1
Introdução

Com a popularização dos computadores, tornou-se interessante disponibilizar recursos de um computador para outros de forma a utilizá-los mais eficientemente. Mas, para fazer uso dos recursos de outros computadores, o programador teria que criar uma aplicação, dita distribuída, que faz a comunicação entre os nós de um sistema, sendo que os nós representam os recursos de cada máquina. Desenvolver essas aplicações distribuídas é uma tarefa complicada, por vários motivos. Para fazer a comunicação entre os processos que formam a aplicação distribuída, o desenvolvedor deve lidar com a codificação e decodificação de mensagens e com o transporte dessas mensagens entre os mesmos. Outro problema é o suporte a plataformas diferentes, já que às vezes é necessário dar suporte aos mais variados tipos de *hardware*, redes, sistemas operacionais e linguagens de programação.

Para auxiliar na construção de sistemas distribuídos são adotadas plataformas de desenvolvimento que solucionam os problemas descritos acima. Tais plataformas, denominadas *middlewares*, apresentam uma plataforma única de programação para o desenvolvedor, de forma a esconder detalhes de heterogeneidade entre sistemas, e tornam disponível uma série de serviços comuns a vários tipos de aplicação. *Middlewares* tradicionais são caixas pretas criadas justamente para esconder essas diferenças e, por isso, não oferecem muitas maneiras de se modificar seu comportamento.

Entretanto, um *middleware* construído para um domínio de aplicação pode não funcionar tão bem para outro domínio. Muitas vezes, os pré-requisitos de uma aplicação são completamente diferentes de outra, podendo chegar a ser contraditórios. Por exemplo, um *middleware* pode ser muito eficiente para aplicações de controle, onde há poucos dados a serem transmitidos mas o retardo da transmissão deve ser muito pequeno ou inexistente. Entretanto, para utilizar o mesmo *middleware* para aplicações multimídia, pode ser que sua implementação, que levava em conta o tamanho pequeno das mensagens enviadas, não seja muito eficiente, mostrando que os requisitos desse tipo de aplicação são bem diferentes das aplicações de controle. Às vezes, mesmo com pré-requisitos parecidos, pode ser interessante obter mudanças pequenas no

comportamento do *middleware*, como políticas de gerência de recursos, ou habilitação de novas funcionalidades como segurança na comunicação.

Para lidar com essas diferenças em requisitos, uma característica importante a ser buscada em um *middleware* é a sua capacidade de adaptação de partes desse *middleware* de acordo com o objetivo final do programa que o utiliza. Neste trabalho, estudamos algumas das técnicas utilizadas na implementação de *middlewares* adaptáveis, analisando alguns exemplos disponíveis na literatura e, para organizar essas técnicas de modo a podermos entendê-las melhor, classificamos os *middlewares* estudados de acordo com a forma que a adaptação acontece e em que momento ela é feita.

Alguns dos *middlewares* estudados baseiam-se apenas na troca de protocolos de comunicação antes da execução (01) (02), enquanto outros se baseiam em mudanças no fluxo de execução e recomposição dinâmica de partes do *middleware* em tempo de execução (03) (04). Além disso, apresentamos algumas das tecnologias utilizadas para a criação dessas adaptações, tais como a reflexão computacional, o projeto baseado em componentes e a programação orientada a aspectos.

Implementamos durante nosso estudo um protótipo de um *middleware* adaptável totalmente desenvolvido com a linguagem interpretada Lua (05), baseado no OiL (06). Com esse protótipo, investigamos como uma das técnicas adotadas em outros *middlewares* – o projeto baseado em componentes – se aplica a um *middleware* implementado em uma linguagem interpretada e dinâmica como Lua. Damos atenção especial à implementação de mecanismos relacionados ao reuso de componentes pelo *middleware*, apresentando exemplos de modificações na sua estrutura através da troca de alguns componentes para a criação de novas funcionalidades, tais como a troca de protocolos de comunicação e a adição de um escalonador baseado em corrotinas (07). Escolhemos neste estudo fazer experimentos relacionados a adaptações estáticas, isto é, adaptações que acontecem antes da execução do *middleware*.

No capítulo 2, apresentamos a taxionomia utilizada e os *middlewares* analisados. No capítulo 3, apresentamos um modelo de componentes usado para a criação da nova arquitetura do OiL. No capítulo 4, descrevemos a divisão do OiL em componentes, explicando em detalhes cada uma das partes que formam o middleware. No capítulo 5, apresentamos os experimentos realizados com a componentização, usando exemplos de adaptação, e comparamos a aplicabilidade do protótipo com a versão original do OiL com testes de desempenho. Finalmente, no capítulo 6, concluímos os resultados do projeto, comparando com os trabalhos estudados e apresentamos os possíveis trabalhos futuros relacionados com os resultados obtidos.