UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APOIO À PESQUISA
PROGRAMA INSTITUCIONAL DE BOLSA DE INICIAÇÃO CIENTÍFICA

UM SISTEMA OPERACIONAL AUTONÔMICO PARA REDES

Bolsista: Paulo César da Rocha Fonseca, CNPq

MANAUS
2010

UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO DE APOIO À PESQUISA
PROGRAMA INSTITUCIONAL DE BOLSA DE INICIAÇÃO CIENTÍFICA

RELATÓRIO FINAL
PIB – E – 067/2009
UM SISTEMA OPERACIONAL AUTONÔMICO PARA REDES

Bolsista: Paulo César da Rocha Fonseca, CNPq
Orientador: Profº Drº Edjard Souza Mota

MANAUS
2010

Esta pesquisa, financiada pelo Conselho Nacional de Pesquisa – CNPq, através do Programa Institucional de Bolsas de Iniciação Científica da Universidade Federal do Amazonas.

Propõe uma solução autonômica para o gerenciamento das redes de computadores levando características autonômicas para o núcleo de um sistema operacional para redes. As redes de computadores modernas possuem muitas funções tornando-se cada vez mais complexas e consequentemente possuindo um gerenciamento cada vez mais complexo. A computação autonômica possui a preocupação de retirar do ser humano a preocupação com tarefas de baixo nível, automatizando-as e deixando a seu cargo apenas a definição de políticas e diretrizes que devem ser seguidas pelo computador. As arquiteturas autonômicas para redes propostas na literatura argumentam a necessidade de existência de uma laço de controle entre os dispositivos da rede e dos agentes de controle, porém a heterogeneidade da rede dificulta a tradução entre as políticas de alto-nível que as atividades necessárias para cumprir tais objetivos. A abordagem escolhida por este projeto é a implementação das características autonômicas em um sistema operacional de redes uma vez que ele provê uma interface entre os agentes e os dispositivos da rede. Um objetivo foi mapear quais características são mantidas em espaço de usuário e quais seriam as mais críticas que seriam implementadas no núcleo para garantir confiabilidade (caso haja um colapso de um ou mais agentes a rede não seja prejudicada). O sistema operacional para redes a ser usado como plataforma é o NOX. O NOX é um controlador de rede do protocolo OpenFlow, o qual permite a manipulação do tráfego de pacotes para fins experimentais. Os testes dos componentes foi feito inicialmente em um ambiente virtual gerado por script utilizando a ferramenta de virtualização qEmu, switchs virtuais VDE e terminais Xterm. Recentemente este script foi substituído pela ferramenta Mininet, a qual foi disponibilizada pelo próprio grupo responsável pelo projeto Openflow, que provê um ambiente de virtualização com uma maior gama de funcionalidades. Foram realizados estudos aprofundados do protocolo OpenFlow e da estrutura das suas mensagens assim como estudos dos componentes presentes no NOX, as suas funcionalidades e as ferramentas disponibilizadas pela suas bibliotecas. Construiu-se um componente que realizava o envio de pacotes do NOX para os hosts com o objetivo de explorar o envio e o recebimento de pacotes na rede utilizando os métodos e bibliotecas disponíveis. Também se estudou o padrão de Fundação para Agentes Físicos Inteligentes (FIPA) para a comunicação entre os agentes da rede. Chegou-se a conclusão que as características autonômicas de auto-configuração e auto-recuperação seriam levadas ao núcleo do sistema operacional, pois a rede deve ter a capacidade de se adequar a diferentes situações e se recuperar de possíveis falhas. Outra característica que deve ser implementada no núcleo do sistema operacional é a plataforma de comunicação entre agentes, uma vez que a troca de informações entre eles é essencial para a decisão de quais ações serão ou não tomadas. Apesar de não ter cumprido todos os objetivos este projeto permitiu a aquisição de conhecimento e experiência com as plataformas utilizadas e serviu como base sólida para futuros projetos.

SUMÁRIO

1. INTRODUÇÃO

A complexidade das redes de computadores vem crescendo constantemente com o passar do tempo, novas tarefas, funções, aplicações e serviços são adicionados as suas características. E quanto mais complexas as redes se tornam, mais difícil é administrá-las em termos de baixo nível.

A computação autonômica tem assumido um papel importante na administração de tarefas complexas, pois se propõe a automatizar as tarefas e decisões de baixo nível, deixando a cargo humano apenas o estabelecimento de políticas de alto nível que seriam as diretrizes do comportamento a ser assumido pelo computador. Existem arquiteturas autonômicas para redes que defendem o uso de *loops* de controle que iriam analisar o contexto da rede através de seus recursos gerenciados, escolher a política adequada e aplicá-la. Porém, os recursos de uma rede diferem entre si tanto hardware quanto em software.

Uma abordagem seria desenvolver agentes específicos para cada dispositivo, porém essa solução torna-se inviável uma vez que a cada novo dispositivo de rede lançado no mercado ou atualização de firmware dos já existentes um agente específico deveria ser desenvolvido para aquele dispositivo.

Outra abordagem seria fornecer uma camada de abstração entre os dispositivos de baixo nível e as políticas de alto nível. Uma pesquisa de doutorado em andamento do Programa de Pós-Graduação em Informática da UFAM defende que esses agentes sejam construídos sobre um sistema operacional de rede que servirá como a camada de abstração entre os agentes e os dispositivos.

A ideia principal é levar a autonomia também ao sistema operacional de rede, criando um sistema operacional autonômico de rede, pois há evidências que o espaço de usuário do sistema não seja uma plataforma robusta para hospedar os agentes uma vez que caso ocorra uma falha grave no sistema todas as aplicações tornar-se-iam instáveis. Para torná-lo autonômico seriam levadas ao seu núcleo as características de auto-configuração, auto-otimização, auto-recuperação e auto-proteção, propriedades intrínsecas de sistemas auto-gerenciados. Assim, as falhas decorrentes de má configuração ou ataques maliciosos podem ser tratadas ainda em nível de sistema operacional e qualquer instabilidade não é repassada as aplicações sendo executadas.

A tarefa do bolsista foi mapear quais tarefas seriam mantidas em espaço de usuário e quais tarefas serão enviadas para o núcleo (uma vez que apenas os elementos autonômicos essenciais deveriam ser passados para o núcleo para não sobrecarregar o núcleo), implementar essas tarefas no núcleo do sistema operacional autonômico de rede e avaliar o seu impacto no desempenho geral do sistema.

2. REVISÃO BIBLIOGRÁFICA

Em março de 2008 o OpenFlow foi apresentado em [McKeown et al. 2008]. OpenFlow é uma maneira de testar protocolos experimentais em redes reais sem interferir com o tráfego de produção, ele é uma característica que pode ser adicionada a switches ethernet que disponham de tal funcionalidade e trabalha à nível de tabela de fluxo, removendo e adicionando novos registros de fluxos nela. Os registros contém o padrão do cabeçalho do pacote pertencente ao fluxo, contadores (para fins estatísticos) e a ação associada ao fluxo. A tabela de fluxo é administrada por um controlador remoto que decide qual ação associar a cada novo fluxo e está conectado ao switch através de um canal seguro (criptografado por SSL). Atualmente um dos principais objetivos do grupo responsável pelo OpenFlow é aumentar a quantidade de empresas que comercializam switches OpenFlow e universidades que o implementam em seu campus. Recentemente foi lançada a versão 1.0 do OpenFlow, sendo a primeira que está oficialmente pronta para ser implementada em switches reais.

Paralelo ao desenvolvimento do OpenFlow ocorre o do NOX, um sistema operacional para redes centralizado baseado no OpenFlow e introduzido em [Gude et al. 2008]. O NOX age como um controlador do OpenFlow, adicionando e removendo registros de fluxo da tabela de fluxo, para exercer controle sobre a rede e seus recursos. Ele provê um plataforma simplificada para escrever componentes que irão gerenciar os recursos da rede. O NOX pode ser dividido em dois aspectos: Os componentes externos de alto-nível que lidam com a API e são feitos majoritariamente em Python, também podem ser escritos em C++ ou ambos, e o núcleo do sistema operacional que lida com tarefas de baixo nível e a sua maior parte é escrita em C++, para conservar a performance, com alguns trechos de código em Python.

Outra área relacionada a este trabalho é a computação autonômica, a qual foi apresentada em [Kephart et al. 2003]. que expôs a computação autonômica, seus conceitos e objetivos, introduzidos pela primeira vez em 2001 por Paul Horn, na época vice-presidente da IBM. No artigo eles definem as características necessárias para um sistema ser autonômico: auto-configuração, auto-otimização, auto-recuperação e auto-proteção.

Uma área de pesquisa que tem se desenvolvido bastante é a de autonomia aplicada a redes [Dobson et al. 2006]. Em [Jennings et al. 2007] uma arquitetura autonômica para redes chamada FOCALE(Foundation Observation Comparison Action Learn rEason). Esta arquitetura tem como objetivo deixar como papel humano o estabelecimento de políticas a serem aplicadas pela rede dependendo do contexto que ela se encontra. A rede é capaz de adquirir informações dos seus recursos, identificar um contexto, aplicar ações associadas a aquele contexto, caso haja alguma, ou estabelecer políticas de ação, caso não haja ações

pré-definidas, avaliar a eficácia da solução empregada e armazenar os resultados para futuras consultas. Esse comportamento é estabelecido através de dois *loops* de gerenciamento: um de manutenção, utilizado quando o estado da rede é igual ao estado desejado ou está indo em direção a ele, e outro de ajuste, utilizado quando uma ou mais políticas de reconfiguração devem ser aplicadas ou novas políticas devem ser codificadas e implementadas.

3. MÉTODOS UTILIZADOS

Inicialmente, foi realizado um estudo sobre as arquiteturas de gerenciamento de redes autonômicas, com ênfase no FOCALE, para ter uma melhor compreensão de como funciona uma arquitetura autonômica para redes e usá-lo como base para orientar o comportamento do futuro núcleo autonômico do NOX. Também se estudou o núcleo do NOX para entender seu funcionamento interno e como ele lida com as aplicações.

Também foram mapeadas quais das quatro características autonômicas (auto-configuração, auto-otimização, auto-proteção e auto-recuperação) deveriam ser migradas para o núcleo do sistema operacional e futuramente ocorrerá a implementação de tais características e avaliação do sistema e do seu desempenho com tais características.

Para se familiarizar com ambiente OpenFlow/NOX foi uma montada uma pequena rede com máquinas reais uma funcionando com um switch OpenFlow, com uma interface de rede a mais fornecida por um cartão PCMCIA Ethernet, outra servindo como NOX e outra como host, todas conectadas através de cabos Ethernet crossover. Posteriormente foi desenvolvido um ambiente virtual utilizando as ferramentas xterm (emulador de terminal), o qEmu, programa de criação de máquinas virtuais e o VDE, um switch Ethernet virtual que foi utilizado para fazer a ligação entre as máquinas virtuais e montar a rede. Este ambiente fornece a opções de adicionar quantos hosts forem necessários para o experimento, a topologia representa uma rede composta por um controlador NOX conectado através de um canal seguro criptografado por SSL a uma porta de um switch OpenFlow e uma quantidade arbitrária de hosts conectados ao switch e com possibilidade de comunicação entre si. Uma versão mais atual deste ambiente fornece a possibilidade de unir várias sub-redes em uma só rede e com comunicação entre duas sub-redes distintas.

4. RESULTADOS E DISCUSSÕES

Uma vez que o objetivo do trabalho é tornar o núcleo do sistema operacional para redes autonômico levando características autonômicas a ele, primeiro foi necessário avaliar quais dessas características autonômicas deveriam ser levadas a ele e quais poderiam ser mantidas em espaço de usuário. As características que seriam levadas para o núcleo do sistema deveriam ser apenas imprescindíveis ao seu funcionamento para não sobrecarregar o processamento do núcleo ao mesmo tempo em que fornece os elementos necessários para a estabilidade da rede. Dentre as quatro características autonômicas(auto-configuração, auto-otimização, auto-recuperação e auto-proteção) as de recuperação e proteção foram identificadas com o perfil proposto pois elas fornecem à rede o suporte necessário para não entrar em colapso caso as aplicações de alto-nível tornem-se instáveis.

Em relação ao núcleo do NOX, estudos foram realizados para identificar as funcionalidades dos componentes internos relativos ao núcleo, uma vez que a documentação deles é escassa, como afirmaram os próprios pesquisadores responsáveis pelo sistema operacional, e em quais componentes já existentes seria necessário implementar tais características. O protocolo OpenFlow também foi estudado através de suas especificações[Heller 2010] uma vez que o conhecimento da estrutura das mensagens que ele troca entre os componentes da redes é necessário para a manipulação das mesmas.

Também foi realizada a adição de componentes simples ao sistema operacional para familiarizar-se com a plataforma e como ela se comporta com a inclusão de outras aplicações. Tais componentes foram adicionados tanto em ambiente real quanto em ambiente virtual e o funcionamento mostrou-se idêntico em ambos mostrando assim que o ambiente virtual é uma plataforma confiável para a realização de testes.

Os testes realizados foram feitos em um ambiente real de pequena escala (com natureza e topologia supracitada) e em um ambiente virtual mais flexível em relação ao número de hosts e a quantidade de sub-redes.

Para prover de uma plataforma mais robusta para a realização dos testes, permitindo experimentos em maior escala e com ambientes mais heterogêneos, foi montada uma máquina virtual no servidor que se encontra no bloco do Departamento de Ciência da Computação. Esse servidor é acessado remotamente.

Nos últimos meses o grupo do Openflow disponibilizou uma nova ferramenta chamada Mininet, ela provê um ambiente virtual robusto para testes e incluem quase todas as funcionalidades e características presentes em uma rede OpenFlow real. O ambiente virtual criado a partir de scripts utilizando o qEmu, que foi utilizado para realizar os testes

iniciais, foi substituído pelo Mininet dado o fato do Mininet prover uma gama de funcionalidades mais completa.

Em relação às características autonômicas, verificou-se que um fator essencial para o bom funcionamento da rede é a comunicação entre os elementos da mesma. O foco então foi redirecionado para dois pontos principais da comunicação: como as mensagens são enviadas e qual a estrutura dessas mensagens. No primeiro aspecto foram estudados quais os métodos/bibliotecas que o NOX provê para o envio e construção de pacotes, quais são os componentes que lidam com eles e de que maneira a segurança da rede limitava a comunicação, uma vez que existem canais seguros na rede que só podem ser usados com autenticação. Para construção de pacotes existem a bibliotecas *packet* que fornecem classes que auxiliam na criação de inúmeros tipos de pacote (ARP,ICMP,etc.) e para o envio existem métodos presentes no conjunto de operações básicas da API de componente. Para lidar com o envio e recebimento de mensagens o NOX utiliza o componente *messenger*.

Para um conhecimento mais profundo da maneira como ocorre à transmissão e o recebimento de pacotes construiu-se um componente de envio de pacotes ICMP utilizando as ferramentas encontradas no NOX. Esses pacotes eram enviados do NOX passavam pelo switch OpenFlow que verificava que o pacote vinha do NOX e executava a ação associada a ele (no caso direcionar o pacote para todas as suas portas menos para a porta pela qual o pacote tinha entrado.) e chegavam em todos os hosts. O componente desenvolvido mostrou-se bem-sucedido na sua tarefa uma vez que todos os hosts acusavam o recebimento dos pacotes.

Enquanto o primeiro aspecto diz respeito a um nível mais baixo da comunicação da rede, o segundo aspecto diz respeito à parte alto-nível dela, a estrutura das mensagens. Uma vez que tal comunicação será realizada entre agentes autonômicos a estrutura destas mensagens deve obedecer a um padrão de comunicação entre agentes. O padrão escolhido foi o da Fundação para Agentes Físicos Inteligentes (FIPA) [O'Brien et al. 1998]. Esta escolha deve-se ao fato de este padrão ser amplamente usado pela comunidade cientifica e já ter sido implementado em outras plataformas de desenvolvimento de agentes como SPADE (desenvolvida em Python que chegou a ser utilizada por um projeto de um bolsista PIBIC do grupo do qual faço parte) e JADE (desenvolvida em Java). A diferença da nossa abordagem é o fato da implementação do padrão a ser utilizado estar focada na eficiência uma vez que as implementações anteriores foram feitas em linguagens de alto-nível.

Foram realizados estudos da linguagem C++[Deitel et al. 2005] (com foco para parte de *template* [Josuttis 1999], uma vez que se constatou que este conceito é utilizado em pontos cruciais da implementação do núcleo do NOX) e das especificações FIPA.

5. CONCLUSÕES

No decorrer deste projeto de pesquisa estudaram-se maneiras de aplicar soluções autonômicas no gerenciamento de redes utilizando um sistema operacional de redes. Verificou-se que aspectos autonômicos devem ser implementados no núcleo e quais podem ser implementados como agentes de alto-nível que estarão em espaço de usuário. Concluiu-se que as características essenciais para o gerenciamento da rede devem ser mantidas no núcleo, essas características foram mapeadas como as de auto-proteção, uma vez que o sistema operacional não pode ficar sujeito a ataques, e auto-recuperação, dado que o sistema deve se recuperar o mais rápido possível depois de um possível colapso ou ataque. Também se constatou que a plataforma de comunicação entre os agentes da rede deve ser implementada no núcleo, uma vez que ela é essencial para um bom funcionamento da rede já que para realizar as ações de gerência os agentes necessitam de uma intensa troca de informação. As especificações FIPA foram escolhidas como padrão de construção das mensagens dessa plataforma.

Contudo nem todos os objetivos principais para este projeto foram cumpridos, mas apesar disso foi possível a aprendizagem dos conceitos e técnicas de computação autonômica, sistemas multi-agentes e a maneira como os agentes se comunicam e interagem, linguagens C++ e Python, com as quais não tínhamos programado antes, conceitos de redes e sistemas operacionais de redes, assim como a experiência com o NOX e Openflow, através do desenvolvimento de componentes NOX, estudo do protocolo Openflow, utilização das ferramentas de controle de rede disponibilizadas pelos grupos de pesquisa atuante na área, técnicas de gerenciamento de rede. O conhecimento adquirido possui um papel crucial nos planos de trabalhos futuros uma vez que existe a intenção de prosseguir com esta linha de pesquisa, uma vez que esta é recente, promissora e vem adquirindo um alcance cada vez maior mundialmente, tendo ganhado, inclusive, uma notoriedade no Brasil através do último Simpósio Brasileiro de Redes de Computadores (SBRC) no qual houve um tutorial de NOX e Openflow comandado pelo grupo de Stanford responsável pela criação de tal tecnologia. Esperamos que na extensão deste projeto possamos dar a nossa contribuição na comunidade acadêmica.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[Gude et al. 2008] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, Scott Shenker. “NOX: Toward an operating system for networks”. ACM SIGCOMM Computer Communications Review: 2008.

[McKeown et al. 2008] Nick McKeown,Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner. “OpenFlow: Enabling Innovation in Campus Network”. ACM SIGCOM Computer Communications: 2008.

[Jennings et al. 2007] Brendan Jennings, Sven van der Meer, Sasitharan Balasubramaniam, Dmitri Botvich, Mícheál Ó Foghlú, William Donnelly, John Strassner. “Towards Autonomic Management of Communications Networks”. IEEE Communications Magazine: 2007. v. 1, p. 112-122.

[Heller 2009] Brandon Heller. “OpenFlow Switch Specication Version 1.0.0”. Disponível em http://www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf, 2009

[Kephart et al. 2003]Jeffrey Chess, David Chess. “The Vision of Autonomic Computing,” Computer, vol. 36, no. 1, Jan. 2003, p. 41–50.

[Dobson et al. 2006] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaiti, Erol, Gelenbe, Fabio Massacci, Padi Nixon, Fabrice Saffre, Nikita Schmidt, Franco Zambonelli. A survey of autonomic communications. ACM Transactions on Autonomous and Adaptative Systems: 2006. v. 1, p. 223-259.

[Deitel et al. 2005] Harvey Deitel, Harvey Deitel.”C++ How to Program. Fifth Edition.” Editora: Prentice Hall. 2005

[Josuttis 1999] Nicolai M. Josuttis, “C++ Standard Library: A Tutorial Reference.” Editora: Addison Wesley. 1999

[O’Brien et al. 1998] Paul O’Brien, Richard Nicol. “FIPA — towards a standard for software agents”. BT Technology Journal v. 16 No 3, July 1998.

7. CRONOGRAMA EXECUTADO

Descrição	Ago 09	Set 09	Out 09	Nov 09	Dez 09	Jan 10	Fev 10	Mar 10	Abr 10	Mai 10	Jun 10	Jul 10
Estudo: Arquiteturas Autonômicas	X	X	X	X								
Estudo: Sistemas Operacionais de Rede			X	X	X							
Mapeamento as características que podem se migradas para o núcleo do sistema operacional			X	X	X	X	X					
Elaboração de componentes de comunicação entre elementos da rede						X	X	X				
Estudo das especificações FIPA									X	X	X	
Estudo da linguagem C++ (Templates)										X	X	
Elaboração do resumo e Relatório Final										X	X	X
Preparação para apresentação											X	X