Um Framework de Serviços para Gerenciamento de Medições por Fluxo de Tráfego

Leobino N. Sampaio, José A. Suruagy Monteiro

[1]NUPERC – Universidade Salvador
R. Ponciano de Oliveira, 126 – 41950-275 - Salvador, BA

leobino, suruagy@{unifacs.br}

Abstract. *It is a great challenge to analyze the data obtained from the great variety of today's traffic flow measurement tools. This task is even more complicated when we need to extract useful information for the performance analysis of networks belonging to different domains due to system and equipment heterogeneity. This paper proposes a Web Services based framework in order to facilitate the access of this measurement's data by management applications, hiding from them the implementation details of proprietary infrastructures, and making possible to build other management scenarios. It is also presented, in this work, the SFM3 model proposed by the authors that was the reference for the development of the service's framework.*
Key Words: *Management, Measurement, Flow Traffic, Web Services, Performance analysis.*

Resumo. *Analisar os dados advindos das medições por fluxo de tráfego realizadas através da variada quantidade de ferramentas existentes hoje, é um grande desafio. A tarefa torna-se ainda mais complicada quando se pretende extrair informações úteis para a avaliação de desempenho das redes que pertencem a domínios diferentes, em função da heterogeneidade de sistemas e equipamentos. Neste sentido, o presente artigo propõe um framework de Serviços Web como elemento facilitador do acesso aos dados destas medições por parte das aplicações de gerenciamento, escondendo delas os detalhes de implementação das infra-estruturas proprietárias e viabilizando cenários de gerenciamento de maior amplitude. Neste trabalho também será apresentado o modelo SFM3 proposto pelos autores que serviu de referência para o desenvolvimento do framework de serviços.*
Palavras Chaves: *Gerência, Medições, Fluxos de tráfego, Serviços Web, Avaliação de desempenho.*

1. Introdução

É notória a variedade de ferramentas e técnicas existentes atualmente para medições em redes IP com a finalidade de verificar a qualidade de serviço que está sendo ofertada aos seus usuários. Esta variedade existe em função da complexidade tecnológica que envolve os domínios administrativos, que de uma forma geral, têm suas próprias soluções de hardware e software. Outra razão para que este fato ocorra se dá pela necessidade do gerenciamento das redes de avaliar parâmetros de qualidade de serviço mais específicos que dificilmente seria feito por uma única ferramenta ou técnica.

O uso de ferramentas de medições diferentes tem suas vantagens uma vez que é possível atingir níveis mais altos de confiança nos resultados das medidas. Por outro lado,

a manipulação destes resultados por parte das aplicações de gerenciamento torna-se um desafio se as mesmas não seguirem um padrão de acesso e apresentação.

Além da uniformidade de acesso aos dados sobre a rede, é de interesse do gerenciamento poder acessar as informações sobre as medidas que pertencem a diferentes domínios bem como correlacioná-las. Nas medições por fluxo de tráfego, por exemplo, esta necessidade se torna ainda mais evidente, já que na maioria das vezes, estas medições ultrapassam as fronteiras dos seus domínios administrativos.

Em termos práticos, existem diversas soluções para a implantação de infra-estruturas de medições por fluxo de tráfego, sendo que genericamente, todas elas se baseiam no modelo RTFM [Brownlee et al., 1999] implementada pelo NetraMet [Brownlee, 1997], NetFlow [Brownlee et al., 1999] e o sFlow [Phaal et al., 2001]. Uma das características mais relevantes na arquitetura dessas soluções é a existência de um elemento coletor[1] que implementa seus próprios métodos de comunicação com os dispositivos de medição. Por este motivo, no intuito de evitar uma diversidade muito grande de soluções para esta troca de dados, o IETF vem trabalhando na padronização dessa comunicação através do grupo de trabalho IPFIX (*IP Flow Information Export*) [IPFIX, 2003].

De todo modo, uma questão crítica que ainda persiste é a forma de acesso aos dados presentes nestes coletores e dispositivos por parte das aplicações de gerência. Atualmente, estas aplicações necessitam implementar soluções próprias que podem variar muito de acordo com a ferramenta de medição em questão, resultando em soluções pouco adaptáveis a novos ambientes. É de consenso geral que o dinamismo nas redes exige ferramentas mais flexíveis, com maior capacidade de adaptação e sem exigir grandes mudanças de infra-estrutura.

É diante desta percepção que surgem iniciativas em torno da padronização no acesso aos dados das medições por fluxo de tráfego. A nível mundial, estas abordagens vêm sendo adotadas por iniciativas tais como a E2E piPEs *(End-to-End Performance Initiative Performance Environment System)* da Internet2 [Internet2, 2003] e INTERMON *(Advanced architecture for INTER-domain quality of service MONitoring, modelling and visualisation)* [INTERMON, 2003]. Nestes grupos existem propostas de uniformização no acesso aos dados resultante das medidas obtidas através de diversas ferramentas. Este esforço faz parte inclusive do escopo dos trabalhos do grupo de Serviços e Desempenho do *Global Grid Forum* [GGF, 2003], que tem como uma de suas metas implantar um ambiente integrado e colaborativo entre os variados sistemas, permitindo um acesso uniforme às diversas fontes de informação. No contexto da RNP[2], através das atividades do GT-QoS2 [Monteiro et al., 2003], o Netflow tem sido utilizado como principal solução para identificar as características de tráfego do *backbone* nacional. Durante a sua implantação, foi possível constatar a necessidade de mecanismos para a consolidação dos dados sobre os fluxos oriundos dos diversos PoPs (pontos de presença).

Em termos de integração entre aplicações, a literatura atual destaca a utilização de Serviços Web (do inglês *Web Services*) [W3C, 2003b] como opção tecnológica de implementação. Isto porque trata-se de uma tecnologia baseada em padrões, independente de plataforma, sistema operacional e linguagens de programação, integrando as estruturas proprietárias, sem exigir grandes mudanças.

Diante do cenário descrito, este artigo apresenta um *framework* de serviços que encapsula os detalhes específicos das estruturas proprietárias presentes nas medições por

[1]Um coletor é formado, normalmente, por um host que, ligado a um equipamento de rede, coleta dados dos tráfegos para serem posteriormente analisados.

[2]Rede Nacional de Ensino e Pesquisa - http://www.rnp.br.

fluxo de tráfego. Cabe então, a este *framework*, a tarefa de disponilizar as informações sobre os fluxos já processadas e no nível que as aplicações de gerenciamento necessitam. O *framework* se baseia nos princípios do modelo de gerenciamento de medições por fluxo de tráfego orientado a serviços SFM3 (do inglês, ***S**ervice-based **F**low traffic **M**easurement **M**anagement **M**odel*), funcionando na primeira camada do modelo e fornencendo serviços à camada imediatamente superior.

O restante deste artigo está assim organizado. Na seção 2 são apresentados os trabalhos relacionados e a fundamentação teórica necessária. Na seção 3 é apresentado o modelo de gerenciamento SFM3 e alguns de seus componentes. A seção 4 apresenta o *framework* de serviços. A seção 5 é dedicada ao protótipo e a seção 6 apresenta os experimentos realizados com o mesmo. Por fim, na seção 7 tem-se a conclusão e trabalhos futuros.

2. Trabalhos Relacionados

Atualmente, manipular os dados das medições dos fluxos que passam nas redes IP é um grande desafio, dada à diversidade de ferramentas e soluções disponíveis. Mesmo com padronização da forma em que estes fluxos são medidos, ainda resta um problema referente à incompatibilidade de comunicação entre sistemas pertencentes a domínios diferentes. Estes fatores resultam nos trabalhos do grupo IPFIX e à abordagem da arquitetura SOA (*Service-Oriented Architecture*) implementada através de Serviços Web.

2.1. IPFIX - *IP Flow Information Export*

No intuito de compatibilizar o grande número de soluções presentes em torno das medições por fluxos de tráfego, o grupo IPFIX foi criado especialmente para definir uma padronização para a coleta dos fluxos pelos dispositivos de medição e exportados para os coletores. Para isso foram estabelecidas algumas atividades com o objetivo de definir, dentre outras coisas, uma arquitetura de funcionamento [Sadasivan and Brownlee, 2003], um protocolo de transporte entre medidores e coletores [Claise et al., 2003] e um modelo de dados [Calato et al., 2003].

De forma resumida, no documento que descreve a arquitetura são identificados os elementos envolvidos na medição, exportação e coleta bem como descritas as suas funcionalidades específicas. Para o protocolo de transporte foram avaliados: o Netflow V9 [Claise, 2003], o LFAP [Calato, 2002], o Streaming IPDR [Meyer, 2002], o CRANE [Zhang et al., 2003] e o Diameter [Zander, 2002]. O resultado da avaliação destes protocolos pode ser encontrado em [Leinen, 2003]. No modelo de dados são relacionados os tipos primitivos utilizados na representação e processamento das informações sobre os fluxos.

Em resumo, o escopo dos trabalhos desenvolvidos pelo IPFIX se limita na identificação dos fluxos[3] e a transferência de seus dados para um coletor, conforme a Figura 1.

O protocolo IPFIX, implementado sobre TCP, é responsável por transferir informações de um equipamento de rede (ex., roteador ou *switch*) para um coletor. Os dados transferidos descrevem os fluxos de interesse, bem como informações complementares sobre os mesmos (ex., número de bytes, portas de transporte, endereços de origem e destino, etc.). Como abordado anteriormente, os trabalhos do IPFIX se reservam à coleta dos dados e sua transferência para os coletores. Uma vez que a coleta e exportação

[3]No contexto do IPFIX, fluxo pode ser considerado como um conjunto de pacotes IP que possuem atributos em comum (ex: IP Origem, IP Destino, porta fonte, porta destino, protocolo, etc.).

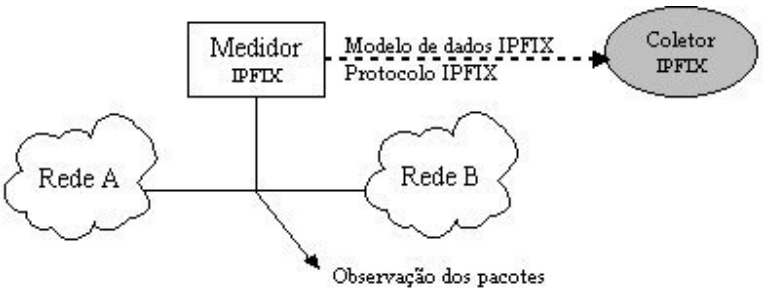

Figura 1: Escopo dos Trabalhos do Grupo IPFIX.

das informações dos fluxos estejam sendo realizadas seguindo esta padronização, outras contribuições que estão fora do escopo das atividades do grupo, ainda se fazem necessárias no que diz respeito ao armazenamento, recuperação e disponibilização dos dados.

2.2. SOA (Arquitetura Orientada a Serviços)

A Arquitetura Orientada a Serviços (*SOA — Service-Oriented Architecture*) [W3C, 2003b] [Papazoglou, 2003] trata de um outro tipo de abordagem no desenvolvimento de software no qual as aplicações passam a ser construídas e reorganizadas como provedoras de serviços (ou operações) específicos e bem definidos. Basicamente, fazem parte desta arquitetura três elementos: um cliente de serviços, um provedor de serviços e um repositório de informações sobre os serviços e as suas localizações.

- **Cliente.** O cliente é o elemento que faz a solicitação dos serviços, portanto, é quem inicia a comunicação e muitas vezes pode utilizar o atendimento dos serviços solicitados para a composição de outros serviços.
- **Provedor.** O provedor contém um ou mais serviços que são disponibilizados aos clientes. Para viabilizar o acesso aos serviços a partir dos clientes, o provedor divulga os seus serviços nos repositórios.
- **Agenciadores.** Os agenciadores armazenam a descrição, classificação e localização dos serviços disponibilizados pelos provedores para todos os participantes do sistema. Os agenciadores têm um papel fundamental no funcionamento da arquitetura, pois facilita a descoberta dinâmica dos serviços por parte dos clientes.

Um dos problemas ainda críticos no SOA se dá em torno da falta de um modelo que facilite o gerenciamento, a composição e utilização dos serviços prestados pelos diversos componentes. Dentro deste escopo, Papazoglou propõe uma arquitetura estendida ao SOA (*ESOA — Extended Service-Oriented Architecture*) [Papazoglou and Georgakopoulos, 2003] na qual os elementos provedores de serviços são divididos em três camadas, que são: (1) descrição e operações básicas, (2) composição de serviços e (3) gerenciamento, como podem ser vistas na Figura 2.

Em resumo, a camada de descrição e operações básicas é responsável pelo fornecimento dos serviços na sua forma mais simples. Na segunda camada, os serviços oferecidos são derivados da composição dos serviços prestados pela primeira. Por fim, na última camada estão localizados os provedores de serviços mais próximos das aplicações e usuários finais. Esta arquitetura define regras e funcionalidades destinadas à consolidação de múltiplos serviços em um único, permitindo que as aplicações usuárias tenham um enfoque direcionado para a sua funcionalidade principal, descartando os detalhes de implementação de outros serviços necessários. Deste modo, essa abordagem oferece

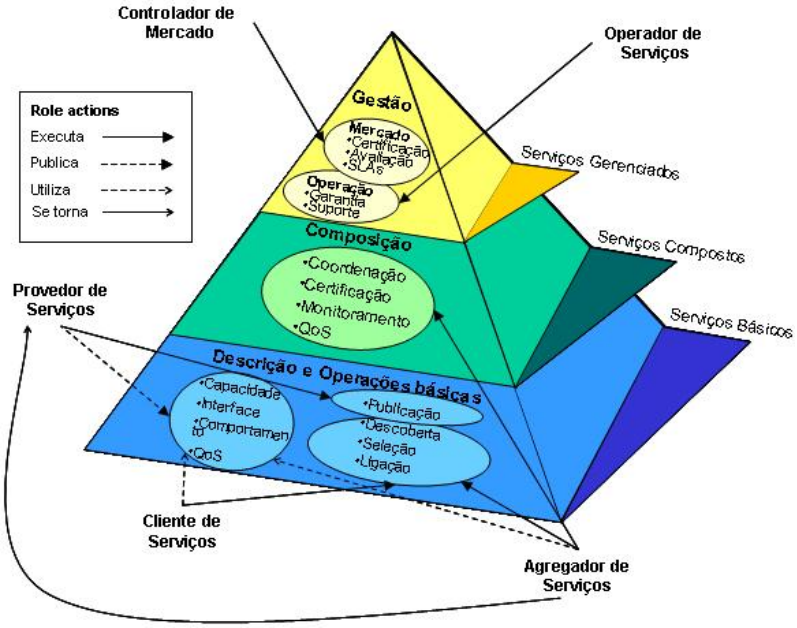

Figura 2: SOA Estendida.
Fonte: [Papazoglou and Georgakopoulos, 2003]

subsídios que facilitam, consideravelmente, o gerenciamento dos serviços, já que os mesmos estão classificados em três camadas.

Em termos práticos, a implementação dos conceitos e requisitos da Arquitetura Orientada a Serviços tem sido realizada através de Serviços Web. Isto acontece simplesmente pelo fato de ser uma solução fortemente baseada na linguagem XML (*eXtended Markup Language*) [W3C, 2000], que é de ampla aceitação e tem se consolidado como a alternativa mais adequada na integração de sistemas heterogêneos.

2.3. Serviços Web (*Web Services*)

Os Serviços Web (WS) podem ser considerados como uma forma de *middleware* que possui aplicações que se comunicam por meio de protocolos tais como o HTTP. Esta característica permite que os problemas de integração, geralmente presentes no relacionamento entre as aplicações de domínios diferentes, sejam bastante reduzidos.

Tais aplicações utilizam a linguagem XML para descrever suas funcionalidades, auxiliar na invocação de seus métodos, definir tipos de dados e publicar seus serviços. Cada uma destas atividades é realizada através de tecnologias derivadas da XML, tais como a WSDL (*Web Services Description Language*) [Chinnici et al., 2003], SOAP (*Simple Object Access Protocol*) [W3C, 2003a] e o UDDI (*Universal Description, Discovery, and Integration*) [OASIS, 2002].

A WSDL é uma linguagem utilizada para descrever os serviços disponibilizados por uma aplicação. Através desta descrição, as outras aplicações podem saber a forma de interação e os protocolos necessários para a comunicação. O SOAP é o protocolo utilizado para encapsular as trocas de mensagens entre objetos WS (invocação de métodos remotos). O uso do SOAP viabiliza que aplicações possam se comunicar através da utilização de protocolos simples como o HTTP. Por fim, o UDDI é uma infra-estrutura pública de diretórios que é utilizada para publicar os serviços e a localização das aplicações.

Com base nestes aspectos, percebe-se que o WS possui uma estreita relação com a Arquitetura Orientada a Serviços. Neste caso, o UDDI faz o papel do agenciador, a comunicação entre os elementos ocorre via protocolo SOAP e, por fim, a forma de

interação entre esses componentes é descrita em documentos WSDL. A Figura 3 apresenta uma visão resumida dos elementos envolvidos nos Serviços Web e sua relação com a SOA.

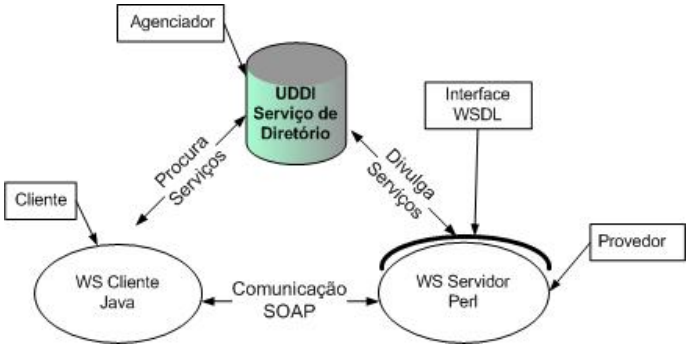

Figura 3: Arquitetura de Serviços Web.

Em termos de gerenciamento de redes, existem na literatura diversas soluções que propõem a utilização de aplicações distribuídas para a realização de medições de desempenho. Na maioria dos casos, por serem baseadas em tecnologias como Corba, são incapazes de trocar informações na Internet, o que não permite uma monitoração ao longo de domínios administrativos diferentes. Diante dos aspectos abordados nesta seção percebe-se que no contexto das medições por fluxo de tráfego, existe a necessidade de uma infra-estrutura que possibilite um acesso independente de plataforma e tecnologias. Neste sentido, a arquitetura SOA surge como uma alternativa em função da sua simplicidade e capacidade de viabilizar a interoperabilidade entre sistemas. O gerenciamento dos dados das medições por fluxos através de WS se torna uma solução bastante conveniente, na medida em que não são exigidas grandes mudanças na infra-estrutura de rede para a sua implementação. Além de contar com a grande flexibilidade provida pela XML que fornece à aplicação a possibilidade de se adaptar a qualquer tipo de ambiente. Por estes motivos, o relacionamento dos trabalhos de padronização do IPFIX, a abordagem da arquitetura SOA estendida (ESOA) e as características do WS resulta no modelo SFM3 apresentado na seção seguinte, que tem como principal meta formalizar a manipulação dos dados sobre os fluxos de rede.

3. SFM3 - *Service-based Flow traffic Measurement Management Model*

Para as aplicações de gerenciamento que estão associadas aos usuários finais, o que interessa são as informações que podem ser disponibilizadas pelos fluxos e não os seus dados na forma em que são coletados. Atualmente, estas aplicações precisam implementar soluções próprias para ter acesso a estas informações, tendo que ter a preocupação com detalhes específicos das plataformas de cada ambiente.

Para estas aplicações, seria ideal uma solução de *middleware* que facilitasse o acesso às informações sobre os fluxos, abstraindo delas os detalhes de implementação. Neste sentido, o modelo SFM3, apresentado na Figura 4, sugere a criação de componentes de software independentes que provêem as suas funcionalidades em forma de serviços. Tais serviços são descritos por meio de interfaces que são escritas em linguagens baseadas em padrões, como o XML.

O modelo SF3M trata-se de uma adaptação da arquitetura ESOA — apresentada na seção 2 — direcionada para a disponibilização dos dados sobre as medições por fluxo de tráfego. Nesta proposta, não somente os serviços são agrupados em camadas, como também os elementos responsáveis pelo uso e provimento.

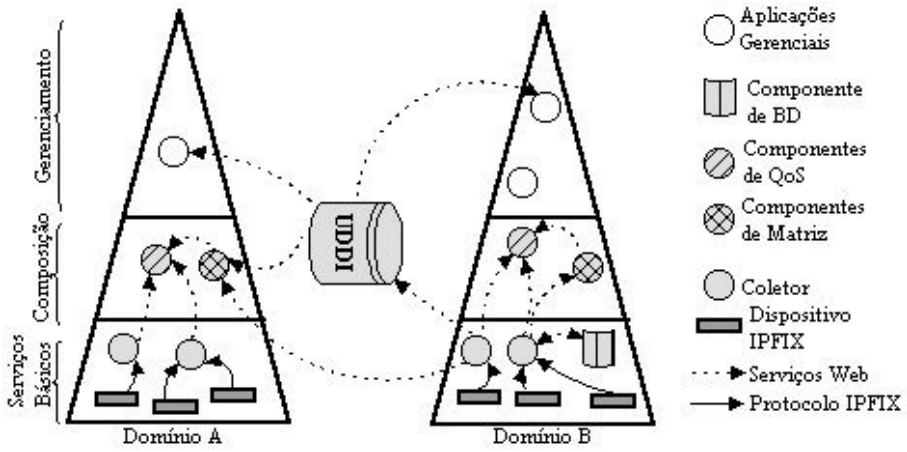

Figura 4: Modelo SFM3.

3.1. Componentes do SFM3

De uma forma geral o modelo é formado por um conjunto de componentes que estão classificados em três camadas: Serviços básicos, Composição e Gerenciamento, conforme Figura 4.

Nas bases das pirâmides estão localizados os dispositivos medidores bem como os seus coletores e componentes de BD. Neste nível, os dados sobre os fluxos estão no seu formato mais primitivo e podem estar armazenados tanto em sistema de arquivos quanto em banco de dados. Assim, as aplicações pertencentes à camada intermediária conseguem um nível de abstração um pouco maior, já que as mesmas não precisam se preocupar com as estruturas internas e as diversas formas de acesso a estes dados; os dados passam a ser acessados através da interação com os componentes da camada inferior.

O contexto no qual está inserido o *framework* de serviços descrito neste documento é a primeira camada das pirâmides do modelo e, por este motivo, serão descritas a seguir apenas as funcionalidades dos componentes coletor e banco de dados (BD).

3.1.1. Componente Coletor

A principal função do coletor nas medições por fluxo de tráfego é receber os dados dos diversos dispositivos que estão sob monitoramento. Porém, no modelo apresentado aqui, o coletor irá incorporar novas funcionalidades. Assim, é preciso ressaltar que existem basicamente dois contextos dos quais o coletor irá fazer parte: um pertence à sua relação com os dispositivos de medição (A) e o outro diz respeito à sua integração com os outros componentes (B), conforme ilustrado na Figura 5.

O coletor é o primeiro componente a receber os fluxos dos dispositivos de medição. Os dados recebidos podem ser armazenados localmente bem como enviados aos componentes de BD para serem armazenados por um período de tempo maior. Este comportamento irá variar a depender das necessidades nas medições. O armazenamento em banco de dados acontece nas situações em que interessa ao gerenciamento, dentre outras razões: saber dados históricos da rede, fazer análise de tendência, avaliar possíveis falhas de segurança e realizar cobrança.

Estas perspectivas de um comportamento adaptativo dos coletores, sugere um conjunto de novas funções, tais como:

- Configurar automaticamente regras para a coleta de fluxos;

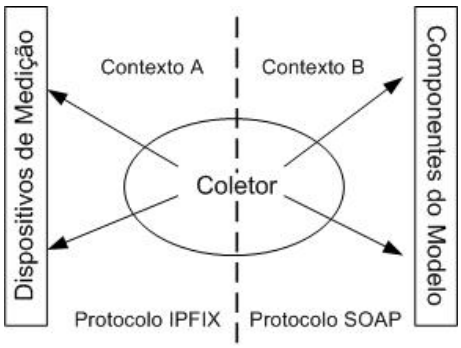

Figura 5: Contextos do coletor.

- Receber fluxos de novos dispositivos de medição em tempo real;
- Remover dados sobre os fluxos que possuem uma determinada característica;
- Anonimizar os fluxos que possuem uma determinada característica;
- Configurar pontos de observação e domínios de observação.

3.1.2. Componente de banco de dados (BD)

Como pode ser notado, o uso de banco de dados se torna importante porque muitas vezes o volume de informações sobre os fluxos coletados pode ser muito grande. Atribuir esta tarefa aos coletores pode resultar em um processamento desnecessário uma vez que nem sempre os dados coletados serão de interesse futuro. Além disso, acumular mais esta função ao coletor não parece ser muito atrativo já que o número de coletores pode ser grande e com isso seria necessário também um número equivalente de sistemas gerenciadores de banco de dados (SGBD). Deste modo, percebe-se que a utilização de um componente especializado na manipulação de um SGBD e responsável pelos dados sobre os fluxos, é bastante adequada.

A principal funcionalidade desses componentes de BD é facilitar o acesso aos dados sobre os fluxos, fornecendo serviços de armazenamento e recuperação de informações de interesse às aplicações usuárias. Ele deverá conhecer a estrutura na qual os dados estão armazenados e os detalhes específicos da arquitetura do seu SGBD, que pode variar de fabricante a depender do domínio em questão (ver Figura 4). Sendo assim, percebe-se que para os usuários dos serviços prestados por estes componentes, pouco importa qual o SGBD que está sendo utilizado e muito menos as consultas em SQL necessárias para a recuperação das informações.

Adicionalmente, o componente de BD deve possuir um conjunto de operações pré-definidas para disponibilizar informações sobre os fluxos. O número de operações vai variar de acordo com os dados disponíveis. Isto porque nem todos os fluxos identificados e transferidos para o coletor serão necessariamente enviados aos componentes de BD.

Dentre as funções do componente de BD, podem ser citadas:

- Manutenção de registros sobre os fluxos e seus tipos armazenados em banco de dados;
- Manutenção de registros sobre os dispositivos de medição, domínios de observação e pontos de observação;
- Geração de estatísticas sobre os fluxos.

Para a recuperação dos dados, este componente deve possuir os softwares (*drivers*) necessários para a comunicação com o SGBD, que pode variar de fabricante

conforme o domínio de rede em questão. A transformação destes dados em informações, acontece por meio de uma ou mais consultas SQL que recupera os registros de fluxos a partir de um determinado critério. Por fim, para a disponibilização em forma de serviços, as informações são convertidas para um formato padrão permitindo que os outros componentes possam acessa-las (ver Figura 6).

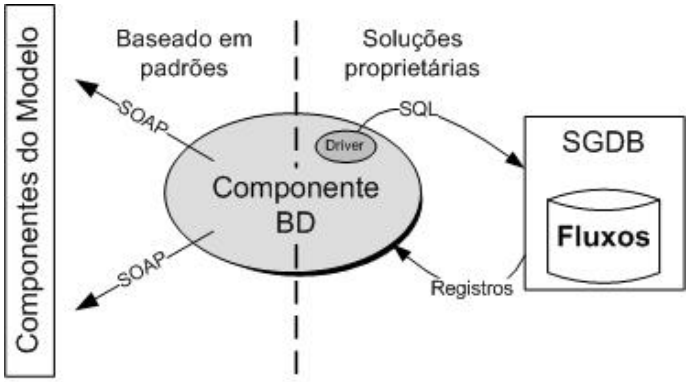

Figura 6: Contexto do Componente de BD

Deste modo, pode-se perceber que o componente de BD pertence basicamente a dois contextos. Um faz parte da interação com os SGBDs, onde são escondidos os detalhes específicos das plataformas proprietárias. O outro faz parte da comunicação com os outros componentes do modelo que têm a sua interoperabilidade garantida.

3.1.3. Outros componentes

Ainda que fora do escopo deste artigo, estão presentes no modelo SFM3 outros componentes importantes, principalmente na camada de serviços compostos. Como exemplo, podem ser citados os componentes de matriz de tráfego que a partir das informações sobre os fluxos entre dois ou mais pontos, constroem matrizes ilustrando, por exemplo, as demandas de tráfego de origem e destino. Outro exemplo seria os componentes de QoS que podem calcular medidas de desempenho sobre os fluxos de uma determinada aplicação. Por fim, um último exemplo seria os componentes de visualização que a partir dos dados sobre os fluxos podem alimentar bases RRD [CAIDA, 2002] e gerar gráficos como os apresentados pela ferramenta Flowscan [CAIDA, 2003] ou até mesmo outras formas de visualização como as apresentadas em [Sampaio et al., 2003].

3.2. Comunicação entre os componentes

Os componentes irão se comunicar com o propósito de prover serviços a aplicações ou a outros componentes. Esta comunicação ocorre por meio do protocolo SOAP, e a sua forma de interação é descrita através da linguagem WSDL. Neste caso, por exemplo, quando um coletor recebe os fluxos e precisa armazenar em banco de dados, o mesmo deve possuir o conhecimento do documento WSDL do componente de banco. A partir desta informação, o coletor saberá os passos necessários para ter sua demanda atendida.

Na Figura 7 é ilustrado um exemplo da comunicação entre os componentes coletores e os de banco de dados. Todo o relacionamento entre os elementos do modelo se faz através de requisições HTTP utilizando o protocolo SOAP para o empacotamento das mensagens, mesmo que tais elementos pertençam a domínios diferentes.

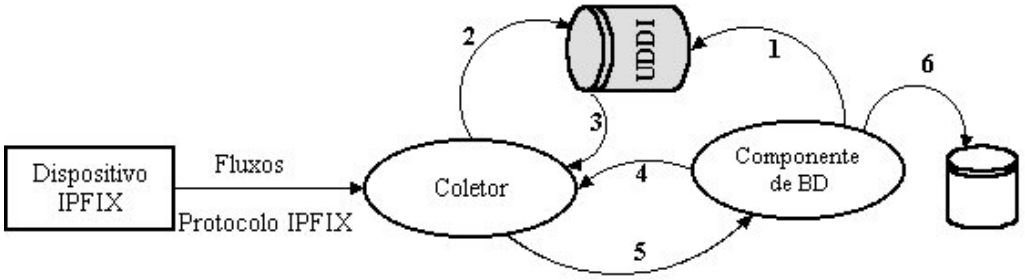

Figura 7: Comunicação entre Componentes.

Neste exemplo, após o coletor receber as informações sobre os fluxos oriundas dos dispositivos IPFIX, o mesmo tem a possibilidade de fazer o armazenamento destas informações em banco de dados. Neste sentido, o coletor deve solicitar ao componente de BD a realização deste serviço. Para isso, é necessário que o coletor tenha mecanismos de descoberta automática de onde encontrar o componente de BD e como interagir com o mesmo. É neste momento que o papel do UDDI se evidencia. Na Figura 7, o componente de BD divulga os seus serviços no diretório público (Passo 1). No Passo 2, o coletor procura por um serviço de armazenamento em banco de dados e recebe a resposta no Passo 3 por meio de uma URI (*Uniform Resource Identifier*). No Passo 4, após fazer referência à URI recebida, o coletor recebe o documento público WSDL, do componente de BD, descrevendo todos os detalhes necessários para a realização da comunicação. De posse deste documento e após a sua análise, o coletor está em condições de interagir com o componente de BD, e é exatamente o que acontece no Passo 5. Por fim, após receber as mensagens do coletore, o componente de BD armazena as informações nas suas bases de dados no Passo 6.

Através do exemplo da Figura 7 é possível perceber algumas vantagens deste modelo de gerenciamento. Dentre elas merecem destaque: Toda a comunicação é feita por meio da linguagem XML, essa possibilidade viabiliza a implementação das aplicações mediante a conveniência dos desenvolvedores sem comprometer a interoperabilidade entre esses sistemas; em função da extensibilidade do XML, cada um dos seus elementos poderá compor as suas funcionalidades a partir do aproveitamento de funções já disponibilizadas por outros elementos; a localização das funções dos componentes acontece de forma dinâmica, contemplando as funcionalidades do RTANS (*Real-Time Application Name Server*) apresentadas por Tham, Jiang and Ko [CK and Ko, 2000]; como a comunicação acontece por meio do protocolo http, diminui a probabilidade de problemas com as políticas de segurança dos domínios; este modelo de comunicação é bastante flexível e adaptável aos diversos ambientes de gerenciamento que tem o XML como base tecnológica.

3.3. Considerações finais sobre o SFM3

O SFM3, ao ser dividido em camadas, se caracteriza por ser bastante adaptável aos diversos domínios de rede, dando inclusive, diferentes níveis de abstração dos dados que estão sendo manipulados pelas aplicações. Dividir as funcionalidades entre componentes especializados e disponibilizá-las em forma de serviços, oferece às aplicações de gerenciamento um ambiente bem flexível e independente de estruturas proprietárias.

É com base no SFM3 que será definido o *framework* de serviços detalhados na próxima seção. Este *framework* fornecerá às aplicações de gerenciamento um conjunto de serviços de manipulação dos dados sobre os fluxos das redes pertencentes aos domínios em questão.

4. *Framework* de Serviços

O *framework* de serviços descrito nesta seção tem como base as funcionalidades dos componentes do modelo apresentados na seção 3. Este *framework* funciona como uma forma de *middleware*, entre as aplicações de gerenciamento e as infra-estruturas proprietárias, como pode ser visto na Figura 8.

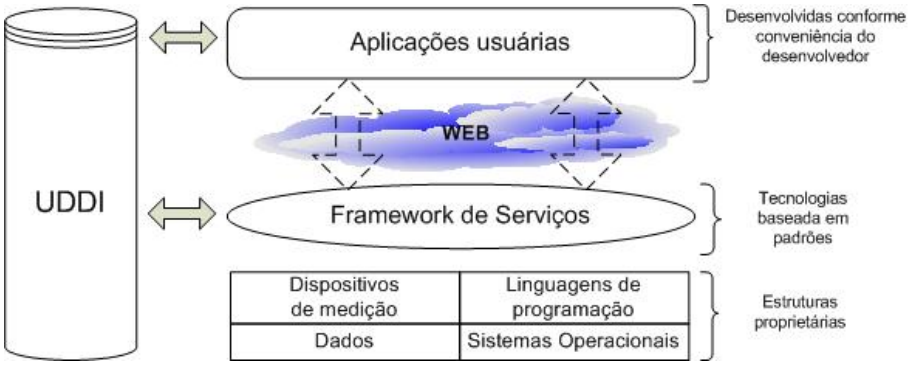

Figura 8: Framework de serviços.

Os serviços são disponibilizados utilizando tecnologias baseadas em padrões, de forma que as aplicações de gerenciamento podem fazer uso dos mesmos sem ter problemas de interoperabilidade. Isto é possível porque os componentes de software responsáveis por tais serviços escondem os detalhes de implementação das linguagens de programação, do sistema operacional, dos dispositivos de medição e dos dados.

Ainda com relação à Figura 8 pode-se perceber a presença dos serviços de diretórios UDDI. Este elemento assume um papel indispensável para o funcionamento deste *framework*, já que através dele é possível realizar a descoberta e uso dinâmico dos serviços prestados pelo componentes. Conforme apresentado na subseção 3.2, os componentes divulgam os seus serviços nestas bases de diretório e as aplicações de gerenciamento as consultam para ter o conhecimento de como realizar o acesso.

Os serviços prestados neste *framework* serão detalhados nas próximas subseções, quando serão descritas as funcionalidades dos componentes coletor e BD.

4.1. Serviços dos componentes coletores

No contexto dos relacionamentos entre os coletores e os outros componentes do modelo, existem diversas interações que podem variar de acordo com o interesse das medições. De uma forma geral, uma parte dos serviços prestados pelos coletores será em torno do comportamento que o mesmo deve ter durante a coleta dos dados oriundos dos dispositivos de medição. A outra parte será dedicada à recuperação dos dados sobre os fluxos que não estão armazenados em banco de dados. A Tabela 1 apresenta uma lista dos principais serviços propostos no modelo SFM3 do componente coletor e as suas respectivas descrições.

Estas funcionalidades do coletor são vistas como serviços na medida em que as suas utilizações visam o atendimento de uma determinada necessidade de outro componente ou de aplicações de gerenciamento.

4.2. Serviços dos componentes de BD

Na maioria das situações, para as aplicações de gerenciamento não é interessante manipular os dados sobre os fluxos na sua forma bruta. Estas aplicações necessitam ter

Tabela 1: Relação de serviços do coletor.

Nome do Serviço	Descrição
AddFlowType	Cria um novo tipo de fluxo
DelFlowType	Remove um tipo de fluxo
AddMDev	Adiciona dispositivo de medição
DelMDev	Remove dispositivo de medição
AddObsD	Adiciona domínio de observação
DelObsD	Remove domínio de observação
AddObsP	Adiciona ponto de observação
DelObsP	Remove ponto de observação
AddCompDB	Adiciona componente de BD
DelCompDB	Remove componente de BD
AnonyFlowDom	Anonimiza fluxos pertencente a um domínio

informações resultantes de um processamento prévio dos dados, visando alguma característica da rede em análise. Como exemplo, pode-se citar experiências de grupos internacionais [Romig et al., 2000], em que as informações sobre os fluxos indentificados pelo Netflow, são apresentadas em agregados de subredes. Isto se deve ao fato de que o grande volume dos dados coletados dificultam o acesso e disponibilização de informações mais específicas. Além disso, o gerenciamento, na maioria das vezes, enfoca nos dados das redes e não os dados de IPs específicos.

Sendo assim, as funções do componente de BD envolvem a realização de consultas ao SGBD e a disponibilização dos resultados destas consultas em forma de serviços. Deste modo, os usuários de tais serviços não precisarão lidar com a manipulação de dados específicos dos fluxos e sim com informações previamente processadas.

O processamento destas informações se dará por meio de consultas SQL. As consultas são elaboradas com base nos dados armazenados no banco de dados que deve estar de acordo com as especificações do modelo de dados do padrão IPFIX. Dentre as informações armazenadas, destaca-se os fluxos, domínio de observação, ponto de observação, dispositivo de medição e os tipos de fluxo.

Como pode ser visto, os componentes de BD são responsáveis por diversos serviços dedicados à disponibilização de informações acerca dos fluxos. Por estes motivos, o modelo SFM3 propõe alguns dos serviços listados na Tabela 2, na qual são relacionadas as funções que retornam o total de octetos, fluxos ou pacotes dos registros de fluxos que possuem em comum uma das características listadas no campo de descrição.

4.3. Considerações finais sobre os serviços

Nesta seção foi possível se ter uma idéia dos principais serviços disponibilizados pelos componentes localizados na primeira camada da pirâmide do modelo SFM3. Este conjunto de serviços fornece a base para o desenvolvimento de aplicações de gerenciamento, bem como pode ser utilizado para a composição de outros serviços dos componentes da camada intermediária do modelo.

5. Protótipo

Para a implantação do *framework* de serviços proposto neste trabalho, foram utilizados como base de desenvolvimento, as tecnologias de componentes servlet Java e a infra-

Tabela 2: Relação de alguns serviços do componente de BD.

Nome do Serviço	Descrição
flowTagSrc	Mesma Tag Origem
flowTagDst	Mesma Tag Destino
flowTagSrcDst	Mesma Tag Origem e Tag Destino
flowTagSrcDstProto	Mesma Tag Origem, Tag Destino e Protocolo
flowTagSrcDstPortSrc	Mesma Tag Origem, Tag Destino, Porta Origem
flowTagSrcDstPortDst	Mesma Tag Origem, Tag Destino, Porta Destino
flowTagSrcDstASSrc	Mesma Tag Origem, Tag Destino, AS Origem
flowTagSrcDstASDst	Mesma Tag Origem, Tag Destino, AS Destino
flowSrcIP	Mesmo IP de origem
flowDstIP	Mesmo IP de destino
flowSrcPort	Mesma porta de origem
flowDstPort	Mesma porta de destino
flowSrcAS	Mesmo AS de Origem
flowDstAS	Mesmo AS de Destino

estrutura de Serviços Web do projeto Apache Axis[4]. Através do pacote Axis o desenvolvedor não precisa se preocupar com o tratamento de mensagens SOAP e o uso de APIs Java mais específicas como JAX-RPC (*Java API for XML-based RPC*) e JAXM (*Java API for XML Messaging*). Além disso, uma vez criadas as classes Java responsáveis pelo serviços, os documentos WSDL com as descrições das interfaces são gerados automaticamente.

O Apache Axis funciona como um *servlet* armazenado no Tomcat. As requisições são passadas em documentos XML, empacotados em envelopes SOAP. Estas mensagens, ao chegarem aos seus destinos, têm seus envelopes SOAP processados pelo Axis que entrega para as aplicações apenas os seus conteúdos. Cabe então ao desenvolvedor, apenas a tarefa de construir estas aplicações, como pode ser observado através da Figura 9.

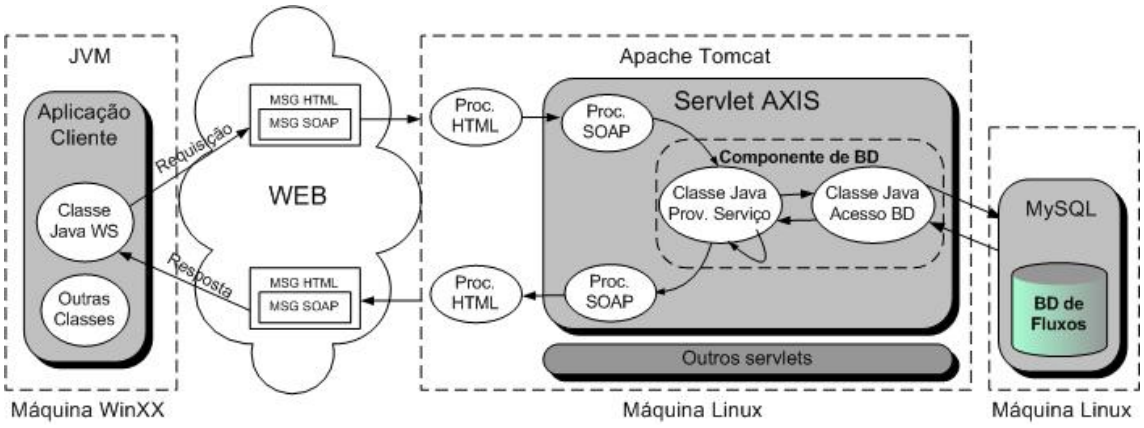

Figura 9: Funcionamento do Apache Axis.

5.1. Implementação do coletor

Como já abordado na subseção 3.1, o coletor possui um contexto ao qual se relaciona com os dispositivos de medição e o outro com os componentes do modelo SFM3, através de Serviços Web. Para o contexto dos dipositivos de medição, foram utilizados os programas

[4]http://ws.apache.org/axis/.

contido no pacote flow-tools [5], os quais possuem funções de captura, filtragem, agregação e estatística. Um resumo das tecnologias envolvidas na implementação do coletor por ser visto na Figura 10 e será descrita a seguir.

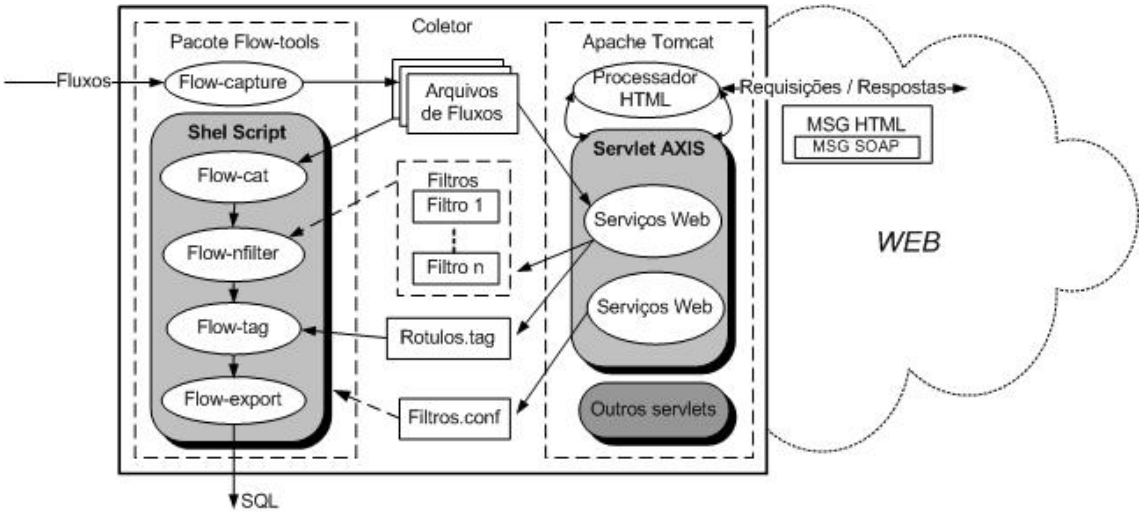

Figura 10: Tecnologias envolvidas na implementação do coletor.

5.1.1. Utilização do pacote flow-tools

Após os fluxos serem identificados pelo Netflow nos dispositivos de medição (que pode ser um roteador ou switch), os mesmos são enviados aos coletores em pacotes UDP. Para capturar estes pacotes, o coletor utiliza o aplicativo flow-capture do pacote flow-tools. O flow-capture com alguns parâmetros de linha de comando, determina, dentre outras coisas, o nível de compressão dos arquivos e o diretório onde os mesmos serão armazenados.

A linha de comando utilizada no protótipo para a captura foi: "*flow-capture -w /var/local/flows -z9 0/0/9800 -n287*". Onde, a opção *-w* indica o diretório no qual os arquivos de fluxos são armazenados, *-z9* indica o nível de compressão dos arquivos, *0/0/9800* indica os IPs de origem e destino dos fluxos e a porta (neste caso, 0 indica que pode receber fluxo de qualquer IP) e *-n* indica o intervalo de tempo em que novos arquivos de fluxos são criados.

Após coletados e armazenados em arquivos de fluxos, o coletor precisa exportar para o banco de dados as informações sobre os fluxos. Para isso, o coletor possui um *script* em bash que utiliza os aplicativos flow-cat, flow-nfilter, flow-tag e flow-export.

O flow-cat é utilizado para concatenar um conjunto de arquivos de fluxos sob um determinado critério (Dia, PoP, Device, etc). Este conjunto de arquivos é passado para o aplicativo flow-nfilter que com base nas informações dos arquivos de filtro seleciona um subconjunto de fluxos que é então passado ao flow-tag. O flow-tag tem um papel importante no protótipo implementado, já que através deste aplicativo é possível adicionar rótulos aos fluxos de acordo com características específicas (tais como subrede, AS, protocolo, etc.). Por exemplo, no caso da RNP, os rótulos foram criados com base nas subredes de cada PoP e, então, os fluxos são transferidos para o banco de dados (através do flow-export) já com as informações do PoP de origem e PoP de destino. Estas informações serão fundamentais para os experimentos da próxima seção.

[5] http://www.splintered.net/sw/flow-tools

5.2. Implementação do Componente de BD

Para implantar o componente de BD foi utilizada toda a infra-estrutura de Serviços Web descrita no início desta seção (ver Figura 9). Foi também necessário o uso de APIs JDBC para acesso ao banco de dados MySQL por parte dos componentes Java.

Como apresentado em [Sampaio et al., 2004], os serviços disponibilizados pelo componente de BD diferem um pouco na sua natureza quando comparado ao coletor. Neste componente o enfoque é voltado para a disponibilização de informações sobre os fluxos contidas nas tabelas. Para isso um conjunto de consultas SQL foram previamente elaboradas e os resultados destas consultas são disponibilizados em forma de Serviços Web (ver Tabela 2).

Estas consultas se baseiam nos dados da tabela do banco que armazena os fluxos e que foi criada de acordo com o esquema definido pela Cisco para os dados do Netflow versão 5 [Systems, 2001]. Como exemplo, pode-se citar a consulta utilizada pela função *FlowToSOcts* que fornece o número total de octetos dos fluxos pertencentes ao ToS (*Type of Service*) de número 192: "*select sum(doctets) from raw where tos=192*". O campo *doctets* registra o número de octetos de cada fluxo, *raw* é o nome da tabela utilizada e *tos* corresponde ao campo que registra os números de ToS.

6. Experimentos

Nesta seção serão apresentados dois experimentos realizados com o protótipo desenvolvido. Os dados utilizados nestes experimentos foram obtidos através da utilização do Netflow no backbone da RNP. Estas atividades fazem parte do escopo dos trabalhos do GT-QoS, o qual identificou a necessidade de uma infra-estrutura de software e hardware para a disponibilização das informações presentes nos arquivos de fluxos.

6.1. Cenário do Experimento

O ambiente de testes foi iniciado com a utilização do Netflow, para identificação dos fluxos, nos PoPs da RNP, com a sua configuração sendo realizada na principal interface de entrada do tráfego do roteador (Cisco série 7500) de cada PoP.

Os arquivos de fluxos destes PoPs são transferidos diariamente e concentrados num servidor central, conforme pode ser visto na Figura 11. O servidor, da marca Gateway modelo 8400 server, possui 4 processadores Pentium III, 2Gb de memória e 7 discos SCSI de 36Gb e o sistema operacional Linux RedHat 7.3 (kernel 2.4.18-3).

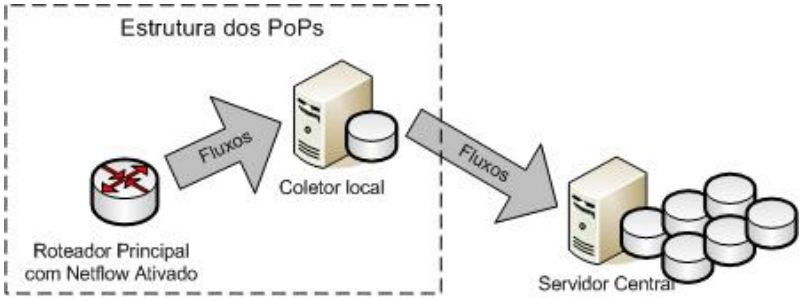

Figura 11: Coleta dos fluxos na RNP e cenário dos Experimentos.

6.1.1. Experimento 1

O objetivo deste primeiro experimento foi o de verificar a funcionalidade do protótipo entre diferentes domínios de rede. Neste sentido foi desenvolvida uma aplicação em java

que recebe como parâmetros a URI do documento WSDL do componente de banco e o nome de uma das suas funções. De acordo com estes parâmetros, o programa recupera o documento WSDL, interpreta suas informações, faz a chamada à função de interesse e por fim recupera os resultados solicitados. A Figura 12 ilustra um pequeno trecho da classe java utilizada para realizar o acesso ao componente de banco. Neste exemplo, o programa acessa a função que faz o processamento da quantidade de octetos pertencentes aos fluxos de um IP de origem num determinado intervalo de tempo (*TotalOctetsIPOrig*).

```
public class cliente1 {
 public static void main(String[] args) {
  try {
    String IPOrig = "82.82.107.225"; String t1 = "38729312"; String t2 = "49982374";
    String wsaddr = "http://200.128.80.170:8080/axis/services/ServicosBD";
    Service servico = new Service();
    Call call = (Call) servico.createCall();
    call.setOperationName(new QName(wsaddr,"TotalOctetsIPOrig"));
    call.setTargetEndpointAddress( new java.net.URL (wsaddr));
    Integer res = (Integer) call.invoke(new Object[] {new String (IPOrig), new String (t1), new String (t2)} );
    System.out.println("Resultado = " + res);
  } catch (Exception e) { System.out.println(e.getMessage());}
 }}
```

Figura 12: Trecho do código da aplicação cliente.

Na Figura 13, pode-se constatar as transferências das mensagens SOAP que foram geradas pela requisição da aplicação Java e pela resposta do componente de banco. Este monitoramento foi possível através do uso da ferramenta SOAPMonitor[6]. Através dela foi possível rastrear o atendimento dos serviços bem como checar o tempo de resposta dos serviços.

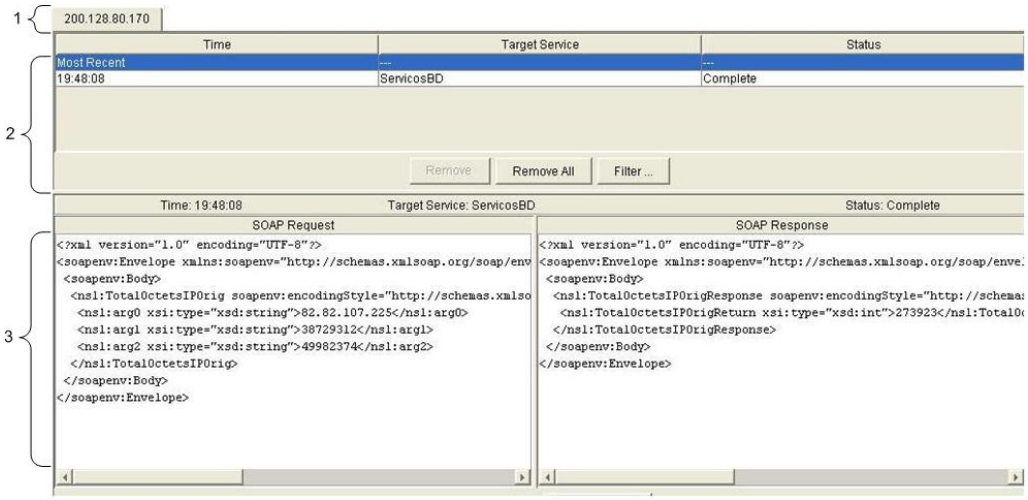

Figura 13: Mensagens monitoradas pelo SOAPMonitor.

No setor 1 da figura, é registrado o IP do servidor dos Serviços Web. No setor 2, são registradas todas as solicitações aos serviços que engloba um conjunto de informações, tais como o tempo de atendimento, o status e o nome do serviço. No setor 3, são ilustradas as mensagens SOAP de solicitação e resposta das interações entre cliente e servidor de Serviços Web.

Por fim, ainda com o objetivo de realizar experimentos com aplicações pertencentes a domínios diferentes e verificar a interoperabilidade, foi utilizado um cliente

[6]Ferramenta de monitoramento do pacote Apache Axis.

SOAP genérico[7], disponível na Internet o qual necessita apenas a URI do Serviço Web para fazer o seu uso. Da mesma forma que para a aplicação Java, não foi constatado nenhum problema no acesso ao serviço disponibilizado.

6.1.2. Experimento 2

Neste segundo experimento foi desenvolvida uma aplicação Java usuária deste *framework* de serviços que tem o objetivo de construir matrizes de tráfego a partir dos dados presentes nos fluxos identificados pelo Netflow.

Existem matrizes de tráfego de diversas naturezas envolvendo, por exemplo, o protocolo de transporte, porta TCP, subredes, sistemas autônomos, etc. Para este experimento a aplicação constrói matrizes que ilustram as demandas de origem e destino do tráfego total entre dois ou mais PoPs da RNP. Com o uso dos serviços disponibilizados no *framework*, o caminho para o desenvolvimento da aplicação ficou bastante facilitado, já que a mesma não precisou acumular funções de acesso aos dados dos fluxos e o seus respectivos processamentos.

Sendo assim, a aplicação necessita apenas fazer uso da função *FlowTagOrigDestOcts* do componente de BD, passando como parâmetro apenas o rótulo dos PoPs de origem (Linhas) e destino (Colunas) e o período de análise desejado. A Figura 14, ilustra um trecho da classe utilizada pela aplicação. Como é possível perceber, a interação entre a aplicação e o componente de BD ocorreu seis vezes (através dos dois laços), solicitando os totais de tráfego entre: PoP-BA e PoP-PE; PoP-PE e PoP-BA; PoP-BA e PoP-RN; PoP-RN e PoP-BA; PoP-PE e PoP-RN; PoP-RN e PoP-PE. Onde o PoP-BA é identificado pelo marcador de número 1, o PoP-PE pelo marcador de número 2 e o PoP-RN pelo marcador de número 3. O exemplo da matriz gerada pode ser visualizado na Figura 15.

```
try {
    String t1 = "38729312";
    String t2 = "49982374";
    String wsaddr = "http://200.128.80.170:8080/axis/services/ServicosBD";
    Service servico = new Service();
    Call call = (Call) servico.createCall();
    call.setOperationName(new QName(wsaddr, "FlowTagOrigDestOcts"));
    call.setTargetEndpointAddress(new java.net.URL(wsaddr));
    for (int l = 0; l < 3; l++) {
      for (int c = 0; c < 3; c++) {
        if (l!=c) {
          String res = (String) call.invoke(new Object[] {new Integer (l), new Integer(c),
                                    new String(t1), new String(t2)});
          tm.setValueAt(res, l, c);
        }
       } }
```

Figura 14: Trecho do cógigo utilizado para a criação da matriz.

6.2. Avaliação e considerações finais sobre os experimentos

Estes experimentos foram fundamentais para a avaliação do modelo SFM3 bem como o *framework* de serviços proposto neste trabalho. Foi possível perceber a viabilidade de implementação dos componentes da base da pirâmide do modelo e o uso das suas funcionalidades por parte dos componentes da camada intermediária.

[7] www.soapclient.com

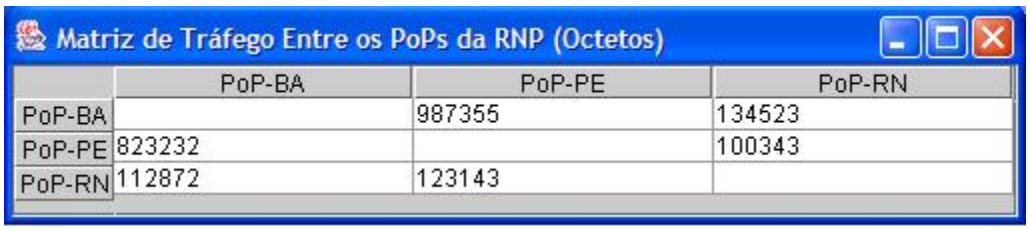

Figura 15: Matriz de tráfego gerada pela aplicação.

Através dos experimentos foi possível também constatar as vantagens da abordagem orientada a serviços na manipulação dos dados sobre os fluxos, já que os detalhes de tecnologias de cada domínio passam a ser transparentes para as aplicações usuárias dos serviços.

Por fim, os experimentos mostraram ser adequados para as ferramentas de gerenciamento, que possuem o suporte de Serviços Web e utilizam dados das medições por fluxo tendo em vista a caracterização das redes.

7. Conclusão e trabalhos futuros

A perspectiva de utilização de uma variada quantidade de ferramentas de avaliação de desempenho das redes IPs, tem motivado a diversas iniciativas na área de integração de ambientes de gerenciamento. No caso específico das medições por fluxo de tráfego, não existe um modelo de acesso aos dados presentes nos coletores de fluxos, que em sua grande maioria são administrados e manipulados por soluções predominantemente proprietárias.

Estes fatores incentivaram a elaboração de um modelo de acesso e gerenciamento dos dados sobre os fluxos dividido em camadas denominado SFM3. Este modelo apresenta diferentes níveis de apresentação das informações sobre os fluxos que podem variar dos dados brutos a informações de um interesse específico de um usuário final.

Cada camada do modelo SFM3 pode ser vista como um *framework* de serviços para a camada imediatamente superior, faciliando o processo de desenvolvimento de aplicações de gerenciamento bem como a integração entre as mesmas. Pautado nas descrições sobre as funcionalidades de cada componente participante, foi desenvolvido um protótipo em Java e utilizando as tecnologias de Serviços Web. Através deste protótipo foi possível observar e avaliar a viabilidade da implantação dos elementos descritos no modelo através de dois experimentos. No primeiro experimento foi constatada a forte tendência para a interoperabilidade com a utilização de Serviços Web, que por se basear em tecnologias como o XML, se adequa à maioria dos ambientes proprietários. Com o desenvolvimento da aplicação de matriz de tráfego no segundo experimento, foi possível verificar as vantagens no uso da abordagem orientada a serviços aliada às medições por fluxo de tráfego no que tange o gerenciamento das medições.

As experiências demonstradas neste artigo servem de base para trabalhos futuros nesta área, principalmente no que diz respeito à transformação da informação recebida pela camada intermediária e a sua transformação em conhecimento produtivo para a gerência. Este conhecimento irá permitir um comportamento cada vez mais proativo dos sistemas de monitoramento/gerência, viabilizando soluções adaptativas e com cada vez menos interferência do homem.

Para finalizar, os autores agradecem todo o apoio que a RNP vem dando ao GT-QoS e GT-QoS2 com o fornecimento da infra-estrutura técnica necessária para a

realização dos experimentos apresentados neste trabalho.

Referências

Brownlee, N. (1997). RFC 2123: Traffic flow measurement: Experiences with NeTraMet. Status: INFORMATIONAL.

Brownlee, N., Mills, C., and Ruth., G. (1999). RFC 2722: Traffic flow measurement: Architecture. Status: INFORMATIONAL.

CAIDA (2002). RRDtool: Round robin database tool. http://www.caida.org/tools/utilities/rrdtool.

CAIDA (2003). FlowScan etwork traffic flow visualization and reporting tool. http://www.caida.org/tools/utilities/flowscan.

Calato, P. (2002). Evaluation of candidate protocols for IP flow information export (IPFIX). hhttp://ipfix.doit.wisc.edu/eval/draft-calato-ipfix-lfap-eval-00.txt. Status: Expired draft.

Calato, P., Meyer, J., and Quittek, J. (2003). Information model for IP flow information export. http://www.ietf.org/internet-drafts/draft-ietf-ipfix-protocol-01.txt. Status: Working draft.

Chinnici, R., Gudgin, M., Moreau, J.-J., and Weerawarana, S. (2003). Web services description language (wsdl) version 1.2 part 1: Core language. http://www.w3.org/TR/2003/WD-wsdl12-20030611. W3C Working draft.

CK, T. and Ko, J. (2000). Monitoring QoS distribution in multimedia networks. Int. J. Network Mgmt. March/April 2000; 10: 75 – 90.

Claise, B. (2003). Evaluation of netflow version 9 against IPFIX requirements. http://www.cisco.com/warp/public/732/Tech/nmp/netflow/docs/draft-claise%-ipfix-eval-netflow-04.txt. Status: Expired draft.

Claise, B., Fullmer, M., Calato, P., and Penno, R. (2003). IPFIX protocol specifications. http://www.ietf.org/internet-drafts/draft-ietf-ipfix-protocol-01.txt. Status: Working draft.

GGF (2003). GGF: Global Grid Forum. http://www.gridforum.org/4_GP/Perf.htm.

INTERMON (2003). INTERMON: Advanced architecture for INTER-domain quality of service monitoring, modelling and visualisation. http://www.ist-intermon.org.

Internet2 (2003). E2E piPEline: End-to-end performance initiative performance environment system architecture. http://e2epi.internet2.edu/E2EpiPEs/e2epipe11.pdf.

IPFIX (2003). IPFIX: IP flow information export. http://www.ietf.org/html.charters/ipfix-charter.html.

Leinen, S. (2003). Evaluation of candidate protocols for IP flow information export (ipfix). http://www.ietf.org/internet-drafts/draft-leinen-ipfix-eval-contrib-01.%txt. Status: Working draft.

Meyer, J. (2002). Evaluation of streaming IPDR against IPFIX requirements. http://ipfix.doit.wisc.edu/eval/draft-meyer-ipfix-ipdr-eval-00.txt. Status: Expired draft.

Monteiro, J. A. S., Sampaio, L. N., and Figueredo, M. (2003). GT-QoS: Grupo de Trabalho de Qualidade de Serviço. http://www.nuperc.unifacs.br/gtqos.

OASIS (2002). Uddi version 1 specifications. http://www.oasis-open.org/committees/uddi-spec/doc/contribs.htm#uddiv1.

Papazoglou, M. (2003). Service-oriented computing: Concepts, characteristics and directions. Keynote for the 4th International Conference on Web Information Systems Engineering (WISE 2003), to appear in IEEE CS.

Papazoglou, M. P. and Georgakopoulos, D. (2003). Introduction: Service-oriented computing. *Communications of the ACM*, 46(10):24–28.

Phaal, P., Panchen, S., and McKee, N. (2001). RFC 3176: Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks.

Romig, S., Fullmer, M., and Luman, R. (2000). The OSU flow-tools package and CISCO netflow logs. In *Proceedings of the Fourteenth Systems Administration Conference (LISA-00)*, pages 291–304, Berkeley, CA. The USENIX Association.

Sadasivan, G. and Brownlee, N. (2003). Architecture model for IP flow information export. http://www.ietf.org/internet-drafts/draft-ietf-ipfix-arch-02.txt. Status: Working draft.

Sampaio, L., Almeida, M., Monteiro, J. A. S., and Mendonça, M. (2003). Um ambiente de gerenciamento de medições por fluxo de tráfego baseado na utilização de mapas em Árvores. In *II WPerformance*, pages 115–128, Campinas, SP. Anais do SBC 2003.

Sampaio, L., Granville, L. Z., and Monteiro, J. A. S. (2004). Gerenciamento de medições por fluxo de tráfego: Uma abordagem baseada no uso de banco de dados e serviços web. In *IX WGRS*, pages 15–26, Gramado, RS. Anais do SBRC 2004.

Systems, C. (2001). NetFlow Services Solutions Guide. http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.htm. White Paper.

W3C (2000). Extensible markup language (XML) 1.0 (second edition). http://www.w3.org/TR/2000/REC-xml-20001006.pdf. W3C Recommendation.

W3C (2003a). Soap version 1.2 part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624. W3C Recommendation.

W3C (2003b). Web services architecture. http://www.w3.org/TR/ws-arch/. Status: Working draft.

Zander, S. (2002). Evaluation of diameter protocol against IPFIX requirements. http://ipfix.doit.wisc.edu/eval/draft-zander-ipfix-diameter-eval-00.txt%. Status: Expired draft.

Zhang, K., Elkin, E., and Ludemann, P. (2003). Evaluation of the crane protocol against ipfix requirements. http://ipfix.doit.wisc.edu/eval/draft-kzhang-ipfix-eval-crane-00.txt. Status: Expired draft.