Gerência de Falhas Distribuída e Confiável Baseada em Clusters de Agentes

Aldri L. dos Santos
aldri@dcc.ufmg.br
Depto. de Ciência da Computação
Univ. Federal de Minas Gerais
Brasil

Elias P. Duarte Jr.
elias@inf.ufpr.br
Depto. de Informática
Univ. Federal do Paraná
Brasil

Glenn Mansfield
cyber@cysol.co.jp
Cyber Solutions Inc.
Aoba-ku Sendai Miyagi
Japão

Abstract

Most network monitoring systems only allow the examination of managed objects of fault-free agents. However it is often useful to examine MIB objects of a crashed or unreachable network element in order to determine why it is faulty. This work presents a new clustering architecture for SNMP agents that supports semi-active replication of managed objects. A cluster of agents provides fault-tolerant object functionality: replicated managed objects of a faulty agent of a given cluster may be accessed through a peer cluster. Furthermore, the cluster behaves as a cache of managed objects that reduces the impact of monitoring on network performance. The proposed architecture is structured in three layers. The lower layer corresponds to the managed objects at the network elements. The middle layer contains management entities called clusters that monitor and replicate managed objects. The upper layer allows the definition of management clusters as well as the relationship between clusters. A practical tool was implemented and is presented, as an application example we show how it was used to determine the occurrence of TCP SYN-Flooding attacks. The impact of replication on network performance is evaluated as well as a probabilistic analysis of replicated object consistency.

Keywords: Network Management, SNMP, Fault Management, Clustering, Data Replication.

Resumo

A maioria dos sistemas de monitoramento de redes somente permite examinar os objetos de gerência dos agentes livres de falhas. Contudo, muitas vezes é útil examinar os objetos de um elemento da rede para determinar a razão na qual esse elemento está falho ou inacessível. Este trabalho apresenta uma nova arquitetura de agrupamento para agentes SNMP que suportam replicação semi-ativa dos objetos de gerência. Um cluster de agentes implementa objetos tolerantes a falhas: objetos replicados de um agente falho que pertence a um cluster podem ser acessados através de seus clusters pares. Além disso, o cluster comporta-se como uma cache de objetos de gerência que reduz o impacto de monitoramento de desempenho da rede. A arquitetura proposta é estruturada em três camadas. A camada inferior corresponde aos objetos de gerência nos elementos da rede. A camada intermediária contém as entidades chamadas clusters que monitoram e replicam objetos de gerência. A camada superior permite a definição de clusters de gerenciamento assim como o relacionamento entre os clusters. Uma ferramenta prática é apresentada. Como aplicação exemplo, mostramos como a ferramenta foi usada para se determinar a ocorrência de ataques TCP SYN-Flooding. O impacto da replicação no desempenho da rede é avaliado assim como uma análise probabilística da consistência dos objetos replicados.

Palavras-chave: Gerência de Redes, SNMP, Gerência de Falhas, Agrupamento, Replicação de Dados.

1 Introdução

Gerência de falhas é uma das funções chaves dos sistemas de gerência de redes integradas. A finalidade da gerência de falhas é localizar, determinar as causas e, se possível, corrigir falhas na rede [1]. Diferentes abordagens têm sido propostas para realizar esta tarefa, incluindo a aplicação de diagnóstico em nível de sistema para monitoração de redes tolerantes a falhas [2, 3], correlação de alarmes [4], detecção pró-ativa de falhas [5], entre outras.

Este trabalho apresenta uma arquitetura de agrupamento de agentes para o *Simple Network Management Protocol* (SNMP) [6] que suporta replicação semi-ativa [7] dos objetos de gerência. Apesar da arquitetura apresentada levar em consideração o framework amplamente usado SNMPv3 [8], ela é uma arquitetura genérica que pode ser aplicada a qualquer framework de gerência distribuída.

A maioria dos sistemas de monitoração somente permite examinar os objetos de gerência de agentes livres de falhas. Os objetos de gerência refletem o estado das entidades gerenciadas. Assim, a leitura da MIB (*Management Information Base*) de um agente falho muitas vezes é útil para se determinar a razão na qual um determinado elemento da rede tornou-se falho ou inacessível. Neste trabalho apresentamos uma arquitetura para agrupamento de agentes em clusters. Um cluster de agentes oferece objetos tolerantes a falhas, ou seja, objetos replicados de um agente falho que pertence a um cluster pode ser acessado através de um chamado cluster par. Além disso, os clusters se comportam como uma cache dos objetos, reduzindo o impacto da monitoração no desempenho da rede. A arquitetura de agrupamento está atualmente em processo de tornar-se um padrão do IETF [9].

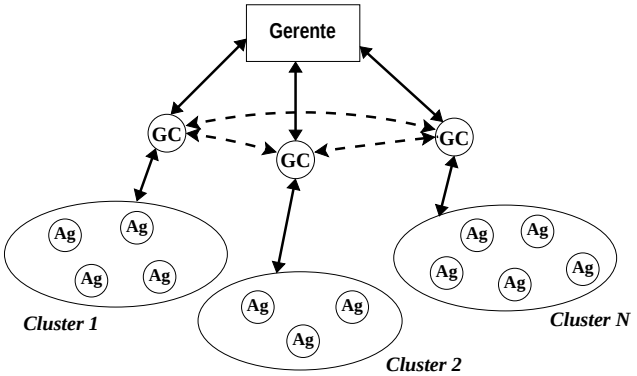

Figura 1: A arquitetura de agrupamento de agentes em três camadas.

A arquitetura proposta é estruturada em três camadas, como mostra a figura 1. A camada inferior corresponde aos objetos de gerência nos elementos da rede. A camada intermediária contém as entidades de gerência que executam a tarefa de monitorar um conjunto de objetos de gerência e replicá-los em outros clusters, chamados clusters pares.

Na camada superior são definidos todos os clusters de gerência assim como o relacionamento entre esses clusters.

Esta arquitetura permite que diferentes meios de comunicação sejam utilizados para enviar instâncias dos objetos replicados entre os clusters. Neste artigo, propomos o uso de um protocolo para comunicação de grupos [10, 11, 12], devido às suas propriedades e serviços como multicast confiável e gerência de grupos, entre outros.

Uma ferramenta prática foi implementada baseada em agentes NET-SNMP [13] que se comunicam usando a ferramenta Ensemble para comunicação de grupos [12]. Como uma aplicação exemplo, mostramos como a leitura de réplicas dos objetos com informações TCP de um agente submetido a um ataque TCP SYN-Flooding proporcionou a determinação das causas da sua falha. Uma avaliação probabilística do impacto da replicação no desempenho do sistema versus consistência das réplicas também é apresentada.

Trabalhos relacionados incluem [14, 15] que descrevem uma arquitetura em duas camadas na qual os objetos replicados são mantidos nos próprios elementos da rede. Em [16] J. Schönwälder apresenta uma abordagem para definição de grupos de agentes SNMP para implementação de controle de grupos sobre SNMP. Em [17] K.H. Lee apresenta um protocolo de comunicação de grupo que proporciona entrega ordenada de mensagens em um sistema de gerência de rede distribuída. Em [18] uma ferramenta de administração de redes baseada numa ferramenta para comunicação de grupos é descrita; esta ferramenta oferece soluções para manter configurações do serviço de nomes distribuídos consistentes, e permite distribuição de software para inúmeras máquinas.

O restante deste artigo está organizado da seguinte maneira. A seção 2 descreve a arquitetura de agrupamento de agentes. A seção 3 descreve o framework SNMP para replicação semi-ativa de objetos de gerência nos clusters de agentes. Na seção 4 uma ferramenta prática de gerência de falhas baseada na arquitetura proposta é apresentada. A seção 5 analisa o impacto da replicação de objetos no desempenho da rede. Conclusões e futuros trabalhos estão na seção 6.

2 Uma Arquitetura de Agrupamento de Agentes

Os sistemas de gerência de redes permitem a monitoração e o controle integrado de redes de computadores [6]. O *Simple Network Management Protocol version 3* (SNMPv3) é a arquitetura de gerência padrão da Internet [8]. Atualmente existe disponível um grande número de sistemas baseado em SNMP, tanto comerciais quanto de domínio público. A arquitetura de agrupamento de agentes apresentada abaixo leva em consideração o framework SNMPv3, visto a sua enorme popularidade no mundo. No entanto, a arquitetura proposta é genérica, e pode ser aplicada a qualquer outro framework de gerência distribuída.

Um sistema SNMPv3 é composto de entidades de gerência que se comunicam usando o protocolo de gerência. A arquitetura define uma base de informações de gerência (MIB) como uma coleção de objetos de gerência relacionados. Esses objetos são mantidos nos elementos gerenciados, permitindo às aplicações de gerência monitorar e controlar esses elementos. As entidades SNMPv3 têm sido tradicionalmente chamadas de gerentes e agentes. Cada elemento gerenciado contém um agente, que é uma entidade de gerência que tem acesso a instrumentação de gerência. Cada sistema tem no mínimo uma Estação de Gerência de Redes, que executa no mínimo uma entidade de gerência (gerente). Os gerentes são coleções de aplicações em nível do usuário, que oferecem serviços tais como avaliação de desempenho, diagnóstico de falhas, e contabilização, entre outros.

A gerência *distribuída* é amplamente reconhecida como uma necessidade para lidar com redes grandes, complexas e dinâmicas [19]. A arquitetura em três camadas tem se tornado popular para gerência distribuída, na qual entidades de gerência no nível médio são incluídas entre os agentes no nível inferior e os gerentes no nível superior. As entidades distribuídas assumem papéis como execução de scripts [20], monitoração de eventos [21], escalonamento de ações [22] ou relatórios de alarmes [23], entre outros.

2.1 Replicação de Objetos Baseada em Clusters de Agentes

Nos atuais sistemas de monitoração de redes, é impossível examinar os objetos de um agente que está falho ou inacessível. Abaixo, apresentamos uma arquitetura de agrupamento em três camadas para agentes SNMP que suportam replicação semi-ativa dos objetos de gerência.

Um cluster de agentes implementa objetos tolerantes a falhas, isto é, os objetos replicados de um agente falho, que pertence a um cluster, podem ser acessados através de um cluster par. Além disso, o cluster desempenha a função de uma cache de objetos de gerência, reduzindo o impacto da monitoração no desempenho da rede.

A abordagem de agrupamento empregada nesta arquitetura é similar à apresentada em [24] para reduzir o número de troca de mensagens em sistemas de gerência de redes móveis. Neste trabalho, utilizamos agrupamento de agentes para suportar replicação de objetos de gerência. Replicação [25] é uma ferramenta útil que tem sido aplicada para introduzir tolerância a falhas em sistemas distribuídos e banco de dados, sendo aplicada também para melhorar o desempenho desses sistemas [7]. Replicação é o processo de fazer uma réplica (uma cópia) de recursos, processos ou dados. Embora replicação seja um conceito intuitivo, ela requer técnicas sofisticadas para que sua implementação seja bem sucedida em um sistema distribuído [26].

Replicação semi-ativa [7, 15] é uma abordagem híbrida envolvendo características das replicações ativa e passiva [27]. Na replicação semi-ativa, os dados replicados podem ser acessados através de qualquer réplica. No entanto, somente o proprietário da réplica origem, conhecido como líder, determina a ordem na qual as réplicas são atualizadas.

A arquitetura proposta é estruturada em três camadas, como mostrado na figura 2 e descrita a seguir. As três camadas são chamadas, de baixo para cima, a camada *membros do cluster*, a camada *cluster*, e a camada *gerente*.

- A camada inferior mantém as entidades de gerência chamadas membros do cluster. Estas entidades têm os seus objetos de gerência monitorados e replicados por um cluster, como definido abaixo.
- A camada intermediária mantém as entidades de gerência chamadas clusters. Cada cluster tem a tarefa de monitorar um subconjunto de objetos de gerência de um conjunto de entidades de gerência e replicar esses objetos nos clusters pares.
- A camada superior mantém as aplicações de gerências que definem os clusters, assim como o relacionamento entre os clusters.

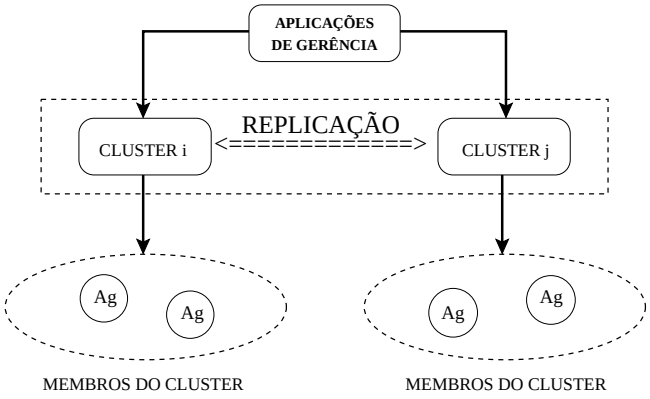

Figura 2: A arquitetura de replicação.

Os clusters de agentes são definidos na camada gerente. Clusters são entidades da camada intermediária que monitoram e replicam objetos dos agentes, que estão na camada inferior. A arquitetura permite que diferentes meios de comunicação, tais como comunicação de grupos, sejam utilizados para enviar valores dos objetos replicados entre os clusters.

2.2 Usando a Arquitetura de Agrupamento

A figura 3 mostra um exemplo no qual três clusters de agentes são monitorados pelos gerentes do cluster `GC1`, `GC2`, e `GC3`. O primeiro gerente do cluster, `GC1`, monitora os objetos dos agentes $Ag1$, $Ag2$, $Ag3$ e $Ag4$. O segundo, `GC2`, monitora os objetos dos agentes $Ag5$, $Ag6$, e $Ag7$; e `GC3` monitora os objetos dos agentes $Ag8$, $Ag9$, $Ag10$, $Ag11$, e $Ag12$. A figura 3 mostra o comportamento de `GC1`. Este além de monitorar os objetos de seus agentes, e manter réplicas desses objetos; também replica essas réplicas nos outros gerentes pares, como mostrado através de linhas pontilhas na figura. Considerando-se a

abordagem de replicação semi-ativa que é empregada, `GC1` é o lider de todos os objetos replicados do seu cluster de agentes.

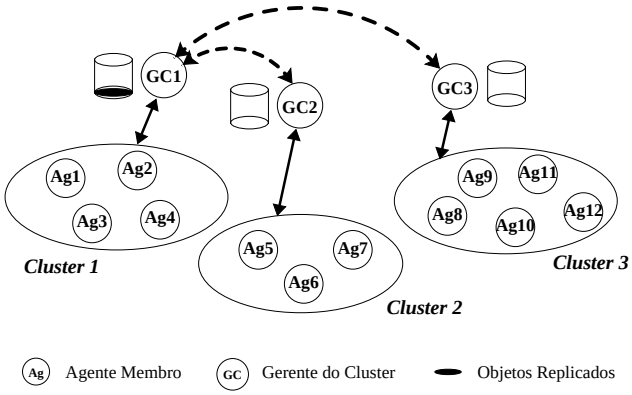

Figura 3: O funcionamento da arquitetura de agrupamento.

A figura 4 apresenta as etapas executadas por uma gerente para obter os objetos monitorados pelo gerente do cluster `GC1`. Neste caso, todos os gerentes de cluster (GC) contêm réplicas dos objetos monitorados por cada cluster. Primeiro, o gerente faz uma consulta a `GC1`. Se nenhuma resposta é obtida, como no exemplo, o gerente tem ainda a opção de obter a informação através dos gerentes clusters pares. Em seguida, o gerente faz uma consulta ao `GC2`, e assim obtém os objetos replicados de `GC1`.

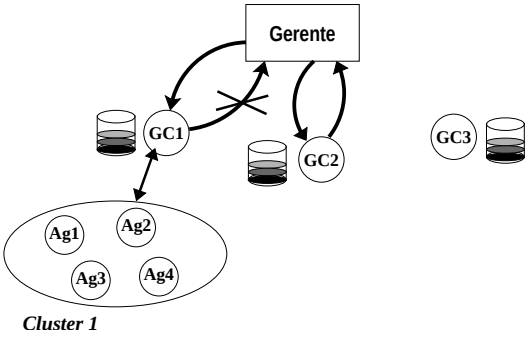

Figura 4: Consultando réplicas.

3 Um Framework SNMP para Replicação de Objetos em Clusters de Agentes

Nesta seção apresentamos um framework de agrupamento de agentes SNMP que suportam replicação de objetos de gerência. A parte da `Replic-MIB` chamada `replicObjects` que permite a definição e utilização dos clusters assim como a replicação é descrita. Esta MIB está dividida em dois grupos: `clusterDefinition` e `clusterReplica`, como mostrado na figura 5 e descrito abaixo. Um exemplo do emprego do framework é apresentado junto com a descrição dos objetos da MIB.

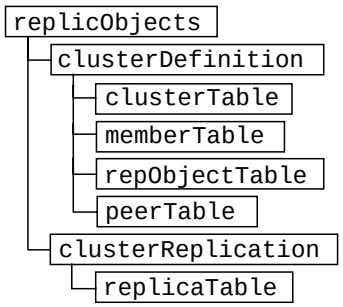

Figura 5: Replic-MIB: estrutura básica.

O grupo `clusterDefinition` consiste de quatro tabelas que são usadas para definir clusters de agentes: `clusterTable`, `memberTable`, `repObjectTable`, e `peerTable`. A `clusterTable` está localizada na aplicação gerente, e contém a definição de todos os clusters de agentes. Os clusters são criados automaticamente pela aplicação gerente. Cada cluster contém a definição de seus membros (`memberTable`), a especificação dos objetos de gerência a serem replicados (`repObjectTable`), e a definição dos clusters pares (`peerTable`), isto é, aqueles nos quais as réplicas são mantidas. O grupo `clusterReplication` consiste de uma única tabela chamada `replicaTable`, que é mantida por cada cluster. Esta tabela é automaticamente construída e mantém os objetos de gerência replicados de cada membro de um cluster assim como de seus clusters pares.

A descrição das principais partes do framework da MIB é apresentada abaixo. A especificação da MIB completa está em [9].

Definindo Clusters de Gerência

Uma aplicação gerente SNMP permite a definição de todos os clusters de agentes, seus membros, e os objetos de gerência a serem replicados. Como descrito abaixo, `clusterTable` é a tabela usada para definir e manter as informações necessárias para criar cada cluster de agentes.

O `clusterIndex` identifica uma entrada da tabela `clusterTable`. O objeto `clusterID` identifica um cluster. Os objetos `clusterAddressType` e `clusterAddress` definem respectivamente o tipo de endereço e o endereço da entidade SNMP que monitora um conjunto de agentes. Os objetos `clusterMemberType` e `clusterMember` definem respectivamente o tipo de endereço e o endereço de cada membro de um cluster. O objeto `clusterOID` é o OID (identificador do objeto em ASN.1) de um objeto de gerência. O objeto `clusterInstanceIndex` define um índice de um objeto de gerência a ser replicado. O objeto `clusterRepClusterID` (RepCID) mantém o identificador dos clusters pares. O objeto `clusterName` mantém informações sobre o gerente humano responsável pelo cluster. O objeto `clusterDescr` descreve a finalidade do cluster, e o objeto `clusterStatus` indica o status do cluster.

Index	ID	Address Type	Address	Member Type	Member	OID	Instance Index	rep CID	Name	Descr	Status
1	Ci	ipv4(1)	10.0.0.1	ipv4(1)	Mb1	ifInOctets	1	Cj	John	example	active(1)
2	Ci	ipv4(1)	10.0.0.1	ipv4(1)	Mb1	ifInOctets	2	Cj	John	example	active(1)
3	Ci	ipv4(1)	10.0.0.1	ipv4(1)	Mb2	ifInOctets	1	Cj	John	example	active(1)
4	Ci	ipv4(1)	10.0.0.1	ipv4(1)	Mb2	ifInOctets	2	Cj	John	example	active(1)
5	Cj	ipv4(1)	10.0.0.2	ipv4(1)	Mb3	ifInOctets	1	Ci	John	example	active(1)
6	Cj	ipv4(1)	10.0.0.2	ipv4(1)	Mb4	ifInOctets	1	Ci	John	example	active(1)

Tabela 1: Exemplo de uma tabela cluster definida no nível da aplicação gerente.

A tabela 1 ilustra uma `repClusterTable` exemplo. Esta tabela define um sistema que consiste de dois clusters: `Ci` and `Cj`, que têm endereços IP, `10.0.0.1` e `10.0.0.2`, respectivamente. O tipo de endereço em cada cluster é `ipv4(1)`. O cluster `Ci` contém dois membros: `Mb1` e `Mb2`, e monitora instâncias dos índices `1` e `2` do OID (identificador do objeto em ASN.1) `ifInOctets`. Esses objetos monitorados são mantidos replicados em `Ci` e também em `Cj` que é o cluster par. O cluster `Cj` contém dois membros, `Mb3` e `Mb4`, e monitora somente a instância do índice `1` do OID `ifInOctets`, replicando os objetos em `Ci` que é seu cluster par. O objeto `ifInOctets` conta o número de octetos que chegaram através de uma interface de rede [6].

Especificando Membros do Cluster

Um cluster é um agente que tem a tarefa de monitorar um subconjunto de objetos de gerência de uma coleção de agentes chamados membros do cluster, e também replicar estes objetos nos clusters pares. A tabela `memberTable` é utilizada para definir e manter informações dos membros dos cluster.

Considere um cluster que consiste de um conjunto de agentes chamados membros. A cada membro é associado um identificador chamado `cmIndex`. Este identificador é um inteiro, sendo único para cada membro. Os objetos `cmAddressType` e `cmAddress` definem respectivamente o tipo de endereço e o endereço do membro do cluster. O mecanismo de segurança utilizado para acessar um agente é definido pelo objeto `cmSecurity` enquanto o objeto `cmStatus` indica se uma entrada da tabela de membros do cluster está completa. A tabela 2 ilustra a `repMemberTable` para este exemplo.

Index	AddressType	Address	Security	Status
1	ipv4(1)	10.0.0.3 (Mb1)	community	active(1)
2	ipv4(1)	10.0.0.4 (Mb2)	community	active(1)

Tabela 2: Exemplo de uma tabela membro no cluster Ci.

Continuando o exemplo acima, considere o cluster `Ci`. Este cluster define os membros `Mb1` e `Mb2` cujos endereços IP são `10.0.0.3` e `10.0.0.4`, respectivamente. O tipo de endereço de cada membro é `ipv4(1)`. Além disso, cada membro tem `community` definido como o mecanismo de segurança.

Especificando Objetos Replicados

Além de especificar os membros de um cluster, é necessário também definir quais objetos são replicados pelo cluster. A tabela `repObjectTable` é utilizada para determinar quais objetos são replicados num dado cluster.

Cada entrada da tabela `repObjectTable` contém um OID (identificador do objeto em ASN.1) de um objeto de gerência e um índice deste objeto, respectivamente `roOID` e `roInstanceIndex`. O objeto `roInterval` define a freqüência de tempo no qual um objeto de gerência é monitorado e replicado. O objeto `roState` determina se um objeto de gerência é replicado no cluster, e o objeto `roStatus` indica se as informações de cada entrada está completa. A tabela 3 ilustra a `repObjectTable` utilizada como exemplo.

Index	OID	InstanceIndex	Interval	State	Status
1	ifInOctets	1	2	active(1)	active(1)
2	ifInOctets	2	2	active(1)	active(1)

Tabela 3: Exemplo de uma tabela de objetos a serem replicados no cluster Ci.

Continuando o exemplo acima, considere o cluster `Ci`. Este cluster especifica dois objetos de gerência `ifInOctets.1` e `ifInOctets.2` para serem monitorados e replicados a cada intervalo de tempo de 2 segundos. O estado dos objetos de gerência é `active(1)`, isto é, ambos os objetos estão sendo monitorados e replicados pelo cluster `Ci`.

Especificando Clusters Pares

Além de monitorar os objetos dos membros dos cluster, cada cluster tem a tarefa de manter réplicas dos objetos de gerência monitorados pelos seus clusters pares. A tabela `peerTable` é usada para definir os clusters pares nos quais são mantidas réplicas dos objetos de gerência.

Para identificar um determinado cluster par, é associado um identificador chamado `pcIndex`. Cada entrada também mantém os objetos `pcAddressType` e `pcAddress` que respectivamente mantém o tipo de endereço e o endereço do cluster par. O objeto `pcROTIndex` indica o índice de um objeto na tabela de objetos replicados, isto é, quais objetos são replicados em dado cluster par. O objeto `pcStatus` indica se uma entrada da tabela de clusters pares está completa.

Index	AddressType	Address	ROTIndex	Status
1	ipv4(1)	10.0.0.2 (Cj)	1	active(1)
2	ipv4(1)	10.0.0.2 (Cj)	2	active(1)

Tabela 4: Exemplo de uma tabela clusters pares no cluster Ci.

Continuando o exemplo acima, considere o cluster `Ci`. Como mostrado na tabela 4, este cluster define `Cj` como seu cluster par. O tipo de endereço de `Cj` é `ipv4(1)`. Os

índices dos objetos replicados definidos em `repObjectTable` e replicados em `Cj` são `1` e `2`, respectivamente.

Mantendo e Acessando Objetos Replicados

A tabela `replicaTable` mantém as réplicas de objetos. Esta tabela é automaticamente construída quando o sistema é inicializado, e mantém instâncias de todos os objetos replicados de cada membro do cluster e dos objetos replicados dos clusters pares.

Cada entrada contém as seguintes informações. O objeto `repIndex` identifica uma entrada da tabela `replicaTable`. Os objetos `repPeerType` e `repPeer` definem respectivamente o tipo de endereço e o endereço do cluster origem que replicou objetos de gerência neste cluster. Os objetos `repMemberType` e `repMember` definem respectivamente o tipo de endereço e o endereço do membro do cluster em `repPeer` cujos objetos de gerência são replicados neste cluster. Os objetos `repOID` e `repInstanceIndex` definem respectivamente o OID (identificador do objeto em ASN.1) e o índice de um objeto de gerência replicado. Os objetos `repValue` e `repValueType` definem respectivamente o valor em um índice de um objeto replicado e o seu tipo de dado. O objeto `repTimeStamp` contém o tempo transcorrido desde a última atualização do valor de um objeto replicado, e o objeto `roStatus` indica se a informação de uma entrada da tabela está completa.

Após o gerente ativar os clusters, cada cluster começa a monitorar os objetos de gerência especificados em todos os seus agentes membros. Para cada objeto de gerência é especificado um intervalo de tempo que determina a freqüência na qual o cluster monitora o objeto. A cada intervalo de tempo de cada objeto o cluster faz polling de todos os membros do cluster e verifica se os valores dos objetos foram atualizados desde o último intervalo. Em seguida, atualiza a tabela `replicaTable`, e replica os objetos em todos os cluster definidos como pares, onde são mantidos nas respectivas tabelas `replicaTable`. Assim, cada `replicaTable` mantém réplicas dos objetos monitorados pelo cluster local e também réplicas dos clusters pares. O intervalo de tempo deve ser cuidadosamente escolhido para garantir a consistência dos objetos nos clusters e clusters pares, além de evitar um alto impacto no desempenho da rede.

Continuando o exemplo acima, a `replicaTable` é mostrada na tabela 5. Esta tabela é mantida por todos clusters. A tabela `replicaTable` é indexada pela tupla `(repIndex, repPeer,repMember)`.

Index	Peer Type	Peer	Member Type	Member	OID	Instance Index	Value	Value Type	TimeStamp	Status
1	ipv4(1)	Ci	ipv4(1)	Mb1	ifInOctets	1	124	counter32(4)	0:00:35.24	active(1)
2	ipv4(1)	Ci	ipv4(1)	Mb1	ifInOctets	2	145	counter32(4)	0:00:36.83	active(1)
3	ipv4(1)	Ci	ipv4(1)	Mb2	ifInOctets	1	120	counter32(4)	0:00:37.89	active(1)
4	ipv4(1)	Ci	ipv4(1)	Mb2	ifInOctets	2	123	counter32(4)	0:00:38.89	active(1)
5	ipv4(1)	Cj	ipv4(1)	Mb3	ifInOctets	1	200	counter32(4)	0:00:32.17	active(1)
6	ipv4(1)	Cj	ipv4(1)	Mb4	ifInOctets	1	300	counter32(4)	0:00:33.77	active(1)

Tabela 5: Exemplo de uma tabela replica no cluster Ci.

Ainda continuando o exemplo acima, considere agora que o agente `Mb1` esteja falho. O gerente então consulta um gerente do cluster (`GC`) para obter os objetos do agente falho. Os seguintes comandos `snmpget` podem ser empregados para acessar instâncias replicadas dos objetos `ifInOctets` do membro `Mb1` monitorado pelo cluster `Ci`.

```
snmpget agent=<GC> object=<clusterReplication>.<Ci>.<Mb1>.<ifInOctets>.<1>
snmpget agent=<GC> object=<clusterReplication>.<Ci>.<Mb1>.<ifInOctets>.<2>
```

Esses comandos são recebidos pelo gerente do cluster `GC`, que pode ser tanto `Ci` como `Cj`, visto que os objetos são replicados em ambos os clusters. Independente do gerente do cluster (`GC`) consultado, a resposta recebida é a exibida abaixo.

```
response: object=clusterReplication.Ci.Mb1.ifInOctets.1 = Counter32: 8282047
response: object=clusterReplication.Ci.Mb1.ifInOctets.2 = Counter32: 117834230
```

4 Uma Ferramenta Prática

Esta seção descreve uma ferramenta prática, baseada no framework apresentado na seção anterior, que permite a observação dos objetos replicados dos agentes falhos ou inacessíveis. A ferramenta é baseada no pacote de domínio público NET-SNMP [13] e na ferramenta para comunicação de grupos Ensemble [12]. A ferramenta desenvolvida permite a criação de clusters de agentes SNMP que suportam replicação semi-ativa dos objetos selecionados. Os clusters são dinâmicos, no sentido que eles podem ser criados e destruídos a qualquer momento. Descrevemos como a ferramenta foi utilizada para determinar que um dado agente falho tinha sofrido ataques TCP SYN-Flooding.

O sistema baseia-se em grupos cooperantes de agentes SNMP definidos com a ferramenta para comunicação de grupos Ensemble. Esta ferramenta é utilizada para adicionar a capacidade de multicast confiável aos clusters de agentes SNMP. Ensemble oferece comunicação aos grupos de processos, proporcionando canais ordernados entre todos os clusters pares. Um serviço para gerenciar múltiplos clusters chamado `mclusted`, implementado em Ensemble, disponibiliza um protocolo de comunicação de grupos. O serviço `mclusted`

permite aos clusters detectarem as falhas, garantindo que os clusters pares também detectem as mesmas falhas. Ensemble suporta multicast IP, e também possui a aplicação `gossip` que simula a difusão seletiva em redes que não têm multicast IP.

Considerando o framework SNMP apresentado acima, cada cluster deve incluir as tabelas `memberTable`, `repObjectTable`, `peerTable`, e `replicaTable`. No momento que um cluster é ativado, os objetos são automaticamente monitorados e replicados para todos os clusters pares. Replicação semi-ativa é empregada entre os clusters pares, sendo que cada gerente do cluster é o líder dos objetos replicados de seus respectivos membros. Uma thread para analisar cada mensagem e atualizar os objetos replicados foi adicionada. Uma consulta a um cluster é processada da mesma forma que uma consulta feita a um agente comum. Além disso, cada objeto replicado de qualquer cluster possui um timestamp que informa o tempo transcorrido da última atualização da réplica do objeto num cluster. A ferramenta permite a qualquer cluster par ser consultado sobre os objetos replicados de qualquer agente que pertença ao cluster.

Detecção de Ataques TCP SYN-Flooding

Um experimento foi realizado numa rede local composta de 26 máquinas baseada em diferentes tipos de processadores como Pentium Intel e K6 AMD, executando Linux e conectados através de uma Ethernet LAN de 100Mbps. O cluster SNMP foi instalado em duas máquinas, *genio* e *ork*, para monitorar um subconjunto de objetos de gerência TCP. Cada cluster monitorava um conjunto de 12 máquinas, e replicava os objetos TCP no cluster par. Uma máquina monitorada pelo cluster *genio* foi submetida a um ataque TCP SYN-Flooding [28]; após alguns segundos essa máquina apresentou o estado falho.

O cluster na máquina *genio* foi requisitado a fim de examinar os objetos TCP replicados do agente falho. Os objetos replicados incluiam `tcpPassiveOpens`, que conta o número de vezes que as conexões TCP fizeram uma transição direta do estado LISTEN para o estado SYN-RCVD [29]. Outro objeto replicado era o `tcpCurrEstab`, número de conexões TCP cujo estado atual é ESTABLISHED ou CLOSE-WAIT. Os valores dos objetos da MIB não deixaram dúvidas quanto a causa do ataque, logs anteriores mostravam que `tcpPassiveOpens` normalmente tinha o valor igual a zero, enquanto que após a falha o valor era igual a 85. Já o objeto `tcpCurrEstab` permaneceu com o valor igual a 12. Os valores obtidos do cluster *genio* também foram obtidos através da réplica no cluster *ork*. Se a ferramenta não estivesse disponível, seria impossível examinar a MIB da máquina submetida ao ataque TCP SYN-Flooding.

5 O Impacto de Agrupamento no Desempenho da Rede

Cada vez que os objetos de um cluster são replicados, as mensagens são enviadas aos clusters pares de maneira que eles possam atualizar suas réplicas; assim a replicação tem um impacto no desempenho da rede em virtude da quantidade de recursos necessários para transmitir mensagens de replicação. Além disso, se as mensagens de replicação não são enviadas numa freqüência suficiente, as réplicas podem tornar-se inconsistentes, isto é, os valores dos objetos replicados podem ser diferentes dos valores mantidos pelo cluster origem. Apresentamos uma avaliação probabilística de quanto inconsistentes as réplicas podem se tornar considerando diferentes intervalos de atualização versus o impacto no desempenho da rede.

O *intervalo de atualização de réplica* é o intervalo de tempo no qual um cluster faz polling em seus membros e envia, caso necessário, mensagens para atualização das réplicas em seus clusters pares. O gráfico na figura 6-A mostra a largura de banda exigida considerando-se que as mensagens de replicação são enviadas em diferentes intervalos de atualização de réplica, variando de 500 milisegundos até 2 segundos cada. O número de mensagens enviadas varia de 1 a 50, e o tamanho das mensagens é de 100 bytes, ou seja, cada mensagem carrega informações de um único objeto.

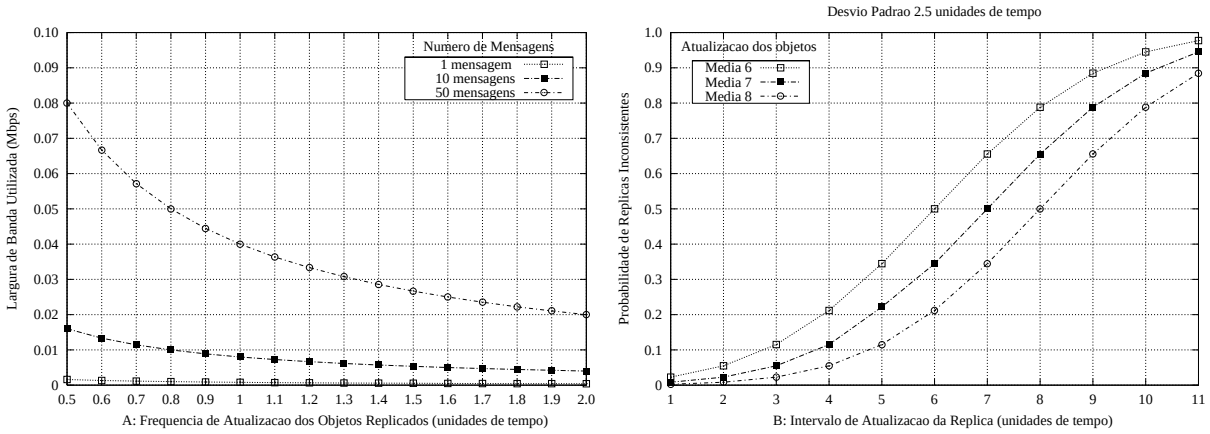

Figura 6: O impacto no desempenho da rede e na consistência das réplicas.

Quando apenas uma mensagem é enviada em um intervalo, a quantidade de bytes transferidos por segundo é de menos de 2Kbps. Quando o número aumenta para 10 mensagens por intervalo, até 16Kbps são necessários. Para um grupo de 50 clusters onde cada um envia uma mensagem por intervalo, a largura de banda exigida varia de cerca de 20Kbps, quando o intervalo é de 2 segundos, até 80Kbps quando as 50 mensagens são enviadas a cada 500 milisegundos. À medida que o intervalo de tempo aumenta, a largura de banda exigida permanece abaixo de 30Kbps. Por outro lado, quando o intervalo de tempo diminui, a largura de banda exigida aumenta. Entretanto, para clusters pequenos

a largura de banda exigida é viável considerando as atuais tecnologias de redes.

Considere o gráfico da figura 6-B. Este gráfico mostra o impacto da variação do intervalo de atualização de réplicas na consistência das mesmas. Consideramos a distribuição gaussiana para os intervalos de atualização do líder. A média varia de 6 a 8 unidades de tempo. O desvio padrão é constante igual a 2.5 unidades de tempo. O gráfico mostra a probabilidade de inconsistência das réplicas à medida que o intervalo de atualização das réplicas varia.

Quando o intervalo de atualização das réplicas é de 1 unidade de tempo, a quantidade de réplicas que se tornarão inconsistentes é menos de 5%. Quando o intervalo de atualização das réplicas aumenta para 4 unidades de tempo, a inconsistência varia de acordo com a média, de 5% para 22%. Se o intervalo de atualização das réplicas é igual ao intervalo de atualização médio do líder, então 50% das réplicas estarão inconsistentes. Finalmente, se o intervalo de atualização das réplicas é de 11 unidades de tempo, então de 88% a 97% das réplicas estarão inconsistentes.

Em conclusão, para se determinar o intervalo de atualização das réplicas, o gerente humano deve estudar o comportamento dos objetos replicados. Os objetos SNMP alteram seus valores em intervalos de tempos completamente diferentes. Assim, uma análise estatística de cada caso é necessário para se determinar o melhor intervalo de atualização da réplica de um objeto. O gerente deve obter amostras suficientes dos intervalos de atualização, computando a média e o desvio padrão. De acordo com esses parâmetros e a avaliação do impacto do desempenho, o intervalo de atualização das réplicas é definido.

6 Conclusão

Sistemas de gerência de redes são mais necessários justamente em situações de emergências, quando a rede está parcialmente falha. Aplicações de gerência de falhas devem ser capazes de funcionar corretamente na presença de falhas na rede. Neste trabalho, apresentamos uma nova arquitetura de agrupamento para agentes SNMP que suporta replicação semi-ativa dos objetos de gerência. Um cluster implementa objetos tolerantes a falhas: objetos de gerência replicados de um agente falho monitorado por um cluster podem ser acessados através do próprio cluster ou através de um cluster par. Além disso, o cluster funciona como uma cache dos objetos de gerência, reduzindo o impacto da monitoração no desempenho da rede.

Uma ferramenta prática foi apresentada, na qual os agentes SNMP se comunicam usando a ferramenta Ensemble para comunicação de grupos. Mostramos um exemplo no qual esta ferramenta foi utilizada para examinar a MIB de um elemento de rede falho, permitindo ao gerente descobrir que o agente tinha falhado devido a um ataque TCP SYN-Flooding. A avaliação do impacto do agrupamento de agentes SNMP no desempenho da rede mostra que a utilização de agrupamento é viável para os sistemas existentes.

A arquitetura de agrupamento está atualmente em processo de tornar-se um padrão do IETF [9]. Trabalhos futuros incluem o desenvolvimento de uma abordagem flexível na qual os elementos da rede possam automaticamente se integrar ou se desligar dos clusters; ao invés da configuração manual da ferramenta atual. Também está sendo desenvolvida uma interface em Java que permite ao gerente monitorar tanto agentes sem-falhas como agentes falhos da rede.

Referências

[1] A. Leinwand and K. F. Conroy, *Network Management: A Practical Perspective,* Addison-Wesley, 1996.

[2] E.P. Duarte Jr., and T. Nanya, "A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm," *IEEE Transactions on Computers*, Vol.47, pp.34-45, No.1, Jan 1998.

[3] E.P. Duarte Jr., F. Mansfield, T Nanya, and S Noguchi, "Non-Broadcast Network Fault-Monitoring Based on System-Level Diagnosis," *Proceedings of the 5th IEEE/IFIP International Symposium on Integrated Network Management (IM'97)*, San Diego CA, 1997.

[4] S. Kätker, and M. Paterok, "Fault Isolation and Event Correlation for Integrated Network Management," *Proceedings of the 5th IEEE/IFIP International Symposium on Integrated Network Management (IM'97)*, San Diego CA, 1997.

[5] C. S. Hood, and C. Ji, "Proactive Network Fault Detection," *Proc. INFOCOM 97*, 1997.

[6] W. Stallings, *Snmp, Snmpv2, Snmpv3 and Rmon 1 and 2,* Addison-Wesley, Reading, MA, 1999.

[7] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme and G. Alonso, "Understanding Replication in Databases and Distributed Systems," Technical Report SSC/1999/035, École Polytechnique Fédérale de Lausa nne, Switzerland, September 1999.

[8] D. Harrington, R. Presuhn, and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks," *RFC 2271*, January 1998.

[9] Aldri L. Santos, Elias P. Duarte Jr., and Glenn Mansfield, "A Clustering Architecture for Replicating Managed Objects," *Internet Draft*, IETF, November 2001. Available at: http://www.rfc-editor.org/internet-drafts/draft-aldri-disman-replication-mib-00.txt

[10] K. Birman, *Building Reliable and Secure Network Applications,* Prentice-Hall, 1996.

[11] R. V. Renesse, K. P. Birman and S. Maffeis, "Horus: A Flexible Group Communication System," *Communications of the ACM*, Vol. 39, No. 4, pp. 76-83, April 1996.

[12] M. G. Hayden, *The Ensemble System,* PhD Thesis, Cornell University, Ithaca, Jan. 1998.

[13] *The NET-SNMP Home Page,* http://net-snmp.sourceforge.net.

[14] E.P. Duarte Jr. and Aldri L. dos Santos, "Semi-Active Replication of SNMP Objects in Agent Groups Applied for Fault Management," *Proceedings of the 7th IFIP/IEEE International Symposium on Integrated Network Management (IM'01)*, Seattle, May 2001.

[15] E.P. Duarte Jr. and Aldri L. dos Santos, "Network Fault Management Based on SNMP Agent Groups," *Proceedings of the IEEE 21st International Conference on Distributed Computing Systems Workshops (ICDCS'2001)*, Workshop on Applied Reliable Group Communications, pp. 51-56, Mesa, Arizona, April 2001.

[16] J. Schönwälder, "Using Multicast-SNMP to Coordinate Distributed Management Agents," *IEEE Workshop on Systems Management*, June 1996.

[17] K.-H. Lee, "A Group Communication Protocol for Distributed Network Management Systems," *In Proc. ICCC 95*, pp. 363-368, 1995.

[18] D. Breitgand, *Group Communication as an Infrastructure for Distributed Systems Management,* Master Dissertation, Hebrew University of Jerusalem, June 1997.

[19] *Distributed Management (DisMan) Charter* - http://www.ietf.org/html.charters/disman-charter.html

[20] D. Levi and J. Schönwälder, "Definitions of Managed Objects for the Delegation of Management Scripts," *RFC 3165*, August 2001.

[21] R. Kavasseri and B. Stewart, "Event MIB," *RFC 2981*, October 2000.

[22] D. Levi and J. Schönwälder, "Definitions of Managed Objects for Scheduling Management Operations," *RFC 3231*, January 2002.

[23] S. Chisholm and D. Romascanu, "Alarm MIB," *Internet Draft*, IETF, December 2001.

[24] W. Chen, N. Jain, and S. Singh, "ANMP: Ad Hoc Network Management Protocol," *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, August 1999.

[25] R. Guerraoui and A. Schiper, "Fault-Tolerance by Replication in Distributed Systems," *International Conference on Reliable Software Technologies*, Springer Verlag (LNCS), 1996.

[26] R. Guerraoui and A. Schiper, "Software-based Replication for Fault Tolerance," *IEEE Computer*, Vol. 30, No. 4, pp. 68-74, April 1997.

[27] F. B. Schneider, "Implementing Fault-Tolerant Services Using The State Machine Approach: A Tutorial," *ACM Computing Surveys*, Vol. 22, No. 4, pp. 299-319, Dec. 1990.

[28] R. Farrow, "TCP SYN Flooding Attacks and Remedies," *Network Computing Unix World*, http://www.networkcomputing.com/unixworld/security/004/004.txt.html.

[29] K. McCloghrie, "SNMPv2 Management Information Base for the Transmission Control Protocol using SMIv2," *RFC 2012*, IETF, November 1996.