Análise da Sinalização na Arquitetura Proposta no IETF ANIMA: o GRASP

João Batista Silva Martins, Jéferson Campos Nobre

[1]Escola Politécnica – Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo – RS – Brasil

jbmv186@gmail.com, jcnobre@unisinos.br

Abstract. *Advancing the technology and complexity of computer networks creates the need for new management mechanisms for such networks. Human intervention will not be more effective, and it is necessary to automate certain processes. Autonomic networks present themselves to operate in complex networks and make decisions without needing a network administrator and passing on the responsibility of solving problems for the autonomic network nodes. The Autonomic Networking Integrated Model and Approach ANIMA Working Group (WG) has as its goal the proposition of an architecture for autonomic networks. In this architecture, the signaling is performed by the GeneRic Autonomic Signaling Protocol, protocol responsible for enabling nodes of a network to act autonomously. This work presents a study on ANIMA, focusing on the use of GRASP.*

Resumo. *O avanço da tecnologia e complexidade de redes de computadores cria a necessidade de novos mecanismos de gerenciamento para tais redes. A intervenção humana não será mais efetiva, sendo necessário automatizar determinados processos. As redes autonômicas se apresentam para atuar em redes complexas e tomar decisões sem necessitar de um administrador de rede e repassando a responsabilidade de resolver problemas para os nós da rede autonômica. O Autonomic Networking Integrated Model and Approach ANIMA Working Group (WG) tem por objetivo propor uma arquitetura para redes autonômicas. Nessa arquitetura, a sinalização é realizada pelo GeneRic Autonomic Signaling Protocol, protocolo responsável por habilitar nós de uma rede para atuar autonomicamente. Este trabalho apresenta um estudo sobre o ANIMA, com foco na utilização do GRASP.*

1. Introdução

O gerenciamento de sistemas, ao passar dos anos, está se tornando cada vez mais complexo devido as novas tecnologias e serviços para atender a alta demanda por sistemas mais rápidos, potentes e com maior capacidade de armazenamento. A evolução das redes de computadores traz obstáculos a serem vencidos, como complexidade, heterogeneidade e falta de mão de obra para atuar no gerenciamento, motivando o desenvolvimento de um novo paradigma de gerenciamento de redes.

Dispositivos que integram uma rede de computadores, em sua grande maioria, necessitam de intervenção humana para que sejam configurados conforme a demanda da rede, sendo suscetível a erros. O grande problema a ser discutido se deve ao tamanho

das redes, pois conforme seu tamanho, o trabalho de identificação de falhas e erros de configuração se torna inviável.

O propósito deste trabalho é apresentar um novo modelo de redes de computadores proposto pelo *Autonomic Networking Integrated Model and Approach* ANIMA *Working Group* (WG) no Internet Engineering Task Force (IETF), onde este modelo tem como principal objetivo tornar as operações a atuarem de forma autonômica. O modelo deve permitir que um nó seja capaz de se auto-gerenciar [Behringer et al. 2015], como assumir configurações de rede automaticamente, ser capaz de se recuperar de problemas e se auto adaptar a mudanças, ser capaz de entender a rede e otimizar seu comportamento e ser capaz de se proteger contra ataques em potencial.

O restante deste trabalho está organizado da seguinte maneira. A Seção 2 traz o referencial teórico do trabalho, abordando as redes autonômicas e o projeto realizado pelo ANIMA WG. A Seção 3 detalha o GRASP, protocolo que permite os nós de rede atuarem autonomicamente em uma rede, explicando suas propriedades, mecanismos e características das mensagens utilizadas pelos nós. A Seção 4 apresenta as características de segurança do GRASP, explicando os mecanismos necessários para suprir as necessidades de segurança do protocolo. A Seção 5, apresenta a conclusão do trabalho.

2. Redes Autonômicas

O conceito de computação autonômica pode ser comparado ao sistema nervoso do corpo humano, que se comporta de forma autonômica. Este sistema é capaz de atuar sobre o corpo humano de forma inconsciente preparando este para atividades do momento, cuidar de funções vitais do corpo humano [Kephart and Chess 2003] sem qualquer iniciativa consciente do organismo. Em uma situação de perigo, o sistema nervoso autonômico é capaz de adequar o corpo para esta situação fazendo com que fique pronto para agir com rapidez, força e determinação.

A computação autonômica tem por objetivo trazer um novo modelo de controle e supervisão para sistemas computacionais, sendo orientados com objetivos de alto nível e políticas pelos administradores tornando-os capazes de gerenciar a si próprios. O ponto principal da computação autonômica é o auto-gerenciamento, repassando a responsabilidade dos detalhes de operação e manutenção do sistema para os dispositivos, isentando a atuação dos administradores humanos.

O sistema autonômico [Horn 2001] é formado pelos Elementos Autonômicos (EA), estes sendo a menor parte do sistema. Os EAs se comunicam e interagem entre si, utilizando a capacidade de auto-gerenciamento e cumprem suas funções autonômicas por meio da execução de ações específicas ao presenciar qualquer alteração de ambiente ou outra situação que depende de ajustes do sistema. A Figura 1 mostra como é a estrutura de um EA.

Os EAs [Horn 2001] podem ser divididos em dois sub-elementos, o gerenciador autonômico e o elemento gerenciado. O gerenciador autonômico é responsável por realizar o monitoramento de um ou mais elementos gerenciados, tendo como objetivo possibilitar que cada componente trabalhe da melhor forma e que ofereça qualidade em seu serviço. O elemento gerenciado pode ser representado por qualquer recurso ou *Switchs*. O elemento possui dois componentes responsáveis pela gestão do mesmo, os sensores,

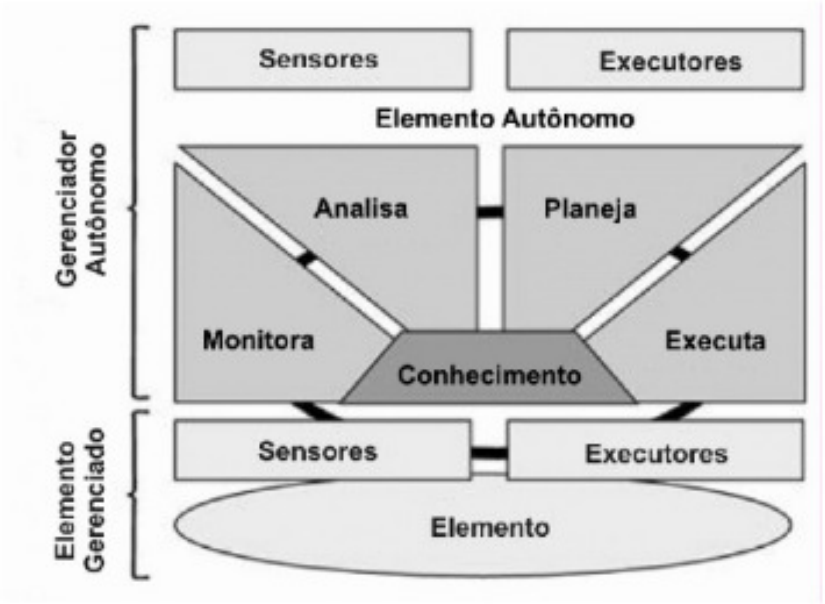

Figura 1. Elemento Autonômico. Adaptado de [Kephart and Chess 2003]

responsáveis pela coleta de informações de um elemento, e os executores, responsáveis pela modificação de um estado de um elemento.

O gerenciador autonômico executa quatro fases no contexto do sistema autonômico. A primeira fase é o monitoramento, onde os sensores realizam a coleta dos dados e estes são enviados a base de conhecimento para análise. Nesta fase, o elemento autonômico torna-se capaz de conhecer o ambiente e o próprio elemento gerenciado. Na segunda fase, é realizada a análise baseada nas informações obtidas pelo monitoramento. Estas informações são comparadas com as já existentes na base de conhecimento para que seja possível identificar algum problema ou modificações do sistema, e assim prever alguma situação futura e a necessidade de mudança. A terceira fase é o planejamento, estratégias de reconfiguração do sistema são planejadas com base nos resultados da análise e as mudanças detectadas. A última fase é a execução, os executores implementam as estratégias de reconfiguração alinhadas na fase de planejamento. As estratégias executadas são armazenadas na base de conhecimento.

A integração entre os elementos autonômicos é de fundamental importância para o funcionamento do sistema. O elemento realiza este relacionamento entre os demais elementos do sistema através de estágios. O primeiro estágio é a especificação, onde cada elemento autonômico possui um conjunto de requisitos ou serviços de entrada e após executar suas funções o elemento responde com um conjunto de serviços de saída. Um padrão deve ser definido para este conjunto de serviços, tanto de entrada como de saída, para que seja possível a comunicação entre os elementos.

O próximo estágio trata a localização, que permite cada elemento autonômico possa localizar serviços de entrada de que precisa, da mesma forma que outros elementos necessitam localizar serviços de saída. Os elementos são cadastrados em um local onde

as informações como tipo de serviço, funcionalidade e os requisitos do elemento são armazenados, sendo possível que outros elementos autonômicos do sistema tenham acesso a este local.

A negociação entre os elementos autonômicos é o próximo estágio, necessário para que o elemento cliente possa solicitar o serviço ao elemento fornecedor. Ao encontrar o elemento para fornecer o serviço, o elemento cliente inicia a negociação, e caso houver recursos suficientes e não tenha mais nenhum elemento solicitando o mesmo serviço, o elemento fornecedor disponibiliza o serviço ao elemento cliente. Caso não houver recursos disponíveis, a aplicação de um método de fila deverá ser feita. Após a negociação, a provisão dos serviços e recursos para o elemento cliente é realizada. Um registro deste elemento é feito em uma lista de execução, para que numa eventual necessidade do elemento solicitar algum serviço.

No estágio de operação, o elemento cliente opera conforme regras ajustadas na negociação e o elemento fornecedor fiscaliza esta operação. A fiscalização garante que o elemento cliente siga o que foi acordado e caso houver alguma violação o elemento pode receber uma suspensão temporária ou então o cancelamento do acordo entre os elementos. Na fase de terminação, os recursos são liberados e o acordo entre os elementos finalizados [Kephart and Chess 2003]. O acordo entre os elementos é cumprido até o final e os dados deste relacionamento são armazenados para futuras consultas de outros elementos.

Um sistema para se tornar autonômico deve passar por uma série de mudanças de forma progressiva. Este processo de transformação deverá ser feita de forma planejada e cuidadosa [Muller et al. 2006]. A evolução deste processo é dividido em níveis, e conforme o sistema vai evoluindo em seu processo de transformação, um novo nível é atingido até chegar a transformação completa. O nível básico se caracteriza pela atuação em todos os processos pelo administrador de rede de forma manual, onde ele é responsável pela configuração, monitoramento e correção de falhas. O nível gerenciado apresenta tecnologias de gerenciamento, permitindo que a coleta de dados possa ser desempenhada pelo próprio sistema e com estes dados os administradores de rede possam planejar e tomar decisões. O nível de previsão é caracterizado pela comunicação entre os elementos e tecnologias que o sistema possui, tornando possível a previsão de mudanças e planejamento de ações. O nível de adaptação é responsável por possibilitar o sistema a atuar e tomar decisões por conta própria baseado nas previsões e avisos. O último nível é o Nível Autonômico, onde existe a presença de um auto-gerenciamento completo. O que difere este nível do adaptativo é que neste nível as modificações são feitas de acordo com as mudanças no ambiente e políticas de negócios implantadas no sistema.

Em redes de computadores, os conceitos de computação autonômica podem ser implementados, formando o conceito de redes autonômicas. As redes de computadores passam a ser capazes de realizar o auto-gerenciamento de seus elementos e conexões entre si. O gerente humano não é necessário para a execução dos serviços e funções de gerenciamento de rede, tornando estas operações de forma transparente para os usuários. A aprendizagem por meio das ações praticadas e a análise dos resultados obtidos, além da execução automática de tarefas, caracteriza-se o aspecto autonômico deste tipo de rede.

3. Autonomic Networking Integrated Model and Approach

O grupo de trabalho ANIMA [Carpenter et al. 2017] tem por objetivo desenvolver um sistema de funções autonômicas. Está sendo desenvolvido um paradigma de controle onde capacitam os processos de rede a coordenar decisões e tornar capazes de aplicar estas decisões em elementos de rede, com base em várias fontes de informações fornecidas pelo administrador ou obtidas através de protocolos existentes. O objetivo inicial é especificar um conjunto mínimo de componentes de infra-estrutura reutilizáveis específicos para suporte a interações autonômicas entre dispositivos e especificar a aplicação destes componentes. Nesta parte do trabalho, um modelo de referência ANIMA será analisado, com o objetivo de entender os principais componentes de uma rede autonômica e as interações entre os nós desta rede e o funcionamento de funções autonômicas em redes tradicionais.

Os sistemas autonômicos, conforme [Behringer et al. 2015], tem como principal objetivo o funcionamento de uma rede com o mínimo de intervenção humana pelos administradores de rede. Também possibilita ao sistema capacidades de auto gestão, incluindo auto configuração, auto otimização, auto proteção e auto recuperação. Neste contexto, o termo autonômico se aplica a qualquer processo ou dispositivo que receba orientações em alto nível por meio de uma entidade central através de *Intent*[1] e que sejam capazes de se adaptar a um ambiente em mudança.

O sistema autonômico é composto pelos nós autonômicos, estes classificados como os nós da rede onde não requerem configuração e podem operar em qualquer camada da pilha de rede. Cada nó autonômico possui agentes de serviço autonômico, responsáveis por implementar uma função autonômica. Um conjunto de nós autonômicos formam uma rede autonômica, compostas exclusivamente por estes nós. Cada rede pode conter um ou mais domínios autonômicos. Dentro de um domínio, existe um conjunto de nós autonômicos que instanciam a mesma *Intent*. A *Intent* é responsável por operar a rede, trata-se de uma política abstrata e de alto nível onde tem como escopo um domínio autonômico e é definida e fornecida por uma entidade central. Ela não possui configuração ou informação para um nó específico, mas pode conter informações pertencentes a um nó específico, como um *Switch*[2] de borda por exemplo, ou algum nó que execute uma função específica.

Como principais objetivos das redes autonômicas, pode-se considerar as seguintes questões: elas devem ser consideradas em coexistência com outros paradigmas de gerenciamento de rede como o SNMP (*Simple Network Management Protocol*), NETCONF (*Network Configuration*) e SDN (*Software Defined Networking*), deverá ser segura por padrão utilizando identidade de domínio em cada nó da rede autonômica para garantir a autenticidade do nó em questão, minimizar as dependências em elementos centrais e problemas devem ser resolvidos de forma distribuída, permitir que funções autonômicas residam em qualquer camada de rede e dar suporte e compreender por completo ao ciclo de vida de um dispositivo (fabricação, testes inicias até a implantação, testes, soluções de problemas e desativação).

As funções autonômicas existentes hoje em redes de computadores necessitam de uma padronização. Dependendo de cada cenário, cada função tem seus próprios mecanis-

[1] Política abstrata e de alto nível usada para operar a rede.

[2] Equipamento utilizado para distribuir informações em uma rede.

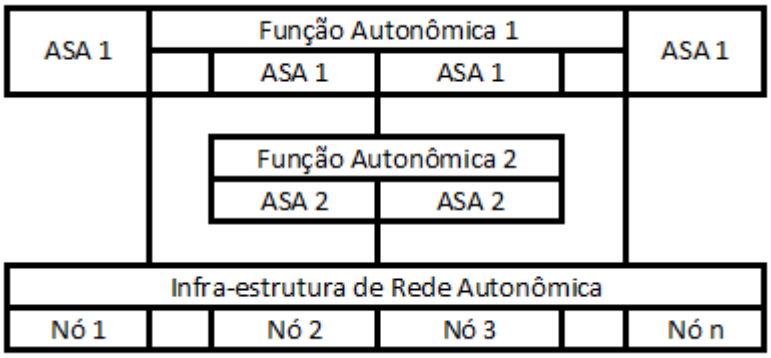

Figura 2. Modelo de um Nó Autonômico

mos de descoberta, identificação de nós, negociação, transporte, mensagens, mecanismos de segurança e interfaces de gerenciamento não autonômicos. Neste contexto, pode-se afirmar que não existe uma infra-estrutura comum para funções distribuídas.

Em documentação [Carpenter et al. 2017] do grupo de trabalho ANIMA, nos fornece uma visão de alto nível de uma Rede Autonômica, conforme a Figura 2. Uma Rede Autonômica possui uma série de nós autonômicos capazes de interagir uns com os outros, fornecendo um conjunto comum de recursos em toda a rede denominada infra-estrutura de rede autonômica (ANI - *Autonomic Networking Infrastructure*). Esta infra-estrutura fornece funções como nomeação, endereçamento, negociação, sincronização, descobertas e mensagens.

As entidades chaves de uma função autonômica são chamados de Agente de Serviço Autonômico (ASA - *Autonomic Service Agents*), que são instanciados nos nós de uma rede autonômica. As funções autonômicas abrangem toda a rede, bem como a infra-estrutura de rede autonômica. Um nó sempre implementa o ANI e podem ter um ou mais agentes de serviço autonômico.

A arquitetura interna de um elemento de rede segue o documento "*Autonomic Networking: Definitions and Goals*", [Behringer et al. 2015] onde trata sobre as fontes de informações que um agente de serviços autonômicos pode atingir, como o autoconhecimento, o conhecimento da rede através da descoberta, o *Intent* e os *loops de feedback*[3]. Um nó autonômico está subdivido em dois níveis, o nível dos agentes de serviços autonômicos e nível de infra-estrutura. A Figura 3 demonstra um modelo de nó autonômico.

O nível de infra-estrutura é responsável por armazenar estrutura de dados específicas do nó, onde constam as informações de confiança sobre ela mesma e seus nós pares na rede e um conjunto genérico de funções. A infra-estrutura como um todo deve dar suporte para vários modelos de agentes de serviço autonômico, por isso deve ser genérica. Já o nível dos ASAs é responsável por implementar o termo autonômico [Behringer et al. 2015] nos nós da rede, utilizando os serviços e as estruturas de dados disponibilizados no nível de infra-estrutura.

[3]Elementos de controle fora do nó que podem interagir com nós autonômicos através de loops de realimentação.

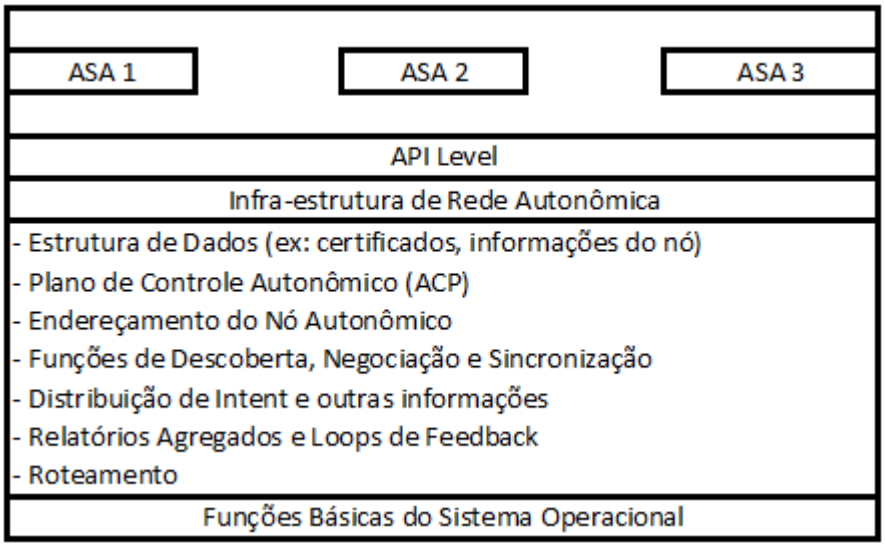

Figura 3. Visão de Alto Nível de uma Rede Autonômica

As funções autonômicas necessitam de uma infra-estrutura estável e robusta para se comunicar. O plano de controle autonômico (ACP - *Autonomic Control Plane*) [Behringer et al. 2017] oferece as funções autonômicas o suporte e segurança necessária para que seja utilizado apenas um mecanismo de descoberta para atingir todos os nós da rede.

O plano de controle autonômico permite que o dispositivo realize o *bootstraping*[4] de forma segura, sem que seja necessário receber configurações de rede, pois o dispositivo receberá um certificado [Pritikin et al. 2017] do domínio autonômico onde pertence. Além de fornecer segurança para as funções autonômicas e possibilitar o *bootstraping* seguro, o ACP permite o alcance permanente sobre o plano de dados de uma rede autonômica possibilitando a conectividade ao plano de dados mesmo que ocorra alguma alteração incorreta nas configurações do dispositivo da rede autonômica.

Generic Autonomic Signaling Protocol

Discutido o modelo de referência para redes autonômicas [Carpenter et al. 2017], sendo como base o documento de estudo do Grupo ANIMA, o GRASP (*Generic Autonomic Signaling Protocol*) [Bormann et al. 2017] vem como solução para implementação real de redes autonômicas. Um parâmetro ou conjunto de parâmetros serão definidos para que seja possível a descoberta, a sinalização e a negociação entre os nós autonômicos da rede, bem como as configurações básicas de endereçamento e roteamento através de um objetivo técnico.

O GRASP é implementado em cada ASA, o qual pertencerá a uma infra-estrutura autonômica. Para que este nó possa se comunicar e confiar nos outros nós da rede autonômica, um plano de controle autonômico (ACP - *Autonomic Control Plane*) será disponibilizado pela infra-estrutura autonômica [Behringer et al. 2017]. Uma Interface de Programação de Aplicação (API - *Application Programming Interface*) será necessária para que haja integração entre o GRASP e os Agentes de Serviços Autonômicos possa

[4]Termo utilizado para inicialização do sistema.

ocorrer. Dentro deste contexto, o ASA executa em espaço de usuário utilizando uma biblioteca do GRASP dentro da API e assim seria capaz de se comunicar através de chamadas de sistema com as principais funções GRASP.

Informações sobre interfaces de redes utilizadas e endereços da rede onde o nó atuará, será disponibilizado via tabela de adjacências [Carpenter et al. 2017]. O ACP é responsável por disponibilizar esta tabela para os nós da rede, caso contrário o próprio GRASP se encarrega destas informações. Para que o GRASP possa atuar em diferentes plataformas, conforme [Bormann et al. 2017], deve seguir um design geral. Este design cita regras de como deve ser a estrutura onde o GRASP será implementado. A plataforma que será utilizada deve ser genérica, pois o protocolo trabalha com um design genérico e independente do conteúdo de sincronização e negociação.

Com relação a segurança, o GRASP não disponibiliza nenhum mecanismo e a estrutura necessita de uma infra-estrutura de segurança a parte. O GRASP deverá ser executado em uma instância como um módulo separado, fornecendo uma API [Carpenter and Gong 2017] para que seja possível interagir com os ASAs.

Os mecanismos GRASP utilizados pelos agentes são construídos em torno de objetivos GRASP, definidos como estrutura de dados contendo informações administrativa.

4. Propriedades e Mecanismos

Os mecanismos são definidos em conjunto no GRASP, porém são mecanismos separados. A ação de descoberta (M_DISCOVERY) será seguido de uma ação de negociação (M_REQ_NEG) ou uma ação de sincronização (M_REQ_SYN) e os resultados obtidos na descoberta podem ser utilizados pelo protocolo de negociação para decidir em qual agente de serviço autonômico irá negociar. Todas as mensagens GRASP também devem seguir em seu formato o padrão de objetos CBOR (Concise Binary Object Representation) [Bormann et al. 2013].

O procedimento de descoberta inicia com uma mensagem *multicast*[4] (M_DISCOVERY) para todos os nós vizinhos da rede. Estes nós, por sua vez, devem apoiar o objetivo da descoberta com uma mensagem de resposta (M_RESPONSE).

Para que o processo termine, será considerado o tempo limite de *loop* e uma contagem (GRASP_DEF_TIMEOUT), conforme tabela 1, para definir se teve sucesso ou não. O endereço *multicast* utilizado se encontra na constante ALL_GRASP_NEIGHBORS, e atinge todos os nós GRASP da rede e precisam responder a mensagem de descoberta. Ao descobrir um nó GRASP que responda a mensagem de descoberta e que suporte um objetivo específico, o nó irá armazenar as informações do nó de destino em cache, incluindo o índice de interface através do qual ele foi descoberto, sendo que estas informações podem ser utilizadas em uma futura negociação ou sincronização.

O procedimento de retransmissão de descoberta inicia a partir de uma instância do protocolo GRASP com várias interfaces, onde esta instância deve apoiar a descoberta em todas as interfaces. Esta instância é definida como uma ”instância de retransmissão”. Ao receber uma mensagem de descoberta em uma determinada interface para um objetivo específico que não suporta e para o qual não tenha armazenado em cache um mecanismo

[4]Entrega de informação para múltiplos destinatários simultaneamente

de resposta de descoberta, a consulta é retransmitida reeditando uma nova mensagem de descoberta e enviado para suas outras interfaces, para que seja possível distribuir na rede via mensagem *multicast*. A mensagem de descoberta retransmitida deverá ter a mesma ID de sessão da mensagem original e deve ser marcada com o endereço IP do seu iniciador original.

O tempo limite definido pelo iniciador deve assumir pelo menos o valor igual ao encontrado na constante GRASP_DEF_TIMEOUT em milissegundos. Para iniciar os procedimentos de uma negociação, um iniciador de negociação envia uma mensagem de solicitação
(M_REQ_NEG) a um outro ASA, incluindo um objetivo de negociação específico. Caso não receber nenhuma resposta dentro do tempo limite, o pedido de negociação será repetido utilizando uma nova ID de sessão. Se o agente de serviço autonômico aplicar imediatamente a configuração solicitada, ele dará uma resposta positiva imediatamente (O_ACCEPT) via mensagem M_END. Com esta mensagem, a fase de negociação será encerrada. Caso contrário, será iniciada uma nova negociação através da mensagem M_NEGOTIATE, respondendo com uma proposta de configuração alternativa que poderá ser aplicada e uma nova negociação bidirecional será iniciada para alcançar o compromisso entre dois agentes de serviços autonômicos.

A negociação entre os agentes será encerrado quando um dos nós da negociação enviar uma mensagem de encerramento de negociação (M_END), onde terá a opção aceitar (O_ACCEPT) ou recusar (O_DECLINE). O encerramento da negociação também pode ser feito ao alcançar o tempo limite de resposta devido alguma falha na comunicação. Um agente de serviço autonômico pode participar de negociações simultâneas sobre diferentes objetivos, isso é possível pois o GRASP pode ser utilizado em um modo *multi-thread*[5].

Para iniciar o procedimento de sincronização, o nó abre uma conexão de transporte para o agente de serviço autonômico de destino usando o endereço, protocolo e porta obtidas durante a descoberta. Após esta operação, uma mensagem de solicitação de sincronização (M_REQ_SYN) será enviada a um agente de serviço autonômico de destino contendo um objetivo de sincronização específico. O agente de destino responde com uma mensagem de sincronização (M_SYNCH) que contém o valor atual do objetivo de sincronização solicitado. Não havendo nenhuma resposta dentro do tempo limite da constante GRASP_DEF_TIMEOUT, o pedido de sincronização pode ser realizado novamente utilizando uma nova ID de sessão. Em alguns casos, onde será necessário envio de dados de sincronização para grandes grupos de nós, o *flooding*[6] de sincronização será utilizado. A mensagem será enviada via *multicast* para todos os nós da rede, utilizando a variável ALL_GRASP_NEIGHBORS e irá conter uma ou mais objetivos de sincronização. O iniciador da mensagem de *flooding* de sincronização irá definir a contagem de *loops* (GRASP_DEF_LOOPCT) para que não ocasione *loops* na rede. Um dispositivo que utilize o GRASP e possua várias interfaces, deve receber a mensagem de *flooding* de sincronização em uma interface e distribuir a mesma para suas outras. A mensagem retransmitida deve ter a mesma ID de sessão da mensagem original e marcado com o endereço IP de seu iniciador original. Nas mensagens utilizadas pelos mecanismos de descoberta, negociação e sincronização, variáveis são utilizadas no envio destas

[5]Capacidade de executar várias tarefas simultaneamente sem que interfira uma na outra.
[6]Designação de envio de uma grande quantidade de mensagens para os destinatários.

mensagens. As variáveis GRASP são utilizadas para definir quais nós de rede serão atingidos pelos mecanismos entre outras utilidades. Na tabela 1 são definidas as variáveis utilizadas, bem como uma breve explicação da função de cada uma.

Tabela 1. Variáveis GRASP

Variável	Descrição
ALL_GRASP_NEIGHBORS	Endereço multicasts usado por um dispositivo habilitado para o protocolo GRASP para descobrir dispositivos vizinhos que também estão habilitados com GRASP. IPv6 Multicast Address: TDB1 IPv4 Multicast Address: TDB2
GRASP_LISTEN_PORT	Porta utilizada para comunicação multicast e unicast, utilizada pelos dispositivos habilitados para o GRASP. Utiliza a varíavel TDB3.
GRASP_DEF_TIMEOUT	Tempo limite padrão para definir se uma descoberta falhou. Tempo de 60000 milissegundos
GRASP_DEF_LOOPCT	Contagem padrão de quantidade de loops, para determinar que uma negociação teve falha e para evitar looping de mensagens. Número de loops realizados é 6.
GRASP_DEF_MAX_SIZE	Tamanho máximo da mensagem. Em bytes, valor padrão de 2048.

Cada nova mensagem de descoberta, *flooding* de sincronização ou negociação, uma nova ID de sessão será criada, utilizando um valor de 32 bits. Esta identificação será gerada utilizando um algoritmo pseudo-aleatório, utilizando criptografia forte [Eastlake 3rd and Crocker 2005] e deverá acompanhar as mensagens trocadas pelos dispositivos. A ID de sessão, ao ser atribuída a mensagem, o GRASP deve verificar se o valor ainda não está em uso e que não tenha sido utilizado recentemente. Se caso for gerado o mesmo identificador de sessão para os dois nós, o nó receptor deve marcar a sessão com o endereço IP do iniciador, para evitar erros.

5. Mensagens e Opções GRASP

As mensagens utilizadas pelo GRASP recebem o mesmo formato de cabeçalho e um formato variável para suas opções. As mensagens utilizadas para comunicação entre os nós são: Descoberta e Resposta da Descoberta (M_DISCOVERY e M_RESPONSE), Requisição de Negociação , Negociação, Confirma Espera e Fim de Negociação (M_REQ_NEG, M_NEGOTIATE, M_WAIT, M_END), Requisição de Sincronização, Sincronização e *Flood* de Sincronização (M_REQ_SYN, M_SYNCH e M_FLOOD) e Sem Operação e Mensagem Inválida (M_NOOP e M_INVALID). A Representação Concisa de Objetos Binários (CBOR - Concise Binary Object Representation) [Bormann et al. 2013] é utilizada como padrão em mensagens GRASP. Toda a mensagem GRASP possui uma ID de sessão, exceto a mensagem Sem Operação. As opções são apresentadas no campo de opções, conforme Figura 4.

```
grasp-message = (message .within message-structure / noop-message)
- message-structure = [MESSAGE\_TYPE, session-id, \?initiator,
*grasp-option]
```

Figura 4. Estrutura de uma mensagem GRASP

Serão descartadas qualquer opção fora do padrão com o MESSAGE_TYPE. Se caso um MESSAGE_TYPE recebido via mensagem *unicast*[7] uma mensagem inválida será retornada. Se caso for recebido via mensagem *multicast*, ele pode ser registrado e deve ser descartado. Os nós que utilizam mensagens GRASP devem ser capazes de receber mensagens *unicast* de tamanho definido no parâmetro GRASP_DEF_MAX_SIZE, em bytes. Se caso o tamanho for maior que o definido, este deve ser explicito e permitido para o objetivo em questão. Nas tabelas 2 e 3, serão definidas as mensagens utilizadas pelo GRASP, suas descrições e o formato utilizado nas trocas de mensagens.

As opções nas mensagens GRASP também devem seguir o padrão de objetos CBOR (Concise Binary Object Representation) [Bormann et al. 2013] e devem seguir este padrão para que não haja sobrecarga. A opção deve iniciar com um valor inteiro identificando o tipo de opção específico carregado. Estas opções devem ser utilizadas nas trocas de mensagens de sinalização e negociação. Na tabela 4, lista as opções disponíveis para as trocas de mensagens.

Em mensagens GRASP, são utilizados os objetivos, é colocado a disposição as opções que serão utilizadas para identificar os objetivos para fins de descoberta, negociação ou sincronização. Todos os objetivos são identificados por um nome único, utilizando uma sequência de caracteres no padrão UTF-8 [RFC3629], por exemplo, ”EX1”, ”225:EX1”, ”exemplo.com:EX1”e ”suporte@exemplo.net:EX1”. Em nomes de objetivos particulares, deve ser incluído os dois pontos (:) em sua estrutura de identificação e em casos de objetivos genéricos, este não será utilizado. Um exemplo de estrutura de um objetivo deve ser seguido, conforme Figura 5

```
objective = [objective-name, objective-flags,
loop-count, ?objective-value]
objective-name = text
objective-value = any
loop-count = 0..255
```

Figura 5. Estrutura de um Objetivo

O campo ”loop-count”será utilizado para definir o encerramento do procedimentos de negociação, descoberta e flooding. O campo ”objective-value”expressa o valor real de um objetivo de negociação ou sincronização. Este campo é opcional em mensagens de descoberta e resposta a descoberta. O campo ”objective-flags”traz qual tipo de mensagem será abordada nas trocas de mensagens entre os nós. Os tipos são definidos em bits, e podem ser combinados caso a mensagem trata mais do que um tipo de mensagem (F_DISC e F_SYNCH, F_DISC e F_NEG, F_DISC e F_NEG e F_NEG_DRY), conforme Figura 6.

[7]Entrega de pacote para apenas um destino.

Tabela 2. Mensagens GRASP

Mensagem	Descrição	Formato CBOR
Descoberta	O iniciador de descoberta manda uma mensagem para todos na rede, via porta UDP, e recebe os resultados da descoberta pela porta TCP.	discovery-message = [M_DISCOVERY, session-id, initiator, objective]
Resposta de Descoberta	Quando a menseagem de descoberta é atendida, uma mensagem de resposta a descoberta retorna para o nó que enviou a primeira mensagem.	response-message = [M_RESPONSE, session-id, initiator, ttl, (+locator-option //divert-option, ?objective)]
Solicitação	Um nó envia uma mensagem de solicitação de negociação ou sincronização para o endereço Unicast de destino.	request-negotiation-message = [M_REQ_NEG, session-id, objective] request-synchronization-message = [M_REQ_SYN, session-id, objective]
Negociação	Mensagem enviada após receber uma solicitação de negociação. Deve conter a opção de objetivo de negociação com seu valor atualizado de acordo com o progresso da negociação.	negotiate-message = [M_NEGOTIATE, session-id,objective]
Final de Negociação	O nó de destino da negociação envia esta mensagem para fechar a negociação. A mensagem deverá ter opção de aceitação ou recusa de negociação.	end-message = [M_END, session-id, accept-option / decline-option]
Confirmação	Mensagem enviada pelo nó de destino caso necessite aguardar uma resposta de negociação adicional.	wait-message = [M_WAIT, session-id, waiting-time] waiting-time = 0..4294967295 em milisegundos
Sincronização	Um nó recebe uma solicitação de sincronização e envia ao nó remetente uma mensagem de sincronização Unicast com os dados para sincronização.	synch-message = [M_SYNCH, session-id, objective]
Flooding de Sincronização	Uma mensagem de sincronização em forma de Flooding é enviada a todos os nós da rede via endereço de multicast.	flood-message = [M_FLOOD, session-id, initiator, ttl, +[objective, (locator-option / []]] ttl = 0..4294967295 em milisegundos

Tabela 3. Mensagens GRASP - Mensagem Inválida e Sem Operação

Mensagem	Descrição	Formato
Inválida	Será enviada quando uma mensagem Unicast é recebida e considerada inválida.	invalid-message = [M_INVALID, session-id, ?any]
Sem Operação	Mensagem enviada quando houver necessidade de iniciar um socket para comunicação.	noop-message = [M_NOOP]

Tabela 4. Opções GRASP

Opção	Descrição	Formato
Desvio	Utilizada,para redirecionar uma solicitação GRASP para outro nó em mensagens de negociação ou sincronização.	divert-option = [O_DIVERT, +locator-option]/
Localizador IPv4	Utilizado para apresentar informações de acessibilidade para um ASA Autonômico, um dispositivo ou uma interface com endereços IPv4.	ipv4-locator-option = [O_IPv4_LOCATOR, ipv4-address, transport-proto, port-number] ipv4-address = bytes .size 4
Localizador IPv6	Utilizado para apresentar informações de acessibilidade para um ASA, um dispositivo ou uma interface, com endereços IPv6.	ipv6-locator-option = [O_IPv6_LOCATOR, ipv6-address, transport-proto, port-number] ipv6-address = bytes .size16
Localizador FQDN e URI	Utiliza o nome de domínio qualificado (FQDN) e o identificador de recurso uniforme (URI) para atingir o alvo	fqdn-locator-option = [O_FQDN_LOCATOR, text, transport-proto, port-number] uri-locator-option = [O_URI_LOCATOR, text, transport-proto, port-number]
Recusa	Utilizado para indicar ao nó remetente da negociação que o conteúdo proposto foi rejeitado.	decline-option = [O_DECLINE, ?reason] reason = texto; mensagem de erro opcional.
Aceitar	Utilizado para indicar ao nó remetente da negociação que o conteúdo proposto foi aceito.	accept-option = [O_ACCEPT]

6. Conclusão

A evolução das redes de computadores traz grandes necessidades de gerenciamento e utilização de mão de obra para manutenção dos elementos da rede. Estas necessidades podem custar valores expressivos. Um novo modelo de rede que possa contornar esta situação se torna de extrema importância. Estudos sobre novos modelos está em constante atualização, um deles podemos citar o modelo autonômico de redes como solução.

Em comparação com os padrões de redes de computadores atuais [Jiang et al. 2015], as redes autonômicas tem muito a contribuir futuramente. Faci-

```
objective-flags = uint .bits objective-flag
objective-flag = &(
F\_DISC: 0 ; válido para descoberta
F\_NEG: 1 ; válido para negociação
F\_SYNCH: 2 ; válido para sincronização
F\_NEG_DRY: 3 ; negociação é dry-run
)
```

Figura 6. Estrutura do Campo objective-flags

litar e otimizar o gerenciamento de redes de computadores tem se tornado uma tarefa complexa e que não será possível utilizar os mecanismos atuais de gerenciamento, abrindo uma brecha para a utilização de redes autonômicas.

A questão de segurança em redes autonômicas abre possibilidades para desenvolvimento de trabalhos futuros. A falta de segurança em determinadas trocas de mensagens entre os nós de uma rede autonômica, torna vulnerável a ataques em grande potencial.

Referências

Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., and Systems, C. (2015). RFC7575 - Autonomic Networking: Definitions and Design Goals.

Behringer, M., Systems, C., Eckert, T., Huawei, Bjarnason, S., and Networks, A. (2017). An Autonomic Control Plane.

Bormann, C., TZI, U. B., Carpenter, B., of Auckland, U., Liu, B., and Tech., H. (2017). A Generic Autonomic Signaling Protocol (GRASP).

Bormann, C., U. B. T., Hoffman, P., and Consortium, V. (2013). RFC7049 - Concise Binary Object Representation (CBOR).

Carpenter, B., Behringer, M., of Auckland, U., Eckert, T., Inc., F. T., Ciavaglia, L., Peloso, P., Nokia, Liu, B., Technologies, H., Nobre, J., of Rio Grande do Sul, F. U., and Strassner, J. (2017). ANIMA - A Reference Model for Autonomic Networking.

Carpenter, B., L. B. W. W. and Gong, X. (2017). Generic Autonomic Signaling Protocol Application Program Interface (GRASP API).

Eastlake 3rd, D., S. J. and Crocker, S. (2005). RFC4086 - Randomness Requirements for Security.

Horn, P. (2001). IBM Autonomic Manifesto.

Jiang, S., Tech., H., Carpenter, B., Auckland, U., Behringer, M., and Systems, C. (2015). RFC7576 - General Gap Analysis for Autonomic Networking.

Kephart, J. and Chess, D. (2003). The Vision of Autonomic Computing. *IEEE Computer Society*, pages 41–50.

Muller, H., O'Brien, L., Klein, M., and Wood, B. (2006). Autonomic Computing. Technical Note. Software Engineering Institute. (CMU/SEI-2006-TN-006).

Pritikin, M., Cisco, Richardson, M., SSW, Behringer, M., Bjarnason, S., Watsen, K., and Networks, J. (2017). Bootstrapping Remote Secure Key Infrastructures (BRSKI).