MARCELO DA MOTA LOPES

Uma Arquitetura Orientada a Serviço para Aplicações com Restrições Temporais

São Paulo
2011

MARCELO DA MOTA LOPES

Uma Arquitetura Orientada a Serviço para Aplicações com Restrições Temporais

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia.

São Paulo
2011

MARCELO DA MOTA LOPES

Uma Arquitetura Orientada a Serviço para Aplicações com Restrições Temporais

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Engenharia.

Área de Concentração:
Sistemas Digitais.

Orientador:
Prof. Livre-Docente Jorge Rady de Almeida Júnior

São Paulo
2011

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo 02 de junho de 2011.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Lopes, Marcelo da Mota
Uma arquitetura orientada a serviço para aplicações com restrições temporais / M. da M. Lopes. -- ed.rev. -- São Paulo, 2011.
127 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sistemas de informação (Confiabilidade;Disponibilidade) I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

DEDICATÓRIA

A Maria Angélica, minha esposa, pelo incondicional apoio, sempre presente durante o desenvolvimento deste trabalho, às minhas filhas Amanda e Camila, pela paciência e compreensão nas inúmeras horas que tive de me ausentar e aos meus pais, Fernando e Elza, que abriram as portas e iluminaram o caminho por onde tudo começou.

AGRADECIMENTOS

Ao Prof. Dr. Jorge Rady de Almeida Júnior pelo apoio e dedicação na orientação deste trabalho.

Ao professor Dr. Paulo Sérgio Cugnasca e a professora Dra. Selma Shin Shimizu Melnikoff pelas contribuições e sugestões para o desenvolvimento deste trabalho.

Aos demais colegas e professores do GAS, Grupo de Análise de Segurança da Escola Politécnica da Universidade de São Paulo, pelo apoio e motivação.

RESUMO

Este trabalho apresenta uma proposta de Arquitetura Orientada a Serviço para desenvolvimento de aplicações com restrições temporais, isto é, aplicações em que o tempo de resposta a uma requisição deve respeitar limites máximos. No desenvolvimento da arquitetura proposta foram considerados os modelos de filas com um único servidor e com múltiplos servidores, por meio da utilização de serviços redundantes e do escalonamento de requisições para melhoria do determinismo no tempo de resposta das requisições efetuadas. Para avaliação da arquitetura proposta foi construído um sistema de testes de forma a ser observado o comportamento do tempo de resposta das requisições em função do número de servidores disponíveis e sua respectiva taxa de utilização. Os resultados obtidos indicam que é possível obter um aumento no determinismo do tempo de resposta das requisições efetuadas (diminuição da dispersão de valores), tendo sido obtidos resultados semelhantes para os dois algoritmos de escalonamento utilizados: por ordem de chegada das requisições e SRPT (*Shortest Remaining Processing Time*).

Palavras chave: Arquitetura Orientada a Serviços. Restrições temporais. Teoria das Filas.

ABSTRACT

This thesis presents a proposal for a Service Oriented Architecture applied to development of time constrained systems, where the timeliness of the results is a major requirement. The development is based on the queuing theory (models using one and multiple servers) and requests scheduling to improve response time determinism. In order to verify the proposal, a test system had been developed to observe the dynamic behavior of the requests response time dispersion according to the number of servers available and associated processing rate. The results obtained show an improvement over the request response time determinism and almost similar performance for the two scheduling algorithms used: request arrival order and SRPT (Shortest Remaining Processing Time).

Keywords: Service Oriented Architecture. Time constraints. Queuing theory.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura dos serviços *Web* (W3C, 2004). 27
Figura 2 – Formato de uma mensagem SOAP (COULOURIS, DOLLIMORE e KINDBERG, 2007). 30
Figura 3 – Exemplo de uma mensagem SOAP (W3C, 2004).....30
Figura 4 – Formato de uma descrição WSDL (W3C, 2003). 31
Figura 5 – Elementos principais do padrão arquitetural – adaptado de (BENZ, 2003) 33
Figura 6 – Modelo de fila (LIU, 2009). 35
Figura 7 – Notação para modelo de fila (ANTONIOL, CIMITILE, et al., 2004). 38
Figura 8 – Relação (R / S, U) para o sistema M / M / 1. 39
Figura 9 – Filas paralelas múltiplas (LIU, 2009). 40
Figura 10 – Fila única com múltiplos servidores (LIU, 2009).....40
Figura 11 – Comparação: múltiplas filas *versus* múltiplos servidores (M / M / 1). 42
Figura 12 – Abstração do SOA em um modelo de rede de filas. Adaptado de (LIU, 2009). 42
Figura 13 – Escalonamento SPRT aplicado em servidores *Web* (HARCHOL-BALTER, SCHROEDER, et al., 2003).....45
Figura 14 – Comparação entre a arquitetura dos serviços *Web* e a arquitetura proposta. 59
Figura 15 – Serviços *Web* desenvolvidos x Padrão WS-I. 59
Figura 16 – Clientes versus Serviço *Web* Distribuidor versus Servidores.....60
Figura 17 – Estrutura interna do serviço *Web* Distribuidor 62
Figura 18 – Protocolo FTT (MARAU, ALMEIDA e PEDREIRAS, 2006). 67
Figura 19 – Componentes do sistema de testes 73
Figura 20 – Plataforma do sistema de testes 75
Figura 21 – Execução do Simulador de Requisições 76
Figura 22 – Classe TServGenKernel.....77
Figura 23 – Execução do Simulador de Requisições 78
Figura 24 – Classe TThrCPUTimeLog 79
Figura 25 – Relacionamentos entre classes do Serviço 1.....80
Figura 26 – Classe T*Web*SrvBrokerImpl 81
Figura 27 – Classe TResult.....84
Figura 28 – Classe TThrRequestManager 85
Figura 29 – Classe TThr*Web*SrvCall.....85

LISTA DE ILUSTRAÇÕES

Figura 30 – Relacionamentos entre classes do serviço *Web* Distribuidor 86
Figura 31 – Gerenciamento das requisições pelo serviço *Web* Distribuidor 87
Figura 32 – *Thread* do TThrRequestManager 89
Figura 33 – Tempos em uma chamada 91
Figura 34 – Distribuição da latência 92
Figura 35 – Distribuição da latência entre 400ms e 650ms 93
Figura 36 – Distribuição do intervalor entre requisições 94
Figura 37 – Serviço 1: dispersão de valores (1 servidor, SRPT inativo) 95
Figura 38 – Serviço 1: histograma do tempo de resposta (1 servidor, SRPT inativo) 96
Figura 39 – Serviço 1: dispersão de valores (1 servidor, SRPT ativo) 96
Figura 40 – Serviço 1: histograma do tempo de resposta (1 servidor, SRPT ativo) .. 97
Figura 41 – Serviço 1: dispersão de valores (2 Servidores, SRPT inativo) 97
Figura 42 – Serviço 1: histograma do tempo de resposta (2 servidores, SRPT inativo) 98
Figura 43 – Serviço 1: dispersão de valores (2 servidores, SRPT ativo) 99
Figura 44 – Serviço 1: histograma do tempo de resposta (2 servidores, SRPT ativo) 99
Figura 45 – Serviço 1: dispersão de valores (3 servidores, SRPT inativo) 100
Figura 46 – Serviço 1: histograma do tempo de resposta (3 servidores, SRPT inativo) 101
Figura 47 – Serviço 1: dispersão de valores (3 servidores, SRPT ativo) 101
Figura 48 – Serviço 1: histograma do tempo de resposta (3 servidores, SRPT ativo) 102
Figura 49 – Serviço 2: dispersão de valores (1 servidor, SRPT inativo) 102
Figura 50 – Serviço 2: dispersão de valores (1 servidor, SRPT ativo) 103
Figura 51 – Serviço 2: histogramas do tempo de resposta (1 servidor) 103
Figura 52 – Serviço 2: dispersão de valores (2 servidores, SRPT inativo) 104
Figura 53 – Serviço 2: dispersão de valores (2 servidores, SRPT ativo) 104
Figura 54 – Serviço 2: dispersão de valores (3 servidores, SRPT inativo) 104
Figura 55 – Serviço 2: dispersão de valores (3 servidores, SRPT ativo) 105
Figura 56 – Serviço 2: histogramas do tempo de resposta (2 servidores) 105
Figura 57 – Serviço 2: histogramas do tempo de resposta (3 servidores) 106
Figura 58 – Serviço 3: dispersão de valores (1 servidor, SRPT inativo) 107
Figura 59 – Serviço 3: dispersão de valores (1 servidor, SRPT ativo) 107

LISTA DE ILUSTRAÇÕES

Figura 60 – Serviço 3: dispersão de valores (2 servidores, SRPT inativo).....107
Figura 61 – Serviço 3: dispersão de valores (2 servidores, SRPT ativo) 108
Figura 62 – Serviço 3: dispersão de valores (3 servidores, SRPT inativo).....108
Figura 63 – Serviço 3: dispersão de valores (3 servidores, SRPT ativo) 109
Figura 64 – Serviço 3: histogramas do tempo de resposta (1 servidor) 109
Figura 65 – Serviço 3: histogramas do tempo de resposta (2 servidores) 110
Figura 66 – Serviço 3: histogramas do tempo de resposta (3 servidores) 110
Figura 67 – Serviço 4: dispersão de valores (1 servidor, SRPT inativo) 111
Figura 68 – Serviço 4: dispersão de valores (1 servidor, SRPT ativo) 111
Figura 69 – Serviço 4: dispersão de valores (2 servidores, SRPT inativo).....112
Figura 70 – Serviço 4: dispersão de valores (2 servidores, SRPT ativo) 112
Figura 71 – Serviço 4: dispersão de valores (3 servidores, SRPT inativo).....112
Figura 72 – Serviço 4: dispersão de valores (3 servidores, SRPT ativo) 113
Figura 73 – Serviço 4: histogramas do tempo de resposta (1 servidor) 113
Figura 74 – Serviço 4: histogramas do tempo de resposta (2 servidores) 114
Figura 75 – Serviço 4: histogramas do tempo de resposta (3 servidores) 114
Figura 76 – Serviço 1: histograma consolidado.....115
Figura 77 – Serviço 2: histograma consolidado.....115
Figura 78 – Serviço 3: histograma consolidado.....116
Figura 79 – Serviço 4: histograma consolidado.....116

LISTA DE TABELAS

Tabela 1- Parâmetros e entidades considerados nos modelos de filas (LIU, 2009)..36
Tabela 2- Filas com prioridades baseadas no tamanho dos arquivos (HARCHOL-BALTER, SCHROEDER, et al., 2003).....46
Tabela 3- Classificação dos Tipos de Sistema de Tempo Real (LAPLANTE, 1992).49
Tabela 4- Mapeamento de requisitos na arquitetura proposta 60
Tabela 5- Filas com prioridades baseadas em restrição temporal. 64
Tabela 6- Tempos médios de execução de cada serviço. 94
Tabela 7- Comparação de tempos de resposta para 3 servidores.....117

LISTA DE EQUAÇÕES

Equação 1 36
Equação 2 37
Equação 3 37
Equação 4 39
Equação 5 39
Equação 6 40
Equação 7 41
Equação 8 41
Equação 9 41
Equação 10 41
Equação 11 67

LISTA DE ABREVIATURAS E SIGLAS

COTS	*Commercial-of-the-Shelf*
CSMA/CD	*Carrier Sense Multiple Access / Collision Detection*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
MAC	*Media Access Control*
OASIS	*Organization for the Advancement of Structured Information Standards*
OSI	*Open System Interconnection*
QoS	*Quality of Service*
RFC	*Request for Comments*
SLA	*Service Level Agreement*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SRPT	*Shortest Remaining Processing Time*
TCP	*Transmission Control Protocol*
UDDI	*Universal Description, Discovery and Integration*
URI	*Uniform Resource Identifier*
XML	*Extensible Markup Language*
WS	*Web Services*
WSDL	*Web Service Description Language*
WS-I	*Web Services Interoperability Organization*
W3C	*World Wide Web Consortium*

SUMÁRIO

1. INTRODUÇÃO 15
1.1. Motivação 16
1.2. Objetivo 18
1.3. Justificativa 19
1.4. Organização do Trabalho 20
2. ARQUITETURA ORIENTADA A SERVIÇO 22
2.1. Conceitos Básicos sobre SOA 22
2.2. Serviços *Web* (*Web Services*) 25
2.2.1. Protocolo SOAP– *Simple Object Access Protocol* 29
2.2.2. WSDL – *Web Service Description Language* 31
2.2.3. UDDI (*Universal Description, Discovery and Integration*) 32
2.3. Considerações Finais deste Capítulo 33
3. MODELOS DE FILAS 35
3.1. Conceitos Básicos sobre Teoria das Filas 35
3.2. Teoria das Filas Aplicada ao SOA 42
3.3. Considerações sobre Escalonamento 43
3.4. Considerações Finais deste Capítulo 46
4. SISTEMAS COMPUTACIONAIS COM RESTRIÇÕES TEMPORAIS 47
4.1. Restrições Temporais 47
4.2. Restrições Temporais Aplicadas ao SOA 49
4.3. Considerações Finais deste Capítulo 54
5. ESPECIFICAÇÃO DA ARQUITETURA 55
5.1. Requisitos 55
5.2. Arquitetura Proposta 56
5.3. Detalhamento do Serviço *Web* Distribuidor 60
5.3.1. Gerenciador das Filas de Requisições e Respostas 62
5.3.2. Escalonador 63
5.4. Trabalhos Correlacionados 64
5.5. Considerações Finais deste Capítulo 70
6. PROJETO DO AMBIENTE DE TESTES 71
6.1. Simulação da Arquitetura Proposta 71
6.2. Componentes do Sistema de Testes 73
6.3. Simulador de Requisições 75
6.4. Serviços *Web* 1, 2, 3 e 4 76
6.5. Serviço *Web* Distribuidor 81
6.5.1. Lógica de Controle 87

SUMÁRIO

6.6. Considerações Finais deste Capítulo 89
7. RESULTADOS OBTIDOS 90
7.1. Condições de Contorno 90
7.2. Verificação de variações nos tempos de resposta 94
7.2.1. Resultados das simulações para o Serviço 1 95
7.2.2. Resultados das simulações para o Serviço 2 102
7.2.3. Resultados das simulações para o Serviço 3 106
7.2.4. Resultados das simulações para o Serviço 4 110
7.3. Resultados Consolidados 114
7.4. Considerações Finais deste Capítulo 118
8. CONSIDERAÇÕES FINAIS 119
8.1. Contribuições 119
8.2. Trabalhos Futuros 121
8.3. Conclusões 121
9. REFERÊNCIAS BIBLIOGRÁFICAS 122

1. INTRODUÇÃO

A Arquitetura Orientada a Serviço (SOA) é um paradigma para organização e utilização de competências distribuídas que estão sob o controle de diferentes domínios proprietários (OASIS, 2006). Essas competências (conhecimentos, habilidades e atitudes), criadas por entidades (pessoas e organizações) para resolução de problemas encontrados no decorrer de seus negócios, podem, então, ser utilizadas por outras entidades na resolução de problemas semelhantes, dependendo de condições de contorno, como compatibilidade de requisitos, por exemplo.

Nesse paradigma, serviços com baixo nível de acoplamento são fornecidos em rede para suporte ao gerenciamento de processos de negócio (MAURIZIO, SAGER, et al., 2008). Por serviço entende-se um componente de software que encapsula uma função, tem uma interface bem definida, possui um conjunto de operações identificadas e um conjunto de mensagens que o serviço recebe e envia (CUOMO, 2005).

Nos últimos anos, a Arquitetura Orientada a Serviço tem sido apresentada como uma tendência para o desenvolvimento de aplicações, devido às vantagens apresentadas por esse tipo de arquitetura, como baixo nível de acoplamento e possibilidade real de reuso e de incorporação de sistemas legados, sem que esses tenham de ser completamente reescritos (SEI, 2008).

Como a Arquitetura Orientada a Serviço se trata de um padrão arquitetural aberto, diversas arquiteturas podem ser derivadas desse padrão, incluindo aquelas que não são destinadas a sistemas de uso geral, mas sim para aplicações específicas.

Dessa forma, uma possível abordagem para uma arquitetura derivada do padrão arquitetural mencionado seria a utilização da infraestrutura consolidada, com larga aplicação no mundo corporativo, para aplicação em um tipo mais restrito de sistema, como, por exemplo, sistemas com restrições temporais. Como exemplo de restrição temporal tem-se o tempo máximo (previamente estabelecido) para execução de uma operação fornecida por um determinado serviço.

1.1. Motivação

Diversas são as vantagens na aplicação da Arquitetura Orientada a Serviço, tornando-a uma das abordagens mais adequadas ao desenvolvimento e integração de sistemas, bem como no encapsulamento de sistemas legados para compatibilização com esse tipo de arquitetura (SEI, 2007).

Além desse fato, os custos decrescentes e a melhoria de qualidade do hardware e software, facilitando a implantação e gerenciamento de redes de comunicação, têm contribuído diretamente para o aumento da utilização desse tipo de sistema, uma vez que, na Arquitetura Orientada a Serviço, o acesso às operações disponíveis em cada serviço ocorre por meio de troca de mensagens, normalmente por meio de redes de comunicação.

De acordo com um relatório do *Gartner Group*, em 2007, 50% das novas aplicações e processos de negócios operando em missão crítica foram projetadas utilizando conceitos de SOA (GARTNER GROUP, 2007).

Em 2008, o *Gartner Group* efetuou uma nova pesquisa (*Gartner 2008 SOA User Survey: Adoption Trends and Characteristics*) incluindo mais de 200 empresas de grande porte (com mais de 1000 funcionários), a qual indicou que 53% já haviam utilizado a Arquitetura Orientada a Serviço em alguma parte da organização e que outros 25%, que ainda não a estavam utilizando, tinham planos para sua utilização nos próximos 12 meses (GARTNER GROUP, 2008).

É fato, então, a intensificação da aplicação desse tipo de paradigma no desenvolvimento de sistemas corporativos. Sua crescente utilização aumenta o interesse, por parte do mundo corporativo, no estabelecimento de padrões para sua construção e, preferencialmente, convergentes em um padrão arquitetural básico.

O estabelecimento de padrões arquiteturais para um determinado tipo de tecnologia favorece a sua utilização, visto que diminui o risco de investimentos no desenvolvimento de sistemas pelas empresas, pois são maiores as probabilidades de compatibilidade entre sistemas de fornecedores distintos e menores as dificuldades em se encontrar alternativas no mercado para novos desenvolvimentos, entre outros fatores.

Dessa forma, estabelece-se o seguinte círculo virtuoso: o desenvolvimento de aplicações com alto grau de compatibilidade simplifica a integração entre sistemas, independentemente da plataforma de execução e da linguagem de programação

utilizada no seu desenvolvimento, favorecendo sua utilização, aumentando a probabilidade de convergência para um padrão arquitetural básico, que por sua vez aumenta a compatibilidade entre sistemas, facilitando a sua aplicação em mais áreas (aumentando a sua utilização).

Embora o aumento na utilização da Arquitetura Orientada a Serviço traga inúmeros benefícios, seja no desenvolvimento de sistemas (possibilidade de reuso, baixo acoplamento entre serviços, etc.), seja na oferta de serviços disponíveis para utilização por usuários finais, existe a contrapartida no aumento da dependência desse tipo de tecnologia por parte dos usuários.

Essa dependência gera a expectativa, por parte dos usuários, que esses sistemas estejam disponíveis 100% do tempo: transações bancárias devem sempre ser completadas corretamente, reservas de passagens aéreas precisam ocorrer sem erros, carros alugados para viagens de férias devem estar disponíveis na chegada dos viajantes e assim por diante.

A demanda por altos níveis de disponibilidade implica a necessidade de se estabelecer estratégias para que esses valores possam ser atingidos. A disponibilidade de um sistema pode ser definida como a probabilidade de um sistema estar em funcionamento, de acordo com suas especificações, em um determinado instante de tempo (JOHNSON, 1989).

Vale ressaltar que, dependendo da especificação do sistema, não necessariamente um componente deve deixar de funcionar completamente para que o sistema seja considerado indisponível. Por exemplo, dependendo de sua especificação, um aumento no tempo de resposta das operações solicitadas além de um valor pré-estabelecido pode indicar uma falha no sistema.

O baixo grau de acoplamento entre os componentes (serviços) de um sistema construído considerando o padrão arquitetural do SOA é uma característica que favorece não somente o reuso de componentes ou a agilidade no desenvolvimento de sistemas. Essa característica também pode ser utilizada no estabelecimento de redundância entre os componentes de um sistema. Por exemplo, cópias de um serviço podem ser executadas simultaneamente em computadores distintos, permitindo que aplicações possam ter seu grau de disponibilidade incrementado.

A redundância de componentes pode também ser utilizada para aplicação de escalonamento de requisições. Estudos apresentam resultados em que o emprego de escalonamento, baseado na teoria das filas, aumenta a possibilidade de melhoria

no tempo de resposta de servidores *Web* percebido pelos usuários (BIERSACK, SCHROEDER e URVOY-KELLER, 2007), por exemplo.

No caso sistemas com restrições temporais, isto é, sistemas em que o tempo de resposta à solicitação de execução de uma operação deve atender limites mínimos e máximos pré-estabelecidos, a utilização de redundância no atendimento às solicitações de um determinado cliente pode permitir que eventuais variações no tempo de resposta (dentro de limites pré-estabelecidos) possam ser mitigadas (utilização de instâncias adicionais para atendimento às requisições) ou, pelo menos, que possam ser parametrizadas de acordo com a taxa de utilização do sistema.

Dessa forma, a Arquitetura Orientada a Serviço pode ser utilizada também na construção de sistemas em que o atendimento a restrições temporais são determinantes para o seu correto funcionamento.

1.2. Objetivo

Este trabalho tem como objetivo principal apresentar uma arquitetura derivada do padrão arquitetural SOA (Arquitetura Orientada a Serviço) a ser empregada no desenvolvimento de sistemas com restrições temporais, por meio de escalonamento de componentes redundantes do sistema. Busca-se, dessa forma, aumentar o determinismo no tempo de resposta na execução de operações fornecidas pelos serviços que constituem o sistema.

Assim, não é uma meta desta tese criar um novo padrão completamente diferente dos já existentes, mas sim adaptar um padrão existente, de modo que possa ser aplicado no desenvolvimento de aplicações com restrições temporais para obtenção da resposta a solicitações de um cliente a um provedor de um serviço.

O desenvolvimento do trabalho está apoiado no estudo dos padrões de arquitetura utilizados para construção de um modelo e sua respectiva utilização na construção de um sistema de software. Porém, vale ressaltar que discussões aprofundadas desses padrões (protocolo SOAP, por exemplo) não fazem parte do escopo desta tese, pois, tanto nas pesquisas, quanto no desenvolvimento do ambiente de testes, esses padrões são utilizados sem modificações, seja para descrição das interfaces dos serviços criados para testes, seja para formatação de mensagens a serem trocadas pelos serviços. A arquitetura desenvolvida encontra-se em um nível acima dos padrões aqui considerados, utilizando-os como suporte no desenvolvimento.

O ambiente de testes desenvolvido possibilita a simulação de situações para verificar a viabilidade de aplicação da arquitetura apresentada. O seu projeto e construção são baseados em componentes e sistemas comercialmente disponíveis (COTS – *Commercial Off-The-Shelf*), incluindo sistema operacional, compilador, servidores HTTP (*HyperText Transfer Protocol*), entre outros.

Esse ambiente de testes tem como função principal permitir que, a partir da execução de simulações, dados sejam obtidos e analisados estatisticamente, de forma que possa ser avaliado o comportamento dinâmico de um sistema baseado na arquitetura proposta.

Além dos componentes existentes na arquitetura tomada como base para o desenvolvimento, técnicas de escalonamento e multiprogramação também são consideradas no desenvolvimento, pois correspondem ao mecanismo utilizado para gerenciamento das requisições e respectivas chamadas a serviços para execução.

1.3. Justificativa

A Arquitetura Orientada a Serviço tornou-se um paradigma tecnológico com utilização em larga escala no desenvolvimento e integração de sistemas, incluindo sistemas legados (SEI, 2007) (LI, HUAI, et al., 2009).

Pelo fato de se tratar de um padrão arquitetural e não uma arquitetura específica, o SOA (*Service Oriented Architecture*), normalmente utilizado na integração de sistemas por meio da composição dinâmica de serviços *Web* existentes na Internet (LI, HUAI, et al., 2009), permite que inúmeras arquiteturas possam ser desenvolvidas a partir desse padrão arquitetural inicial (LEWIS, MORRIS, et al., 2007).

Assim sendo, arquiteturas específicas podem ser planejadas de forma que determinadas classes de sistemas possam se valer de características disponíveis nesse padrão arquitetural para satisfazer seus requisitos específicos, como, por exemplo, sistemas com restrições temporais. A aplicação do SOA em sistemas de Tempo Real, que são, por definição, sistemas com restrições temporais, foi analisada por TSAI, LEE, et al. (2006), que apresenta uma possível arquitetura para aplicação do SOA em sistemas de tempo real, incluindo considerações de projeto para que seja possível obter-se comunicação, processamento e composição de serviços em tempo real. No trabalho de PRÜTER, MORITZ, et al. (2008) também são apresentados resultados sobre a aplicação de SOA em tempo real, indicando

que é possível determinar tempos máximos de processamento e transferência de dados para serviços conhecidos, quando as transferências de dados são executadas em redes de comunicação fechadas.

Ressalta-se que os sistemas desenvolvidos a partir da arquitetura proposta nesta tese são aqueles classificados como sistemas de missão crítica que tenham seu desempenho degradado, porém não comprometido severamente na ocorrência de uma falha em seu funcionamento. Ainda que seja considerada a situação em que o seu desempenho seja severamente degradado, esse estado de funcionamento não deve resultar em danos materiais nem a pessoas, ou seja, a latência não é considerada um requisito crítico para segurança, mas um requisito crítico para o negócio (atendimento a metas do negócio, por exemplo) (O'BRIEN, MERSON e BASS, 2005).

Exemplos de sistemas que podem ser incluídos nessa classe incluem sistemas de monitoração, indicadores de manutenção em indústrias, sistemas de suporte ao relacionamento com o consumidor, sistemas de suporte à tomada de decisão, entre outros.

1.4. Organização do Trabalho

Este trabalho está organizado de acordo com a seguinte estrutura:

No capítulo 1 é apresentada uma introdução contendo o objetivo, a motivação e a justificativa para o trabalho proposto.

O capítulo 2 contém uma descrição dos principais conceitos sobre o padrão arquitetural SOA (*Service Oriented Architecture* ou Arquitetura Orientada a Serviço) e sua implementação mais comum, os serviços *Web*. São apresentados os conceitos básicos que determinam a estrutura dessa arquitetura, para que seja possível fundamentar a proposta desta tese.

Os conceitos básicos sobre a teoria das filas e seu emprego na estimativa de tempo de resposta de sistemas computacionais de acordo com sua taxa de utilização, incluindo sistemas baseados na Arquitetura Orientada a Serviço, são apresentados no capítulo 3. Nesse capítulo também são apresentados conceitos sobre escalonamento e como sua utilização pode auxiliar na redução do tempo de resposta de sistemas computacionais.

No capítulo 4 são apresentados conceitos sobre os sistemas que apresentam restrições temporais e as características que identificam e classificam esse tipo de sistema. Esses conceitos são então aplicados a sistemas baseados na Arquitetura Orientada a Serviço, sendo relacionadas às principais características desse padrão arquitetural com os conceitos referentes às restrições temporais em sistemas computacionais.

A arquitetura proposta neste trabalho, que tem por objetivo permitir a aplicação da Arquitetura Orientada a Serviço em sistemas com restrições temporais, é apresentada no capítulo 5.

No capítulo 6 apresenta-se a implementação da arquitetura proposta, bem como um sistema de testes para execução de simulações. São descritas as técnicas para implementação da arquitetura proposta, bem como as considerações e premissas de projeto adotadas.

O capítulo 7 contém a descrição dos resultados obtidos para as simulações efetuadas com o sistema projetado e implementado (arquitetura proposta e ambiente de testes).

Finalmente, no capítulo 8 são apresentadas as considerações finais e conclusões sobre os resultados obtidos e as possibilidades abertas para estudos futuros.

2. ARQUITETURA ORIENTADA A SERVIÇO

Este capítulo apresenta os conceitos básicos sobre a Arquitetura Orientada a Serviço (SOA – *Service Oriented Architecture*), objeto de estudo desta tese, bem como sobre sua implementação mais comum, os serviços *Web*.

2.1. Conceitos Básicos sobre SOA

O crescimento da aplicação da Arquitetura Orientada a Serviço (SOA) nos sistemas corporativos apresenta-se como uma tendência, mesmo com padrões em constante evolução. Esse tipo de arquitetura é, tipicamente, caracterizado pelas seguintes propriedades (W3C, 2004):

- **Visão Lógica**:
  Um serviço pode ser definido como um componente de software que encapsula uma função, tem uma interface bem definida, possui um conjunto de operações identificadas e um conjunto de mensagens que recebe e envia (CUOMO, 2005). Assim, um serviço corresponde a uma visão lógica abstrata de programas, bancos de dados, processos de negócios, entre outros. É definido em termos do que faz, tipicamente atuando como uma operação no nível de negócio.
- **Orientação a Mensagens**:
  Um serviço é formalmente definido pelas mensagens trocadas entre provedores e consumidores e pelas características dos agentes participantes. A estrutura interna de um agente (linguagem de programação utilizada na sua implementação, estrutura de base de dados, etc.) são abstraídas, devendo ser transparentes para quem utiliza (ou fornece) um serviço. Esse tipo de característica é particularmente útil na migração de sistemas legados, que podem ser encapsulados em um serviço sem que sua estrutura interna tenha de ser exposta.
- **Orientação a Descrição**:
  Um serviço é descrito por metadados passíveis de processamento por computadores. Nessa descrição somente devem ser incluídos os detalhes públicos e importantes para sua utilização.

- **Granularidade**:
  Serviços tendem a utilizar um pequeno número de operações com mensagens relativamente grandes e complexas.
- **Orientação a Rede de Comunicação**:
  Serviços normalmente são utilizados em um ambiente conectado por meio de redes de comunicação, embora esse não seja um requisito absoluto.
- **Neutralidade de Plataforma**:
  Mensagens são enviadas em um formato padrão, independente de plataforma. Um formato que atende essas restrições, por exemplo, é o XML (*Extensible Markup Language*).

A Arquitetura Orientada a Serviço ou SOA (*Service Oriented Architecture*) corresponde a um padrão arquitetural (e não a uma arquitetura, como indica a tradução direta da sigla), o qual pode ser utilizado para o desenvolvimento de arquiteturas de sistemas orientados a serviço (LEWIS, MORRIS, et al., 2007).

Um padrão arquitetural corresponde a uma descrição de elementos e tipos relacionados com um conjunto de restrições sobre como eles podem ser utilizados (BASS, CLEMENTS e KAZMAN, 2003). Ele orienta a criação de uma arquitetura nele baseada, de forma que seja possível alavancar a utilização das melhores práticas previamente incorporadas ao padrão arquitetural original.

Em Fevereiro de 2005, o consórcio OASIS – *Organization for the Advancement of Structured Information Standards* – constituiu um comitê técnico para o desenvolvimento do modelo de referência para o SOA (OASIS *Service Oriented Architecture Reference Model Technical Committee*, abreviado como OASIS SOA-RM TC) (OASIS, 2005).

Um modelo de referência é um *framework* abstrato para entendimento dos relacionamentos significantes entre as entidades e algum ambiente que habilite o desenvolvimento de arquiteturas específicas usando padrões consistentes ou especificações que suportem esse ambiente (OASIS, 2006).

O modelo de referência desenvolvido pelo OASIS não está associado a nenhum tipo particular de implementação ou conjunto de padrões, mas define o vocabulário e identifica os elementos comuns existentes em uma Arquitetura Orientada a Serviço (OASIS, 2006).

Na versão 1.0, o modelo de referência define a Arquitetura Orientada a Serviço como sendo um paradigma para a organização e utilização de competências

distribuídas que estão sob controle de diferentes domínios proprietários. Essa organização permite o desenvolvimento de soluções com alto grau de reuso, escalabilidade e interoperabilidade.

Nos processos de negócios, as necessidades de execução de tarefas e as competências para executá-las existem independentemente da Arquitetura Orientada a Serviço (SOA), porém, no SOA, um serviço corresponde ao mecanismo pelo qual as necessidades e as competências são colocadas de forma conjunta.

Os serviços constituem-se no ponto central do padrão arquitetural e consideram as seguintes características (OASIS, 2006):

- Possuir a competência de executar uma tarefa no lugar de outro serviço ou sistema;
- Publicar a especificação da tarefa a ser executada para outro serviço ou sistema;
- Ofertar a execução de uma tarefa para outro serviço ou sistema.

Dessa forma, um serviço corresponde a um mecanismo que habilita o acesso a uma ou mais competências, por meio da definição de uma interface, cujo acesso baseia-se no respeito às políticas de acesso e restrições, conforme especificadas na descrição do serviço. Cada serviço deve possuir uma descrição de suas funções e suas interfaces, isto é, entradas, saídas e semânticas associadas ao serviço (OASIS, 2006).

Um serviço é fornecido por um ou mais provedores de serviço (*service providers*) e consumido pelos consumidores do serviço (*service consumers*). Normalmente, os consumidores de um serviço não tomam conhecimento de como o processamento das informações solicitadas é executado, ou seja, o serviço corresponde a uma caixa preta, para a qual somente são detalhadas suas interfaces, as quais expõem os modelos de informação e o comportamento do serviço, para que seja possível determinar se um serviço atende às necessidades de um consumidor.

A descrição de um serviço e a respectiva política de acesso, fornecida pelo provedor, são necessárias para o estabelecimento de mecanismos de descoberta, de forma que um consumidor possa descobrir os serviços disponíveis. O mecanismo de descoberta possibilita a interação entre consumidores e provedores:

- O Provedor de Serviço **publica** os serviços disponíveis e **responde** a chamados de um ou mais consumidores de serviços;

- O Consumidor de Serviço **busca** os serviços disponíveis e **efetua a chamada** de serviço para o Provedor de Serviço escolhido.

Segundo o padrão SOA (OASIS, 2006), a descrição do serviço deve incluir dados suficientes de forma que um consumidor de serviço e o respectivo provedor possam interagir. Essa descrição pode incluir metadados, como, por exemplo, a localização do serviço e quais protocolos de informação o serviço suporta e/ou requer. Além da descrição, também podem ser incluídas informações sobre o comportamento dinâmico do serviço, como indicações do estado do serviço (disponível ou indisponível), por exemplo.

Embora seja um padrão arquitetural que permita a construção de diversas arquiteturas, aquelas derivadas do padrão arquitetural determinado pelo SOA compartilham algumas características básicas como: **encapsulamento** (*self-contained*), **baixo acoplamento** (*loose coupling*) e **troca assíncrona de mensagens**.

A característica de **encapsulamento** refere-se ao fato de que os serviços que compõem a arquitetura do sistema devem possuir fronteiras (interfaces) bem definidas, não exigindo a exposição de seu funcionamento nem de suas estruturas internas.

O **acoplamento** de um software refere-se ao grau de interdependência entre os módulos que o compõe. Aplicando-se esse conceito ao SOA, tem-se que os serviços que compõem um sistema devem ser unidades funcionais independentes, porém coesas, isto é, com um nível de acoplamento (interdependência) baixo.

A **troca assíncrona de mensagens** permite que seja minimizado o tempo de espera do consumidor por uma resposta, pois, fazendo uma chamada assíncrona (para a qual existirá outra mensagem, contendo a resposta), o cliente não tem, necessariamente, que aguardar pela resposta do serviço, podendo executar outras tarefas enquanto o processamento de sua requisição é executado (BROWN, JOHNSTON e KELLY, 2002).

2.2. Serviços *Web* (*Web Services*)

Dentre as arquiteturas baseadas no SOA, destaca-se a denominada serviços *Web* (*Web Services*), definida em 2004 pelo *World Wide Web Consortium* (W3C), por

meio da identificação de seus componentes funcionais e os seus respectivos relacionamentos (W3C, 2004).

Nessa arquitetura, um Serviço *Web* é definido como um sistema de software projetado para suportar interação entre dois computadores sobre uma rede de comunicação, sendo a descrição de sua interface feita em um formato que possibilite que o processamento seja realizado por um computador. A Figura 1 apresenta a estrutura em camadas de um Serviço *Web* típico.

De forma geral, serviços *Web* são apropriados quando aplicados em sistemas com as seguintes características (W3C, 2004):

- Operam sobre a Internet, situação em que a confiabilidade e a velocidade de transmissão de pacotes de dados não pode ser garantida;
- Não é possível fazer com que todos os clientes e provedores sejam atualizados com a versão mais recente do sistema de uma só vez;
- Os componentes do sistema distribuído são executados em diferentes plataformas, de diferentes fabricantes;
- Uma aplicação existente (legada) necessita ser exposta para uso sobre a rede e pode ser encapsulada como um serviço *Web*.

Dentre os diversos grupos de padrões estabelecidos pelo consórcio OASIS, alguns são escolhidos e mantidos como especificação pelo consórcio WS-I (*Web Services Interoperability Organization*) (WS-I, 2010).

Esse consórcio tem como objetivo estabelecer melhores práticas a fim de promover a interoperabilidade entre grupos selecionados de padrões referentes aos serviços *Web*. A interoperabilidade considera independência de plataforma, sistemas operacionais e linguagens de programação. Dessa forma, mesmo que atualizações em determinadas especificações sejam propostas pelo consórcio OASIS, o consórcio WS-I mantém versões estáveis de especificações para manutenção e garantia de interoperabilidade.

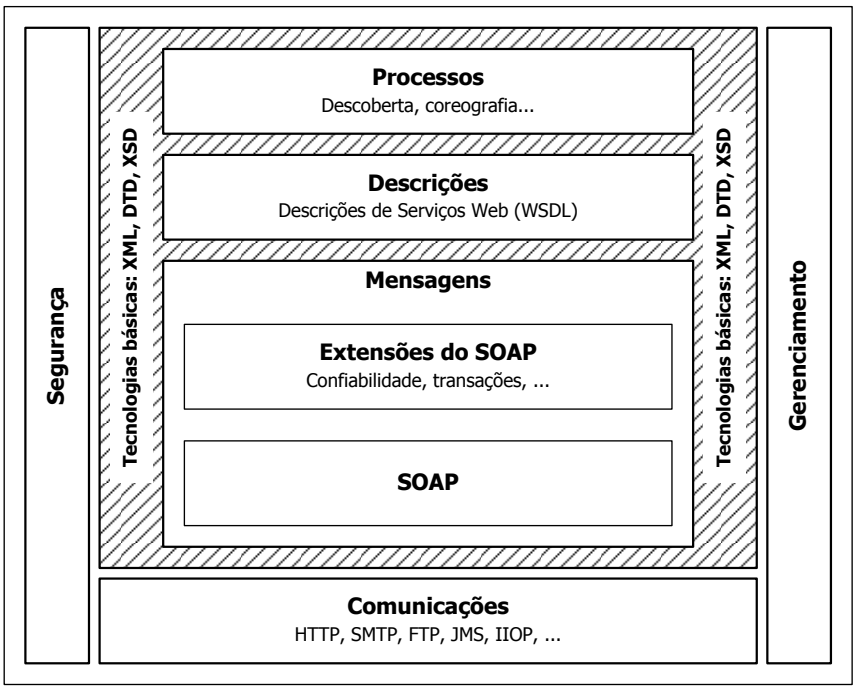


Figura 1 – Arquitetura dos serviços *Web* (W3C, 2004).

Na Figura 1 é possível observar a existência de seis seções principais:

- **Processos**:

  Um processo é basicamente um programa de computador em execução, ao qual está associado um conjunto de registradores (contador de programa e ponteiro de pilha, por exemplo) e o seu espaço de endereçamento, no qual é armazenado o código executável, os dados do programa e sua pilha (TANENBAUM, 2003). No caso dos serviços *Web*, os processos correspondem aos serviços propriamente ditos.

  Na Figura 1, são indicados os processos de **Descoberta** de serviços (*Discovery*) e **Coreografia** (*Choreography*), que também podem estar presentes em um sistema baseado nessa arquitetura.

  O processo de Descoberta corresponde à determinação dinâmica (em tempo de execução do serviço consumidor) do endereço de um serviço provedor que satisfaça os requisitos estabelecidos pelo serviço consumidor. Normalmente, é utilizado um serviço com a capacidade de executar operações de pesquisa em um repositório de dados sobre serviços disponíveis, como, por exemplo, um repositório UDDI – *Universal Description, Discovery and Integration* (LEWIS, MORRIS, et al., 2007).

A descoberta e composição de serviços, de forma dinâmica, são características que podem ser implementadas em sistemas baseados em SOA, permitindo que serviços alternativos sejam substituídos em caso de falha funcional ou desempenho (CHEN e ZHANG, 2007).

A Coreografia define a sequência e as condições sob as quais múltiplos agentes cooperativos efetuam trocas de mensagens para que uma tarefa seja executada ou para que se atinja um determinado objetivo. Vale ressaltar que podem ser confundidos os termos Coreografia e Orquestração, porém, no caso da Orquestração, a interação entre os agentes é coordenada por um único participante, ao contrário da Coreografia, em que múltiplos agentes cooperam entre si (OASIS, 2010).

- **Descrições de Serviços *Web***:

  Um serviço *Web* apresenta uma interface que é descrita utilizando-se uma linguagem específica, formatada em XML, denominada WSDL (*Web Service Description Language*). A descrição em WSDL descreve serviços *Web*, começando pelas mensagens que são trocadas entre os agentes consumidor e provedor (W3C, 2004).

  Embora seja uma linguagem para descrição de uma interface, o acrônimo WSDL é comumente referenciado como sendo a descrição da interface de um determinado serviço.

- **Mensagens**:

  A interação entre consumidores e provedores de serviços é efetuada por meio de mensagens formatadas no protocolo SOAP (*Simple Object Access Protocol*), tipicamente transportadas pelo protocolo HTTP (*HyperText Transfer Protocol*). Vale ressaltar que discussões aprofundadas desses protocolos não fazem parte do escopo desta tese.

- **Comunicações**:

  Visto que os serviços *Web* dependem da rede de comunicação para troca de mensagens, a camada de transporte de mensagens é fundamental para o funcionamento dessa solução, podendo ser utilizados protocolos de transporte como HTTP (*HyperText Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*), JMS (*Java Message Service*) ou IIOP (*Internet Inter-Orb Protocol*, sendo *Orb* o acrônimo de *Object Request Broker*), por exemplo.

- **Gerenciamento**:

  O gerenciamento de serviços *Web* é possível por meio de um conjunto de competências que habilitam o monitoramento, controle e emissão de relatórios sobre características de qualidade e taxas de utilização de um serviço. Como características de qualidade podem ser incluídas: o nível de disponibilidade (número de instâncias do serviço), desempenho (latência e taxas de falha) e acessibilidade (número de nós disponíveis).

- **Segurança**:

  Técnicas para proteção de mensagens transmitidas entre provedores e consumidores de serviços podem ser aplicadas para que seja aumentada a segurança na utilização de serviços *Web*.

  Entre as técnicas utilizadas, podem ser citadas: criptografia e assinatura de mensagens XML (RFC 2807, RFC 3275), *Web Services Security* (WSS), desenvolvida pelo consórcio OASIS, a qual define uma extensão para o protocolo SOAP, provendo integridade, confidencialidade e autenticação de mensagens, *XML Key Management Specification*, que utiliza criptografia de chave pública para criptografar, assinar, autorizar e verificar a autenticidade da informação transmitida na Internet, entre outras.

2.2.1. Protocolo SOAP– *Simple Object Access Protocol*

O protocolo SOAP – *Simple Object Access Protocol* – (W3C, 2000) é o protocolo padrão utilizado para formatação de mensagens transmitidas entre consumidores e provedores de serviços *Web*. Ele provê uma estrutura padrão extensível e configurável para empacotamento e troca de mensagens baseadas em XML (W3C, 2003).

De forma geral, uma mensagem SOAP representa a informação necessária para invocação de um serviço ou reflete os resultados de uma invocação previamente efetuada, contendo a informação especificada na interface de definição do serviço.

Quando se utiliza a representação RPC (*Remote Procedure Call*) do SOAP, uma mensagem SOAP representa a invocação de um método em um objeto remoto. Nesse caso, a serialização da lista de argumentos do método local deve ser movida para o ambiente remoto. Como serialização entende-se a transformação de um

objeto em uma sequência de *bits* a ser transmitida, a qual é recuperada pelo receptor e utilizada na reconstrução do objeto (processo inverso).

Uma mensagem SOAP é formatada em XML e apresenta uma seção principal, denominada envelope. Esse envelope contém duas subseções, denominadas Cabeçalho (*Header*) e Corpo (*Body*), que definem a mensagem propriamente dita, conforme indicado na Figura 2.

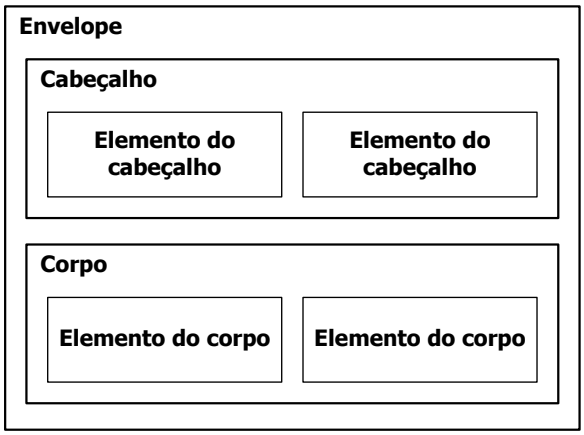


Figura 2 – Formato de uma mensagem SOAP (COULOURIS, DOLLIMORE e KINDBERG, 2007).

A Figura 3 apresenta um exemplo de uma mensagem SOAP, contendo um bloco de cabeçalho (delimitado pelos *tags* <env:Header> e </env:Header>) e um bloco que corresponde ao corpo da mensagem (delimitado pelos *tags* <env: Body > e </env:Body>).

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
   <n:priority>1</n:priority>
   <n:expires>2001-06-22T14:00:00-05:00</n:expires>
  </n:alertcontrol>
 </env:Header>
 <env:Body>
  <m:alert xmlns:m="http://example.org/alert">
   <m:msg>Pick up Mary at school at 2pm</m:msg>
  </m:alert>
 </env:Body>
</env:Envelope>
```

Figura 3 – Exemplo de uma mensagem SOAP (W3C, 2004).

É possível verificar ainda, na Figura 3, que as seções de cabeçalho e corpo da mensagem são delimitadas pelo envelope da mensagem (definido pelos *tags* <env:Envelope> e </env: Envelope >).

2.2.2. WSDL – *Web Service Description Language*

O WSDL é uma linguagem utilizada para descrição de serviços *Web*, incluindo a descrição das mensagens trocadas entre agentes consumidores e provedores de serviços. As mensagens propriamente ditas são descritas de forma abstrata e estão ligadas a protocolos de rede e formatadas para transmissão (parte concreta da mensagem) (W3C, 2003).

Um documento WSDL típico contém descrições para serviços de rede como sendo um conjunto de nós (terminações) em uma rede, operando sobre mensagens contendo informações orientadas a documentos (dados a serem transmitidos) e orientadas a procedimentos (serviço a ser executado).

O modelo conceitual do WSDL considera a linguagem de descrição como sendo um conjunto de componentes contendo determinadas propriedades. Cada componente da descrição é representado por um conjunto de informações formatadas em XML.

Nesse nível de abstração, o componente de definições engloba duas categorias de componentes: componentes WSDL (interfaces, ligações e serviços) e componentes de sistema (definições de elementos e tipos de dados vinculados a um determinado tipo de sistema).

Um componente de sistema corresponde ao conjunto de definições de elementos (identificados), cada um associado (de forma isomórfica, isto é, em relacionamentos um a um) a um tipo simples ou complexo definido no esquema XML. Normalmente, elementos de um esquema XML fazem referência aos tipos de dados definidos.

Uma mensagem WSDL é composta por um ou mais conjuntos de informações (*Infosets*), que corresponde a um ou mais elementos do tipo ***definitions***, indicados no esquema XML representado na Figura 4 (W3C, 2003):

```
<definitions targetNamespace="xs:anyURI" >
<documentation />? [ <import /> | <include /> ]* <types />? [ <interface /> | <binding /> | <service /> ]*
</definitions>
```

Figura 4 – Formato de uma descrição WSDL (W3C, 2003).

Um elemento ***definitions*** é composto pelos seguintes elementos:

- ***targetNamespace***: um *namespace* representa um nome (não ambíguo) para a semântica do conjunto de informações que constituem o documento WSDL. O URI (*Uniform Resource Identifier*) identificado no targetNamespace (*namespace* objetivo) deve apontar para uma localização que contenha um

documento que possa ser processado por uma máquina ou por um ser humano.

- ***Documentation*** **(Documentação)**: contém descrições com propósito de documentação do modelo WSDL, de forma que possa ser lido por humanos ou processado por computadores.
- ***Import*** **(Importação):** contém o conjunto de documentos WSDL pertencentes a diferentes *namespaces* objetivo, que são agregados à definição WSDL corrente.
- ***Include*** **(Inclusão):** contém o conjunto de documentos WSDL a serem agregados à definição WSDL corrente e que fazem referência ao mesmo *namespace* objetivo da definição WSDL corrente.
- ***Types*** **(Tipos de dados)**: esquemas descritos em XML que descrevem as estruturas de dados utilizadas nas interfaces;
- ***Interface*** **(Interface)**: conjunto de definições (identificadas) das interfaces;
- ***Binding*** **(Ligação)**: conjunto de ligações (identificadas). Um componente de ligação descreve o formato de uma mensagem e o protocolo de transmissão utilizado na definição de uma terminação (*endpoint*), isto é, local em que um determinado serviço está disponível;
- ***Service*** **(Serviço)**: conjunto de definições (identificadas) de um serviço.

2.2.3. UDDI (*Universal Description, Discovery and Integration*)

O UDDI (*Universal Description, Discovery and Integration*) corresponde a um registro que armazena dados sobre um grupo de serviços, os quais expõem informações sobre lógica de negócios e suas interfaces.

Os registros UDDI são baseados em servidores *Web* e mantidos por múltiplos *sites* e podem ser acessados (gratuitamente) por aqueles que desejam publicar informações sobre um ou mais negócios (ou entidades), bem como por aqueles que desejam encontrar esse tipo de informação (OASIS, 2002). A Figura 5 apresenta a visão geral da arquitetura dos serviços *Web*, incluindo o registro UDDI.

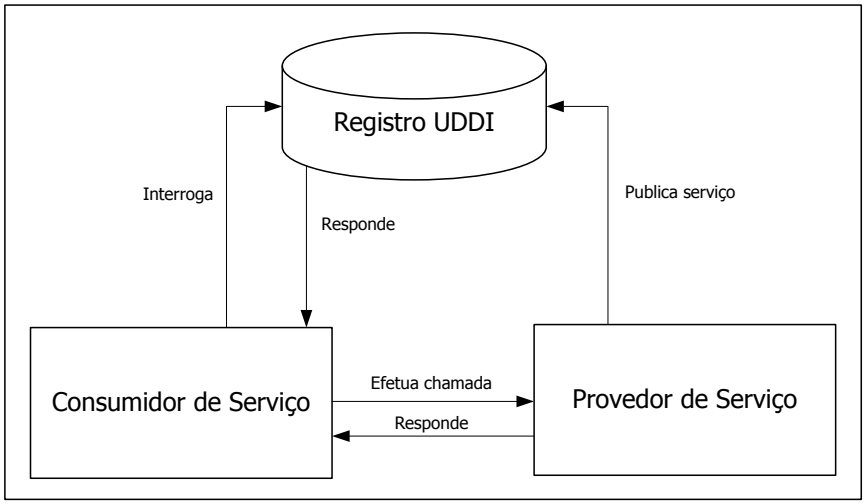


Figura 5 – Elementos principais do padrão arquitetural – adaptado de (BENZ, 2003)

A estrutura do UDDI é baseada nos padrões definidos para os serviços *Web*, o que significa que um registro UDDI pode ser acessado pelos mesmos meios que os demais serviços *Web*. Resumidamente, o UDDI é um serviço *Web* cuja principal função é servir de diretório para serviços *Web* nele publicados (BENZ, 2003).

O benefício de se acessar esse tipo de informação reside na possibilidade de permitir a outros sistemas interagir com os serviços fornecidos por um determinado negócio, como comércio eletrônico, por exemplo.

Os tipos de informação que podem ser publicadas incluem diversos tipos de dados simples que auxiliam na busca de serviços por meio de respostas a perguntas como: quem?, o que?, onde? e como?

Por exemplo, podem ser cadastradas informações como nome de negócios, identificadores de negócios, códigos universais de produtos e classificações de produtos (o que?), informações para contato (quem?), URLs (*Uniform Resource Locators*) ou endereços de e-mail (onde?) e descrições das interfaces para conexão com os serviços (como?).

2.3. Considerações Finais deste Capítulo

Neste capítulo foram apresentadas as principais características do padrão arquitetural SOA e os principais componentes de uma arquitetura derivada desse padrão, os serviços *Web*.

A partir dessa visão geral e dos conceitos introduzidos nos capítulos seguintes, como teoria das filas, escalonamento e restrições temporais, é fundamentada uma

proposta para desenvolvimento de uma arquitetura baseada no padrão SOA, visando sua utilização no desenvolvimento de sistemas com restrições temporais.

Assim, o Capítulo 3 introduz conceitos básicos sobre teoria das filas e como esses conceitos podem ser utilizados no dimensionamento de sistemas computacionais, considerando parâmetros como sua taxa de utilização, por exemplo.

3. MODELOS DE FILAS

Este capítulo apresenta conceitos básicos sobre teoria das filas, utilizados na construção de modelos analíticos que permitem estudar o relacionamento do tempo de resposta de um sistema computacional com o seu nível de utilização. Esses modelos são, posteriormente, aplicados na Arquitetura Orientada a Serviço.
No final do capítulo também são introduzidos conceitos sobre escalonamento e sua utilização na melhoria de tempos de resposta de sistemas computacionais.

3.1. Conceitos Básicos sobre Teoria das Filas

A teoria das filas é um campo cujos estudos datam de 1900, tendo sido aplicada, com sucesso, na resolução de uma grande variedade de problemas: projeto de redes telefônicas, gerenciamento de centros de serviço, alocação de mão de obra em centros de reparo e manutenção, etc. (ANTONIOL, CIMITILE, et al., 2004).
Nesta seção são introduzidos conceitos básicos sobre a teoria das filas, suficientes para entendimento dos modelos avaliados, que são posteriormente considerados na avaliação da arquitetura proposta nesta tese. Não se trata, portanto, de um estudo aprofundado sobre a teoria das filas. Para consultas mais aprofundadas deve ser verificada a referência (LIU, 2009).
O modelo básico de fila considera requisições feitas por clientes para execução de um serviço, por um ou mais servidores. Ao final da execução de um serviço executado o resultado é entregue ao cliente que fez a solicitação. Novas requisições recebidas que não puderem ser imediatamente atendidas por um dos servidores são colocadas em uma fila, para posterior atendimento, conforme ilustrado na Figura 6.

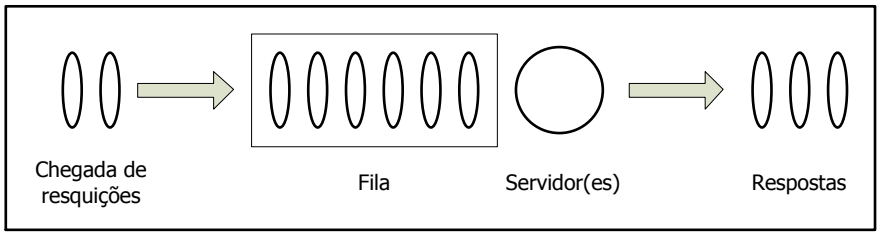


Figura 6 – Modelo de fila (LIU, 2009).

Em se tratando de um modelo matemático, que permite formalizar estudos sobre o desempenho e a escalabilidade de sistemas computacionais, definições sobre

entidades e parâmetros considerados nessa modelagem são necessários para o estabelecimento desse formalismo. A Tabela 1 apresenta as principais entidades, parâmetros e simbologia (quando aplicável) considerados nessa abordagem.

Tabela 1- Parâmetros e entidades considerados nos modelos de filas (LIU, 2009)

Parâmetro	Símbolo	Descrição
Servidor	Não se aplica	Entidade responsável pelo atendimento de requisições dos consumidores.
Consumidor	Não se aplica	Entidade que requisita um serviço a ser prestado (atendido) por um servidor.
Requisição	Não se aplica	Solicitação feita por um consumidor a um servidor de um serviço a ser prestado.
Fila	Não se aplica	Entidade que armazena as requisições recebidas que não puderam ser atendidas imediatamente.
Comprimento da Fila	N	Número de requisições no sistema, incluindo as que estão em serviço.
Tempo de Espera	W	Tempo médio decorrido entre a entrada de uma requisição na fila e o início do seu atendimento.
Tempo de Serviço	S	Intervalo de tempo médio decorrido entre os inícios de tratamento de duas requisições consecutivas.
Tempo de Resposta	R	Tempo médio total entre a chegada de uma requisição e o seu atendimento. Corresponde à soma do Tempo de Espera com o Tempo de Serviço.
Taxa de Chegada	λ	Número médio de requisições recebidas em por unidade de tempo.
Taxa de Utilização	U	Porcentagem de tempo que um servidor permanece executando operações.
Taxa de Serviço	X	Número de consumidores que são atendidos por unidade de tempo (*Throughput*).

A lei de Little (LIU, 2009) mostra que as três métricas mais utilizadas na teoria das filas (taxa de serviço, tempo de resposta e comprimento da fila) são correlacionadas entre si, sendo o número de consumidores (em atendimento ou aguardando atendimento) igual ao produto da taxa de serviço (X) pelo tempo de resposta (R), conforme apresentado na Equação 1:

$$N = X \times R \qquad \text{Equação 1}$$

O processo de recebimento de requisições, enfileiramento e execução corresponde a um processo dinâmico, para o qual não se pode determinar o número de requisições recebidas em um período de tempo, por ser aleatório por natureza, nem estabelecer o tempo de execução de um serviço, por não ser constante. Por se tratar, então, de um processo probabilístico, os modelos utilizados na teoria das filas correspondem a modelos probabilísticos.

Um dos modelos probabilísticos mais utilizados é o modelo de Markov, para o qual os estados futuros do processo em estudo independem de sua história passada, dependendo somente do estado presente. Para aplicação do processo de Markov na teoria das filas, são feitas as seguintes considerações (LIU, 2009):

- A taxa de chegada de requisições (variável discreta) segue a distribuição de probabilidade de Poisson, representada na Equação 2:

$$p(k) = \frac{(\lambda \mathrm{t})^k e^{-\lambda t}}{k!} \qquad \text{Equação 2}$$

  Por exemplo, para um sistema em que a taxa média de chegada de requisições corresponde a uma requisição a cada dois segundos (λ = 0,5) e o tempo de observação empregado é de dez segundos (t = 10), tem-se que a probabilidade de se observar a chegada de seis requisições (k = 6) nesse período é de apenas 14,6%.
- Os tempos entre chegadas de requisições (variável contínua) seguem a distribuição Exponencial, representada na Equação 3:

$$f(t) = \lambda e^{-\lambda t} \qquad \text{Equação 3}$$

- Os tempos de serviço (variável contínua) seguem a distribuição Exponencial, conforme representado na Equação 3.

O formalismo matemático também é considerado na notação utilizada para caracterização do modelo de fila. A notação de Kendall (KENDALL, 1953) descreve o modelo de filas baseando-se em vários aspectos do sistema considerado, apresentando o formato indicado na Figura 7.

A / B / m / a_1 / a_2, sendo:

A:	Tipo de distribuição probabilística do tempo entre recebimento de requisições;
B:	Tipo de distribuição probabilística do tempo de serviço;
m:	Número de servidores;
a_1:	Capacidade da fila;
a_2:	Política utilizada no gerenciamento da fila.

Figura 7 – Notação para modelo de fila (ANTONIOL, CIMITILE, et al., 2004).

Na Figura 7 os tipos de distribuição normalmente utilizados são Markoviana (M), Determinística (D), Erlang (E_k) ou Geral (G) e as políticas de gerenciamento das filas indicam como as requisições são retiradas da fila: FIFO (*First-In-First-Out*), LIFO (*Last-In-Last-Out*), retirada aleatória e retirada por prioridade. Caso os dois últimos parâmetros sejam omitidos (**a_1** e **a_2**), assume-se que a fila tem capacidade infinita e a política de gerenciamento da fila é FIFO (ANTONIOL, CIMITILE, et al., 2004).

O modelo M / M / 1 aberto (sem realimentação) é um modelo que pode ser tratado analiticamente e possui boa precisão para sistemas que operam em condição de equilíbrio. Sempre que fórmulas analíticas são utilizadas para predição de comportamento de sistemas, devem ser respeitados o contexto de aplicação e as condições nas quais o modelo foi desenvolvido. Assim, para o modelo em questão, devem ser observadas as seguintes condições (LIU, 2009):

1) O sistema não está perto da saturação e sua utilização está abaixo de 70%, valor a partir do qual a razão entre o tempo de resposta (R) e o tempo de serviço (S) aumenta consideravelmente;
2) A fila do sistema está vazia ou com baixa ocupação, significando que o sistema não está sendo executado perto da saturação e pode ainda receber novas requisições sem que sejam observados impactos significativos em seu desempenho.

Pela notação de Kendall (Figura 7), esse modelo (M / M / 1 aberto) representa uma fila com chegada de requisições com distribuição Markoviana, tempo de serviço com

distribuição exponencial, 1 servidor, fila com capacidade infinita e política FIFO de gerenciamento de fila (os dois últimos parâmetros são assumidos com esses valores, caso sejam omitidos na especificação).

Esse modelo assume três condições de contorno (LIU, 2009):

1) A taxa de chegada (λ) é conhecida;
2) Todas as requisições recebidas são tratadas (taxa de serviço é igual à taxa de chegada, ou X = λ), não havendo realimentação de saídas na entrada a fim de completar uma operação;
3) A demanda total do sistema é conhecida e determinada pela Equação 4, sendo que V corresponde ao número de visitas à fila.

$$D = V \times S \qquad \text{Equação 4}$$

Nessas condições, o tempo de resposta (R) é determinado pela Equação 5:

$$R = W + S = \frac{S}{1 - U} \qquad \text{Equação 5}$$

A partir da Equação 5, pode-se observar que para pequenas taxas de utilização (U), o valor do tempo de resposta (R) é aproximadamente o valor do tempo de serviço (S). É possível ainda relacionar a taxa de utilização com a razão entre o tempo de resposta e o tempo de serviço, de acordo com a taxa de utilização, conforme indicado na Figura 8.

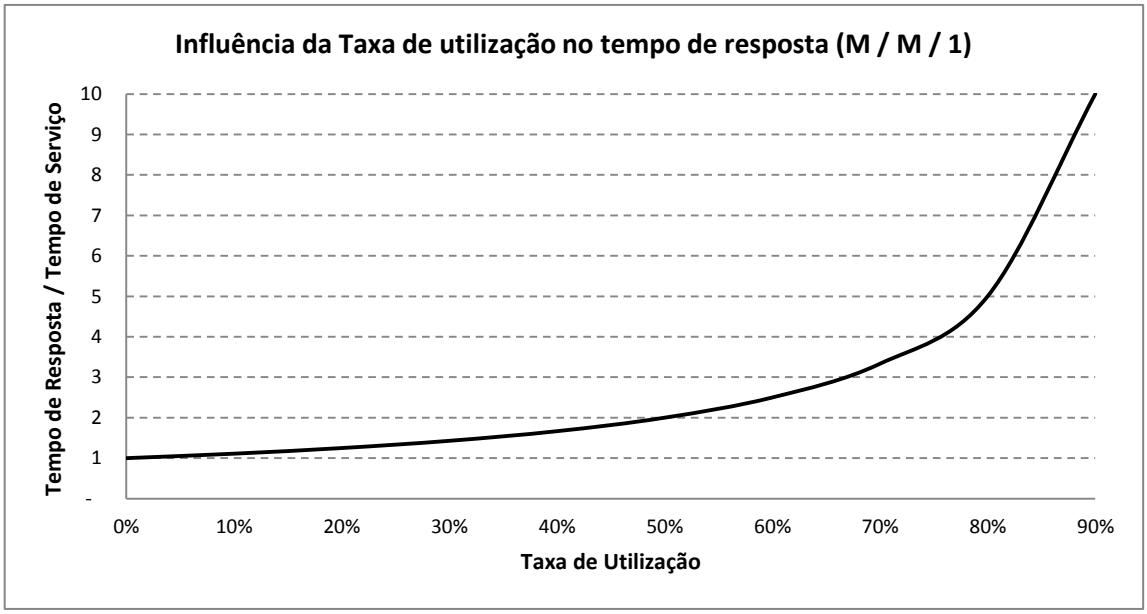


Figura 8 – Relação (R / S, U) para o sistema M / M / 1.

Visto que a taxa de utilização é determinante para o desempenho do sistema, isto é, o tempo de resposta corresponde ao tempo de serviço multiplicado por um fator não linear (relacionado diretamente com a taxa de utilização), manter o sistema com baixos níveis de taxa de utilização permite que o tempo de resposta possa ser mantido dentro de limites estimados. Assim, o desempenho e a escalabilidade de um software podem ser melhorados controlando-se a taxa de utilização do sistema, e, nesse caso, são propostas duas opções (LIU, 2009): filas paralelas múltiplas (Figura 9) ou fila única com múltiplos servidores (Figura 10).

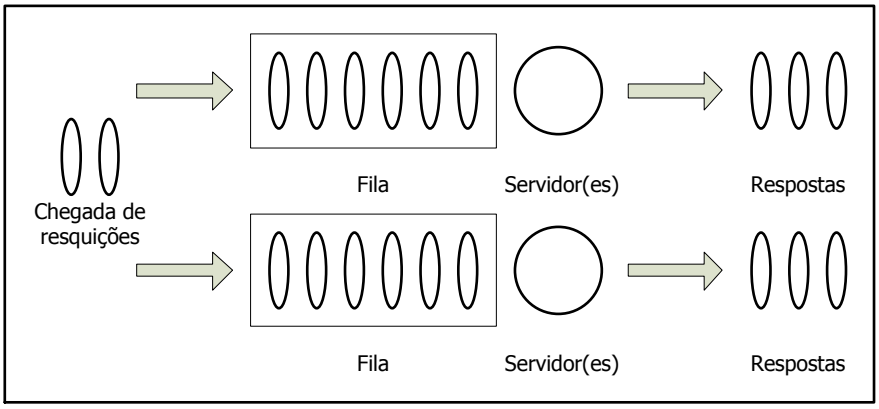


Figura 9 – Filas paralelas múltiplas (LIU, 2009).

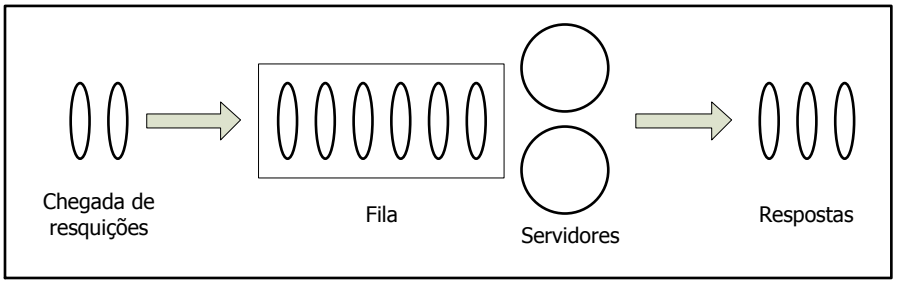


Figura 10 – Fila única com múltiplos servidores (LIU, 2009).

Considerando-se os dois casos mencionados, tem-se:

- Caso 1: Filas paralelas múltiplas:

  Nesse caso, considerando que existam ***m*** filas paralelas, normalizando o valor da taxa de utilização, conforme a Equação 6:

$$\rho = \frac{U}{m}, \qquad 0 \leq \rho < 1 \qquad \text{Equação 6}$$

A Equação 5 ainda se aplica e, substituindo-se U por ρ, obtém-se a Equação 7:

$$R = \frac{S}{1-\rho} \qquad \text{Equação 7}$$

- Caso 2: Fila única com múltiplos servidores:

  Nesse caso, considera-se uma fila única com ***m*** servidores associados. O valor da taxa de utilização normalizado ainda se aplica (Equação 6), porém o tempo de resposta depende a função C de Erlang, apresentada na Equação 8:

$$C(m,\rho) = \frac{(m\rho)^m}{m!} \Big/ \left[(1-\rho)\sum_{k=0}^{m-1}\frac{(m\rho)^k}{k!} + \frac{(m\rho)^m}{m!}\right] \qquad \text{Equação 8}$$

  A partir da Equação 8, pode-se determinar o tempo de resposta, apresentado na Equação 9:

$$R = S\left[1 + \frac{C(m,\rho)}{m(1-\rho)}\right] \qquad \text{Equação 9}$$

  Simplificando a Equação 9, pode ser obtido o valor aproximado do tempo de resposta, apresentado na Equação 10:

$$R \approx \frac{S}{1-\rho^m}, \qquad 0 \leq \rho < 1 \qquad \text{Equação 10}$$

Visto que o valor da taxa de utilização normalizada (ρ) varia entre zero e um, os valores de ρ são muito maiores que os valores de ρ^m, para grandes valores de m. Dessa forma, pode-se observar que o denominador da Equação 10 aumenta com o aumento do valor de m, logo o tempo de resposta do sistema com fila única e múltiplos servidores (Equação 10) tende a ser menor que o tempo de resposta para sistemas com filas paralelas múltiplas (Equação 7), conforme ilustrado na Figura 11.

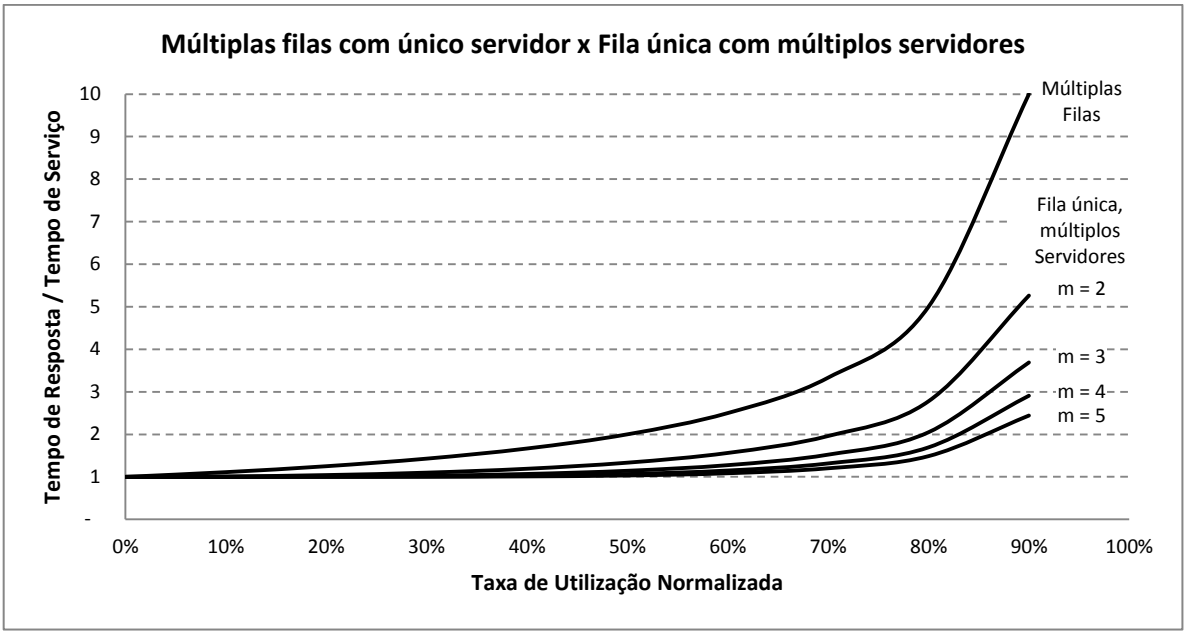


Figura 11 – Comparação: múltiplas filas *versus* múltiplos servidores (M / M / 1).

3.2. Teoria das Filas Aplicada ao SOA

O modelo de fila M / M / 1 aberto pode ser aplicado na modelagem de uma arquitetura derivada do padrão arquitetural SOA, para estimativa de capacidade e desempenho. A Figura 12 apresenta uma rede de filas nas quais os principais processos são modelados por filas.

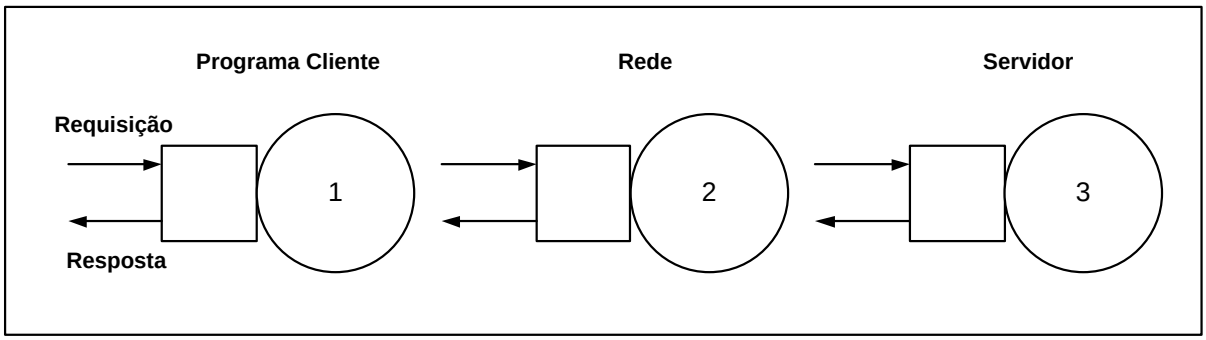


Figura 12 – Abstração do SOA em um modelo de rede de filas. Adaptado de (LIU, 2009).

Na Figura 12 existem três filas que encapsulam os principais processos envolvidos entre uma requisição efetuada por um cliente e o recebimento da respectiva resposta: requisição efetuada pelo cliente, transferência de dados pela rede para o servidor (solicitação do serviço), execução do serviço pelo servidor para obtenção da resposta, transferência de dados pela rede para o cliente e, finalmente, o recebimento da resposta pelo cliente.

Vale ressaltar ainda que serviços também podem ser clientes de outros serviços, sendo as filas apresentadas na Figura 12 replicadas a cada chamada de um serviço por outro. Essa abordagem também pode ser empregada no processo de descoberta dinâmica de serviços (não indicado na Figura 12), visto que aplicações baseadas em SOA têm a habilidade de descobrir e compor serviços em tempo de execução, permitindo que serviços alternativos sejam substituídos em caso de falha funcional ou desempenho (CHEN e ZHANG, 2007).

As filas apresentadas na Figura 12 poderiam ser decompostas, ainda, em outras filas, que representariam processos internos a cada fila indicada. Por exemplo, para o cliente (fila identificado pelo número um na Figura 12), poderiam ser considerados os processos de requisição e criação dos objetos para acesso ao servidor. No caso do servidor (fila identificada pelo número três na Figura 12), poderia ser citada a fila referente ao acesso a uma base de dados, por exemplo.

A fila referente à rede (identificada pelo número dois na Figura 12) representa a latência de rede (referente às tarefas de decomposição e recomposição de pacotes de dados para transmissão e recepção e roteamento de pacotes, entre outras.).

No caso de redes locais, a latência de rede pode apresentar valores bem abaixo dos valores de latência encontrados em redes externas (como Internet, por exemplo). Redes locais permitem, ainda, maior controle e atuação de administradores para gerenciamento de desempenho em relação a redes externas.

Por exemplo, o tamanho das mensagens que trafegam na rede também influencia no seu desempenho, pois pode variar de poucos kilobytes até vários megabytes, fato que pode aumentar a latência dependendo da largura de banda disponível na rede (LIU, 2009). No caso de redes locais, essa largura de banda pode ser gerenciada com maior facilidade, visto que todo hardware e software existente na rede está sob gerência de um ou mais administradores.

3.3. Considerações sobre Escalonamento

O escalonamento é utilizado por sistemas operacionais, em computadores multiprogramados, para gerenciamento de processos que competem pelos mesmos recursos (processador, memória e sistema de discos, por exemplo). O escalonador determina qual processo deve ter acesso ao processador, por meio de um algoritmo de escalonamento (TANENBAUM, 2003).

Algoritmos de escalonamento distintos possuem políticas de escalonamento distintas, que determinam como os processos serão servidos. Por exemplo, pode ser atendido primeiro aquele que for o primeiro a chegar (o primeiro a ter entrado em uma fila), ou o que apresenta tempo de execução menor (previamente conhecido), ou ainda o escalonamento pode ser feito por alternância circular (*round-robin*), entre outros.

A teoria das filas permite verificar que a utilização de políticas de escalonamento afeta de forma significativa o desempenho percebido pelo usuário. Por exemplo, políticas de escalonamento que priorizam execuções de processos de tempo de execução menor introduzem melhorias significativas no tempo de resposta. Porém, esse tipo de política quase não é utilizado em sistemas atuais, pois esses são significativamente mais complexos que aqueles modelados por uma fila M / M / 1 e, ainda, porque existe a possibilidade que processamentos que demandem maior tempo não sejam atendidos (BIERSACK, SCHROEDER e URVOY-KELLER, 2007).

Além do gerenciamento da competição por recursos, o escalonamento pode ser utilizado para melhoria da eficiência no consumo desses recursos e, ainda, prover menores tempos de resposta para todas as requisições de um sistema. Estudos apresentam resultados em que inclusive sistemas que apresentam períodos transientes de sobrecarga, como servidores *Web*, podem se beneficiar do escalonamento para obter melhoria nos seus tempos de resposta. Vale ressaltar que, de acordo com a modelagem apresentada para o sistema M / M / 1, o aumento da taxa de utilização de um sistema (nesse caso, proporcionado pelo aumento de tráfego em períodos difíceis de serem previstos) tem relação direta com o aumento do tempo de resposta.

No estudo apresentado por (HARCHOL-BALTER, SCHROEDER, et al., 2003), no qual foi implementada a política de escalonamento SRPT (*Shortest Remaining Processing Time*) em servidores *Web*, foram obtidos resultados que indicam reduções significativas nos tempos médios de resposta e latência de rede, além do fato que não foi verificada penalização na execução de processos que demandavam maior quantidade de recursos. A política de escalonamento utilizada prioriza a execução de requisições que demandem arquivos menores, reduzindo o tempo médio de espera na fila e, consequentemente, o tempo de resposta.

A modificação proposta em (HARCHOL-BALTER, SCHROEDER, et al., 2003) consiste na utilização de filas múltiplas para atendimento das requisições (de acordo com sua prioridade), conforme indicado na Figura 13.

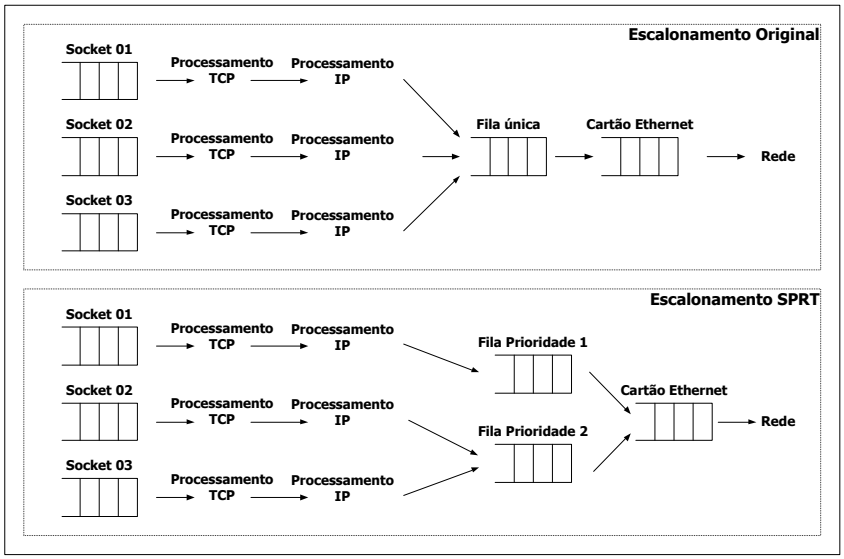


Figura 13 – Escalonamento SPRT aplicado em servidores *Web* (HARCHOL-BALTER, SCHROEDER, et al., 2003).

Assim, os processos que, na versão original, aguardam em fila única (com tempo de atendimento do servidor compartilhado), são armazenados em um conjunto de filas, com prioridades distintas de atendimento.

Para que os processos de menor prioridade sejam atendidos pelo servidor, todos os processos de todas as filas com maior prioridade devem ser atendidos primeiro. A prioridade é estabelecida de acordo com o tamanho do arquivo solicitado pelo processo e, tendo sido estabelecidas, para testes, seis filas (quanto menor o índice, maior a prioridade), conforme indicado na Tabela 2.

Tabela 2- Filas com prioridades baseadas no tamanho dos arquivos (HARCHOL-BALTER, SCHROEDER, et al., 2003)

Prioridade	Tamanho do arquivo solicitado (KBytes)
1	≤ 1K
2	1K – 2K
3	2K – 5K
4	5K – 20K
5	20K – 50K
6	> 50K

A Tabela 2 apresenta faixas de valores que determinam a prioridade de execução de cada solicitação ao servidor. No caso de objetivos com maior nível de complexidade, pode ser utilizada uma função utilidade (TIAN, LI, et al., 2009), na qual poderiam ser considerados além do tamanho do arquivo solicitado, parâmetros como demanda corrente e modificações na capacidade do sistema (largura de banda, por exemplo), entre outros.

3.4. Considerações Finais deste Capítulo

Neste capítulo foram apresentados conceitos sobre teoria das filas e escalonamento, que fundamentam a proposta a ser desenvolvida neste trabalho, a qual pretende associar algumas características presentes na Arquitetura Orientada a Serviço para o desenvolvimento de uma arquitetura que permita a implantação de sistemas com restrições temporais.

Em relação aos sistemas com restrições temporais, o capítulo 4, a seguir, introduz conceitos sobre restrições temporais em sistemas computacionais e principais fatores a serem considerados em sistemas desse tipo.

4. SISTEMAS COMPUTACIONAIS COM RESTRIÇÕES TEMPORAIS

Neste capítulo são descritos conceitos básicos sobre sistemas computacionais que apresentam restrições temporais, bem como a classificação aplicada a esse tipo de sistema, de acordo com o impacto no meio em que está inserido, caso essas restrições não sejam atendidas. No final do capítulo, são correlacionados os conceitos apresentados com a Arquitetura Orientada a Serviço.

4.1. Restrições Temporais

Um sistema pode ser definido como o mapeamento de um conjunto de entradas em um conjunto de saídas (LAPLANTE, 1992). Essa definição, embora genérica, expressa a função principal de sistemas de software que, por meio da execução de um conjunto de instruções por um hardware específico, transforma estímulos de entrada em respostas ou atuações no meio em que o sistema está inserido.

Considerando-se um hardware ideal para execução do sistema, a somatória dos tempos envolvidos no processamento dessas entradas, para que fossem produzidas respostas na sua saída, seria zero, o que não ocorre em aplicações no mundo real, visto que o tempo de processamento é sempre maior que zero.

Desta forma, pode-se definir o tempo de resposta de um sistema como sendo o atraso existente entre a apresentação do estímulo de entrada e a sua reação ou fornecimento das saídas associadas.

Em sistemas de uso geral (processadores de textos, por exemplo) atrasos no tempo de resposta não acarretam graves consequências, como danos ao patrimônio e a pessoas.

Porém, existem sistemas em que atrasos no tempo de resposta podem causar prejuízos materiais e pessoais. Por exemplo, em sistemas de comércio eletrônico, atrasos no tempo de resposta de transações podem reduzir o número total de transações comerciais efetivadas em um determinado período (danos materiais). Em sistemas de controle metroviário, esse tipo de atraso pode acarretar em consequências ainda mais graves, pois além de prejuízos materiais podem ocorrer prejuízos à vida humana.

Sistemas que devem atender a restrições explícitas de tempos de resposta sob o risco de que, se não forem atendidas, podem ocorrer disfunções que acarretem no

desencadeamento de graves consequências são denominados Sistemas de Tempo Real (LAPLANTE, 1992).

Uma disfunção (*failure*) é a consequência da exteriorização de um erro (*error*) que, por sua vez, corresponde a uma ou mais falhas (*fault*) em um módulo do sistema (JOHNSON, 1989).

Para que seja possível identificar se um sistema apresenta uma disfunção (ou não) é necessário tomar como referência a especificação de seus requisitos. Desta forma, considera-se que há uma disfunção em um sistema se este deixar de atender a um ou mais requisitos presentes em sua especificação formal.

Para sistemas com restrições temporais, a verificação desses requisitos não deve ser feita baseando-se somente nos estados das saídas de acordo com suas entradas, mas também por meio da verificação de que as saídas foram geradas respeitando-se os tempos de resposta máximos especificados.

Na verdade, para esse tipo de sistema, os tempos máximos de resposta também devem fazer parte da especificação de requisitos do sistema e sua verificação deve incluir, além da avaliação estática de Entradas x Saídas, a observação de seu comportamento dinâmico (tempos de resposta do sistema para as entradas associadas).

Tanto para sistemas de tempo real, como para sistemas de uso geral, a ocorrência de disfunções pode conduzir o sistema a um estado indisponível, porém, dependendo do tipo de aplicação, as consequências de uma disfunção (perda de *timing*) possuem níveis de gravidade diferentes, que auxiliam no estabelecimento de uma classificação para esse tipo de sistema (*soft*, *firm* ou *hard*), conforme indicado na Tabela 3.

Vale ressaltar que esta classificação não é absoluta, pois dois sistemas distintos podem possuir classificações diferentes de acordo com sua aplicação. Um sistema pode ser classificado como *soft* em relação ao aspecto de danos físicos a uma pessoa (falha em uma transação de um caixa eletrônico, por exemplo), porém o mesmo sistema pode ser classificado *hard* do ponto de vista de segurança (*security*) em relação aos dados privados dessa mesma pessoa.

Tabela 3- Classificação dos Tipos de Sistema de Tempo Real (LAPLANTE, 1992)

Tipo	Descrição
Soft	Corresponde ao sistema cujo desempenho é degradado, porém não destruído na ocorrência de uma falha no atendimento a uma ou mais especificações de tempos de resposta.
Firm	Corresponde ao sistema cujo desempenho é degradado, porém não destruído na ocorrência de uma ou poucas falhas no atendimento às especificações de tempos de resposta. Caso o número de falhas exceda um limite crítico, o sistema pode ser conduzido a um estado para o qual a disfunção pode ter consequências catastróficas.
Hard	Corresponde ao sistema para o qual uma falha no atendimento a uma ou mais especificações de tempos de resposta pode conduzir o sistema a uma disfunção cujas consequências são, normalmente, catastróficas.

4.2. Restrições Temporais Aplicadas ao SOA

Como mencionado nas seções anteriores, para sistemas baseados na Arquitetura Orientada a Serviço, o acesso às operações disponíveis em cada serviço ocorre por meio de troca de mensagens, normalmente utilizando redes de comunicação. A Arquitetura Orientada a Serviço permite, então, o desenvolvimento de sistemas distribuídos, nos quais os componentes de hardware e software, localizados em computadores interligados por uma rede, comunicam-se entre si e coordenam suas ações somente por meio de troca de mensagens (COULOURIS, DOLLIMORE e KINDBERG, 2007).

Nesse tipo de arquitetura, em que a troca de mensagens utiliza uma rede de comunicação, existe a possibilidade de ocorrência de problemas como: grande latência e baixa confiabilidade da camada de transporte, problemas decorrentes em um cenário em que há uma falha parcial, dificultando a execução de transações consistentes e, ainda, problemas de compatibilidade decorrentes da introdução de um novo participante com uma versão desatualizada do software em questão (W3C, 2004).

É possível obter transações consistentes por meio da utilização do gerenciamento de transações, permitindo que transações efetuadas entre os diversos agentes participantes do sistema (clientes e provedores, basicamente) tenham um alto grau de confiabilidade sobre a sua execução. No caso de uma eventual falha, toda a transação seria cancelada, não restando partes de transações confirmadas e partes não confirmadas (considera-se, nesse caso, o aumento da integridade das

transações, mas não o provável aumento do tempo de transação, decorrente da necessidade de confirmação de sucesso na operação por todos os participantes).

Mesmo utilizando o gerenciamento de transações a fim de se estabelecer transações íntegras entre os seus participantes, nem sempre os métodos para sua implementação são determinados pelas arquiteturas. Por exemplo, no caso dos serviços *Web*, não são fornecidas sugestões de como implementar controle transacional de forma confiável (W3C, 2004), porém esse tipo de controle pode ser implementado no nível da aplicação.

Ainda que seja implementado o gerenciamento de transações para garantia de integridade das transações efetuadas, problemas como o não determinismo no tempo de tráfego dos dados e falta de garantia na entrega das mensagens, por exemplo, também devem ser considerados na aplicação desse tipo de arquitetura.

Um sistema de mensagens confiável, na prática, requer a verificação de que mensagens enviadas tenham sido recebidas pelo destinatário esperado e de forma consistente (sem alteração). No pior caso, o agente que tenha transmitido a mensagem (e não tenha recebido a confirmação) pode reenviar essa mensagem.

Um protocolo confiável para entrega de mensagens (WS-Reliability 1.1) foi oficializado como padrão pelo consórcio OASIS em 2005 (OASIS, 2004). Nesse padrão, são definidas, por exemplo, modificações no cabeçalho do protocolo SOAP, no qual são adicionados elementos, como os elementos *Request* e *Response*. Esses elementos possuem campos para armazenamento de dados referentes ao número de identificação da mensagem (campo *MessageId*), indicação de solicitação de reconhecimento de recebimento (campo *AckRequested*), indicação para eliminação de mensagens duplicadas (campo *DuplicateElimination*), entre outros. Vale ressaltar que esse protocolo tem por objetivo garantir a entrega de mensagens entre quem solicita e quem fornece o serviço, porém não é estabelecido o gerenciamento de transações, isto é, não são feitas interpretações das mensagens para verificação de sua consistência (OASIS, 2004).

A troca de mensagens por meio de redes de comunicação aumenta o tempo total de resposta de uma solicitação, visto que esse passa a depender não somente do tempo de processamento da transação. Além disso, normalmente não se pode garantir o determinismo da latência na transmissão de pacotes de dados, o que implica, por exemplo, que a Arquitetura Orientada a Serviço não seria uma boa escolha para desenvolvimento de sistemas com restrições temporais. Porém, o SOA

representa um desafio para sistemas que se aproximam de tempo real em que a latência não seja um requisito crítico para segurança, mas, por exemplo, seja um requisito crítico para o negócio (atendimento a metas do negócio, por exemplo) (O'BRIEN, MERSON e BASS, 2005).

Essas considerações não se aplicam somente a redes com topologia aberta como a Internet, mas também a redes locais, por exemplo. No desenvolvimento e implantação de redes locais, o padrão Ethernet (IEEE 802.3) tem sido largamente utilizado.

No padrão Ethernet (camada de Enlace de dados), a subcamada MAC (*Media Access Control*), para controle de acesso ao meio físico, utiliza o método CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*) para organização do fluxo de dados. Os termos camada e subcamada referem-se às camadas de protocolo (Aplicação, Apresentação, Sessão, Transporte, Rede, Enlace de dados e Física) existentes no modelo OSI (*Open System Interconnection*) da ISO (*International Organization for Standardization*).

Nesse método, antes de uma estação começar a transmissão de um pacote de dados, essa verifica se o meio para transmissão está livre, isto é, se não foi detectada nenhuma transmissão corrente. Em caso afirmativo, uma transmissão é iniciada.

Um problema em potencial é a ocorrência de uma colisão, isto é, situação em que duas estações começaram a transmitir dados simultaneamente. Caso uma colisão seja detectada, a transmissão é interrompida e um intervalo de tempo aleatório deve ser aguardado para que uma nova transmissão seja iniciada. No método CSMA/CD, o hardware de rede deve verificar, permanentemente, se o meio de transmissão está livre, na tentativa de minimizar a ocorrência de colisões (PRÜTER, MORITZ, et al., 2008).

A introdução de intervalos aleatórios para novas tentativas de transmissão reduz a probabilidade de ocorrência de colisões entre pacotes de dados transmitidos por estações distintas. Embora essa probabilidade seja reduzida, não há garantia da entrega dos dados (visto que o número de colisões pode ser muito elevado), nem determinismo no tempo de entrega (tempo máximo que um pacote leva para trafegar entre uma estação e outra).

No trabalho de PRÜTER, MORITZ, et al. (2008), há uma proposta para utilização de redes locais padrão Ethernet em sistemas de tempo real tipo *hard*, a qual sugere a

alteração do software de controle do hardware de acesso à rede (*drivers* das placas de rede). Essa proposta introduz o conceito de *time slots* (ou intervalos de tempo) nas camadas mais baixas do protocolo de rede, fazendo com que uma transmissão somente seja iniciada por uma estação caso haja um intervalo disponível. Caso nenhum intervalo esteja disponível, o cartão aguarda a próxima oportunidade.

Busca-se, dessa forma, o determinismo na entrega dos dados, porém, para que restrições temporais possam ser atendidas, a duração de cada intervalo de tempo deve ser muito menor que o período de tempo válido (restrição temporal) para execução da transferência de dados.

Além do não determinismo característico do padrão Ethernet, nas camadas superiores do protocolo de comunicação, protocolo TCP, por exemplo, também existem fontes de não determinismo.

O protocolo normalmente utilizado para troca de mensagens entre clientes e servidores em uma arquitetura baseada em serviços *Web* (*Web* Services) corresponde ao HTTP (*Hypertext Transfer Protocol*), o qual utiliza o protocolo TCP (*Transmission Control Protocol*) como protocolo de transporte.

O protocolo TCP é um protocolo orientado a conexão (uma conexão deve ser estabelecida entre os dois sistemas que efetuarão a transmissão e o recebimento dos dados). Em sua utilização são abstraídos detalhes de funcionamento como: organização de pacotes recebidos, retransmissão de pacotes de dados perdidos, fluxo de controle e descarte de pacotes duplicados, entre outros.

Dessa forma, mensagens transmitidas entre duas aplicações são divididas em pacotes menores e transmitidas via protocolo IP (*Internet Protocol*). É de responsabilidade do protocolo TCP, montar e organizar novamente os pacotes de dados recebidos, fornecendo a mensagem completa ao destinatário.

No caso de pacotes perdidos, o TCP solicita sua retransmissão, fazendo com que a probabilidade de entrega dos pacotes seja grande, porém pequeno o determinismo referente a restrições temporais.

Os problemas até aqui citados não cobrem totalmente as fontes de não determinismo em sistemas computacionais. O estabelecimento de restrições temporais na execução de tarefas depende do determinismo no sistema operacional que controla essas tarefas. Caso o sistema operacional não isole corretamente os processos que gerencia e não determine (faça cumprir) as restrições temporais

corretamente, não se pode garantir que os sistemas sob sua responsabilidade cumprirão suas restrições temporais.

Existem fatores associados ao tempo de processamento que não dependem exclusivamente da velocidade do hardware utilizado em um sistema. O sistema operacional deve gerenciar a concorrência por recursos entre processos (processador, memória e sistema de discos), agendamento de execução de tarefas, mudanças de contexto (modo supervisor e usuário), gerenciamento de interrupções (garantindo que não haja inversão de prioridades), entre outras atribuições.

Atrasos não previsíveis ou dedicação ilimitada de recursos no processamento de determinadas rotinas podem ainda ser resultado da utilização de métodos de sincronização sem a devida avaliação de fatores como: tempos de processamento, sobrecarga do sistema operacional, acessos diretos à memória, processadores de uso geral, alocação dinâmica para armazenamento de dados, *garbage collection*, entre outras características (THIELE e WILHELM, 2004).

No projeto de processadores de uso geral, por exemplo, são utilizados caches, pipelines e execução especulativa (desvios especulativos, por exemplo) para otimização de desempenho, a qual considera o tempo médio das execuções de aplicações. Porém, em aplicações com restrições temporais, mais importante que o desempenho no caso médio é o determinismo no tempo de execução das rotinas. Com as técnicas de projeto de processadores mencionadas, o comportamento temporal na execução de instrução não pode ser determinado localmente, pois depende do histórico de execução.

Esses fatores ainda são agravados pelos métodos de projeto e ferramentas utilizadas no desenvolvimento de sistemas, como compiladores, ferramentas de validação e simulação, cada vez mais baseadas em componentes COTS (*Commercial-off-the-Shelf*).

Em uma Arquitetura Orientada a Serviço, os efeitos desses problemas tendem a ser maiores, visto que o baixo acoplamento entre os componentes do software permite que plataformas diferentes sejam utilizadas para execução de serviços distintos, aumentando ainda mais a complexidade do sistema como um todo. Ressalta-se, porém, que este trabalho apresenta propostas, no capítulo 5, para mitigação de efeitos decorrentes da eventual manifestação desses tipos de problemas.

Assim sendo, a execução de testes de forma que sejam cobertas todas as combinações de entradas no sistema é praticamente inviável. Pelos mesmos motivos, se torna inviável a utilização de métodos formais para sua especificação.

4.3. Considerações Finais deste Capítulo

O capítulo 4 apresentou conceitos sobre restrições temporais e fontes de não determinismo em sistemas computacionais. Algumas dessas fontes de não determinismo têm grande impacto em sistemas baseados na Arquitetura Orientada a Serviço, como, por exemplo, a latência na transmissão de mensagens em redes de comunicação e nos respectivos protocolos de comunicação utilizados.

O capítulo 5, a seguir, apresenta a arquitetura proposta neste trabalho, considerando os conceitos estudados sobre a Arquitetura Orientada a Serviço, teoria das filas, escalonamento e restrições temporais, apresentados nos capítulos 2, 3 e 4.

5. ESPECIFICAÇÃO DA ARQUITETURA

Este capítulo apresenta a arquitetura baseada no padrão arquitetural SOA a ser aplicada no desenvolvimento de sistemas com restrições temporais. São descritos os requisitos básicos dessa arquitetura, bem como restrições ou condições de contorno que se aplicam.

5.1. Requisitos

Neste trabalho é proposta uma Arquitetura Orientada a Serviço que possibilite o desenvolvimento de aplicações com restrições temporais. De acordo com a classificação apresentada para sistemas com restrições temporais no capítulo 4, são considerados, para a arquitetura proposta, os sistemas de tempo real do tipo *soft*, cujo desempenho é degradado, porém não destruído na ocorrência de uma falha no atendimento a uma ou mais especificações de tempos de resposta (LAPLANTE, 1992). Não são considerados, nessa arquitetura, sistemas de tempo real em que a latência seja um requisito crítico para segurança, mas, que seja um requisito crítico para o negócio (atendimento a metas do negócio, por exemplo) (O'BRIEN, MERSON e BASS, 2005).

Assim sendo, a arquitetura proposta apresenta os seguintes requisitos:

1. A arquitetura proposta deve ser derivada do padrão arquitetural SOA;
2. A arquitetura proposta deve ser baseada em softwares COTS (*Commercial Off The Shelf*);
3. A arquitetura proposta deve ser compatível com padrões abertos, de forma que não sejam necessárias adaptações em protocolos de comunicação ou linguagens específicas para descrições de interface;
4. A arquitetura proposta deve permitir o desenvolvimento de sistemas com restrições temporais em relação ao tempo de resposta de uma requisição de serviço por um cliente;
5. A arquitetura proposta não se destina ao desenvolvimento de sistemas críticos de segurança, isto é, embora a arquitetura deva permitir o desenvolvimento de sistemas que atendam a restrições temporais, o não atendimento a essas restrições não deve implicar em danos a pessoas ou ao patrimônio.

5.2. Arquitetura Proposta

A utilização dos serviços *Web* para desenvolvimento da arquitetura proposta satisfaz três dos cinco requisitos inicialmente propostos: são uma arquitetura derivada do padrão arquitetural SOA (requisito 1), possuem implementações em softwares COTS (requisito 2) e são baseados em padrões abertos (requisito 3).

Como o requisito 4 (a arquitetura proposta deve permitir o desenvolvimento de sistemas com restrições temporais) foi dividido em dois itens (4a e 4b), são utilizadas abordagens distintas para seu atendimento.

Em relação ao requisito 4a (o sistema deve garantir que um pacote de dados chegue ao receptor dentro de um intervalo de tempo pré-definido), o não determinismo da latência nas redes de comunicação não permite que esse requisito seja atendido. O que se propõe, então, é uma restrição no campo de aplicação da arquitetura, de forma que seja utilizada em redes de comunicação que possam ser gerenciadas e controladas (topologia conhecida).

Considera-se, para esse tipo de rede, que há compatibilidade de hardware e software entre todos os nós que a compõe e que alterações na taxa de utilização da rede possam ser monitoradas e gerenciadas, de forma que seja possível observar a latência na transmissão de pacotes e atuar para manter seu valor dentro de limites pré-estabelecidos. Por exemplo, podem ser considerados, nesse caso, os resultados obtidos nos estudos de (MARAU, ALMEIDA e PEDREIRAS, 2006) e (SANTOS, VIERIA, et al., 2010).

O sistema desenvolvido a partir da arquitetura proposta nesta tese poderia, então, operar em redes locais em que existe a possibilidade de gerenciamento (monitoramento e atuação sobre a latência de rede) e são mantidas as características dos protocolos de comunicação referentes ao padrão WS-I (como TCP/IP e HTTP, por exemplo).

Vale ressaltar que esse tipo de gerenciamento não garante o determinismo nas transações, porém permite verificar a ocorrência de períodos em que o tráfego de rede compromete o tempo de resposta das transações e dimensionar recursos de rede para mitigar seus efeitos.

Para o requisito 4b (deve ser garantida disponibilidade de 100%), aplica-se a redundância de serviços para que a disponibilidade assintótica se aproxime desse valor. Como exposto, a redundância, nesse caso não tem como objetivo somente

aumentar o nível de disponibilidade dos sistemas, mas também aplicar os conceitos da teoria das filas (múltiplos servidores) e escalonamento, conforme apresentado na seção 5.4.

A proposta para a arquitetura apresentada nesta tese considera uma combinação das propostas de HARCHOL-BALTER, SCHROEDER, et al. (2003) e TSAI, LEE, et al. (2006) que utiliza o algoritmo SPRT (utilizado para determinação das filas de prioridade) e, além das filas de prioridades do algoritmo SRPT, adiciona-se um segundo nível de prioridade estabelecido para as requisições pendentes em uma determinada fila (são priorizadas as solicitações com menor tempo restante para que seja atingida a restrição temporal). O algoritmo SPRT é utilizado para escalonamento de solicitações ao servidor, com o conjunto de filas de prioridade sendo criado a partir de faixas pré-definidas de tempos de resposta e não a partir de faixas de tamanho dos arquivos transferidos, como apresentado no estudo original.

Para a inclusão de uma requisição em uma das filas é utilizada a restrição temporal estabelecida no instante de uma requisição (a partir do tempo máximo de resposta estabelecido, determina-se em qual fila a requisição deve ser armazenada). O atendimento prioritário de requisições com menor tempo de execução em relação àquelas com maior tempo de execução, resultaria, então, em um tempo médio de resposta menor que no caso em que várias transações curtas aguardam a execução de uma transação longa.

Além do tempo de resposta, o tempo restante para conclusão da solicitação também é considerado, priorizando as transações com menor tempo restante para conclusão (*deadline*) dentro de uma determinada fila.

No caso da arquitetura proposta, a métrica principal de qualidade de serviço é o tempo máximo de resposta. Como as restrições temporais fazem parte da requisição de serviço, deve ser considerado estado falho do sistema (ou indisponível) quando o tempo de execução do serviço viola as fronteiras de tempo impostas pelas restrições especificadas.

O termo Qualidade de Serviço (QoS – *Quality of Service*), está normalmente associado à área de redes de computadores e tem como objetivo estabelecer a probabilidade de que um determinado sistema possa atingir as especificações de desempenho determinadas em contratos de fornecimento (SLA – *Service Level Agreement*). Um exemplo de parâmetro observado seria o número de pacotes de dados transmitidos por unidade de tempo.

Para a descoberta de serviços, a arquitetura proposta nesta tese utiliza uma combinação das propostas de (TSAI, LEE, et al., 2006) e (O'BRIEN, MERSON e BASS, 2005), em que o repositório UDDI é substituído por uma lista de serviços previamente verificados, porém modificável durante a execução do sistema, de acordo com o determinismo no tempo das transações executadas.

A modificação nessa lista ocorre caso um serviço não atenda os requisitos temporais especificados no seu cadastro (valores pré-estabelecidos de execução). Essa restrição atua, então, como um parâmetro de qualidade dos serviços (QoS – *Quality of Service*) prestados.

Vale ressaltar que, embora haja a possibilidade de que essa lista seja modificada dinamicamente, os valores máximos de tempo busca podem ser previamente estabelecidos, uma vez que o tamanho da lista é controlado (considera-se o tamanho inicial como sendo o máximo). Essa lista não está descrita e incorporada ao código fonte do sistema, mas armazenada em um arquivo local modificável, caso haja alguma modificação na distribuição de serviços disponíveis (para efeitos de manutenção do sistema, por exemplo).

O requisito 5 (a arquitetura proposta não se destina ao desenvolvimento de sistemas críticos de segurança) corresponde a uma restrição no campo de aplicação e é atendido automaticamente quando respeitada essa restrição na utilização da arquitetura proposta.

A troca de mensagens na arquitetura proposta nesta tese considera a existência de um serviço responsável pelo recebimento de requisições dos clientes. Esse serviço é identificado, nessa arquitetura, como serviço *Web* Distribuidor e o recebimento de requisições e retorno de respostas ocorre da seguinte maneira:

1) O requisitante solicita, ao serviço *Web* Distribuidor, uma requisição de serviço, especificando o tempo máximo para sua execução;
2) Uma resposta para o requisitante é imediatamente fornecida e seu conteúdo consiste no sucesso ou falha do recebimento da requisição;
3) Posteriormente, o cliente consulta o serviço *Web* Distribuidor para saber se a resposta para a requisição feita está disponível para consulta.

A Figura 14 apresenta uma comparação entre a arquitetura original dos serviços *Web* e a arquitetura proposta nesta tese, com a substituição do registro UDDI por um registro local e com a introdução do serviço *Web* Distribuidor, para escalonamento das requisições.

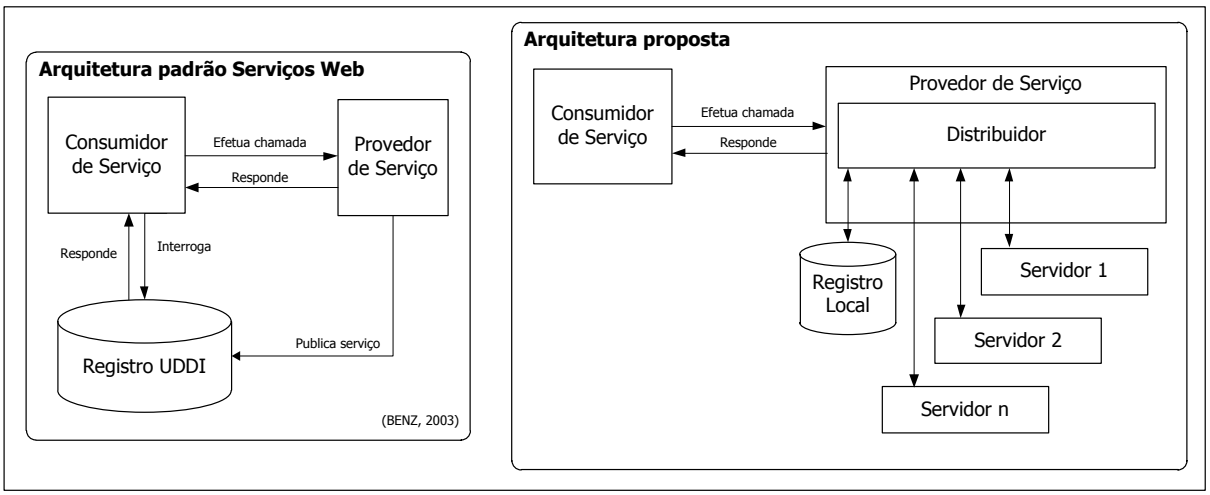


Figura 14 – Comparação entre a arquitetura dos serviços *Web* e a arquitetura proposta.

Embora a arquitetura proposta seja uma arquitetura diferente da arquitetura especificada pelo consórcio WS-I, a infraestrutura de comunicação (troca de mensagens) entre serviços não é alterada, isto é, são utilizados os protocolos de comunicação e empacotamento especificados nesse padrão (requisito 3), como, por exemplo, o protocolo HTTP sobre o protocolo TCP/IP para transmissão de dados, o protocolo SOAP para empacotamento de mensagens e o padrão WSDL para descrição das interfaces dos serviços, conforme indicado na Figura 15.

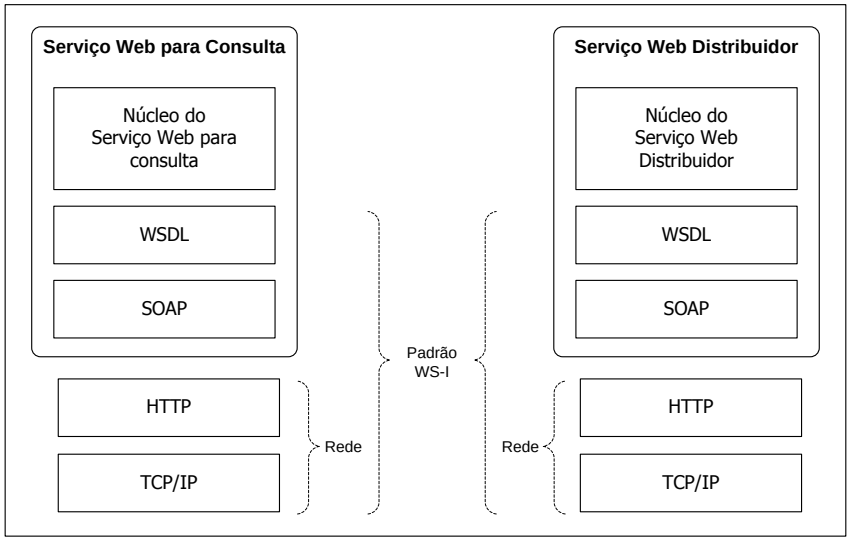


Figura 15 – Serviços *Web* desenvolvidos x Padrão WS-I.

A partir dos requisitos estabelecidos e das soluções propostas, dentro das condições de contorno especificadas, pode-se fazer o mapeamento das soluções propostas para os requisitos especificados, conforme apresentado na Tabela 4.

Tabela 4- Mapeamento de requisitos na arquitetura proposta

Requisito	Proposta
Arquitetura derivada do padrão SOA	Utilização de uma arquitetura derivada dos serviços *Web*.
Arquitetura baseada em software COTS	Utilização de uma arquitetura derivada dos serviços *Web*.
Arquitetura compatível com padrões abertos	Utilização de uma arquitetura derivada dos serviços *Web*.
Arquitetura utilizada no desenvolvimento de sistemas com restrições temporais	• Redundância de software (serviços); • Operação em redes de comunicação controladas; • Escalonamento por níveis de prioridade; • Registro local de serviços.
Arquitetura incompatível com desenvolvimento de sistemas críticos de segurança	Arquitetura utilizada no desenvolvimento de sistemas com restrições temporais, classificados como tempo real *soft* (LAPLANTE, 1992).

5.3. Detalhamento do Serviço *Web* Distribuidor

O serviço *Web* Distribuidor atua entre os computadores que efetuam requisições de execução de serviços (clientes) e os computadores que processam as requisições efetuadas (servidores), conforme apresentado na Figura 16.

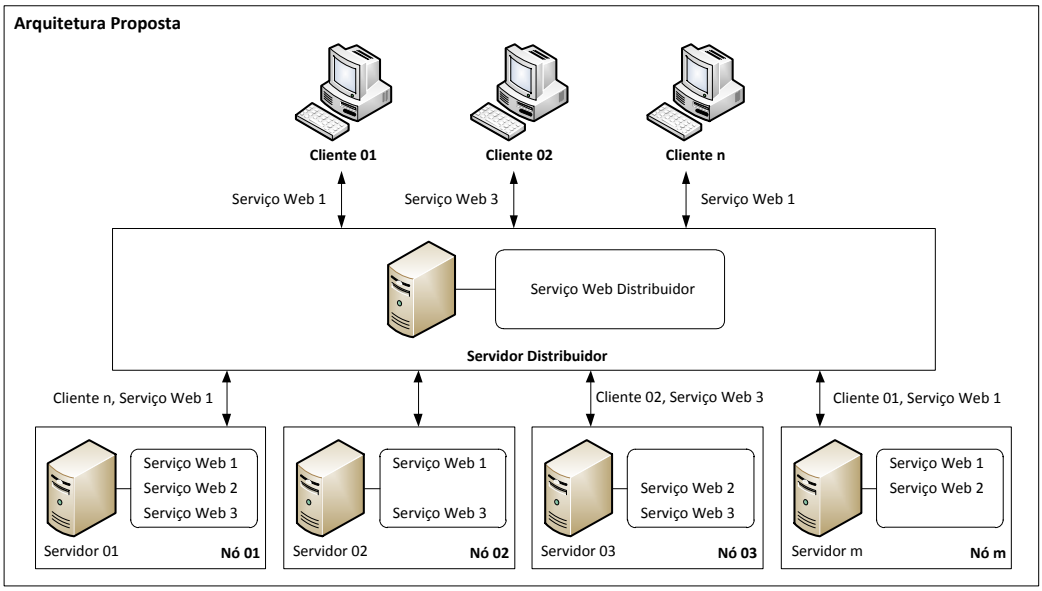


Figura 16 – Clientes versus Serviço *Web* Distribuidor versus Servidores

Pela Figura 16, pode-se observar que a distribuição das requisições para diferentes servidores deve considerar que diferentes servidores podem executar um mesmo tipo de serviço e que cada servidor pode executar mais de um tipo de serviço.

O serviço *Web* Distribuidor é, então, responsável pela execução das seguintes tarefas:

- **Gerenciamento das requisições de serviços**.

  As requisições de serviço são recebidas de forma que seja minimizado o tempo de espera para a sua resposta.

  Esse baixo tempo de espera na resposta permite que o sistema requisitante possa executar os demais processos enquanto a resposta esperada não é fornecida ou simplesmente bloquear outros processos até que seja recebida a resposta. Como o tempo esperado de execução é fornecido na requisição, o sistema requisitante pode determinar o instante de tempo em que deve fazer a consulta pelo valor da resposta.

- **Escalonamento e endereçamento de mensagens**.

  O escalonamento deve considerar as restrições temporais impostas na requisição de um serviço e servidores disponíveis para execução da requisição efetuada (visto que mais de um servidor pode fornecer o mesmo tipo de serviço). Considera-se que os serviços em questão trabalham de forma independente, isto é, nenhum serviço depende de outro para processamento de uma requisição e que o sucesso nesse processamento corresponde ao envio da requisição e recebimento da respectiva resposta dentro das restrições temporais especificadas.

  A política de escalonamento, conforme descrito nesta seção, prioriza a execução de requisições com menor tempo de execução e com menos tempo restante para conclusão (especificado na chamada).

Para que seja possível o gerenciamento das requisições e respostas, bem como o escalonamento de execuções, o serviço *Web* Distribuidor é dividido em dois processos principais: Escalonador e Gerenciador das Filas de Requisições e Respostas, conforme apresentado na Figura 17.

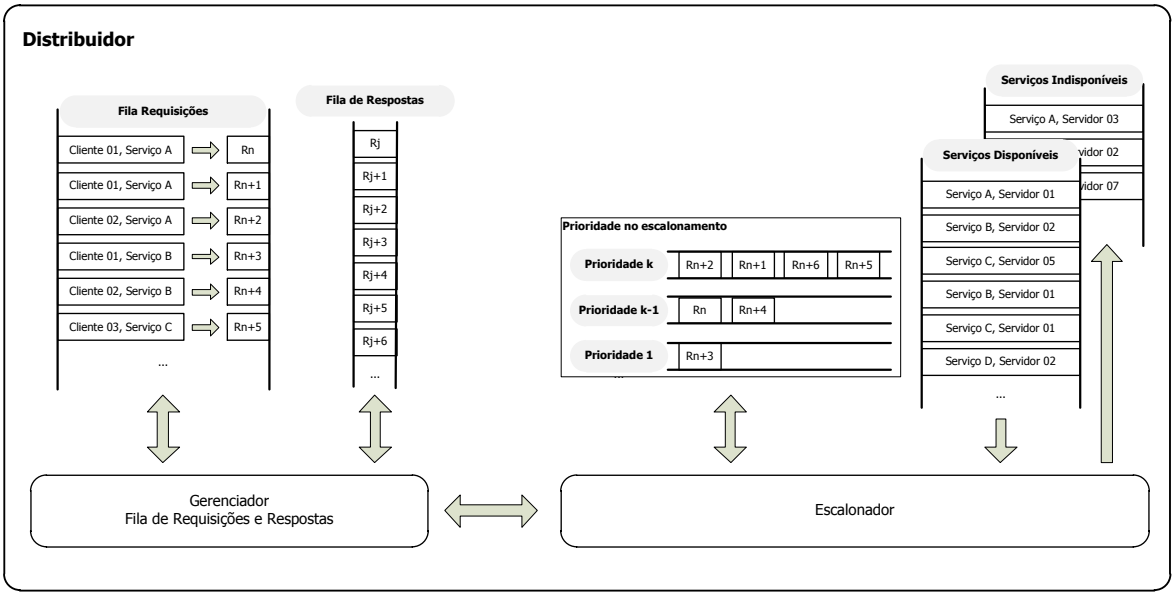


Figura 17 – Estrutura interna do serviço *Web* Distribuidor

5.3.1. Gerenciador das Filas de Requisições e Respostas

Esse processo atua como interface do serviço *Web* Distribuidor com os clientes, tanto do ponto de entrada das requisições de execução de serviços, bem como ponto de saída para fornecimento dos resultados das requisições.

As requisições são recebidas e armazenadas em uma fila para que seja possível fazer o escalonamento. As respostas obtidas para cada solicitação também são enfileiradas, de forma que possam ser consultadas posteriormente pelos clientes. As duas filas principais são:

- **Fila de requisições**: armazena as requisições de execução de serviços. Cada requisição é univocamente identificada e armazena a identificação do cliente que fez a requisição de execução do serviço, tipo de serviço solicitado, sintaxe de chamada do serviço e restrições temporais para sua execução.
- **Fila de respostas**: armazena os resultados das execuções das requisições dos clientes, despachadas pelo serviço *Web* Distribuidor para os servidores que executam as requisições solicitadas.

  Em caso de falha na execução, uma resposta indicando o status da transação também é armazenada. Como exemplos de erros na execução, tem-se: serviço não identificado, serviço indisponível, restrição temporal abaixo dos valores mínimos disponíveis, entre outros.

5.3.2. Escalonador

O escalonador é responsável pela distribuição de requisições para os diversos servidores disponíveis, de acordo com a prioridade estabelecida.

A lista de servidores disponíveis, bem como dos serviços disponíveis em cada servidor, se encontra armazenada em um arquivo local e é carregada em memória na iniciação do serviço (Lista de **Serviços Disponíveis**, conforme apresentado na Figura 17). Essa lista pode ser alterada, isto é, reduzida, de acordo com a disponibilidade dos serviços (servidores), sendo os serviços não mais utilizados colocados em uma lista de **Serviços Indisponíveis**, conforme apresentado na Figura 17.

O estabelecimento de prioridades é baseado nas restrições temporais determinadas no instante em que as requisições são efetuadas. É o cliente que fornece a restrição temporal para uma requisição, ou seja, o escalonador é informado, pelo cliente, qual o instante de tempo máximo em que a resposta para a requisição solicitada deve estar disponível.

O algoritmo de escalonamento opera da seguinte forma:

1) Uma requisição recebida é classificada de acordo com o tempo de resposta esperado, sendo armazenada em uma fila de execução com prioridade inversamente proporcional à sua restrição temporal. Dessa forma, requisições com maior tempo para término da execução são classificadas como menos prioritárias que requisições com menos tempo para término da execução;
2) Ao ser selecionada a fila na qual será inserida, a requisição é classificada, em relação às restrições existentes nessa fila, de acordo com o tempo restante para que o prazo limite de sua restrição temporal seja atingido, isto é, requisições com menos tempo para expirar têm prioridade no atendimento.

O estabelecimento de filas de prioridade baseia-se no trabalho proposto por HARCHOL-BALTER, SCHROEDER, et al. (2003), no qual foram estabelecidas filas de prioridade de acordo com o tamanho dos arquivos requisitados pelos clientes (perfil observado experimentalmente).

Visto que, neste estudo, o parâmetro principal a ser considerado é o tempo de resposta de uma requisição, o estabelecimento de classes de prioridade deve considerar, então, o tempo máximo para execução de uma requisição ao invés do

tamanho em bytes de um arquivo solicitado. Um exemplo de distribuição de prioridades está indicado na Tabela 5.

Tabela 5- Filas com prioridades baseadas em restrição temporal.

Prioridade	Restrição temporal (ms)
1	≤ 1.000ms
2	1.000ms – 2.000ms
3	2.000ms – 5.000ms
4	5.000ms – 20.000ms
5	20.000ms – 50.000ms
6	> 50.000ms

5.4. Trabalhos Correlacionados

Os Serviços *Web*, que correspondem a uma arquitetura derivada do SOA padronizada pelo consórcio WS-I (*Web Services Interoperability Organization*), são utilizados em larga escala e apresentam um padrão relativamente estável. Correspondem a uma das abordagens possíveis para aplicação do SOA no desenvolvimento de sistemas e se tornaram a principal tecnologia utilizada na sua implementação, sendo um dos principais motivos nessa larga utilização, o fato de serem baseados em padrões abertos e não em soluções proprietárias (LEWIS, MORRIS, et al., 2007).

Antes da introdução dos serviços *Web* não existia a padronização de um protocolo de comunicação que pudesse ser utilizado efetivamente por sistemas baseados em diferentes tecnologias, de forma a operar em escala mundial (O'BRIEN, MERSON e BASS, 2005). Vale ressaltar que a maturidade e consequente aumento na sua utilização foram atingidos após problemas nas implementações iniciais, como: baixo desempenho, pouca interoperabilidade e esquemas complexos de gerenciamento (CUOMO, 2005).

Dessa forma, a utilização dos serviços *Web* no desenvolvimento da arquitetura proposta atende a vários requisitos estabelecidos, pois são uma arquitetura derivada do padrão arquitetural SOA (requisito 1), possuem implementações em softwares COTS (requisito 2) e são baseados em padrões abertos (requisito 3).

O quarto requisito da arquitetura proposta refere-se ao determinismo no tempo de resposta de uma requisição de serviço. Conforme verificado nos trabalhos de O'Brien (O'BRIEN, MERSON e BASS, 2005), Tsai (TSAI, LEE, et al., 2006) e Prüter (PRÜTER, MORITZ, et al., 2008), o não determinismo na latência da rede de comunicações compromete esse requisito, que é a proposta central deste trabalho, isto é, apresentar uma arquitetura baseada no padrão SOA que possa ser utilizada em sistemas com restrições temporais.

Segundo Prüter (PRÜTER, MORITZ, et al., 2008), com o objetivo de se proporcionar características de tempo real para serviços *Web*, dois requisitos de tempo real devem ser satisfeitos: o sistema deve garantir que um pacote de dados chegue ao receptor dentro de um intervalo de tempo pré-definido e uma disponibilidade próxima a 100% deve ser garantida.

Como esses dois requisitos não fazem parte da especificação inicialmente proposta e para clareza da proposta aqui descrita, esses dois requisitos adicionais, conforme propostos por PRÜTER, MORITZ, et al. (2008), são incorporados aos requisitos inicialmente propostos, porém como subitens do requisito 4:

4a. O sistema deve garantir que um pacote de dados chegue ao receptor dentro de um intervalo de tempo pré-definido;

4b. Deve ser garantida disponibilidade de 100%.

Visando ao atendimento do primeiro requisito para comunicação em tempo real (requisito 4a) existem soluções propostas, como o protocolo RTNet, por exemplo (KISZKA e WAGNER, 2005). Essa solução foi inicialmente desenvolvida para transmissões efetuadas sobre o protocolo UDP e apresenta compatibilidade com uma pequena quantidade de adaptadores de rede (RTNET, 2010).

O protocolo RTNet introduz *time slots* (ou intervalos de tempo) nas camadas mais baixas do protocolo de rede, com o objetivo de melhorar o nível de determinismo. Porém, modificam a especificação dos protocolos de transmissão originais, reduzindo a probabilidade de interoperabilidade.

Esse protocolo também não está em conformidade com o padrão TCP, fato que inviabiliza a sua utilização, uma vez que o segundo requisito estabelecido para a arquitetura proposta determina que devam ser utilizados softwares COTS, que atendam a padrões estabelecidos.

Vale ressaltar que a própria especificação do protocolo TCP indica que, no caso de erro na transmissão de dados, o pacote deve ser retransmitido (caso não seja

recebido um reconhecimento enviado pelo receptor, dentro de um intervalo de tempo), o que não pode ser considerado determinístico (PRÜTER, MORITZ, et al., 2008).

Uma das vantagens que favorece a utilização dos serviços *Web* é a possibilidade de utilização sobre a Internet, situação em que a confiabilidade e a velocidade de transmissão de pacotes de dados não pode ser garantida (W3C, 2004). Essa característica, porém, não consiste em uma vantagem para aplicação em sistemas com restrições temporais, uma vez que, nesse caso, a predição de tempos de transmissão de pacotes de dados é muito difícil, devido à variação no tamanho das mensagens, taxa de transmissão de mensagens e roteamento de pacotes de dados, entre outros fatores.

Uma solução proposta por (MARAU, ALMEIDA e PEDREIRAS, 2006) consiste na implementação de um protocolo baseado no protocolo FTT-Ethernet (*Flexible Time-Triggered*), que permite o controle de tráfego em redes compartilhadas. O protocolo proposto FTT-SE (*Switched Ethernet*), a ser implementado em *switches* Ethernet COTS, contribui para melhorias nas características da comunicação em tempo real. Essa proposta foi revista por (SANTOS, VIERIA, et al., 2010), com a introdução de hardware externo para execução do protocolo FTT-SE, sem que seja necessária a utilização de *drivers* de rede especiais para os adaptadores de rede existentes em cada nó participante.

Protocolos FTT organizam a comunicação em intervalos fixos de tempo (*time slots*), denominados Ciclos Elementares (ECs ou *Elementary Cycles*), com uma mensagem mestre por ciclo, denominada Mensagem Gatilho (TM ou *Trigger Message*), que contém o escalonamento de tempo para o Ciclo Elementar em questão. Mensagens periódicas (denominadas síncronas) são sincronizadas com o escalonador de tráfego periódico. O protocolo também suporta tráfego não periódico, que é gerenciado em segundo plano, no intervalo de tempo restante dentro de um Ciclo Elementar, após toda a execução do tráfego periódico, conforme indicado na Figura 18. Vale ressaltar, que estudos aprofundados sobre a proposta de utilização do protocolo FTT citado não fazem parte do escopo dessa tese.

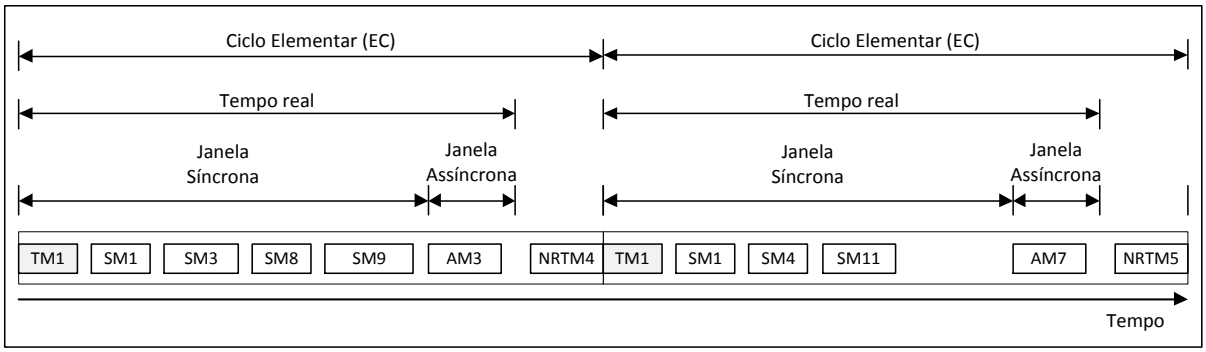


Figura 18 – Protocolo FTT (MARAU, ALMEIDA e PEDREIRAS, 2006).

O segundo requisito para comunicação em tempo real (requisito 4b) refere-se ao nível de disponibilidade do sistema. A disponibilidade de um sistema pode ser definida como a probabilidade de um sistema estar em funcionamento, de acordo com suas especificações, em um determinado instante de tempo (JOHNSON, 1989). Estando relacionada diretamente com a confiabilidade do sistema, a disponibilidade assintótica (valor da disponibilidade instantânea quando o tempo tende ao infinito) pode ser estimada em função dos respectivos parâmetros de confiabilidade (LALA, 2001):

$$Disponibilidade_{Assintótica} = \frac{MTTF}{MTTF + MTTR} \qquad \text{Equação 11}$$

Onde:

- MTTF (*Mean Time To Failure*):
  Corresponde ao tempo estimado (em média) para que o sistema esteja funcionando até a ocorrência de uma falha. O MTTF também pode ser denominado como *Uptime*;
- MTTR (*Mean Time To Repair*):
  Corresponde ao tempo médio gasto para que o sistema seja reparado após a ocorrência de uma falha. O MTTR também pode ser denominado como *Downtime*.

De acordo com a Equação 11, pode-se observar que níveis de disponibilidade próximos a 100% seriam possíveis somente se o tempo de reparo fosse igual a zero, isto é, caso não fosse feita nenhuma parada para manutenção do sistema, seja preventiva ou corretiva, o que é uma hipótese improvável. Porém, podem ser utilizadas técnicas para aumento da disponibilidade de sistemas computacionais, de forma a se atingir valores próximos a 100%, sendo as duas abordagens principais consideradas: o aumento da qualidade dos componentes utilizados na sua

construção (aumentando o MTTF) e a diminuição do tempo de reparo (diminuição do MTTR).

Entre as técnicas utilizadas para diminuição no tempo de reparo do sistema (tempo que o sistema permanece indisponível), tem-se a aplicação de redundância nos componentes do sistema, de forma que um componente adicional desempenhe as funções do componente defeituoso, enquanto esse é reparado, sem que todo o sistema tenha que permanecer indisponível (LOPES, 2005).

Uma das características da Arquitetura Orientada a Serviço é o baixo acoplamento entre os serviços que compõem o sistema. Essa característica facilita a utilização de redundância de software, uma vez que serviços idênticos podem operar simultaneamente e, na eventual falha de um desses componentes, um serviço redundante poderia substituí-lo. Porém, no caso de sistemas com restrições temporais, a avaliação de disponibilidade deve considerar também como falha (implicando na diminuição do MTTF) o não atendimento a uma solicitação dentro dos limites temporais impostos.

Conforme apresentado no capítulo 3, a teoria das filas indica que, em sistemas baseados em uma fila com múltiplos servidores, a relação tempo de resposta / tempo de serviço apresenta uma taxa de crescimento menor com o aumento da taxa de utilização do que em sistemas com múltiplas filas e um único servidor (LIU, 2009). Dessa forma, a utilização da redundância de software (múltiplos servidores / serviços), permite não somente a substituição automática de um sistema que tenha apresentado falha por um equivalente, mas também contribui para o atendimento às restrições temporais.

A teoria das filas tem demonstrado também que a utilização de políticas de escalonamento afeta de forma significativa o desempenho percebido pelo usuário, conforme apresentado por (HARCHOL-BALTER, SCHROEDER, et al., 2003), no qual foi implementada a política de escalonamento SRPT (*Shortest Remaining Processing Time*) em servidores *Web* e que obteve resultados que indicam reduções significativas nos tempos médios de resposta e latência de rede, além do fato de não ter sido verificada penalização na execução processos que demandavam maior quantidade de recursos.

Conforme apresentado na seção 3.3 deste trabalho, a proposta de (HARCHOL-BALTER, SCHROEDER, et al., 2003) para implantação do algoritmo SPRT utiliza um conjunto de filas de requisições, cada uma com uma prioridade de atendimento

diferente. Para que as requisições de uma determinada fila sejam atendidas, todas as requisições dos níveis mais prioritários devem ter sido atendidas previamente (no trabalho apresentado, o número de filas corresponde ao número de categorias criadas para faixas de valores dos tamanhos dos arquivos transmitidos).

A utilização do escalonamento também foi considerada no trabalho de TSAI, LEE, et al. (2006), em que a prioridade para atendimento a uma requisição considera, além da ordem de chegada da requisição, a possibilidade de atendimento prioritário de acordo com o nível de qualidade de serviço contratado na requisição do serviço. No trabalho proposto por TIAN, LI, et al. (2009), o escalonamento é baseado em uma função utilidade parametrizada pelo nível do consumidor (classificação de importância do consumidor), valor do serviço a ser executado (quanto maior o valor, mais importante o serviço) e o tempo de resposta médio do serviço.

Na arquitetura dos serviços *Web*, a identificação de servidores disponíveis para execução das transações solicitadas considera um serviço de descoberta UDDI (*Universal Description, Discovery and Integration*). É por meio de consultas ao servidor UDDI que um cliente pode determinar quais servidores estão disponíveis para execução do serviço selecionado.

No trabalho proposto por TSAI, LEE, et al. (2006), o mecanismo de descoberta de serviços (UDDI), é substituído por referências estáticas, aumentando o determinismo na identificação do par (serviço, servidor), em que o repositório dinâmico de busca é substituído por um repositório estático, contendo endereços de serviços previamente verificados (qualidade do determinismo no tempo de resposta, por exemplo) e prontos para uso, sem que seja necessária a execução de testes adicionais.

A quantidade de tempo adicionada ao tempo total de execução devido à chamada extra ao servidor que mantém o repositório de serviços também é considerada no trabalho de O’BRIEN, MERSON e BASS (2005), indicando como possível alternativa a utilização de endereços mantidos no código fonte. Vale ressaltar que não é o objetivo principal, nesta tese, reduzir o tempo médio de resposta a requisições efetuadas, mas aumentar o determinismo nos tempos de resposta dessas requisições.

5.5. Considerações Finais deste Capítulo

Neste capítulo foi apresentada a especificação da arquitetura proposta, incluindo seus requisitos e condições para que esses requisitos fossem satisfeitos. A arquitetura baseada no padrão arquitetural SOA corresponde uma arquitetura derivada dos serviços *Web*, com a introdução de um serviço auxiliar para escalonamento de requisições.

No próximo capítulo é apresentado o projeto do ambiente de testes utilizado para avaliação da arquitetura proposta.

6. PROJETO DO AMBIENTE DE TESTES

Este capítulo descreve o ambiente de testes criado para a avaliação da arquitetura proposta, bem como os requisitos da arquitetura a serem avaliados.

6.1. Simulação da Arquitetura Proposta

O requisito fundamental a ser avaliado na arquitetura proposta é o determinismo nos tempos de resposta das requisições efetuadas pelos clientes, isto é, busca-se verificar se a arquitetura proposta contribui para a diminuição da variância nos tempos resposta de requisições aos serviços.

Para que a arquitetura possa ser avaliada, são respeitadas as seguintes condições de contorno, conforme descrito no capítulo 3:

1) Os tempos entre chegadas de requisições apresentam distribuição Exponencial;
2) A fila do sistema está vazia ou com baixa ocupação, significando que o sistema não está sendo executado perto da saturação;
3) O sistema não está perto da saturação e sua utilização está abaixo de 70%, valor a partir do qual a razão entre o tempo de resposta (R) e o tempo de serviço (S) aumenta consideravelmente.

Deve ser verificada, então, a contribuição da arquitetura proposta para o determinismo do valor da relação entre os tempos de resposta e serviço (relação R/S), em diferentes valores de taxa de utilização do sistema. Vale ressaltar que o Tempo de Serviço (S) corresponde ao intervalo de tempo decorrido entre os inícios de tratamento de duas requisições consecutivas e o tempo de resposta (R) corresponde ao tempo total entre a chegada de uma requisição e o seu atendimento (soma do tempo de espera com o tempo de serviço).

A implementação do sistema de testes considera as seguintes características, propostas pela arquitetura:

1) Utilização de escalonamento para distribuição das requisições. O escalonamento é baseado em filas de prioridades (o tempo de execução previsto define a fila na qual será incluída uma requisição). Em uma determinada fila, o tempo restante para que seja atingida a restrição temporal define a prioridade de execução;

2) Utilização de múltiplos servidores para atendimento das requisições, pois esse tipo de configuração contribui para que a relação tempo de resposta / tempo de serviço apresente uma taxa de crescimento menor com o aumento da taxa de utilização do que em sistemas com múltiplas filas e um único servidor, conforme apresentado no capítulo 3;
3) A descoberta dos serviços é feita de forma estática, isto é, a lista dos endereços de servidores / serviços é obtida na iniciação do sistema, a partir da leitura de um arquivo contendo esses dados;
4) Embora tenha sido proposta na arquitetura uma lista para armazenamento de serviços indisponíveis, na implementação essa lista não é utilizada, uma vez que se pretende observar a frequência desse tipo de ocorrência. Na arquitetura proposta, um serviço é cadastrado como indisponível caso não atenda às restrições temporais impostas, sendo removido da lista de serviços disponíveis;
5) Como apresentado no capítulo 5, uma das premissas para a aplicação da arquitetura proposta é sua utilização em redes locais gerenciáveis, isto é, de redes em que seja possível monitorar e atuar sobre as causas determinantes para a latência na rede (como utilização de *switches* e gerenciamento de banda de transmissão disponível por usuários, por exemplo). No desenvolvimento do ambiente de testes é utilizada uma rede local dedicada, para a qual se considera a latência como sendo constante, de forma a simplificar a medição dos tempos envolvidos.

6.2. Componentes do Sistema de Testes

Na Figura 19 podem ser observados os componentes que foram desenvolvidos para construção do sistema de testes, apresentados a seguir:

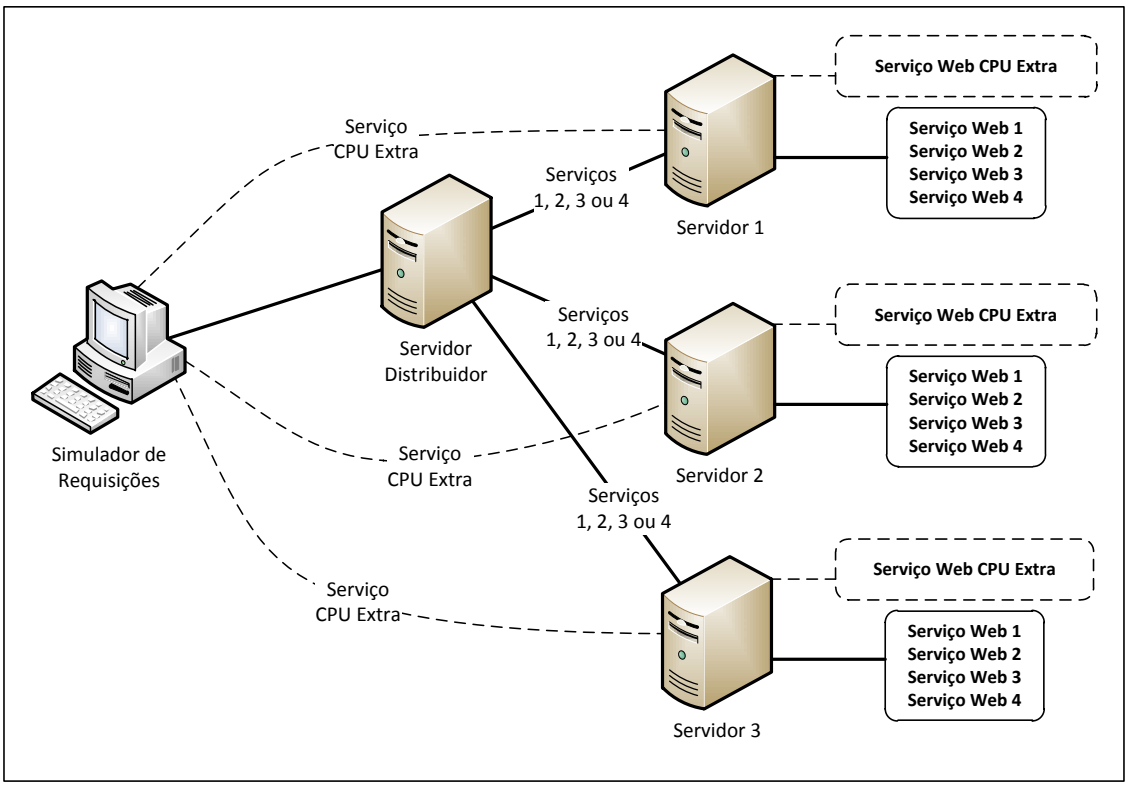


Figura 19 – Componentes do sistema de testes

1) **Simulador de Requisições**: programa que gera as requisições para o servidor que executa o serviço *Web* Distribuidor. As requisições geradas apresentam distribuição exponencial do tempo entre requisições;
2) **Serviço *Web* Distribuidor**: serviço responsável pelo recebimento de requisições, escalonamento dessas requisições para os diversos servidores que executam os serviços solicitados (servidores 1, 2 e 3) e fornecimento de respostas para as requisições efetuadas;
3) **Serviços *Web* 1, 2, 3 e 4**: serviços *Web* que são executados nos servidores mediante o recebimento de requisições originadas pelo Simulador de Requisições e escalonadas pelo serviço *Web* Distribuidor;

   Os serviços *Web* desenvolvidos para simulação executam instruções que permitem impor uma determinada taxa de utilização nos servidores em que são executados. São considerados, então, como caixas pretas que recebem uma requisição, processam os dados necessários e retornam uma resposta.

4) **Serviço *Web* CPU Extra**: serviço chamado diretamente pelo simulador, que tem por objetivo impor uma determinada quantidade de processamento concorrente ao processamento das requisições sendo atendidas pelos servidores 1, 2 e 3.

   Por meio desse serviço, é possível impor diferentes níveis de processamento nos servidores, de forma a se mapear os diferentes tempos de resposta do sistema de acordo com diferentes níveis de solicitação.

   Sua chamada é feita diretamente pelo Simulador de Requisições, pois não se trata de um serviço oferecido pelo serviço *Web* Distribuidor, uma vez que não faz parte da arquitetura proposta.

   Um nível de processamento extra é solicitado pelo simulador ao serviço *Web* CPU Extra, por meio de passagem de parâmetro na chamada, que indica o nível de processamento extra ao qual será submetido o computador que executa o serviço.

O desenvolvimento do cliente para execução da simulação, dos serviços *Web* para consulta e imposição de consumo extra de processamento e do serviço *Web* Distribuidor é baseado no paradigma orientado a objetos, cujas classes principais são detalhadas nas seções seguintes.

A implementação de todo o código fonte é baseado no ambiente de desenvolvimento Borland RAD Studio 2007 (C++ Builder). Os serviços desenvolvidos são gerenciados por servidores configurados com sistema operacional Windows XP 32 bits Service Pack 3 (WinXP32SP3) e servidor *Web Internet Information Services* 5.1 (IIS 5.1).

A escolha desse sistema operacional se deve ao fato de ser uma plataforma compatível com os recursos de hardware disponíveis para execução das simulações. Nas simulações, cada servidor corresponde a uma máquina virtual, gerenciadas pelo software VMware Server 2.0.2.

As máquinas virtuais são executadas em uma unidade de processamento com um processador Intel Core 2 Quad Q8200 2.33Ghz (4 núcleos), 8GB de memória RAM e 1TB de espaço de armazenamento em disco, gerenciados pelo sistema operacional Windows 7 64 bits. A Figura 20 apresenta a plataforma do sistema de testes.

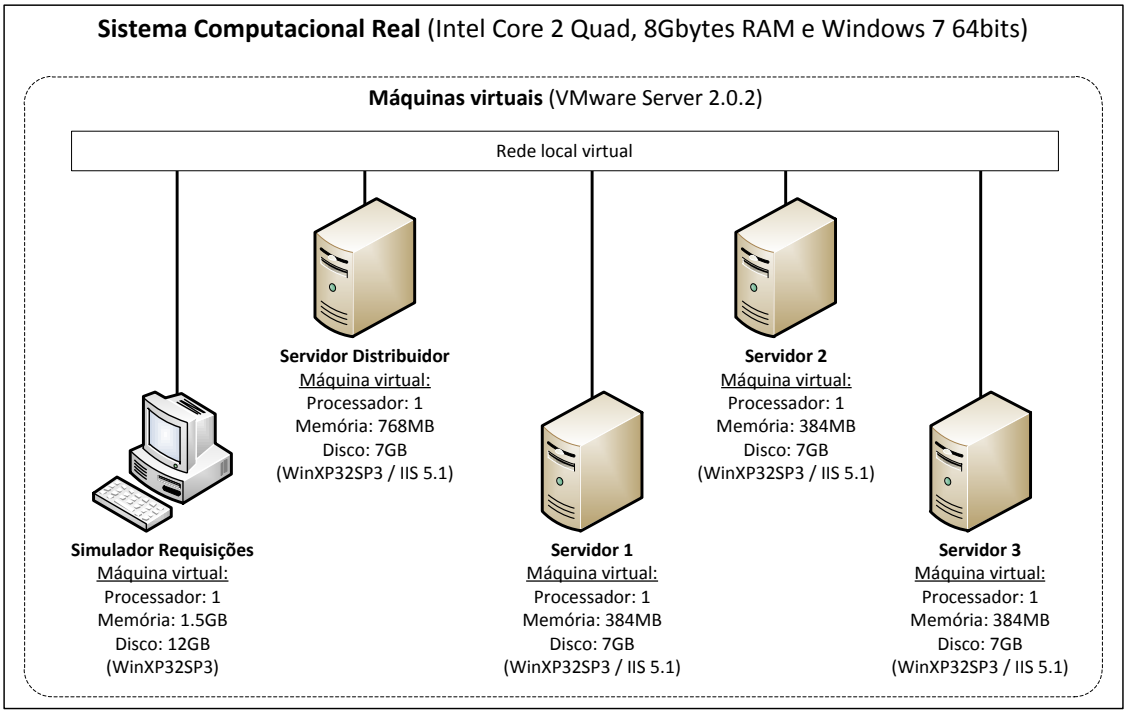


Figura 20 – Plataforma do sistema de testes

6.3. Simulador de Requisições

O Simulador de requisições corresponde a um cliente do serviço *Web* Distribuidor e tem como funções principais:

1) Gerar as requisições de serviços a serem transmitidas para o serviço *Web* Distribuidor; e
2) Permitir a geração de um relatório sobre os resultados de cada requisição efetuada.

O sistema é implementado, basicamente, por um software que possui quatro laços aninhados, que permitem a execução de requisições de forma que sejam combinados os parâmetros relevantes para a análise:

- Número de servidores disponíveis para atendimento das requisições (esse valor é configurável dinamicamente no serviço *Web* Distribuidor);
- Ativação ou não do escalonamento SRPT (*Shortest Remaining Processing Time*);
- Taxa extra de processamento (concorrente) para simulação de carga (transmitido pelo cliente diretamente para os servidores 1, 2 e 3, que executam os serviços). A taxa de processamento adicional (variando entre 10% e 60% do tempo de processamento) permite que sejam obtidos os

tempos de resposta para diversos níveis de processamento, de forma que seja possível avaliar o comportamento do tempo de resposta para diversas taxas de processamento do sistema.

O laço mais interno é responsável pela geração de 600 requisições, com distribuição exponencial do intervalo entre chamadas. Para números de requisições maiores que esse não foram observadas alterações significativas na distribuição de valores dos tempos de resposta, logo, esse valor foi utilizado para que o tempo total na simulação fosse reduzido.

A Figura 21 apresenta um diagrama que representa a sequência de execuções no Simulador de Requisições.

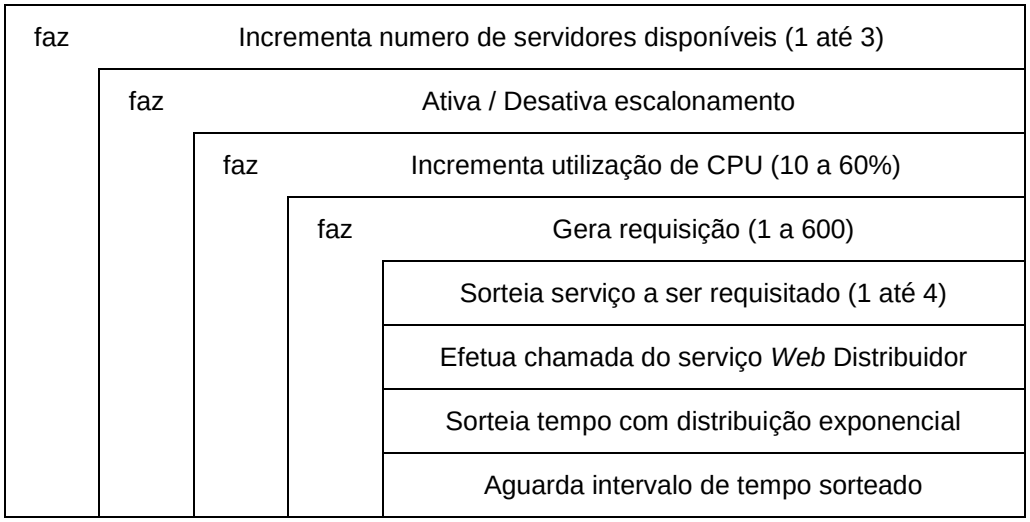


Figura 21 – Execução do Simulador de Requisições

Na execução do simulador estão envolvidos dois sorteios (número do serviço a ser chamado e intervalo de tempo a ser aguardado para a execução da próxima requisição), que são efetuados por meio da geração de números pseudoaleatórios, obtidos pela chamada de funções disponíveis no ambiente de desenvolvimento.

6.4. Serviços *Web* 1, 2, 3 e 4

Os serviços *Web* utilizados para as simulações foram desenvolvidos especificamente para esse fim. Como se pretende avaliar a contribuição da arquitetura proposta para o determinismo do tempo de resposta das requisições efetuadas, utiliza-se, como variável controlada, o tempo de execução desses serviços.

Dessa forma, foram gerados quatro serviços para os quais é possível determinar o tempo de execução médio, com baixa variância, obtido quando o sistema apresenta baixa carga de processamento (taxa de processamento menor que 3%, por exemplo). Esse valor médio pode ser comparado, então, com os valores (crescentes) de tempo de processamento quando o sistema apresenta diferentes taxas de utilização. Um maior tempo de execução contribui para que o tempo de espera das requisições pendentes também seja maior, implicando no aumento do tempo de resposta dessas requisições.

O algoritmo de processamento utilizado, nesse caso, é irrelevante para o propósito deste estudo, uma vez que a variável controlada é o tempo de processamento. Assim sendo, o resultado do processamento efetuado também não é considerado.

Os quatro serviços desenvolvidos baseiam-se em um mesmo núcleo, no qual se modifica a quantidade de processamento necessária para sua execução, de forma que os serviços 1, 2, 3 e 4 apresentem tempos médios de execução distintos.

Esses tempos médios distintos são utilizados para determinação da fila de prioridade no escalonamento, em que faixas de valores de tempos de execução determinam a prioridade de execução (serviços com tempo de execução médio menor têm maior prioridade de execução). Vale ressaltar que, nesse caso, o valor constante é utilizado para que se possa analisar o comportamento do tempo de resposta dos serviços, bem como da influência (ou não) das soluções propostas neste trabalho.

O núcleo do serviço *Web* padrão é implementado pela classe TServGenKernel, cuja representação está ilustrada na Figura 22.

Figura 22 – Classe TServGenKernel

A classe TServGenKernel corresponde ao núcleo de processamento de um serviço e é responsável pela execução da rotina de cálculo que demanda uma determinada quantidade de processamento do sistema no qual é executada (especificada por um parâmetro especificado no construtor da classe). Os principais métodos dessa classe são:

1) **CalcExec**:

   Execução do cálculo propriamente dito, que demanda uma determinada quantidade de processamento. Essa quantidade é diretamente proporcional a um parâmetro passado no construtor da classe que identifica o tipo de serviço a ser executado (1, 2, 3 ou 4).

   A rotina de cálculo consiste em três laços aninhados, que executam uma operação de multiplicação em ponto-flutuante. A cada 20 execuções do laço externo, a rotina tem sua execução suspensa por 1ms, fazendo com que o processador não permaneça 100% do tempo em nível máximo de processamento. Essa suspensão pode ser interpretada como uma simulação de operações de acesso ao sistema de discos, por exemplo.

   A Figura 23 apresenta um diagrama que representa a sequência de execuções no método CalcExec. A variável maxLoop indicada na Figura 23 é carregada com diferentes valores, de acordo com o tipo de serviço sendo executado (1, 2, 3 ou 4).

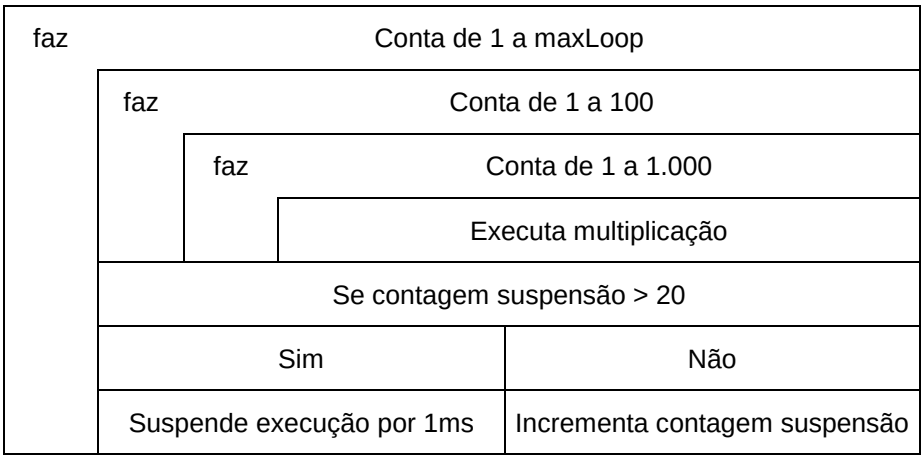


Figura 23 – Execução do Simulador de Requisições

   Para os serviços 1, 2, 3 e 4, os valores atribuídos à variável maxLoop são 750, 1.500, 2.000 e 4.000, respectivamente. O valor mínimo atribuído para a variável maxLoop, assim como para os laços internos de 1 a 100 e 1 a 1000,

implica em um tempo de processamento total compatível com uma baixa taxa de amostragem do nível utilização do processador (quanto menor essa taxa de amostragem, menor a interferência no tempo de processamento do sistema).

Os demais valores determinam diferentes (maiores) tempos totais de processamento, de forma que o comportamento do algoritmo SRPT possa ser avaliado.

2) **GetExecTime**:

   Fornece o tempo total de execução da rotina;

3) **GetStartTime**:

   Fornece o instante em que a execução foi iniciada;

4) **GetStopTime**:

   Fornece o instante em que a execução foi concluída;

5) **GetCPUUtilLastStart**:

   Fornece o valor médio da taxa de processamento durante a execução dos cálculos. Esses valores são fornecidos por rotinas fornecidas pelo sistema operacional.

   Essas rotinas são chamadas, periodicamente, por uma *thread* disparada na execução do construtor da classe TServGenKernel. Essa *thread* consiste em um único laço que executa chamadas a rotinas do sistema operacional para cálculo do nível de processamento, a cada 20ms.

   Os valores obtidos são utilizados para cálculo do nível médio de processamento durante a execução dos cálculos. A Figura 24 apresenta o diagrama da classe TThrCPUTimeLog. Nessa classe, o método **GetCPUAvgUsage** fornece o tempo médio calculado durante a execução do método **CalcCpuUsage**.

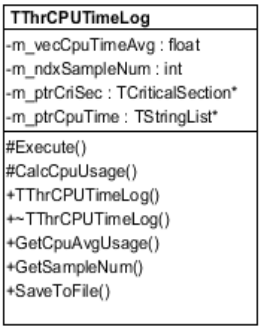


Figura 24 – Classe TThrCPUTimeLog

Na implementação dos módulos que constituem o sistema foram utilizadas diversas *threads* de execução, com o objetivo de compartilhar o espaço de endereçamento do processo principal (programa em execução, acompanhado dos valores atuais do contador de programa, registradores e variáveis). A utilização de *threads* de execução permite que um único processo possa executar mais de uma função quase que simultaneamente, por meio de compartilhamento de recursos e tempo. O gerenciamento do contexto de execução é feito pelo sistema operacional e um dos fatores que contribuem para o ganho de desempenho é o fato de que na troca da *thread* que estiver sendo executada não há a necessidade de trocar também o espaço de endereçamento, como aconteceria se a troca ocorresse entre processos distintos (TANENBAUM, 2003).

Como mencionado, a classe TServGenKernel corresponde ao núcleo de processamento dos serviços 1, 2, 3 e 4. A Figura 25 apresenta o relacionamento entre as classes utilizadas na implementação do serviço 1 (representado pela classe T*Web*SrvGen01Impl). Vale ressaltar que, para os demais serviços, os relacionamentos são análogos, substituindo-se a identificação T*Web*SrvGen01Impl por T*Web*SrvGen02Impl, T*Web*SrvGen03Impl e T*Web*SrvGen04Impl.

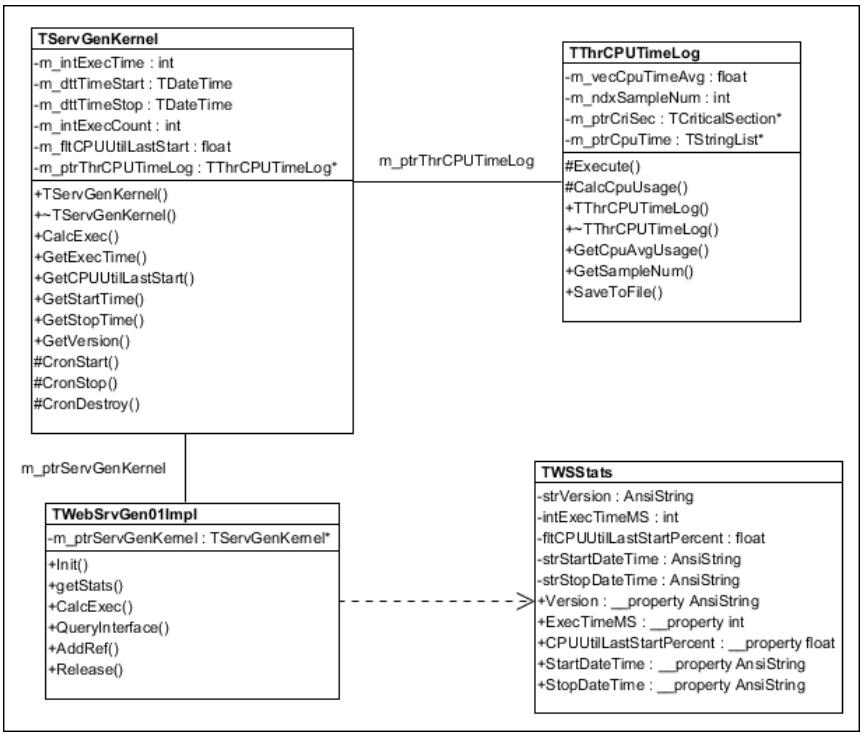


Figura 25 – Relacionamentos entre classes do Serviço 1

Na Figura 25, a classe T*Web*SrvGen01Impl apresenta dois métodos principais:

1) **CalcExec**:

   Executa a chamada do método **CalcExec** da classe TServGenKernel;

2) **getStats**:

   Responsável pelo fornecimento das estatísticas dos serviços, em objetos instanciados a partir da classe TWSStats. Os valores armazenados nesses objetos correspondem aos valores fornecidos pelos métodos públicos da classe TServGenKernel.

6.5. Serviço *Web* Distribuidor

O serviço *Web* Distribuidor é o serviço responsável pelo recebimento das requisições, análise da requisição recebida, escalonamento dessa requisição e solicitação para execução pelo servidor correspondente.

A classe T*Web*SrvBrokerImpl, cujo diagrama é apresentado na Figura 26, corresponde à classe que implementa esse serviço.

Figura 26 – Classe T*Web*SrvBrokerImpl

Os principais métodos da classe T*Web*SrvBrokerImpl são:

1) **setSRPTON**:

   Ativa ou desativa o escalonamento SRPT. Caso o escalonamento SRPT não esteja ativo, o escalonamento das requisições é determinado pela ordem de chegada das requisições.

   A retirada de requisições da fila de requisições sempre considera todos os servidores disponíveis, seja no escalonamento SRPT, seja no escalonamento por ordem de chegada, isto é, a remoção de uma requisição da fila de requisições depende somente da disponibilidade de um servidor para sua execução. A diferença, nesse caso, entre os dois tipos de escalonamento é que, para o escalonamento SRPT, a execução das requisições em uma fila menos prioritária somente ocorre quando todas as requisições existentes em uma fila mais prioritária tiverem sido consumidas e, no caso do escalonamento por ordem de chegada, as requisições prioritárias são as que foram inseridas em primeiro lugar na fila (FIFO – *First In First Out*).

2) **setServersSet**:

   Determina o número de servidores ativos para execução das requisições. Esse método é utilizado para execução de simulações com diferentes quantidades de servidores disponíveis.

   A configuração do total de servidores, bem como os serviços disponíveis por servidor é lida no construtor dessa classe, porém esse método determina quais servidores devem ser utilizados para execução das requisições.

3) **setRequest**:

   Recebe uma requisição de um cliente. Essa requisição é decodificada pelo método **ParseRequest** e colocada na fila para atendimento. A fila em que a requisição é inserida depende do tipo de escalonamento utilizado: sem escalonamento, escalonamento por ordem de chegada (fila única) ou escalonamento SRPT (quatro filas de prioridade, sendo prioritárias em uma mesma fila as requisições com menor tempo previsto de execução, passado como parâmetro).

   Na chamada do serviço são fornecidos os dados da requisição a serem utilizados na sua execução, como identificação do serviço a ser executado,

parâmetros de entrada e restrição temporal (que corresponde ao tempo médio de execução do serviço a ser executado, previamente determinado).

O serviço a ser executado corresponde a um dos serviços *Web* criados para esse fim, conforme detalhado no item 6.4 deste documento. Assim, na criação de uma requisição, o cliente não especifica o endereço do servidor no qual essa requisição será executada, mas apenas a identificação do tipo de serviço desejado e sua restrição temporal.

Ao ser processada a requisição, o serviço *Web* Distribuidor retorna para o cliente um identificador unívoco criado para a chamada efetuada. Esse identificador pode ser utilizado pelo cliente para acompanhamento do processamento de sua requisição.

4) **getResult**:

Fornece o resultado de uma requisição, cujo identificador (gerado na chamada de setRequest) é passado como parâmetro.

Estão implementados também outros métodos para leitura de listas de resultados, como os métodos getResults (retorna os resultados disponíveis para todas as requisições cuja execução já foi concluída) e getResultsRange (retorna os resultados para um intervalo de identificadores passado como parâmetro).

Os valores obtidos na execução de uma requisição são armazenados em objetos criados a partir da classe TResult, cujo diagrama é apresentado na Figura 27.

Figura 27 – Classe TResult

Nesse tipo de objeto são armazenados dados como número da requisição, número do serviço executado, quantidade de servidores disponíveis no momento da execução, instantes em que a requisição foi colocada na fila, instante em que a requisição teve sua execução iniciada, tempo de execução e tempo total de resposta (incluindo latência de rede), entre outros.

5) **getResultsCount**:
   Fornece a quantidade de requisições disponíveis para consulta de resultados.

O serviço *Web* Distribuidor possui uma *thread* de gerenciamento, implementada na classe TThrRequestManager, cujo diagrama é apresentado na Figura 28.

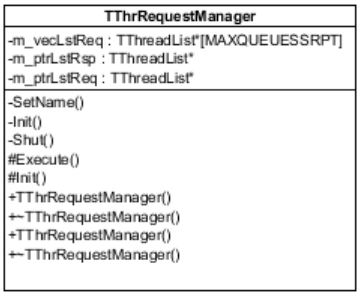


Figura 28 – Classe TThrRequestManager

Essa *thread* de execução é responsável pela leitura das filas de requisições e disparo dessas requisições para os servidores disponíveis. Para o disparo de cada requisição é criada uma nova *thread* de execução, implementada na classe TTHr*Web*SrvCall, cujo diagrama é apresentado na Figura 29.

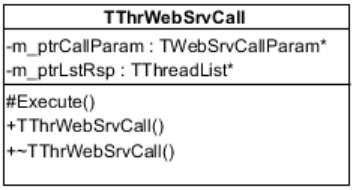


Figura 29 – Classe TThr*Web*SrvCall

Ao final da execução de uma *thread* de serviço, o resultado é armazenado na fila de respostas do sistema. A criação de uma nova *thread* recebe parâmetros de execução, que são armazenados em objetos criados a partir da classe T*Web*ServCallParam.

A Figura 30 apresenta a classe T*Web*SrvCallParam e os relacionamentos entre as principais classes que constituem o serviço *Web* Distribuidor.

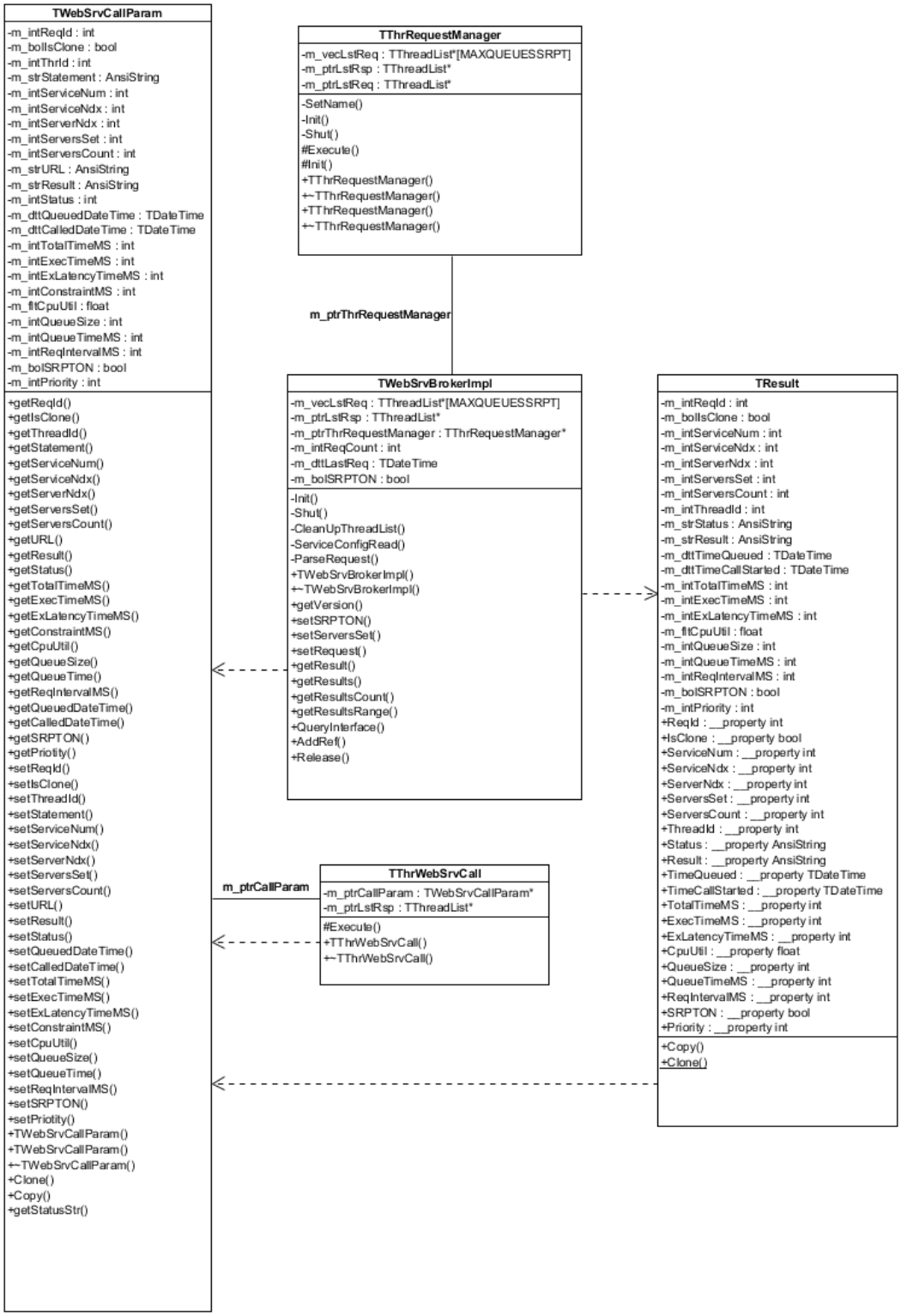


Figura 30 – Relacionamentos entre classes do serviço *Web* Distribuidor

Os objetos criados a partir da classe T*Web*ServCallParam são estruturalmente semelhantes aos objetos criados a partir da classe TResult, sendo os objetos criados a partir da classe T*Web*ServCallParam utilizados pelas estruturas internas de

armazenamento do serviço *Web* Distribuidor e os objetos da classe TResult transmitidos para os clientes quando solicitados.

O tratamento de uma requisição consiste, basicamente, no recebimento da chamada, geração do identificador da requisição (mantido por um contador global de requisições mantido pelo serviço) e enfileiramento dessa chamada. O identificador da requisição é retornado para o cliente para que, posteriormente, o resultado da execução possa ser consultado.

O processamento dessa requisição é executado pela *thread* TThrRequestManager do serviço *Web* Distribuidor e o seu resultado é armazenado na fila de respostas.

A Figura 31 apresenta o ciclo de execução de requisições, incluindo solicitação, retirada da fila de requisições pelo gerenciador de requisições (TThrRequestManager), criação das *threads* de execução (TThr*Web*SrvCall) e armazenamento na fila de respostas.

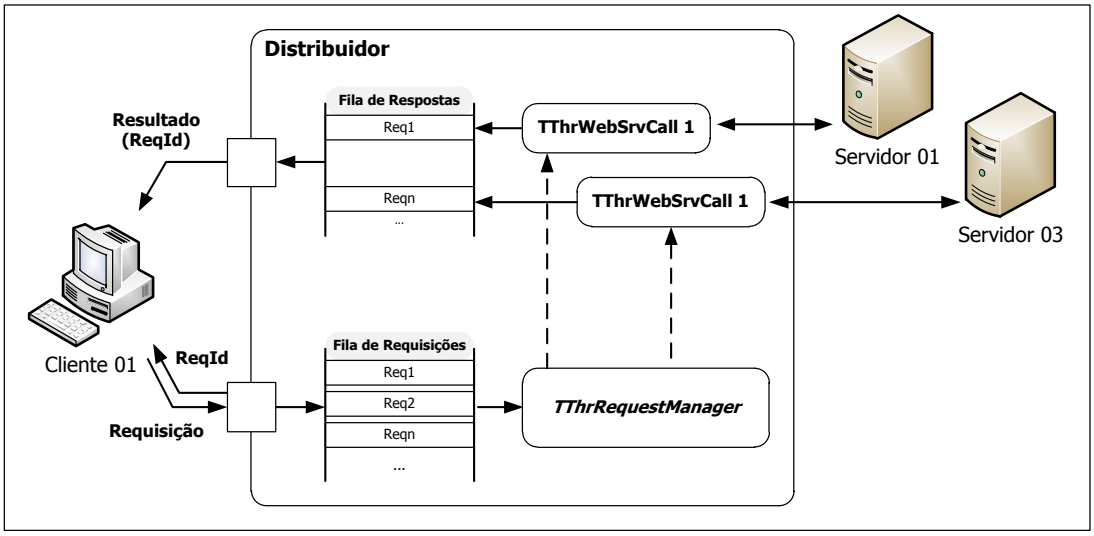


Figura 31 – Gerenciamento das requisições pelo serviço *Web* Distribuidor

6.5.1. Lógica de Controle

O serviço *Web* Distribuidor recebe as chamadas dos clientes e as insere em uma fila de requisições. Essa fila de requisições tem seu acesso protegido por meio da imposição de bloqueios, mantendo a integridade dos dados no caso em que ocorram tentativas simultâneas de acesso. Esse tipo de proteção é necessário devido à natureza assíncrona deste tipo de sistema, isto é, não se pode prever o comportamento dos clientes nem os instantes exatos em que retornarão as respostas das consultas efetuadas aos diversos servidores, não sendo possível,

dessa forma, estabelecer sincronismo entre os diversos eventos que ocorrem durante o funcionamento do sistema. Para a fila de resposta valem as mesmas considerações, isto é, seu acesso também é gerenciado por meio de bloqueios.

Como mencionado, os serviços são executados em servidores virtuais, controlados pelo gerenciador de máquinas virtuais. O gerenciamento de recursos pelo gerenciador de máquinas virtuais corresponde a uma fonte extra de não-determinismo, pois não se pode determinar, antecipadamente, como são distribuídos os recursos reais (processadores e sistema de discos, por exemplo) para execução das máquinas virtuais.

Dessa forma, para que seja minimizado esse efeito, na execução das simulações em que o número de servidores utilizados não corresponde à totalidade de servidores disponíveis, requisições extras são disparadas, de forma que o gerenciador das máquinas virtuais sempre tenha uma demanda semelhante por recursos, uniformizando a distribuição de recursos para diferentes quantidades de servidores utilizados.

Por exemplo, considerando três servidores (máquinas virtuais) disponíveis para gerenciamento e, durante uma etapa da simulação, somente um desses servidores estiver configurado como disponível, dois servidores ficarão ociosos. Para que a distribuição de recursos pelo gerenciador das máquinas virtuais seja aproximadamente uniforme, requisições extras (clones) são disparadas para os servidores ociosos, porém seu resultado é descartado ao final da execução.

Vale ressaltar que a criação de *threads* extras de execução (clones) somente ocorre quando o número de servidores utilizados para simulação é menor que o número total de servidores configurados como disponíveis. Para, por exemplo, três servidores gerenciados pelo gerenciador das máquinas virtuais e somente um configurado como disponível na simulação, dois clones da requisição são gerados para execução pelos servidores ociosos e, assim por diante.

A Figura 32 apresenta a sequência de execuções no gerenciador de requisições (classe TThrRequestManager). Nessa figura é possível observar que, entre verificações de existência de itens na fila de requisições, o sistema aguarda um intervalo de 2ms. Esse intervalo de espera é utilizado para que o laço de verificação não conduza o sistema a uma taxa de processamento de 100% (quanto menor o tempo, maior a influência na taxa de processamento), porém é importante observar também que esse valor não pode introduzir um atraso significativo no começo de

atendimento de uma requisição (quanto maior o tempo, maior o atraso no atendimento).

<table>
<tr><td>Faz</td><td colspan="3">Para sempre</td></tr>
<tr><td></td><td>faz</td><td colspan="2">Enquanto houver itens na fila de requisições</td></tr>
<tr><td></td><td></td><td colspan="2">Servidor disponível para execução?</td></tr>
<tr><td></td><td></td><td>Sim</td><td>Não</td></tr>
<tr><td></td><td></td><td>Retira item prioritário da(s) fila(s) de requisições</td><td rowspan="3"></td></tr>
<tr><td></td><td></td><td>Cria thread para execução da requisição</td></tr>
<tr><td></td><td></td><td>Cria thread(s) para execução do(s) clone(s) da requisição</td></tr>
<tr><td></td><td colspan="3">Aguarda intervalo de 2ms</td></tr>
</table>

Figura 32 – *Thread* do TThrRequestManager

6.6. Considerações Finais deste Capítulo

Este capítulo apresentou a implementação dos principais elementos que constituem os sistemas utilizados na simulação da arquitetura proposta, cujos resultados são apresentados no próximo capítulo.

Foram apresentadas as principais classes utilizadas no serviço *Web* Distribuidor, responsável pelo gerenciamento e escalonamento das requisições, bem como os algoritmos utilizados na construção do sistema simulador e no escalonamento de requisições.

7. RESULTADOS OBTIDOS

Este capítulo apresenta os resultados obtidos de simulações efetuadas a partir da execução de sistemas implementados de acordo com a arquitetura descrita no capítulo 5.

Para observação do comportamento dos sistemas que compõem o ambiente de testes, as simulações são parametrizadas por quantidade de servidores envolvidos na simulação, taxa de utilização da unidade de processamento e tipo de escalonamento utilizado.

São apresentadas as condições de contorno impostas para avaliação dos resultados das simulações, os resultados propriamente ditos e, ao final, os resultados consolidados, que correspondem a comparações entre os resultados obtidos para diferentes parâmetros de execução.

7.1. Condições de Contorno

A proposta de uma Arquitetura Orientada a Serviços que permita a sua utilização em sistemas com restrições temporais (não críticos quanto à segurança) tem como objetivo a melhoria do determinismo no tempo de resposta das requisições de clientes ao sistema.

Para que seja possível avaliar os resultados, o parâmetro principal a ser observado é a variação no tempo de resposta das requisições, de acordo com diferentes configurações do sistema em estudo.

Pelo exposto na seção 3.2, o modelo de filas utilizado nas simulações corresponde a um modelo M / M / 1 aberto. Nesse tipo de sistema, considerando conhecida a taxa de chegada de requisições (com distribuição exponencial de média λ) e com o sistema operando com sua fila vazia ou com baixa ocupação, é possível verificar, teoricamente, o crescimento exponencial dos valores da relação entre o tempo de resposta e o tempo de serviço, de acordo com o aumento na taxa de utilização do sistema. Para sistemas com múltiplos servidores, esse crescimento é menor quanto maior o número de servidores disponíveis.

No modelo teórico, são considerados os tempos de serviço e de resposta (tempo de serviço somado ao tempo de espera), porém, na simulação realizada existem tempos adicionais que devem ser considerados, como, por exemplo, a latência de

rede. Conforme apresentado na seção 5.2, a arquitetura proposta considera como requisito a operação em redes locais gerenciadas, nas quais a latência de rede possa ser monitorada, sendo possível estabelecer valores máximos de ocorrência.

A Figura 33 apresenta um diagrama que representa a requisição feita por um cliente ao serviço *Web* Distribuidor, a qual aguarda em fila e é posteriormente transmitida ao servidor destino, que executa o serviço solicitado cuja resposta é, então, colocada na fila de respostas para que o cliente possa consultá-la.

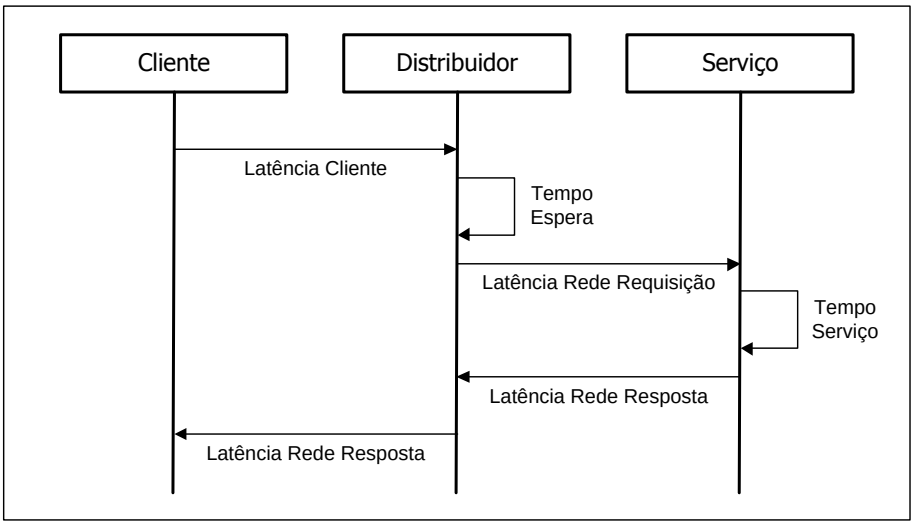


Figura 33 – Tempos em uma chamada

No sistema utilizado para simulação, a rede em questão corresponde a uma rede local virtual, conforme descrito na seção 0. Para obtenção dos resultados, são feitas três considerações:

1) Por simplificação, não são considerados os tempos envolvidos na transmissão da requisição pelo cliente (Latência Cliente, na Figura 33), nem na consulta, pelo cliente do resultado obtido (Latência Rede Resposta, na Figura 33). Os valores de tempo considerados são entre o serviço *Web* Distribuidor e os Serviços destino, uma vez que o sistema de filas se localiza entre esses componentes, simplificando os cálculos de latência da rede (considera-se o instante inicial a chegada da requisição e o instante final a inclusão da respectiva resposta na fila de respostas).
2) A latência é considerada aproximadamente constante, de forma que para o cálculo do tempo de requisição sejam considerados o tempo de serviço, o tempo de espera e o tempo de latência de rede;

3) A latência total de rede (entre serviço *Web* Distribuidor e serviço destino) é obtida subtraindo-se a soma dos valores do tempo de espera e serviço do tempo total da requisição.

Para consideração do tempo de latência como aproximadamente constante, foram observados os valores de latência introduzidos nas requisições efetuadas. Para as requisições efetuadas e consideradas nessa análise (total de 24.989), a distribuição de frequências obtida é apresentada na Figura 34.

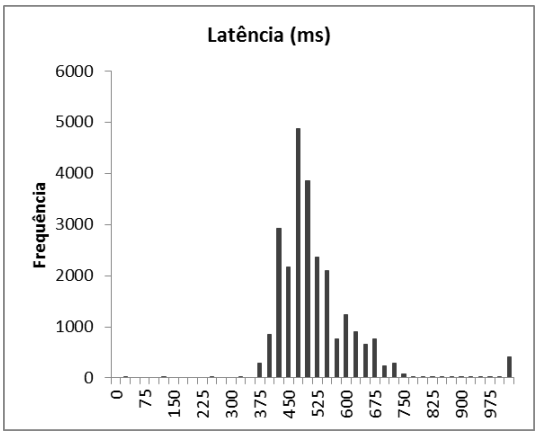


Figura 34 – Distribuição da latência

Considerando-se que as latências totais de rede situam-se entre 400ms e 650ms, 87,4% dos resultados obtidos situam-se nessa faixa (reduzindo-se o valor inicial de 24.989 para 21.839 registros), fornecendo um valor médio de latência de 492,9±59,3ms (desvio padrão de, aproximadamente, 12%). Esse descarte permite que sejam considerados os registros onde a latência apresenta um pequeno desvio, ou seja, pode ser considerada aproximadamente constante, conforme proposto como condição de contorno.

A Figura 35 apresenta a distribuição de latência após a eliminação dos registros com valores fora dos limites considerados.

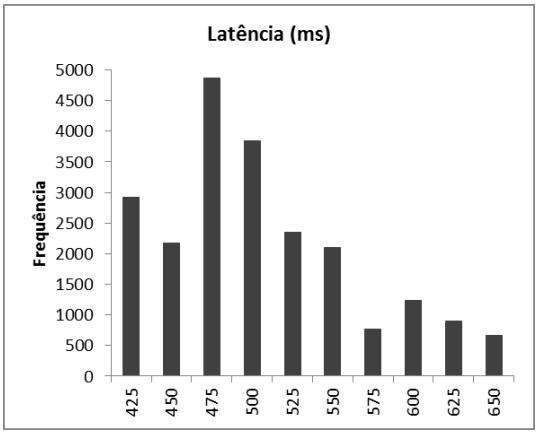


Figura 35 – Distribuição da latência entre 400ms e 650ms

O modelo proposto considera que o intervalo de tempo entre requisições apresenta distribuição exponencial, com média λ. No sistema construído para execução dos testes, a escolha do valor de λ considera a condição de contorno que especifica que o sistema deve operar sua fila vazia ou com baixa ocupação.

Dessa forma, o valor médio da taxa de chegada de requisições deve ser compatível com o valor médio do tempo de serviço (ou tempo de execução).

Conforme descrito na seção 6.4, foram desenvolvidos quatro serviços distintos, com tempo de execução aproximadamente constante. Os serviços desenvolvidos apresentam tempos de serviço distintos, de forma que se possa avaliar a influência do escalonamento SRPT (*Shortest Remaining Processing Time*) no determinismo do tempo de resposta.

Assim sendo, o valor utilizado para o intervalo médio entre requisições corresponde ao valor máximo entre os valores médios de cada serviço. Os valores médios de tempo de serviço para cada serviço *Web* desenvolvido estão apresentados na Tabela 6.

Os tempos médios de serviço indicados na Tabela 6 consideram os tempos de serviço obtidos nas execuções com taxa de utilização menores ou iguais a 3%.

Tabela 6- Tempos médios de execução de cada serviço.

Serviço	Tempo médio de serviço (ms)	Número de amostras
1	544,4±10,4	641
2	1.084,3±10,1	770
3	1.798,0±14,7	668
4	2.880,5±17,6	670

Como se considera o maior tempo de serviço entre os quatro serviços disponíveis, o valor médio para o intervalo entre requisições corresponde ao valor médio do tempo de execução do serviço 4, que é de aproximadamente 2.880ms, conforme apresentado na Tabela 6. Porém, para execução das simulações que resultaram nos valores apresentados na Tabela 6, o valor utilizado como intervalo de tempo entre requisições corresponde a 2.831ms e não 2.880ms. Esse valor de 2.831ms para o tempo médio de execução do serviço 4 é resultado de simulações executadas previamente, com o objetivo de estimar o tempo médio de execução desse serviço.
A Figura 36 apresenta a distribuição dos intervalos entre requisições, considerando as requisições para as quais a latência total de rede se situa entre 400ms e 650ms.

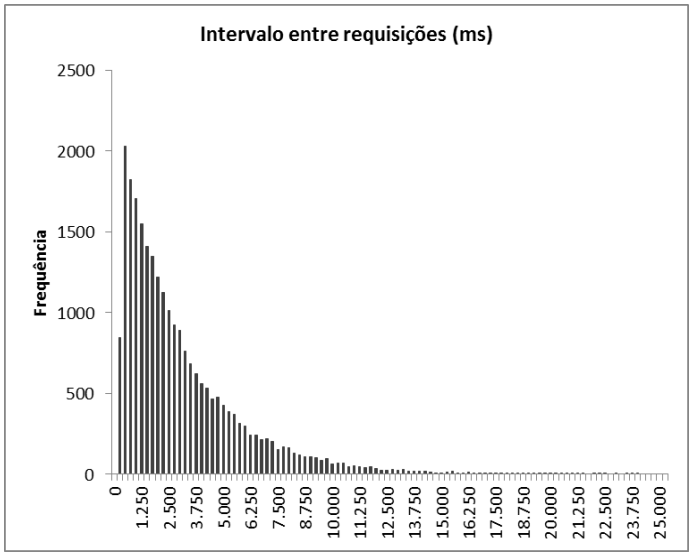


Figura 36 – Distribuição do intervalor entre requisições

7.2. Verificação de variações nos tempos de resposta

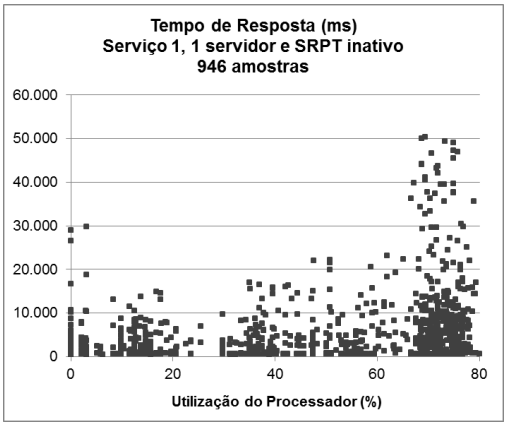
Estabelecidas as condições para execução do sistema (intervalo médio entre requisições e considerações sobre latência de rede), são apresentados os

resultados para as simulações efetuadas. Os resultados obtidos (Tempo de Resposta e a relação Tempo de Resposta / Tempo de Serviço) são parametrizados em função do serviço solicitado (1, 2, 3 ou 4), do número de servidores disponíveis para execução (1, 2 ou 3) e do tipo de escalonamento de requisições utilizado (escalonamento ativo ou não).

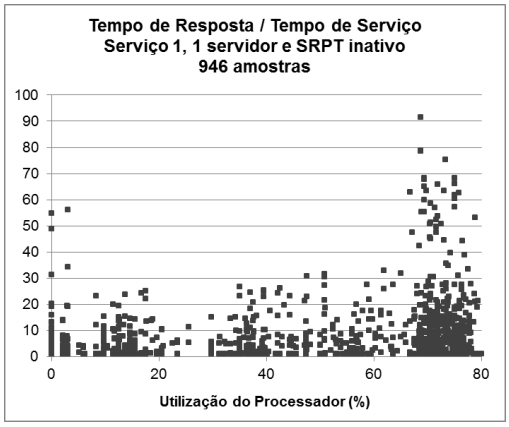
7.2.1. Resultados das simulações para o Serviço 1

A Figura 37 (a e b) e a Figura 38 apresentam os resultados obtidos para o Serviço 1, utilizando um único servidor e sem utilização de escalonamento.

Pode-se observar nos gráficos de dispersão de valores de tempo de resposta (Figura 37a) e da relação entre tempo de resposta e tempo de serviço (Figura 37b) em função da taxa de utilização que o aumento da taxa de utilização implica no aumento, não necessariamente linear, no tempo de resposta.



(a)



(b)

Figura 37 – Serviço 1: dispersão de valores (1 servidor, SRPT inativo)

Observando-se a Figura 38, que apresenta a distribuição de frequência desses valores, é possível observar que os valores de tempo de resposta estão distribuídos em diferentes faixas de valores e que há uma grande concentração de tempos de resposta em valores acima de 10 vezes o tempo médio de execução do serviço, que indica um aumento no tempo de espera para execução.

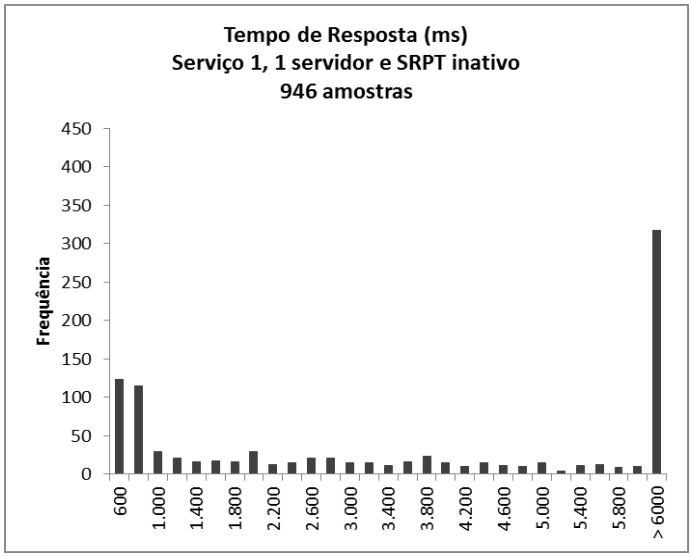


Figura 38 – Serviço 1: histograma do tempo de resposta (1 servidor, SRPT inativo)

A Figura 39 (a e b) e a Figura 40 apresentam os resultados para o Serviço 1, também utilizando um único servidor, porém com o escalonamento SRPT ativado.

Inicialmente, pode-se observar uma considerável redução nos valores das ordenadas apresentadas na Figura 39 (a e b) em relação à Figura 37 (a e b), que indicam uma redução da ordem de até 9 vezes no tempo máximo de execução.

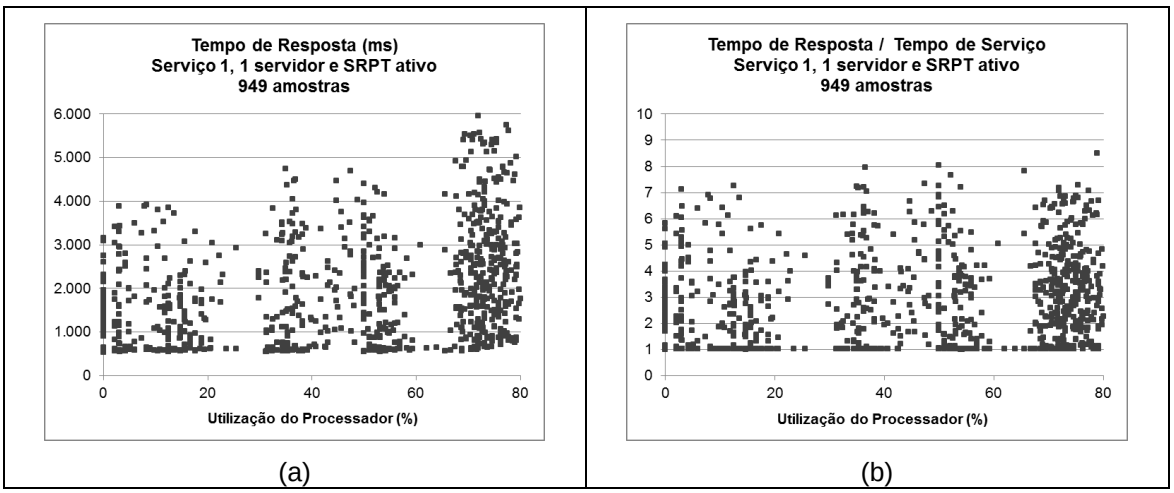


Figura 39 – Serviço 1: dispersão de valores (1 servidor, SRPT ativo)

Na Figura 40 é possível observar que a ativação do escalonamento SRPT contribui para uma maior concentração dos valores de tempo de resposta próximos ao tempo médio de serviço, que no caso do Serviço 1 é, aproximadamente, 544ms.

Pode-se observar também, na Figura 40, que valores de tempo de resposta acima de 6000ms são praticamente inexistentes. Essa melhora na distribuição de valores se deve ao fato de que, no algoritmo SRPT, as requisições com menores tempos de serviço são prioritárias em relação àquelas com maiores tempos de serviço. Vale

ressaltar que o Serviço 1 corresponde ao serviço com menor tempo médio de serviço (trata-se de uma característica incorporada ao serviço para execução das simulações).

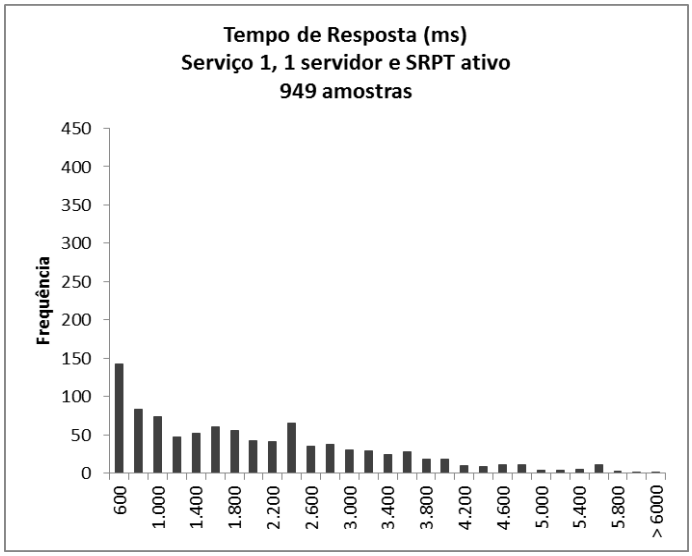


Figura 40 – Serviço 1: histograma do tempo de resposta (1 servidor, SRPT ativo)

A Figura 41 (a e b) e a Figura 42 apresentam os valores obtidos para os tempos de resposta do Serviço 1, porém considerando múltiplos servidores. São utilizados dois servidores para atendimento das requisições e sem utilização do escalonamento SRPT. Porém ao chegar uma nova requisição, se houver servidor disponível para atendimento dessa requisição, essa requisição é enviada ao servidor disponível.

Dessa forma, é utilizado um escalonamento que estabelece que a prioridade de atendimento de uma requisição depende do instante de chegada.

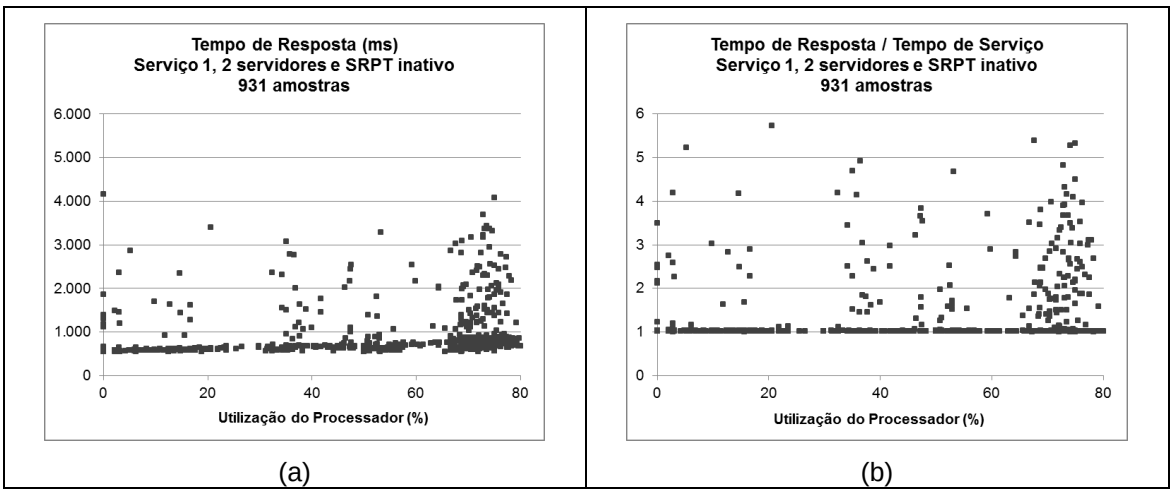


Figura 41 – Serviço 1: dispersão de valores (2 Servidores, SRPT inativo)

Pode-se observar na Figura 41 (a e b) uma diminuição na dispersão dos valores ao se comparar com os resultados indicados na Figura 40 (único servidor com escalonamento SRPT ativo).

Essa diminuição na dispersão de valores é evidenciada na Figura 42, que apresenta a distribuição de frequência de tempos de resposta, sendo que aproximadamente 85% das execuções apresentam valores menores que 1.000ms.

É importante ressaltar que o tempo de resposta calculado não inclui a latência de rede, sendo considerados somente o tempo de espera e o tempo de serviço. Porém, as execuções das requisições são efetuadas sobre diferentes níveis de solicitação, isto é, com a taxa de utilização dos servidores variando entre 0 e 80%.

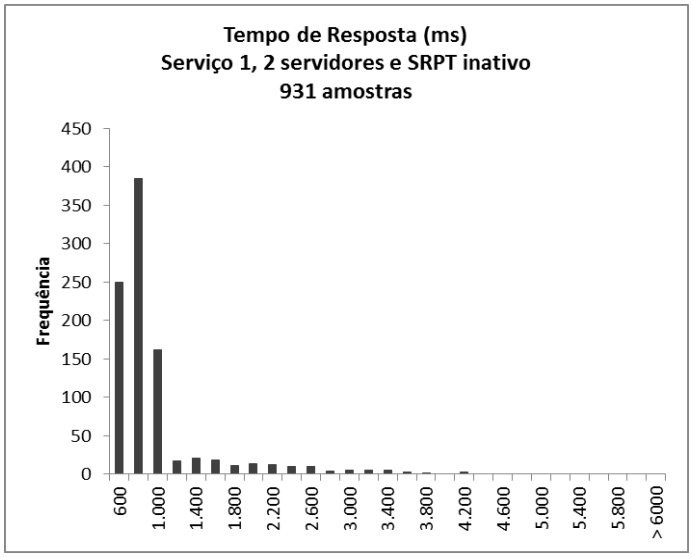


Figura 42 – Serviço 1: histograma do tempo de resposta (2 servidores, SRPT inativo)

A Figura 43 (a e b) e a Figura 44 apresentam os dados obtidos para os tempos de resposta do Serviço 1, com 2 servidores para atendimento, porém utilizando o escalonamento SRPT, no qual as requisições são atendidas de acordo com a seguinte ordem: requisições do Serviço 1, requisições do Serviço 2, requisições do Serviço 3 e, finalmente, requisições do Serviço 4.

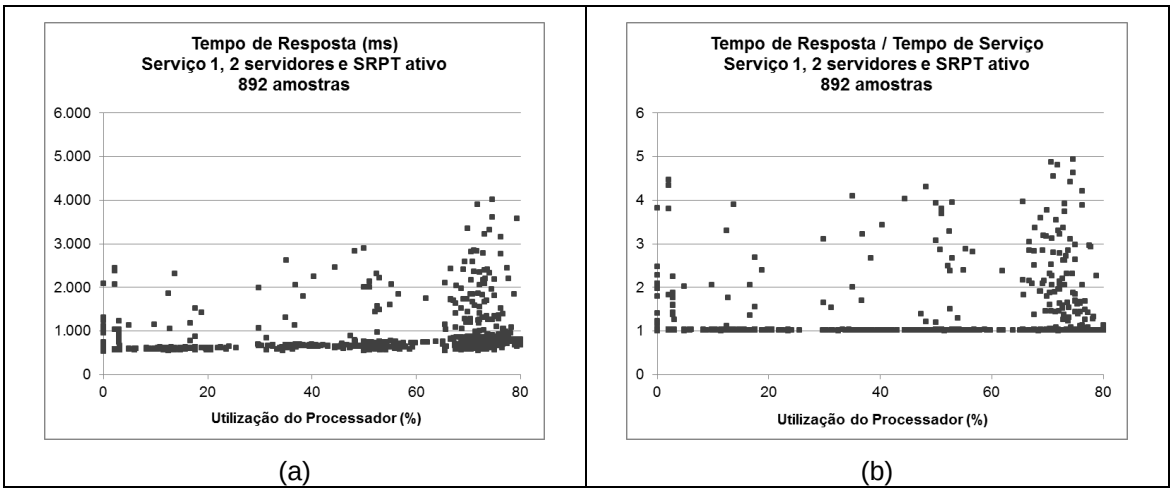


Figura 43 – Serviço 1: dispersão de valores (2 servidores, SRPT ativo)

Não se nota, nesse caso, uma modificação no perfil de distribuição de tempos de resposta, pois, comparando a Figura 42 com a Figura 44 não é possível observar mudança significativa nesse perfil. Considerando, então, os dados utilizados, tem-se que, com a utilização de 2 servidores, sem o escalonamento SRPT ativado, aproximadamente 85% dos valores de tempo de resposta estão abaixo do 1000ms, valor que se repete no caso em que o escalonamento SRPT é ativado.

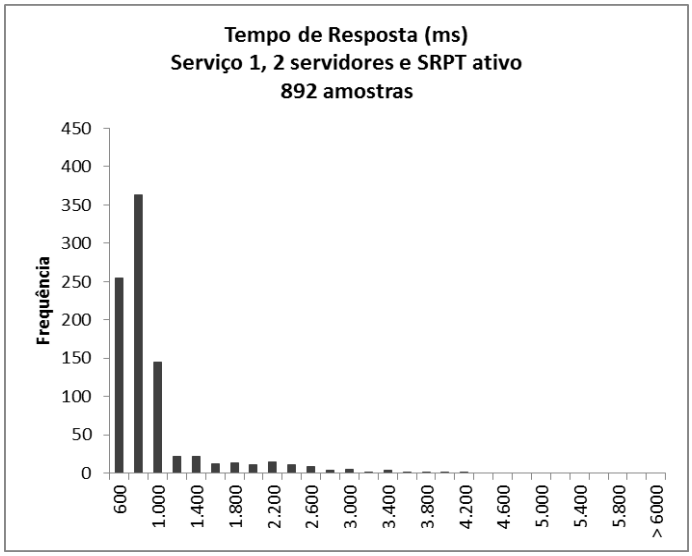


Figura 44 – Serviço 1: histograma do tempo de resposta (2 servidores, SRPT ativo)

Ainda para o Serviço 1 é utilizada uma terceira configuração de servidores, na qual 3 servidores estão disponíveis para o atendimento a requisições.

Pela Figura 45 (a e b) é possível observar uma redução ainda maior da dispersão dos valores de tempo de resposta em relação aos valores apresentados na Figura 43 (a e b). Pode-se observar também na Figura 45b que os valores da relação

tempo de resposta / tempo de serviço tendem a um, uma vez que o tempo de espera é muito pequeno, fazendo com que o valor do tempo de resposta se aproxime do valor do tempo de serviço.

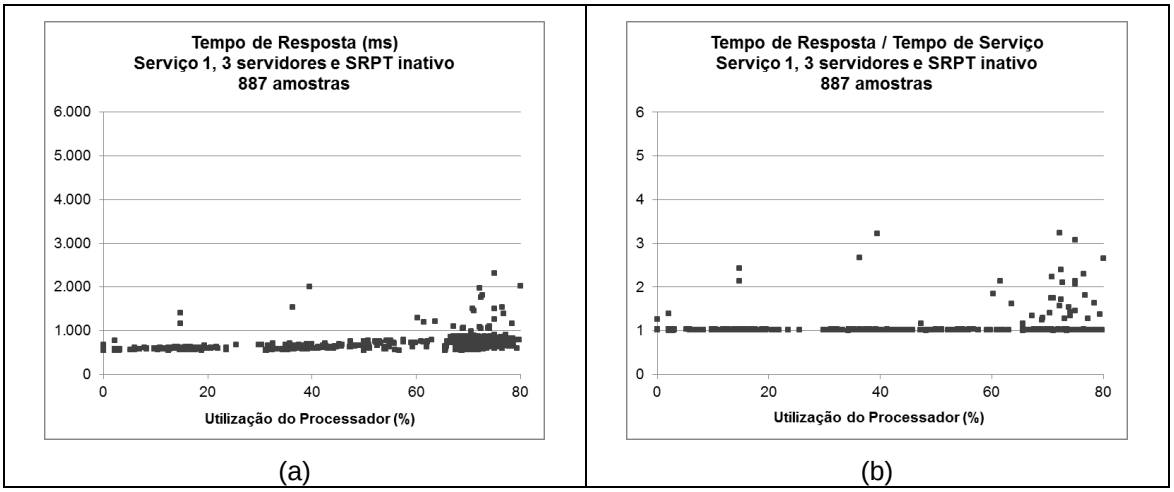


Figura 45 – Serviço 1: dispersão de valores (3 servidores, SRPT inativo)

A redução no tempo de espera deve-se ao aumento no número de servidores disponíveis para execução de requisições e também pelo fato de que o intervalo médio entre chegadas de requisições é cerca de 4,5 vezes o tempo médio de serviço para o Serviço 1.

Na Figura 46 é possível observar o efeito da introdução de mais um servidor para atendimento das requisições, aumentando a concentração de tempos de resposta abaixo de 1.000ms. Os resultados obtidos com 2 servidores disponíveis para atendimento das requisições, indicam que aproximadamente 85% das requisições são atendidas com tempo de resposta de até 1.000ms e, no caso de 3 servidores, esse valor aumenta para, aproximadamente, 96%.

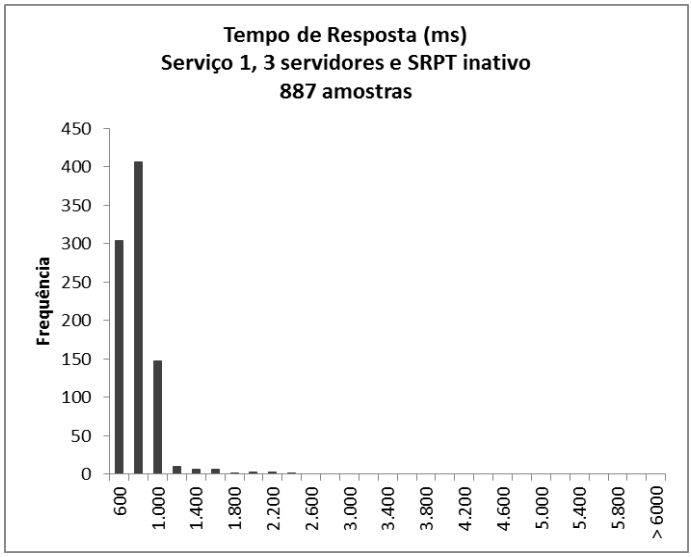


Figura 46 – Serviço 1: histograma do tempo de resposta (3 servidores, SRPT inativo)

A Figura 47 (a e b) e a Figura 48 apresentam os resultados para a dispersão dos tempos de resposta do Serviço 1, utilizando 3 servidores para atendimento e com o escalonamento SRPT ativado.

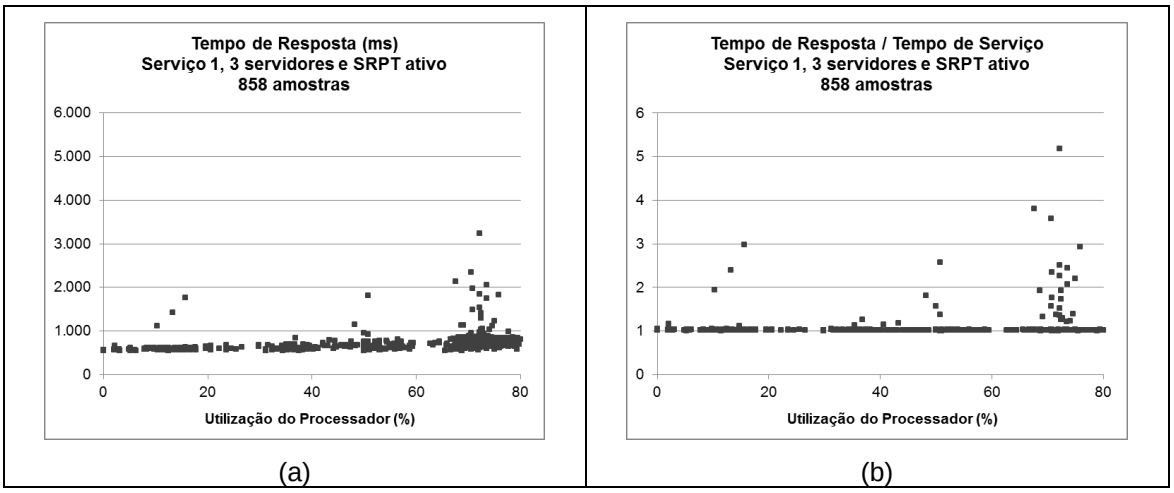


Figura 47 – Serviço 1: dispersão de valores (3 servidores, SRPT ativo)

Por comparação da Figura 48 com a Figura 46 não é possível observar modificação no perfil de distribuição dos tempos de resposta. Mesmo se forem considerados os dados utilizados na construção dos histogramas, há uma indicação de que aproximadamente 97% dos tempos de resposta situam-se abaixo de 1.000ms, contra os 96% obtidos sem a utilização do escalonamento SRPT. Essa diferença não pode ser considerada significativa em termos absolutos e também vale ressaltar que o número de amostras é, aproximadamente, 3% menor na simulação efetuada com o escalonamento SRPT ativo.

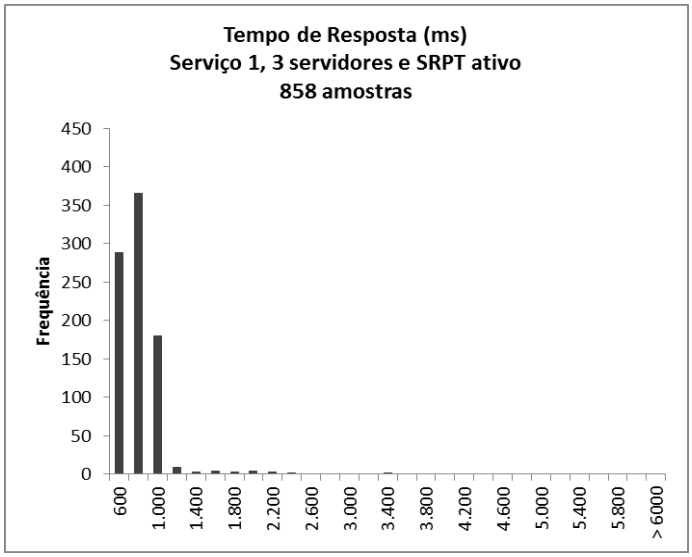


Figura 48 – Serviço 1: histograma do tempo de resposta (3 servidores, SRPT ativo)

7.2.2. Resultados das simulações para o Serviço 2

Os resultados das simulações para o Serviço 2 se aproximam dos resultados obtidos para o Serviço 1.

Para um único servidor disponível, é possível observar uma redução significativa da dispersão dos valores de tempo de resposta e da relação entre tempo de resposta e tempo de serviço ao se utilizar o escalonamento SRPT, conforme apresentado na Figura 49 (SRPT inativo) e na Figura 50 (SRPT ativo).

Vale ressaltar que valores distintos de tempos de processamento são obtidos, pois são impostas diferentes taxas de utilização nos sistemas sob teste.

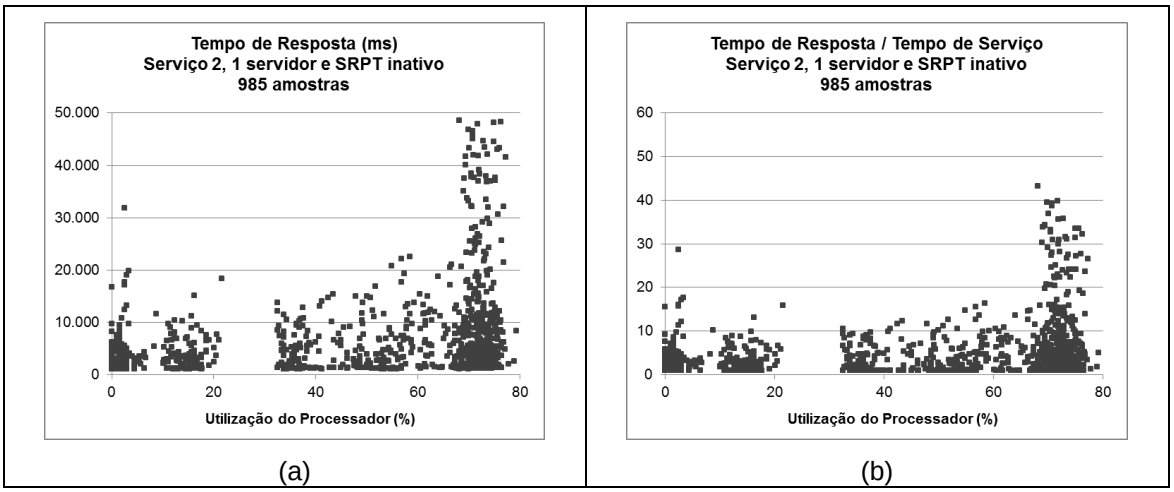


Figura 49 – Serviço 2: dispersão de valores (1 servidor, SRPT inativo)

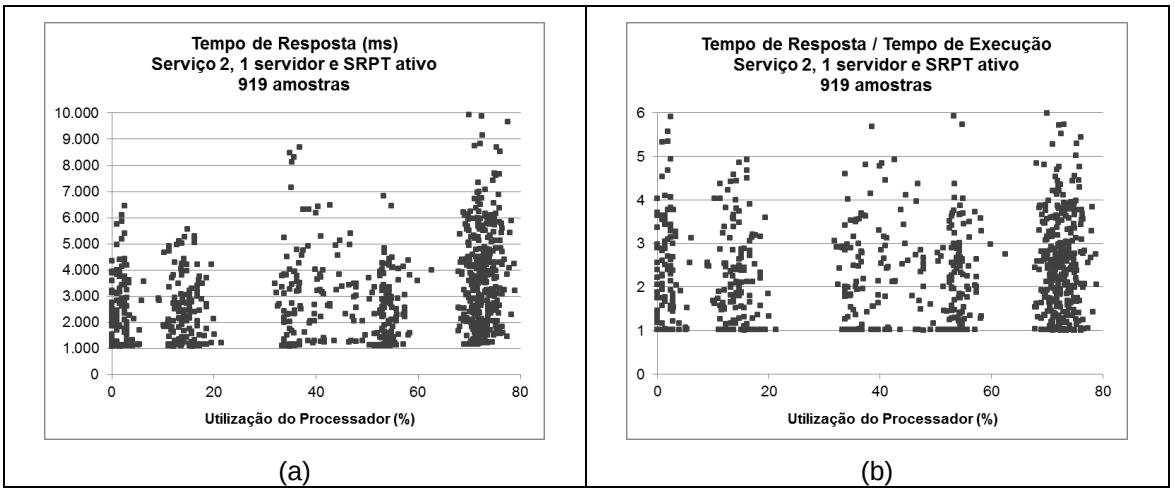


Figura 50 – Serviço 2: dispersão de valores (1 servidor, SRPT ativo)

Essa redução da dispersão pode também ser observada na comparação da Figura 51 (a e b) na qual pode ser observada uma redução na frequência de valores de tempo de resposta acima de 5.000ms, por exemplo, quando se utiliza ou não o escalonamento SRPT.

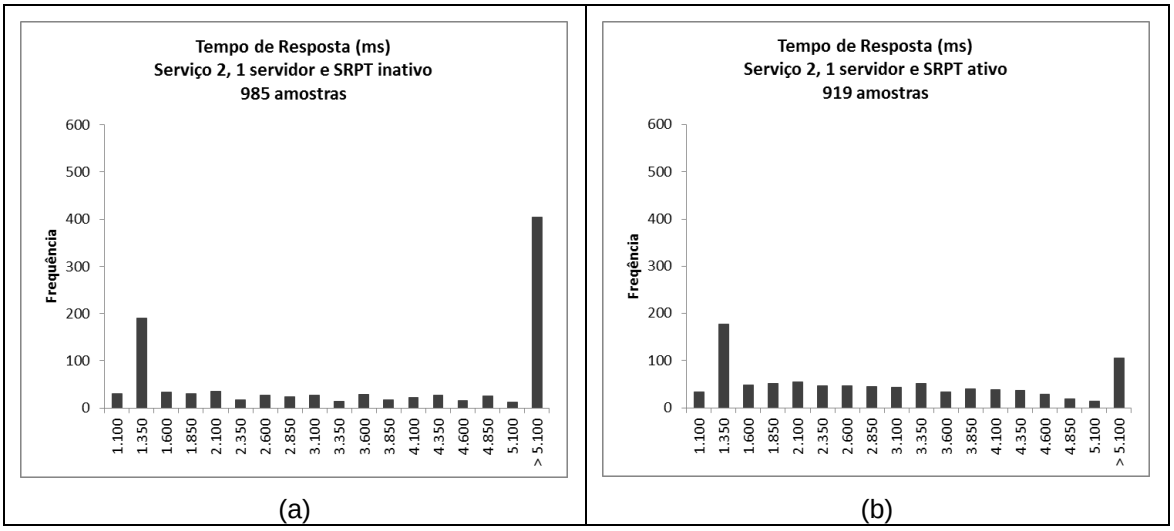
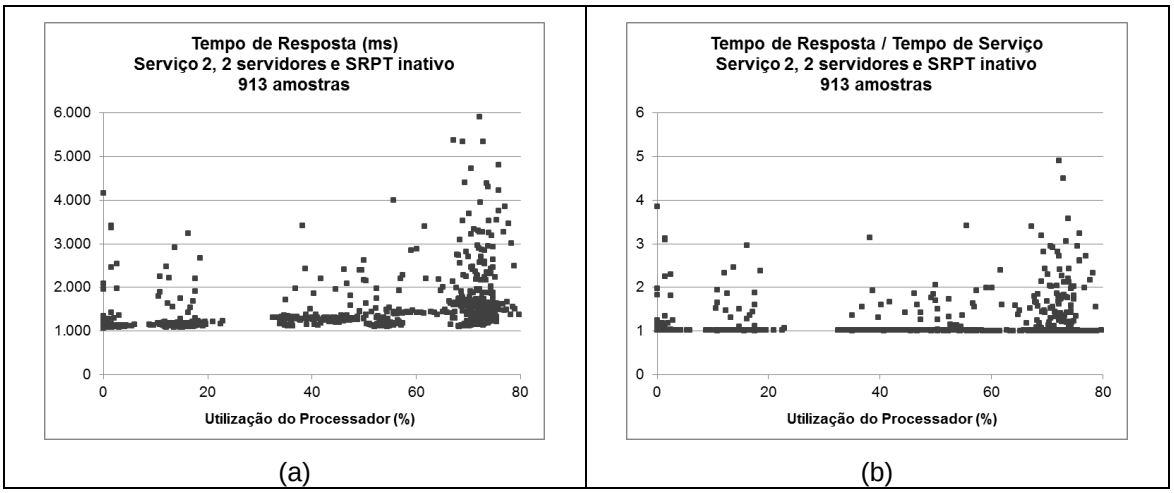


Figura 51 – Serviço 2: histogramas do tempo de resposta (1 servidor)

De forma semelhante aos resultados obtidos para os tempos de resposta das requisições do Serviço 1, é possível observar que, a introdução de mais servidores para atendimento às requisições contribui significativamente com a diminuição na dispersão dos valores dos tempos de resposta, para diferentes taxas de utilização impostas nos sistemas destino.

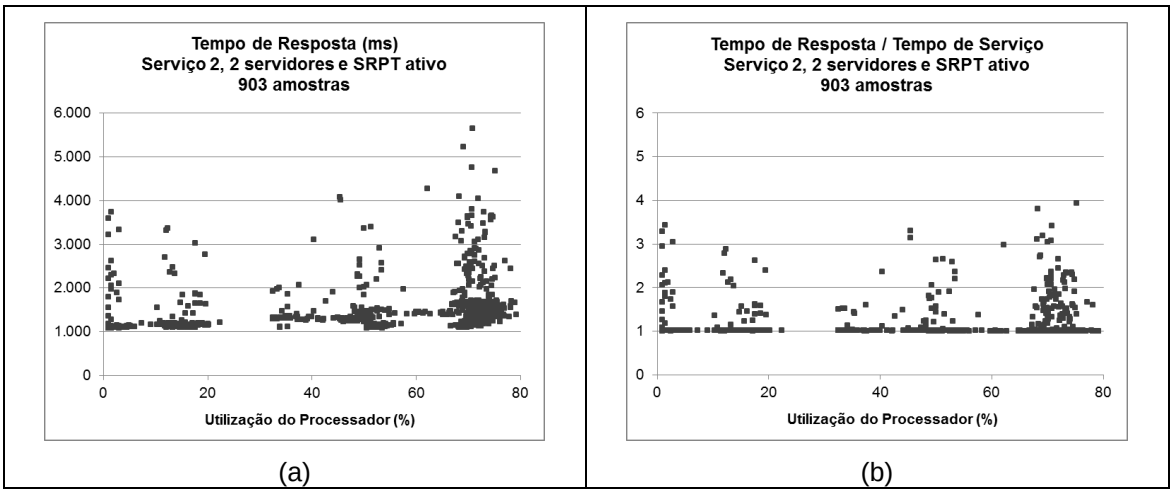
Ainda comparando os resultados obtidos para o Serviço 2 com os resultados obtidos para o Serviço 1, também, no caso do Serviço 2 não se observam contribuições significativas para diminuição da dispersão dos valores de tempo de execução com a

ativação do escalonamento SRPT. Os gráficos de dispersão são apresentados na Figura 52, Figura 53, Figura 54 e na Figura 55.



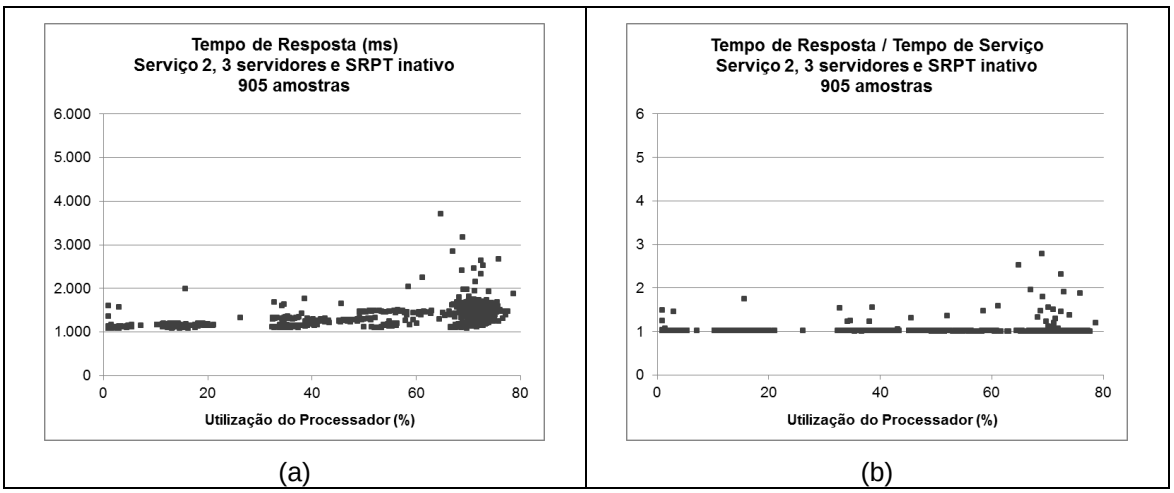
(a) (b)

Figura 52 – Serviço 2: dispersão de valores (2 servidores, SRPT inativo)



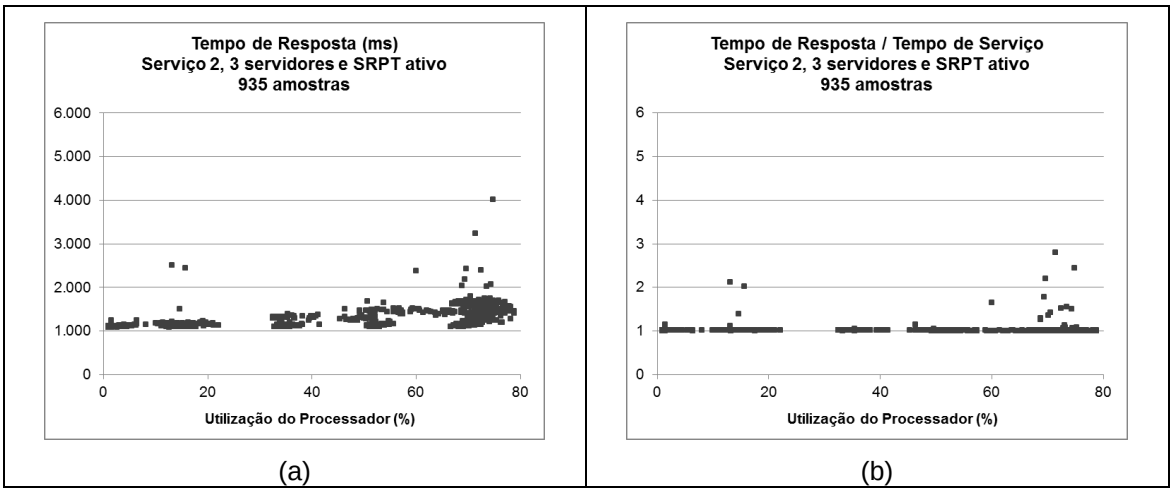
(a) (b)

Figura 53 – Serviço 2: dispersão de valores (2 servidores, SRPT ativo)



(a) (b)

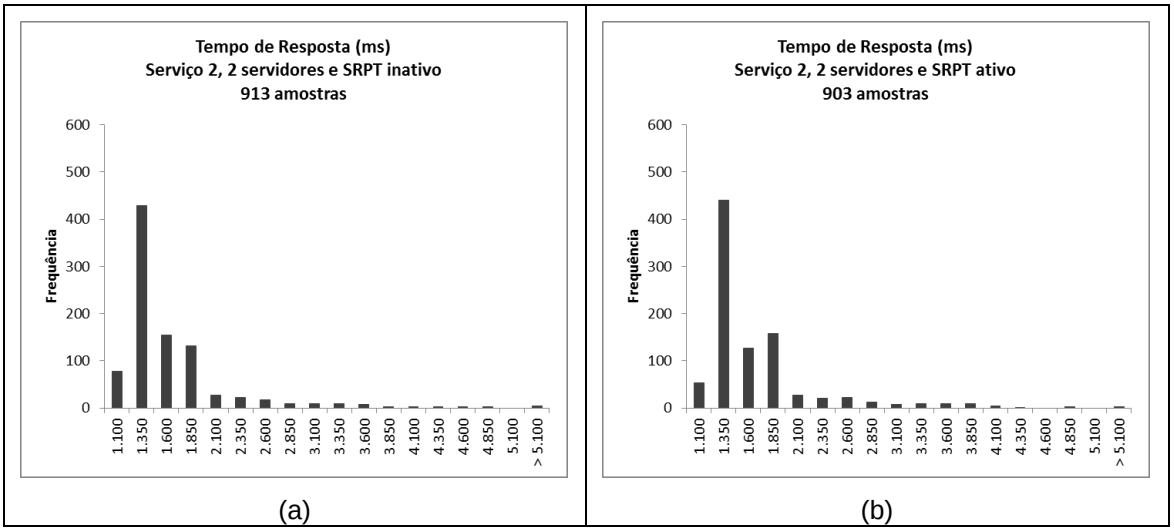
Figura 54 – Serviço 2: dispersão de valores (3 servidores, SRPT inativo)



(a) (b)

Figura 55 – Serviço 2: dispersão de valores (3 servidores, SRPT ativo)

Os histogramas de dispersão, apresentados na Figura 56 (a e b) e na Figura 57 (a e b) permitem concluir que a introdução de novos servidores contribui significativamente para o determinismo nos tempos de resposta, uma vez que é possível observar a concentração de valores em menos classes de distribuição. Porém o escalonamento SRPT não apresenta resultados significativamente melhores do que o escalonamento por ordem de chegada das requisições.



(a) (b)

Figura 56 – Serviço 2: histogramas do tempo de resposta (2 servidores)

É possível observar, ainda na Figura 57, uma pequena redução da frequência de tempos de resposta na primeira classe de distribuição (1.500ms). Essa redução, considerando-se apenas o determinismo dos tempos de resposta, não é significativa, uma vez que mesmo sem o escalonamento SRPT a distribuição dos tempos de resposta ocorre em duas classes principais.

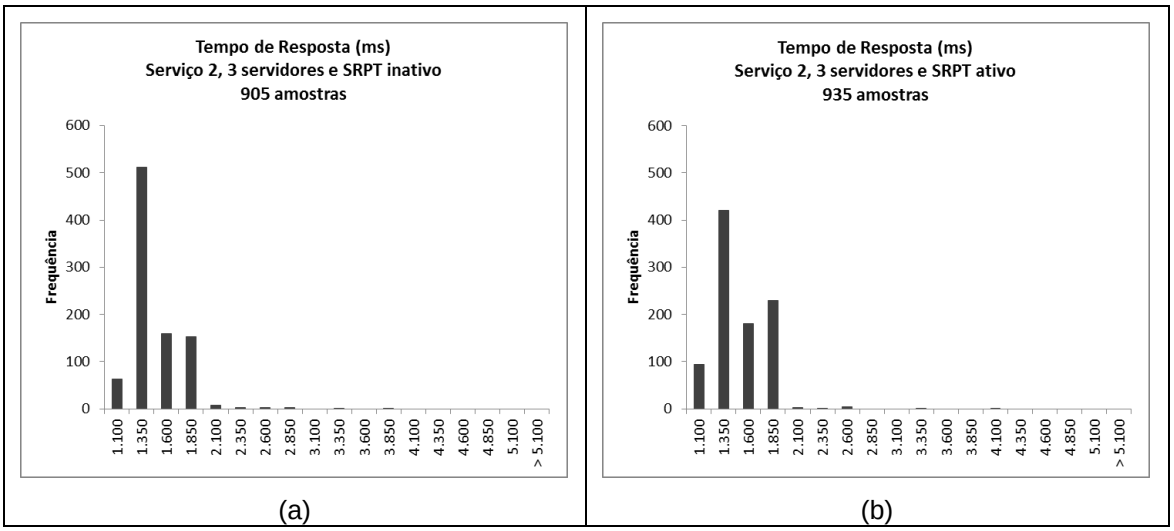


Figura 57 – Serviço 2: histogramas do tempo de resposta (3 servidores)

7.2.3. Resultados das simulações para o Serviço 3

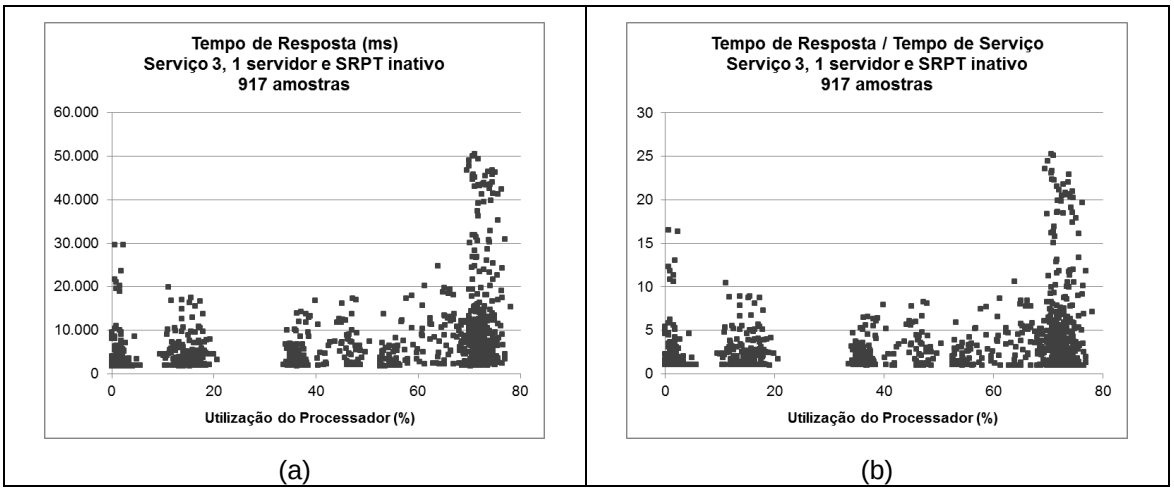
Os resultados obtidos para os tempos de resposta do Serviço 3 são, em linhas gerais, aderentes aos resultados obtidos para os tempos de resposta dos serviços 1 e 2.

Quando utilizado somente 1 servidor para processamento, o escalonamento SRPT contribui de forma significativa para redução do tempo de resposta, porém, nesse caso, em uma escala menor.

Considerando-se a redução nos tempos máximos de resposta obtida para o Serviço 1, a introdução do escalonamento SRPT contribuiu com uma redução de até 9 vezes no tempo máximo de resposta. Comparando a dispersão de valores apresentada na Figura 58 (a e b) com a dispersão de valores apresentada na Figura 59 (a e b), pode observar reduções que variam entre 3 e 4 vezes.

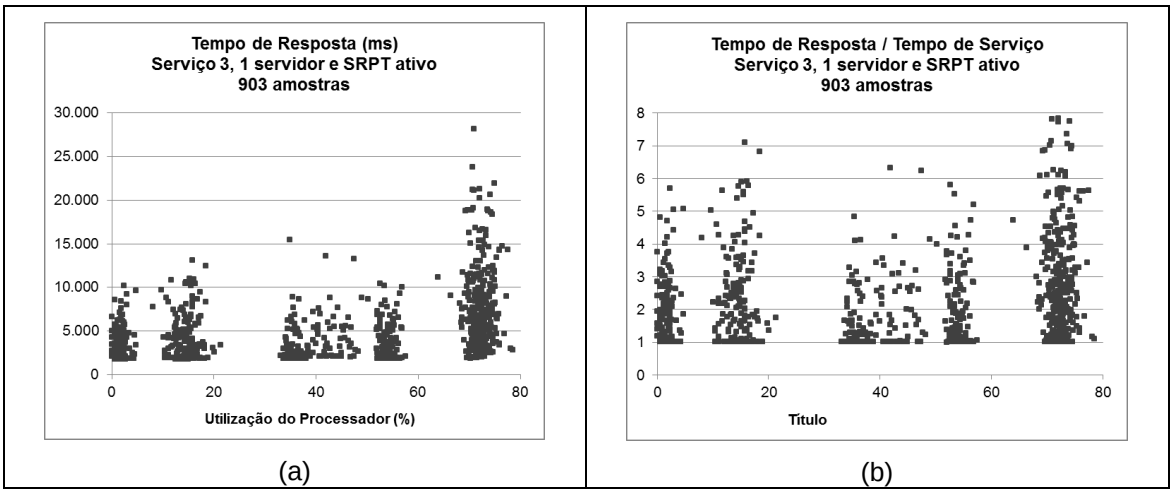
É importante frisar, nesse caso, que o Serviço 3 e o Serviço 4 (cujos resultados das simulações são apresentados na seção seguinte) são os serviços considerados menos prioritários pelo escalonamento SRPT, uma vez que apresentam tempo de serviço maior que os serviços 1 e 2.

As dispersões apresentadas na Figura 60 (a e b), Figura 61 (a e b), Figura 62 (a e b) e Figura 63 (a e b) mantêm o mesmo comportamento apresentado para as simulações anteriores, nas quais o aumento do número de servidores contribui para a diminuição na dispersão de valores (e consequente aumento no determinismo do tempo de resposta).



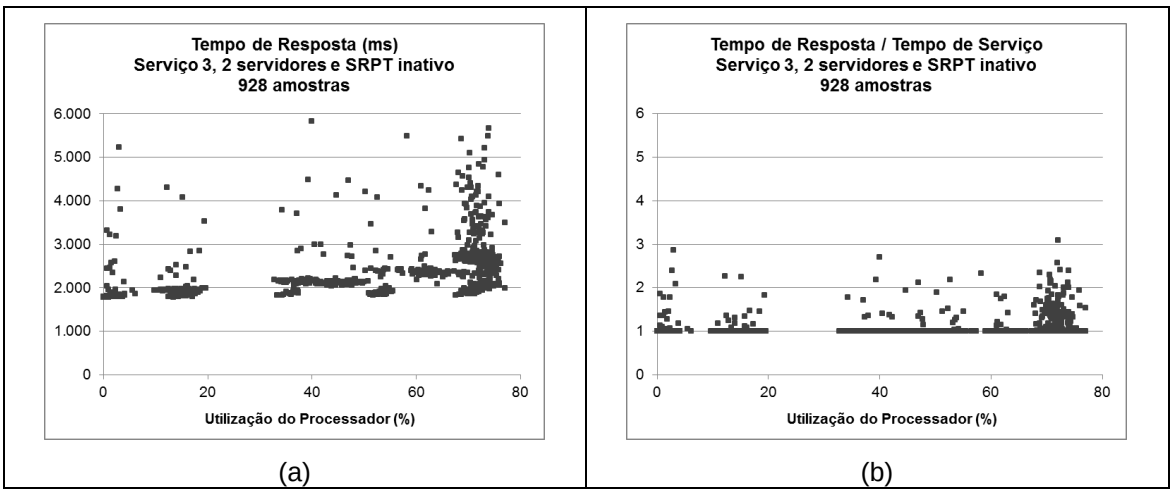
(a) (b)

Figura 58 – Serviço 3: dispersão de valores (1 servidor, SRPT inativo)



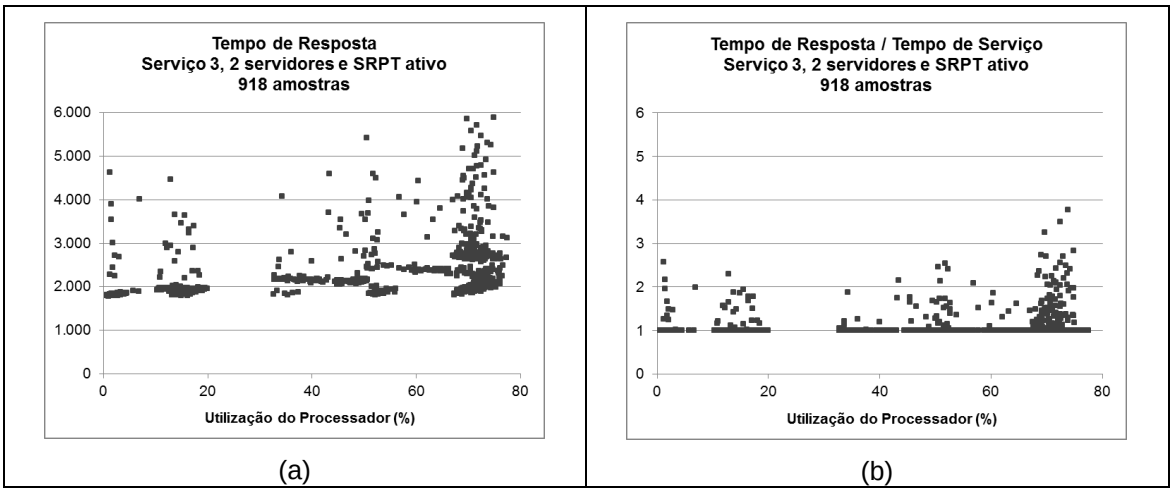
(a) (b)

Figura 59 – Serviço 3: dispersão de valores (1 servidor, SRPT ativo)



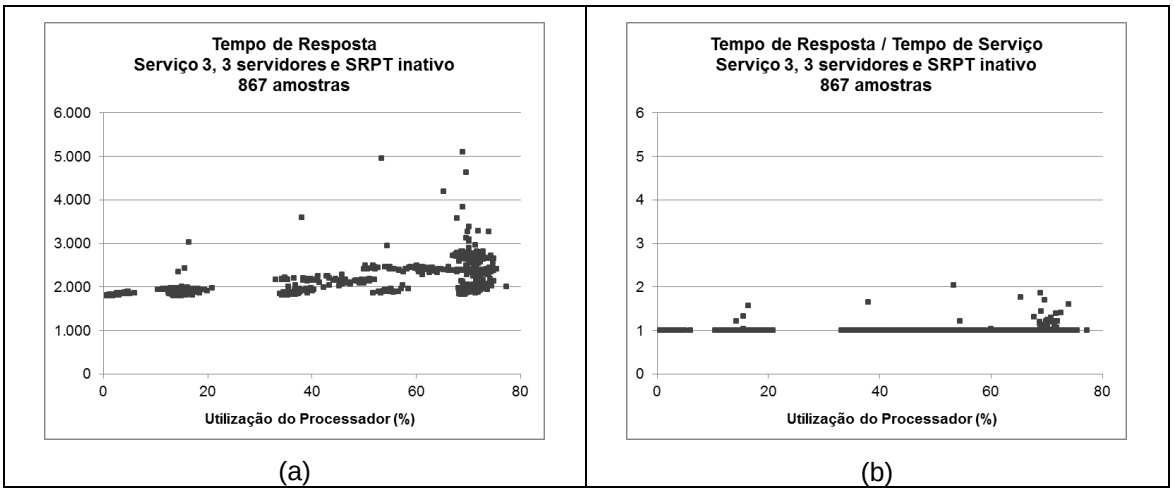
(a) (b)

Figura 60 – Serviço 3: dispersão de valores (2 servidores, SRPT inativo)



(a) (b)

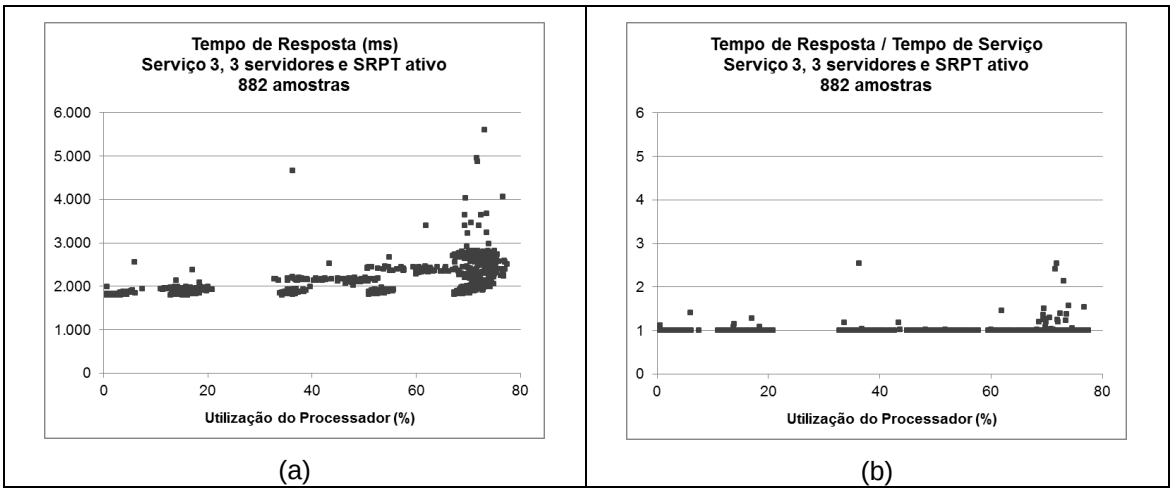
Figura 61 – Serviço 3: dispersão de valores (2 servidores, SRPT ativo)



(a) (b)

Figura 62 – Serviço 3: dispersão de valores (3 servidores, SRPT inativo)

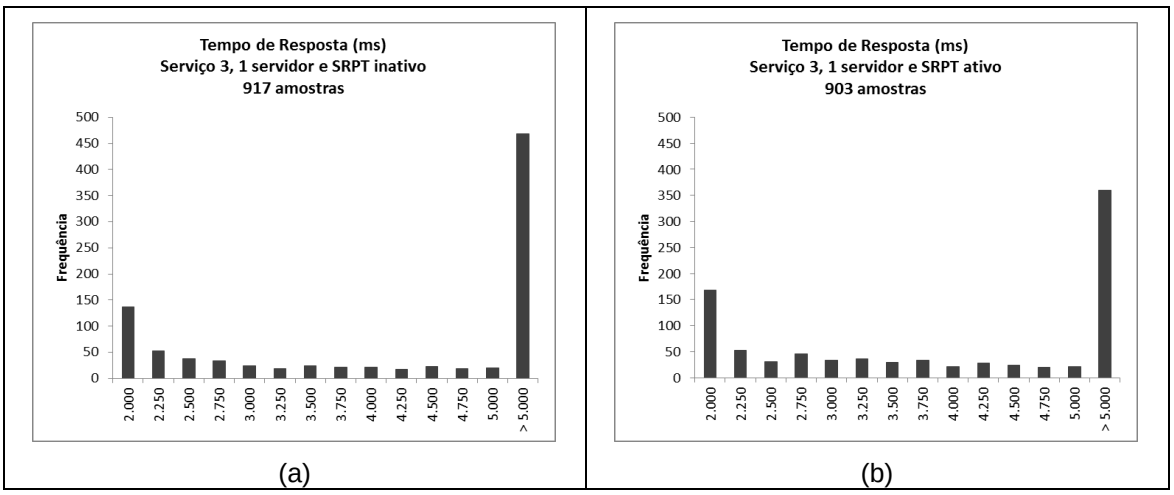
Na Figura 63 é possível observar também a tendência ao valor unitário assumido pela relação tempo de resposta / tempo de serviço, pois, devido ao maior número de servidores disponíveis para execução das consultas, o tempo de espera tende a ser menor, fazendo com que o valor do tempo de resposta se aproxime do valor do tempo de serviço.



(a) (b)

Figura 63 – Serviço 3: dispersão de valores (3 servidores, SRPT ativo)

Os histogramas de tempos de execução apresentados na Figura 64 (a e b), na Figura 65 (a e b) e na Figura 66 (a e b) indicam a diminuição da dispersão do tempo de resposta em diferentes classes com o aumento do número de servidores disponíveis para execução das requisições. De forma análoga, aos resultados obtidos para as simulações anteriores, a contribuição do escalonamento SRPT para o aumento do determinismo não é significativa quando existe mais de um servidor disponível para execução das requisições.



(a) (b)

Figura 64 – Serviço 3: histogramas do tempo de resposta (1 servidor)

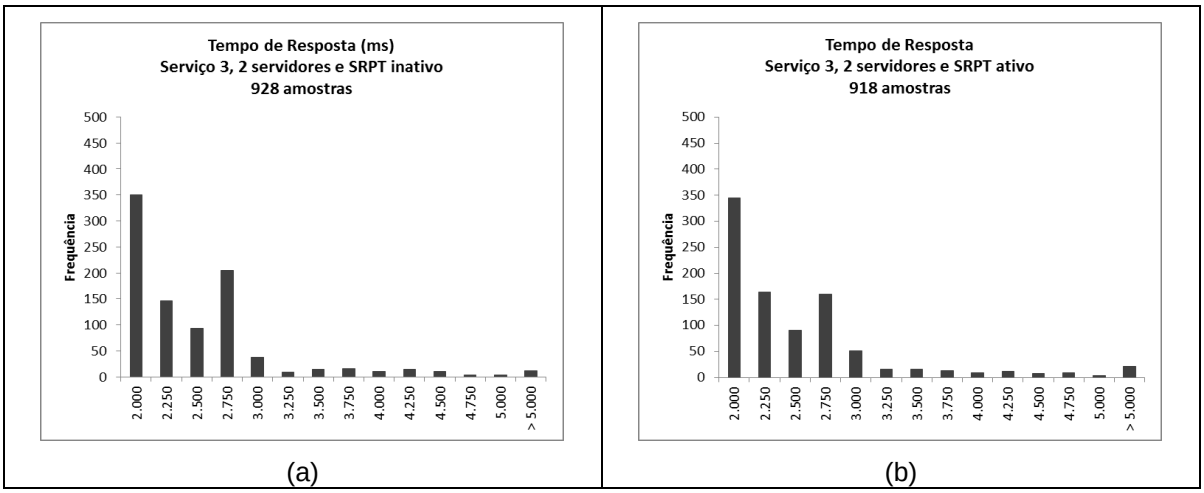


Figura 65 – Serviço 3: histogramas do tempo de resposta (2 servidores)

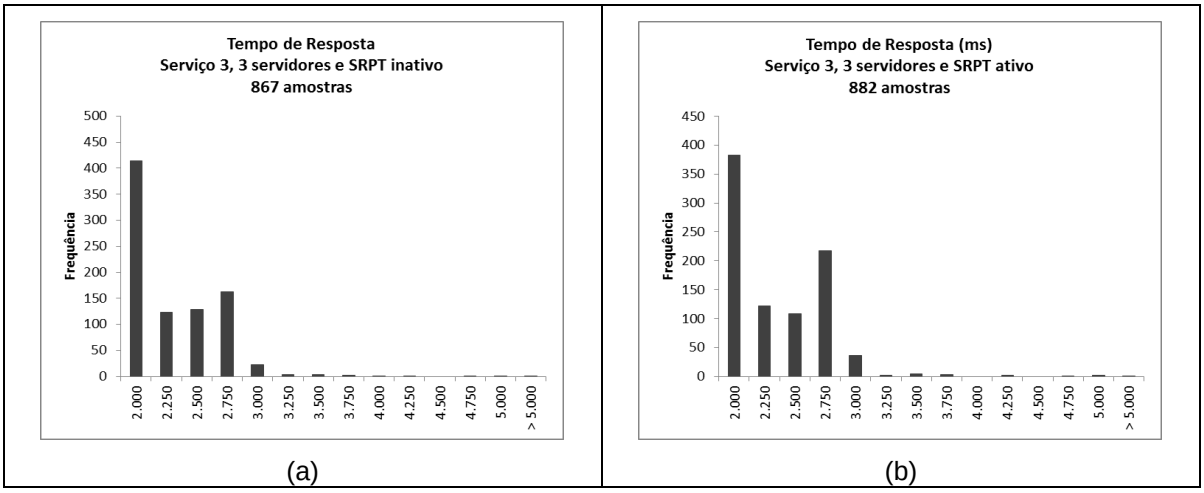


Figura 66 – Serviço 3: histogramas do tempo de resposta (3 servidores)

7.2.4. Resultados das simulações para o Serviço 4

Os resultados obtidos para os tempos de resposta referentes ao Serviço 4 seguem, em linhas gerais, o mesmo padrão dos resultados obtidos para os serviços 1, 2 e 3. Para o Serviço 4, porém, quando a configuração do sistema utiliza apenas 1 servidor sem escalonamento (Figura 67), o tempo de resposta máximo é de 3 a 4 vezes menor que o tempo de reposta máximo (Figura 68) quando o escalonamento SRPT está ativo, comportamento esse inverso do observado para os demais serviços.

Esse fato pode ser explicado pela baixa prioridade de execução do Serviço 4, que é o menos prioritário entre todos os serviços concorrentes.

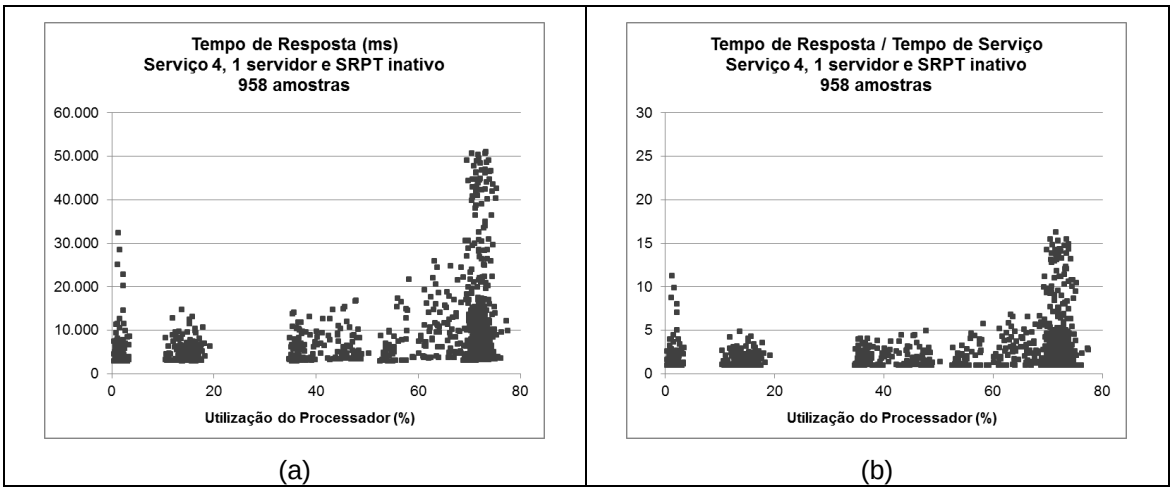


Figura 67 – Serviço 4: dispersão de valores (1 servidor, SRPT inativo)

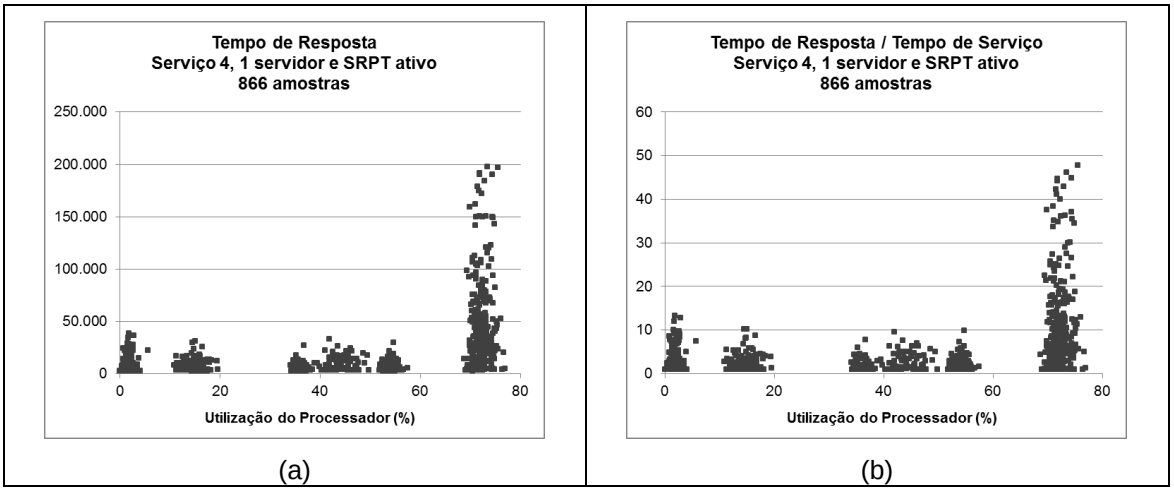
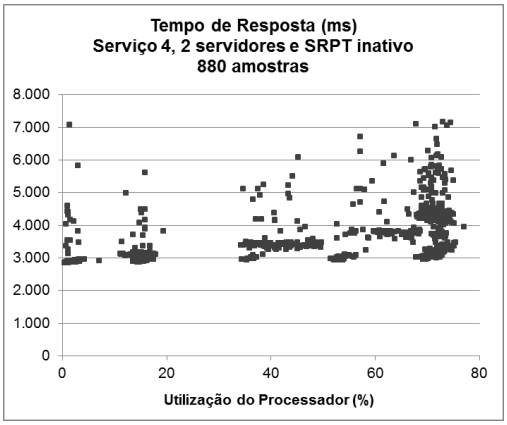


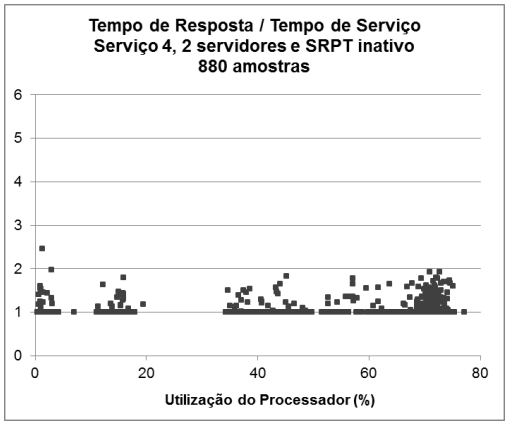
Figura 68 – Serviço 4: dispersão de valores (1 servidor, SRPT ativo)

Nas demais execuções, isto é quando há mais de um servidor disponível para execução das requisições, o comportamento observado repete o padrão observado para os demais serviços: a introdução de servidores adicionais para execução das requisições contribui, de forma significativa, para o aumento no determinismo do tempo de resposta das requisições (concentração dos tempos de execução em um número menor de classes) e, ao mesmo tempo, a introdução do escalonamento SRPT, nesse caso, não contribui para que o resultado seja diferente da utilização do escalonamento por ordem de chegada das requisições.

Os resultados obtidos para as configurações de 2 ou 3 servidores, com o escalonamento SRPT ativo (a) ou inativo (b) são apresentados na Figura 69 (a e b), Figura 70 (a e b), Figura 71 (a e b) e Figura 72 (a e b).

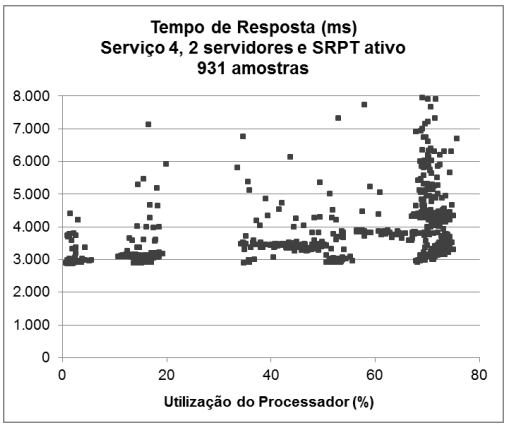


(a)

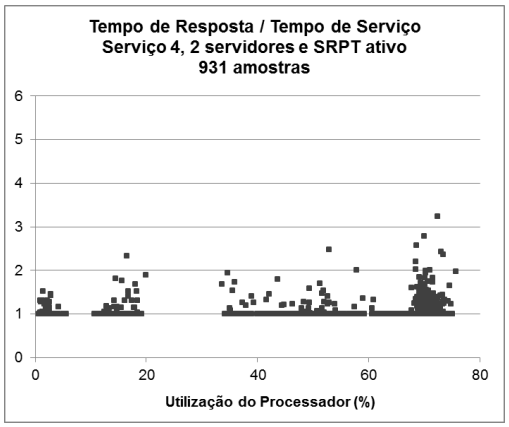


(b)

Figura 69 – Serviço 4: dispersão de valores (2 servidores, SRPT inativo)

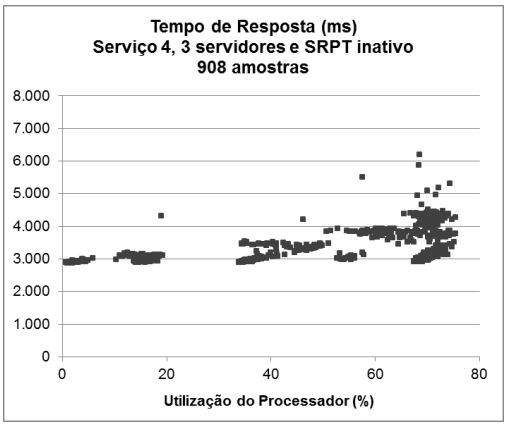


(a)

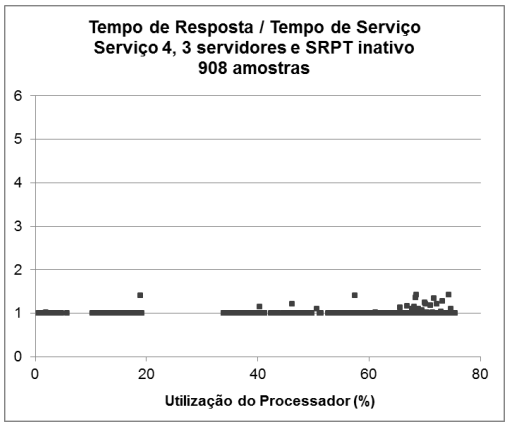


(b)

Figura 70 – Serviço 4: dispersão de valores (2 servidores, SRPT ativo)



(a)



(b)

Figura 71 – Serviço 4: dispersão de valores (3 servidores, SRPT inativo)

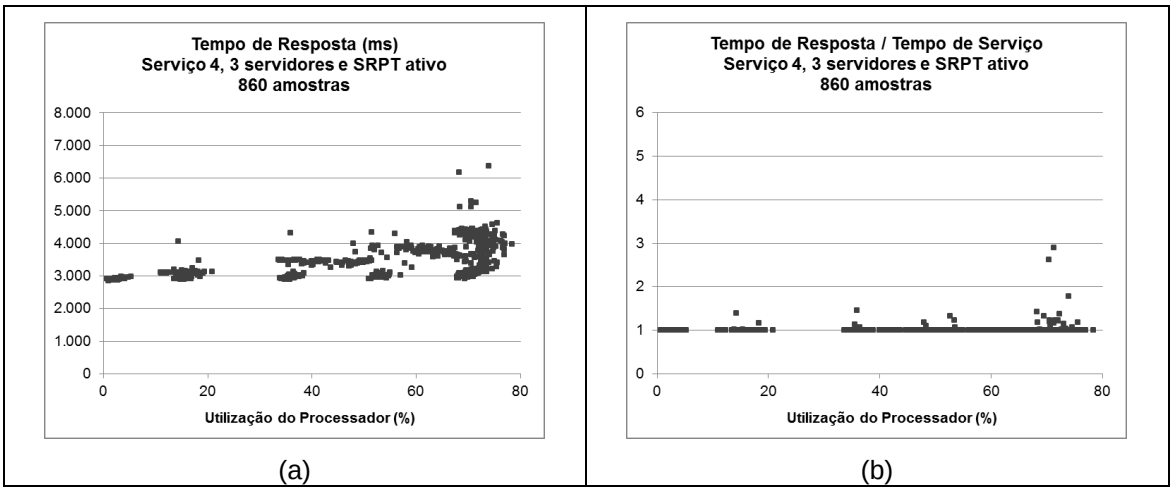


Figura 72 – Serviço 4: dispersão de valores (3 servidores, SRPT ativo)

Os histogramas referentes aos tempos de resposta para as três configurações simuladas com 1, 2 ou 3 servidores são apresentados, respectivamente, na Figura 73 (a e b), na Figura 74 (a e b) e na Figura 75 (a e b).

Observa-se na Figura 73b, um aumento na frequência de tempos de resposta maiores que 7.000ms com a ativação do escalonamento SRPT para 1 único servidor. Esse comportamento é o inverso do comportamento obtido nas simulações para os serviços 1, 2 e 3, no qual é possível notar uma diminuição da frequência de tempos de reposta nas classes referentes aos tempos de resposta muito acima da média do tempo de serviço.

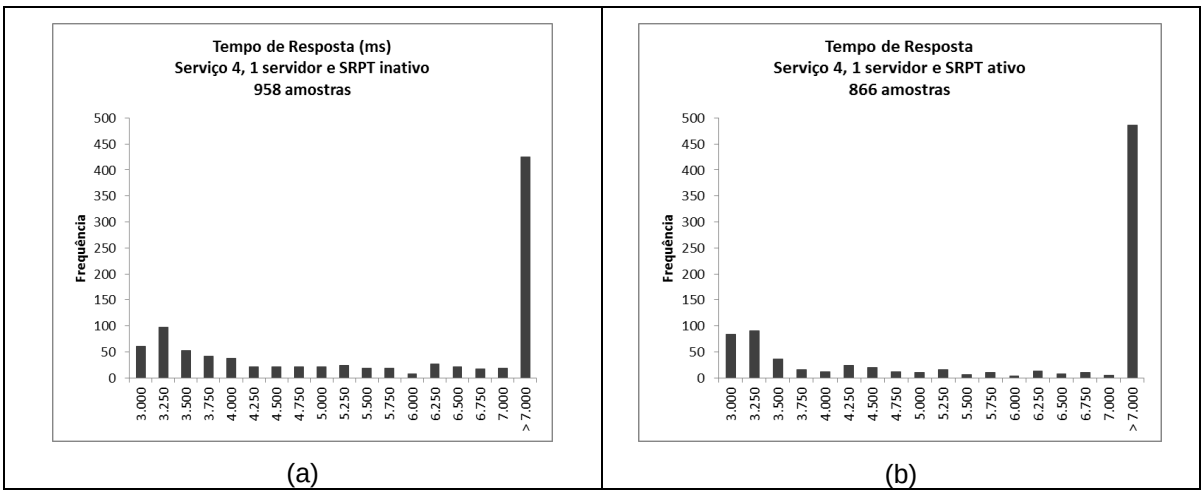


Figura 73 – Serviço 4: histogramas do tempo de resposta (1 servidor)

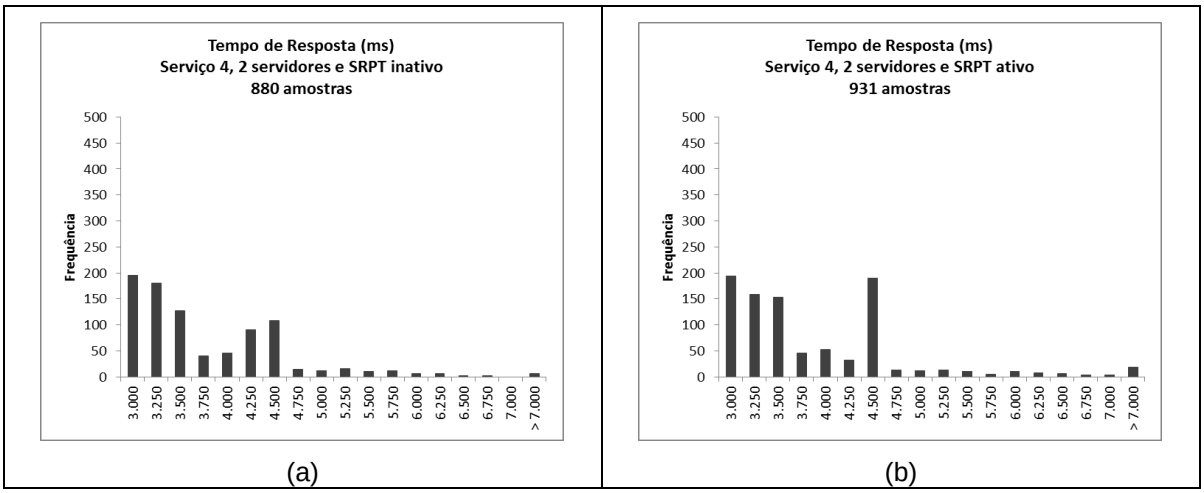


Figura 74 – Serviço 4: histogramas do tempo de resposta (2 servidores)

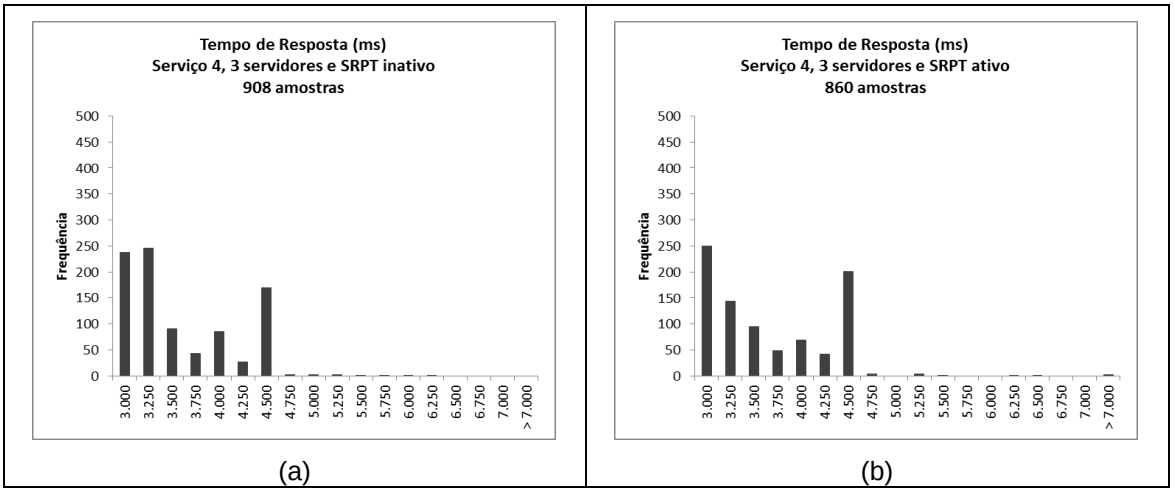


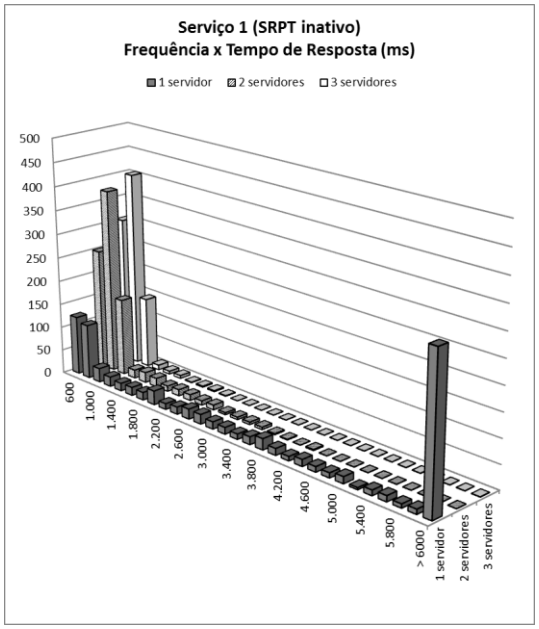
Figura 75 – Serviço 4: histogramas do tempo de resposta (3 servidores)

De forma semelhante aos demais serviços, também pode ser observado na Figura 73b, na Figura 74b e na Figura 75b que a ativação do escalonamento SRPT não contribui para o aumento no determinismo do tempo de resposta, mas também não compromete os resultados obtidos, uma vez que a ativação do escalonamento SRPT não distribui em um maior número de classes os tempos de respostas obtidos.

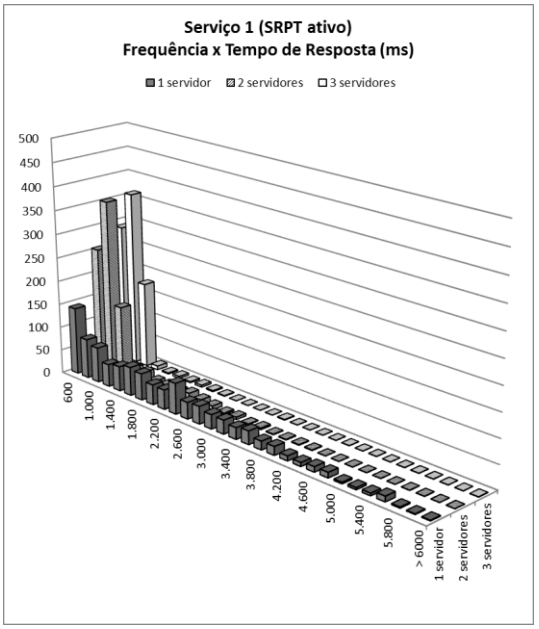
7.3. Resultados Consolidados

Nesta seção são apresentados os resultados consolidados para os tempos de resposta das requisições efetuadas para os quatro serviços desenvolvidos, parametrizados pelo escalonamento SRPT inativo (índice a das figuras) ou ativo (índice b das figuras) e pelo número de servidores utilizados no atendimento às requisições. Os resultados obtidos são apresentados na Figura 76 (a e b) - Serviço

1, Figura 77 (a e b) - Serviço 2, Figura 78 (a e b) - Serviço 3 e Figura 79 (a e b) - Serviço 4.

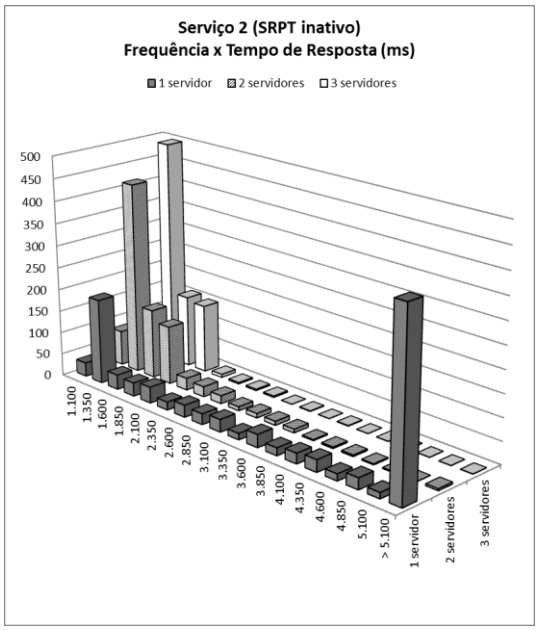


(a)

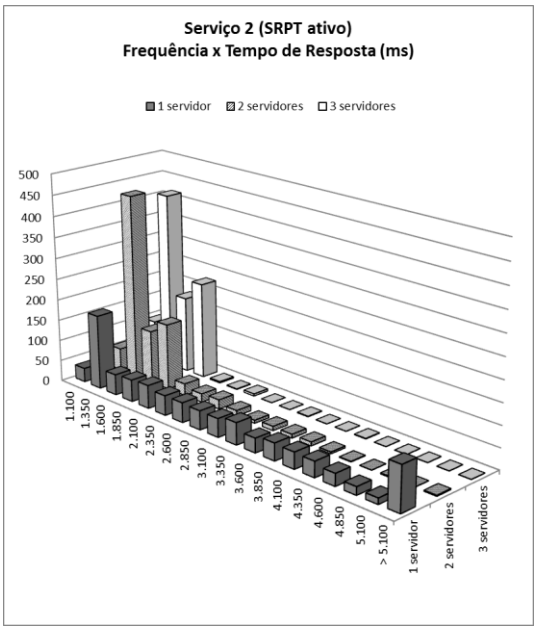


(b)

Figura 76 – Serviço 1: histograma consolidado

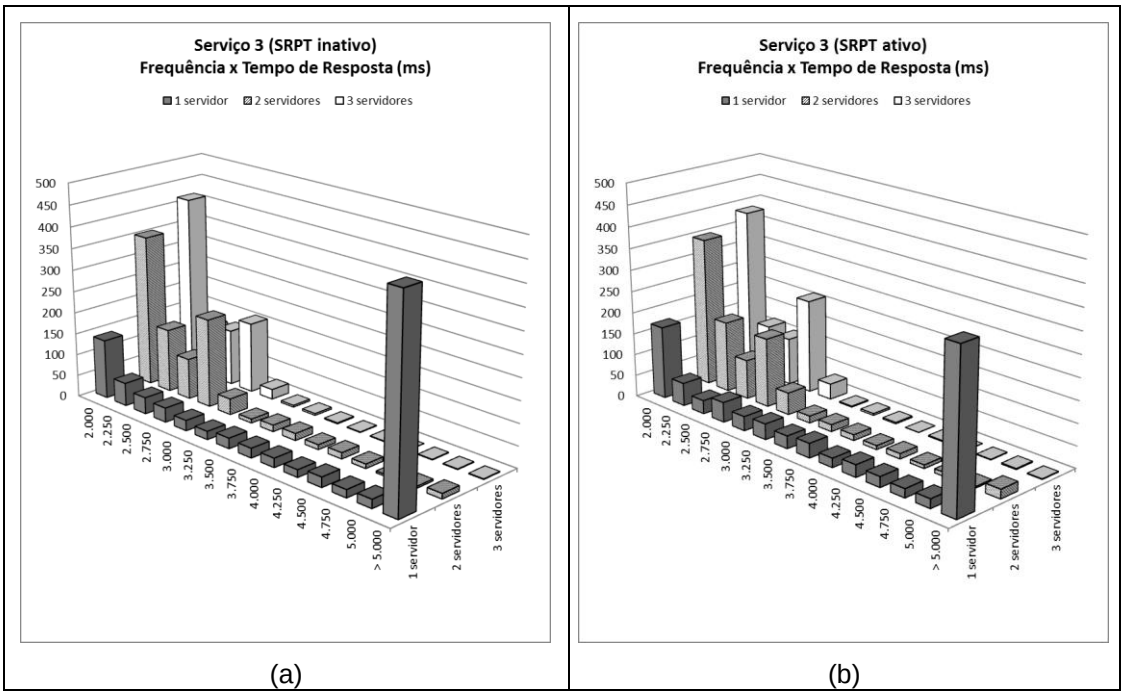


(a)



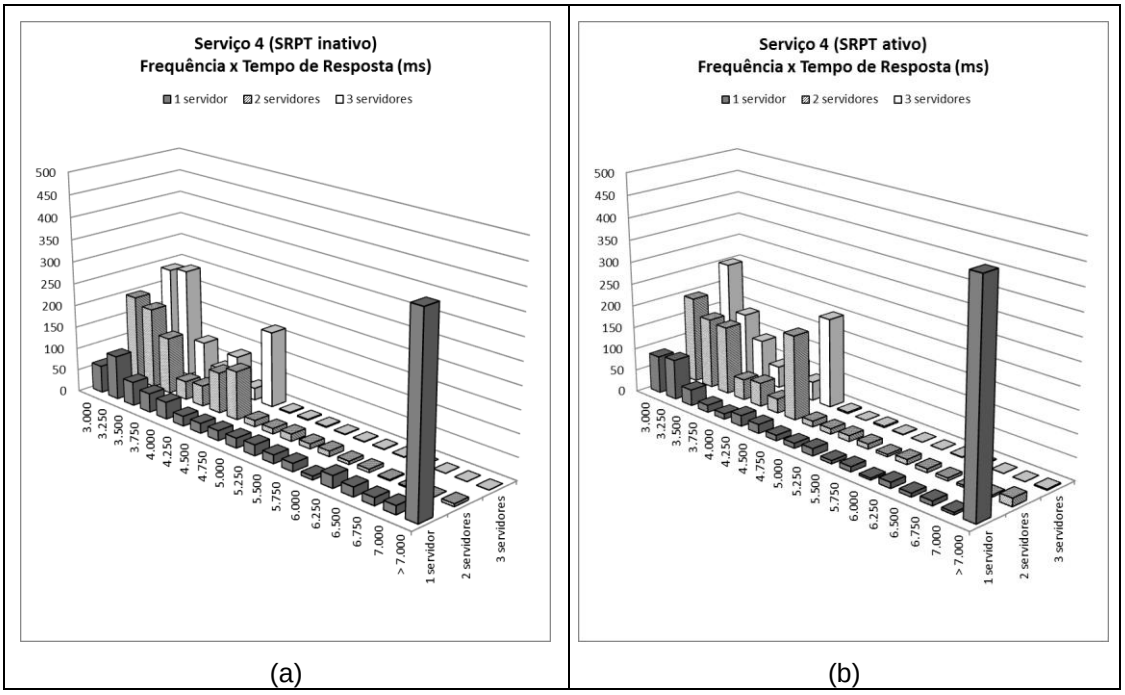
(b)

Figura 77 – Serviço 2: histograma consolidado



(a) (b)

Figura 78 – Serviço 3: histograma consolidado



(a) (b)

Figura 79 – Serviço 4: histograma consolidado

Em linhas gerais, é possível observar que:

1) O escalonamento SRPT apresenta resultados significativos para o aumento do determinismo nos tempos de resposta quando é utilizado um único servidor para atendimento às requisições e o tempo médio de serviço é menor

que o tempo médio entre requisições (parâmetro λ da distribuição exponencial);

2) A utilização de mais de um servidor para atendimento às requisições tem impacto significativo no aumento do determinismo dos tempos de resposta das requisições efetuadas.

   Nesse caso, o escalonamento de requisições por ordem de chegada apresenta resultados iguais ou melhores que o escalonamento SRPT utilizado;

3) Para os tipos de serviços criados, quando se utiliza a configuração com 3 servidores e escalonamento por ordem de chegada (SRPT inativo), os tempos máximos de resposta estão, em sua grande maioria, abaixo de 2 vezes o tempo médio de serviço.

   Embora essa não seja uma regra geral que possa ser aplicada a qualquer caso, para qualquer tipo de serviço, permite estimar, de forma quantitativa, o nível de melhoria no determinismo do tempo de resposta com a utilização de múltiplos servidores nas simulações efetuadas e, ainda, se aproxima do valor teórico para a razão entre tempo de resposta e tempo de serviço apresentado na teoria (Capítulo 3.1, página 42, Figura 11). No modelo teórico (sistema M / M / 1 – fila única e múltiplos servidores), em que a taxa de processamento seja menor que 70%, a relação entre o tempo de resposta e o tempo de serviço é, aproximadamente, menor ou igual a dois.

   Os resultados obtidos estão apresentados na Tabela 7.

Tabela 7- Comparação de tempos de resposta para 3 servidores.

Serviço	Tempo médio de serviço x 2 (ms)	Classe limite	Amostras acumuladas	Proporção
1	1.088,8	1.200	868	868 / 887 = 97,9%
2	2.168,6	2.100	894	894 / 905 = 98,8%
3	3.596,0	3.500	860	860 / 867 = 99,2%
4	5,761,0	5.750	906	906 / 908 = 99,8%

7.4. Considerações Finais deste Capítulo

Este capítulo apresentou os resultados obtidos para as simulações efetuadas com o sistema construído, baseado na arquitetura proposta. Nos resultados apresentados para as diversas configurações foi possível identificar contribuições do modelo proposto para a melhoria do determinismo nos tempos de resposta. Parte das soluções apresentadas no modelo proposto, particularmente o escalonamento SRPT, não apresentou resultados satisfatórios, porém também não contribuíram negativamente para o resultado geral obtido.

8. CONSIDERAÇÕES FINAIS

Este capítulo apresenta as conclusões e contribuições deste trabalho, bem como sugestões para trabalhos futuros.

8.1. Contribuições

A principal contribuição desse trabalho é a proposta de uma arquitetura derivada do padrão arquitetural SOA (Arquitetura Orientada a Serviço), baseada em softwares COTS, a ser empregada no desenvolvimento de sistemas com restrições temporais.

Para a proposta apresentada, o ambiente de testes desenvolvido teve a finalidade específica de realizar simulações relativas à proposta apresentada, de forma a possibilitar a observação do comportamento dos tempos de resposta de requisições aos serviços fornecidos pelo ambiente criado. Os resultados obtidos apresentados no capítulo 7 permitem concluir que o modelo proposto contribuiu positivamente para o determinismo nos tempos de resposta das requisições efetuadas, validando a arquitetura proposta.

Essa arquitetura não se aplica ao desenvolvimento de sistemas de tempo real onde o não atendimento a essas restrições possa ter consequências catastróficas para o ambiente em que é executado (danos materiais ou a pessoas). A proposta desenvolvida contemplou uma arquitetura a ser aplicada no desenvolvimento de sistemas com restrições temporais, cujo desempenho possa ser degradado, porém não destruído na ocorrência de uma falha no atendimento a uma ou mais especificações de tempos de resposta, ou, de acordo com a classificação proposta por LAPLANTE (1992), um sistema de tempo real do tipo *soft*.

Os requisitos considerados na arquitetura proposta especificam ainda uma restrição no seu campo de aplicação, a qual se refere ao determinismo da latência na rede de comunicações utilizada para troca de mensagens. Dessa forma, na arquitetura proposta (bem como nos testes efetuados), considerou-se a latência de rede como sendo constante.

Em aplicações comerciais, essa restrição corresponde à utilização de sistemas desenvolvidos a partir da arquitetura proposta em redes de comunicação que possam ser gerenciadas e controladas, isto é, onde seja possível gerenciar a

latência de rede (MARAU, ALMEIDA e PEDREIRAS, 2006), (SANTOS, VIERIA, et al., 2010).

Conforme apresentado no capítulo 5, a arquitetura proposta (baseada em serviços *Web*), não utiliza mecanismos dinâmicos de descoberta (O'BRIEN, MERSON e BASS, 2005), (TSAI, LEE, et al., 2006). Além disso, a introdução de serviços redundantes e o escalonamento das requisições a esses serviços, utilizando o escalonamento SRPT – *Shortest Remaining Processing Time* (BIERSACK, SCHROEDER e URVOY-KELLER, 2007), completam a proposta deste estudo.

Nas simulações efetuadas, foram utilizados três parâmetros principais para avaliação dos resultados obtidos: taxa de processamento do sistema em que o serviço é executado (impondo uma taxa de processamento entre 0 e 80%), o número de servidores disponíveis para execução das requisições (entre 1 e 3 servidores) e a utilização ou não de escalonamento (sem utilização escalonamento, escalonamento por ordem de chegada ou escalonamento SRPT, com prioridade por ordem de chegada em uma mesma fila).

Conforme apresentado no capítulo 7, é possível observar que a contribuição para o determinismo do tempo de resposta das requisições é significativa com a utilização de múltiplos servidores para atendimento das requisições, mesmo para diferentes taxas de processamento no sistema que hospeda o serviço.

Os resultados obtidos para as simulações com múltiplos servidores estão de acordo com o comportamento previsto pela modelagem do sistema utilizando-se teoria das filas, conforme apresentado no capítulo 3. É possível verificar, por meio dos resultados apresentados no capítulo 7, que há uma redução significativa na dispersão dos valores de tempo de resposta das requisições, mesmo com diferentes taxas de processamento nos sistemas onde as requisições são executadas.

Ainda considerando a utilização de múltiplos servidores, não se observa contribuição signficativa na utilização do escalonamento SRPT em relação à utilização do escalonamento por ordem de chegada das requisições.

A utilização do escalonamento SRPT permite observar uma contribuição significativa para o determinismo do tempo de execução das transações ao utilizar um único servidor para atendimento às requisições, porém para os serviços com menor tempo de execução (mais prioritários).

8.2. Trabalhos Futuros

Algumas condições de contorno consideradas na arquitetura permitem que sejam abertas novas propostas de estudo, como, por exemplo, a observação do comportamento da arquitetura proposta ao se utilizar redes locais como meio de troca de mensagens. Nesse caso, poderia ser observada a latência de rede por meio do monitoramento do tráfego de pacotes tempos de resposta do sistema.

Considerando uma das condições de contorno utilizadas na implementação, foi assumido que os tempos de execução dos serviços (tempos de serviço) são aproximadamente constantes (apresentam um tempo médio com baixa variância). Essa premissa se deve ao fato de se utilizar o tempo de serviço como variável controlada, tornando possível a monitoração do seu comportamento em função das taxas de utilização dos servidores em que os serviços são hospedados.

Em sistemas reais, dependendo do tipo de aplicação, essa premissa pode não ser válida, porém, podem ser estimados tempos máximos de execução e, ainda, determinar a distribuição dos tempos de execução e a frequência de um determinado serviço no tempo.

A análise desse comportamento poderia, então, ser utilizada no estabelecimento de filas de prioridade dinâmicas (algoritmo SRPT), considerando-se diferentes períodos de utilização do sistema.

8.3. Conclusões

É importante ressaltar que, embora a arquitetura proposta tenha sido baseada na adaptação de uma Arquitetura Orientada a Serviços existente (particularmente, os Serviços *Web*, com grande utilização no desenvolvimento de sistemas comerciais) e também na utilização de softwares COTS, os resultados obtidos para as simulações efetuadas não foram obtidos em sistemas comerciais reais, uma vez que todos os sistemas utilizados na simulação foram desenvolvidos especificamente para esse fim.

Dessa forma, além da contribuição referente à arquitetura proposta, espera-se que todo o esforço empregado no desenvolvimento dos sistemas utilizados nas simulações, na sua execução e na análise dos resultados obtidos possa ser utilizado como ponto de partida para estudos futuros e aplicação em sistemas comerciais.

9. REFERÊNCIAS BIBLIOGRÁFICAS

ANTONIOL, G. et al. Assessing Staffing Needs for a Software Maintenance Project through Queuing Simulation. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v. 30, n. 1, p. 43-58, 2004.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice, Second Edition**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

BENZ, B. **XML Programming Bible**. New York, NY, USA: John Wiley & Sons, Inc., 2003.

BIERSACK, E. W.; SCHROEDER, B.; URVOY-KELLER, G. Scheduling in practice. **SIGMETRICS Perform. Eval. Rev.**, New York, NY, USA, v. 34, n. 4, p. 21-28, 2007.

BROWN, A.; JOHNSTON, S.; KELLY, K. **Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications**. Rational Software Corporation. [S.l.]. 2002.

CHEN, H.-P.; ZHANG, C. A Queueing-Theory-Based Fault Detection Mechanism for SOA-Based Applications. **E-Commerce Technology, IEEE International Conference on, and Enterprise Computing, E-Commerce, and E-Services, IEEE International Conference on**, Los Alamitos, CA, USA, v. 0, p. 157-166, 2007.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos:** conceitos e projeto. Porto Alegre, Brasil: Bookman, 2007.

CUOMO, G. **IBM SOA "on the edge"**. SIGMOD ’05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM. 2005. p. 840-843.

ECKERT, J. et al. **Queuing-based capacity planning approach for Web service workflows using optimization algorithms**. Digital Ecosystems and Technologies, 2008. DEST 2008. 2nd IEEE International Conference on. [S.l.]: [s.n.]. 29 26 2008. p. 313-318.

GARTNER GROUP. Gartner Says SOA Will Be Used in More Than 50 Percent of New Mission-Critical Operational Applications and Business Processes Designed in 2007. **Gartner Newsroom**, 2007. Disponivel em: <http://www.gartner.com/it/page.jsp?id=503864>. Acesso em: 13 jul. 2010.

GARTNER GROUP. Gartner Says the Number of Organizations Planning to Adopt SOA for the First Time Is Falling Dramatically. **Gartner Newsroom**, 2008. Disponivel em: <http://www.gartner.com/it/page.jsp?id=790717>. Acesso em: 13 jul. 2010.

HARCHOL-BALTER, M. et al. Size-based scheduling to improve web performance. **ACM Trans. Comput. Syst.**, New York, NY, USA, v. 21, n. 2, p. 207-233, 2003.

JOHNSON, B. W. **Design & analysis of fault tolerant digital systems**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

KENDALL, D. G. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. **The Annals of Mathematical Statistics**, v. 24, n. 3, p. 338-354, 1953.

KISZKA, J.; WAGNER, B. **RTnet - a flexible hard real-time networking framework**. [S.l.]: [s.n.]. sep. 2005. p. 8-456.

LALA, P. K. **Self-checking and fault-tolerant digital design**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

LAPLANTE, P. A. **Real-Time Systems Design and Analysis:** An Engineer's Handbook. Piscataway, NJ, USA: IEEE Press, 1992.

LEWIS, G. A. et al. Common Misconceptions about Service-Oriented Architecture. **Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'07)**, March 2 26 2007 2007. 123-130.

LI, X. et al. **SOArMetrics:** A Toolkit for Testing and Evaluating SOA Middleware. SERVICES '09: Proceedings of the 2009 Congress on Services - I. Washington, DC, USA: IEEE Computer Society. 2009. p. 163-170.

LIU, H. H. **Software Performance and Scalability:** A Quantitative Approach. [S.l.]: Wiley Publishing, 2009.

LOPES, M. M. **Aplicação de Mineração de Dados na identificação de padrões de falha em Sistemas Gerenciadores de Bancos de Dados Relacionais**. Universidade de São Paulo. São Paulo. 2005.

MARAU, R.; ALMEIDA, L.; PEDREIRAS, P. **Enhancing real-time communication over cots ethernet switches**. [S.l.]: [s.n.]. jun. 2006. p. 295-302.

MAURIZIO, A. et al. **Service Oriented Architecture:** Challenges for Business and Academia. HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences. Washington, DC, USA: IEEE Computer Society. 2008. p. 315.

O'BRIEN, L.; MERSON, P.; BASS, L. **Quality Attributes for Service-Oriented Architectures**. SEI - Software Engineering Institute. Carnegie Mellon University. 2005. (CMU/SEI-2005-TN-014).

OASIS. UDDI Version 2.03 Data Structure Reference. **http:** //www.oasis-open.org/, 12 Julho 2002. Disponivel em: <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm>. Acesso em: 12 out. 2009.

OASIS. Web Services Reliable Messaging TC WS-Reliability 1.1. **OASIS**, 2004. Disponivel em: <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm>. Acesso em: 07 ago. 2010.

OASIS. Call for Participation: OASIS SOA-RM TC. **The Cover Pages**, 2005. Disponivel em: <http://xml.coverpages.org/SOA-RM-CFP.html>. Acesso em: 26 jul. 2010.

OASIS. Modelo de Referência para Arquitetura Orientada a Serviço 1.0. **OASIS**, 19 Julho 2006. Disponivel em: <http://www.pcs.usp.br/~pcs5002/oasis/soa-rm-csbr.pdf>. Acesso em: 12 jul. 2010.

OASIS. OASIS Web Services Business Process Execution Language TC. **OASIS**, 2010. Disponivel em: <http://www.oasis-open.org/committees/wsbpel/faq.php>. Acesso em: 26 jul. 2010.

PRÜTER, S. et al. Applicability of Web Service Technologies to Reach Real Time Capabilities. **11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)**, Washington, DC, USA, 2008. 229-233.

RTNET. RTNet - Hard Real-Time Networking for Real-Time Linux. **RTNet**, 2010. Disponivel em: <http://www.rtnet.org/doc.html>. Acesso em: 22 ago. 2010.

SANTOS, R. et al. **Flexible, Efficient and Robust Real-Time Communication with Server-based Ethernet Switching**. 8th IEEE International Workshop on Factory Communication Systems (WFCS'10). [S.l.]: [s.n.]. May 2010.

SEI. Software Engineering Institute. **Migrating Legacy Systems to SOA Environments - eLearning**, 2007. Disponivel em: <http://www.sei.cmu.edu/training/v06.cfm>. Acesso em: 12 jul. 2010.

SEI. **SMART: Analyzing the Reuse Potential of Legacy Components in a Service-Oriented Architecture Environment**. SEI - Software Engineering Institute. [S.l.], p. 37. 2008. (CMU/SEI-2008-TN-008).

TANENBAUM, A. S. Sistemas Operacionais Modernos. In: TANENBAUM, A. S. **Sistemas Operacionais Modernos**. São Paulo: Prentice Hall, 2003.

THIELE, L.; WILHELM, R. Design for Timing Predictability. **Real-Time Systems**, Netherlands, v. 28, n. 2-3, p. 157-177, 2004.

TIAN, Q. et al. **A Novel Dynamic Priority Scheduling Algorithm of Process Engine in SOA**. Web Services, 2009. ICWS 2009. IEEE International Conference on. [S.l.]: [s.n.]. 10 jun. 2009. p. 711-718.

TSAI, W. T. et al. RTSOA: Real-Time Service-Oriented Architecture. **Service-Oriented System Engineering, 2006. SOSE '06. Second IEEE International Workshop**, Oct. 2006. 49-56.

W3C. Simple Object Access Protocol (SOAP) 1.1. **http:** //www.w3.org, 2000. Disponivel em: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>. Acesso em: 29 jul. 2010.

W3C. SOAP Version 1.2 Part 1: Messaging Framework. **http:** //www.w3.org, 24 Junho 2003. Disponivel em: <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>. Acesso em: 12 out. 2009.

W3C. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. **http:** //www.w3.org, 10 Novembro 2003. Disponivel em: <http://www.w3.org/TR/2003/WD-wsdl20-20031110/>. Acesso em: 12 out. 2009.

W3C. Web Services Architecture. **http:** //www.w3.org, 11 Fevereiro 2004. Disponivel em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 11 Outubro 2010.

WS-I. Basic Profile Version 2.0. **The Web Services-Interoperability Organization**, 2010. Disponivel em: <http://ws-i.org/profiles/BasicProfile-2.0-WGD.html>. Acesso em: 26 jul. 2010.