Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Mestrado em Informática

Disciplina: Redes de Computadores
Professora: Luci Pirmez

O estado da arte da pesquisa em redes ativas

Abril de 2000

Autores:
Edmundo Lopes Cecilio
Reinaldo de Barros Correia

Glossário

AA	Aplicação Ativa.
ABone	Active Backbone
AE	Ambiente de Execução.
ANANA	Active Networks Assigned Numbers Authority
ANEP	Active Network Encapsulation Protocol
ANETD	Active Network Daemon
ANMP	Active Network Message Protocol
ANN	Active Network Node
ANTS	Active Node Transfer System
ARP	Active Reservation Protocol
ASP	Active Signaling Protocol
ATM	Asynchronous Transfer Mode
BBN	Bolt, Beranek and Newman
DAN	Distributed code caching for Active Networks
DARPA	Defense Advanced Research Projects Agency
FEC	Forward Error Correction
FPGA	Field Programmable Gate Arrays
IP	Internet Protocol
IPP	Input Port Processor
IPvX	Internet Protocol version X
LC	Line Cards
LI	Link Interface
MBone	Multicast Backbone
OPP	Output Port Processor
PAN	Pratical Active Network
PE	Processing Element
PLAN	Packet Language for Active Networks
RFC	Request For Comments
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
SANE	Secure Active Network Environment
SE	Switching Element
TCL	Tool Command Language
TCP	Transmission Control Protocol
TLV	Type – Length – Value
UDP	User Datagram Protocol
VNE	Virtual Network Engine

Sumário

Resumo

Redes ativas representam um significativo passo na evolução das redes de comutação de pacotes. O tradicional encaminhamento dos pacotes é substituído por uma funcionalidade mais geral que permite controle e modificações dinâmicas do comportamento e do estado dos nós da rede. O principal objetivo da tecnologia é a rápida disponibilização de novos serviços. A pesquisa na área é recente e a comunidade ainda está dividida quanto a questões como risco de baixo desempenho devido à complexidade e necessidade de rigorosos esquemas de segurança e proteção. O propósito desse trabalho é apresentar o estado da arte na pesquisa do assunto. Será descrito o modelo de referência que pode-se dizer que foi adotado pela comunidade e serão resumidos as principais categorias de áreas de pesquisa os principais projetos em andamento. Adicionalmente, algumas questões ainda em aberto serão apontadas.

1. Introdução

A tecnologia de redes de computadores em franca utilização nos dias de hoje é baseada em nós, sejam roteadores (comutação de pacotes) ou comutadores (comutação rápida de pacotes), verticalmente integrados. Os programas, sistema operacional e hardware desses nós são completamente integrados entre si e fornecidos pelo mesmo fabricante. O máximo que se obtém em termos de flexibilidade é a atualização dos códigos dos programas instalados via operações manuais de gerenciamento dos administradores das redes ou dos próprios fornecedores dos equipamentos.

Outro fator importante é que nas tecnologias em utilização atualmente, o ciclo desde a criação de um novo protocolo ou mesmo de um novo serviço para ser implementado em um protocolo existente até a sua utilização em larga escala pode levar uma década.

A capacidade de processamento disponível nos nós de comutação vem aumentando significativamente. A eletrônica tem seus preços reduzidos em intervalos cada vez menores. Já é possível se instalar quantidades muito grandes de memórias voláteis e não voláteis em roteadores e comutadores. Essa característica possibilita a implementação de complexas computações nos fluxos de pacotes que passam por um nó.

Da mesma forma, o surgimento de novas técnicas de desenvolvimento e utilização de softwares, como encapsulamento, transferência e execução seguras e eficientes, a interposição de programas e/ou fragmentos de programas e a utilização de código intermediário abre novos horizontes na área de software.

O desenvolvimento de novas aplicações que lidam com tráfegos multimídia e suas exigências quanto a qualidade de serviço, como videoconferências e voz sobre IP, associado à crescente necessidade de segurança vêm exigindo que mais e mais aplicações sejam instaladas no interior da rede. As seguintes categorias de aplicações podem ser citadas como exemplo: controle de congestionamento, controle de tráfego, roteamento em multicast, caching de servidores web, proxies e filtragens, entre outras.

Unindo-se os fatores apresentados, a saber, a necessidade de maior rapidez na instalação de novos serviços, a maior disponibilidade de capacidades de processamento e armazenamento e a necessidade da presença de mais aplicações no interior da rede, pode-se dizer que surge a clemência por um novo paradigma, dinâmico, de instalação de software nos nós de comutação das redes.

O conceito de redes ativas nasceu em discussões da comunidade de pesquisa do DARPA nos EUA, na época (1994 e 1995) que eram justamente questionadas as futuras direções das redes de computadores.

As futuras direções podem, atualmente, ser concentradas em duas grandes opções: as redes programáveis e as redes ativas. Na primeira opção a programação é realizada através de operações de gerenciamento executadas por administradores. Na segunda, a programação é realizada através da execução de códigos transportados nos próprios fluxos de pacotes ou sob demanda, mediante solicitações presentes também nos fluxos de pacotes.

Esse trabalho tem como objetivos explicar o paradigma das redes ativas e apresentar o estado da arte na sua pesquisa. Inicialmente, será apresentado o modelo genérico para a arquitetura de um nó ativo. Serão feitas as definições necessárias e tecidas considerações sobre os níveis de software e o hardware presentes nesse tipo de nó. Na seção seguinte será discutido o estado da arte da pesquisa em redes ativas. São citadas as categorias nas quais, de acordo com o modelo genérico, a pesquisa se divide e são explicados os principais projetos existentes. Ainda nessa seção, são apresentadas e comentadas as questões em aberto de maior relevância no que concerne a funcionalidade das redes ativas como um todo.

2. O modelo de referência adotado

O modelo apresentado reflete o consenso que a comunidade do DARPA alcançou após diversos congressos realizados até o ano de 1999 [1], [5], [6].

Convém destacar que o modelo aqui apresentado diz respeito apenas à arquitetura do nó (ativo) de comutação de uma rede ativa, ou seja, define como os pacotes são processados e os recursos do nó são alocados. Uma arquitetura da rede propriamente dita engloba aspectos como endereçamento e alocação de recursos fim a fim. Esse modelo de arquitetura de nó ativo adotado foi concebido visando a instalação e o uso de mais de uma arquitetura de rede em paralelo.

2.1. Visão geral do modelo

Uma rede ativa consiste de um conjunto de nós interconectados por qualquer tecnologia de nível de enlace. Nem todos os nós da rede precisam ser ativos. Quando esse for o caso, deve haver o tunelamento entre dois nós ativos adjacentes, a exemplo do que ocorre no MBone. Cada nó ativo é composto por um hardware, por um sistema operacional, por um ou mais ambientes de execução e por aplicações ativas. O sistema operacional é o responsável pela alocação e escalonamento dos recursos do nó, como, por exemplo, banda de transmissão no enlace, ciclos de CPU e capacidades de memória e armazenamento. Os ambientes de execução atuam como interface entre as aplicações (chamadas de aplicações ativas) que são executadas em benefício dos pacotes e o sistema operacional do nó. Cada ambiente de execução implementa uma máquina virtual que interpreta o conteúdo dos pacotes ativos que chegam ao nó. Os ambientes de execução podem ser desde completamente genéricos, ou seja, capazes de executar qualquer programa, (como uma máquina de Turing completa) até especializados em uma única tarefa, como, por exemplo, encaminhar pacotes baseado em cabeçalhos do protocolo IPv4. Os usuários obtêm serviços fim a fim de uma rede ativa através de aplicações ativas que programam as máquinas virtuais (uma em cada nó da rota) providas pelos ambientes de execução.

Os conceitos que foram apresentados podem ser ilustrados com a seguinte comparação: um nó ativo pode ser implementado em uma estação de trabalho do tipo RISC (o hardware), com o seu sistema operacional (o sistema operacional do nó), uma máquina virtual Java (o ambiente de execução) recebendo, processando e encaminhando pacotes que contêm programas escritos em Java em seu interior (as aplicações ativas).

O funcionamento do nó ativo pode então se resumido nas seguintes etapas:

1. Chegada de um pacote no nó ativo.
2. Classificação desse pacote em um ambiente de execução.
3. Execução de uma aplicação ativa em benefício desse pacote.

4. Encaminhamento do pacote à sua porta de saída.

Premissas básicas

Na consolidação do modelo é assumido que:

- A unidade de multiplexação é o pacote, ou seja, os serviços implementados são de comutação de pacotes e não de circuitos.
- A função primária da rede ativa é a de comunicação e não a de computação, ou seja, o nó ativo não é projetado para servir a um sistema de computação distribuído.
- A tecnologia de enlace é independente do nó ativo e pode evoluir continuamente.
- Cada nó é controlado por uma administração, sendo que não é necessário que nós constituintes de uma rota pertençam à mesma administração.
- Há relacionamentos de confiança entre nós.

Objetivos

O modelo busca os seguintes objetivos para as redes ativas:

- Minimizar a necessidade de padronização.
- Ter compatibilidade reversa com nós tradicionais, ou seja, o simples encaminhamento de pacotes IPv4 deve ser possível de ser viabilizado/realizado.
- Apresentar pelo menos o mesmo desempenho do que os nós das redes tradicionais quando da utilização de serviços legados, como IPv4 ou IPv6.
- Permitir experimentações em paralelo com a utilização normal.
- Ser escalável para redes ativas globais.
- Prover mecanismos de segurança e proteção ao nó ativo.
- Oferecer o suporte ao gerenciamento em todos os níveis.
- Prover mecanismos para oferecer suporte a diferentes níveis, qualidades e classes de serviço.

2.2. O modelo de referência

O modelo genérico de um nó ativo que será usado como base desse trabalho é apresentado na Figura 1.

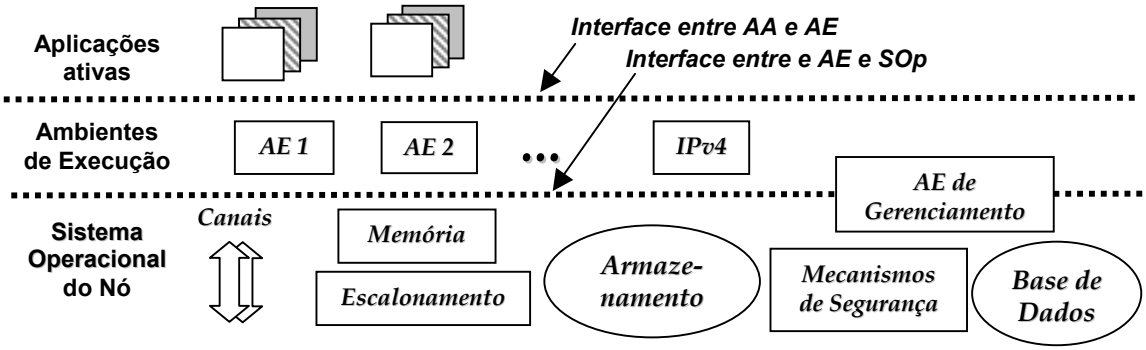

Figura 1 - Modelo genérico de nó de comutação ativo

Aplicação ativa (AA)

Uma aplicação ativa é um programa que, quando executado pela máquina virtual fornecida por um ambiente de execução, implementa um serviço com determinadas características. Conforme já foi citado, os usuários se utilizam de uma seqüência das mesmas aplicações ativas, uma em cada nó da rota, para obter o serviço fim a fim desejado. Os detalhes de como esses programas (as AA) são carregados nos nós relevantes da rede são determinados pelo ambiente de execução da AA em questão. O código do programa pode ser carregado nos pacotes – in-band – ou ser carregado quando necessário – out-of-band. Nessa última abordagem, o carregamento pode ser realizado sob demanda, na chegada do pacote, ou em uma fase de sinalização anterior ao envio dos pacotes. Essas duas abordagens serão discutidas em detalhes posteriormente.

Aplicações ativas podem também já estar inteira ou parcialmente carregadas em um nó. Nesse caso, os pacotes podem ou não adicionar facilidades a elas.

Os ambientes de execução permitem a execução de mais de uma AA – ou mais de um serviço – por vez.

A linguagem (presente no pacote) utilizada para a programação da AA pode variar desde o simples uso de parâmetros que permitam a seleção de escolhas predefinidas até a utilização de uma linguagem capaz de implementar qualquer computação. No entanto, linguagens especiais, com restrições, podem ser utilizadas em função dos possíveis problemas que podem decorrer do uso de uma linguagem completa, que possibilite a construção de qualquer computação. Alguns dos problemas pode ser travamento do programa e/ou alocação excessiva de recursos. Exemplos dessas restrições são a ausência ou limitação da existência de loops, a verificação estática de tipos e o pouco acesso à memória.

A granularidade da execução de uma AA pode variar desde uma computação exclusiva para o pacote que a instalou até a alteração do comportamento do nó para todos os pacotes por que passam por ele.

Ambiente de execução (AE)

O AE pode ser comparado ao programa shell utilizado em sistemas operacionais de computadores genéricos. É através do AE que os recursos do nó podem ser acessados pelas aplicações ativas. Múltiplos AE devem estar presentes em um nó. No entanto, o desenvolvimento e a instalação de AE são consideradas tarefas não triviais e, portanto, não é esperado que venha a existir um grande número de AE em funcionamento em um nó.

Visando a implementação de compatibilidade reversa com protocolos legados, ambientes de execução não ativos como, por exemplo, para a execução do protocolo IPv4, podem existir.

A função da máquina virtual implementada por um AE não é definida no modelo. É tarefa do desenvolvedor do AE divulgar quais são as funcionalidades disponíveis no mesmo e como elas devem ser acessadas.

A instalação de AE é de responsabilidade do administrador do nó, podendo ser realizada via AE de gerenciamento, ou seja, à distância. Convém destacar que ambientes de execução não são instalados por pacotes ativos. Isso pode acontecer apenas com as aplicações ativas. Obviamente medidas de segurança como autenticação e verificação de permissões devem ser observadas antes de que tal instalação seja permitida.

Sistema operacional do nó

A existência de um sistema operacional advém da necessidade de permitir a existência simultânea de vários AE. É o sistema operacional quem facilita e gerencia o acesso aos recursos do nó, sejam compartilhados ou não. Os recursos implementados sob a proteção do sistema operacional são aqueles relacionados às funções de armazenamento, acesso à memória, transmissão e processamento necessários às computações (escalonamento) e segurança. Capacidade de arma-

zenamento é necessária para que os pacotes aguardem sua vez no escalonamento realizado pelo sistema operacional tanto para sofrerem computações, como para serem transmitidos pelos enlaces de saída. A base de dados armazena informações de configuração e variáveis de ambiente que contêm dados sobre o nó.

Os principais requisitos de um sistema operacional para o nó ativo são flexibilidade no escalonamento e existência de mecanismos de segurança.

A flexibilidade no escalonamento é necessária para permitir o atendimento às exigências em relação a tempo (retardo) que os diversos tipos de fluxos podem exigir quando da especificação dos parâmetros das qualidades de serviço desejadas. Tal flexibilidade pode ser implementada pela existência de diversas políticas de escalonamento já presentes no sistema operacional utilizado ou através da instalação de extensões do kernel quando necessário.

No entanto, só a flexibilidade no escalonamento não é suficiente. A troca de contexto de processos e threads dos AA e AE difere dos modernos sistemas operacionais de propósito genérico. Pacotes podem conter código de programas que levam a trocas de contexto e a taxa de chegada de pacotes que requeiram essas trocas de contexto ou a criação de novos contextos seguramente é maior [18] do que a taxa de criação de novos processos e threads em um sistema operacional de uso genérico. É necessária então uma troca de contexto mais rápida do que aquelas que as soluções habituais oferecem.

O acesso aos recursos do nó pelo AE é permitido somente após a execução dos mecanismos necessários de autenticação e verificação de permissões do próprio AE e/ou do usuário para o qual está sendo solicitado o serviço do AE. Os mecanismos implementadores da segurança no acesso ao nó também podem se beneficiar da programação sob demanda.

Os canais de comunicação são utilizados pelos AE para o recebimento e envio de pacotes. São constituídos pelos enlaces de comunicação, como ATM e Ethernet e/ou pelos protocolos de mais alto nível, como, por exemplo, IP, TCP, UDP. Na seção seguinte, sobre o funcionamento do modelo, o canal será explicado mais detalhadamente.

O gerenciamento do próprio nó é também implementado via um AE, viabilizando o gerenciamento ativo.

As interfaces do modelo

As três principais interfaces do modelo (vide Figura 1) são:

- Entre as aplicações de usuário e as aplicações ativas.
- Entre as aplicações ativas e os ambientes de execução (linha tracejada superior da Figura 1).
- Entre os ambientes de execução e o sistema operacional do nó (linha tracejada inferior).

A última deve ser implementada na forma de chamadas ao sistema. A segunda é específica por ambiente de execução e diz respeito à programabilidade do AE disponível para a AA. A primeira pode ser materializada desde por simples campos dos pacotes que identificam os serviços desejados até a presença de códigos (as próprias AA) para implementarem o serviço ou os serviços solicitados pela aplicação do usuário.

Pode ser visualizado o cenário de utilização plena de redes ativas onde existirão os desenvolvedores tanto de AE como de AA. Caberá aos primeiros a especificação da interface a ser utilizada com seus AE, enquanto caberá aos últimos o desenvolvimento de serviços que serão utilizados pelas aplicações dos usuários.

O hardware do nó ativo

Os recentes avanços na tecnologia eletrônica vêm permitindo não só o aumento da capacidade de entrada e saída dos nós de comutação como também o aumento da complexidade do processamento que é realizado nos fluxos de pacotes. O atual emprego de sofisticadas técnicas de escalonamento, enfileiramento e filtragem em roteadores e comutadores são exemplos do aumento da capacidade de processamento. No entanto, a capacidade de memória dos atuais equipamentos ainda é limitada.

É então fácil de se admitir que o próximo passo na evolução do hardware seja a incorporação de componentes programáveis especializados, tipo FPGA, acompanhados de significativa quantidade de memória.

Além da capacidades de armazenamento em memória e de processamento elevadas, o hardware no nó ativo deve ser escalável, de forma a permitir o seu gradual aumento de capacidade de acordo com as demandas.

Para lidar com a capacidade de processamento esperada de um grande nó de comutação ativo é necessário haver distribuição no processamento, conforme pode ser observado na Figura 2. A matriz de comutação (Switch Element) é interligada a interfaces de enlace (LI – Link Interface) e a um processador de controle.

Cada interface de enlace é praticamente um sistema de computação independente. Ali estão presentes placas de interface de rede (LC – Line Cards), processadores de entrada (IPP), processadores de saída (OPP) e um elemento processador (PE). A implementação de um nó ativo de alta performance descrita em [4] utiliza na interface de enlace um processador Pentium com cache secundária, memória principal, controlador de interrupções e barramento PCI para comunicação entre os componentes internos e com a matriz de comutação.

O processador de controle, por sua vez, também é praticamente um sistema de computação completo, contendo até dispositivos de memória secundária (discos). Suas funcionalidades estão ligadas ao sistema em geral, como, por exemplo, implementar os protocolos de roteamento, interfaces de gerenciamento e de sinalização.

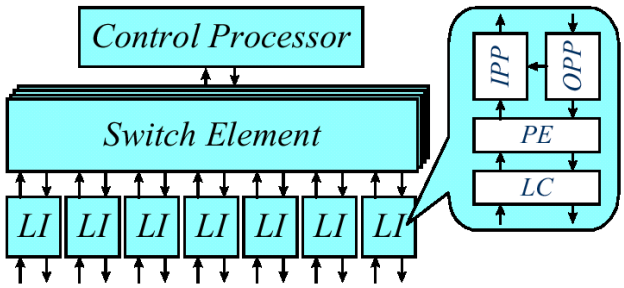

Figura 2 - Arquitetura de hardware de nó ativo.

2.3. Funcionamento

O funcionamento de um nó ativo é explicado com o auxílio da Figura 3.

A abstração mais importante do ponto de vista do encaminhamento de pacotes é o canal. Um canal é formado, no máximo, pela associação entre as seguintes computações: processamento de entrada, que engloba os processamentos da tecnologia de enlace, do nível de rede e de transporte, processamento de uma ou várias aplicações ativas e processamento de saída, normalmente restrito à tecnologia do nível de enlace em questão para a transmissão do pacote. Alguns desses processamentos podem não estar presentes, incluindo aqueles relativos às aplicações ativas, no caso dos canais que oferecem suporte às redes legadas.

Os canais são criados pelo sistema operacional mediante solicitações das AA aos AE ou dos próprios AE. Ao chegar no nó um pacote é classificado por uma thread ou processo específico

para essa finalidade, de acordo com as informações de controle contidas em seu cabeçalho. O pacote é então descartado ou atribuído a um canal. O relacionamento entre as informações do cabeçalho e um canal é determinado pelo AE ao classificador no momento da solicitação de criação do canal. Conforme já foi citado, visando manter compatibilidade reversa com tecnologias já existentes, canais podem não dispor de AE ativos. Exemplos das informações de cabeçalho do pacote utilizadas para a classificação do mesmo em um canal são portas UDP ou TCP, tipo de protocolo transportado no pacote IP, tipo de quadro Ethernet e outros, além de possíveis combinações desses campos. Essa classificação entrega o pacote ao ambiente de execução apropriado. Este, por sua vez, de forma específica por AE, entrega o pacote à AA apropriada, que executará a computação necessária para aquele pacote ou fluxo de pacotes.

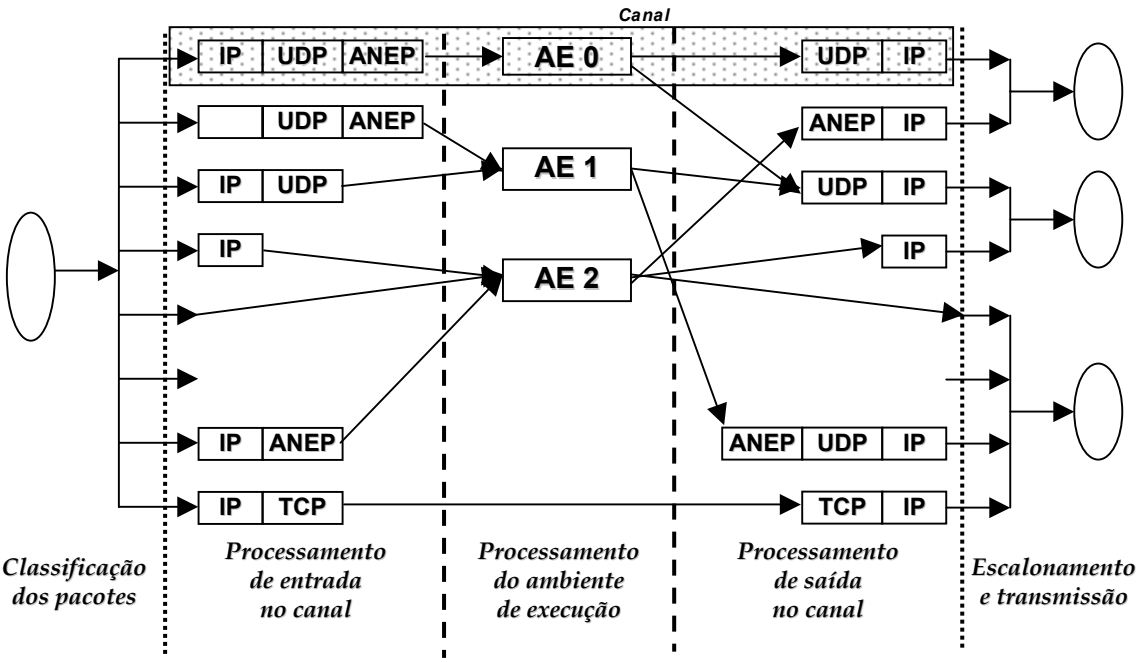

Figura 3 - Esquema de funcionamento de um nó ativo

Conforme será apresentado na seção sobre os projetos de pesquisa em andamento, esforços estão sendo feitos no sentido de se padronizar um protocolo para encapsulamento de pacotes para redes ativas, o ANEP (Active Network Encapsulation Protocol). Este protocolo é implementado em um cabeçalho que contém campos para identificar o AE a ser utilizado e outras informações referentes aos mecanismos de segurança.

Cabe ressaltar e repetir que todas as execuções relevantes, incluindo a criação de canais e de AE são submetidas à apreciação dos mecanismos de segurança do nó.

Uma vez classificados, o pacote ou os pacotes de um fluxo são submetidos ao processamento das aplicações ativas apropriadas, que implementam o serviço utilizado pelo pacote ou fluxo. Convém destacar novamente que as aplicações ativas podem estar sendo transportadas no próprio fluxo de pacotes ou podem já estar instaladas. Na realidade, três [16], [14] são as abordagens para se implementar o paradigma ativo em um nó no que diz respeito à instalação de aplicações ativas:

- Abordagem integrada: os programas que implementam as aplicações ativas são encapsulados nos próprios pacotes, juntamente com os dados. Nesse caso, a aplicação ativa é instalada imediatamente, na hora de sua utilização. Os pacotes ativos referentes a essa abordagem são chamados de cápsulas.

- Abordagem discreta: os programas não são integrados aos dados. Os pacotes ativos transportam apenas identificadores das aplicações ativas (ou rotinas das mesmas) que devem neles serem executadas. Nesse caso, as aplicações ativas são instaladas por um esquema de download de código sob demanda a partir de servidores. Essa abordagem, embora seja conhecida como "comutador programável", não deve ser confundida com a tecnologia de "redes programáveis", onde a programação do nó é realizada via operações de gerenciamento por um administrador.
- Abordagem mista: quando ambas as abordagens são utilizadas, isto é, os programas podem estar encapsulados nos pacotes mas também podem ser instalados via download, sob demanda.

É fundamental que o escalonamento do sistema operacional para as computações necessárias à classificação, ao ambiente de execução, à aplicação ativa, e ao processamento de transmissão viabilize certas garantias de qualidade de serviço. Exemplos dessas garantias podem ser banda para transmissão e limite no tempo de permanência do pacote no nó.

Proteção

Proteção diz respeito à isolar ou minimizar os efeitos de um comportamento errôneo de um programa sobre o nó de comutação.

Um pacote ativo pode sobrecarregar um recurso ou usar uma grande porção dos ciclos de CPU disponíveis. Assim, pacotes deixarão de ser encaminhados ou serão encaminhados sem que os requisitos temporais do fluxo do qual fazem parte não serão respeitados. Ainda, um programa pode entrar em loop infinito ou causar um erro de execução que impeça o nó de continuar a funcionar. Sendo assim, é essencial a existência de mecanismos de proteção contra a ocorrência de erros e mal comportamento na execução das aplicações ativas.

Essa proteção deve ser implementada na forma das seguintes restrições [7]:

- O estado instalado ou o comportamento atual de um nó deve ser possível de ser reinstalado ou recuperado caso haja interrupção no funcionamento.
- O armazenamento temporário dos pacotes ativos no nó deve ser persistente, de forma que mesmo que o nó seja reiniciado os pacotes não sejam perdidos.
- Deve haver limitação tanto do tempo de execução de uma thread ou um processo, bem como do número de threads e processos que podem ser criados por uma AA.
- O número de nós por onde pacotes podem passar e de vezes que podem se duplicar devem ser limitados.
- A quantidade de memória que um programa pode alocar deve ser limitada.
- Alocações de memória dinâmica (da heap) devem ser restringidas, uma vez que consomem tempo.
- Um programa não deve manipular diretamente a tabela de rotas do nó. A execução de rotinas do próprio nó devem ser solicitadas a realizarem tais manipulações através de chamadas (primitivas) ao sistema operacional.
- O uso de linguagem fracamente tipadas, como "C", por exemplo, deve ser limitado ou mesmo impedido. A ocorrência de erros do tipo utilização de um elemento de um vetor fora dos limites do mesmo, por exemplo, pode ter conseqüências graves no nó.
- A utilização de loops de programação deve ser cuidadosa e restrita, visando evitar a ocorrência de loops infinitos causados por erros de programação ou de execução.

- A sincronização entre processos e threads deve ser cuidadosamente utilizada ou até mesmo evitada, de forma a se prevenir a ocorrência de deadlocks.

A implementação dessas restrições pode ser dividida entre o uso de um sistema operacional e uma linguagem adequados.

Segurança

Segurança diz respeito a evitar que ações deliberadas impeçam o correto funcionamento do nó ativo. Diz respeito também a evitar que informações ou recursos sejam acessados sem a devida autorização.

Os motivos pelos quais uma rede é chamada de ativa são os próprios motivadores da necessidade de existência de fortes mecanismos de segurança. Um programa de um pacote pode destruir ou alterar o conteúdo de recursos e serviços de um nó através da reconfiguração, mudança ou mesmo remoção dos mesmos da memória. Um nó, por sua vez, pode apagar um pacote. Ainda, pacotes podem apagar outros pacotes. Um pacote pode também acessar informações privativas de um nó.

Ainda que sejam criptografados, os pacotes estão sujeitos a serem interceptados quando forem decriptografados para serem processados pelo nó. Normalmente a criptografia é utilizada apenas quando o pacote está em trânsito. No entanto, já existem técnicas, como a criptografia móvel, onde o pacote pode ser executado mesmo criptografado.

A geração de pacotes por unidade de tempo em um nó deve ser limitada de forma a se previnir, já em sua origem, ataques de negação de serviço, intencionais ou não.

Os trabalhos relativos à segurança em agentes móveis de software são aplicáveis no contexto das redes ativas.

As principais formas de se implementar a segurança do nó ativo podem ser resumidas nas operações de autenticação e monitoramento. A autenticação de pacotes ativos assegura que o pacote provém de uma fonte segura. Não garante, no entanto, que o pacote não contém conteúdo malicioso. Um mecanismo de monitoramento de referência deve ser utilizado para restringir as informações, recursos do sistema e serviços que pacotes ativos têm permissão para usar. Uma política de segurança é consultada para determinar se o acesso desejado é possível. Ainda assim, não é garantido que o conteúdo de um pacote autorizado seja seguro.

Em resumo, um pacote que contenha um programa, ao chegar em um nó deve sofre o seguinte processamento [7]:

1. Se for o caso, ter seu conteúdo decriptografado.
2. Ter a autenticidade de suas credenciais verificadas.
3. Ter sua origem identificada.
4. Com base nas suas credenciais, ter autorização para acessar os recursos necessários.
5. Ter a sua execução permitida com base nas autorizações e na política de segurança.
6. Ter monitorado e controlado o acesso aos recursos do nó.
7. Se for o caso, ter seu conteúdo criptografado.

3. Estado da arte

3.1. Principais áreas de pesquisa

De acordo com o modelo genérico que foi apresentado neste trabalho, podemos enquadrar as pesquisas sobre redes ativas nas seguintes categorias:

1. A plataforma, onde é investigada a arquitetura de hardware dos nós ativos.
2. O sistema operacional do nó, onde extensões ao kernel visando a implementação de políticas de escalonamento que permitam certas garantias de qualidade de serviço são estudadas.
3. Os mecanismos de segurança no acesso aos recursos do nó.
4. A linguagem de programação na qual os programas (que irão originar as aplicações ativas) a serem transportados pelos pacotes são escritos, no intuito de viabilizar a proteção ao nó, rapidez na execução e minimização do tamanho do código a ser transportado.
5. Os ambientes de execução.
6. As aplicações que podem ter seu desempenho melhorado pela utilização do paradigma redes ativas.

Cabe destacar que já existem redes de teste (testbeds) e simuladores onde são implementadas e/ou simuladas redes ativas experimentais.

3.2. Principais projetos em andamento

CANEs e Bowman, da universidade Georgia Tech

Abordagem utilizada: mista (integrada e discreta).

Categorias de áreas de pesquisa abordadas:

- Ambiente de execução: CANEs.
- Sistema operacional de nó: Bowman.

Outra área de pesquisa do projeto :

- Multicasting confiável.

O ambiente de execução CANEs [12] utiliza o paradigma de programação orientada a componentes. Uma parte do AE é fixa e é executada em todos os pacotes. Essa parte fixa possui espaços vagos (chamados de slots) que podem ser ocupados por programas com funcionalidades específicas para o fluxo de pacotes em questão. A composição de serviços no CANEs é então dividida em duas etapas. Na primeira etapa a parte fixa do AE selecionada e na segunda etapa um ou um conjunto de programas é selecionado para completar a parte fixa. Esses programas podem estar disponíveis no nó ou serem buscados em um nó remoto. Não é obrigatório o carregamento de um programa em um slot.

O sistema operacional de nó ativo Bowman [12] é formado pela sobreposição de uma camada com funcionalidades específicas para redes ativas no sistema operacional do hardware do nó. O sistema operacional nativo é quem disponibiliza os mecanismos de acesso aos recursos de mais baixo nível. Bowman provê três abstrações básicas para o suporte às funcionalidades requeridas para redes ativas: o canal, o fluxo de pacotes e o armazenamento de estado. Adicionalmente, um mecanismo de extensão similar aos módulos carregáveis de um sistema operacional tradicional.

Canais diferentes implementam diferentes protocolos. Os canais podem ser criados, destruídos e ter seu status verificado pelos usuário. Computações envolvendo compressão ou FEC podem ser realizadas nos canais. O fluxo é a unidade de computação do sistema operacional Bowman. Cada fluxo possui pelo menos uma thread associada. A funcionalidade de armazenamento de estado está disponível para os fluxos possam armazenar estados de forma recuperável.

ANTS do MIT

Abordagem utilizada: discreta.

Categorias de áreas de pesquisa abordadas:

- Linguagem de programação para as aplicações ativas: Java.
- Ambiente de execução: máquina virtual Java modificada.
- Aplicações exploradas: multicasting e acesso a estações móveis em roaming.

Componentes adicionais pesquisados:

- Sistema de distribuição de código.

Uma rede ativa ANTS [17] dispõe de nós que possuem o ambiente de execução ANTS. O serviço oferecido pela rede não é fixo, uma vez que qualquer protocolo pode ser instalado nos nós. Os pacotes ativos (chamados de cápsulas) transportam o identificador de um protocolo e as rotinas de encaminhamento desse protocolo.

Um sistema de distribuição de código é utilizado para enviar o código referente a um novo protocolo ao longo da rota de um fluxo de pacotes. A aplicação de origem registra o identificador de um novo protocolo e envia cápsulas ao próximo nó da rota. Nesse e nos demais nó, o código de encaminhamento referente ao identificador da cápsula é procurado. Se não for encontrado, o processamento daquela cápsula é suspenso e o nó anterior é solicitado a enviar o código desejado. Posteriormente, esse código é mantido em cache para que, se porventura voltar a ser necessário, já esteja disponível.

O sistema operacional utilizado no nó é o Linux.

O ambiente de execução limita o acesso aos recursos compartilhados. Todos os acessos a recursos compartilhados, como por exemplo, tabelas de rotas, deve ser autenticado.

Implementa mecanismos de proteção.

Por serem escritos em Java, os programas não podem acessar os recursos de baixo nível do nó. A performance, também pelo mesmo motivo, é prejudicada. No entanto, como o principal objetivo do projeto é a experimentação prática do paradigma das redes ativas, estas desvantagens não são significativas.

Active IP, do MIT

Abordagem utilizada: integrada.

Categorias de áreas de pesquisa abordadas:

- Linguagem de programação para as aplicações ativas: várias podem ser utilizadas. A linguagem TCL foi implementada.
- Ambiente de execução: interpretador TCL simplificado.
- Aplicação explorada: roteamento.

Nesse projeto, o campo de extensões IP, previsto na versão 4 protocolo IP, é utilizado para indicar a presença de código embutido no pacote. Nesse caso, o pacote (ativo) é chamado de cápsula. Estes pacotes ativos podem então instalar aplicações ativas nos nós da rede. Várias linguagens podem ser utilizadas. Uma das opções para o uso de uma cápsula é justamente interrogar um nó ativo para que este responda quais linguagens que são por ele entendidas.

Cápsulas podem instalar códigos que podem vir a serem utilizados por outras cápsulas, cujos códigos dependam daqueles primeiros já instalados.

Uma vez que o esquema é baseado em uma extensão do campo IP options do protocolo IPv4, as capacidades da tecnologias são limitadas. A primeira implementação utilizou a linguagem TCL. Um interpretador simplificado dessa linguagem deve estar presente no nó ativo como ambiente de execução.

SwitchWare, da Universidade da Pensylvannia

Abordagem utilizada: mista.

Categorias de áreas de pesquisa abordadas:

- Linguagem de Programação: PLAN e Caml.
- Mecanismo de Segurança: SANE.

Aplicação explorada: recuperação automática de falha de um algoritmo Spanning Tree.

A arquitetura SwitchWare [2] foi concebida para dotar a rede com facilidades de maneira que nós ativos (pontes, comutadores e roteadores) permitam a carga e execução de programas remotamente instalados (em pacotes ativos). Por outro lado, essa arquitetura promove um equilíbrio entre a flexibilidade introduzida e exigências quanto à segurança e proteção que são inerentes a qualquer sistema compartilhado. Essa flexibilidade permite que novos protocolos sejam facilmente incorporados a infraestrutura da rede sem a necessidade de modificar os programas de todos os nós. Verificação de integridade, autenticação baseada em criptografia, verificação estática de tipo além de nós (roteadores) com infraestrutura segura são os recursos de segurança e proteção utilizados compensando a vulnerabilidade intrínseca de redes programáveis (ativas).

PLAN (Programming Language for Active Networks) é a linguagem de programação desenvolvida que gera os programas que são encapsulados nos pacotes ativos. Essa linguagem com funcionalidade mínima possui, no seu modelo de execução, mecanismo para avaliação remota de programas PLAN. Os programas PLAN, visando a segurança, são fortemente tipados e podem ser estaticamente verificados antes de serem enviados na rede, visando a eliminação de erros de tipo em nós ativos remotos. Tais programas não podem alterar o estado do nó, além de possuir severas restrições que limitam o acesso a recursos do nó. Os programas PLAN, uma vez instalados nos nós, podem acessar rotinas de serviços de autenticação e outros mecanismos de segurança.

Outra linguagem, chamada Caml, sem as restrições da linguagem PLAN ao acesso a recursos dos nós, também é utilizada para gerar códigos móveis. Devido sua grande generalidade e objetivando prover segurança e imunização a ataques, metodologias formais de prova de correção dos programas e mecanismos de autenticação foram incorporadas.

Extensões ativas são aplicações ativas residentes em roteadores que tem como função dota-los com capacidade computacional suficiente, uma vez que na arquitetura switchware protocolos complexos são implementados através dos programas PLAN dos pacotes ativos em conjunto com essas extensões ativas. A comunicação entre essas extensões ativa, que não são móveis, é estabelecida através de pacotes ativos. Como extensões ativas são acessadas somente quando necessárias, suas implementações sem a preocupação de otimização de código podem ser escritas em qualquer linguagem de uso geral desde que cuidados quanto à segurança sejam observados.

Um protótipo de uma ponte ativa, que interliga redes locais Ethernet de 100 Mbps, foi construído para o estudo de certas funcionalidades introduzidas. A mais importante é a capacidade de recuperação automática de falha de implementação de um algoritmo spanning-tree. Extensões ativas chamadas switchlets foram incorporadas. Todos os programas, escritos em Caml, rodam em uma plataforma Intel e sob o controle dos sistemas operacionais Linux e OpenBSD. Duas extensões ativas switchlets estão presentes no sistema. A primeira disponibiliza recursos necessários simplificando a conexão com as redes locais. Uma vez que o desempenho é altamente dependente da velocidade dos dados nos canais, uma outra switchlet, com funcionalidade de autoaprendizado, foi incorporada ao sistema. Isto permite o descarte de quadros duplicados preservando a banda passante.

SANE (Secure active network environment), da Universidade da Pennsylvania

Categorias de áreas de pesquisa abordadas:

- Segurança no acesso aos recursos do nó.

SANE é uma arquitetura [2] para segurança em redes ativas que é organizada em dois níveis, estático e dinâmico. O nível estático lida com as verificações que são realizadas com baixa freqüência, como boostrap do nó ativo e inicialização do sistema operacional. O nível dinâmico lida com as verificações constantes, ao longo do tempo de funcionamento do nó ativo, como autenticação e verificação de permissões para o tratamento de pacotes e de usuários.

A arquitetura foi implementada e o seu desempenho foi medido. Testes indicaram que a performance na transferência de fluxo de dados com o uso do SANE sem autenticação sofreu uma degradação de 28% enquanto que com autenticação essa degradação cresce para 62%. Um teste mais simples, como o envio de um ping, mostrou que a relação de tempo entre o processamento com autenticação e sem autenticação é de cerca de 1,6.

DAN (Distributed code caching Active Network), da Universidade de Washington

Abordagem utilizada: discreta.

Categorias de áreas de pesquisa abordadas:

- Linguagem de programação para as aplicações ativas: “C”, mas já compilada para a plataforma que é o nó ativo.

Os pacotes da arquitetura DAN [10] contêm uma seqüência finita de identificadores de funções e de parâmetros para essas funções. A seqüência de funções dita a ordem em que as mesmas serão executadas. O código destas funções já está carregado no nó ativo ou então é buscado sob demanda. A primeira função a ser executada é determinada automaticamente pelo canal através do qual o pacote foi recebido. A seqüência de funções executadas em um pacote constitui a aplicação ativa.

Se o nó não possui já instalada uma função invocada por um pacote, o processamento nesse pacote é temporariamente suspenso e um servidor de código é solicitado a enviar o código necessário. Esses servidores possuem endereços bem conhecidos. Os códigos das funções estão disponíveis já compilados a partir da linguagem “C” para uma variedade de sistemas operacionais e arquiteturas de hardware. Uma vez carregado o código que estava faltando, este permanecerá armazenado em definitivo de forma a evitar novas transferências a partir do servidor.

Esta abordagem é flexível porque novos códigos que implementam novas funcionalidades précisam ser carregados apenas nos servidores. Quando necessário, os demais nós da rede os buscarão. Em compensação, o retardo para o início da execução de uma aplicação ativa pode ser grande. Uma possível solução é realizar a busca de todo o código necessário para o processamento antes do envio do fluxo de dados propriamente dito. Seria a implementação de uma fase de sinalização. Exceto pelo carregamento do código faltante, o desempenho das aplicações ativas é muito bom, pois as mesmas já estão presentes em linguagem de máquina da plataforma que constitui o nó ativo.

A segurança é provida através da utilização de servidores de códigos bem conhecidos que só realizam as transferências de programas mediante autenticação. Adicionalmente, os próprios códigos possuem assinaturas digitais dos seus desenvolvedores que podem ser verificadas através de esquemas de chave pública. Dessa forma é possível também aos administradores permitir o carregamento de código nos servidores diretamente a partir dos desenvolvedores, de forma automática, mas segura.

ANN (Active Network Node), da universidade de Washington (St. Louis)

Categorias de áreas de pesquisa abordadas:

- Sistema operacional de nó ativo: baseado em UNIX com kernel extensível.
- Hardware: ANN.

ANN [7], [9] é um sistema de computação a ser utilizado como plataforma de pesquisa para nós ativos. O projeto inclui extensões ao sistema operacional.

ANN foi projetado com objetivo de ser uma plataforma de pesquisa aberta, de alto desempenho. Diversas opções de configuração de software e hardware estão disponíveis. O sistema é construído como uma matriz de comutação escalável cujas portas de entrada e saída possuem processadores de uso genérico.

Os processadores do ANN utilizam um sistema operacional de uso genérico, o UNIX Net BSD, mas extensões ao kernel podem ser instaladas de forma que computações específicas podem ser realizadas nos fluxos de pacotes.

O ANN é utilizado como plataforma para as aplicações ativas DAN.

Smart Packets, da BBN

Abordagem utilizada: integrada.

Categorias de áreas de pesquisa abordadas:

- Linguagem de programação para as aplicações ativas: Sprocket e Spanner.
- Ambiente de execução: máquina virtual para Sprocket e Spanner.
- Aplicações exporadas: gerenciamento.

O projeto especifica um pacote ativo, chamado Smart Packet, que encapsula um programa que pode ser enviado aos nós da rede. O programa, por sua vez, é encapsulado em um pacote do ANEP e este em um pacote IP tradicional. Um ambiente de execução baseado no ANEP é executado em cada nó ativo.

A linguagem de programação utilizada pode ser Sprocket, que é uma linguagem de alto nível, semelhante a "C", ou sua versão compilada, chamada Spanner. Esta última é formada por códigos binários independentes de uma arquitetura específica de hardware. A razão de duas novas linguagens terem sido especificadas é, segundo os autores, porque as linguagens existentes não são capazes de permitir a geração de um programa com razoáveis capacidades, independente de máquina, em menos de 1 KB. Esta limitação é um dos objetivos do projeto: encapsular um programa inteiro em menos de 1 KB e em um único pacote, que não pode ser fragmentado.

A aplicação alvo do projeto é o gerenciamento.

Liquid software, da universidade do Arizona

Abordagem utilizada: integrada.

Categorias de áreas de pesquisa abordadas:

- Linguagem de Programação: Java modificada.
- Aplicações exploradas: procura através de código móvel.

Liquid Software [11] é a designação dada a infraestrutura necessária para agregar funcionalidades (móveis) ao longo de toda a rede através de códigos móveis (pacotes ativos) considerando questões como eficiência, segurança e alocação de recursos, entre outras.

Três são as abordagens do Liquid Software: compilação rápida do código móvel, aplicação de pesquisa utilizando pacotes ativos e suporte oferecido pelo sistema operacional.

Eficiência na execução de códigos móveis impõe que sejam efetuadas compilações nesses códigos em vez da interpretação byte a byte usualmente praticada. Entretanto, os tempos dessas compilações passam a ter papel preponderante no desempenho do sistema. A tecnologia desenvolvida para geração dinâmica de código possibilitará o emprego de técnicas de compilação rápidas. A meta a ser atingida é a de realizar a compilação em uma maquina com clock de 200 Mhz em tempo hábil de receber um pacote ativo em um canal de 5 a 10 Mbps.

Aplicação de busca dinâmica e configurável de informações combinando técnicas de browsing e searching está sendo desenvolvida. O objetivo é de estabelecer um padrão de busca de informação em rede além de demonstrar a aplicabilidade de códigos móveis (pacotes ativos) em situações reais no contexto do Liquid Software.

O sistema operacional Scout é o responsável pelo gerenciamento dos recursos e implementação da política de segurança a ser disponibilizados nos nós ativos.

Joust e Scout, da Universidade do Arizona

Abordagem utilizada: integrada.

Categorias de áreas de pesquisa abordadas:

- Ambiente de execução: Joust
- Sistema operacional de nó ativo: Scout.

Joust [11] é um ambiente de execução baseado em uma máquina virtual Java modificada

Scout é um sistema operacional orientado a objetos desenvolvido especificamente para ser utilizado em um hardware utilizado para comunicações e não de emprego genérico. O kernel do Scout é uma composição configurável de primitivas de comunicação de baixo nível implementadas como módulos que são combinados para formar canais. Esses módulos são as unidades de configuração do Scout, cada um tendo uma funcionalidade específica e independente dos demais módulos. Exemplos de módulos pode ser implementações de protocolos como UDP, TCP, IP, tipos de escalonadores e drivers de dispositivos para armazenamento. Para formar um sistema completo, módulos individuais são conectados em um grafo de configuração. Para ser possível a conexão entre dois módulos deve estar definida a interface entre ambos. Quando utilizado em nós ativos, Scout é configurado para otimizar o processamento de entrada e saída da rede.

A máquina virtual Joust é, na realidade, um módulo do sistema operacional Scout. Assim, Joust é otimizado em relação a uma máquina virtual Java tradicional no que diz respeito a utilização de recursos do sistema, como canais de entrada e saída, escalonamento e sincronização entre t-hreads. Pode-se dizer então que o ambiente de execução Joust é, neste projeto, interno ao sistema operacional do nó. Esta característica já demonstrou em medições comparativas realizadas entre Joust e máquinas virtuais Java tradicionais que o seu desempenho é significativamente superior [11].

Joust e Scout são desenvolvidos como plataforma para o projeto Liquid Software, abordado anteriormente, mas já foi utilizado também como plataforma para o ambiente de execução ANTS.

NetScript da Universidade de Columbia

Abordagem utilizada: mista.

Categorias de áreas de pesquisa abordadas:

- Linguagem de programação para as aplicações ativas: NetScript.
- Ambiente de execução: máquina virtual para NetScript.

O projeto provê uma arquitetura que permite a programação dinâmica do nó ativo e uma linguagem, NetScript, para a criação de aplicações ativas. Pacotes ativos contendo código escrito em

NetScript são utilizados para o controle e programação dos nós para que esses processem fluxos de pacotes de dados. Estes, por sua vez, contêm apenas um cabeçalho NetScript mínimo para permitir a identificação do fluxo a qual pertencem. Em cada nó ativo existe um ambiente de execução, chamado de Virtual Network Engine (VNE), para interagir com as aplicações ativas, que são executadas em thread.

Cada nó ativo contém, além do ambiente de execução (VNE), um módulo de conectividade e outro de gerenciamento de recursos. O primeiro é responsável pela interação com a plataforma do nó para a alocação e desalocação de canais, escalonamento (para transmissão) e transmissão de pacotes.

A linguagem NetScript é orientada a objetos e foi projetada especificamente para tarefas relacionadas à comunicação de dados, e pode operar sobre fluxos de pacotes.

ARP (Active Reservation Protocol) – University of Southern California

Abordagem utilizada: discreta.

Categorias de áreas de pesquisa abordadas:

- Ambiente de execução: máquina virtual Java.
- Aplicações ativas: para RSVP e RIP, escritas em Java.
- Classe de aplicações exporadas: sinalização.

O projeto ARP [4] explora a instalação dinâmica de protocolos de sinalização, quer seja para atualização ou para a disponibilização de novos serviços. Trata-se de um ambiente de execução baseado em uma máquina virtual Java. Os pacotes ativos transportam código em bytecodes Java. O ambiente de execução é chamado de ASP – (Active Signalling Protocol).

Duas aplicações ativas já foram desenvolvidas: Jrsvp e Jrip. A primeira implementa o protocolo RSVP enquanto a segunda implementa o protocolo RIP.

Alguns nós da rede de testes ABONE já possuem o ASP e as aplicações ativas citadas instaladas.

Messenger (M0), Universidades da Califórnia e de Genebra

Abordagem utilizada: integrada.

Categorias de áreas de pesquisa abordadas:

- Linguagem de programação para as aplicações ativas: M0.
- Ambiente de execução: plataforma (proprietária) para interpretação de M0.

O projeto M0 [13] foi concebido com ênfase no paradigma de agentes móveis para a implementação de sistemas operacionais distribuídos e computação distribuída em geral. Trata-se de uma linguagem de programação similar à linguagem Postscript, de alto nível (sem ponteiros), que é interpretada em uma plataforma M0 – o ambiente de execução. Esta plataforma é escrita em linguagem "C", não orientada a objetos. Os pacotes que contém códigos são chamados de mensageiros (messengers) e transportam um programa completo. Uma thread é criada para a execução de cada código, constituindo a aplicação ativa. É também definido pelo projeto o formato do mensageiro e o identificador (nome) do ambiente de execução a executar o código. Os mensageiros são entregues através do uso de protocolos de melhor esforço.

A plataforma M0 disponibiliza abstrações que provêm o acesso ao sistema operacional e ao hardware do sistema de computação utilizado no nó ativo. Adicionalmente, é prevista a execução de processos nativos da plataforma em questão através do uso do comando "_run" da linguagem M0.

Não há interação entre aplicações ativas diferentes, ou seja, não há um protocolo para comunicação entre AA, o que simplifica o código a ser encapsulado em um mensageiro. Apenas computações locais podem ser realizadas.

Atualmente, não há o suporte à mecanismos de segurança e proteção na linguagem M0.

ABONE, rede de teste do DARPA

ABONE é uma rede de testes virtual do programa de pesquisa em redes ativas do DARPA. A rede é composta por um conjunto de sistemas de computação onde podem ser instaladas redes ativas virtuais. Os sistemas operacionais utilizados são versões do UNIX, como Linux, FreeBSD, Solaris ou sistemas operacionais específicos para nós ativos. Aplicações ativas e ambientes de execução podem também ser testados. É escalável até milhares de nós.

A topologia da rede é dividida em duas partes. Um núcleo é composto por nós ativos públicos que estão sempre disponíveis para testes. Redes ativas privativas podem se integrar diretamente ou através de tunelamento a um nó da rede do núcleo. Nessas redes privativas os desenvolvedores podem iniciar os testes de seus ambientes de execução, sistema operacional de nó ou aplicações ativas, que podem ou não estar disponíveis para acesso externo às suas redes. Mecanismos de segurança estão implementados.

Os nós do ABONE executam um ambiente de execução de gerenciamento chamado ANETD. Este software é experimental e foi projetado para permitir a instalação, operação e o controle da rede. Através do seu uso, outros ambientes de execução podem ser instalados. O ANETD também é o responsável por realizar a classificação dos pacotes que chegam ao nó.

A rede do núcleo conta atualmente com 36 nós. Os ambientes de execução ANTS, PLAN e ASP estão instalados nos nós ativos do ABONE.

Protocolos de encapsulamento

A BBN está propondo uma RFC [15] para o ANMP (Active Network Message Protocol). Este seria um protocolo a ser utilizado por sobre o IP, recebendo um número para ser utilizado no campo IPv4 que especifica o protocolo que está sendo transportado. O ANMP oferece o suporte à utilização de múltiplos ambientes de execução, permitindo a demultiplexação dos pacotes recebidos e encaminhamento dos mesmos para os seus ambientes de destino. As mensagens ou programas não são objeto de padronização por este protocolo. Apenas a identificação de ambientes de execução o é. A aplicação ativa do ambiente de execução que deve tratar o pacote é especificada no payload do pacote do ANMP, estando portanto, fora da especificação do protocolo.

Enquanto o ANMP especifica apenas o ambiente de execução a ser utilizado, uma outra proposta de RFC, o ANEP, possui um cabeçalho que contém informações, como, por exemplo, para autenticação, criptografia e verificação da integridade do pacote. Há também a previsão para campos opcionais no formato TLV (Type, Lenght, Value). Além disso, a utilização do pacote do ANEP não é restrita ao IP, podendo ser utilizado em qualquer protocolo de nível de rede ou de enlace.

3.3. Principais aplicações

Os projetos de pesquisa específicos sobre aplicações que podem obter proveito no uso da tecnlogia de redes ativas estão fora do escopo deste trabalho. Serão apresentados a seguir as principais áreas de aplicação que podem ser beneficiadas e a descrição das vantagens mais siginifcativas que as mesmas podem obter das redes ativas.

Multicasting

O emprego de roteadores inteligentes (ativos) com capacidade computacional (nível de aplicação) em nós estratégicos da rede permite que a técnica de multicasting seja implementada de

maneira confiável e eficiente, além de aumentar a escalabilidade de serviços da rede. A idéia é não permitir que ocorram retransmissões de pacotes (perdidos) desde a origem até o destino. Esta tarefa passa a ser regional, atribuindo a responsabilidade de multicasting a nós ativos em posições mais afastadas da origem. Neste caso, consegue-se limitar as retransmissões até o ponto da falha ou perda. Os nós ativos receptores de multicasting devem ter conhecimento de nós vizinhos anteriores para que as retransmissões sejam realizadas. Memória cache incorporada a estes nós ativos possibilitam o armazenamento rápido do seu estado no momento da falha ou perda de pacotes. Logo, as retransmissões são minimizadas, pois estas ocorrem cada vez mais próximas do(s) (destino(s)). A quantidade de cache e tempo de armazenamento depende de fatores como rota dos pacotes, volume de tráfego e qualidade dos canais de comunicação. A quantidade de cache, também, é uma questão de compromisso com a banda passante e desempenho.

Qualidade de serviços

A qualidade de diversos serviços fica bastante degradada com a ocorrência de congestionamento devido, tanto ao aumento do fluxo de dados (tráfego) quanto de enlaces com elevados índices de erros. Além disso, este problema é agravado pelo dimensionamento inadequado de nós (passivos) e a incapacidade dos mesmos de se adequarem dinamicamente ao tráfego. Os procedimentos tradicionais que tratam de congestionamento mostram-se ineficientes, uma vez que um longo tempo é gasto entre a ocorrência do problema (nó distante da estação de gerenciamento) e a efetiva ação saneadora, que pode acontecer quando o congestionamento não mais estiver presente. Isto pode acarretar em perdas significativas para o receptor, tendo em vista a crescente demanda de serviços de multimídia.

O congestionamento, por ser um problema eminentemente interno à rede, estimula que seu controle e tratamento sejam implementados no interior da rede através de nós ativos, trazendo uma agilização e otimização impossíveis de serem alcançados com as redes passivas tradicionais. As redes ativas podem executar várias funções no tratamento do congestionamento tais como:

- Monitoramento e controle da taxa de utilização da banda passante juntamente com fluxo de dados (trafego).
- Transformações podem ser efetuadas nos dados de acordo com o grau de congestionamento no momento.
- Aplicação de políticas de descarte seletivo de informações menos importantes.
- Compartilhamento de informações (estado dos nós, memória cache disponível, mapeamento de objetos armazenados, etc) entre nós de uma determinada região permitindo que ações localmente (nó) executadas não tenham conseqüências negativas a nível regional.

Caching

A crescente demanda de serviços que necessitam obter objetos de servidores vem provocando um aumento substancial no tráfego e do tempo de resposta causando, conseqüentemente, a degradação da qualidade de serviços em toda a rede. O esquema de configuração de cache, nas redes convencionais, é realizado manualmente numa hierarquia estática que exige um grande esforço administrativo. Isto torna as redes atuais inábeis a reagir a condições dinâmicas, apresentando então grande latência. Geralmente, grandes quantidades de cache são posicionadas em nós (passivos) específicos tais como: nós de transito e nós na borda da rede. Apesar deste esquema apresentar certa melhora no acesso a servidores, a flexibilização de alocação de cache dos nós de uma região poderia agregar maior funcionalidade à rede.

Com as redes ativas é possível direcionar pedidos de requisição de cache a outros nós que possuam esquemas de cache pré-configurado e informações sobre configurações de nós vizinhos. Isto permite um balanceamento de carga em toda a hierarquia de cache, ou seja, reposiciona in-

formações armazenadas e descarta outras não importantes. Com o objetivo de ilustrar a redução de tráfego alcançado com nós ativos, suponha que determinado objeto (por exemplo, uma página web) tenha uma ocorrência acentuada em um nó próximo ao cliente. Caso este nó ativo tenha capacidade de perceber tal ocorrência, o objeto pode ser armazenado em cache para futuras requisições do próprio nó e também de nós vizinhos evitando fluxo de pacotes até o servidor.

Nós ativos devem estar providos de capacidade computacional suficiente para coordenar o mecanismo geral de requisição e alocação de cache, bem como do mapeamento de objetos já armazenados em cache de nós vizinhos. Além disso, tendo em vista que um grande número de páginas da web é processado dinamicamente, um nó ativo próximo ao cliente deve oferecer suporte ao armazenamento e execução dos programas que geram estas páginas.

Multicasting confiável requer que a duração e localização de objetos em cache sejam determinadas dinamicamente tornando cache ativo uma ferramenta importante na sua implementação.

Serviços de participantes de grupos são implementados de forma otimizada na medida que inclusões e exclusões de membros acarretam, geralmente, em reconfiguracão da árvore de multicasting e possivelmente da distribuição de cache dos nós. Do ponto de vista de gerenciamento, este esquema torna a rede praticamente imune à mudança de participantes.

Gerenciamento

A aquisição de informações para gerenciamento de redes é, na maioria dos casos, realizada através de polling entre a estação de gerenciamento (EGR - mestre) e os equipamentos (escravos) da rede. A centralização das informações, freqüentemente redundantes, juntamente com o aumento do número de nós e a complexidade das redes tornaram a tarefa de gerenciamento problemática. A concentração de pacotes em um único ponto (EGR) e a duplicidade de pacotes acarretam em desperdício considerável de banda passante. Além disso, o tempo de resposta entre a ocorrência da anomalia em algum nó da rede e a ação efetiva para o restabelecimento do seu estado normal é bastante elevado, podendo o problema já não mais existir no momento da ação reparadora.

As redes ativas permitem que o gerenciamento seja executado de maneira distribuída ao longo da rede. Esta descentralização tem como conseqüência uma significativa redução do tempo de restabelecimento do nó e em uma otimização da utilização da banda passante. Programas devidamente encapsulados em pacotes podem der inseridos na rede e agir imediatamente em nós problemáticos. Outros pacotes de programas podem investigar a rede a procura de anomalias e reportá-las à estação de gerenciamento ou a um outro nó ativo vizinho com capacidade de aplicar medidas que resolvam o problema. Há, ainda, pacotes que se instalam em nós para aquisição de informações (estado) que depois de processadas são enviadas ou não ao EGR. Esta medida diminui a geração de tráfego, pois somente as informações importantes são enviadas ao EGR.

Devido à alta flexibilidade e uma maior escalabilidade das redes ativas, mudanças de políticas de gerenciamento também são facilmente implementadas, assim como a inclusão de novos serviços.

3.4. Questões em aberto

Uma vez que a pesquisa em redes ativas é recente, diversas questões ainda estão sem serem abordadas ou as abordagem já realizadas não são conclusivas ou, pelo menos, ainda não foram medidas e questionadas.

A interface entre ambiente de execução e sistema operacional do nó deve ser padronizada ? A padronização viabilizaria uma interoperabilidade sem precedentes na área de redes, pois qualquer desenvolvedor poderia fabricar ambientes de execução para qualquer tipo de nó ativo. A diferenciação entre nós ativos recairia apenas no desempenho do hardware. No entanto, algumas pesquisas já englobam o ambiente de execução no interior do sistema operacional do nó, visan-

do justamente o aumento de desempenho na execução das aplicações ativas. No mínimo, uma cópia de dados entre as áreas de usuário e kernel é poupada. Há inclusive medições publicadas que comprovam esse aumento, conforme foi apresentado no projeto Joust – Scout. Essa abordagem contraria a padronização, pois exige um desenvolvimento específico para o AE.

A interface entre o pacote ativo e o ambiente de execução deve ser padronizada ? Esforços têm sido envidados no sentido de se desenvolver um protocolo que permita a especificação dos requisitos do pacote para o nó. ANEP e ANMP são exemplos desse esforço. No entanto, algumas soluções utilizam os campos de opções do pacote IP, sem necessitar, portanto, de mais um nível de protocolo. Nesse caso, ao contrário do que foi apresentado na padronização entre AE e sistema operacional de nó,

Qual o nível de proteção que deve existir nos nós ativos ? Qual a conseqüência da proteção para o desempenho ? Um mínimo de proteção é necessário, visando garantir que pacotes ativos não causem danos – ainda que não intencionais – ao nó. É preciso haver garantias de que os programas das aplicações ativas terminem e que os recursos do nó sejam acessados com parcimônia. Algumas pesquisas abordam exclusivamente a linguagem de programação que deve ser utilizada nos pacotes ativos, tentando torná-la o mais restritiva possível sem no entanto limitar sua funcionalidade.

Como deverão ser a sintaxe e semântica, bem como a granularidade, das especificações aos recursos do nó ? Taxas média e de pico em bits por segundo, variação máxima do retardo e taxa de erros aceitável são alguns dos requisitos de qualidade de serviço fim a fim que aplicações podem especificar e/ou requerer da rede. Visando atender a esses requisitos, é necessário ter-se algum controle ou pelo menos uma forma de se solicitar certos comportamentos do escalonador do sistema operacional. A existência de uma padronização na forma de expressar esses requisitos sem dúvida favorecerá o desenvolvimento de ambientes de execução e de aplicações ativas.

Onde deverão estar localizados servidores de código na rede ? Eles podem migrar ? No caso da abordagem discreta citada na seção 2.3, a localização dos servidores de código no interior da rede deve ser cuidadosamente escolhida visando minimizar o tráfego de download bem como o tempo de instalação de rotinas ou aplicações ativas. Ao longo da utilização e evolução da rede, é possível que a localização desses servidores tenha que mudar. O paradigma dos agentes móveis pode ser utilizado. Quando e para onde mudar ainda é questão em aberto.

Quais serão e quem será a autoridade para designação de valores de campos de identificação dos protocolos de encapsulamento para redes ativas ? Embora algumas pesquisas já até assumam que tal autoridade é a ANANA (Active Networks Assigned Numbers Authority), os campos propriamente ditos e seus valores, precisam ser definidos.

4. Conclusões

O surgimento de tecnologias que oferecem suporte ao encapsulamento, execução segura e eficiente, interposição de programas e fragmentos de programas torna possível a rede ativa. Ainda, funcionalidades como eficiência, mobilidade e proteção vem sendo endereçados nas áreas de sistemas operacionais e linguagens de programação.

Ainda assim, apesar do aparente clamor por redes ativas, a comunidade de pesquisa está cautelosa principalmente no que diz respeito à complexidade da tecnologia e o inerente risco de queda de desempenho.

Adicionalmente, a comunidade de telecomunicações investiga com profundidade o paradigma das redes programáveis, que podem ser consideradas um passo intermediário entre as redes tradicionais e as ativas. Segundo os seus defensores, as redes programáveis estão menos sujeitas a problemas de segurança e proteção, bem como a ter seu desempenho prejudicado por causa da menor complexidade inerente ao seu paradigma.

A comunidade comercial limita-se a acompanhar de longe o progresso dessa pesquisa ou, no máximo, financiar alguns projetos. O motivo é que as redes ativas exigem uma profunda mudança na estrutura dos roteadores em relação a que está em franca comercialização nos dias de hoje.

É inegável, no entanto, que as vantagens são significativas no uso das redes ativas:

- Um menor tempo é gasto na instalação de novos serviços, pois não é necessário esperar o término de longos processos de padronização ou realizar trocas significativas de equipamentos.
- Os próprios usuários ou desenvolvedores específicos podem criar serviços mais voltados às necessidades das aplicações.
- Pesquisadores podem experimentar em escala real sem a necessidade de interrupções na prestação de serviços sobre a rede.

A rede de teste ABONE é fundamental para o desenvolvimento da tecnologia. Diversos projetos já a utilizam e medições em escalas reais e globais já podem e estão sendo realizadas. Pode-se dizer que essa rede será o núcleo de uma futura Internet via rede ativa, caso a tecnologia venha mesmo a obter sucesso.

5. Referências

[1] Alexander, S. *et al*, "Active network encapsulation protocol – RFC draft", Julho de 1997.

[2] Alexander, S. et al, "The SwitchWare active network architecture", IEEE Network Magazine, junho de 1998.

[3] Alexander, D., "A secure active network environment architecture: realization in SwitchWare", IEEE Network Magazine, Junho de 1998.

[4] Braden, B., "Introduction to the ASP execution environment", tech report, USC Information Science Institute, Fevereiro de 2000.

[5] Calvert, K. L., "Architectural framework for active networks", Draft de RFC, julho de 1999.

[6] Calvert, K. L. et al, "Directions in active networks", IEEE Communications Magazine, outubro de 1998.

[7] Chen, T. M., e Jackson, A. W., "Active and programmable networks", Guest editorial, IEEE Network Magazine, Junho de 1998.

[8] Choi, S. et al, "Design of a flexible open plataform for high performance active networks", Universidade de Washington, 1998.

[9] Decasper, D. et al, "A scalable, high performance active network node", outubro de 1998.

[10] Decasper, D., "DAN - distributed code caching for active networks", Proceedings of Infocom'98, Abril de 1998.

[11] Hartman, J. H. *et al*, "Joust, a platform for liquid software", IEEE Network Magazine, Julho de 1998.

[12] Merugu, S. et al, "Bowman and CANEs: implementation of an active network", 1998.

[13] Muhugusa, M., "Implementing Distributed Services with Mobile Code: The Case of the Messenger", Proceedings of International Conference on Parallel and Distributed Systems, Julho de 1998.

[14] Psounis, K., "Active networks: applications, security, safety and architectures", IEEE Communications Surveys, primeiro quadrimestre de 1999.

[15] Schwartz, B. I., "Active Network Message Protocol – RFC draft", BBN, Junho de 1997.

[16] Tennenhouse, D. L. et al, "A survey of active network reasearch", IEEE Communications Magazine, janeiro de 1997.

[17] Wetherall, D. *et al*, "ANTS: a toolkit for building and dynamically deploying network protocols", IEEE OPENARCH'98, Abril de 1998.

[18] Wolf, T. e Decasper, D., "CPU scheduling for active processing using feedback deficit round robin", Universidade de Washington, 1997.