Avaliações de Sistemas de Redes Ativas : Uma experiência prática

Mauro S. P. Fonseca1,2, Nadjib Achir1, Yacine M. Ghamri Doudane1, Nazim Agoulmine1, Ahmed Mehaoua3.

1 Laboratório LIP6
Universidade de Paris VI
França

2 Pontifícia Universidade
Católica do Paraná
Brasil

3 Laboratório PriSM
Universidade de Versailles
França

ABSTRACT

The aim of the Active Networking today effort is to design, develop and implement new communication architectures that allow rapid, safe and dependable creation, reconfiguration, and deployment of advanced networking services, protocols and management components. The Active Networking paradigm achieves this goal on one hand through the concepts of (1) self-directing data units called Capsule or Smart Packets, and (2) Open Programmable Nodes. Active packets can direct their own processing and deliver new services to the interior network nodes. Programmable Nodes permit remote and secure processing and building of enhanced network services from generic network software elements. The conjunction of these concepts will ultimately result into networks with greatly improved communication services that meet user requirements efficiently and securely. This paper reviews and evaluates two major architectures of Active Networking, the well-known Active Node Transport System (ANTS) and the SwitchWare framework. We discuss the differences between the two architectures and platforms, and feedbacks from practical experiences concerning the development of the well know ping service.

Resumo

O objetivo de redes ativas é planejar, desenvolver e implementar uma nova arquitetura de comunicação, que permita a criação rápida, segura e confiável, a reconfiguração e o emprego de serviços de redes avançados, protocolos e elementos de gerência. O paradigma de redes ativas alcança este objetivo através do conceito de (1) unidades de dados auto-executáveis chamadas Cápsulas ou Pacotes Ativos, e (2) nós programáveis. Os Pacotes Ativos podem controlar seu processamento e fornecer novos serviços para os nós programáveis do interior da rede. Os nós programáveis permitem o processamento remoto e seguro e a construção de serviços de rede a partir de seus elementos genéricos. A união destes conceitos resulta em redes com grande acréscimo de serviços, que atendam aos requerimentos de eficiência e segurança. Este artigo examina e avalia as duas maiores arquiteturas de Redes Ativas, Active Node Transporte System (ANTS) e SwitchWare. Nós apresentamos as diferenças entre as duas arquiteturas e plataformas, assim como os resultados das experiências práticas realizadas com a implementação do serviço de ping.

1. Introdução

As redes ativas podem ser definidas simplesmente como a inserção de funções executáveis, controladas pelo usuário, dentro de todos os nós da rede. Este novo paradigma pode ser visto como oposto a tradicional comunicação "aplicação-a-aplicação" que é a base das arquiteturas de comunicações convencionais [1]. O principio "aplicação-a-aplicação" é utilizado para evitar a implementação de funções alto nível na rede. Tendo a próxima geração de redes que gerenciar uma grande variedade de tráfego de aplicações, o modelo de processamento de redes uniforme e passivo é claramente ineficiente para prover a qualidade diferenciada de serviços. Então, tem se demonstrado que alguns serviços de rede específicos, como vídeo multicasting, podem ser melhor suportados ou aprimorados usando as duas técnicas, (1) a informação que é somente encontrada dentro da rede (ex. o tempo e o local onde ocorre um congestionamento, localização de perdas de pacotes dentro de uma árvore de distribuição multicast, etc.), e (2) o conhecimento e a ajuda da aplicação final para reconhecer a semântica dos dados (ex. a importância e a relação entre os pacotes de vídeo I, P e B de um fluxo MPEG,…).

Deste modo, as redes de hoje podem se beneficiar do paradigma de redes ativas com o emprego dinâmico de serviços de redes que podem ser adaptados aos requisitos do usuário. Nos últimos cinco anos, diferentes visões do paradigma de redes ativas conduziu a várias implementações do conceito. Estas arquiteturas de redes ativas se diferenciam em muitos pontos. Assim, o objetivo deste trabalho é implementar e avaliar dois dos maiores sistemas de redes ativas disponíveis e avaliá-los em relação a perspectiva da rede e da aplicação. Os critérios usados para avaliar as plataformas são: a facilidade de instalação e configuração, a concepção e desenvolvimento de aplicações de redes ativas e finalmente assuntos relacionados a performance em relação a execução da aplicação e mobilidade do código.

Os dois modelos de redes ativas considerados são: o Sistema de Transporte de Nós Ativos (ANTS - Active Node Transporte System) do Instituto de tecnologia de Massachusetts, e a arquitetura SwitchWare da Universidade da Pensilvânia. Este trabalho é motivado pelo fato de que o ANTS e a arquitetura SwitchWare são facilmente disponíveis e flexíveis o suficiente para modificações e experiências, e são considerados como os

dois candidatos para desenvolver uma rede ativa para a França sobre Ipv6 dentro do projeto AMARRAGE do governo Francês .

Nas próximas duas seções, nós apresentamos a descrição do modelo ANTS e sua implementação dentro do kit de desenvolvimento,e a arquitetura SwitchWare com seus componentes associados e suas implementações. Na seção 4, a comparação de desempenho entre os dois sistemas é apresentada no contexto de uma aplicação ativa implementando o serviço ping. A Seção 5 fornece uma visão geral de outras arquiteturas de redes ativas concorrentes e na seção 6 apresentamos a conclusão do artigo.

2. Sistema de Transporte de Nós Ativos

O Sistema de Transporte de Nós Ativos (Active Node Transport System - ANTS) [2,3,4] é baseado sobre código móvel, carga sobre demanda e técnicas de cache. Ele permite que novos protocolos sejam empregados dinamicamente em nós da rede, sem sincronização ou interação entre os nós físicos e os protocolos executados. A arquitetura pode ser descrita usando os conceitos: (1) Pacotes IP Tradicionais são trocados por cápsulas que personalizam os serviços de rede; (2) Roteadores clássicos são substituídos por nós ativos que processam as cápsulas entrantes e mantém dados armazenados por um período de tempo. (3) Um mecanismo de distribuição de código é usado para empregar dinamicamente e automaticamente as rotinas requeridas nos nós.

2.1 Modelo da Cápsula

No ANTS, a cápsula (Fig. 1) é um específico pacote IP que encapsula dados do usuário e referencia uma rotina de transferência (forwarding) associada. A rotina de transferência é selecionada pela aplicação no nó de origem. Esta rotina é executada por todos os nós ativos onde a cápsula passa, e caso seja um nó convencional, a rotina clássica de transferência IP é executada. Estas rotinas de transferência utilizam a API fornecida pelo ANTS para acessar o ambiente do nó, manipular os dados da aplicação (um objeto guardado por um curto período de tempo em um nó) e finalmente encaminhar as cápsulas para outros nós. Usuários concorrentes podem empregar diferentes rotinas de transferência sem conflitos, pois existe uma associação explícita entre as rotinas de transferência e as cápsulas.

Como as cápsulas viajam através dos nós da rede, o código correspondente é enviado para estes nós.

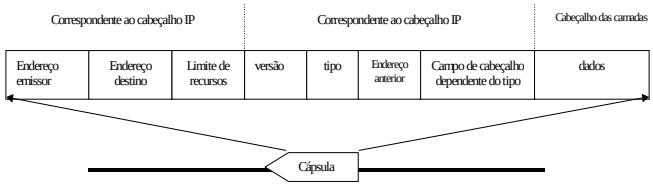

Figura 1. Formato da cápsula.

2.2 Distribuição de Código

Para considerações de desempenho, o código de rotinas de transferência é enviado e armazenado nos nós ativos, o mesmo é transportado em cada cápsula do fluxo. O argumento é que rotinas de transferência de uso geral serão freqüentemente executadas por cápsulas subseqüentes, pertencentes a mesma aplicação. Uma aplicação do usuário pode iniciar o uso de um novo protocolo a qualquer momento pelo registro da definição do código na máquina local. Novas cápsulas podem ser injetadas na rede e também podem ser recebidas da rede. Como as cápsulas passam através dos nós da rede, o código correspondente é enviado para estes nós. Para facilitar e acelerar o envio do código, cápsulas são reunidas em grupos de código dependentes entre si. Assim, se um tipo de cápsula fizer referência a outra, suas definições serão agrupadas para serem transferidas juntas. Conseqüentemente o transporte do código é executado com base no grupo de cápsulas e não no código das cápsulas individuais.

O protocolo de transporte de código (Fig. 2) trabalha como descrito abaixo:

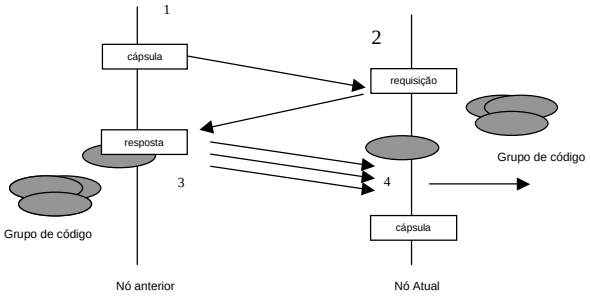

Figura 2. Demanda de um grupo de código.

Na recepção de uma cápsula em um nó, a rotina de transferência referenciada é verificada no cache de código. Se o código não estiver presente, a execução da cápsula é suspensa e uma requisição de código é enviada para o nó emissor. A requisição é baseada em tipos de cápsulas e protocolos.

Quando um nó recebe uma requisição de código e tem o código correspondente, ele o envia imediatamente para o nó requerente.

Quando um nó recebe um código requisitado, ele o coloca em seu cache de código e executa a cápsula suspensa. Caso não receba, a cápsula é simplesmente descartada.

Esta estratégia de distribuição de código, restringe sua distribuição em localizações específicas da rede, ou seja, somente na rota ativa é que haverá o código.

2.3 Nó Ativo

O principal desafio para redes ativas é garantir a segurança do ambiente e do código do novo serviço na rede. Na arquitetura ANTS, os nós ativos seguem a seguinte regra. Estes nós exportam um conjunto de primitivas (API) para serem utilizadas pelas aplicações e cápsulas. Eles também provêem um modelo de execução que suporta estas primitivas que viabilizam os objetivos de segurança e administração de recursos. Estes dois

mecanismos permitem que o código venha de qualquer fonte e evitem erros acidentais e intencionais.

2.4 Primitivas dos Nós

O ANTS fornece um pequeno e básico conjunto de primitivas (API) para a definição de rotinas de transferência de cápsulas para todos os nós ativos. A combinação destas primitivas básicas permite a definição de várias e complexas rotinas de transferência. Estas primitivas são agrupadas em cinco categorias:

(1) Acesso ao ambiente: para acessar o estado da comunicação e a tabela de roteamento do nó local. Um exemplo é para descobrir o nó vizinho, que pode ser realizado pela execução de primitivas internas do nó.

(2) Manipulação da cápsula: com acesso aos campos de cabeçalho e dados;

(3) Operações de controle: para permitir o envio, descarte e suspensão/execução das cápsulas por elas mesmas.

(4) Cache: para armazenar e acessar estados, na forma de objetos com definição do tempo de validade, para curtos períodos. Então cápsulas que fazem parte do mesmo protocolo podem se comunicar com outras pela criação de objetos de dados dentro dos nós visitados.

(5) Reunião: para permitir que cápsulas se encontrem com outras e coordenar suas ações.

2.5 Exemplo ANTS

Nós consideramos o seguinte exemplo, o mais básico dos diagnósticos de rede o ping. Atualmente isto é feito por um tipo de pacote especial no protocolo ICMP [5]. No ANTS o pacote ping foi alterado por uma cápsula ping ativo. Esta cápsula contém o método evaluate que é executado a cada nó ativo visitado pela cápsula até o nó de destino ser alcançado. O algoritmo é o seguinte:

```
if ( n.getAddress() == getDst()) {
     ping = true;
   } else if (ping!=true) {
     return n.routeForNode(this, getDst());}
```

3. A Arquitetura SwitchWare

O SwitchWare [6] utiliza uma arquitetura em camadas que abrange diferentes características como flexibilidade, segurança, confiança, performance, e usabilidade. Como mostrado nas próximas três seções, estas funcionalidades estão divididas entre as camadas de acordo com a relação flexibilidade e segurança necessária a cada camada. As camadas mais altas garantem níveis de alta performance e segurança, enquanto as mais baixas são caracterizadas por pouca funcionalidade e baixo desempenho, devido aos excessivos controles de segurança. O SwitchWare define três camadas: (1) a camada de pacotes ativos "active packet", (2) a camada de extensões ativas "active extension", e a camada de infraestrutura de roteador ativo seguro "secure active router infrastructure".

3.1 Pacotes ativos

Pacotes ativos transportam programas compostos de dados e códigos que substituem os pacotes clássicos com cabeçalho e dados. O código, parte de um pacote ativo é habilitado a executar as funções de um cabeçalho com muito mais flexibilidade, uma vez que ele pode interagir com o ambiente do roteador de uma forma muito mais complexa e personalizada do que uma procura na tabela de roteamento. Os dados dentro de um pacote ativo são apresentados em forma de uma estrutura personalizada que pode ser facilmente manipulada pelo programa.

A linguagem de programação usada por esta camada é o PLAN (Linguagem para pacotes de redes ativas - Packet Language for Active Networks) [7]. Como examinado na seção 3.3, o PLAN é uma linguagem muito simples e leve. Isto permite uma execução com recursos limitados, mas sem a necessidade de autenticação, apesar de permitir uma funcionalidade mínima, podem ser executadas ações autorizadas quando necessário. Para superar estas restrições, programas PLAN utilizam rotinas de serviços residentes no nó (Extensões ativas), que implementam os mecanismos mais pesados para prover a segurança necessária.

O modelo de execução do PLAN inclui um mecanismo para executar programas remotamente sobre outros roteadores. Este mecanismo é a forma pela qual o PLAN transporta os próprios programas através da rede. Os programas PLAN são compostos de três partes distintas: (1) um código, (2) uma identificação das funções que devem ser executadas por primeiro quando o programa chega no roteador; (3) dados que são os argumentos da função. O PLAN é considerado como uma linguagem fortemente tipada que fornece uma execução segura através da verificação estática do tipo antes da introdução na rede.

3.2 Extensões ativas

As extensões ativas, também chamadas de "switchlets", são associadas a camada do meio da arquitetura SwitchWare. Elas podem ser parte dos componentes do roteador ou serem carregadas dinamicamente. Apesar disso, elas não são móveis e para se comunicarem com outros roteadores elas devem usar pacotes ativos. Os " Switchlets " são inseridos dentro de pacotes ativos que podem implementar protocolos ou funcionalidades variadas. Como eles são invocados somente onde e quando necessários, não existe a obrigação que as extensões sejam leves. Esta característica permite várias composições, assim sendo protocolos complexos podem ser implementados no SwitchWare com a combinação dos programas PLAN e extensões ativas.

Dentro da corrente implementação, os "switchlets" são escritos em Caml. Eles utilizam a verificação estática de tipo, quando os pacotes chegam no roteador e podem usar uma variedade de mecanismos de segurança, incluindo autenticação baseada em criptografia e verificação de programa. Dessa forma, eles podem criar ou alterar estados do roteador e ter acesso direto as interfaces de rede do roteador.

A mais completa implementação desta camada é a ponte ativa "active bridge" [8]. A ponte ativa é programada em Caml, que oferece diversas vantagens para esta aplicação. Primeiro, como Java, Caml gera bytecode que é lido dinamicamente e é independente de máquina (permitindo assim extensões ativas).

Essencialmente, Caml prove de forma limitada, mas adequada, o controle sobre o espaço de nomes visíveis para os módulos lidos dinamicamente. Quando isto é combinado com a forte tipagem de Calm, permite um grande e preciso nível de controle de recursos entre as extensões concorrentes do roteador.

3.3 Linguagem PLAN

Qualquer técnica de redes ativas deve balancear a força entre os seguintes tópicos: (1) flexibilidade, porque o objetivo de redes ativas é permitir usuários ou aplicações de personalizar a rede; (2) segurança e confiança, principalmente devido a natureza de compartilhamento da rede (internet); (3) Desempenho, porque a inclusão de novas funcionalidades deve ser feita sem comprometer o desempenho já oferecido pela rede; e finalmente, (4) a usabilidade.

Com respeito aos requisitos descritos anteriormente, o projeto SwitchWare define uma técnica baseada em dois níveis distintos, entre a interoperabilidade de camadas baseada na nova linguagem de programação PLAN [9] e o nível de serviços residentes no nó, o qual pode ser escrito em outra linguagem de propósitos gerais. Cada pacote contém programas PLAN que substituem o cabeçalho e os dados. A combinação destes programas e a chamadas de serviços, produzem funções de redes mais complexas.

3.3.1 Gerenciamento de segurança e recursos

O PLAN segue a política de não adicionar características a linguagem a menos que : (1) isto seja necessário para aplicações, (2) isto não comprometa a segurança, ou (3) isto aumente a usabilidade. As principais características estão resumidas a seguir:

- PLAN é uma linguagem fortemente tipada e verificada dinamicamente.

- Possui um controle de fluxo simples baseado em construção de instruções em seqüência, execução dependente, interação sobre lista, e exceções.

- Chamadas de funções são suportadas mas sem recursividade. Logo a carência de recursões e interações ilimitadas garantem que todos os programas PLAN terminem.

No PLAN a quantidade máxima de recursos consumidos por um simples pacote é armazenada em um campo RB (Resource Bounds - Limite de Recursos). Este campo, parte do cabeçalho do pacote, é decrementado cada vez que o pacote vai para um nó ou cada vez que ele é duplicado para criar um novo pacote (pela atribuição de algum RB para o novo pacote). O RB pode ser associado aos seguintes parâmetros: largura de banda, memória, ciclos de CPU. Limitando este parâmetro é possível fixar contadores para CPU e memória em cada nó para restringir o custo do recurso [10].

3.3.2 Exemplo PLAN

Nesta seção é apresentado um código PLAN para a função ping. Com o SwitchWare, o ping do protocolo ICMP [5] é substituído pelo programa PLAN em um pacote ativo que é enviado do nó emissor até o receptor. O algoritmo é o seguinte:

```
Fun ping (src:host, dest:host) : unit =
   if (not thisHostIs(dest)) then
     OnRemote(|ping|(src,dest), dest, getRB(), defaultRoute)
   else
    OnRemote(|ack|(), src, getRB(), defaultRoute)
Fun ack() : unit = print("Success");
```

3.4 Implementação

PLANet [11] é uma implementação de rede ativa baseada na arquitetura PLAN. Ele implementa o serviço de camada de rede diretamente sobre a camada de ligação (Ethernet, ATM, Frame Relay, …). PLANet é ativo por dois motivos: (1) PLANet usa pacotes ativos com programas PLAN ao invés dos tradicionais cabeçalhos e fornece o controle de como os pacotes operam na rede. (2) Uma vez que os programas PLAN são pequenos, eles não podem expressar toda a funcionalidade necessária, logo os programas PLAN devem utilizar rotinas de serviços disponibilizadas nos nós. Estas rotinas específicas, chamadas de extensões ativas, são adicionadas aos nós dinamicamente. Extensões ativas incrementam ou aperfeiçoam funcionalidades existentes. Assim sendo, os programas PLAN podem ser comparados com comunicação inteligente entre nós adjacentes. Para determinar o próximo passo para um pacote PLAN, um nó intermediário deve executar o campo RouteFun do pacote, que informa o serviço de transferência necessário. Conseqüentemente um processamento leve é executado para encontrar o próximo nó evitando uma verificação dos dados no pacote. O único custo adicional é o de encontrar a rotina de transferência. PLANet implementa uma rotina de transferência padrão (DefaultRoute) que é parecido com o RIP (Routing Information Protocol) [12].

4. Arquiteturas ANTS Vs. SwitchWare

Para comparar as duas arquiteturas de redes ativas, nós implementamos o serviço ping, que é simples mas é o mecanismo freqüentemente usado para avaliar desempenhos de arquiteturas de redes ativas. A avaliação consiste em mandar vários pacotes de ping sucessivos em uma rede ativa composta de três nós ativos (um emissor, um roteador e um destino) .

Para implementar este serviço de rede com os modelos ANTS e SwitchWare, nós usamos o ANTS Toolkit e PLANet respectivamente. Como mostrado na seção 3.4, PLANet implementa um subconjunto da arquitetura SwitchWare, ou seja as duas camadas superiores (Active Packet e Schwitchlet), omitindo a camada inferior com características de segurança. A escolha do PLANet também é justificada pelo fato de que, é a implementação mais completa da arquitetura SwitchWare no momento [13,14]. Apesar da disponibilidade de implementações distintas de cada camada [15], a integração das três camadas é complexa e ineficiente.

Inicialmente, a principal diferença que nós podemos identificar entre as duas arquiteturas diz respeito ao processo de distribuição de códigos entre nós ativos. O ANTS é baseado em "nós ativos", isto é, o pacote não carrega o código, mas somente identificadores ou referências para funções predefinidas serem utilizadas dinamicamente. Em oposição, o SwitchWare usa uma

técnica híbrida baseada em "nós e pacotes ativos", isto é, os pacotes podem carregar códigos que são relativamente simples e restritos, enquanto os nós ativos podem fornecer qualquer código complexo que é carregado dinamicamente se necessário.

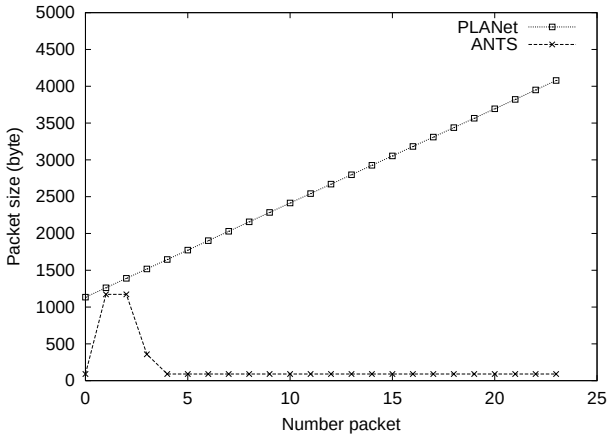

Figura 3. Tamanho do pacote Vs Número do pacote.

A Fig. 3 mostra o efeito da transferência de código com o ANTS e o PLANet. Com o ANTS, a transmissão da primeira cápsula gera transferências adicionais de cápsulas contendo o código faltante do ping no nó de destino. Com o PLANet, o código já esta dentro do pacote, e conseqüentemente não é necessário transmissões de cápsulas adicionais. Também pode ser dito que o tamanho dos pacotes PLAN de 1134 bytes, são maiores que os 91 bytes da cápsula ANTS.

O transporte do código ativo em todos os pacotes, é justificado como um serviço de redes baseado em pacotes, como o roteamento de dados dinâmico. Entretanto, isto é um desperdício de recursos em serviços de rede baseado em fluxo como o ping. Pelo contrário, de acordo com o processo de transferência do ANTS, topologias de rede com grandes alterações dinâmicas ou fluxos pequenos com limitados números de cápsulas levam a várias transferências de código que sobrecarregam muito as ligações da rede e conseqüentemente reduzem o desempenho global.

Nós acreditamos que é possível otimizar o tamanho do pacote PLAN. Na atual implementação, o código ativo é transportado no formato ASCII em todo o pacote sem otimizações, isto é, caracteres não utilizáveis estão presentes, como o espaço e linhas em branco do código. Apesar disso, o uso do bytecode PLAN, que interpretado nos roteadores ativos, pode ser a solução para este problema.

Como ilustrado na Fig. 3, o tamanho do cabeçalho dos pacotes PLAN aumentam linearmente com o número de passagens pelos roteadores, na proporção de 32 bytes por vez. Nós também notamos que para o mesmo serviço, ping, o código PLAN é mais compacto que o bytecode Java usado no ANTS. Realmente, o código PLAN é transportado inteiramente dentro de um simples pacote de 1173 bytes, enquanto o bytecode Java ANTS é transportado usando 3 cápsulas diferentes utilizando respectivamente 1173 bytes, 1173 bytes e 358 bytes.

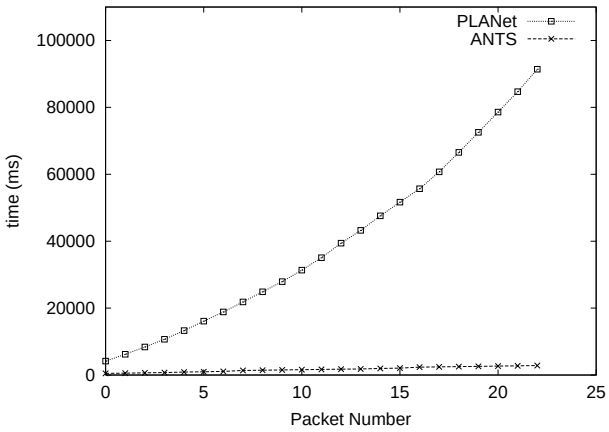

Figura 4. Número do pacote Vs Tempo.

A Fig. 4 apresenta o tempo decorrido entre o início da seção e a recepção do resultado dos pacotes ping no nó emissor. O início da seção corresponde a transmissão do primeiro pacote do ping. Nós podemos dizer que o tempo gasto na rede, incluindo o tempo de CPU, é menos significativo no ANTS que no PLANet. Este tempo é fortemente proporcional ao tamanho do pacote que atravessa a rede.

Outras diferenças entre os dois modelos de rede ativa estão resumidas na tabela 1.

Critério de comparação	**ANTS**	**SWITCHWARE**
Ambiente de execução	Máquina virtual Java	interpretador PLAN / JDK ou OCaml
Requerimentos	JDK 1.0.2	Bibliotecas JDK 1.1.X e PIZZA.
Linguagem Programação	Java	PLAN / OCaml ou Java
Camada de ligação	UDP	UDP (entre os nós ativos) / TCP (entre aplicações e os primeiros nós ativos)
Distribuição de código	Distribuição automática do código por cápsulas em canais separados.	Explícito
Roteamento	Roteamento estático.	Roteamento estático e dinâmico.
Tempo de vida da cápsula/pacote	Definido pelo usuário na criação da cápsula.	Limite de recursos definidos na criação do pacote.
Uso do cache	Diminui o tempo de vida quando do uso do cache.	Não existe.
Ciclo de CPU	Sem controle.	Sem controle.
Proteção do nó	Baseado no mecanismo de segurança Java.	Baseado em mecanismo de autenticação e namespace.
Proteção do código	Via autenticação .	Via namespace.

Tabela 1. Comparação entre ANTS e SwitchWare.

5. Trabalhos relacionados

Redes ativas constituem uma nova técnica para aumentar a Qualidade de Serviço (QoS) pela possibilidade de maior processamento inteligente dentro dos nós da rede. Muitos grupos de pesquisa tem apresentado diferentes arquiteturas para demonstrar a usabilidade e a força do paradigma de redes ativas. Neste documento nós apresentamos somente duas destas arquiteturas, o ANTS e o SchwitWare. A seguir é apresentado sucintamente as outras arquiteturas.

- O sistema M0 [16], fornece uma arquitetura de rede ativa baseada em código móvel. Mensageiros "Messengers " foram

propostos para substituir o tradicional paradigma de trocas de mensagem disponíveis nas atuais arquiteturas de rede. Mensageiros são trocados entre nós e contém programas para serem executados nos nós específicos. O ambiente M0 é implementado como uma ferramenta para identificar serviços mínimos que os nós ativos devem fornecer, e como o ambiente de programação era baseado em instruções de comunicação, pode ser estudado como rede ativa, sistemas operacionais distribuídos e inteligência artificial distribuída.

- A arquitetura Tempest desenvolvida na Universidade de Cambridge [17] permite que muitas arquiteturas de controle de rede executem simultaneamente no mesmo switch de rede ATM. Cada rede virtual pode ser gerenciada por diferentes e descentralizados operadores.

- O projeto Netscrip da Universidade Columbia [18] objetiva definir um ambiente virtual que forneça uma abstração completa de um ambiente de rede programável. As aplicações de comunicação podem ser desenvolvidas em uma linguagem comum neutra que forneça funcionalidades de redes para acesso universal.

6. Conclusão

O objetivo do paradigma de redes ativas é de produzir uma nova plataforma de rede aberta, flexível e extensível em tempo de execução para comportar rapidamente as evoluções e desenvolvimentos de tecnologias de rede, e também para aumentar os serviços complexos requisitados pelas aplicações dos usuários [19].

Neste documento, nós fizemos uma pesquisa e comparamos a funcionalidade e o desempenho de duas grandes arquiteturas de redes ativas chamadas ANTS e SwitchWare. Nós testamos suas habilidades em utilizar serviços de rede de forma flexível e dinâmica. Também foi estudado a implementação e a arquitetura dos dois sistemas dentro do contexto de um serviço ativo, o ping. A principal conclusão da avaliação é que existe uma relação direta entre o desempenho, a simplicidade e outros aspectos como a segurança e o gerenciamento de recursos. O modelo SwitchWare se preocupa mais que o ANTS com a segurança e o gerenciamento de recursos, mas também recebe uma alta penalidade no processamento nos nós. Estes sistemas utilizam duas técnicas diferentes para distribuir as rotinas de transferência controladas pelo usuário no nó ativo. A avaliação do desempenho mostra que a técnica de "out of band" do ANTS é melhor apresentado que a técnica "in-band" do SwitchWare, por minimizar a quantidade de dados de controle enviados na rede durante toda a seção de comunicação. Por outro lado sistemas com seções rápidas terão melhor desempenho usando o sistema SwitchWare.

7. Referências

[1] S. Bhattacharjee, K. Calvert, E. Zegura, “Active Networking and the end to end argument”, in Proceedings of ICNP`97.

[2] D. Wetherall e al., “ANTS : A Toolkit for Building and Dynamically deploying Network Protocols”. in Proceedings of IEEE OPENARCH'98, São Francisco, CA, 1998.

[3] D. Wetherall, “Service Introduction in an Active Network”, tese de Ph.D., disponível como MIT/LCS/TR-773, 1999.

[4] D. Wetherall, “Developing Network Protocols with the ANTS Toolkit”, Design Review, 1997.

[5] J. Postel “Internet Control Message”, RFC 792, ISI , 1981.

[6] D. S. Alexander, W. A. Arbaugh, M. W. Hicks, P. Kakkar, A. D. Keromytis, J. T. Moore, C. A. Gunter, S. M. Nettles, and J. M. Smith, “The SwitchWare Active Network Architecture”, IEEE Network Special Issue on Active and Controllable Networks, 1998, vol. 12 no. 3, pp. 29 - 36.

[7] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter and S. Nettles, “PLAN: A Packet Language for Active Networks”, Proceedings of the International Conference on Functional Programming (ICFP) '98.

[8] D. S. Alexander, M. Shaw, S. M. Nettles, and J. M. Smith. “Active Bridging”. Proceedings of the ACM SIGCOMM'97 Conference, Cannes, França, 1997.

[9] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter e S. Nettles, “PLAN: A Packet Language for Active Networks”, Proceedings of the ICFP '98.

[10] P. Menage, “RCANE: A Resource Controlled Framework for Active Network Services”, in the Proceedings of the First International Working Conference on Active Networks (IWAN'99), Berlin, Alemanha, 1999.

[11] M. Hicks, J. T. Moore, D. S. Alexander, C. A. Gunter, e S. M. Nettles, “PLANet: An Active Internetwork”, Proceedings of the IEEE INFOCOM'99, Nova York, USA.

[12] C. Hedrick, “Routing Information Protocol”, Tech. Rep., RFC 1058, 1988.

[13] D. S. Alexander, W. A. Arbaugh, M. W. Hicks, P. Kakkar, A. D. Keromytis, J. T. Moore, C. A. Gunter, S. M. Nettles, e J. M. Smith, “The SwitchWare Active Network Architecture”, IEEE Network Special Issue on Active and Controllable Networks, 1998, vol. 12 no. 3, pp. 29 - 36.

[14] D. S. Alexander, M. W. Hicks, P. Kakkar, A. D. Keromytis, M. Shaw, J. T. Moore, C. A. Gunter, T. Jim, S. M. Nettles, e J. M. Smith, “The SwitchWare Active Network Implementation”, The 1998 ACM SIGPLAN Workshop on ML held in conjunction with the International Conference on Functional Programming (ICFP) '98.

[15] A pagina WEB do projeto SwitchWare, “http://www.cis.upenn.edu/~switchware/”.

[16] Ch. Tschudin, “The Messenger Environment M0 – A Condensed Description”, In: Mobile Object System – Towards the Programmable Internet, J. Vitek and Chr. Tschudin (eds.), Springer LNCS 1222, 1997, pp. 149-156.

[17] S. Rooney, “An Innovative Control Architecture for ATM Networks”, Integrated Network Management V, 1997.

[18] Y.Yemini,S. Da Silva, “Towards Programmable Networks”, in Proceedings of IFIP/IEEE International Workshop on Distributed Systems, Operation and Management, Italia, 96.

[19] K. Psounis, “Active Networks : Applications, Security, Safety, and Architectures”, 1999, IEEE Communications Surveys.