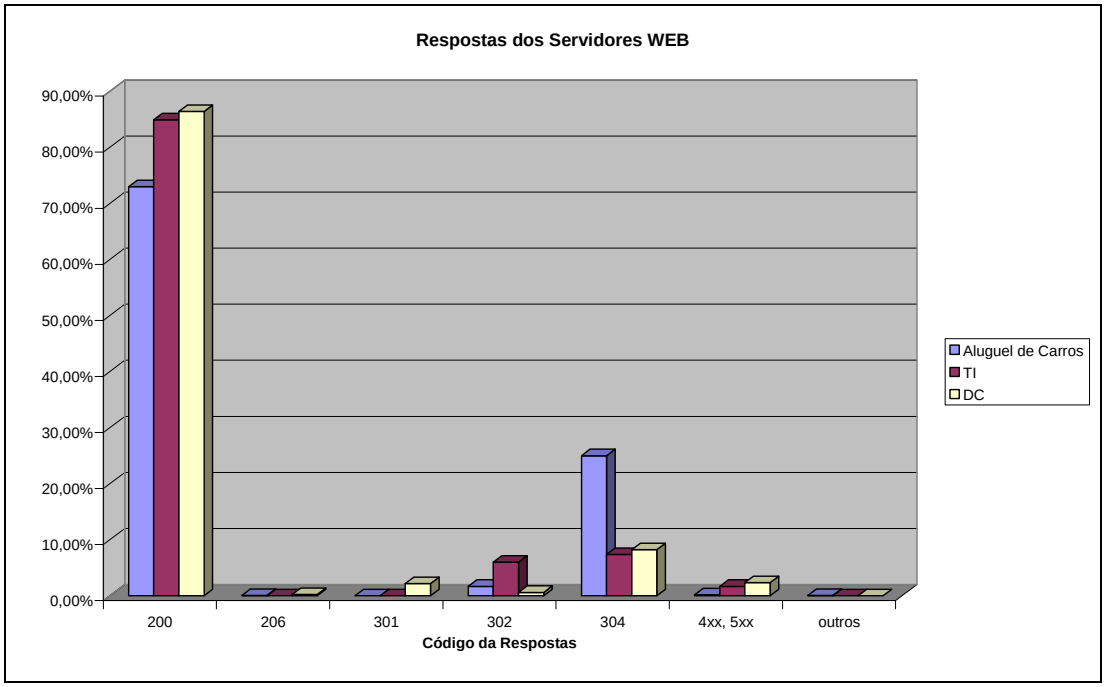


Figura 6.3 – Gráfico Códigos de Resposta do trabalho de Wang

Em relação ao tipo dos objetos WANG classificou-os nas seguintes categorias:

- Imagem (arquivos .gif, .jpg e outros);
- *HTML* (arquivos .*HTML*, .htm, .shtml);
- XML (arquivos .xml, .xsl, .xls, .dtd, .xsd, .xslt, .sty, *and* .xlt).;
- CSS (*Cascading Style Sheet*);
- *JavaScript* (arquivos .js);

- Dinâmico (arquivos .jsp, .asp, .cgi, .pl, .php, .tcl);
- Áudio (arquivos .mp3, .wav, .wma);
- Vídeo (arquivos .avi, .asf, .asx, .rm, .mpeg, .mpg, .wmv, .mov);
- Compactados (arquivos .zip, .tar);
- Formatados (arquivos .txt, .dat, .doc, .pdf, .ppt, .ps, .tex, .bib, .bbl, e outros);
- Programas (.java, .c, .cpp, .sql, .jav, .lisp, .clp, .class, .exe, .dll, .bat)

As tabelas 6.3, 6.4 e 6.5 apresentam a porcentagem de requisições e a porcentagem de *bytes* transferidos de cada tipo de objeto dos *sites* de Aluguel de carros, da Empresa de TI e do Departamento de Computação, respectivamente.

Tabela 6.3 – Objetos requisitados do site de Aluguel de Carros

Tipo Objeto	Requisições	Bytes Transferidos
Imagem	90,29%	46,34%
Dinâmico	5,53%	37,35%
CSS	2,14%	1,62%
JavaScript	1,02%	11,16%
HTML	0,98%	3,18%
Programa	0,028%	0,34%
Outros	0,01%	0,01%
Formatado	0,004%	0,001%
XML	0%	0%
Vídeo	0%	0%
Compactado	0%	0%
Áudio	0%	0%

Tabela 6.4 – Objetos requisitados do site da Empresa de TI

Tipo Objeto	Requisições	Bytes Transferidos
Imagem	68,88%	19,43%
JavaScript	16,08%	7,52%
Dinâmico	13,44%	70,00%
HTML	1,52%	3,04%
CSS	0,063%	0,001%
XML	0%	0%
Vídeo	0%	0%
Programa	0%	0%
Outros	0%	0%
Formatado	0%	0%
Compactado	0%	0%
Áudio	0%	0%

Tabela 6.5 – Objetos requisitados do site do DC da Universidade

Tipo Objeto	Requisições	Bytes Transferidos
Imagem	59,40%	38,68%
HTML	14,76%	10,56%
Dinâmico	14,480%	7,490%
JavaScript	4,680%	4,570%
CSS	3,940%	0,410%
Formatado	1,62%	24,32%
Programa	0,94%	5,08%
Compactado	0,10%	7,44%
XML	0,035%	0,018%
Outros	0,024%	0,29%
Vídeo	0,009%	1,090%
Áudio	0,0008%	0,017%

As figuras 6.4 e 6.5 ilustram respectivamente a porcentagem de requisições e bytes transferidos dos servidores para os objetos: imagem, dinâmico, *HTML* e *JavaScript*. De acordo com o trabalho realizado por Wang pode-se verificar que *sites* de *e-commerce* utilizam uma quantidade menor de objetos *HTML* em relação a objetos dinâmicos.

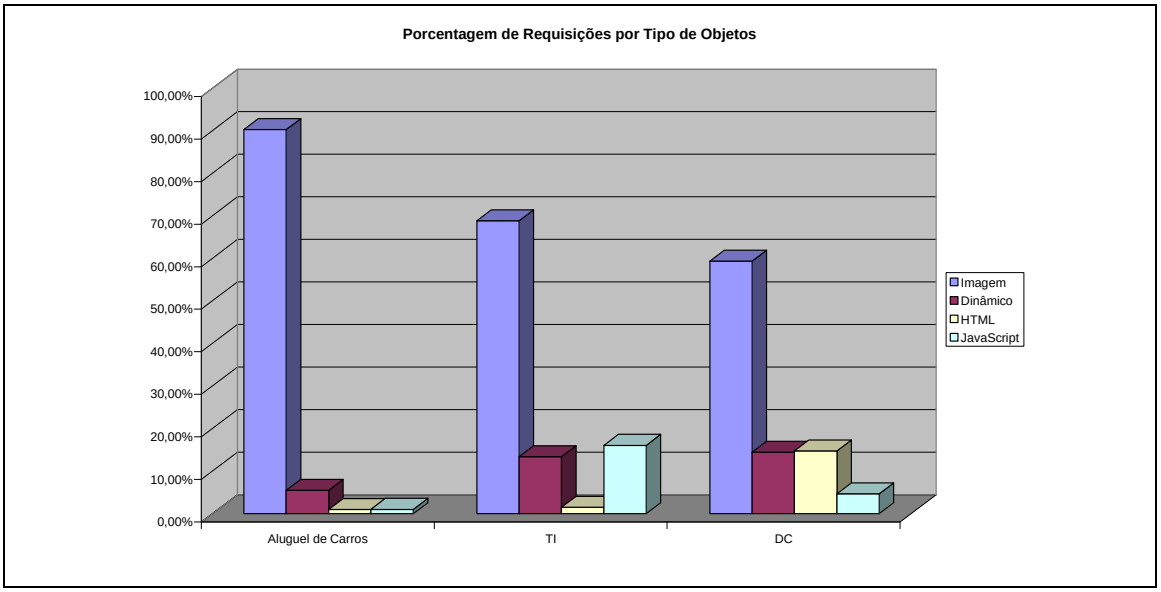


Figura 6.4 – Porcentagem de Requisições por Tipo de Objetos em sites de E-Commerce

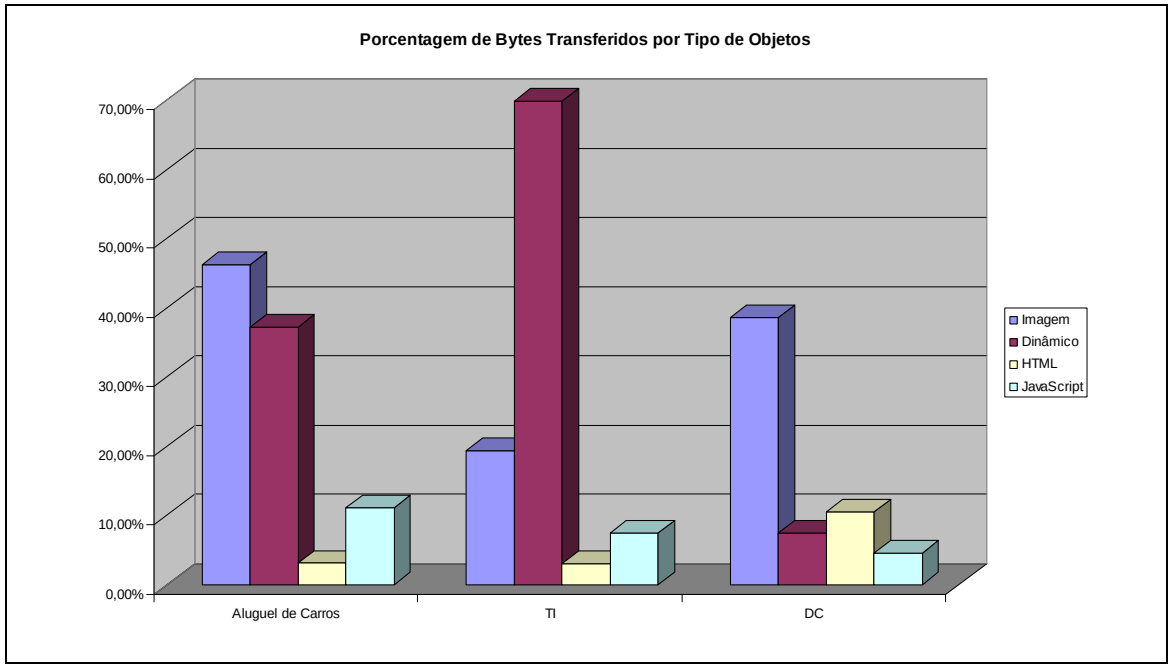


Figura 6.5 – Porcentagem de Bytes Transferidos por Tipo de Objetos em sites de E-Commerce

6.3 Considerações Finais

Vários são os estudos disponíveis na literatura para a caracterização de carga de trabalho para a *Web*. Esses estudos apresentam diferentes caracterizações de acordo com as particularidades de cada ambiente estudado. Entretanto, algumas tendências podem ser destacadas de uma forma geral:

- A maior porcentagem de requisições é para objetos do tipo **imagem;**

- O número de requisições para objetos dinâmicos vem crescendo e em alguns ambientes chega a ser maior que o número de requisições para objetos estáticos;

- Os códigos de resposta da classe 2xx e 3xx formam a grande maioria das respostas dos servidores;

O desempenho de um servidor *Web* está diretamente relacionado às requisições dos clientes que ele deve atender. Entender a natureza dessas requisições é o primeiro passo para melhorar a qualidade de serviço oferecida. Porém, a dificuldade em se gerar

modelos de carga de trabalho é grande devido, principalmente, à diversidade de serviços e aplicações oferecidos através da *Web*. O acesso a traces de servidores *Web* é restrito e as informações desses *logs* são cada vez mais sigilosas, particularmente devido a interesses comerciais e a problemas envolvendo segurança. A alternativa adotada para a caracterização da carga de trabalho utilizada foi baseada em estudos realizados no LaSDPC.

No próximo capítulo é discutida a necessidade da construção de modelos, que permitam compreender a complexidade da *Web*. São apresentados alguns modelos de ambiente *Web* pesquisados na literatura, os quais serviram de base para o desenvolvimento deste trabalho. Finalmente, é descrito o modelo proposto para avaliar o desempenho do ambiente *Web*.

A partir do modelo proposto foram definidos diferentes cenários que são utilizados para avaliar o desempenho do ambiente *Web* com configurações distintas de *QoS*. O modelo geral proposto não impõe nenhuma restrição, ou organização específica, adaptando-se tanto a um sistema distribuído, quanto a uma arquitetura mono ou multiprocessada. A facilidade em se adequar todos os parâmetros do modelo (número de usuários, tipo de aplicação, congestionamento da rede, por exemplo) permite que ele seja facilmente adaptado.

Capítulo

7

Modelagem do ambiente Web

7.1 Considerações Iniciais

A experimentação de um ambiente real, em que se podem obter resultados diretamente a partir de um sistema em funcionamento, em geral apresentam bons resultados, pois normalmente fornecem dados mais próximos da realidade do que outras técnicas. Entretanto, devido ao tamanho da *Web* e sua rápida evolução, torna-se difícil, senão impossível, realizar experimentos para avaliar desempenho do ambiente como um todo. É nesse contexto que se insere a necessidade da definição e construção de modelos, que permitam compreender a complexidade da *Web* atual, e futura, de uma maneira geral, seja qual for a forma escolhida para solucioná-los.

Isso não significa que técnicas como o uso de *benchmarks* ou coleta de dados, devam ser desprezadas. Pelo contrário, a coleta de dados é essencial para verificar se a realidade corresponde àquilo que, implicitamente, se assume como verdadeiro (FLOYD *and* PAXSON 2001). *Logs* de atividade em servidores *Web,* ou do tráfego na rede, por exemplo, podem servir de entrada em simulações baseadas em *traces*. Por outro lado, os resultados de *benchmarks* podem ser utilizados para estimar os parâmetros de entrada para os modelos (MENASCÉ *and* ALMEIDA 2002).

No caso da Internet, devido ao seu tamanho e constante evolução, muitas vezes torna-se economicamente inviável realizar uma experimentação em larga escala. Portanto, a avaliação de desempenho baseada em modelos torna-se a alternativa mais adequada. Entretanto, é bom se ter cuidado para não se cair em algumas armadilhas, entre elas a de

criar modelos por demais simplificados, que não incorporem aspectos fundamentais do comportamento da Internet (FLOYD *and* PAXSON 2001).

A seguir, são apresentados alguns modelos de ambiente *Web* pesquisados na literatura, os quais servem de base para o desenvolvimento deste trabalho. Ainda, são apresentados os modelos de cenários *Web* criados, a descrição dos componentes dos cenários, a carga de trabalho utilizada e as possibilidades de aplicações dos modelos proposto.

7.2 Exemplos de Modelos de Ambiente Web

Menascé e Almeida (MENASCÉ *and* ALMEIDA 2002) discutem alguns modelos de ambientes *Web*, tais como os apresentados nas Figuras 7.1 e 7.2.

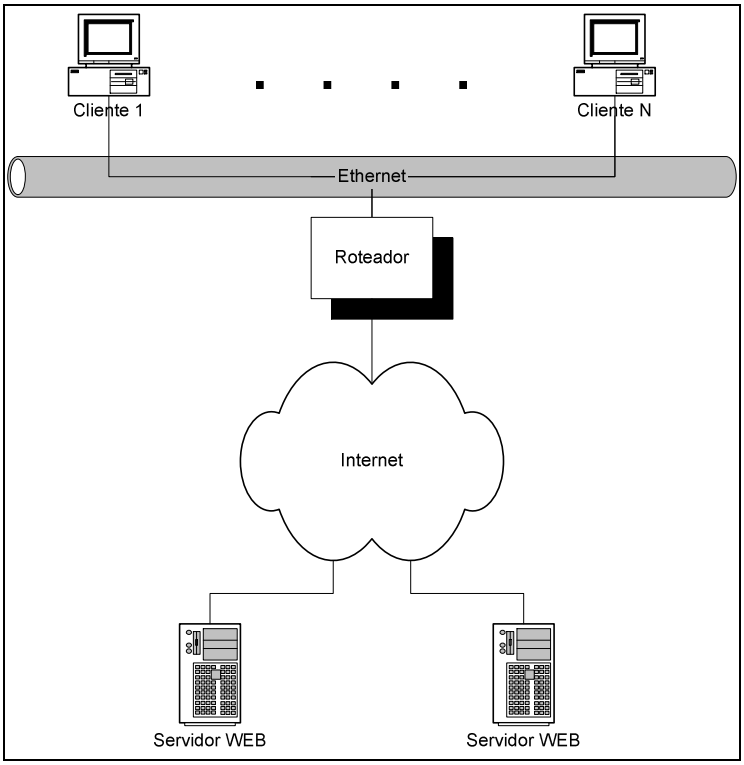


Figura 7.1 - Ambiente *Web* sem servidor *Cache* no Cliente

O primeiro ambiente, Figura 7.1, consiste em um conjunto de N Clientes acessando a *Web*. Esses Clientes estão conectadas a uma LAN, que por sua vez está conectada através de um roteador à Internet. Os clientes realizam as requisições diretamente aos servidores que armazenam o conteúdo desejado.

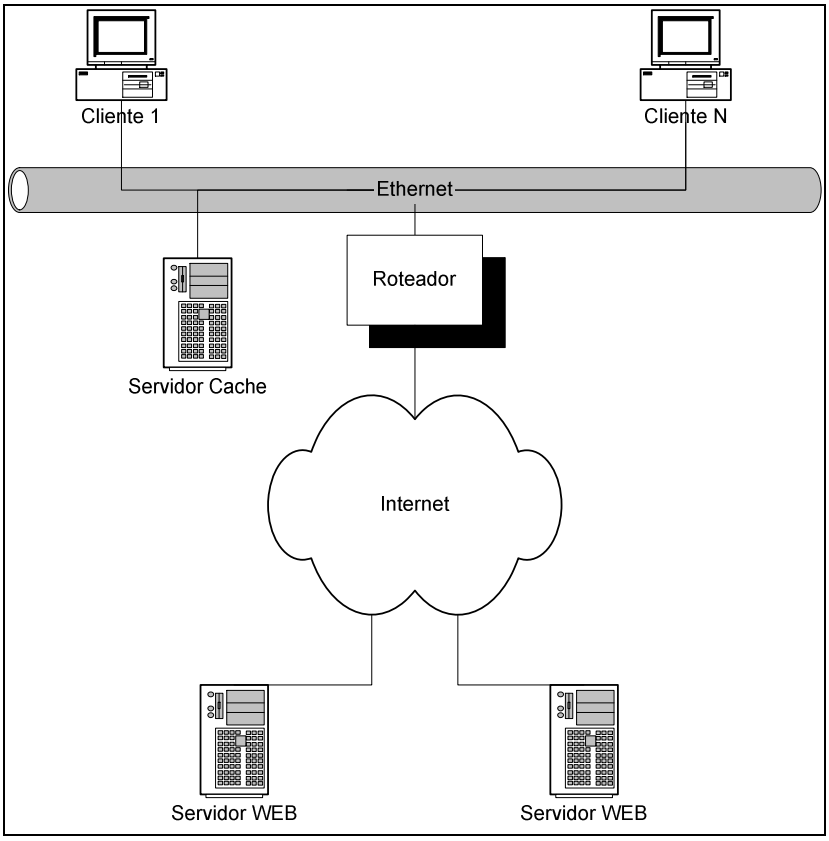


Figura 7.2 - Ambiente *Web* com servidor *Cache* no Cliente

O segundo ambiente, Figura 7.2, apresenta uma configuração semelhante à da Figura 7.1, com o acréscimo de um servidor *cache* junto aos clientes. As requisições dos clientes passam inicialmente pelo servidor *cache*, que mantém copia dos objetos mais requisitados. Se o documento solicitado estiver presente no *cache*, diz-se que ocorreu um acerto (*hit*). Nesse caso, o documento é retornado ao cliente; caso contrário (*miss*), o servidor *cache* atua como cliente e estabelece uma conexão com o servidor *Web*, solicitando o objeto. Após o retorno do objeto, esse é armazenado no servidor *cache* e depois retornado ao cliente que o requisitou. Um documento não localizado no *cache* leva mais tempo para ser recuperado devido aos atrasos impostos no servidor *cache*.

O desempenho do ambiente *Web*, observado pelos usuários clientes depende de diversos fatores, dentre os quais se destacam:

- característica de desempenho da plataforma cliente (CPU, memória, navegador, etc.);
- largura da banda da LAN que conecta os clientes ao roteador;
- largura da banda do *link* que conecta o roteador à Internet;
- características de desempenho do roteador;
- atrasos impostos pela Internet (congestionamento, por exemplo);
- características de carga de trabalho das requisições geradas pelos clientes;
- taxa de acerto do servidor *cache* (caso esse esteja presente);
- atrasos para recuperar o objeto requisitado do servidor *Web* remoto;

7.3 Modelo Proposto de Ambiente Web

A experimentação de um ambiente real geralmente apresenta bons resultados, pois fornecem dados mais próximos da realidade do que outras técnicas. Entretanto, devido ao tamanho da *Web* e sua rápida evolução, torna-se difícil, senão impossível, realizar experimentos para avaliar desempenho do ambiente como um todo. É nesse contexto que se impõe a necessidade da construção de modelos, que permitam compreender a complexidade da *Web* atual, e futura, de uma maneira geral, seja qual for a forma escolhida para solucioná-los. No caso da Internet, devido ao seu tamanho e constante evolução, muitas vezes torna-se economicamente inviável realizar uma experimentação em larga escala. Portanto, a avaliação de desempenho baseada em modelos torna-se a alternativa mais adequada.

Para avaliar o desempenho do ambiente *Web* foram criados cenários que utilizam soluções integradas que visam à melhoria do desempenho do ambiente. Foram modelados cenários com qualidade de serviço em nível de rede, cenários com utilização

de servidores *cache* nos clientes e cenários que utilizam as duas soluções de forma integrada. O objetivo principal é investigar o resultado que se obtém quando se integra *QoS* em alguns pontos da rede e quando se introduz servidores *cache*. Foi criado um modelo geral, chamado de cenário base, que serve de ponto inicial para todos os outros modelos utilizados nas simulações desenvolvidas neste trabalho. Nas seções a seguir serão descritos o cenário base e os cenários derivados dele.

7.3.1 Cenário base

O cenário base serve de ponto inicial para todos os modelos utilizados nas simulações desenvolvidas neste trabalho e é composto por:

- uma LAN composta por 400 clientes;
- um servidor *cache* cliente;
- uma *switch* (SC) responsável por interconectar a LAN, o servidor *cache* e o roteador cliente.
- um roteador (RC) responsável pela conexão com a *switch* SC e com o roteador RS;
- um roteador (RS) responsável pela conexão com a *switch* SS e com o roteador RC ;
- um servidor *Web*;

A tabela 7.1 descreve os componentes do ambiente e suas configurações na modelagem realizada no *OPNET Modeler*. Vale destacar que todos os componentes utilizados são validados pelo seu fabricante e seu desempenho é semelhante ao dos equipamentos reais.

Tabela 7.1 – Configuração dos objetos do modelo *Web* proposto

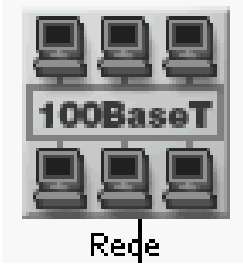
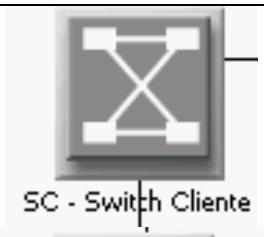
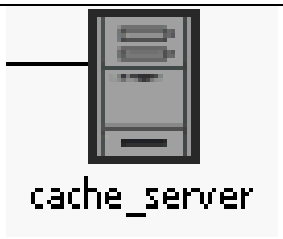
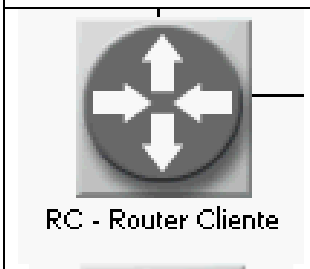
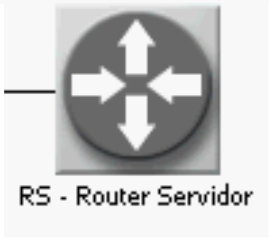
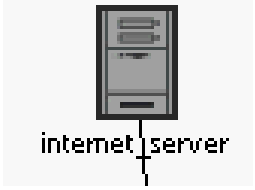
	A Rede é formada por um objeto chamado **100BaseT_LAN** que representa uma *LAN Fast Ethernet*. Esse objeto é formado por um conjunto de N clientes. Essa rede suporta aplicações *FTP*, *HTTP*, *email*, *Rlogin*, *Video*, dentre outras. Essas aplicações são executadas sobre os protocolos *TCP* e/ou *UDP*. Para cada aplicação suportada pela LAN deve ser especificado o grupo de clientes que pertencem a LAN. Essa LAN é formada por 400 clientes. Os clientes são divididos em quatro categorias: **padrão, bronze, prata e ouro** (100 clientes em cada uma). As classes de usuários possuem diferentes prioridades de serviço, para avaliar os mecanismos de diferenciação no atendimento das requisições desses usuários. Usuários ouro possuem maior prioridade sobre usuários prata que possuem maior prioridade sobre os usuários bronze que por sua vez possuem maior prioridade sobre os usuários padrão.
 	As *switches* SC e SS são objetos *ethernet16_layer4_switch.* Esse objeto representa uma *switch* de camada 4 com 16 interfaces Ethernet. *Switches* de camada 4 (L4) permitem o redirecionamento das requisições clientes através de informações nos pacotes *TCP* e/ou nos cabeçalhos de requisições *HTTP*. Nos ambientes em que o servidor *cache* está presente, o SC é o responsável por realizar todo o redirecionamento das requisições dos usuários ao servidor *cache*, de forma transparente aos usuários.

Tabela 7.1 – Configuração dos objetos do modelo *Web* proposto

	O servidor *cache* é um objeto chamado *ethernet_cache_server_adv*. Esse objeto representa um nó servidor com as aplicações sendo executadas sobre os protocolos *TCP*/IP e *UDP*/IP. Esse nó suporta *cache Web*. O servidor *cache* opera de forma transparente aos usuários. Esse tipo de implementação utiliza componentes da rede para redirecionar as requisições para os servidores *cache*, nesse caso a *switch* SC. O servidor *cache* recebe todas as requisições dos usuários e verifica se o objeto requisitado está presente; em caso afirmativo (*hit*) retorna o objeto ao usuário; em caso negativo (*miss*) realiza a requisição do objeto ao servidor *Web*. A taxa de acerto do *cache* foi configurada em 30%.
 	Os roteadores RC e RS são objetos ethernet2_slip8_gtwy_base. Esses objetos são *gateway* IP. A conexão entre os roteadores RC e RS é feita através de enlaces PPP DS1 em dois canais de 1024 Kbps. Para avaliar o comportamento do ambiente em condições de tráfego extra na rede, foram gerados fluxos de dados na rede que ocuparam em 0%, 40%, 60% e 80% de sua disponibilidade.
	O servidor internet é um objeto ethernet_server. Esse modelo representa um nó servidor com as aplicações sendo executadas sobre os protocolos *TCP*/IP e *UDP*/IP. As aplicações suportadas por esse servidor são definidas no atributo Supported Services.

A conexão entre os componentes do cliente (LAN, SC e RC) e a conexão dos componentes do servidor (SS e servidor *Web*) são feitas por enlaces 100BaseT. A

conexão entre os roteadores RC e RS é feita através de enlaces PPP DS1 em dois canais de 1024 Kbps. A figura 7.3 ilustra o cenário base. O ANEXO C contém o documento XML com a descrição do Cenário Base.

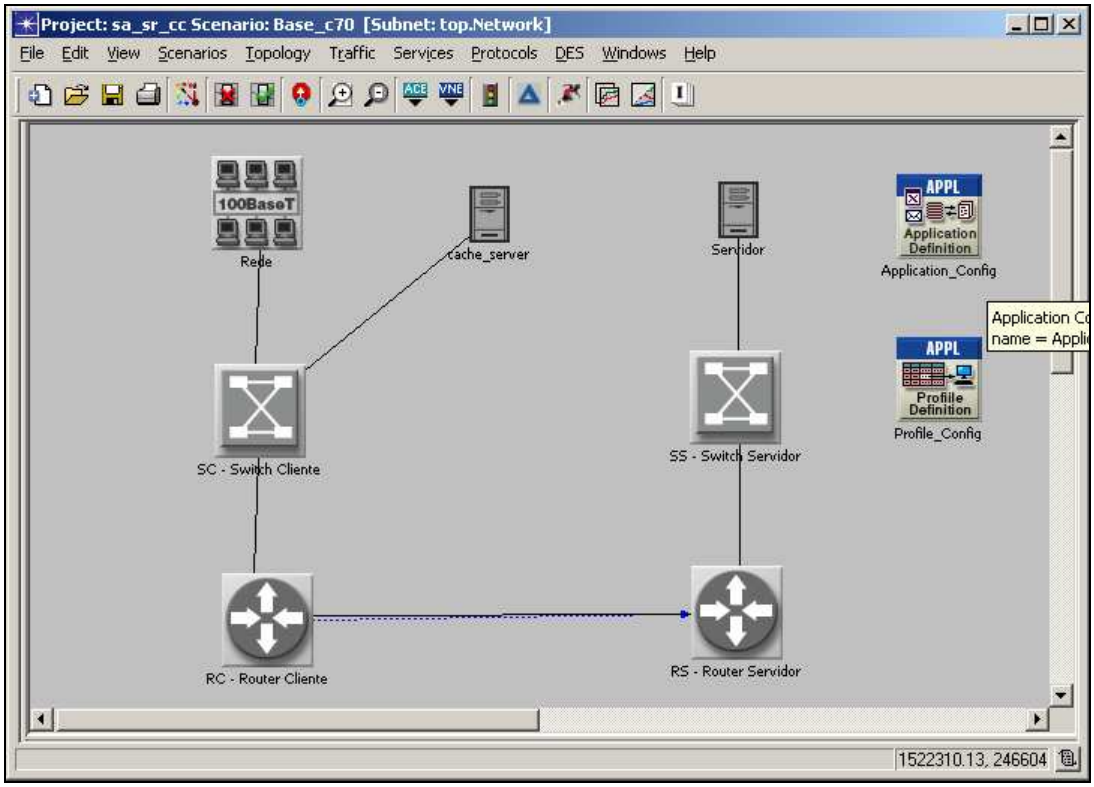


Figura 7.3 - Cenário BASE utilizado nas simulações

7.4 Avaliação, Validação e Experimentação do Modelo

A metodologia utilizada para avaliação de desempenho neste trabalho segue a proposta de Jain (Jain, R., 1991), que divide este tipo de trabalho em 10 etapas:

1. Determinar os objetivos do trabalho e definir o sistema.
2. Listar os serviços do sistema e possíveis resultados.
3. Selecionar as métricas.
4. Listar os parâmetros do sistema e da carga.
5. Selecionar os fatores.
6. Selecionar as técnicas de avaliação.
7. Selecionar a carga de trabalho.
8. Projetar os experimentos.
9. Analisar e interpretar os resultados.
10. Apresentar os resultados. Iniciar novamente, se necessário.

7.4.1 Determinar os objetivos do trabalho e definir o sistema

Este é o primeiro passo em qualquer projeto de avaliação de desempenho. Deve-se conhecer os objetivos e saber exatamente a formação do sistema. Isso é necessário, pois, dados os mesmos conjuntos de hardware e software, as características relevantes para a avaliação do sistema podem variar dependendo dos objetivos da avaliação.

O objetivo geral deste trabalho é investigar alternativas para melhorar o desempenho de aplicações *Web*, avaliando o impacto da utilização de servidores *cache na Web*, em ambientes com e sem diferenciação de serviços.

7.4.2 Listar os serviços do sistema e os possíveis resultados

Todo sistema possui uma lista de serviços com um conjunto de resultados e cada resultado, dentro da perspectiva do usuário, pode atingir os objetivos desejados ou não. Por exemplo, quando um usuário requisita uma página na *Web*, ele espera que os dados sejam recuperados em um determinado tempo de resposta. Esta etapa objetiva definir os serviços disponíveis aos usuários e seus possíveis resultados, selecionando as métricas de desempenho e cargas de trabalho mais adequadas.

7.4.3 Selecionar as métricas de desempenho

Definir as métricas do sistema significa estabelecer critérios para a comparação do desempenho. As métricas podem ser: velocidade, precisão, disponibilidade dos serviços, dentre outras. Neste trabalho, é utilizado o tempo de resposta geral das requisições, o tempo de resposta para cada classe de usuário, o número total de requisições realizadas e o número de requisições por classe de usuário. O tempo de resposta é medido a partir do início da requisição dos usuários até a recuperação total de todos os objetos da requisição. O número total de requisições é a soma das requisições que obtiveram sucesso e das requisições canceladas.

7.4.4 Listar os parâmetros do sistema e da carga

São todos os parâmetros que afetam o desempenho. A lista pode ser dividida em parâmetros do sistema (hardware e software, normalmente imutáveis durante os experimentos) e parâmetros da carga (requerimentos do usuário, normalmente variáveis entre os experimentos). A lista pode não ser completa, pois, após a primeira bateria de análises se pode descobrir novos parâmetros afetando o desempenho.

7.4.5 Selecionar os fatores e seus valores

A lista de parâmetros pode ser divida naqueles que variam durante a avaliação e aqueles que não variam. Os que variam são chamados de fatores e seus valores são denominados níveis. Em geral, a lista de fatores e seus possíveis níveis são maiores que

os possíveis de serem analisados. Deve-se começar com uma lista pequena, escolhendo como fatores os parâmetros cuja variação afeta de forma mais intensa o desempenho. Os fatores considerados neste trabalho são:

- **Diferenciação na Rede**

  O fator diferenciação de rede possui dois níveis: com e sem diferenciação. Os cenários que são configurados com diferenciação de serviço utilizam a política de enfileiramento por prioridades - *Priority Queuing* (PQ).

- **Servidor *cache***

  O fator servidor cache possui três níveis: sem servidor cache, com servidor cache tradicional ou com servidor cache CDF. A taxa média de acerto do servidor cache é de 30%.

- **Congestionamento na rede**

  O fator congestionamento na rede possui quatro níveis: 0Kbits/s (0%), 616Kb/s (40%), 924Kb/s (60%) e 1232Kb/s (80%). Esse congestionamento está presente entre os roteadores RC e RS.

- **Tipo de requisição considerada**

  O fator tipo de requisição considerada possui dois níveis: carga estática e carga dinâmica.

7.4.6 Selecionar as técnicas de avaliação

A técnica utilizada neste trabalho é a modelagem e a abordagem utilizada para a solução do modelo foi a simulação orientada a eventos, principalmente por permitir verificar, sem grandes dificuldades, diferentes configurações do modelo. A simulação é uma abordagem bastante flexível, permitindo verificar diferentes configurações do modelo prontamente. Os resultados podem ser obtidos com mais rapidez do que se fossem realizados experimentos tradicionais (o que, neste caso seria uma tarefa bastante complexa). Os modelos foram implementados utilizando-se o ambiente de simulação *OPNET Modeler* (*OPNET* 2008). Esse ambiente é empregado para modelar e simular

sistemas computacionais, permitindo o projeto e o estudo de redes de comunicação, dispositivos, protocolos e aplicações com flexibilidade e escalabilidade.

7.4.7 Selecionar a carga de trabalho

Um dos problemas existentes na simulação de sistemas relacionados à *Web* consiste na geração de carga de trabalho para os experimentos. Neste estudo foi utilizada uma carga de trabalho mista, contendo requisições estáticas e dinâmicas. As cargas de trabalho utilizadas são baseadas nas análises dos arquivos de *logs* do servidor apache do CISC1 (Centro de Informática de São Carlos), realizadas por Silva (SILVA 2006). A carga de trabalho estática é descrita na Tabela 7.2 e a carga dinâmica na Tabela 7.3.

Tabela 7.2 - Características das Requisições Estáticas por Classe de Usuário

Usuário	**Classe de Serviço**	**Aplicação**	**Configuração de uma Requisição *HTTP* (Objetos por Página)**
Padrão	BE	*HTTP* 1.1	1 objeto *HTML* tam. bytes = lognormal (8.55,1.42) 2 objetos IMAGEM tam bytes= lognormal (8.25,1.62)
Bronze	AF1x, AF2x		
Prata	AF3x, AF4x		
Ouro	EF		

Tabela 7.3 - Características das Requisições Dinâmicas por Classe de Usuário

Usuário	**Classe de Serviço**	**Aplicação**	**Configuração de uma Requisição *HTTP* (Objetos por Página)**
Padrão	BE	*HTTP* 1.1	**1 objeto DINÂMICO** tam. bytes = lognormal (8.55,1.42) 2 objetos IMAGEM tam bytes= lognormal (8.25,1.62)
Bronze	AF1x, AF2x		
Prata	AF3x, AF4x		
Ouro	EF		

Como pode ser observado nas Tabelas 7.2 e 7.3, a diferença entre as requisições estáticas e dinâmicas está somente em um objeto, *HTML* na estática e DINÂMICO na dinâmica. Os outros objetos são idênticos. Além disso, os objetos *HTML* e DINÂMICO possuem o mesmo tamanho e a diferença entre eles está no fato do objeto DINÂMICO ser o resultado de uma consulta realizada no servidor que demanda processamento interno e execução de chamadas à base de dados. O tempo de resposta médio da

1 www.icmc.usp.br

execução e retorno do objeto DINÂMICO é uma função exponencial com média de 0.81 segundos. Esse tempo de processamento do objeto DINÂMICO foi baseado em estudos realizados pela IBM, no tempo de processamento de uma parte de uma transação de e-commerce em base de dados DB2 (MARTIN, *et al.* 2002).

7.4.8 Projetar os experimentos

Os experimentos a serem realizados devem permitir a determinação da influência de cada fator e a interação entre os fatores. As primeiras fases contemplam mais fatores e menos níveis, passando-se para vários níveis nos fatores que mais influenciaram no desempenho. A partir dos fatores e seus níveis, é decidida a seqüência de experimentos que visam a obter o máximo de resultados. Os experimentos deste trabalho foram projetados para responder às seguintes questões básicas:

- Qual o impacto da utilização de servidores *cache* em ambientes com diferenciação de serviços?
- A utilização de diferentes classes de usuário pode apresentar ganhos para os usuários com maior prioridade?
- Qual a influência do congestionamento da rede no tempo de resposta dos pedidos dos usuários?
- A influência da utilização de servidores *cache* é a mesma se a carga de trabalho for estática ou dinâmica?
- Qual a influência da diferenciação de serviços na rede nos diferentes tipos de carga de trabalho?
- As soluções de diferenciação de serviços apresentam resultados positivos se aplicadas de forma isolada?

Para responder a essas perguntas deve ser utilizada uma metodologia. Neste trabalho optou-se por apresentar os resultados com todos os níveis propostos e por analisar esses resultados considerando-se dois níveis para cada fator e utilizar um projeto fatorial 2^4. Isto é, consideram-se 4 fatores com 2 níveis cada um.

7.4.9 Analisar e interpretar os dados

Medições e simulações, por tratar-se de experimentos aleatórios, resultam em resultados estocásticos. É necessário levar em conta a variação dos resultados, pois, simplesmente comparar as médias pode conduzir a conclusões errôneas. A interpretação dos resultados é essencial para entender que uma análise conduz a resultados e não a conclusões, resultados estes que servirão como base para que os responsáveis pelas decisões possam realizar as conclusões.

Neste trabalho, para cada experimento foram coletadas quinze amostras, visando à estabilidade dos dados e ao estabelecimento do intervalo de confiança desejado. Foram calculados a média e o desvio padrão para todos os dados coletados. Os resultados apresentados correspondem à média desses dados, dentro de um intervalo de confiança de 95%.

7.4.10 Apresentar os resultados. Iniciar novamente, se necessário

A apresentação dos resultados deve ser realizada de maneira que permita uma fácil compreensão dos dados. Neste ponto, pode-se considerar um retorno aos passos iniciais e a retomada dos experimentos, com base nos valores alcançados.

7.5 Organização dos experimentos

O experimento realizado seguiu uma organização fatorial parcial. Foram considerados quatro fatores conforme detalhado na seção 7.1.5. Os níveis definidos por esses fatores foram totalmente combinados entre si, com exceção da combinação do *cache* CDF e a inexistência de diferenciação na rede, uma vez que esse caso não faria sentido pois as marcações de prioridade realizadas pelo servidor *cache* não seriam utilizadas pelos roteadores, apresentando os mesmos resultados que um servidor *cache* tradicional.

Dessa forma, a combinação entre os níveis de *cache* e diferenciação de serviços dá origem a cinco cenários que são descritos a seguir. É importante observar que a combinação entre dois fatores, um com dois níveis e um com três níveis daria origem a seis cenários, mas, como a combinação entre rede com diferenciação e com servidor *cache* tradicional não foi considerado, têm-se cinco cenários e um planejamento fatorial parcial.

Esses cinco cenários serão experimentados combinados com os níveis dos outros dois fatores analisados: tipo de requisição e congestionamento da rede. Desta forma, foram realizados quarenta experimentos distintos. Esses experimentos permitem verificar a influência de cada um dos fatores e a interação entre eles. A seguir são descritos os cinco cenários.

- **Cenário SS**

  O cenário SS é um modelo que representa a Internet em sua proposta original, sem nenhuma forma de diferenciação de serviços e sem servidor *cache* cliente. O serviço oferecido nesse cenário baseia-se em um modelo de melhor esforço (*best-effort*).

- **Cenário SC**

  O serviço oferecido pelo cenário SC também se baseia em um modelo de melhor esforço (*best-effort*), porém com a presença de um servidor *cache*. Todas as requisições dos usuários são submetidas ao servidor *cache* para verificar sua presença em *cache* ou não.

- **Cenário CS**

  O cenário CS está configurado com diferenciação de serviço em nível de rede e não possui servidor *cache* cliente. As regras de diferenciação dos usuários são aplicadas nos nós de ingresso da rede (roteadores RC e RS). Requisições com menor prioridade ficam armazenadas em uma fila nos roteadores até que as requisições com maior prioridade sejam despachadas pela rede.

- **Cenário CC**

  O cenário CC está configurado com diferenciação de serviço em nível de rede e com servidor *cache* cliente padrão. O cenário CC utiliza um servidor *cache* padrão que realiza todas as requisições ao servidor *Web* com tipo de *serviço best-effort*.

- **Cenário CDF**

  O cenário CDF está configurado com diferenciação de serviço em nível de rede e com servidor *cache* com diferenciação de serviço implementado. Toda a requisição do servidor *cache* para o servidor *Web* é realizada com a mesma prioridade que o cliente solicitou originalmente.

A Tabela 7.4 ilustra um resumo das configurações dos cenários.

Tabela 7.4 - - Configuração dos Cenários

Cenário	Diferenciação de Serviço	Servidor *Cache*	Classe de Usuário	Congestionamento Rede	Carga de Trabalho
SS	Não	Não	ouro – bronze – prata - padrão	0% - 40% 60% - 80%	Estática Dinâmica
CS	Sim	Não	ouro – bronze – prata - padrão	0% - 40% 60% - 80%	Estática Dinâmica
SC	Não	Padrão	ouro – bronze – prata - padrão	0% - 40% 60% - 80%	Estática Dinâmica
CC	Sim	Padrão	ouro – bronze – prata - padrão	0% - 40% 60% - 80%	Estática Dinâmica
CDF	Sim	CDF	ouro – bronze – prata - padrão	0% - 40% 60% - 80%	Estática Dinâmica

Nos experimentos, as amostragens foram de 600 segundos e os dados coletados correspondem ao intervalo de tempo de 90 a 600 segundos. Foram descartados os 90 segundos inicias da simulação, pois esse tempo é o necessário para que sejam alocados dinamicamente o endereçamento IP de todos os nós do modelo e iniciar as requisições dos usuários. Cada amostra foi coletada quinze vezes, visando à estabilidade dos dados e estabelecimento do intervalo de confiança desejado.

7.6 Aplicação dos Modelos/Cenários Propostos

Os cenários propostos para avaliar o desempenho do ambiente *Web* podem ser adaptados para realizarem investigações que visam à melhoria do desempenho de diversos ambientes. Todos os elementos utilizados nos modelos (servidores, roteadores, *switches*) são validados pelo fabricante, e seu desempenho é semelhante ao dos equipamentos reais.

Os modelos podem ser utilizados para planejar/avaliar o desempenho de redes de computadores que necessitem de diferenciação de serviços para seus usuários como, empresas de comércio eletrônico, universidades, dentre outras.

Em uma universidade, por exemplo, os usuários internos podem ser divididos em quatro classes: alunos (padrão), funcionários (bronze), professores (prata) e gestores (ouro). Os gestores são usuários com maior prioridade e devem ser atendidos com melhor qualidade de serviço. Através do modelo, podem ser alterados os números de usuários de cada classe e pode-se avaliar o desempenho da rede com o crescimento do número de usuários com menor prioridade, alunos. Com esse levantamento é possível prever a necessidade de aumento ou não da velocidade de conexão existente entre os clientes e os servidores da universidade.

Da mesma forma que em universidades, podem-se utilizar os modelos para avaliar o desempenho de Intranets de empresas que necessitem de diferenciação de serviços para seus usuários. Adicionalmente, o modelo pode ser adaptado para realizar diferenciação de serviços não para classe de usuários, mas sim para classe de aplicações. Uma aplicação de comércio eletrônico terá mais prioridade para ser atendida que uma aplicação de consulta no site da empresa.

A flexibilidade de alterar os parâmetros de número e tipos de usuários, carga de trabalho, velocidade das conexões entre os componentes, tipos de aplicações (estática/dinâmica) permitem que o modelo seja adaptado para outros modelos de ambiente *Web*.

7.7 Considerações Finais

Este capítulo discutiu a necessidade da construção de modelos, que permitam compreender a complexidade da *Web.* A avaliação de desempenho baseada em modelos apresenta-se como forma adequada no processo de avaliação de desempenho de sistemas *Web*. Foram apresentados os modelos de ambiente *Web* descritos por Menascé e Almeida (MENASCÉ *and* ALMEIDA 2002), os quais serviram de referência para o desenvolvimento deste trabalho. Finalmente, foi descrito o modelo construído no *OPNET Modeler* que serviu de base para todos os experimentos realizados nesse trabalho.

O modelo proposto pode servir como base para a implantação de infra-estrutura de ambiente *Web*, voltados ao atendimento diferenciado de clientes. O modelo não impõe nenhuma restrição ou organização específica, adaptando-se tanto a um sistema distribuído, quanto a uma arquitetura mono ou multiprocessada. A facilidade em se adequar todos os parâmetros do modelo (número de usuários, tipo de aplicação, congestionamento da rede, por exemplo) permite que ele seja utilizado desde a fase de planejamento.

A metodologia utilizada para a avaliação de desempenho neste trabalho segue a proposta de Jain (Jain, R., 1991), que é dividida em dez etapas. Todas as etapas foram descritas e detalhadas. Foi definido que as métricas de desempenho utilizadas são o tempo de resposta geral das requisições, o tempo de resposta para cada classe de usuário, o número total de requisições realizadas e o número de requisições por classe de usuário. Foram selecionados e descritos os quatro fatores: diferenciação na rede, servidor *cache*, congestionamento na rede e tipo de requisição considerada. Foi também descrita a carga de trabalho composta por requisições estáticas e dinâmicas utilizada nos experimentos.

Adicionalmente, foi descrito como os experimentos foram organizados para que se pudessem combinar todos os fatores da melhor maneira. O capítulo seguinte apresenta a análise dos resultados obtidos nas simulações dos experimentos descritos neste capítulo. Através das tabelas comparativas entre os cenários é possível verificar qual o melhor modelo para determinadas configurações de ambiente *Web*.

Capítulo

8

Análise das Simulações

8.1 Considerações Iniciais

Neste capítulo são apresentados e discutidos os principais resultados obtidos após as simulações dos diferentes cenários em suas configurações distintas. Foram analisados o tempo médio de reposta das requisições *HTTP*, o número total de *downloads* e de requisições canceladas, e a utilização do *link* entre os roteadores. Foram realizadas, em média, 39.100 requisições, com esse valor sofrendo pequenas variações nos diferentes cenários, porém, essa variação ocorreu dentro do intervalo de confiança estipulado.

Nas próximas seções serão analisados separadamente o:

- tempo médio de resposta geral para todas as requisições dos usuários;
- tempo médio de resposta para cada classe de usuário;
- tempo médio de resposta para as requisições estáticas;
- tempo médio de resposta para as requisições dinâmicas.

Os resultados são apresentados em forma de gráfico e com uma tabela comparativa entre os cenários. Através das tabelas é possível comparar um a um o desempenho do tempo de resposta das requisições entre os cenários.

8.2 Tempo de Resposta HTTP Geral

O tempo de resposta *HTTP* Geral é o resultado médio do tempo gasto de atendimento de todas as requisições solicitadas pelos usuários, tanto requisições de objetos estáticos como de objetos dinâmicos. A figura 8.1 ilustra o tempo médio de resposta *HTTP* Geral nos ambientes com 0%, 40%, 60% e 80% de congestionamento. Conforme pode ser observado, os cenários que utilizam servidor *cache* SC, CC e CDF apresentam um tempo de resposta *HTTP* menor que os cenários que não utilizam servidor *cache* (SS e CS). Comparando o cenário SS com o cenário SC, este apresenta, em média, um tempo de resposta 2% menor nos ambientes com 0% e 40% de congestionamento, 5% menor no ambiente com 60% de congestionamento e 95% menor no ambiente com 80% congestionamento. Em adição, os cenários CC e CDF apresentam melhor desempenho que o cenário SC quando a rede fica congestionada em 60% e 80%. Tal fato deve-se à quantidade de retransmissões devido aos descartes que ocorrem no roteador RC. Nos cenários com diferenciação de serviços praticamente não há descartes nos roteadores, pois as requisições ficam armazenadas em filas e não são descartadas diretamente como nos outros cenários (SS e SC). No ANEXO B é apresentado um trecho do *log* que apresenta os pedidos de retransmissões e cancelamentos solicitados pelos clientes.

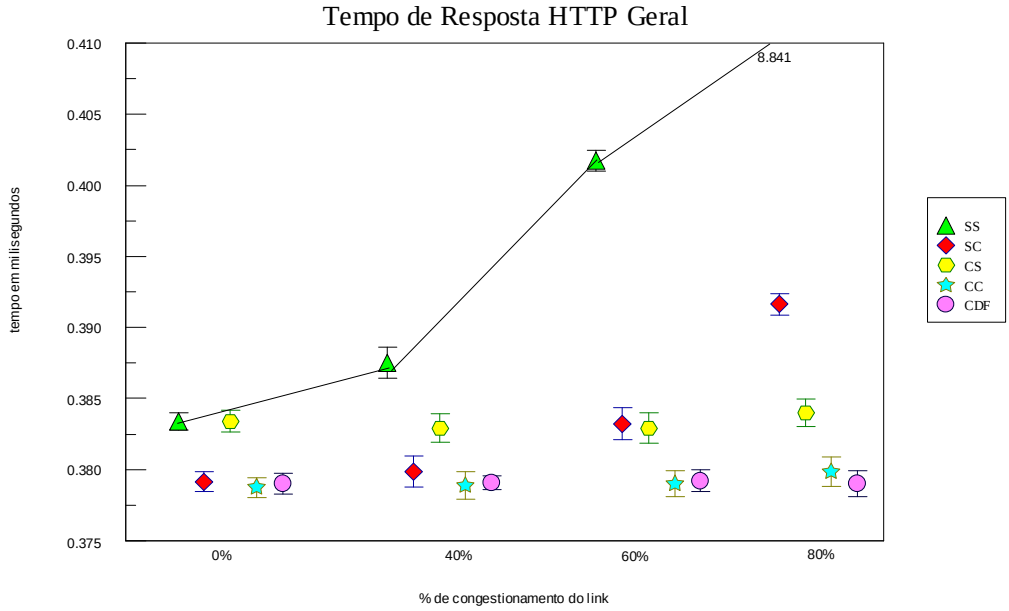


Figura 8.1 – Tempo de Resposta *HTTP* Geral

Para melhor visualização dos dados, as figuras 8.2, 8.3, 8.4 e 8.5 ilustram o tempo médio de resposta *HTTP* Geral, nos ambientes com 0%, 40%, 60% e 80% de congestionamento, separadamente. Já as tabelas 8.1, 8.2, 8.3 e 8.4 apresentam o tempo médio de resposta *HTTP* por cenário e a porcentagem da diferença existente entre eles.

Conforme pode ser observado na figura 8.2 e na tabela 8.1, no cenário sem congestionamento, o tempo médio de resposta dos cenários sem servidor *cache*, SS e CS, apresentam desempenho semelhante com uma diferença temporal de apenas 0,02 *ms*. Da mesma forma, os cenários com servidor *cache* SC, CC e CDF apresentam desempenho semelhante com diferenças de menos de 1,1%.

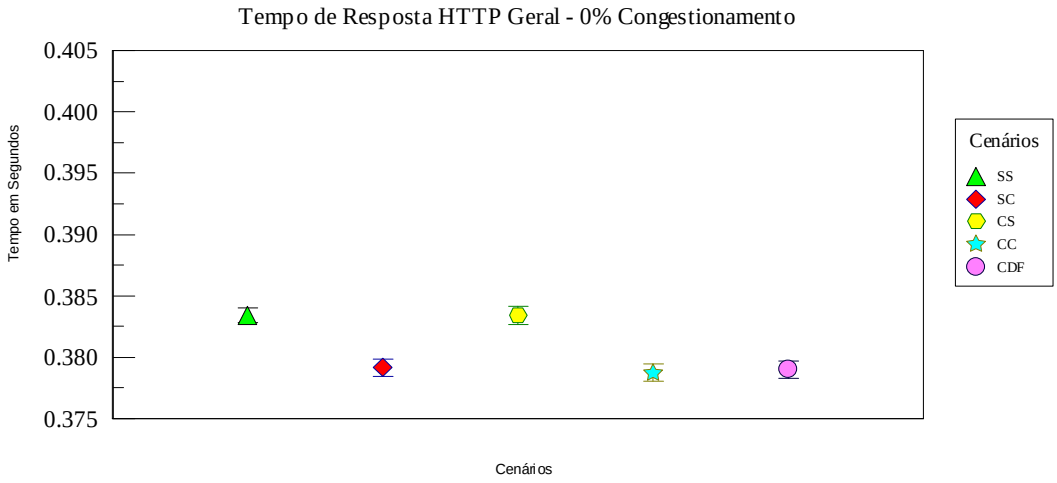


Figura 8.2 – Tempo de Resposta *HTTP* Geral com 0% de congestionamento na rede

Tabela 8.1 - Tempo de Resposta *HTTP* Geral 0% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,38339		1,119%	-0,004%	1,227%	1,162%
SC	0,37915	-1,106%		-1,110%	0,107%	0,043%
CS	0,38341	0,004%	1,123%		1,232%	1,166%
CC	0,37874	-1,213%	-0,107%	-1,217%		-0,065%
CDF	0,37899	-1,149%	-0,043%	-1,153%	0,065%	

A figura 8.3 e a tabela 8.2 apresentam os dados do cenário com 40% de congestionamento. Com o acréscimo de tráfego extra na rede já é possível verificar o aumento do tempo de resposta do cenário SS em relação ao cenário CS em 1,2%, ao CC em 2.28% e 2.229% ao CDF. Os cenários com diferenciação de serviços e servidor *cache* CC e CDF apresentam desempenho semelhante com uma diferença de apenas 0.2*ms*.

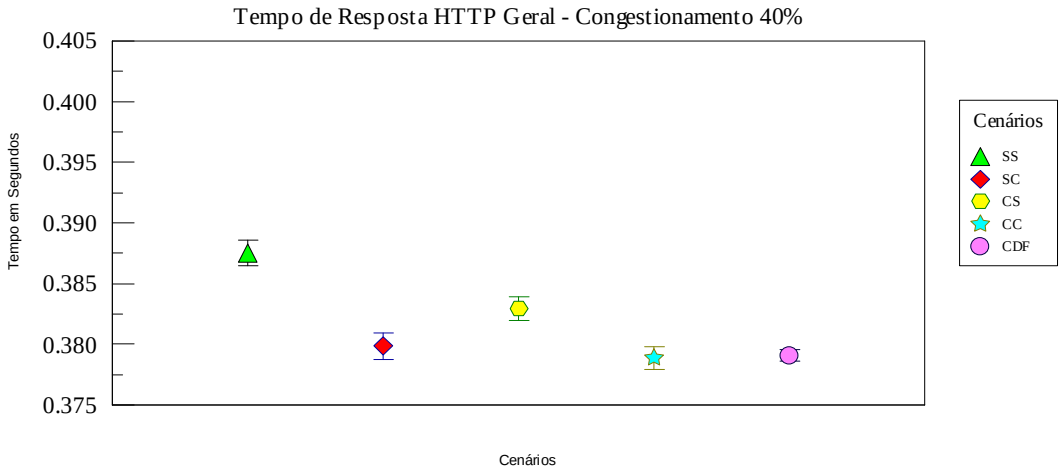


Figura 8.3 – Tempo de Resposta *HTTP* Geral com 40% de congestionamento na rede

Tabela 8.2 - Tempo de Resposta *HTTP* Geral com 40% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,38753		2,020%	1,203%	2,282%	2,229%
SC	0,37985	-1,980%		-0,802%	0,256%	0,205%
CS	0,38292	-1,188%	0,808%		1,067%	1,015%
CC	0,37888	-2,231%	-0,256%	-1,055%		-0,051%
CDF	0,37908	-2,181%	-0,204%	-1,004%	0,051%	

A figura 8.4 apresenta os dados do cenário com 60% de congestionamento. O desempenho do cenário SS é comprometido com o aumento do tráfego para 60%. A diferença para os outros cenários já varia de 4.80% a 6.00% conforme ilustrado na tabela 8.3. Os cenários SC e CS apresentam um desempenho semelhante com uma diferença de apenas 0.31*ms*.

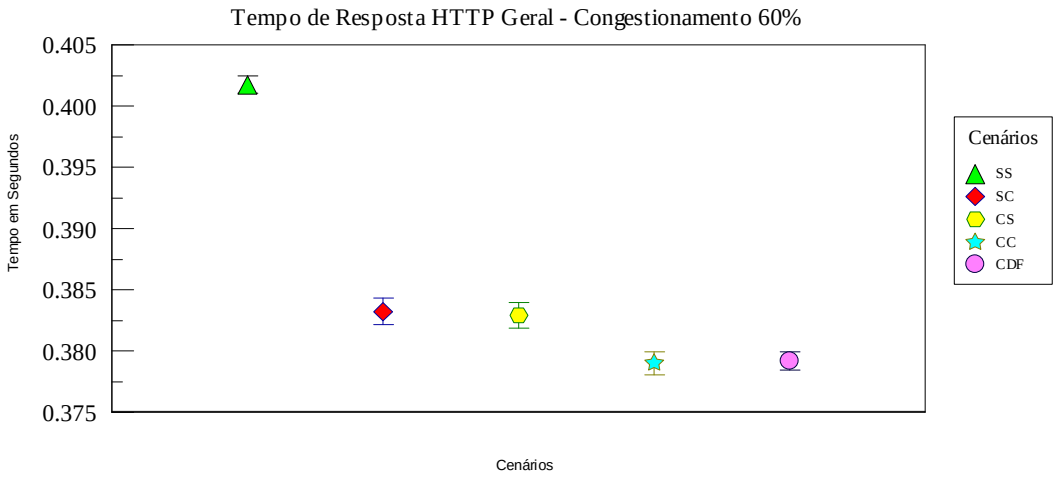


Figura 8.4 – Tempo de Resposta *HTTP* Geral com 60% de congestionamento na rede

Tabela 8.3 - Tempo de Resposta *HTTP* Geral com 60% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,40174		4,829%	4,913%	6,000%	5,940%
SC	0,38323	-4,606%		0,081%	1,117%	1,060%
CS	0,38292	-4,683%	-0,081%		1,036%	0,979%
CC	0,37900	-5,660%	-1,105%	-1,025%		-0,056%
CDF	0,37921	-5,607%	-1,049%	-0,969%	0,056%	

No cenário com 80% de congestionamento ilustrado na figura 8.5 e com os dados da tabela 8.6, é possível verificar a degradação do ambiente SS com um tempo de resposta de mais de 8s. Tal fato se deve ao grande número de retransmissões que ocorre nesse ambiente. No ambiente com diferenciação de serviços, CS, o número de requisições descartadas é bem menor que no cenário SS, pois o roteador (RC) tem uma capacidade de armazenamento em fila dos pedidos maior que o cenário SS.

O cenário CDF apresenta o melhor desempenho com um tempo de resposta *HTTP* 3.21% menor que o cenário SC, 1.29% menor que CS e 0.21% menor que o cenário CC.

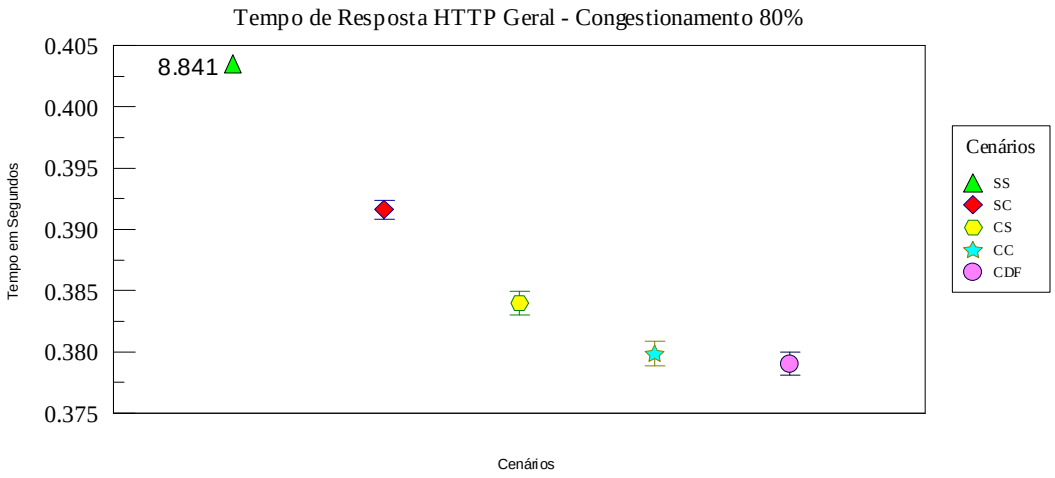


Figura 8.5 – Tempo de Resposta *HTTP* Geral com 80% de congestionamento na rede

Tabela 8.4 - Tempo de Resposta *HTTP* Geral com 80% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	8,84099		2158%	2202,4%	2227,4%	2232,5%
SC	0,39162	-95,570%		1,988%	3,095%	3,321%
CS	0,38399	-95,657%	-1,949%		1,086%	1,307%
CC	0,37986	-95,703%	-3,002%	-1,074%		0,219%
CDF	0,37903	-95,713%	-3,214%	-1,290%	-0,219%	

8.3 Tempo de Resposta Requisições Estáticas

O tempo de resposta *HTTP* das requisições estáticas é o resultado médio do tempo gasto de atendimento somente das requisições estáticas descritas no capítulo anterior. As figuras 8.6, 8.7, 8.8 e 8.9 ilustram, respectivamente, o tempo médio de resposta *HTTP* das requisições estáticas, por classe de usuário, nos ambientes com 0%, 40%, 60% e 80% de congestionamento. As tabelas 8.5, 8.6, 8.7 e 8.8 apresentam o tempo médio de

resposta de todas as classes por cenário e a porcentagem da diferença existente entre eles.

Conforme se pode observar na figura 8.6 e na tabela 8.5, os ambientes SS, SC e CC apresentam o mesmo tempo de resposta para todas as classes de usuários. Os cenários com servidor *cache* apresentam um desempenho muito bom se comparados aos cenários SS e CS com tempos de resposta entre de 25% a 30% menores. Tal fato se deve a capacidade do servidor *cache* armazenar objetos estáticos diminuindo o tempo de recuperação desses objetos e o tráfego de informações na rede.

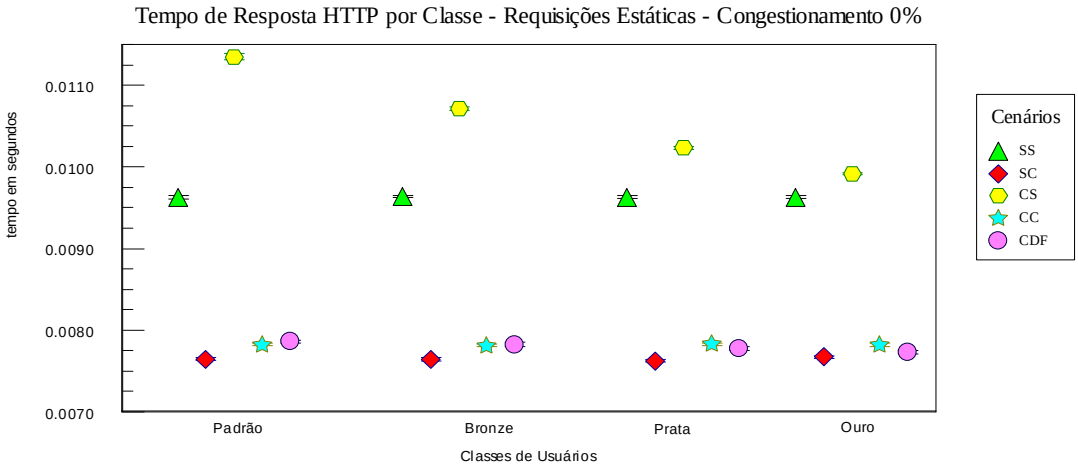


Figura 8.6 – Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 0% de congestionamento na rede

Tabela 8.5 - Tempo de Resposta *HTTP* Requisições Estáticas com 0% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,0096313		25,956%	-8,72%	23,13%	23,49%
SC	0,0076466	-20,61%		-27,53%	-2,24%	-1,96%
CS	0,0105515	9,55%	37,990%		34,90%	35,29%
CC	0,0078218	-18,79%	2,292%	-25,87%		0,29%
CDF	0,0077994	-19,02%	1,998%	-26,08%	-0,29%	

Em relação à figura 8.7 e à tabela 8.6 destaca-se a degradação do ambiente SS que apresenta os piores tempos de resposta. Destacam-se as diferenças entre os cenários SC e CS: no ambiente sem congestionamento o cenário SC é 27.53% mais rápido que o cenário CS, já no ambiente com 40% de congestionamento essa diferença é de apenas 8,88%.

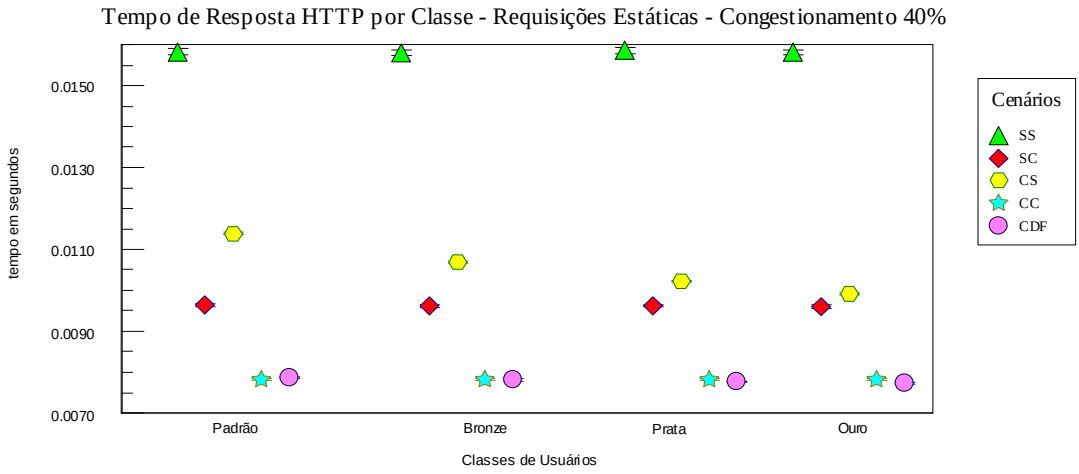


Figura 8.7 –Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 40% de congestionamento na rede

Tabela 8.6 - Tempo de Resposta *HTTP* Requisições Estáticas com 40% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,0158261		64,549%	49,94%	102,31%	103,02%
SC	0,0096178	-39,23%		-8,88%	22,95%	23,38%
CS	0,0105551	-33,31%	9,745%		34,93%	35,40%
CC	0,0078227	-50,57%	-18,665%	-25,89%		0,35%
CDF	0,0077955	-50,74%	-18,948%	-26,15%	-0,35%	

A figura 8.8 e a tabela 8.7 ilustram o cenário com 60% de congestionamento. A degradação do ambiente SS continua aumentando. Nessa configuração, o cenário CS já apresenta melhor tempo de resposta que o cenário SC, aproximadamente 20% menor. Tal fato se justifica pela capacidade de armazenamento das requisições nos roteadores dos cenários com diferenciação de serviços, fato que não ocorre nos cenários sem diferenciação.

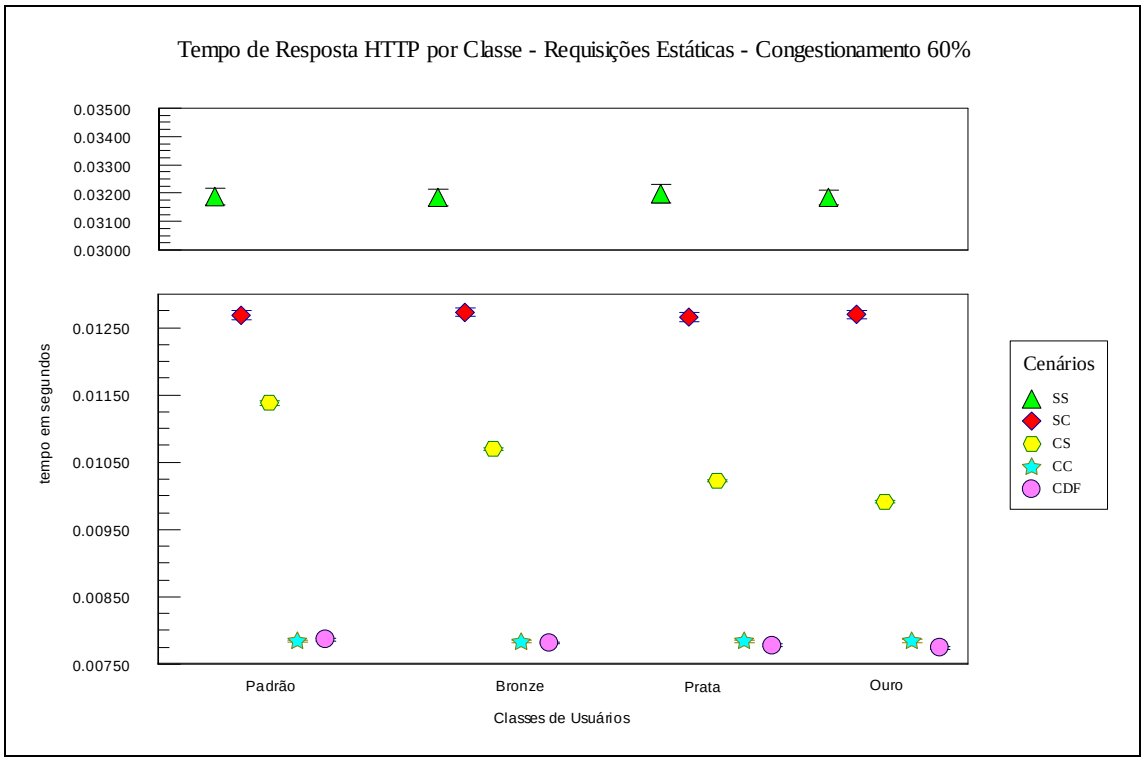


Figura 8.8 – Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 60% de congestionamento na rede

Tabela 8.7 - Tempo de Resposta *HTTP* Requisições Estáticas com 60% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,0318850		151,224%	202,08%	306,75%	308,73%
SC	0,0126918	-60,19%		20,24%	61,91%	62,70%
CS	0,0105551	-66,90%	-16,835%		34,65%	35,31%
CC	0,0078389	-75,42%	-38,237%	-25,73%		0,49%
CDF	0,0078009	-75,53%	-38,536%	-26,09%	-0,48%	

A figura 8.9 e a tabela 8.8 descrevem o resultado do ambiente com 80% de congestionamento. O ambiente SS apresenta o pior desempenho médio, com um tempo de resposta de mais de 10s. O cenário CDF apresenta o menor tempo médio de reposta para as classes com maior prioridade outro, prata e bronze. Para os usuários padrão o cenário CC apresenta tempo menor que o CDF. Destaca-se que todas as requisições do cenário CC são realizadas na classe padrão.

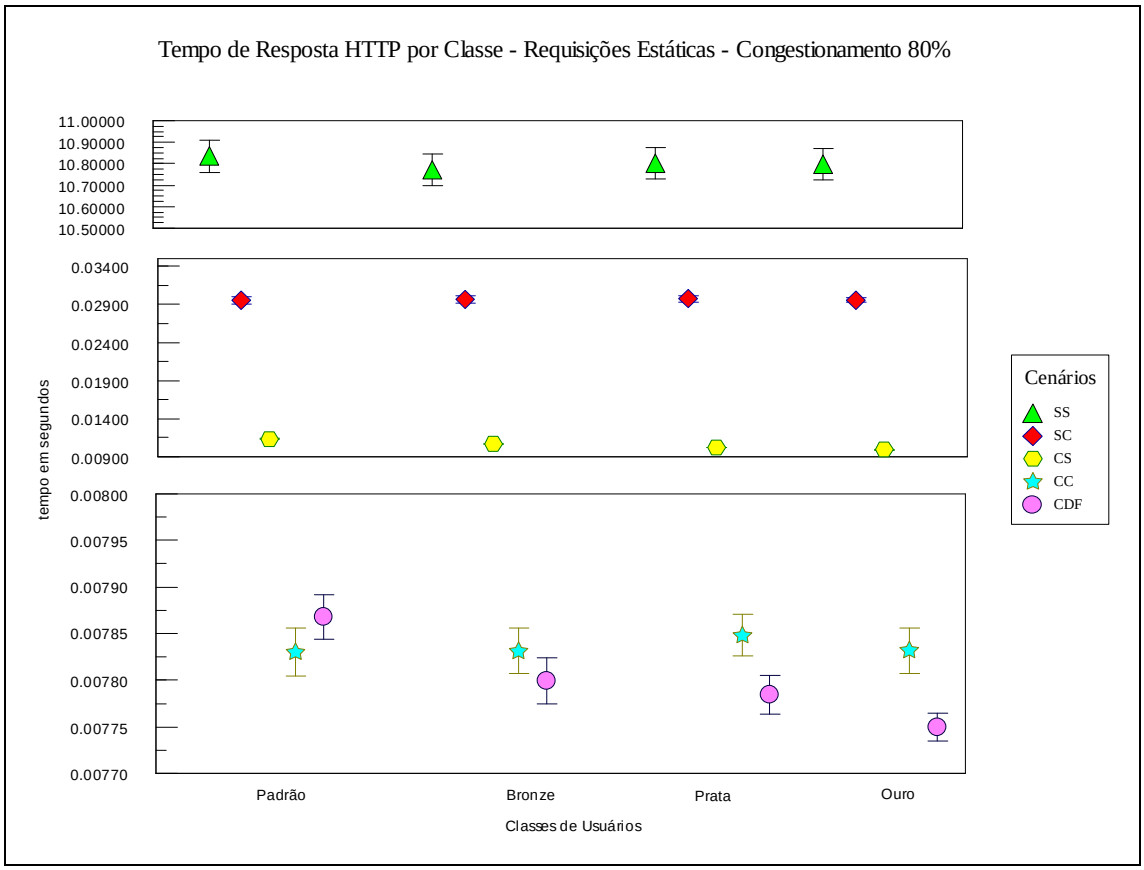


Figura 8.9 – Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 80% de congestionamento na rede

Tabela 8.8 - Tempo de Resposta *HTTP* Requisições Estáticas com 80% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	10,801248		36354,803%	102207%	137749,33%	138371,3%
SC	0,0296292	-99,73%		180,6%	278,14%	279,84%
CS	0,0105577	-99,90%	-64,367%		34,74%	35,35%
CC	0,0078355	-99,93%	-73,555%	-25,78%		0,45%
CDF	0,0078004	-99,93%	-73,673%	-26,12%	-0,45%	

De uma maneira geral, em todos os ambientes com as diferentes configurações de congestionamento, o desempenho dos cenários com servidor *cache* é muito superior aos cenários que não possuem esse recurso (SS e CS). Verifica-se também que o tempo de resposta nos cenários com diferenciação e com *cache* (CC e CDF) são em média 25% mais rápidos que o cenário com diferenciação e sem *cache* (CS). Além disso, observa-se que o ambiente CC apresenta o mesmo tempo de resposta para todas as classes de usuários. Tal fato se justifica por toda a requisição realizada do servidor *cache* tradicional, para o servidor *Web*, ser uma requisição to tipo *best-effort* (padrão).

8.4 Tempo de Resposta para Requisições Dinâmicas

O tempo de resposta *HTTP* das requisições dinâmicas é o resultado médio do tempo gasto de atendimento somente das requisições dinâmicas, conforme descritas no capítulo anterior. As figuras 8.10, 8.11, 8.12 e 8.13 ilustram, respectivamente, o tempo médio de resposta *HTTP* das requisições dinâmicas por classe de usuário, nos ambientes com 0%, 40%, 60% e 80% de congestionamento. As tabelas 8.5, 8.6, 8.7 e 8.8 apresentam o tempo médio de resposta de todas as classes por cenário e a porcentagem da diferença existente entre eles.

Conforme pode ser observado na Figura 8.10, o cenário sem *cache* SS possui praticamente o mesmo tempo de resposta para todas as classes de usuários, 0.785s. Os usuários do cenário CS possuem tempo de resposta maior que o cenário SS para todas as classes de usuários, os valores são 0.12% maiores na classe padrão, 0.077% na classe bronze, 0.036% na prata e 0.005% na classe ouro. Os cenários com servidor *cache* SC, CC e CDF apresentam praticamente o mesmo tempo médio de resposta para as páginas dinâmicas 0.78093, 0.78090 e 0.78084, respectivamente.

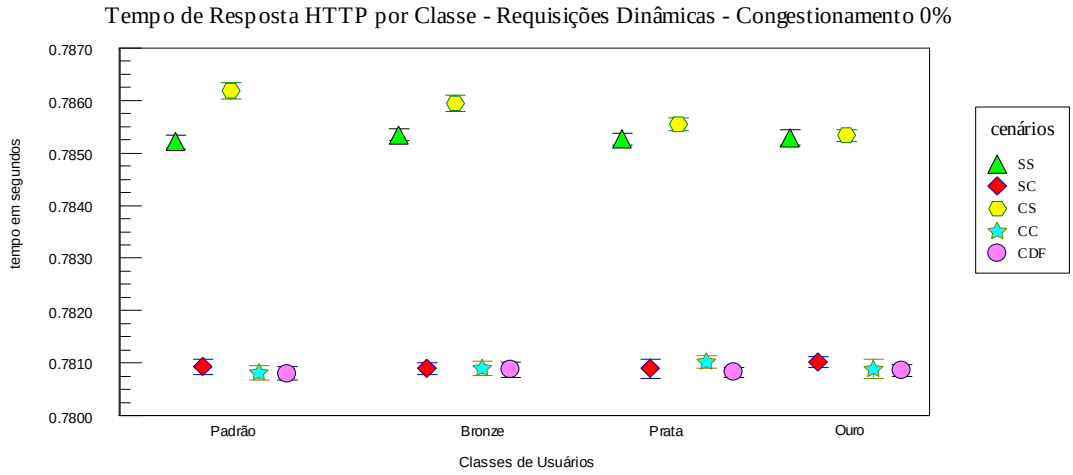


Figura 8.10 – Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 0% de congestionamento na rede

Conforme podem ser observados na tabela 8.9, os cenários com diferenciação de serviços apresentam um desempenho um pouco inferior aos cenários sem esse recurso. Nesse caso, o tempo gasto para se realizar a diferenciação de serviço nos roteadores é prejudicial ao desempenho desses cenários.

Tabela 8.9 - Tempo de Resposta *HTTP* Requisições Dinâmicas com 0% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,7852786		0,556%	-0,061%	0,560%	0,568%
SC	0,7809327	-0,553%		-0,614%	0,004%	0,012%
CS	0,7857545	0,061%	0,617%		0,621%	0,629%
CC	0,7809032	-0,557%	-0,004%	-0,617%		0,008%
CDF	0,7808400	-0,565%	-0,012%	-0,625%	-0,008%	

A figura 8.11 e a tabela 8.10 apresentam os resultados do ambiente com 40% de congestionamento. O cenário CDF apresenta o menor tempo de resposta para todas as classes de usuários. Os cenários SC e CC apresentam praticamente o mesmo desempenho com uma diferença inferior a 0.007%.

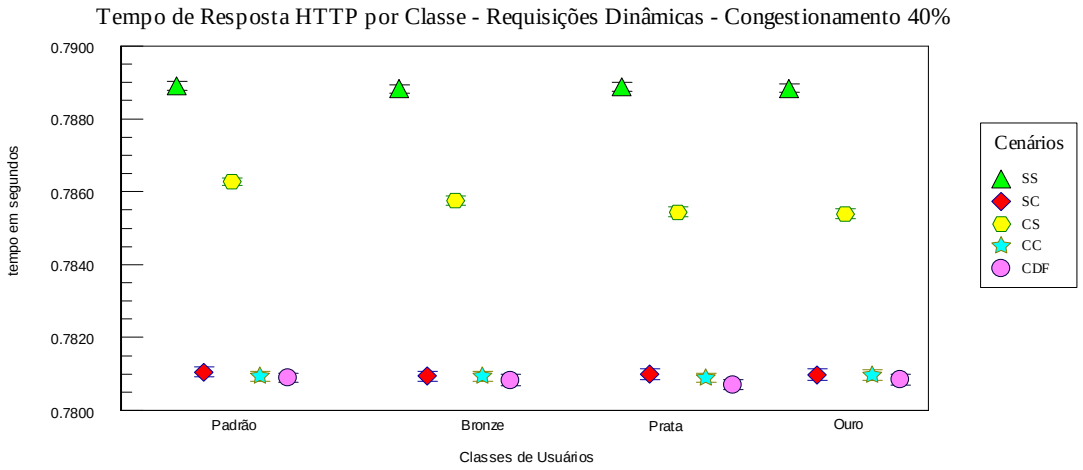


Figura 8.11 – Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 40% de congestionamento na rede

Tabela 8.10 - Tempo de Resposta *HTTP* Requisições Dinâmicas com 40% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,7888654		1,008%	0,401%	1,016%	1,030%
SC	0,7809911	-0,998%		-0,601%	0,007%	0,022%
CS	0,7857136	-0,400%	0,605%		0,612%	0,627%
CC	0,7809339	-1,005%	-0,007%	-0,608%		0,015%
CDF	0,7808191	-1,020%	-0,022%	-0,623%	-0,015%	

A figura 8.12 e a tabela 8.11 ilustram o cenário com 60% de congestionamento. O cenário SS apresenta o pior desempenho. O desempenho dos cenários SC e CC são praticamente o mesmo com uma diferença mínima de 0.008%.

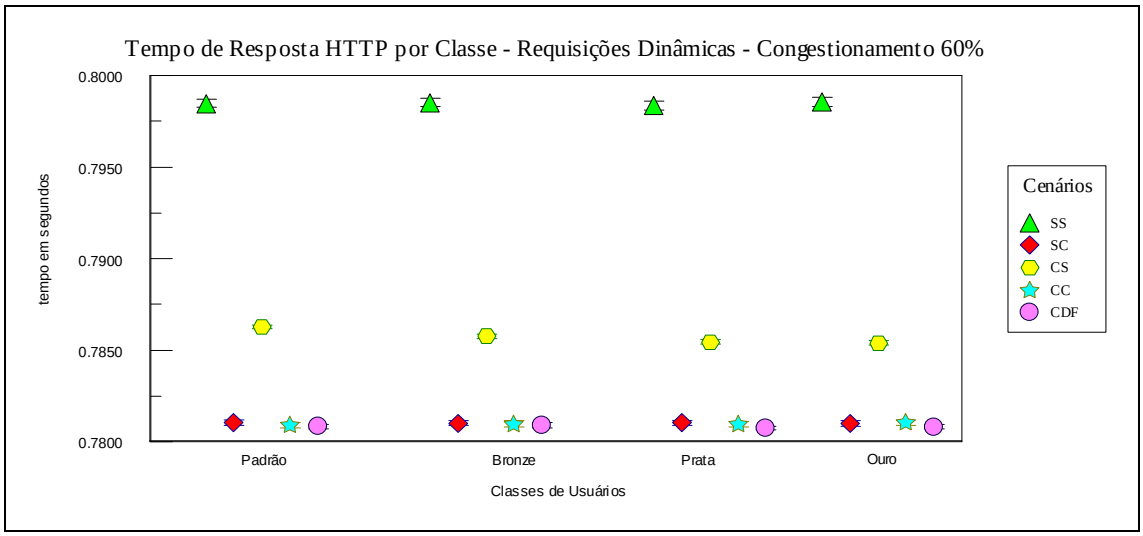


Figura 8.12 – Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 60% de congestionamento na rede

Tabela 8.11 - Tempo de Resposta *HTTP* Requisições Dinâmicas com 60% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	0,7984827		2,236%	1,625%	2,244%	2,262%
SC	0,7810204	-2,187%		-0,597%	0,008%	0,025%
CS	0,7857136	-1,599%	0,601%		0,609%	0,627%
CC	0,7809557	-2,195%	-0,008%	-0,606%		0,017%
CDF	0,7808218	-2,212%	-0,025%	-0,623%	-0,017%	

A figura 8.13 e a tabela 8.12 descrevem o resultado do ambiente com 80% de congestionamento. O ambiente SS apresenta o pior desempenho médio, com um tempo de resposta de mais de 7s. O cenário CDF apresenta o menor tempo médio de reposta.

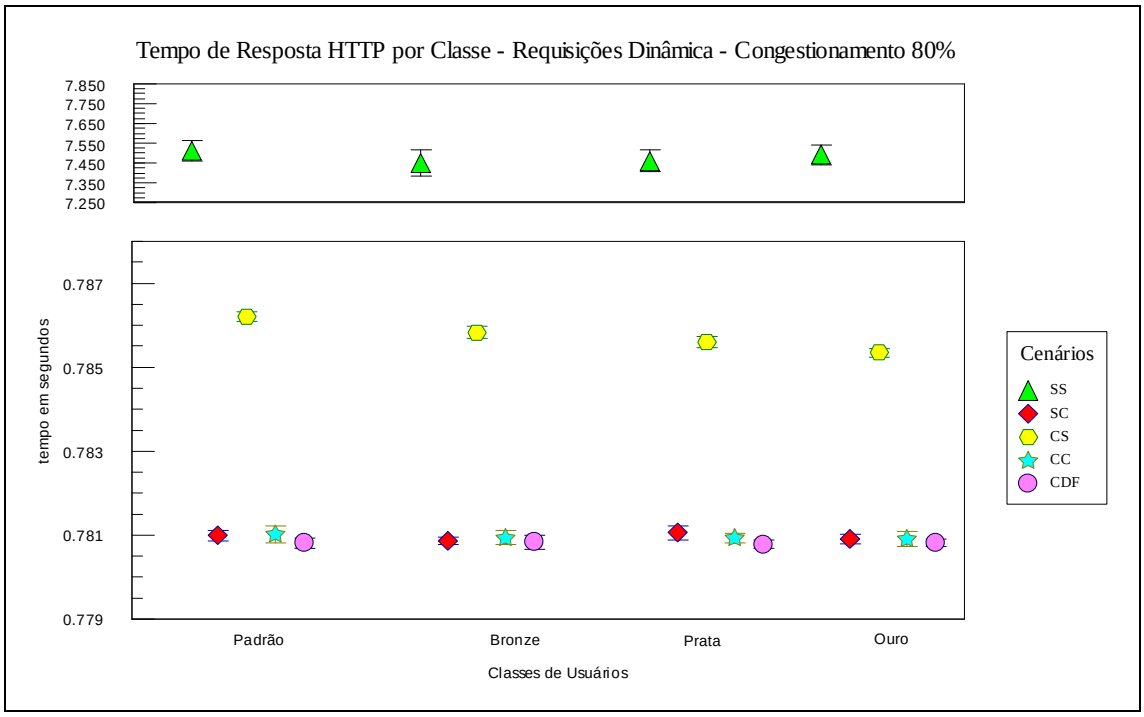


Figura 8.13 – Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 80% de congestionamento na rede

Tabela 8.12 - Tempo de Resposta *HTTP* Requisições Dinâmicas com 80% de congestionamento na rede

		% diferença entre os cenários				
Cenário	**Tempo Médio (s)**	**SS**	**SC**	**CS**	**CC**	**CDF**
SS	7,4790515		857,682%	851,8%	857,7%	857,9%
SC	0,7809534	-89,558%		-0,610%	0,001%	0,018%
CS	0,7857452	-89,494%	0,614%		0,614%	0,632%
CC	0,7809489	-89,558%	-0,001%	-0,610%		0,018%
CDF	0,7808105	-89,560%	-0,018%	-0,628%	-0,018%	

Verifica-se que os cenários que possuem servidores *cache* apresentam um tempo de resposta *HTTP*, em média, de **0.5%** melhor que os cenários sem servidor *cache*. Apesar de nenhuma requisição dinâmica ser atendida pelo *cache*, o tempo de resposta ainda é um pouco inferior devido ao tráfego da rede ser menor nesses cenários, pois algumas requisições estáticas são atendidas pelo servidor *cache* e não trafegam pela rede.

Vale destacar que para as requisições dinâmicas a diferença do tempo de resposta entre as classes de usuário é muito pequena. Tal fato é explicado pelo tempo de processamento das requisições no servidor *Web* serem o mesmo para todas as classes.

8.5 Análise Estatística

Neste trabalho optou-se por apresentar os resultados com todos os níveis propostos e por analisar esses resultados considerando-se dois níveis para cada fator e utilizar um projeto fatorial 2^k. A seguir são descrito os resultados para o tempo de resposta *HTTP* geral.

8.5.1 Tempo de Resposta HTTP Geral

Para a análise do tempo de resposta *HTTP* Geral foram considerados três fatores com dois níveis cada, descritos na tabela 8.13.

Tabela 8.13- Descrição dos Fatores e seus Níveis

Fator	Descrição	Nível 1	Nível -1
A	Rede	Sem diferenciação	Com diferenciação
B	*Cache*	Sem servidor *Cache*	Com servidor *cache*
C	Congestionamento da Rede	40%	60%

Considerando-se um estudo com três fatores, A, B e C, cada um deles com dois níveis, um projeto fatorial completo terá um total de $2^3 = 8$ experimentos, que correspondem às oito linhas de uma tabela padronizada de sinais, como a tabela 8.14 abaixo.

Tabela 8.14- Tabela de Sinais

Experimento	Média	A	B	C	AB	AC	BC	ABC	R
1	1	-1	-1	-1	1	1	1	-1	0,379032078
2	1	1	-1	-1	-1	-1	1	1	0,38323287
3	1	-1	1	-1	-1	1	-1	1	0,382924354
4	1	1	1	-1	1	-1	-1	-1	0,401737876
5	1	-1	-1	1	1	-1	-1	1	0,379078259
6	1	1	-1	1	-1	1	-1	-1	0,379854398
7	1	-1	1	1	-1	-1	1	-1	0,382924354
8	1	1	1	1	1	1	1	1	0,387529119

A coluna R corresponde aos resultados obtidos nas simulações. Após a inclusão dos resultados, pode-se então realizar a operação de multiplicação da matriz transposta dos sinais pelo vetor de resposta R. Desta maneira, obtém-se:

Matriz Transposta									R		MULTIPLICADA
1	1	1	1	1	1	1	1		0,379032078		3,076313308
-1	1	-1	1	-1	1	-1	1		0,38323287		0,028395218
-1	-1	1	1	-1	-1	1	1		0,382924354		0,033918098
-1	-1	-1	-1	1	1	1	1	*	0,401737876	=	-0,01754105
1	-1	-1	1	1	-1	-1	1		0,379078259		0,018441356
1	-1	1	-1	-1	1	-1	1		0,379854398		-0,01763341
1	1	-1	-1	-1	-1	1	1		0,382924354		-0,01087647
-1	1	1	-1	1	-1	-1	1		0,387529119		-0,0107841

Cada valor obtido deverá ser dividido por 2^k. Assim, para o cálculo final dos efeitos e da média, divide-se cada um dos elementos do vetor por $2^k = 8$. O vetor resposta terá os seguintes valores:

Tabela 8.15 - Vetor de Resposta

Média	0,384539
A	0,003549
B	0,00424
C	-0,00219
AB	0,002305
AC	-0,0022
BC	-0,00136
ABC	-0,00135

Após o cálculo do vetor de resposta é possível calcular a Variação Total de R por meio da chamada Soma dos Quadrados Totais (SQT). Tal cálculo compreende uma soma das contribuições de cada um dos fatores e de suas interações. Calculados a SQT é possível determinar a contribuição de cada fator no tempo de resposta *HTTP* Geral. Os resultados encontrados nos permitem concluir que:

- O Fator A (rede) contribui com 25,60% da variação do tempo de resposta geral;
- O Fator B (*cache*) contribui com 36,52% da variação;
- O Fator C (congestionamento) contribui com 9,77%;
- As interações entre dois fatores contribuem com 10,80% (interação AB), 9,87% (interação AC) e 3,76% (interação BC);
- A interação entre os três fatores ABC contribui com apenas 3,69%.

8.6 Considerações Finais

Este capítulo apresentou a análise dos principais resultados encontrados após a realização das simulações. Foram apresentados e discutidos os resultados obtidos em relação ao tempo médio de resposta das requisições *HTTP*.

Separadamente, foi também analisado o tempo médio de resposta geral para todas as requisições, o tempo médio para cada classe de usuário e o tempo médio para as requisições estáticas e para as requisições dinâmicas. Os dados foram apresentados em formas de gráficos e tabelas, o que permite a comparação entre os cinco diferentes cenários: SS, SC, CS, CC e CDF; a comparação entre os ambientes com 0%, 40%, 60% e 80% de congestionamento; e a comparação entre as diferentes classes de usuários: padrão, bronze, prata e ouro.

As soluções para melhoria de desempenho testadas neste trabalho mostram que soluções integradas para a melhoria de *QoS* apresentam resultados positivos. Analisando-se os dados obtidos das simulações, podem-se destacar:

- A utilização de servidores *cache* diminui o tráfego gerado na rede aumentando a disponibilidade desse recurso, para aplicações estáticas. Já para aplicações dinâmicas os servidores *cache* não surtem o efeito desejado, pois os objetos dinâmicos não ficam armazenados nos servidores *cache* e sempre devem ser recuperados do servidor *Web*;
- A utilização de servidores *cache* diminui em aproximadamente **24%** o tempo de resposta *HTTP* nas requisições estáticas. Já para as requisições dinâmicas, o servidor *cache* não teve influência no tempo de resposta *HTTP*, pois todas as requisições são atendidas somente pelo servidor *Web*;
- A utilização de diferenciação de serviço apresenta bons resultados quando se necessita realizar o tratamento diferenciado para classes de usuários. O tempo de resposta *HTTP* para os usuários com maior prioridade é menor. Porém, o atendimento aos usuários padrão fica prejudicado.
- A utilização de servidores *cache* sem suporte à diferenciação de serviço em ambiente com diferenciação de serviço, cenário CC, elimina os esforços de priorizar classes de usuários, pois todas as requisições do servidor *cache* para o servidor *Web* são realizadas na mesma classe de usuário padrão;
- O cenário CDF, que possui diferenciação de serviço e servidor *cache* com suporte à diferenciação, apresentou os melhores resultados em relação ao tempo de resposta *HTTP* para as classes de usuário com maior prioridade ouro e prata, em média, respectivamente, 4,12% e 1,95% mais rápidas que o cenário CC. Em contrapartida, as classes bronze e padrão são 1,15% e 4,72% mais lentas.

O capítulo seguinte apresenta as conclusões e principais contribuições desta tese.

Capítulo

9

Conclusão

9.1 Considerações Iniciais

A *Web* se tornou o maior e mais acessado serviço da Internet. Esse desenvolvimento ocorreu de maneira espontânea devido a algumas facilidades, tais como interatividade, conectividade, redução do custo de acesso ao sistema, número de serviços oferecidos, *etc*. Esses fatores permitiram o crescimento acelerado e a popularização da *Web*, gerando uma sobrecarga na Internet e nos servidores, acarretando maiores tempos de resposta às requisições, bem como degradação dos sistemas e serviços. Quando são feitos pedidos para servidores em conexões lentas, existe geralmente uma demora considerável na recuperação de objetos remotos. Além disso, a alta taxa de transferência de objetos pela rede leva a um aumento de tráfego que acaba reduzindo a largura de banda disponível e, também, introduzindo atrasos perceptíveis aos usuários.

Com esse crescimento o modelo de serviço do tipo *best-effort* da Internet, tem dado claros sinais de estrangulamento. A introdução de *QoS* na Internet se tornou uma necessidade e uma exigência das empresas que oferecem serviços pela *Web*. *QoS* é necessária nos diferentes níveis de estrutura da *Web*, tais como no nível de rede e de aplicação. Soluções desconexas de *QoS* podem não gerar o efeito desejado, pois existe a necessidade de que todos os elementos da rede permitam esse controle diferenciado, não existindo pontos de estrangulamento que possam degradar o sistema.

A finalidade deste trabalho de doutorado foi buscar um melhor entendimento dos aspectos relacionados à utilização de mecanismos de diferenciação de serviços na *Web*

em todos os pontos do sistema, desde o usuário até os servidores. O uso de modelagem como ferramenta de análise permitiu abstrair as principais características do ambiente *Web*, com ênfase na diferenciação de serviços nos diferentes elementos desse ambiente.

Inicialmente, fez-se uma revisão sobre diversos tópicos pertinentes à área. A infra-estrutura de Internet foi discutida no capítulo 2, destacando-se os principais aspectos que lhe dão suporte, especialmente os protocolos *TCP/IP* e *HTTP*. No capítulo 3, discutiram-se as limitações do modelo atual de serviços da Internet. Foram apresentados conceitos de qualidade de serviço e detalhadas as arquiteturas de serviços integrados e diferenciados, com ênfase nesta última.

O capítulo 4 apresentou os principais conceitos envolvidos com servidores *cache* na *Web* e como sua utilização pode auxiliar para melhorar o desempenho do ambiente *Web*. Adicionalmente, foi descrito nesse capítulo um servidor *cache* proposto com suporte à diferenciação de serviços, chamado *cache* CDF.

No capítulo 5 foi apresentada uma visão geral do processo de avaliação de desempenho de sistemas, com ênfase na modelagem, com solução por meio de simulação. Neste trabalho foi escolhido o ambiente *OPNET* para o desenvolvimento dos modelos e das simulações. A dificuldade e os estudos relacionados, bem como a caracterização da carga de trabalho da *Web*, foram discutidos no capítulo 6.

O capítulo 7 apresentou a necessidade da construção de modelos, que permitam compreender a complexidade da *Web*. Foram apresentados os modelos de ambiente *Web* descritos na literatura e a descrição do modelo construído no *OPNET Modeler*, que serviu de base para todos os experimentos realizados neste trabalho. A análise dos principais resultados obtidos nas simulações foi apresentada no capítulo 8. Os dados foram dispostos em formas de gráficos e tabelas, o que permitiu a comparação de todos os cenários e todas as classes de usuários.

9.2 Principais Resultados e Contribuições

Os principais resultados deste trabalho são relacionados a seguir, destacando-se as contribuições relevantes para a área:

- **Revisão Bibliográfica**

  Foi feita uma revisão crítica da bibliografia existente, com o objetivo de contextualizar o presente trabalho e de relatar os mais recentes avanços obtidos na área. Destaca-se especialmente o levantamento feito sobre o fornecimento de serviços diferenciados, em nível de rede e na utilização de servidores *cache* na *Web*. Esta revisão constitui uma contribuição, no sentido de que aglutina muitas informações importantes e que se encontram espalhadas em inúmeras publicações científicas. Os futuros trabalhos podem partir da síntese apresentada nesta tese, facilitando o entendimento da área e o desenvolvimento de novas contribuições.

- **Caracterização da Carga de Trabalho**

  Foram estudadas diferentes abordagens para a caracterização da carga de trabalho para a simulação, tanto de forma sintética quanto usando traces de acesso a servidores *Web* reais. A dificuldade em se gerar modelos de carga de trabalho é grande, principalmente devido à diversidade de serviços e aplicações oferecidos através da *Web*. O acesso a traces de servidores *Web* é restrito e as informações desses *logs* são cada vez mais sigilosas. A alternativa adotada para a caracterização da carga de trabalho utilizada foi baseada em trabalhos realizados no LaSDPC.

- **Servidor *Cache* com suporte a Diferenciação de Serviços**

  Foi desenvolvido um servidor *cache* com suporte a diferenciação de serviços, denominado *cache* CDF. O objetivo principal desse servidor é o de avaliar o comportamento do ambiente com a diferenciação de serviços implementada em todos os componentes da rede. Constatou-se que esse servidor *cache* obteve os melhores resultados para as classes de usuário com maior prioridade. Este servidor constitui uma contribuição dentro deste trabalho, pois, complementa o cenário de diferenciação de serviços de forma positiva, fazendo com que os ganhos obtidos em outras etapas do sistema não sejam perdidos no momento da utilização do *cache.*

- **Modelo de ambiente *Web***

  Foi desenvolvido e implementado um modelo de ambiente *Web,* que possibilita avaliar o impacto da utilização de servidores *cache* e diferenciação de serviços, em nível de rede e em nível de aplicação. O modelo criado pode ser facilmente adaptado para diferentes aplicações e dimensões, conforme descrito no capítulo 7, seção 7.5. Outras características relevantes na concepção do modelo que podem ser citadas como contribuições relevantes desta tese de doutorado são:

    - facilidade para a realização de testes com outros tipos de carga de trabalho;
    - facilidade na alteração do número de usuários por classe no ambiente;
    - possibilidade de alteração das políticas de escalonamento nos roteadores;
    - reconfiguração dos parâmetros de Qualidade de Serviço;
    - realização de estudos comparativos entre os cenários com e sem Qualidade de Serviço ;
    - avaliação dos impactos da utilização de classes de usuários com diferentes níveis de prioridade no ambiente *Web;*

- facilidade para a verificação de qual dos cenários *Web* propostos pode ser mais adequados para a melhoria de desempenho de acordo com os parâmetros do ambiente;

- **Implementação Real dos Modelos dos ambientes *Web***

  Os modelos propostos neste trabalho podem ser adaptados e implantados no mundo real. Todos os equipamentos utilizados no modelo possuem características reais e foram avaliados e validados pelos fabricantes que os desenvolveram. Adicionalmente, é possível trocar o equipamento no modelo por outro de outro fabricante, desde que disponível na biblioteca de equipamentos, e avaliar o desempenho do ambiente com esse novo modelo. A partir dessas comparações é possível avaliar qual o melhor equipamento para uma determinada situação. Isto também constitui uma contribuição deste trabalho, uma vez que o estudo apresentado não se restringe a uma modelagem teórica, abordando aspectos bem próximos da realidade, constituindo um possível suporte de gerenciamento de sistemas *Web*.

- **Estudo Comparativo dos Diferentes Cenários**

  O estudo comparativo realizado entre os diferentes cenários pode ser utilizado rapidamente para verificar qual o modelo de ambiente *Web* que melhor se adapta às condições de um ambiente real estudado. Outras características relevantes observadas na análise dessas comparações e que podem ser citadas como contribuições relevantes desta tese de doutorado são

  - A utilização de servidores *cache* tradicionais diminui o tráfego gerado na rede e o tempo de resposta *HTTP* em 24%, para aplicações estáticas. Já para aplicações dinâmicas os servidores *cache* não surtem esse mesmo efeito;
  - A utilização de diferenciação de serviço apresenta bons resultados quando se necessita realizar o tratamento diferenciado para classes de usuários. O tempo de resposta *HTTP* para os usuários

com maior prioridade é menor. Porém, o atendimento aos usuários padrão fica prejudicado;

- A utilização de servidores *cache* sem suporte à diferenciação de serviço (tradicionais) em ambiente com diferenciação de serviço, cenário CC, elimina os esforços de priorizar classes de usuários, pois todas as requisições do servidor *cache* para o servidor *Web* são realizadas na mesma classe de usuário padrão;
- O cenário CDF, que possui diferenciação de serviço e servidor *cache* com suporte à diferenciação, apresentou os melhores resultados em relação ao tempo de resposta *HTTP* para as classes de usuário com maior prioridade ouro e prata, em média, respectivamente. Em contrapartida as classes bronze e padrão são mais lentas que nos outros cenários.

Assim, conclui-se que esta tese atingiu plenamente os objetivos a que se propôs, isto é, a investigação de alternativas para melhorar o desempenho de ambientes *Web*, avaliando o impacto da utilização de servidores *cache*, em ambientes com e sem diferenciação de serviços. O modelo proposto engloba os principais aspectos do ambiente *Web* de forma flexível, com todos os requisitos para os testes de qualidade de serviço implementados. Os resultados obtidos comprovam que *QoS* é necessária nos diferentes níveis de estrutura da *Web*, tais como nível de rede e nível de aplicação, pois, soluções desconexas de *QoS* podem não gerar o efeito desejado. Além disso, observou-se que o uso de servidores *cache* tradicionais, sem diferenciação de serviço, impõe restrições no que se refere a obtenção de *QoS*, sendo necessário também ter-se diferenciação implementada no *cache.*

Um dos maiores problemas para a implantação, monitoração e administração de ambientes *Web* é o dimensionamento adequado da infra-estrutura necessária para oferecer a qualidade de serviço exigida pelos usuários. A partir desses modelos é possível verificar o resultado que será obtido no ambiente quando ele for implementado, pois essa modelagem se aproxima dos resultados que serão obtidos no ambiente real. Essa facilidade auxiliará administradores de ambientes *Web* a realizarem a implantação e adequação de serviços de forma rápida, planejada e segura.

9.3 Trabalhos Futuros

O estudo do comportamento do ambiente *Web* visando a alternativas para melhorar o seu desempenho é complexo e com vários componentes interferindo diretamente nos resultados. A proposta do modelo do ambiente *Web* utilizando mecanismos de diferenciação de serviços em todos os pontos da rede, consiste em uma definição inicial, a qual pode ser ampliada de diversas formas, tanto no que se refere ao modelo, quanto à carga de trabalho utilizada, aos mecanismos de diferenciação de serviços empregados e à qualidade de serviço desejada para as aplicações. A seguir, são comentadas algumas sugestões para trabalhos futuros, organizadas por assunto, as quais servem também para demonstrar que o desenvolvimento desta tese abre possibilidade para uma linha de pesquisa ampla, permitindo seu prosseguimento através de trabalhos de iniciação científica, dissertações de mestrado e, possivelmente, outras teses de doutorado.

- **Aumentar a taxa de acerto do servidor *cache* para classes com maior prioridade**

  Neste trabalho a taxa de acerto do *cache* foi a mesma para todas as classes de usuários. Estudos para aumentar a taxa de acerto de classes mais prioritárias podem ser realizados. Pode-se diferenciar o tamanho do espaço utilizado para armazenar os objetos em *cache* de acordo com a classe de usuário solicitante. Adicionalmente, políticas diferentes de substituição dos objetos em *cache* podem também ser testadas.

- **Utilizar *Cache* no servidor *Web***

  Verificar o impacto da utilização de um *cache* no servidor *Web* armazenando páginas dinâmicas pré-processadas. Isso poderá diminuir o tempo de resposta das requisições e o processamento do servidor *Web*.

- **Avaliar outras políticas de formação de filas nos roteadores**

No modelo proposto os roteadores são os responsáveis por realizar a análise e a classificação dos pacotes de acordo com sua prioridade. O pacote é então encaminhado para uma fila de espera e escalonado, de forma a cumprir os requisitos de *QoS* da classe a que pertence. Neste trabalho foi utlizado somente o mecanismo de enfileiramento por prioridades, *Priority Queuing* (PQ). Outros mecanismos como *Weighted Fair Queuing* (WFQ) e *Class-Based Queuing* (CBQ) podem ser avaliados.

- **Analisar o modelo *Web* proposto com diferentes cargas de trabalho**

Analisar o impacto da utilização de diferentes cargas de trabalho como aplicações multimídia, vídeo-conferência, dentre outras.

- **Implementar alguns dos cenários estudados**

Realizar um estudo prático através da implantação de alguns dos cenários estudados com os mesmos parâmetros e aferir os resultados através da coleta de dados. Isso permitirá realizar um estudo comparativo com os dados obtidos durante as simulações.

- **Utilizar o modelo para avaliar o canal de retorno de TV Digital Interativa**

O modelo *Web* proposto pode ser utilizado para análise e planejamento de canais de retorno entre os clientes de TV Digital Interativa e os Provedores de Serviço de Internet (PSI) (SANTOS 2005) (SANTOS et al. 2007).

Bibliografia

ABRAMS, M.; STANDRIDGE C.R.; ABDULLA G.; WILLIAMS S.; FOX E.A.; "Caching Proxies: Limitations *and* Potentials." *Proceedings of the 4th International World-Wide Web Conference*, 1995.

ALMEIDA, J.; DABU M.; MANIKUTTY A.; CAO P.. "Providing differentiated levels of service in *Web* content hosting." *SIGMETRICS Workshop on Internet Server Performance*, 1998.

ARLITT, M. F.; WILLIAMSON C. L. "*Web* server workload characterization: the search for invariants." *Proceedings of the 1996 ACM SIGMETRICS international Conference on Measurement and Modeling of Computer Systems*, 1996.

ARLITT, M. F.; CHERKASOVA L.; DILLEY J.; FRIEDRICH R.J.; JIN T. Y. "Evaluating content management techniques for *Web* proxy *caches*." *ACM SIGMETRICS Perform. Eval.;* Março de 2000.

ARLITT, M. F.; FRIEDRICH R. J.; JIN T. Y. *Performance evaluation of Web proxy cache replacement policies.* Tech. rep. HPL-98-97(R.1), Palo Alto, CA: Hewlett-Packard Company, 1999.

ARLITT, M.; JIN T. "A workload characterization study of the 1998 World Cup *Web* site." *Network, IEEE* , 2000.

BANKS, J. *Handbook of Simulation.* Georgia, Atlanta: John Wiley & Sons, Inc.; 1998.

BERNARDO, M. "Simulation analysis of dynamic server selection algorithms for replicated *Web* services." *Proceedings. Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Agosto de 2001.

BERNERS-LEE, T.; FIELDING R.; FRYSTYK H. *Hypertext Transfer Protocol HTTP/1.0.* RFC, IETF , 1992.

BHATTI, N.; FRIEDRICH R.. "*Web* server support for tiered services." *IEEE Network*, 1999.

BLAKE, S.; BLACK D. L.; CARLSON M.; DAVIES E.; WANG Z.; WEISS W.;. “An Architecture for Differentiated Services.” *Internet RFC 2475.* Dezembro de 1998.

BRADEN, B.; ESTRIN D.; HERZOG S.; JAMIN S. "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification." *Internet RFC 2205.* Setembro 1997.

BRADEN, R.; CLARK D. D.; SHENKER S. "Integrated Services in the Internet Architecture: an Overview." *Internet RFC 1633.* Junho 1994.

BRUSCHI, S. M. “ASDA - Um Ambiente para Simulação Distribuída Automático.” *Tese (Doutorado).* São Carlos: ICMC-USP, Novembro de 2002.

CALZAROSSA, M.; SERAZZI G. "Workload characterization: a survey." *Proceedings of the IEEE*, 1993.

CARPENTER, B.E.; NICHOLS K. "Differentiated services in the Internet." *Proceedings of the IEEE, Volume 90*, Setembro 2002.

CETIC.br. “Pesquisa sobre o uso das Tecnologias da Informação e da Comunicação no Brasil.” *Centro de Estudos sobre as Tecnologias da Informação e da Comunicação.* julho de 2008. *HTTP*://www.cetic.br/ (acesso em julho de 2008).

CGI.br. “Comitê Gestor da Internet no Brasil.” *CGI.br.* 2008. www.cgi.br (acesso em julho de 2008).

CHANG, X. “Network Simulations with *OPNET*.” *Proceedings of the 1999 Winter Simulation Conference*, 1999.

CHEN, W.; MARTIN P.; HASSANEIN H. “Differentiated Caching of Dynamic Content using Effective Page Classification.” *IEEE International Conference on Performance, Computing, and Communications*, 2004 .

CHEN, W.; MARTIN P., HASSANEIN H.. “Caching dynamic content on the *Web*.” *IEEE Canadian Conference on Electrical and Computer Engineering - CCECE 2003*, 2003.

CHOI, H.; LIMB J. O. "A Behavioral Model of *Web* Traffic." *Proceedings of the Seventh Annual international Conference on Network Protocols* , 1999.

CHUNG, S.; LEE Y.S.. "Modeling *Web* applications using java *and* XML related technologies." *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, janeiro 2003.

COMER, D. E. *Internetworking with TCP/IP: Principles, Protocols and Architecture.* Prentice Hal, 2000.

EGGERT, L.; HEIDEMANN J. "Application-level differentiated services for *Web* servers." *World Wide Web Journal*, 1999: 133–142.

EL-GENDY, M.A.; BOSE A.; SHIN K.G.. "Evolution of the Internet *QoS and* support for soft real-time applications." *Proceedings of the IEEE Volume 91,*, julho 2003.

FELTEN, J.; GURBUZ O; OWEN H.; GROBMANN T.; KUSSMANN G; SCHROCK W. “Modeling, Simulation, *and* Verification of an Enterprise Network.” *Global Telecommunication Conference – Globecom*, 1999.

FIELDING, R.; *et al.* "Hypertext Transfer Protocol *HTTP*/1.1." *IETF RFC 2616.* 1999.

FLOYD, S.; PAXSON V. "Difficulties in Simulating the Internet." *IEEE/ACM Transactions on Networking, vol. 9*, agosto 2001.

FRANCÊS, C. R. L. *Stochastic feature charts: uma extensão estocástica para os statecharts.* Dissertação Mestrado Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 1998.

FRANCÊS, C. R. L.; SANTANA M. J.; SANTANA R. H. C.; ORLANDI R. C. G. S.. "Tools *and* Methodologies For Performance Evaluation of Distributed Computing Systems - A Comparison Study." *Proceedings of the 1997 Summer Computer Simulation Conference (SCSC'97)*, 1997.

FRANCÊS, C.R.L, SANTANA M.J.; SANTANA R.H.C.; VIJAYKUMAR N.L.; SOLON V. C.. "Performance Evaluation Based on *Statecharts* Specification using Markov Chians *and* smpl Simulation: a Case Study of a Distributed Environment with a File Server." *Eurosim 2001*, 2001.

GEVROS, P.; CROWCROFT J; KIRSTEIN P; BHATTI S. "Congestion control mechanisms *and* the best effort service model Network." *IEEE Network, Volume 15*, maio 2001: 16-26.

GUO, L.; CHEN S.; XIAO Z.; ZHANG X. "Analysis of multimedia workloads with implications for internet streaming." *Proceedings of the 14th international Conference on World Wide Web*, 2005.

HASSANEIN, H.; LIANG Z.; MARTIN P. "Performance Comparison of Alternative *Web* Caching Techniques. ." *Proceedings of the Seventh international Symposium on Computers and Communications - ISCC*, Julho de 2002.

HEINANEN, J.; BAKER F.; WEISS W.; WROCLAWSKI J. *Assured Forwarding PHB Group.* RFC 2597, IETF, 1999.

HOSANAGAR, K.; KRISHNAN R.; CHUANG J.; CHOUDHARY V. "Pricing *and* Resource Allocation in Caching Services with Multiple Levels of Quality of Service." *MANAGEMENT SCIENCE*, 2005.

ISC. "Internet Systems Consortium, Inc. ." *Internet Systems Consortium, Inc.* . julho de 2007. *HTTP*://www.isc.org (acesso em julho de 2008).

JACOBSON, V.; NICHOLS K.; PODURI K. *An Expedited Forwarding PHB.* RFC 2598, IETF, 1999.

JAIN, R. “The Art of Computer Systems Performance Analysis”. John Wiley, 1991.

KILGORE, R.A. “Simulation *Web* services with.net technologies .” *Proceedings of the Winter Simulation Conference* , Dezembro de 2002.

LEINER, B.M.; CERF V.G. "A brief history of the Internet." *Internet Society.* dezembro 2003. www.isoc.org.

LI, J.; FEN G.; CHEN J.; CHEN P. “Modeling *Web* application architecture with UML.” *Proceedings. 36th International Conference on Technology of Object-Oriented Languages and Systems, 2000. TOOLS 200*, 04 de novembro de 2000.

LIANG, Z.; HASSANEIN H.; MARTIN P. “Transparent Distributed *Web* Caching.” *Proceedings of the 26th Annual IEEE Conference on Local Computer Networks - LCN*, Novembro de 2001.

LUCIO, G. F.; FARRERA M. P.; JAMMEH E.; FLEURY M.; REED M. J. “*OPNET Modeler and* NS-2: Comparing the Accuracy Of Network Simulators for Packet-Level Analysis using a Network Testbed.” *WSEAS Transactions on Computers*, Julho de 2003.

MACDOUGALL, M.H.; “Simulating Computer Systems Techniques *and* Tools”, The MIT Press, 1987.

MENASCÉ, D.A.; ALMEIDA V.A.F. *Planejamento de Capacidade para Serviços na Web.* CAMPUS, 2002.

MENASCÉ, D.A.; ALMEIDA V.A.F., DOWDY L.W. *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems.* Prentice Hall, 1994.

MENAUD, J.M.; ISSARNY V.; BANATRE M. D. "Improving effectiveness of *Web* caching" *Recent Advances in Distributed Systems. Lecture Notes in Computer Science, vol. 1752.;* 2000.

MODESTO, M.; PEREIRA Jr A.R.; ZIVIANI N.; CASTILLO C.; BAEZA-YATES R. "Um Novo Retrato da *Web* Brasileira." *XXXII Seminário Integrado de Software e Hardware (SEMISH 05)*, julho de 2005.

MOORE, K. E.; BRENNAN J. E. "Petri Nets *and* Simulation: a Tutorial." *Proceeding of the 1995 Summer Computer Simulation Conference (SCSC'95),* 1995.

Network Simulator - ns-2. 2008. *HTTP*://nsnam.isi.edu/nsnam/ (acesso em 2008).

OMNeT++ Community. 2008. *HTTP*://www.omnetpp.org/ (acesso em 2008).

"*OPNET Modeler* – Accelerating Network R&D." *OPNET* . 2008. *HTTP*://www.*OPNET*.com (acesso em julho de 2008).

PEGDEN, C.D.; SHANNON R.E.; SADOWSKI R.P. *Introduction to Simulation using SIMAN*. MacGraw-Hill International Editions, 1995.

PETRI, C.A. "Communication with automata". Ph.D. dissertation. Rome Air Development Center, Rome, NY (1966).

PINHEIRO, J. C. "Avaliação de Políticas de Substituição de Objetos em *Cache*s na *Web*." *Dissertação de Mestrado.* São Carlos: USP/ ICMC, 2001.

PODLIPNIG, S.; BÖSZÖRMENYI L. "A survey of *Web cache* replacement strategies." *ACM Computing Sureys. Volume 35*, Dezembro 2003: 374-398.

RAO, G.; RAMAMURTHY B. "*DiffServ*er: Application level differentiated services for *Web* servers." *IEEE International Conference on Communications*, 2001.

SANTANA, M.J. *Advanced Filestore Architeture for a Multiple- Lan Distributed Computing System*. Phd thesis, University of Southamptom, 1990, 1990a.

SANTANA, R.H.C. *Performance Evaluation of Lan-Based File Server*. Phd thesis, University of Southamptom, 1990, 1990b.

SANTANA, R.H.C.; SANTANA M.J.; BARBATO A.K.; TRALDI O.A. *Avaliação de Desempenho: Modelagem e Simulação.* Relatório Técnico, São Carlos: Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2004.

SANTOS JUNIOR, J. B. NEW INTERACTION CRITERIA FOR INTERACTIVE DIGITAL TELEVISION ENVIRONMENTS: AN APPROACH BASED ON CONTEXT-AWARENESS. In: IV Conferencia Iberoamericana en Sistemas, Cibernética e Informática - CISCI 2005, 2005, Orlando. Proceedings of CISCI2005, 2005.

SANTOS JUNIOR, J. B. ; LOIOLA, M. A. ; ÁVILA, P. M. ; MARQUES, H. N. ; ABRÃO, I.C. ; LIMA, A. B. N. . A Java-XML-Based Platform for Prototyping of Interactive Digital Television Programs. In: 9th International Conference on Enterprise Information Systems, 2007, Funchal, Ilha da Madeira. Proceedings of the 9th International Conference on Enterprise Information Systems. ISI Proceedings., 2007.

SHENKER, S.; PARTRIDGE C.; GUERIN R. *Specification of Guaranteed Quality of Service*. RFC 2212, IETF, 1997.

SILVA, L. H. C.; Caracterização de carga de trabalho para testes de modelos de servidores Web; Dissertação de Mestrado – ICMC – University of São Paulo – USP; setembro 2006.

SOARES, L.F.G.; LEMOS G.; COLCHER S. Redes de Computadores. Das LANs, MANs e WANs às Redes ATM. Editora Campus, 1995.

STALLINGS, W. *High-Speed Networks and Internets: Performance and Quality of Service*. Prentice Hall, 2002.

SWET, R. J.; DRAKE G. R. "The Arena Product Family: Entreprise Modeling Solutions." *Proceeding of Winter Simulation Conference (WSC 2001),* 2001.

TANENBAUM, A. S. *Computer Networks.* Prentice Hal, 2002.

TEIXEIRA, M. A. M. “Suporte a Serviços Diferenciados em Servidores *Web*: Modelos e Algoritmos.” *Tese de Doutorado.* São Carlos: ICMC-USP, maio de 2004.

TEIXEIRA, M. M.; SANTANA M. J.; SANTANA R. H. C. "Servidor *Web* com Diferenciação de Serviços: Fornecendo *QoS* para os Serviços da Internet." *XIII Simpósio Brasileiro de Redes de Computadores (SBRC)*, maio 2005.

TEIXEIRA, M. M.; SANTANA M. J.; SANTANA R. H. C. "Avaliação de Algoritmos de Escalonamento de Tarefas em Servidores *Web* Distribuídos." *XXX Seminário Integrado de Hardware e Software (SEMISH),* Agosto 2003.

VASILIOU, N.; LUTFIYYA H. “Providing a differentiated quality of service in a World Wide *Web* server.” *Performance Evaluation Review*, 2000: 22–28.

VIANA, A. C.; JUKEMURA A. S.; XAVIER D. A.; CARDOSO K. V. “Perspectivas sobre Qualidade de Serviço nos Protocolos da Internet - Estudo de Caso: Aplicações de Vídeo Sob Demanda.” *NEWSGEN, Boletim sobre tecnologia de redes produzido e publicado pela RNP – Rede Nacional de Ensino e Pesquisa*, julho de 2000.

WANG, Q.; MAKAROFF D. J.; EDWARDS H. K. "Characterizing customer groups for an e-commerce *Web*site." *Proceedings of the 5th ACM Conference on Electronic Commerce*, 2004.

WANG, Q.; MAKAROFF D.; EDWARDS H. K.; THOMPSON R. “Workload characterization for an E-commerce *Web* site. .” *Conference of the Centre For Advanced Studies on Collaborative Research*, 2003.

WEICKER, R. “An overview of common benchmarks.” *IEEE Computer*, 1990.

WHITE, P.P. “RSVP *and* integrated services in the Internet: a tutorial.” *Communications Magazine, IEEE*, 1997.

WROCLAWSKI, J. “*Specification of the Controlled-Load Network Element Service.*” RFC 2211, IETF, 1997.

XIAO, X.; NI L.M. "Internet *QoS*: a big picture." *IEEE Network*, Março 1999.

YAN, G.; JIN, Y.; MA Y.; CHENG S.; MA J. “ “An Efficient Method to Simulate Communication Networks”.” *IEEE International Conference on Communications*, 2000: 192-194 .

YING L.; ABDELZAHER T.F.; SAXENA A. "Design, Implementation, *and* Evaluation of Differentiated Caching Services," IEEE Transactions on Parallel *and* Distributed Systems, vol. 15, no. 5, pp. 440-452, May, 2004.

ZHAO, W.; OLSHEFSKI D.; SCHULZRINNE H. "Internet quality of service: an overview." Relatório Técnico CUCS-003-00, Columbia University, 2000.

ZHOU, J.; HASSANEIN H.; MARTIN P. "*QoS* differentiation in switching-based *Web* caching." *IEEE International Conference on Performance, Computing, and Communications*, 2004.

ZHU, H.; YANG T. “Class-based *cache* management for dynamic *Web* content.” *IEEE INFOCOM. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies* , 2001.

ZOU, Q.; MARTIN P.; HASSANEIN H. "Transparent distributed *Web* caching with minimum expected response time." *In Proceedings of the IEEE International Performance, Computing, and Communications Conference*, 2003.

Anexo

A

Tabelas com Valores do Tempo de Resposta HTTP

A tabela A.1 apresenta o tempo médio de resposta *HTTP* Geral nos ambientes com 0%, 40%, 60% e 80% de congestionamento.

As tabelas A.2, A.3, A.4 e A.5 apresentam, respectivamente, o tempo médio de resposta *HTTP* das requisições estáticas por classe de usuário, nos ambientes com 0%, 40%, 60% e 80% de congestionamento.

As tabelas A.6, A.7, A.8 e A.9 apresentam, respectivamente, o tempo médio de resposta *HTTP* das requisições dinâmicas por classe de usuário, nos ambientes com 0%, 40%, 60% e 80% de congestionamento.

Tabela A.11.1- Tempo de Resposta *HTTP* Geral

	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
% Cong	Média	IC95%	Média	IC95%	Média	IC95%	Média	IC95%	Média	IC95%
0	0,383393732	0,000602455	0,379151848	0,000711593	0,383409297	0,0007506	0,378744876	0,000695196	0,378990072	0,000725492
40	0,387529119	0,001078434	0,379854398	0,001106851	0,382924354	0,000986834	0,378883173	0,000948357	0,379078259	0,000480578
60	0,401737876	0,000716793	0,38323287	0,001104472	0,382924354	0,001055114	0,378999042	0,00092244	0,379212823	0,000745131
80	8,840989603	0,053571419	0,391620154	0,000766381	0,383986484	0,000988765	0,379862247	0,001020277	0,379032078	0,000918541

Tabela A.11.2 - Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 0% de congestionamento na rede

0%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	Média	IC95%	Média	IC95%	Média	IC95%	Média	IC95%	Média	IC95%
Padrão	0,009626074	2,02907E-05	0,007647761	1,63977E-05	0,011349658	3,81506E-05	0,007823501	1,87422E-05	0,00786042	1,97892E-05
Bronze	0,009636553	1,37285E-05	0,007641751	2,2629E-05	0,01070926	2,05377E-05	0,007812772	1,84555E-05	0,00782514	2,61192E-05
Prata	0,009630509	1,46011E-05	0,007625295	1,70132E-05	0,010231715	1,68515E-05	0,007829509	1,66749E-05	0,007778326	1,98077E-05
Ouro	0,009632241	1,28871E-05	0,007671564	1,93061E-05	0,009915436	1,54804E-05	0,007821516	1,95327E-05	0,007733598	1,97267E-05

Tabela A.11.3 - Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 40% de congestionamento na rede

40%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	Média	IC95%	Média	IC95%	Média	IC95%	Média	IC95%	Média	IC95%
Padrão	0,015832176	7,1288E-05	0,009633557	2,47611E-05	0,01138292	3,50168E-05	0,007828841	2,59615E-05	0,007871479	2,71176E-05
Bronze	0,015799831	6,02158E-05	0,009613997	3,40232E-05	0,010697127	2,14309E-05	0,007819452	2,19618E-05	0,007812084	2,32472E-05
Prata	0,015856827	8,00591E-05	0,009624618	2,87052E-05	0,010226064	1,57904E-05	0,007819632	2,58625E-05	0,007768284	1,99692E-05
Ouro	0,015815368	5,37411E-05	0,009599216	3,64249E-05	0,009914361	1,8216E-05	0,007822957	1,68788E-05	0,007730076	1,94417E-05

Tabela A.11.4 - Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 60% de congestionamento na rede

60%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**
Padrão	0,031875025	0,000284494	0,012685286	7,23944E-05	0,01138292	3,50168E-05	0,007842211	2,01821E-05	0,007867701	2,03522E-05
Bronze	0,031849665	0,000300944	0,01272891	5,99156E-05	0,010697127	2,14309E-05	0,007833667	2,04804E-05	0,007813168	1,60574E-05
Prata	0,031972158	0,000322939	0,012660165	6,47845E-05	0,010226064	1,57904E-05	0,007837213	2,03707E-05	0,00777921	1,96313E-05
Ouro	0,031842997	0,000268265	0,012692996	5,63358E-05	0,009914361	1,8216E-05	0,007842457	2,46401E-05	0,00774348	1,79618E-05

Tabela A.11.5 - Tempo de Resposta *HTTP* por Classe – Requisições Estáticas com 80% de congestionamento na rede

80%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**
Padrão	10,83531363	0,074613314	0,029538225	0,000467186	0,011362589	4,99229E-05	0,007830338	2,59781E-05	0,007867845	2,39216E-05
Bronze	10,770049	0,072746496	0,029656752	0,000444631	0,010724982	1,64263E-05	0,007831584	2,43005E-05	0,007799248	2,4569E-05
Prata	10,8017953	0,074078819	0,029735806	0,000434043	0,01022583	2,22884E-05	0,007848477	2,25254E-05	0,007784307	2,08366E-05
Ouro	10,79783564	0,073789796	0,029585819	0,000337238	0,009917207	1,86991E-05	0,007831788	2,43314E-05	0,007750014	1,49075E-05

Tabela A.11.6 - Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 0% de congestionamento na rede

0%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**
Padrão	0,785212951	0,000129201	0,780930418	0,0001478	0,786190344	0,000150955	0,780813586	0,00014436	0,780801901	0,000127859
Bronze	0,785341037	0,000112479	0,780890678	0,000117818	0,78594686	0,000149053	0,780893247	0,000139322	0,780877263	0,000145468
Prata	0,785264428	0,000111337	0,780890161	0,000181413	0,785547549	0,000124878	0,781020121	0,000120062	0,780822267	8,83749E-05
Ouro	0,785295828	0,000149436	0,781019431	0,000108066	0,785333068	0,000118612	0,780885922	0,000180023	0,780858422	0,000115706

Tabela A.11.7 - Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 40% de congestionamento na rede

40%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**
Padrão	0,78890688	0,000119279	0,781048897	0,000136739	0,786267576	0,000100509	0,780932483	0,000130426	0,780896296	0,000120463
Bronze	0,788827873	0,000111209	0,780937469	0,000135863	0,785750538	0,000125744	0,780935005	0,000140312	0,780830732	0,000168906
Prata	0,788884157	0,000120722	0,780998144	0,000149436	0,78544105	0,000139757	0,780896688	0,000120164	0,780705569	0,00012727
Ouro	0,788842676	0,000105986	0,780979882	0,00016668	0,78539539	0,000133134	0,780971554	0,000143747	0,780843827	0,000140352

Tabela A.11.8 - Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 60% de congestionamento na rede

60%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**
Padrão	0,798464849	0,000227714	0,781049029	0,000135529	0,786267576	0,000100509	0,780899939	0,000153806	0,780822109	0,000114515
Bronze	0,798527772	0,000222941	0,781003004	0,000130576	0,785750538	0,000125744	0,780952128	0,000173298	0,780903743	0,000143027
Prata	0,79837311	0,000256973	0,781037457	0,000131265	0,78544105	0,000139757	0,780940588	0,000134183	0,780751747	0,000110526
Ouro	0,798565252	0,00023956	0,780992111	0,000147553	0,78539539	0,000133134	0,781030044	0,000154402	0,78080956	0,000137817

Tabela A.11.9 - Tempo de Resposta *HTTP* por Classe – Requisições Dinâmicas com 80% de congestionamento na rede

80%	Cenário SS		Cenário SC		Cenário CS		Cenário CC		Cenário CDF	
Classe	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**	**Média**	**IC95%**
Padrão	7,511395091	0,054184449	0,780992219	0,000126231	0,786204387	0,000117724	0,781025836	0,000205127	0,780809491	0,000120107
Bronze	7,450725194	0,06806353	0,780861599	9,08616E-05	0,785833143	0,000150796	0,780936035	0,000165704	0,780831729	0,00017689
Prata	7,461440301	0,053973416	0,78106023	0,000164654	0,785598014	0,00013274	0,780927192	0,000108638	0,7807842	0,000105085
Ouro	7,492645348	0,049952993	0,780899463	0,000112502	0,78534542	0,000106785	0,780906401	0,00018053	0,780816393	9,53193E-05

Anexo

B

Tabela descritiva da troca de pacotes TCP entre os clientes e o servidor Web

A tabela B.1 abaixo ilustra um trecho das trocas de pacotes *TCP* entre cliente (Rede) e o servidor *Web* (internet_server) no cenário SS com 80% de congestionamento. É possível verificarmos na tabela que a rede está congestionada, pois há vários pacotes *TCP* com *flag* RST resetando a conexão e alguns pacotes ilustram a tentativa de ajuste do tamanho da janela de transmissão para que diminua o congestionamento da rede.

A condição de congestionamento em uma rede de pacotes como a Internet ocorre quando o desempenho da rede é degradado pela presença em excesso de pacotes. Vários fatores podem acarretar o congestionamento. Nesse caso, vários fluxos de pacotes chegam no roteador cliente (RC) e todos esses fluxos devem ser escoados pela mesma porta de saída; a fila deste roteador enche, e a quantidade de memória para armazenar todos os pacotes é insuficiente, então os pacotes são descartados.

Tabela B.1 – Trecho de *log* de troca de pacotes entre os Clientes (Rede) e o Servidor *Web* (internet_server)

Source	Destination	Protocol	Summary
[Rede]	[internet_server]	*TCP*	D=80 S=6264 PSH SYN ACK=0 SEQ=0 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=6265 PSH SYN ACK=0 SEQ=0 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=5925 ACK=36348855 SEQ=351 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=5763 ACK=35620781 SEQ=701 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6009 S=80 PSH SYN ACK=1 SEQ=36629920 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=4757 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=4744 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=6463 SYN ACK=0 SEQ=0 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=4777 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=4726 RST ACK=0 SEQ=351 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6010 S=80 PSH SYN ACK=1 SEQ=36644629 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=4274 RST ACK=0 SEQ=1401 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=5550 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=4953 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=4955 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=4956 RST ACK=0 SEQ=351 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6011 S=80 PSH SYN ACK=1 SEQ=36646187 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=5553 RST ACK=0 SEQ=351 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=5207 RST ACK=0 SEQ=701 LEN=0 WIN=0
[Rede]	[internet_server]	*TCP*	D=80 S=6008 RST ACK=0 SEQ=1 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6012 S=80 PSH SYN ACK=1 SEQ=36647060 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6013 S=80 PSH SYN ACK=1 SEQ=36650713 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6014 S=80 PSH SYN ACK=1 SEQ=36654886 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6015 S=80 PSH SYN ACK=1 SEQ=36658683 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=6015 RST ACK=0 SEQ=1 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6017 S=80 PSH SYN ACK=1 SEQ=36665362 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6018 S=80 PSH SYN ACK=1 SEQ=36674758 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=6018 RST ACK=0 SEQ=1 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6019 S=80 PSH SYN ACK=1 SEQ=36674885 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6020 S=80 PSH SYN ACK=1 SEQ=36680920 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=6020 RST ACK=0 SEQ=1 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6021 S=80 PSH SYN ACK=1 SEQ=36686564 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6022 S=80 PSH SYN ACK=1 SEQ=36687376 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6023 S=80 PSH SYN ACK=1 SEQ=36694157 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6024 S=80 PSH SYN ACK=1 SEQ=36714371 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=6024 RST ACK=0 SEQ=1 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6025 S=80 PSH SYN ACK=1 SEQ=36714498 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6027 S=80 PSH SYN ACK=1 SEQ=36722745 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6029 S=80 PSH SYN ACK=1 SEQ=36733684 LEN=0 WIN=8760
[Rede]	[internet_server]	*TCP*	D=80 S=6029 RST ACK=0 SEQ=1 LEN=0 WIN=0
[internet_server]	[Rede]	*TCP*	D=6030 S=80 PSH SYN ACK=1 SEQ=36735989 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6031 S=80 PSH SYN ACK=1 SEQ=36743920 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=6032 S=80 PSH SYN ACK=1 SEQ=36747153 LEN=0 WIN=8760
[internet_server]	[Rede]	*TCP*	D=4436 S=80 ACK=1051 SEQ=30415506 LEN=0 WIN=8760

Anexo

C

Documento XML com a Descrição do Cenário Base

A figura C.1 abaixo apresenta a descrição dos tipos de dados – DTD – dos modelos de rede do *OPNET Modeler*. A figura C.2 apresente o documento XML, em conformidade com a DTD da figura C.1, contendo os dados dos equipamentos que compõem o cenário BASE.

As Figuras C.3, C.4 e C.5 contêm documentos XML, igualmente em conformidade com a DTD da figura C.1, e descrevem respectivamente: os dados da configuração das requisições, a configuração dos perfis de usuários e a configuração dos *links* do cenário BASE.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- ============================================================ -->
<!-- OPNET network topology DTD v0.9 -->
<!-- ============================================================ -->

<!-- The top-level network -->
<!ELEMENT network (profile*, subnet*, (node|bus)*, (link|tap)*, path*, demand*,
 (characteristic|attr|ext-attr|pstate|
 annotation|view-props)*) >
<!-- name       The network's name/label
        locale  The local setting the file was written in
        version        Version of OPNET network xml
        attribute_processing How attribute setting is handled -->
<!ATTLIST network
 name CDATA    #IMPLIED
        locale  CDATA   #IMPLIED
        version       CDATA    #IMPLIED
        attribute_processing (explicit|loose)       "loose" >
<!-- Subnet -->
<!ELEMENT subnet (subnet*, (node|bus)*, (link|tap)*,
 (characteristic|attr|ext-attr|pstate|
 annotation|view-props|ui_status)*) >
<!-- name The subnet's name/label (unique within
 parent; may not contain dots or be "top")
 mobility mobility class -->
<!ATTLIST subnet
 name CDATA #REQUIRED
 mobility (fixed|mobile|satellite) "fixed" >

<!-- Node -->
<!ELEMENT node  (port*, (characteristic|attr|ext-attr|pstate|ui_status)*) >
<!-- name       Name (unique within net, no dots, not "top")
 type    Machine type (see list at end of file)
 model   OPNET model name
 sysObjID       SNMP System Object ID
 vendor  Hardware manufacturer
 product        Product name
 mobility       mobility class
        min_match_score      cutoff score for matching (may be "strict matching" or a
number)
 ignore_questions Missing characteristics will not
               cause questions to occur during
               import          -->
<!ATTLIST node
 name    CDATA #REQUIRED
 min_match_score      CDATA                  #REQUIRED
 type    CDATA #IMPLIED
 model   CDATA #IMPLIED
 sysObjID       CDATA #IMPLIED
 vendor  CDATA #IMPLIED
 product        CDATA #IMPLIED
 mobility       (fixed|mobile|satellite) "fixed"
 ignore_questions     (true|false)    "true" >

<!-- Network interface on a node -->
<!ELEMENT port (characteristic|attr|pstate)* >
<!-- name Name (unique within node, no dots, not "top")
 type Interface type (see list at end of file)
 group Interface group (sharing a common type)
 class Port class
        ifclass       Interface class (how port connects to protocol
        stack; e.g. switch or router) -->
<!ATTLIST port
 name CDATA #REQUIRED
 class (bus|pointtopoint|radio) "pointtopoint"
 type CDATA #IMPLIED
 group CDATA #IMPLIED
        ifclass        CDATA          #IMPLIED >

<!-- Point-to-point link -->
<!ELEMENT link (characteristic|attr|ext-attr|pstate|ui_status)* >
<!-- name       Name (unique within net, no dots, not "top")
 type    Interface type (see list at end of file)
        group   Interface group (sharing a common type)
 model   OPNET model name
 srcNode        Source node, relative to current subnet (if
               any); subnets appear from highest to lowest
```

```
              level followed by node, with dots in between
              every element.)
 destNode       Destination node, as above.
 srcPort        Port on source node to connect to.
 destPort       Port on destination node to connect to.
        min_match_score      required cutoff score for matching (may be "strict
matching")
 ignore_questions Missing characteristics will not
              cause questions to occur during
              import          -->
<!ATTLIST link
 name    CDATA         #IMPLIED
 min_match_score      CDATA                #REQUIRED
 type    CDATA         #IMPLIED
 group   CDATA                       #IMPLIED
 model   CDATA         #IMPLIED
 srcNode        CDATA         #REQUIRED
 destNode       CDATA         #REQUIRED
 srcPort        CDATA         #IMPLIED
 destPort       CDATA  #             IMPLIED
 ignore_questions     (true|false)    "true" >

<!-- Bus -->
<!ELEMENT bus (characteristic|attr|ext-attr|pstate|ui_status)* >
<!-- name Name (unique within net, no dots, not "top")
 type Interface type (see list at end of file)
 group Interface group (sharing a common type)
 model OPNET model name
 ignore_questions Missing characteristics will not
              cause questions to occur during
              import          -->
<!ATTLIST bus
 name CDATA #REQUIRED
 min_match_score      CDATA                #REQUIRED
 type CDATA #IMPLIED
 group CDATA #IMPLIED
 model CDATA #IMPLIED
 ignore_questions (true|false)       "true" >

<!-- Bus tap (connects a port to a bus) -->
<!ELEMENT tap (characteristic|attr|ext-attr|pstate|ui_status)* >
<!-- name Name (unique within net, no dots, not "top")
 type Interface type (see list at end of file)
 group Interface group (sharing a common type)
 model OPNET model name
 bus Name of the bus this tap connects to
 node Name of the node this tap connects to
 port Name of the port on the node this tap
 connects to
 ignore_questions Missing characteristics will not
              cause questions to occur during
              import          -->
<!ATTLIST tap
 name CDATA #IMPLIED
        min_match_score      CDATA                  #REQUIRED
 bus CDATA #REQUIRED
 node CDATA #REQUIRED
 port CDATA #IMPLIED
 type CDATA #IMPLIED
 group CDATA #IMPLIED
 model CDATA #IMPLIED
 ignore_questions (true|false)       "true" >

<!-- Path Object -->
<!ELEMENT path (path-element*, (characteristic|attr|ext-attr|pstate|ui_status)*) >
<!-- name Name (unique within net, no dots, not "top")
        min_match_score      required cutoff score for matching (may be "strict
matching")
 type Interface type (see list at end of file)
 group Interface group (sharing a common type)
 model OPNET model name
 ignore_questions Missing characteristics will not
              cause questions to occur during
              import          -->
<!ATTLIST path
 name CDATA #REQUIRED
 model CDATA #REQUIRED
```

```
 min_match_score      CDATA                #REQUIRED
 ignore_questions (true|false)       "false" >

<!-- path element -->
<!-- refers to a node or link the PATH should
 pass through -->
<!ELEMENT path-element EMPTY >
<!-- name name of referenced element, interpreted
 relative to the subnet containing the PATH
 unless it begins with "top."
 type type of referenced element -->
<!ATTLIST path-element
 name CDATA #REQUIRED
 type (node|link) "node" >

<!-- Demand Object -->
<!ELEMENT demand (demand-path*, (characteristic|attr|ext-attr|pstate|ui_status)*) >
<!-- name Name (unique within net, no dots, not "top")
        min_match_score      required cutoff score for matching (may be "strict
matching")
 type Interface type (see list at end of file)
 group Interface group (sharing a common type)
 model OPNET model name
        src    node
        dest   node
 ignore_questions Missing characteristics will not
              cause questions to occur during
              import          -->
<!ATTLIST demand
 name CDATA           #REQUIRED
        srcNode       CDATA                #REQUIRED
        destNode      CDATA                #REQUIRED
 model CDATA   #REQUIRED
 min_match_score      CDATA          #REQUIRED
 ignore_questions (true|false)       "false" >

<!-- demand path -->
<!-- refers to a path object used by demand  -->
<!ELEMENT demand-path EMPTY           >
<!-- name name of referenced path     -->
<!ATTLIST demand-path
 name CDATA #REQUIRED>

<!-- Profile Attribute Value -->
<!ELEMENT profile     (point*) >
<!ATTLIST profile
        name   CDATA    #REQUIRED
        x_units       CDATA    #REQUIRED
        y_units       CDATA    #REQUIRED >

<!ELEMENT point       EMPTY>
<!ATTLIST point
        x      CDATA    #REQUIRED
        y      CDATA    #REQUIRED
        string CDATA    #IMPLIED>

<!-- Characteristic used for matching -->
<!--     Not necessary for current models (existing
 fields cover them), but present to support
 expansion. -->
<!ELEMENT characteristic
 EMPTY >
<!ATTLIST characteristic
 name CDATA #REQUIRED
 value CDATA #REQUIRED
        score CDATA           "100.0"
        pass_to_devcr (true|false)    "true"
        comparison    (Equals|Contains|EqualsOneOf|ContainsOneOf) "Contains">

<!-- Attribute value assignment -->
<!ELEMENT attr (attr)* >
<!-- name Name of the attribute receiving a value;
 examples include "IP address",
 "IP subnet mask"
 intended      Did the user set this attribute?
 symbolic Is the attribute symbolic? -->
<!ATTLIST attr
```

```
 name CDATA #REQUIRED
 intended (true|false) "true"
 symbolic (true|false) "false"
 value CDATA #REQUIRED >

<!-- Simple extended attribute def with value -->
<!ELEMENT ext-attr (default-value?, comments?, ext-attr*) >
<!-- name Name of the extended attribute receiving a value;
 examples include "Chassis Name"
 type    Data type of extended attribute
         group   Group under which ext. attr. should be displayed (if any)
         units   Units of the attribute (if any)
         value   Value of extended attribute   -->
<!ATTLIST ext-attr
 name CDATA #REQUIRED
 type    (integer|double|string|toggle|typed_file|profile|color|compound) #REQUIRED
         group   CDATA   #IMPLIED
         units   CDATA          #IMPLIED
         value  CDATA #IMPLIED >

<!-- Default value for extended attribute definition
 NOTE: This is intended to be compatible with the XML
 format for attributes as supported by process,
 node, packet, etc. model export and import -->
<!ELEMENT default-value EMPTY >
<!--     value   Default value of extended attribute  -->

<!ATTLIST default-value
 value CDATA #REQUIRED >

<!-- Comments for extended attribute definition
 NOTE: This is intended to be compatible with the XML
 format for attributes as supported by process,
 node, packet, etc. model export and import -->
<!ELEMENT comments (#PCDATA) >

<!-- Persistent state -->
<!ELEMENT pstate (attr)* >
<!-- name Name of state type
 level Level at which state applies -->
<!ATTLIST pstate
 name CDATA #REQUIRED
 level (object|network|project) "object" >

<!-- Subnet annotation -->
<!ELEMENT annotation (attr|ext-attr)* >
<!-- name Name of annotation
 type Class of annotation -->
<!ATTLIST annotation
 name CDATA #IMPLIED
 type (text|box|ellipse|line) #REQUIRED >

<!-- view properties -->
<!ELEMENT view-props (attr)* >

<!-- UI status of object -->
<!ELEMENT ui_status   EMPTY   >
<!-- name Name of UI status property
 value Value of UI status property -->
<!ATTLIST ui_status
 name
(hidden|expanded|bkg_objs_hidden|annot_hidden|site_label_hidden|link_label_shown|top_layer
|label_boxed|cut_disabled|copy_disabled)    #REQUIRED
         value   (true|false)  #REQUIRED >
```

Figura C.1 – Descrição dos tipos (DTD) da rede no *OPNET Modeler*

```
<?xml version="1.0"?>
<network locale="C" version="1.6" attribute_processing="explicit">
<profile name="40%" x_units="seconds" y_units="bits/second">
</profile>
<subnet name="Network">
       <node name="SC - Switch Cliente" model="ethernet8_switch_adv"
ignore_questions="true" min_match_score="strict matching">
        <attr name="tooltip" value="Ethernet Switch"/>
       </node>

       <node name="RC - Router Cliente" model="ethernet2_slip8_gtwy_adv"
ignore_questions="true" min_match_score="strict matching">
       <attr name="tooltip" value="IP Router"/>
       <attr name="IP Processing Information [0].Datagram Forwarding Rate"
value="50,000"/>
       </node>

       <node name="RS - Router Servidor" model="ethernet2_slip8_gtwy_adv"
ignore_questions="true" min_match_score="strict matching">
       <attr name="tooltip" value="IP Router"/>
       <attr name="IP Processing Information [0].Datagram Forwarding Rate"
value="50,000"/>
       </node>

       <node name="SS - Switch Servidor" model="ethernet8_switch_adv"
ignore_questions="true" min_match_score="strict matching">
       <attr name="tooltip" value="Ethernet Switch"/>
       </node>

       <node name="internet_server" model="ethernet_server" ignore_questions="true"
min_match_score="strict matching">
       <attr name="tooltip" value="Ethernet Server"/>
       <attr name="Application: Destination Preferences" value="None" symbolic="true"/>
       <attr name="Application: Supported Services" value="All" symbolic="true"/>
       <attr name="Server Address" value="internet_server"/>
       </node>

       <node name="Rede" model="100BaseT_LAN" ignore_questions="true"
min_match_score="strict matching">
       <attr name="tooltip" value="100BaseT Switched LAN"/>
       <attr name="Application: Destination Preferences.count" value="1"
symbolic="true"/>
       <attr name="Application: Destination Preferences [0].Application" value="All
Applications" symbolic="true"/>
       <attr name="Application: Destination Preferences [0].Symbolic Name" value="HTTP
Server"/>
       <attr name="Application: Destination Preferences [0].Actual Name.count" value="1"
symbolic="true"/>
       <attr name="Application: Destination Preferences [0].Actual Name [0].Name"
value="internet_server"/>
       <attr name="Application: Destination Preferences [0].Actual Name [0].Selection
Weight" value="10" symbolic="true"/>
       <attr name="Application: Supported Profiles.count" value="4"/>
       <attr name="Application: Supported Profiles [0].Profile Name" value="padrao"/>
       <attr name="Application: Supported Profiles [0].Number of Clients" value="100"/>
       <attr name="Application: Supported Profiles [1].Profile Name" value="bronze"/>
       <attr name="Application: Supported Profiles [1].Number of Clients" value="100"/>
       <attr name="Application: Supported Profiles [2].Profile Name" value="prata"/>
       <attr name="Application: Supported Profiles [2].Number of Clients" value="100"/>
       <attr name="Application: Supported Profiles [3].Profile Name" value="ouro"/>
       <attr name="Application: Supported Profiles [3].Number of Clients" value="100"/>
       <attr name="Application: Supported Services" value="None" symbolic="true"/>
       <attr name="Number of Workstations" value="400"/>
       </node>
</network>
```

Figura C.2 – Documento XML dos equipamentos do cenário BASE

```
       <node name="dinamico_applications" model="Application Config"
ignore_questions="true" min_match_score="strict matching">
       <attr name="tooltip" value="Application Configuration"/>
       <attr name="Application Definitions.count" value="9"/>
       <attr name="Application Definitions [0].Name" value="database_access"/>
       <attr name="Application Definitions [0].Description.count" value="1"/>
       <attr name="Application Definitions [0].Description [0].Custom.count" value="1"/>
       <attr name="Application Definitions [0].Description [0].Custom [0].Task
Description.count" value="1" symbolic="true"/>
       <attr name="Application Definitions [0].Description [0].Custom [0].Task
Description [0].Task Name" value="ecommerce"/>
       <attr name="Application Definitions [0].Description [0].Custom [0].Task
Description [0].Task Weight" value="10"/>
       <attr name="Application Definitions [0].Description [0].Custom [0].Task Ordering"
value="Serial (Ordered)" symbolic="true"/>
       <attr name="Application Definitions [0].Description [0].Custom [0].Transport
Protocol" value="TCP" symbolic="true"/>
       <attr name="Application Definitions [0].Description [0].Custom [0].RSVP
Parameters" value="None" symbolic="true"/>
       <attr name="Application Definitions [1].Name" value="BE_Dinamico"/>
       <attr name="Application Definitions [1].Description.count" value="1"/>
       <attr name="Application Definitions [1].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page
Properties.count" value="1" symbolic="true"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application.count" value="1"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Custom Application Name" value="database_access"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Execution Probability" value="Always"
symbolic="true"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Dinamico"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [1].Description [0].HTTP [0].Type of Service"
value="Best Effort (0)" symbolic="true"/>
       <attr name="Application Definitions [2].Name" value="BE"/>
       <attr name="Application Definitions [2].Description.count" value="1"/>
       <attr name="Application Definitions [2].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page
Properties.count" value="2" symbolic="true"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.55, 1.42)"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Linguagem de Marcacao"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[1].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[1].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[1].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[1].Back-End Custom Application" value="Not Used" symbolic="true"/>
```

```
       <attr name="Application Definitions [2].Description [0].HTTP [0].Page Properties
[1].Object Group Name" value="Imagem"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [2].Description [0].HTTP [0].Type of Service"
value="Best Effort (0)" symbolic="true"/>
       <attr name="Application Definitions [3].Name" value="AF11_Dinamico"/>
       <attr name="Application Definitions [3].Description.count" value="1"/>
       <attr name="Application Definitions [3].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page
Properties.count" value="1" symbolic="true"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application.count" value="1"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Custom Application Name" value="database_access"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Execution Probability" value="Always"
symbolic="true"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Dinamico"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [3].Description [0].HTTP [0].Type of Service"
value="AF11" symbolic="true"/>
       <attr name="Application Definitions [4].Name" value="AF11"/>
       <attr name="Application Definitions [4].Description.count" value="1"/>
       <attr name="Application Definitions [4].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page
Properties.count" value="2" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.55, 1.42)"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Linguagem de Marcacao"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[1].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[1].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[1].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[1].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Page Properties
[1].Object Group Name" value="Imagem"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [4].Description [0].HTTP [0].Type of Service"
value="AF11" symbolic="true"/>
       <attr name="Application Definitions [5].Name" value="AF31_Dinamico"/>
       <attr name="Application Definitions [5].Description.count" value="1"/>
       <attr name="Application Definitions [5].Description [0].HTTP.count" value="1"/>
```

```
       <attr name="Application Definitions [5].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page
Properties.count" value="1" symbolic="true"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application.count" value="1"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Custom Application Name" value="database_access"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Execution Probability" value="Always"
symbolic="true"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Dinamico"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [5].Description [0].HTTP [0].Type of Service"
value="AF31" symbolic="true"/>
       <attr name="Application Definitions [6].Name" value="AF31"/>
       <attr name="Application Definitions [6].Description.count" value="1"/>
       <attr name="Application Definitions [6].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page
Properties.count" value="2" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.55, 1.42)"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Linguagem de Marcacao"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[1].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[1].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[1].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[1].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Page Properties
[1].Object Group Name" value="Imagem"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [6].Description [0].HTTP [0].Type of Service"
value="AF31" symbolic="true"/>
       <attr name="Application Definitions [7].Name" value="EF_Dinamico"/>
       <attr name="Application Definitions [7].Description.count" value="1"/>
       <attr name="Application Definitions [7].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page
Properties.count" value="1" symbolic="true"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
```

```
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application.count" value="1"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Custom Application Name" value="database_access"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application [0].Execution Probability" value="Always"
symbolic="true"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Dinamico"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [7].Description [0].HTTP [0].Type of Service"
value="EF" symbolic="true"/>
       <attr name="Application Definitions [8].Name" value="EF"/>
       <attr name="Application Definitions [8].Description.count" value="1"/>
       <attr name="Application Definitions [8].Description [0].HTTP.count" value="1"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].HTTP
Specification" value="HTTP 1.1" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Interarrival
Time" value="exponential (10)"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page
Properties.count" value="2" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[0].Object Size" value="lognormal (8.55, 1.42)"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[0].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[0].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[0].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[0].Object Group Name" value="Linguagem de Marcacao"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[1].Object Size" value="lognormal (8.25, 1.62)"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[1].Number of Objects" value="constant (1)"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[1].Location" value="HTTP Server" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[1].Back-End Custom Application" value="Not Used" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Page Properties
[1].Object Group Name" value="Imagem"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Server
Selection.count" value="1"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].RSVP Parameters"
value="None" symbolic="true"/>
       <attr name="Application Definitions [8].Description [0].HTTP [0].Type of Service"
value="EF" symbolic="true"/>
       </node>
```

Figura C.3 – Documento XML da configuração das requisições estáticas e dinâmicas do cenário BASE

```
<node name="Profile_Config" model="Profile Config" ignore_questions="true"
min_match_score="strict matching">
       <attr name="tooltip" value="Profile Configuration"/>
       <attr name="Profile Configuration.count" value="4"/>
       <attr name="Profile Configuration [0].Profile Name" value="padrao"/>
       <attr name="Profile Configuration [0].Applications.count" value="2"
symbolic="true"/>
       <attr name="Profile Configuration [0].Applications [0].Name" value="BE"/>
       <attr name="Profile Configuration [0].Applications [0].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [0].Applications [0].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [0].Applications [0].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [0].Applications [1].Name" value="BE_Dinamico"/>
       <attr name="Profile Configuration [0].Applications [1].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [0].Applications [1].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [0].Applications [1].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [0].Operation Mode" value="Simultaneous"
symbolic="true"/>
       <attr name="Profile Configuration [0].Start Time" value="uniform (20, 50)"/>
       <attr name="Profile Configuration [0].Duration" value="constant (700)"/>
       <attr name="Profile Configuration [0].Repeatability" value="Unlimited"
symbolic="true"/>
       <attr name="Profile Configuration [1].Profile Name" value="bronze"/>
       <attr name="Profile Configuration [1].Applications.count" value="2"
symbolic="true"/>
       <attr name="Profile Configuration [1].Applications [0].Name" value="AF11"/>
       <attr name="Profile Configuration [1].Applications [0].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [1].Applications [0].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [1].Applications [0].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [1].Applications [1].Name"
value="AF11_Dinamico"/>
       <attr name="Profile Configuration [1].Applications [1].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [1].Applications [1].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [1].Applications [1].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [1].Operation Mode" value="Simultaneous"
symbolic="true"/>
       <attr name="Profile Configuration [1].Start Time" value="uniform (50, 75)"/>
       <attr name="Profile Configuration [1].Duration" value="constant (700)"/>
       <attr name="Profile Configuration [1].Repeatability" value="Unlimited"
symbolic="true"/>
       <attr name="Profile Configuration [2].Profile Name" value="prata"/>
       <attr name="Profile Configuration [2].Applications.count" value="2"
symbolic="true"/>
       <attr name="Profile Configuration [2].Applications [0].Name" value="AF31"/>
       <attr name="Profile Configuration [2].Applications [0].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [2].Applications [0].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [2].Applications [0].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [2].Applications [1].Name"
value="AF31_Dinamico"/>
       <attr name="Profile Configuration [2].Applications [1].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [2].Applications [1].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [2].Applications [1].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [2].Operation Mode" value="Simultaneous"
symbolic="true"/>
       <attr name="Profile Configuration [2].Start Time" value="uniform (50, 75)"/>
       <attr name="Profile Configuration [2].Duration" value="constant (700)"/>
       <attr name="Profile Configuration [2].Repeatability" value="Unlimited"
symbolic="true"/>
       <attr name="Profile Configuration [3].Profile Name" value="ouro"/>
```

```
       <attr name="Profile Configuration [3].Applications.count" value="2"
symbolic="true"/>
       <attr name="Profile Configuration [3].Applications [0].Name" value="EF"/>
       <attr name="Profile Configuration [3].Applications [0].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [3].Applications [0].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [3].Applications [0].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [3].Applications [1].Name" value="EF_Dinamico"/>
       <attr name="Profile Configuration [3].Applications [1].Start Time Offset"
value="uniform (0, 10)"/>
       <attr name="Profile Configuration [3].Applications [1].Duration" value="End of
Profile" symbolic="true"/>
       <attr name="Profile Configuration [3].Applications [1].Repeatability"
value="Unlimited" symbolic="true"/>
       <attr name="Profile Configuration [3].Operation Mode" value="Simultaneous"
symbolic="true"/>
       <attr name="Profile Configuration [3].Start Time" value="uniform (50, 75)"/>
       <attr name="Profile Configuration [3].Duration" value="constant (700)"/>
       <attr name="Profile Configuration [3].Repeatability" value="Unlimited"
symbolic="true"/>
       <attr name="condition" value="enabled"/>
</node>
```

Figura C.4 – Documento XML da configuração dos perfis de usuários do cenário BASE

```
<link name="SC - Switch Cliente &lt;-&gt; RC - Router Cliente" model="100BaseT"
class="duplex" srcNode="SC - Switch Cliente" destNode="RC - Router Cliente"
ignore_questions="true" min_match_score="strict matching">
       <attr name="transmitter a" value="SC - Switch Cliente.hub_tx_0"/>
       <attr name="receiver a" value="SC - Switch Cliente.hub_rx_0"/>
       <attr name="transmitter b" value="RC - Router Cliente.hub_tx_8_0"/>
       <attr name="receiver b" value="RC - Router Cliente.hub_rx_8_0"/>
       </link>
<link name="LINK RC &lt;-&gt; RS" model="PPP_DS1" class="duplex" srcNode="RC - Router
Cliente" destNode="RS - Router Servidor" ignore_questions="true" min_match_score="strict
matching">
       <attr name="transmitter a" value="RC - Router Cliente.pt_0_0"/>
       <attr name="receiver a" value="RC - Router Cliente.pr_0_0"/>
       <attr name="transmitter b" value="RS - Router Servidor.pt_0_0"/>
       <attr name="receiver b" value="RS - Router Servidor.pr_0_0"/>
       <attr name="tooltip" value="PPP DS1"/>
       <attr name="Background Load.count" value="1"/>
       <attr name="Background Load [0].Average Packet Size (A -&gt; B)" value="Default"
symbolic="true"/>
       <attr name="Background Load [0].Average Packet Size (B -&gt; A)" value="Default"
symbolic="true"/>
       <attr name="Background Load [0].Traffic Intensity (A -&gt; B)" value="40%"/>
       <attr name="Background Load [0].Traffic Intensity (B -&gt; A)" value=""/>
       </link>
       <link name="SS - Switch Servidor &lt;-&gt; RS - Router Servidor" model="100BaseT"
class="duplex" srcNode="SS - Switch Servidor" destNode="RS - Router Servidor"
ignore_questions="true" min_match_score="strict matching">
       <attr name="transmitter a" value="SS - Switch Servidor.hub_tx_1"/>
       <attr name="receiver a" value="SS - Switch Servidor.hub_rx_1"/>
       <attr name="transmitter b" value="RS - Router Servidor.hub_tx_9_0"/>
       <attr name="receiver b" value="RS - Router Servidor.hub_rx_9_0"/>
       <attr name="packet formats" value="ethernet_v2"/>
       </link>
       <link name="internet_server &lt;-&gt; SS - Switch Servidor" model="10BaseT"
class="duplex" srcNode="internet_server" destNode="SS - Switch Servidor"
ignore_questions="true" min_match_score="strict matching">
       <attr name="transmitter a" value="internet_server.hub_tx_0_0"/>
       <attr name="receiver a" value="internet_server.hub_rx_0_0"/>
       <attr name="transmitter b" value="SS - Switch Servidor.hub_tx_0"/>
       <attr name="receiver b" value="SS - Switch Servidor.hub_rx_0"/>
       </link>
       <link name="Rede &lt;-&gt; SC - Switch Cliente" model="10BaseT" class="duplex"
srcNode="Rede" destNode="SC - Switch Cliente" ignore_questions="true"
min_match_score="strict matching">
       <attr name="transmitter a" value="Rede.tx_0_0"/>
       <attr name="receiver a" value="Rede.rx_0_0"/>
       <attr name="transmitter b" value="SC - Switch Cliente.hub_tx_1"/>
       <attr name="receiver b" value="SC - Switch Cliente.hub_rx_1"/>

       </link>
```

Figura C.5– Documento XML dos *links* do cenário BASE

Anexo

D

Implementação do Servidor Cache CDF

A figura D.1 ilustra a alteração no código original do servidor *cache* para dar suporte a diferenciação de servidos. Foi alterado o código da abertura de conexão entre o Servidor *Cache* (CDF) e o Servidor *Web*. Ao abrir uma nova conexão entre o *Cache* CDF e o Servidor *Web* é analisado o tipo de serviço solicitado pelos usuários e é realizada uma conexão com o servidor *Web* com o mesmo tipo de serviço.

```
/*****************************************************************************************************************/
/
// Função reponsável por abrir um Conexão TCP entre o Servidor Cache CDF e o Internet Server
//
/*****************************8*****************************************************************************************/

static void
gna_clsvr_mgr_cache_spawn_session (GnaT_Cli_Mgr_Session* sess_ptr, int sess_command, int server_index,
                                   int HTTP_command, List* inline_objects_list, double req_size,
                                   GnaT_Nam_Appl*  appl_info_ptr, SimT_Pk_Id pkt_id)
{
        GnaT_Cli_HTTP_Params*           HTTP_cli_params_ptr;
        Key_Desc*                       key_info_ptr;
        int                             signal_command = HTTP_NO_SIGNAL;
        Queued_Object_Desc*             queued_object_ptr = OPC_NIL;
        int                             object_count;
        int                             number_of_send_objects;
        List*                           send_lptr;
        GnaT_HTTP_Request*              HTTP_req_ptr;



        /** This function opens a client TCP session. If the session pointer      **/
        /** is NIL, then a new session is started, otherwise the session                  **/
        /** is reused.                                                                    **/
        FIN (gna_clsvr_mgr_cache_spawn_session (sess_ptr, sess_command, server_index,
        HTTP_command, inline_objects_list, req_size, GnaT_Nam_Appl* appl_info_ptr));

        /* Create a structure which contains info about the HTTP session */
        HTTP_cli_params_ptr = (GnaT_Cli_HTTP_Params*) prg_cmo_alloc (cmo_handle, sizeof
(GnaT_Cli_HTTP_Params));

        /* No connections to this server are established as of now */
        /* Open a new connection now.                                                    */
        if ((sess_command == NEW_CONNECTION) || (sess_ptr == OPC_NIL))
        {
        if (trace_arch_active)
```

```
        {
        op_prg_odb_print_major ("gna_clsvr_mgr spawning process with \"gna_HTTP_cli\" process model ",
OPC_NIL);
                        }

        /* Create a session record for the client*/
                sess_ptr = (GnaT_Cli_Mgr_Session *) prg_cmo_alloc (cmo_handle, sizeof
(GnaT_Cli_Mgr_Session));
                sess_ptr->prohndl = op_pro_create ("gna_HTTP_cli", OPC_NIL);
                sess_ptr->sess_type = GNAC_SESSION_TYPE_ACTIVE;
                sess_ptr->objects_queued_lptr = op_prg_list_create ();
                key_info_ptr = (Key_Desc*) prg_cmo_alloc (cmo_handle, sizeof (Key_Desc));

                /* Insert the record in the session record handle */
                HTTP_cli_params_ptr->sess_key = oms_dt_item_insert (session_record_handle, sess_ptr);

                key_info_ptr->session_key = HTTP_cli_params_ptr->sess_key;
                HTTP_cli_params_ptr->sess_handle = session_record_handle;
                HTTP_cli_params_ptr->mgr_prohndle = op_pro_self ();

                /* Insert the key in the server key array */
                op_prg_list_insert (server_info_array_ptr [server_index]->server_session_lptr,
                                                        key_info_ptr, OPC_LISTPOS_TAIL);
                }

        /* Inform manager about received packets */
        signal_command = HTTP_SIGNAL_CACHE;

        if (inline_objects_list != OPC_NIL)
                {
                /* Send request to the server */
                /* Create a list of object sizes which will be passed to the HTTP client */
                send_lptr = op_prg_list_create ();

                /* Recalculate the size of the send list */
                number_of_send_objects = op_prg_list_size (inline_objects_list);

                /* Construct inline send list for the HTTP client */
                for (object_count = 0; object_count < number_of_send_objects; object_count++)
                {
                /* Allocate memory for size entry */
                HTTP_req_ptr = (GnaT_HTTP_Request*) prg_cmo_alloc (cmo_handle, sizeof
(GnaT_HTTP_Request));
                /* Get object info */
                queued_object_ptr = (Queued_Object_Desc*) op_prg_list_remove (inline_objects_list,
OPC_LISTPOS_HEAD);

                /* Set object size */
                HTTP_req_ptr->size = queued_object_ptr->size;
                strcpy (HTTP_req_ptr->group_name, "Not Used For Stats");
                /* Copy over the ca related information */
                strcpy (HTTP_req_ptr->custom_app_name, queued_object_ptr->custom_app_name);
                HTTP_req_ptr->hit_rate = queued_object_ptr->hit_rate;

                /* Insert object is the send list */
                op_prg_list_insert (send_lptr, HTTP_req_ptr, OPC_LISTPOS_TAIL);

                /* Free memory */
                op_prg_mem_free (queued_object_ptr);
                }

                /* Reinitialize the list, just in case.. */
        op_prg_list_init (inline_objects_list);
        HTTP_cli_params_ptr->packet_sizes_lptr = send_lptr;
        HTTP_cli_params_ptr->exp_req_size = req_size;
        HTTP_cli_params_ptr->cache_pkt_id = pkt_id;

        }
        else
                {
                HTTP_cli_params_ptr->packet_sizes_lptr = OPC_NIL;
                HTTP_cli_params_ptr->exp_req_size = 0;
                HTTP_cli_params_ptr->cache_pkt_id = 0;
                }
```

```
/* Set the name of the server, HTTP type and size */
            HTTP_cli_params_ptr->server_name = (char*) op_prg_mem_copy_create (server_info_array_ptr
[server_index]->server_name,sizeof (char) * (strlen (server_info_array_ptr [server_index]->server_name) + 1));
            HTTP_cli_params_ptr->app_info_ptr = appl_info_ptr;
            HTTP_cli_params_ptr->serv_index = server_index;
            HTTP_cli_params_ptr->command = HTTP_command;
            HTTP_cli_params_ptr->signal = signal_command;
            HTTP_cli_params_ptr->end_time = OPC_DBL_INFINITY;

/*****************************************************************************************/
/
// SET TOS Cache CDF
// Alterações Realizadas no Servidor Cache Original.
// Verifica-se qual o TOS da requisição feita pelo cliente e estabelece o mesmo tipo de conexão entre o Servidor Cache
CDF e o Internet Server
//
/*****************************************************************************************/
if(!strcmp(app_cdf,"EF"))
 {
 HTTP_cli_params_ptr->tos = 184;
 HTTP_cli_params_ptr->command = 500;
 sess_ptr->type_of_service = 184;
 }
 else
 if(!strcmp(app_cdf,"AF31"))
 {
 HTTP_cli_params_ptr->tos = 104;
 HTTP_cli_params_ptr->command = 400;
 sess_ptr->type_of_service = 104;
 }
 else
 if(!strcmp(app_cdf,"AF11"))
 {
 HTTP_cli_params_ptr->tos = 40;
 HTTP_cli_params_ptr->command = 300;
 sess_ptr->type_of_service = 40;
 }

 else if(!strcmp(app_cdf,"BE"))
 {
 HTTP_cli_params_ptr->tos = 0;
 HTTP_cli_params_ptr->command = 200;
 sess_ptr->type_of_service = 0;
 }

/* Invoke the HTTP client */
op_pro_invoke (sess_ptr->prohndl, HTTP_cli_params_ptr);
FOUT;
}
```

Figura D.1– Código do Servidor *Cache* alterado para dar suporte a diferenciação de serviços

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo