Definição de Requisitos de QoS em Coreografias de Serviços Web

Alfonso Phocco Diaz[1], **Daniel M. Batista** [1]

[1] Departamento de Ciência da Computação
Universidade de São Paulo (USP) – São Paulo, SP – Brazil

{alfonso7,batista}@ime.usp.br

Abstract. *Service choreography allows the composition of services in a collaborative way, because of global description and decentralized coordination using P2P interactions between participants. However, since infrastructures and implementations aren't mature enough to enact choreographies, the evaluation of QoS requirements and composition behaviour is a difficult task.*
In this work, we propose an approach for QoS requirements between participant services involved into a choreography taking into account QoS composition and infrastructure aspects. We developed a choreography simulator in order to simulate the enactment of choreographies. The establishment of QoS constraints are based on simulation results of CDN scenario according to the failure model of communication issues.

Resumo. *Coreografias de serviços Web permitem que serviços sejam compostos de forma colaborativa, por causa da descrição global e coordenação P2P descentralizada. Como as infraestruturas e implementações não são maduras o suficientes para executar (enact) coreografias, a avaliação de requisitos de QoS e o comportamento da composição é uma tarefa difícil. Neste trabalho, é proposta uma abordagem para a definição de requisitos de QoS entre os serviços dos participantes envolvidos em uma coreografia, levando em consideração composição de QoS e aspectos de infraestrutura. Para tanto, foi desenvolvido um simulador de coreografias, a fim de simular a execução de coreografias. A proposta é avaliada por meio de simulações de uma coreografia de serviços que implementa uma aplicação de CDN.*

1. Introdução

O modo de desenvolver aplicações e sistemas complexos evoluiu com o passar do tempo até convergir para arquiteturas de software e modelos de computação orientados a serviços, o que é chamado de Computação Orientada a Serviços (*Service Oriented Computing* – SOC) [Papazoglou et al. 2007]. As aplicações baseadas em serviços requerem composições de serviços como um fator chave para alavancar um desenvolvimento rápido, de baixo acoplamento e flexível na integração com outros sistemas [Papazoglou et al. 2007].

Atualmente existem duas principais abordagens para compor serviços, a orquestração e a coreografia. A orquestração de serviços é uma composição centralizada, já que uma entidade denominada *orquestrador* é responsável por coordenar a comunicação dos serviços participantes. Por outro lado, a coreografia de serviços é uma

composição descentralizada já que é uma descrição de interações ponto a ponto entre os serviços participantes, ou seja, nesse modelo, não há a figura de um controlador central [Barker et al. 2009b].

Devido ao número crescente de dispositivos móveis que se conectam à Internet, uma abordagem orientada a serviços centralizada como a orquestração pode não ser escalável em termos de largura de banda para lidar com o número cada vez mais crescente de dispositivos e serviços. Nesse cenário, uma abordagem descentralizada, como a coreografia, pode se tornar a mais adequada para as características da Internet do Futuro [Stuttgart 2012].

Durante a execução de coreografias de serviços[1], o estado dos elementos de rede desempenham um papel fundamental. Deve haver garantias de Qualidade de Serviço - QoS, de modo que haja vantagens usando um modelo de negócios descentralizado. Um método comum para definir garantias entre um provedor de serviços e um cliente é por meio de um Acordo de nível de serviço - SLA. Atualmente, implementar e executar uma coreografia de serviço real é uma tarefa complexa já que a tecnologia para suportar esse paradigma de composição de serviços está imatura, especialmente pela falta de motores de execução cientes de coreografia [Kopp et al. 2010]. Assim, os mecanismos para medir parâmetros de QoS, e estabelecer requisitos de QoS não estão bem desenvolvidos para coreografias.

Este trabalho apresenta uma proposta para detectar falhas não funcionais dos participantes de uma coreografia na etapa de projeto e a partir delas estabelecer requisitos e restrições de QoS de maneira a definir SLAs iniciais. Nossa abordagem realiza uma avaliação de desempenho mediante simulações de cenários definidos a partir da coreografia alvo. Tais cenários são instâncias de coreografias mas com diferentes configurações de acordo com um modelo de falhas de atributos de QoS. Os atributos de QoS envolvem aspectos de serviços Web, troca de mensagens dos participantes, e especialmente parâmetros de desempenho da rede. Dessa maneira, os requisitos e restrições de QoS estão baseados na análise dos resultados das simulações de cenários com diversos níveis de falha.

O simulador utilizado para avaliar a nossa proposta também representa uma contribuição deste artigo. Este novo simulador estende aquele apresentado em [Guimaraes et al. 2012] por meio da adição do suporte a composição e monitoramento de QoS.

As contribuições desse trabalho são:

- Uma metodologia para estabelecer requisitos e restrições de QoS iniciais em coreografia de serviços Web.
- Um modelo de falhas e de QoS para coreografias de serviços.
- Um simulador de coreografias com suporte a requisitos de QoS.

Nosso trabalho difere dos encontrados na literatura porque realiza uma análise de desempenho em etapas antes do desenvolvimento (como na modelagem e projeto) de coreografias, o que serve também como primeira etapa para definir os requisitos de QoS

[1]Utilizaremos o termo executar coreografias por falta de uma tradução ideal para o termo *to enact choreographies*

e SLA entre os participantes. Além disso, este trabalho faz uso do modelo de interação de coreografias oferecido no padrão BPMN na versão 2.0.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta os conceitos básicos necessários para a compreensão da proposta como coreografias de serviços Web, BPMN e QoS. Na Seção 3 são apresentados os trabalhos relacionados a esta pesquisa. Na Seção 4 é descrita a proposta que inclui a descrição do modelo de QoS e de falhas, o cálculo e agregação de QoS, e a arquitetura do simulador de coreografias. A Seção 5 apresenta os cenários simulados para avaliar o modelo proposto e os resultados das simulações. Finalmente, na Seção 6 são apresentadas as conclusões e trabalhos futuros.

2. Conceitos Básicos

2.1. Coreografia de Serviços

Uma coreografia de serviços é uma forma de compor serviços de maneira colaborativa e é uma descrição ponto a ponto (P2P) de interações dos comportamentos externamente observáveis dos seus participantes (serviços). Coreografias diferem de orquestrações por não haver um ponto central de controle ou de coordenação.Em cenários onde há transferência intensa de dados entre os participantes, a coreografia é mais eficiente do que a orquestração porque não sobrecarrega um único serviço [Barker et al. 2009a] e [Guimaraes et al. 2012].

Existem duas maneiras de especificar coreografias de serviços, com modelos de interação e com modelos de interconexão. O padrão BPMN[2] suporta a especificação de coreografias para ambos os modelos, sendo que só a partir do BPMN versão 2.0 o modelo de interação é suportado. O modelo de interação tem como bloco de construção de coreografias as interações atômicas entre participantes por meio de troca de mensagens. O modelo de interação para especificar coreografias é usado neste trabalho.

A Figura 1 mostra alguns dos elementos BPMN do modelo de interação que são considerados na construção do simulador. Informações detalhadas de todos os elementos BPMN de coreografias podem ser encontradas em [OMG 2011]. As tarefas de coreografias (*choreography task*) representam as interações atômicas que envolvem dois participantes por meio de troca de mensagens. Os *gateways* são elementos que permitem definir condições e caminhos de execução e também sincronização. Os eventos são elementos que indicam o que está ocorrendo, como o começo e finalização de uma instância de coreografia, chegada de uma mensagem, entre outros.

2.2. QoS e Monitoramento

Na computação orientada a serviços (SOC), o fornecimento de serviços com garantia de qualidade exige mecanismos que incluam modelos de qualidade de serviço (QoS) como característica chave. De fato, os modelos de QoS fornecem uma base adequada para o cumprimento de QoS em ambientes orientados a serviços [Mabrouk et al. 2009].

Em tais ambientes, a garantia de QoS pode ser afetada por diversos fatores incluindo o hardware, a infraestrutura de rede, o nível de qualidade oferecido pelos serviços

[2]BPMN: `http://www.bpmn.org/`

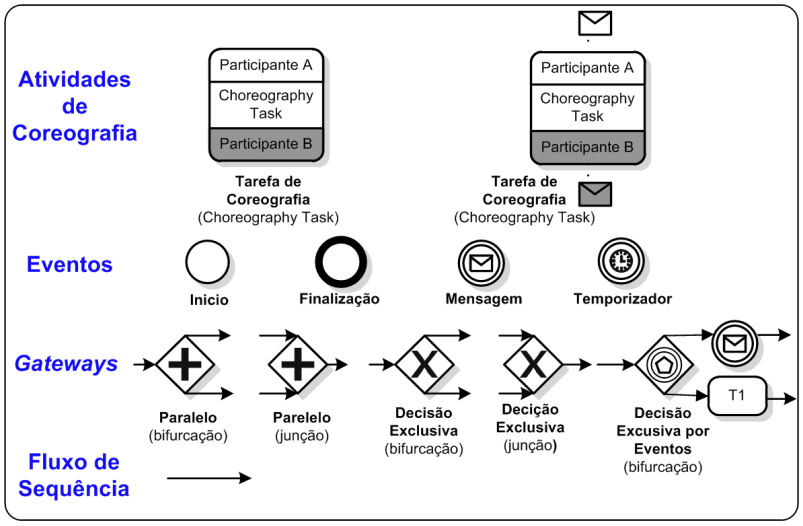

Figura 1. Elementos BPMN de coreografias no modelo de interação

de aplicação e pelas características do usuário final (mobilidade por exemplo). Isso implica que, a fim de se obter uma avaliação exata da QoS, nenhum destes aspectos devem ser negligenciados na fase de modelagem de QoS. Parâmetros que devem ser considerados em um modelo de QoS para coreografias são similares àqueles considerados em sistemas P2P [Mabrouk et al. 2009]: (i) o ambiente do serviço e o hardware e infraestrutura de rede subjacente, (ii) os serviços disponíveis, e (iii) os usuários finais.

O modelo de QoS apresentado neste artigo foca em atributos não funcionais presentes em serviços Web, principalmente de desempenho e de disponibilidade. Tais atributos de QoS abrangem os pontos citados acima (i,ii e iii) e podem estar relacionados com algum dos seguintes aspectos: serviço, mensagem e comunicação. A seguir, uma breve descrição dos atributos considerados.

- **Tempo de Execução**: Tempo necessário para executar um serviço.
- **Tempo de Resposta**: Tempo necessário para executar um serviço e retornar sua resposta ao cliente.
- **Vazão**: Número de requisições realizadas em um período de tempo.
- **Formato da mensagem**: Conteúdo consistente tanto sintaticamente quanto semanticamente da mensagem.
- **Latência de rede**: Atraso de rede ao enviar as requisições ou ao retornar as respostas.
- **Largura de banda**: Medida da capacidade de transmissão do canal de comunicação.

Um modelo de falhas descreve os tipos de falhas que podem ocorrer em um sistema enquanto está sendo executado, e ajuda a determinar quais mecanismos de tolerância a falhas deveriam ser aplicados [Liu et al. 2010]. Levando em consideração o trabalho [Liu et al. 2010], pode-se classificar para um modelo de falhas em serviços Web quatro tipos de falhas: lógicas, de sistema, de conteúdo e de nível de serviço (SLA). A seguir, uma breve descrição desses tipos:

- **Falhas Lógicas**: Detectadas na lógica definida na especificação da composição.

- **Falhas de Sistema**: Surgem no suporte do ambiente de execução e estão relacionados a falhas nos computadores, rede, sistema operacional, etc.
- **Falhas de Conteúdo**: Devido a dados ou mensagens corrompidos.
- **Falhas de SLA**: Devido a violações de QoS especificados em um SLA.

O modelo de QoS e o modelo de falhas propostos neste artigo são apresentados na Subseção 4.1. No contexto de coreografias, já que há serviços compostos, os atributos de QoS são individuais e agregados de acordo com os padrões de fluxo de trabalho e suas dependências. Um monitor deve ficar responsável pela coleta, medição e agregação de tais atributos, assim como por acompanhar a execução da coreografia e detectar falhas e violações de restrições de QoS.

3. Trabalhos Relacionados

Existem poucas infraestruturas para implementação de coreografias de serviços Web [Barker et al. 2009a]. O Pi4SOA [Zhou et al. 2006] é um arcabouço que fornece um editor para modelagem de coreografias em WS-CDL[3], e realiza verificações, validações e simulações. Porém, não suporta execução de coreografias e as suas simulações servem apenas para encontrar inconsistências na especificação. O WS-CDL+ [Kang et al. 2007] é uma proposta de um motor de execução de coreografias especificadas em WS-CDL, mas foi implementado na forma de protótipo e somente a versão 1.0 foi lançada. O OpenKnowledge [Barker et al. 2009a] é um arcabouço que fornece a capacidade de rodar sistemas distribuídos em uma arquitetura P2P (ponto a ponto), podendo rodar coreografias também, mas de maneira limitada.

Muitos simuladores para sistemas e ambientes distribuídos foram propostos. Por exemplo, o arcabouço GridSim [Buyya et al. 2002], o Pi4SOA [Zhou et al. 2006], e o arcabouço SimGrid [Casanova et al. 2008]. O arcabouço GridSim [Buyya et al. 2002] é um motor de simulação de ambientes distribuídos baseado em eventos. Ele implementa entidades para emular usuários. As requisições dos usuários são escalonadas por meio de um *broker* que os aloca nos recursos de simulação. O SimGrid [Casanova et al. 2008] é um arcabouço para simular diversos sistemas distribuídos e permite avaliar mecanismos de *clusters* e grades.

Como podemos notar, não há soluções baseadas em simulações para apoiar a execução de coreografias e menos ainda com suporte a QoS. Por conta disso, em [Guimaraes et al. 2012], um dos autores deste artigo desenvolveu um simulador inicial para demonstrar que coreografias são mais eficientes do que orquestrações de serviços. Contudo, esse simulador não suporta experimentos de coreografias com o objetivo de avaliar mecanismos relacionados com QoS ou SLA.

Por outro lado, existem alguns trabalhos acerca de coreografias de serviços que estudam mecanismos de QoS. Em [Haq and Schikuta 2010] é proposta uma abordagem baseada em agregação de SLAs para garantir qualidade de serviço (QoS) em coreografias de serviços Web, para suportar vários níveis de hierarquia de coreografias. Mas a proposta está focada na formalização do conceito de coreografia de SLA e na definição de um modelo de agregação baseado em vistas de SLA. Em [Buccafurri et al. 2008], apresenta-se uma abordagem para o tratamento de QoS em serviços Web focando-se na degradação

[3]WSCDL: Linguagem de especificação de coreografias proposto pela W3C

da qualidade. Também se desenvolveu um arcabouço para fornecer funcionalidades de adaptação por meio de um monitoramento e ações de recuperação em caso de falhas. Apesar de descrever brevemente uma abordagem do monitoramento de uma coreografia para um exemplo aplicado a CDN em tempo real, o trabalho não foca na especificação de QoS nem na agregação de QoS em coreografias.

Nosso trabalho diferencia-se dos demais por apresentar um novo simulador para coreografias de serviços Web que permite a avaliação de mecanismos de garantia de QoS e SLA. Além disso também é apresentada uma metodologia para estabelecer requisitos de QoS e um modelo de falhas e QoS para coreografias de serviço.

4. Metodologia

4.1. Modelo de QoS e Falhas

Os atributos de QoS considerados neste trabalho estão envolvidos em interações de serviços, isto é, de acordo a um modelo de requisição e resposta para um serviço (individual ou composto). Essas interações entre serviços estão baseadas nas interações atômicas (atividades de coreografia) do BPMN2. A Figura 2 mostra o mapeamento das atividades de coreografia para seu equivalente em interações de serviço, de maneira a definir aí o modelo de QoS (atributos, métricas, cálculos, entre outros).

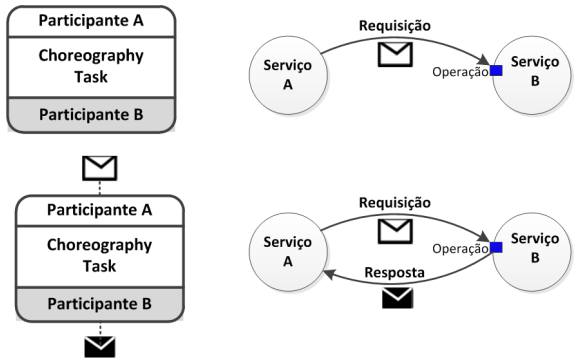

Figura 2. Interação de serviços a partir de interações atômicas do BPMN2.

A Figura 3 mostra os atributos de QoS básicos envolvidos em uma invocação de um serviço tais como, o tempo de comunicação na requisição, o tempo de execução, o tempo de comunicação na resposta e o tempo de resposta. O tempo de comunicação depende de atributos de QoS de rede como a largura de banda e a latência; o tempo de resposta depende do tempo de execução e do tempo de comunicação da requisição e da resposta. Além do mais, no caso de serviços compostos o cálculo dos atributos de QoS depende das medidas dos outros atributos de QoS e das dependências com outros serviços.

Os atributos de QoS podem ser calculados da seguinte forma:

- **Tempo de comunicação**: $t_C = L_{ij} + S/Bij$. Onde o S é o tamanho da mensagem, e L_{ij} e Bij são a latência e largura de banda de rede entre o ponto i e o ponto j.
- **Tempo de execução**: $t_{\text{Execução}} = t_3 - t_2$.
- **Tempo de resposta**: $t_R = t_{\text{Execução}} + t_{C1} + t_{C2}$. Onde o t_{C1} e t_{C2} são os tempos de comunicação na requisição e na resposta respectivamente.

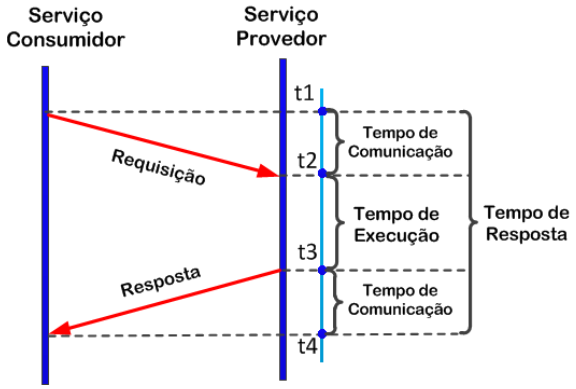

Figura 3. Atributos de QoS em uma interação com um serviço Web.

- **Tempo de execução efetivo**: $t_{\text{Execução}} = t_{Rcomposto} + t_{\text{Execução}\_individual}$. Onde $t_{Rcomposto}$ é o tempo de resposta acumulado dos serviços que são dependências do serviço atual, e $t_{\text{Execução}\_individual}$ é o tempo de execução do serviço atual.
- **Tempo de resposta composto**: $t_{R_composto} = F(aggregationType, dado, t_{R1}, ..., t_{Rk})$. Onde os $t_{R1}, ..., t_{Rk}$ são os tempos de resposta das serviços que são dependências do serviço atual, F é a função de agregação que depende do tipo de padrão de *workflow* (*gateway*) $aggregationType$, e $dado$ é a informação associada. Os *gateways* atualmente suportados são a sequência, paralelismo e a decisão exclusiva (ver Figura 1):
  - **Sequência**: $F(SEQUENCE, dado, t_{R1}) = t_{R1}$. Onde t_{R1} é o serviço em sequência do atual.
  - **Paralelismo**: $F(PARALLEL, dado, t_{R1}, ..., t_{Rk}) = max\{t_{R1}, ..., t_{Rk}\}$.
  - **Decisão exclusiva**: $F(EXCLUSIVE, dado, t_{R1}, ..., t_{Rk}) = t_{Ry}$. Onde os t_{Ry} é o tempo de resposta do serviço escolhido de acordo ao valor de $dado$.

A Figura 4 mostra as requisições e respostas enviada para e transmitidas de um serviço composto. Essas interações (requisições e respostas) são registradas mediante eventos por parte de um serviço ou cliente solicitador (eventos 1 e 4) e eventos por parte das dependências (eventos 2 e 3). Os eventos indicam quando e quais atributos de QoS devem ser medidos e garantem a ordem para garantir valores consistentes. Os atributos de QoS com as suas métricas e tipos de falhas associadas são apresentados na Tabela 1.

Tabela 1. Modelo de QoS e de falhas

Tipo	Atributo de QoS	Métrica	Tipo de Falha
Serviço	Tempo de Resposta	ms	temporização, violação de QoS
Serviço	Vazão	#requisições/s	serviço não disponível,violação de QoS
Mensagem	Formato da Mensagem	-	probabilidade de falha
Comunicação	Latência	ms	Erro de comunicação/violação de QoS
Comunicação	Largura de Banda(máxima)	Mb/s	Erro de comunicação

4.2. Simulador

A simulação é utilizada pois a implementação e a execução de coreografias de serviços Web reais ainda é difícil por conta da imaturidade das tecnologias [Kopp et al. 2010].

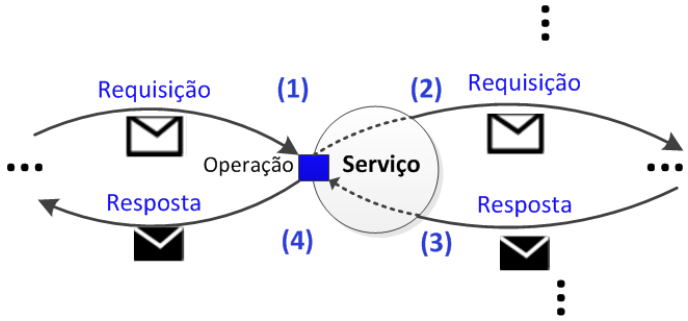

Figura 4. Atributos de QoS calculados em um evento dado. (1) Recebendo requisições de um cliente ou serviço, (2) enviando requisições para um outro serviço, (3) recebendo resposta de um outro serviço (dependência) e (4) enviando resposta para um cliente ou serviço solicitador.

Porém, implementar um simulador por completo é uma tarefa complexa. Por conta disso, decidimos usar um framework de simulação existente, o SimGrid [Casanova et al. 2008]. Como o SimGrid permite a simulação de ambientes distribuídos, ele é suficiente para servir como base para o nosso simulador de coreografias com suporte a avaliação de mecanismos de QoS.

A Figura 5 mostra a arquitetura do simulador de coreografias (ChorSim) com suporte a QoS, onde cada bloco representa um componente do simulador. Um componente depende do componente ou dos componentes que estão embaixo dele. Cada um dos componentes serão explicados a seguir. A base do ChorSim está construído sobre o arcabouço SimGrid, para suportar a definição de *hosts*, topologia de rede, comunicação entre serviços, e especificação do consumo de recursos. O "Motor de Execução de Coreografias" do ChorSim permite a criação de instâncias de coreografias e iniciar as interações dos diversos serviços envolvidos que resultarão em um grafo de requisições e um outro grafo de informações de QoS.

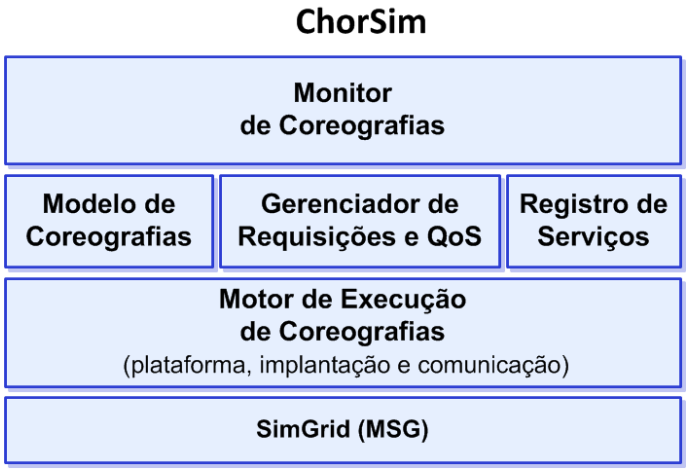

Figura 5. Arquitetura do simulador de coreografias ChorSim.

A topologia da coreografia (host, canais de comunicação e links) é configurada por meio de um arquivo XML de especificação de plataforma. Os serviços foram modelados como um conjunto de *threads* que recebem uma tarefa enviada através da rede, a

executam e em seguida enviam outra tarefa através da rede para atuar como uma resposta da requisição.

O "Modelo de Coreografias" é construído a partir de uma especificação baseada em XML que define os serviços participantes, suas dependências e suas interações. Os recursos computacionais necessários para executar, a quantidade de *threads*, o tamanho médio das respostas dadas nas interações dos serviços e suas operações são configurados por meio de um arquivo XML de implantação, que está especificado de acordo como o modelo de coreografias. O arquivo de implantação serve também como base para construir o "Registro de Serviços".

O "Gerenciador de Requisições e QoS" gerencia o grafo de requisições e o grafo de informações de QoS de acordo com o avanço e execução na interação dos serviços. Acima desta infraestrutura o "Monitor de Coreografias" é desenvolvido, usando o "Modelo de Coreografias", o "Registro de Serviços" e o "Gerenciador de Requisições e QoS". Esse monitor é responsável pela medição dos atributos de QoS dos serviços individuais e por agregá-los para calcular os atributos de QoS compostos. A agregação de QoS é realizada de acordo com as dependências, interações e os padrões de fluxo de trabalho (*gateways*) do modelo de coreografia. Por meio do monitor criam-se e gerenciam-se as instâncias de coreografias e seus respectivos identificadores para serem usados nas mensagens durante as interações.

5. Simulações e Análise de Desempenho

5.1. Metodologia dos Experimentos

Para atestar a eficácia da proposta, a coreografia apresentada em [Buccafurri et al. 2008] é utilizada. Essa coreografia representa uma aplicação de CDN (*Content Delivery Network*) para fornecimento de conteúdo multimídia como audio, vídeo e imagens. A Figura 6 ilustra a coreografia. Ela é composta de cinco serviços Web (WS_1,WS_2, WS_3, WS_4 e WS_5). Cada serviço pertence a um participante diferente e há sete canais de comunicação definindo a topologia G_{chor}, onde:

$G_{chor} = (V_{WS}, E)$ é um grafo não orientado, $WS_i \in V_{WS}$ é um serviço Web da coreografia e $e \in E$ é a comunicação entre dois serviços. $V_{WS} = \cup_{i=1}^{5}\{WS_i\}$ e , $E = \{(WS_1, WS_2), (WS_1, WS_3), (WS_1, WS_4), (WS_2, WS_4), (WS_2, WS_5), (WS_3, WS_5)\}$.

Nem sempre todos os serviços de uma coreografia são utilizados em uma instância de coreografia, isto é, há interações entre serviços que não acontecem porque a especificação de uma coreografia abrange várias possibilidades e todas as possíveis interações. Assim, os serviços utilizados em uma instância de coreografia dependem do ponto de entrada, ou seja, a primeira requisição para algum serviço da coreografia. Neste caso as instâncias de coreografias a simular, conforme a Figura 6, se iniciam com requisições do cliente para uma operação do serviço composto WS_1, que resulta em interações com os serviços WS_3 e WS_5.

O objetivo dos experimentos é analisar o comportamento do tempo de resposta total do serviço composto WS_1 em função do tamanho da resposta de WS_1 e diferentes valores de largura de banda. Para tanto, utilizam-se dois cenários definidos pelo comportamento da largura de banda da rede. O primeiro cenário possui valores de largura de banda que são fixos ao longo do tempo. O segundo cenário consiste em usar valores de

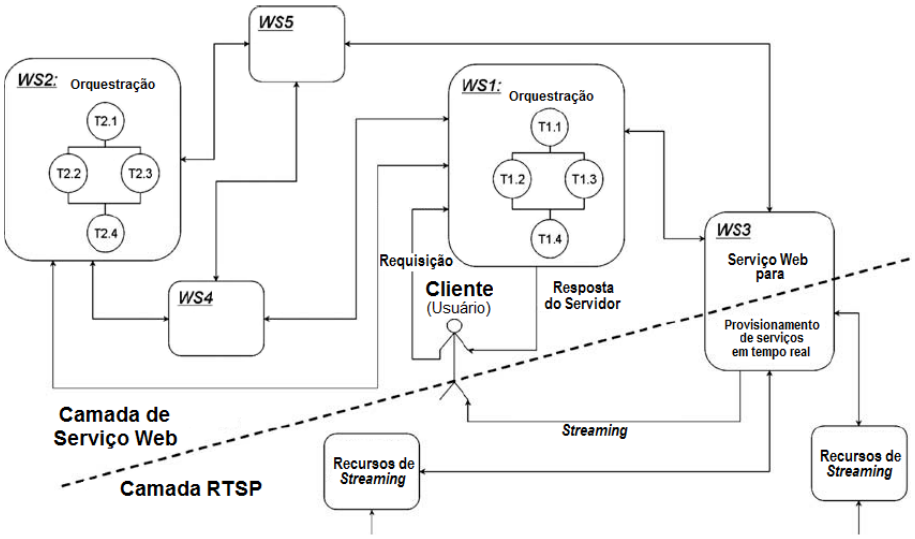

Figura 6. Coreografia de serviços da aplicação de CDN [Buccafurri et al. 2008]

largura de banda que são variáveis no decorrer do tempo representando assim a dinâmica do ambiente e a degradação por conta de falhas. A Figura 7 mostra como a largura de banda varia em um período de 100 segundos. Esta é a variação na simulação do segundo cenário. Essas variações foram geradas aleatoriamente.

A variável independente nos experimentos é o tamanho da resposta do serviço composto WS_1 que varia de $1KB$ até $100MB$. A variável dependente é o tempo médio da resposta total do serviço composto WS_1 de várias requisições simultâneas (que variam de 1 para 10 requisições) e de acordo à largura de banda da rede. Essa largura de banda está definida entre o canal de comunicação do Cliente e o serviço composto WS_1, e varia de 1Mbps até 16Mbps.

As Tabelas 2 e 3 apresentam os valores dos atributos de QoS das requisições e das respostas que são usados para as simulações. Cada simulação consistiu na execução de instâncias de coreografia iniciadas por requisições de um cliente. Desse modo, foram realizadas 960 simulações (1 a 5 requisições, 120 valores de tamanhos resposta e 16 valores de largura de banda) para cada cenário (modelo normal e variável da largura de banda). As simulações foram executadas em um computador equipado com processador Intel Core $i7-2700$K 3.5Ghz, 16GB de memória RAM e 1TB de espaço em disco rodando o sistema operacional Debian GNU/Linux versão 6.0.

Tabela 2. Configuração de valores dos atributos de QoS nas requisições

Requisições	Largura de banda	Tamanho da requisição	latência	# requisições
$Cliente$ a WS_1	1Mbps	1.95MB	$0.002s$	De 1 a 10
WS_1 a WS_3	1Mbps	5.47MB	$0.002s$	De 1 a 10
WS_3 a WS_5	1Mbps	5.47MB	$0.002s$	De 1 a 10

5.2. Análise de Resultados

Os gráficos das Figuras 8 e 9 mostram os resultados dos dois cenários de simulação, isto é, para o modelo normal e para o modelo variável com falhas. No modelo normal

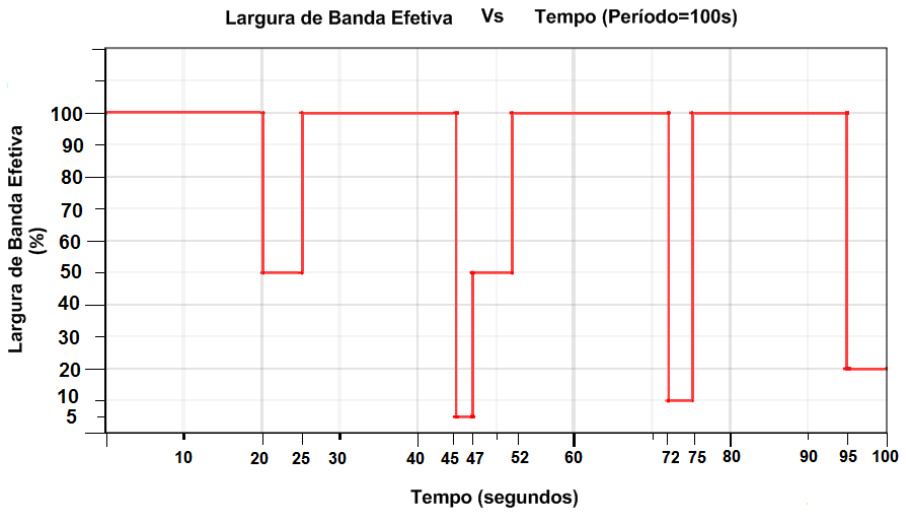

Figura 7. Modelo de falhas que mostra a largura de banda efetiva devido à degradação da largura de banda referencial em um período de 100 segundos.

Tabela 3. Configuração de valores dos atributos de QoS nas respostas

Respostas	Largura de banda	Tamanho de resposta	latência	*timeout*
WS_1 a $Cliente$	1Mbps a 16Mbps	1KB a 100MB	$0.002s$	$1000s$
WS_3 a WS_1	20Mbps	8MB	$0.002s$	$1000s$
WS_5 a WS_3	40Mbps	200MB	$0.002s$	$1000s$

(Figura 8) percebe-se que o comportamento do tempo de resposta é menor quanto maior é a largura de banda, e estabiliza-se a partir de $5Mbps$ com um tempo de resposta de $150ms$. Além disso, com valores pequenos nos tamanhos de resposta de até $30MB$ os tempos de resposta são similares com larguras de banda a partir de $2Mbps$. Com base nisso já pode-se definir restrições de QoS tais como, poder oferecer tempos de resposta menores que $450ms$ desde que se tenha uma infraestrutura para suportar uma largura de banda de mais de $5Mbps$.

Já no modelo variável com falhas, os tempos de resposta diminuem quanto maior é a largura de banda, mas diferente do modelo normal, neste cenário começa a se estabilizar a partir de $14Mbps$ de largura de banda, apesar de a partir dos $8Mbps$ os tempos de resposta serem bem próximos. Além disso, o tempo de resposta tem quase o mesmo comportamento para todos os tamanhos de resposta, diferente do modelo normal, onde para tamanhos de resposta menores que $30MB$ podia se obter algumas vantagens. Neste cenário, houveram estouros de temporização (*timeouts*) com as larguras de banda de $1Mbps$ e $2Mbps$ com mais de 2 requisições simultâneas e por conta disso essas curvas não aparecem no gráfico. Dessa maneira, as restrições de QoS estariam focadas em garantir larguras de banda mínimas maiores que $2Mbps$ e para garantir tempos de resposta mínimos são necessários valores de largura de banda maiores que $14Mbps$.

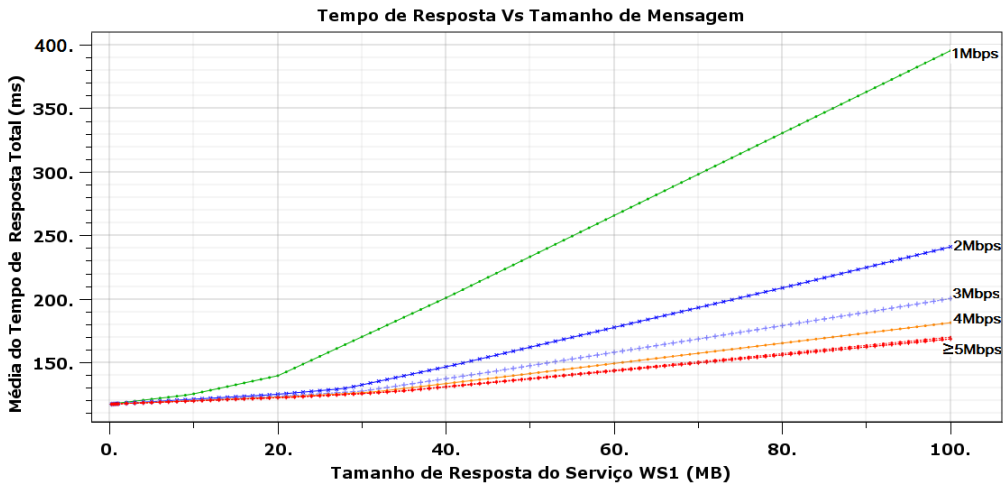

Figura 8. Tempo médio de resposta total da coreografia em função do tamanho de resposta do serviço WS1 com larguras de banda de 1Mbps até 16Mbps (intervalos de confiança não são visíveis por terem ficado muito pequenos).

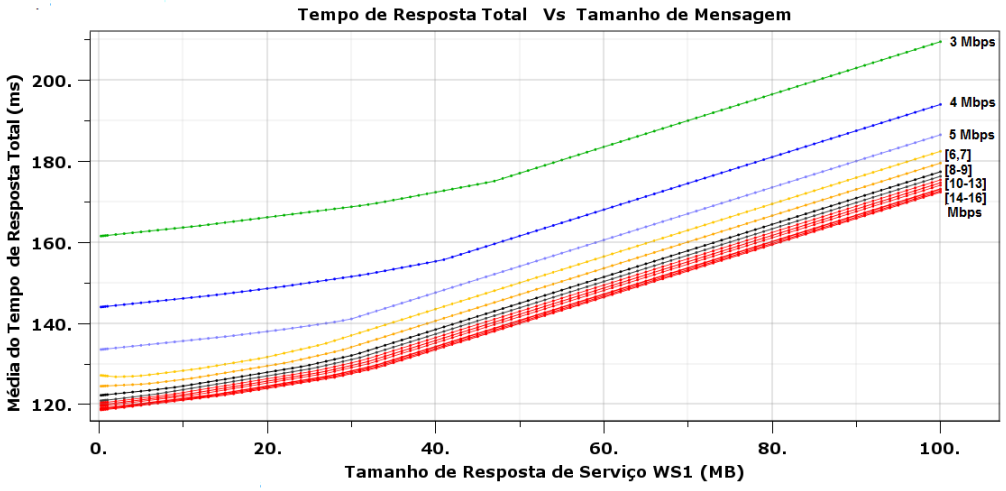

Figura 9. Tempo médio de resposta total da coreografia em função do tamanho de resposta do serviço WS1 segundo o modelo de falha. A largura de banda varia de 1Mbps até 16Mbps

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma proposta para detectar falhas não funcionais dos participantes de uma coreografia na etapa de projeto. A partir dos modelos de QoS e de falhas e por meio dos resultados das simulações podem se estabelecer requisitos e restrições de QoS de maneira a definir SLAs iniciais.

Para executar coreografias de serviços Web foi desenvolvido um simulador de coreografias para suporte de QoS chamado de ChorSim. A sua construção se baseou em elementos (atividades de coreografia, *gateways*, entre outros) do modelo de interação de

coreografias que é oferecido no padrão BPMN2. O simulador permite realizar medições de atributos de QoS atômicos e compostos e possui um monitoramento para acompanhar a execução das instâncias de coreografias.

Dessa maneira, é possível simular comportamentos que atualmente não podem ser realizados pela falta de implementações maduras e que permitam realizar pesquisas em coreografias de serviços Web, especialmente as relacionadas com QoS e monitoramento.

Trabalhos futuros envolvem suporte de mais elementos de coreografias de BPMN2 para poder simular completamente coreografia de processos.

Agradecimentos

Este trabalho é apoiado pela HP Brasil sob o acordo de cooperação técnica número HP-045/12 (Projeto Baile - `http://ccsl.ime.usp.br/baile/` e pela *European Community's Seventh Framework Programme* FP7/2007-2013 sob o acordo número 257178 (Projeto CHOReOS - Coreografias de Larga Escala para o Internet do Futuro - `http://www.choreos.eu`).

Referências

Barker, A., Besana, P., Robertson, D., and Weissman, J. B. (2009a). The Benefits of Service Choreography for Data-Intensive Computing. In *In Proceedings of the 7th international workshop on Challenges of large applications in distributed environments*, CLADE '09, pages 1–10.

Barker, A., Walton, C. D., and Robertson, D. (2009b). Choreographing Web Services. *IEEE Transactions on Services Computing*, 2(2):152–166.

Buccafurri, F., Demeo, P., Fugini, M., Furnari, R., Goy, a., Lax, G., Lops, P., Modafferi, S., Pernici, B., and Redavid, D. (2008). Analysis of QoS in cooperative services for real time applications. *Data & Knowledge Engineering*, 67(3):463–484.

Buyya, R., Murshed, M., Campus, C., and Campus, G. (2002). GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE (CCPE)*, 14:1175–1220.

Casanova, H., Legrand, A., and Quinson, M. (2008). SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, UKSIM '08, pages 126–131. IEEE Computer Society.

Guimaraes, F. P., Kuroda, E. H., and Batista, D. M. (2012). Performance Evaluation of Choreographies and Orchestrations with a New Simulator for Service Compositions. In *Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks (CAMAD)*, pages 140–144.

Haq, I. U. and Schikuta, E. (2010). Aggregation Patterns of Service Level Agreements. In *In Proceedings of the 8th International Conference on Frontiers of Information Technology (FIT '10)*, page 6. ACM Press.

Kang, Z., Wang, H., and Hung, P. C. (2007). WS-CDL+: An Extended WS-CDL Execution Engine for Web Service Collaboration. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, pages 928–935. IEEE Computer Society.

Kopp, O., Engler, L., Lessen, T. V., and Leymann, F. (2010). Interaction Choreography Models in BPEL: Choreographies on the Enterprise Service Bus. In *SBPM ONE 2010 the Subjectoriented BPM Conference (2010)*.

Liu, A., Li, Q., Member, S., Huang, L., and Xiao, M. (2010). FACTS: A Framework for Fault-Tolerant Composition of Transactional Web Services. *Framework*, 3(1):46–59.

Mabrouk, N. B., Georgantas, N., and Issarny, V. (2009). A Semantic End-to-End QoS Model for Dynamic Service Oriented Environments. *Business*, (i):34–41.

OMG (2011). Documents Associated With Business Process Model And Notation (BPMN) Version 2.0.

Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40(11):38–45.

Stuttgart, U. (2012). Research Challenges on Adaptive Software and Services in the Future Internet: Towards an S-Cube Research Roadmap. In *Software Services and Systems Research - Results and Challenges (S-Cube), 2012 Workshop on European*, pages 1–7.

Zhou, X., Tsai, W., Wei, X., Chen, Y., and Xiao, B. (2006). Pi4SOA: A Policy Infrastructure for Verification and Control of Service Collaboration. *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'06)*, pages 307–314.