1
Introdução

A evolução das redes de comunicação tem propiciado um crescimento significativo no número e na importância de sistemas distribuídos. Como exemplo temos sistemas computacionais separados geograficamente, desenvolvimento impulsionado pela cooperação e a integração destes idealmente de maneira uniforme e transparente para os usuários. Entretanto, uma infra-estrutura que combina o hardware e software é fundamental para o melhor aproveitamento destes sistemas. Na parte do software encontramos os algoritmos distribuídos, cujo comportamento correto é crucial para o funcionamento desta infra-estrutura.

Nesse mesmo ritmo temos também o avanço da computação móvel. Este fato pode ser facilmente comprovado pelo número expressivo de celulares e pda's hoje existentes. Quanto à infra-estrutura, organização é a palavra chave para o bom funcionamento destas redes: organização na comunicação e na estruturação das redes. Para a comunicação, inúmeros protocolos e algoritmos distribuidos tem sido criados para diversos fins (PJ96, JMB01, RE02, CL00, ESO00, Wik05a, Wik05b, Lyn96). Na estruturação das redes, vários modelos tem sido propostos, destacando-se as redes estruturadas e as redes ad hoc.

Quanto maior é a complexidade da rede, maior é o esforço para gerenciá-la. Parte deste esforço concentra-se na especificação, simulação e verificação da corretude de protocolos e algoritmos para estas redes. Várias ferramentas tem sido utilizadas para a verificação de protocolos (Hol97, CCGR00, BLL+95, BGM02, BGL+00, EMS03, Lin01, Z.102, PFO98, ZBG98, BBE+99, RE01). Entretanto, seja pela pouca difusão ou desconhecimento destas ferramentas, seja pela alta curva de aprendizagem, são constantemente encontrados na arquitetura especificações destes protocolos sem a devida verificação.

Estes métodos de verificação de corretude tem suas vantagens e desvantagens, e a escolha de uma ferramenta usualmente depende: da experiência do testador; das características do protocolo (tempo-real, probabilístico, para redes ad hoc ou estruturadas, etc) e da compatibilidade destes com a linguagem do verificador (formato de entrada da especificação e saída do processo de verificação). Cada verificador adota sua linguagem para a especificação do

modelo e das propriedades. É comum encontrarmos estas linguagens num nível de abstração muito baixo, o que dificulta a especificação já que a codificação aumenta, aumentando também a probabilidade do surgimento de erros. Isto compromete o entendimento de todo o processo de análise e, consequentemente, do resultado obtido.

Motivados por parte destes problemas, propomos uma arquitetura voltada para facilitar a especificação e verificação formal de protocolos. Tendo em vista a dificuldade normalmente encontrada na verificação destes protocolos, como protocolos complexos, linguagem de especificação inadequada, resultados retornados pelas ferramentas de verificação pouco conclusivos e a escassez de ferramentas formais de domínio específico para estas tarefas, apresentamos uma proposta de arquitetura que, apresentada de forma simplificada, define uma camada mais abstrata sobre o processo usual de verificação formal de protocolos.

Nos testes realizados, a verificação formal foi feita utilizando verificadores de modelos. Entretanto, a abordagem não está restrita à estes verificadores, bastando a criação de módulos tradutores específicos para a ferramenta adotada, os quais serão comentados na descrição da arquitetura (seção 3.1). Quanto ao escopo do problema, dada a proximidade entre protocolos e algoritmos distribuídos, também podemos utilizar a arquitetura para validar estes, o que veremos ao longo do texto.

O elemento central da arquitetura é a sua linguagem de especificação (LEP - Linguagem de Especificação de Protocolos), a qual foi projetada para ser legível e concisa como as descrições de algoritmos encontradas nos livros didáticos. Esta permite a definição da topologia sobre a qual o protocolo será verificado, o protocolo propriamente dito e as propriedades a serem verificadas. Esta separação entre a especificação da topologia e do protocolo não é normalmente encontrada nas linguagens de especificação dos verificadores formais, visto que estas são usualmente de propósito mais geral. Para nós esta não-separação, dentre outros fatores, dificulta a legibilidade do código especificado. Além disso, para vários algoritmos, o comportamento dos nós independe da quantidade e das conexões destes na rede.

LEP contém construções abstratas, chamadas de *pronomes*, que têm por objetivo tornar as especificações dos protocolos e de suas propriedades mais simples, compactas e, consequentemente, mais legíveis. Pronomes são uma maneira uniforme e pontual de se fazer referência a um conjunto de elementos num sistema. Para uniformizar entradas e saídas da arquitetura, prevemos o retorno dos contra-exemplos dos verificadores no mesmo nível de abstração de LEP.

Na validação de protocolos e algoritmos distribuídos, a topologia inicial é normalmente fornecida de forma manual, com algumas variações. Nesta tese propomos a idéia de *Gramática de Grafos com Atributos - AGG* (descrita no Apêndice A), que mistura os conceitos de gramática de grafos (CGP00) e de atributos (SSK95). O objetivo é permitir a geração automática de topologias iniciais e a definição dos pronomes em função destas topologias. De forma a simplificar a especificação, LEP disponibiliza algumas macros pré-definidas para geração de topologias comuns, como as topologias em estrela, as arbitrárias e as totalmente conectadas. Assim, ao invés de analisar um único modelo, a arquitetura passa a verificar uma classe de modelos (modelos cuja topologia é definida por uma AGG).

Com o uso de AGG podemos gerar topologias incompatíveis com o protocolo ou algoritmo a ser verificado. Por exemplo, essa incompatibilidade ocorre se num problema de alocação de recursos, como o problema dos filósofos, temos menos recursos (talheres neste caso) que o exigido pelo problema. Para evitar estes cenários, definimos o conceito de *Modelo Mínimo - MM*, que define o menor número de instâncias numa rede necessárias para que se verifique um protocolo. Também apresentamos um algoritmo para a obtenção do MM. com o intuito de não infringirmos a filosofia da arquitetura, de verificação automática e transparente. De maneira complementar, discutimos conceitualmente o que limita estes modelos no outro extremo, ou seja, o que seria o modelo máximo.

Como estudos de caso apresentamos dois (2) protocolos projetados para computação móvel: o RDP (ESO00), para redes estruturadas e similar ao IP-móvel (PJ96), e o DSR (JMB01), para redes ad hoc. O motivo inicial da escolha destes protocolos foi o interesse em se tratar estes dois (2) modelos (estruturado e ad hoc) de forma similar, o que é contemplado pela arquitetura proposta. Além destes, também especificamos alguns algoritmos distribuídos, os quais são apresentados ao longo do texto. Por isso, deste ponto em diante, as ocorrências dos termos protocolos e algoritmos distribuídos são intercambiáveis e de igual valor para a arquitetura.

Como exemplo comparativo do quão compacto se tornam as especificações em LEP, listamos no Apêndice C especificações do algoritmo de Eleição de um Líder em um anel nas linguagens SDL (Z.102), Murφ (Dil96), CDL (KG02), Promela (Hol97) e XL (YRS03), alem de LEP. É importante salientar que está comparação se restringe apenas ao nível sintático das especificações. Ou seja, não investigamos questões como a relação entre o tamanho das especificações e o esforço computacional para suas avaliações.

1.1

Organização da Tese

O restante da tese está organizado da seguinte forma. No Capítulo 2 falamos brevemente sobre as formas de validação comumente utilizadas para a análise de protocolos. Ao final, discutimos cada um dos métodos utilizados, visando questões como usabilidade e curva de aprendizado.

No Capítulo 3 apresentamos a arquitetura proposta. Descrevemos o seu funcionamento, dando ênfase para a linguagem de especificação LEP. Na instanciação da arquitetura para algum método de verificação, utilizamos os verificadores Spin (Hol97) e NuSMV (CCGR00) e descrevemos como a arquitetura se comporta com estes. Ao final, relatamos como se dá a recuperação dos resultados retornados pelo verificador no nível da linguagem de entrada (LEP).

Em seguida, no Capítulo 4, apresentamos dois (2) estudos de caso de especificação de protocolos. Um para redes estruturadas e outro para redes ad-hoc, de forma a demonstrar a independência da arquitetura quanto a uma ou outra abordagem. Ao final, comparamos brevemente LEP com outras linguagens de especificação através de especificações do Algoritmo de Eleição de um Líder em um Anel (Lyn96). Estas especificações são apresentadas no Apêndice C.

No Capítulo 5 apresentamos uma discussão sobre otimização dos modelos gerados. Como a arquitetura funciona de maneira essencialmente automática, é necessário que controlemos a geração de estados de forma a minimizar o problemas de explosão de estados dos verificadores de modelos (seção 2.2.2).

Após, discutimos no Capítulo 6 um conjunto de trabalhos coletados ao longo do desenvolvimento da tese. Estes trabalhos foram divididos em propostas similares, linguagens de especificação, ferramentas utilizadas e trabalhos afins.

Finalmente no Capítulo 7 discutimos as conclusões obtidas com este trabalho. Além disso, apontamos algumas extensões e trabalhos futuros, baseados na abordagem apresentada.

Nos Apêndices A, B e C temos algumas definições utilizadas ao longo do texto, a especificação de LEP em BNF e diversas especificações do Algoritmo de Eleição de um Líder num Anel, respectivamente.