Monitoramento de Tráfego de Backbones Baseado em Sistemas Gerenciadores de Streams de Dados

Christian Lyra Gomes[1], **Elias Procópio Duarte Junior**[2], **Carmem Satie Hara**[2]

[1]Ponto de Presença da RNP no Paraná - PoP-PR
Caixa Postal 19037 – 81531-990 – Curitiba, PR – Brazil

[2]Departamento de Informática
Universidade Federal do Paraná – Curitiba, PR – Brazil

lyra@pop-pr.rnp.br,{elias,carmem}@inf.ufpr.br

Abstract. *Traffic monitoring of long distance network backbones presents a number of challenges due to the huge amount of traffic, large number of network components, and geographic and administrative distribution. Stream Processing Engines (SPE) are designed to perform stream analysis in real-time. This work describes a strategy for employing SPEs for backbone traffic monitoring. We have developed a tool that allows arbitrary queries to be issued about the traffic on the backbone as a whole. The tool is based on the Borealis SPE and obtains backbone traffic information based on a stream of Netflow data. Several Borealis nodes can be deployed across the backbone in a distributed fashion in order to perform real-time traffic monitoring. The architecture of the tool is presented, as well as case studies designed for validation as well as performance evaluation.*

Resumo. *O monitoramento de tráfego de backbones de redes de longa distância apresenta uma série de desafios. O volume de tráfego, o número de componentes da rede e a distribuição geográfica e administrativa destes componentes são alguns dos principais fatores que tornam árdua tal tarefa. Ferramentas tradicionais apresentam diversas limitações, em geral não permitindo consultas arbitrárias sobre as condições da rede em tempo real. Este trabalho descreve uma estratégia de monitoramento de tráfego de backbones baseada na utilização de Sistemas Gerenciadores de Streams de Dados (SGSDs). Esses sistemas foram desenvolvidos para a análise de fluxos ou streams de informações em tempo real. Foi implementada uma ferramenta baseada no SGSD Borealis e o protocolo Netflow. A ferramenta foi validada e avaliada a partir de um Ponto de Presença do backbone da RNP. Estudos de caso são apresentados, incluindo a execução de consultas diversas executadas de forma distribuída e em tempo real.*

Palavras-chave

Sistemas Gerenciadores de Streams de Dados, Medição e Monitoramento de Tráfego

1. Introdução

No gerenciamento de rede, as informações sobre a rede como um todo e seus componentes individualmente fornecem a base para a tomada de decisões relativas à gerência de

falhas, desempenho, segurança, configuração e contabilização [BRISA]. Uma parte importante dessas informações são obtidas pelo monitoramento do tráfego de rede, ou seja, da quantificação e qualificação do tráfego de uma rede.

Nos sistemas de gerência tradicionais, as informações sobre a rede e seus componentes podem ser obtidas através dos processos de *polling* e emissão de alarmes com o auxílio de ferramentas como, por exemplo, aquelas baseadas no protocolo SNMP [Stallings 1996]. As informações obtidas trazem dados diversos mantidos por elementos gerenciados como, por exemplo, quantidade de bytes ou o número de pacotes que passaram por uma determinada interface. Embora de utilização simples, as informações obtidas dessa forma podem não ser suficientes para a execução de todas as tarefas de gerência, e em particular do monitoramento de tráfego.

Os fabricantes de equipamentos de rede têm desenvolvido e incluído em seus produtos recursos que permitem a obtenção de informações mais detalhadas sobre o tráfego de rede. Protocolos como o *Netflow* [Cisco 2007] e *Sflow* [Phaal et al. 2001] são utilizados para obter informações sobre *fluxos* de pacotes. Um fluxo de pacotes pode ser definido como um conjunto de pacotes com os mesmos protocolos, endereços de origem e endereço de destino [Cisco 2007]. As informações obtidas com o auxílio destes protocolos podem ser armazenadas e analisadas com ferramentas como o *flow-tools* [Fullmer 2007, Fullmer e Romig 2000] e o *ntop* [Ntop Team 2007].

Outra maneira de obter mais informações sobre o tráfego de rede é através da utilização de programas conhecidos como *sniffers* que são capazes de armazenar e decodificar todo o tráfego de rede direcionado para eles [Jacobson et al.]. Embora essa seja uma maneira que permite colher maior número de informações, é necessário fazer com que todo o tráfego de rede seja visto pelos *sniffers*, o que envolve a utilização de equipamentos especiais, ou configurações especiais de rede. Além disso, em uma rede de médio/grande porte a quantidade de dados a serem tratados pode ser muito grande.

A utilização tanto de *sniffers*, quanto de protocolos como *Netflow*, implica em um processamento que geralmente não é feito em tempo real, ou seja, os dados são coletados e armazenados, para serem processados e apresentados posteriormente. Os Sistemas Gerenciadores de Streams de Dados (*SGSDs*) foram desenvolvidos visando oferecer funcionalidades semelhantes aos Sistemas Gerenciadores de Banco de Dados (SGBD), mas permitindo a execução de consultas sobre fluxos contínuos de dados (*streams*), com a obtenção de respostas em tempo real.

O presente trabalho apresenta uma estratégia para o monitoramento de tráfego baseada em um SGSD e que permite ao administrador de uma rede definir consultas arbitrárias sobre os dados de tráfego obtidos de forma distribuída em todo um backbone. Foi implementada uma ferramenta para efetuar tal tarefa, que utiliza a capacidade de execução de consultas de forma distribuída de um SGSD específico: o Borealis [Abadi et al. 2005]. A ferramenta foi validada e avaliada, e a sua funcionalidade é demonstrada através de dois estudos de caso onde ela foi aplicada para a construção de uma matriz de tráfego e detecção de varreduras de rede.

As principais contribuições deste trabalho incluem: a definição de uma estratégia para monitoramento de tráfego baseado em um SGSD; a implementação de uma ferramenta que permite o monitoramento de tráfego de um backbone de forma genérica e em

tempo real; e a avaliação de desempenho da ferramenta.

O restante deste trabalho está organizado da seguinte maneira. A seção 2 descreve trabalhos relacionados. A seção 3 apresenta os Sistemas Gerenciadores de Streams de Dados (SGSDs). A seção 4 apresenta uma estratégia para o monitoramento de tráfego baseado em um SGSD e a ferramenta implementada. As seções 5 e 6 trazem os testes e estudos de caso da utilização da ferramenta, respectivamente, seguidas pela conclusão na seção 7.

2. Trabalhos Relacionados

A área de gerência de redes e monitoramento de tráfego é rica em trabalhos e ferramentas relacionadas à proposta deste trabalho. Em [Cottrell 2008] é possível encontrar uma extensa lista destas ferramentas.

Ferramentas que trabalham no nível de pacotes, como o *tcpdump* [Jacobson et al.] e o *wireshark* [Combs], que capturam e mostram esses pacotes são bastante populares entre os administradores de redes. No entanto, tais ferramentas não permitem a realização de consultas arbitrárias sobre o tráfego capturado. Além disso, estas ferramentas usualmente são alimentadas por uma única fonte de dados. Em [Munz e Carle 2008] é descrita uma solução que consiste de coletores distribuídos capazes de receber informações no formato *Netflow* e exportar os dados recebidos no formato *pcap* para que o *wireshark* os analise. No entanto os autores reconhecem que a quantidade de informação pode ser grande e recomendam a filtragem dos dados de interesse nos próprios coletores.

O *ntop* [Deri e Suin 2000] é uma ferramenta capaz de trabalhar tanto no nível de pacotes, quanto no nível de fluxos através de um *plugin* que é capaz de receber informações de tráfego via *Netflow*. Embora seja capaz de apresentar vários tipos de relatórios, ela também não permite a utilização de consultas arbitrárias definidas pelo administrador de redes.

Dentre as ferramentas que trabalham com *Netflow*, uma das primeiras a ser distribuída com código aberto foi a *cflowd* [CAIDA 1998], que mais tarde deu origem ao *flowscan* [Plonka]. Os autores de [Dubendorfer et al. 2005] afirmam que sistemas que trabalham no nível de pacotes não escalam em redes de alta velocidade e descrevem um *framework* para monitoramento em tempo real de backbones e detecção de ataques baseado em *Netflow*. O *framework* descrito funciona de forma centralizada e requer que os usuários escrevam *plugins* para cada tarefa a ser realizada.

Em [Bin et al. 2008] é descrito um sistema para monitoramento de tráfego baseado em *Netflow*, no qual as informações são armazenadas em um banco de dados *Oracle*. A utilização de bancos de dados permite uma grande flexibilidade na realização das consultas. No entanto, tal sistema ainda requer que a informação seja armazenada para depois ser processada, além de funcionar de forma centralizada.

Os autores da ferramenta *SMART* [Zhou et al. 2008] argumentam que as ferramentas tradicionais que utilizam *Netflow* trabalham armazenando informações em disco para depois processá-las, o que impede o processamento de grandes volumes de tráfego de maneira eficiente. Afirmam também que Sistemas Gerenciadores de Streams de Dados (SGSDs, descritos na seção 3) não foram feitos especificamente para o monitoramento de tráfego com *Netflow*. O *SMART* utiliza o armazenamento e processamento dos dados em

memória. No entanto, os resultados obtidos pelo *SMART*, que foi capaz de processar 30 mil registros por segundo, é similar ao obtido no presente trabalho, que utiliza um SGSD.

Em [Cranor et al. 2003, Cranor et al. 2002] é descrito o sistema *Gigascope*, que funciona como um SGSD específico para o monitoramento de redes de alta velocidade. Embora os resultados sejam os mais significativos em se tratando de monitoramento de redes, o *Gigascope* é uma ferramenta comercial e de código fechado, utilizada pela AT&T.

Motivados em investigar a utilização de um SGSD de código aberto para o monitoramento de tráfego, os autores de [Plagemann et al. 2004] descrevem um estudo de caso que utilizou o SGSD TelegraphCQ. Os resultados obtidos, em termos de quantidade de tráfego analisado, foram modestos, mas serviram de motivação para a realização de outros trabalhos, como o da ferramenta *PaQueT* [Ligocki et al. 2008]. A *PaQueT* utiliza o SGSD Borealis e permite que o administrador de redes efetue consultas arbitrárias no nível de pacote sobre o tráfego capturado em uma interface de rede, ou sobre um arquivo no formato *pcap*.

Assim como a *PaQueT*, o presente trabalho também utiliza o SGSD Borealis para o monitoramento de tráfego. Porém, ao contrário da *PaQueT*, que permite o monitoramento de *um* segmento de rede, a ferramenta apresentada no presente trabalho permite o monitoramento de tráfego de um backbone, tendo como características principais: a possibilidade de utilizar múltiplas fontes de dados, processamento distribuído em múltiplos nodos Borealis, separação das funções de captura, controle e visualização, além de trabalhar no nível de fluxo utilizando o *Netflow*.

Embora não tenha sido utilizado no presente trabalho, uma alternativa ao uso do *Netflow* é o *SFlow*, que é uma tecnologia para monitoramento de tráfego definida pela RFC 3176 [Phaal et al. 2001]. Nela é definido um mecanismo de amostragem de pacotes em um agente *SFlow*, uma MIB e um formato de dados utilizado na comunicação entre o agente e um coletor *SFlow*. Como no agente *SFLow* não é necessário manter informações de fluxos, a implementação do agente é bastante simplificada. Além disso, a especificação foi projetada para atender de forma precisa o monitoramento de interfaces a velocidade de Gigabits por segundo ou mais. O *SFlow* utiliza amostragem estatística de pacotes. No entanto, como o coletor que recebe as amostras conhece a razão de amostragem, é possível reconstruir com razoável precisão o tráfego amostrado [Phaal e Panchen 2007, Duffield et al. 2002].

3. Sistemas Gerenciadores de Streams de Dados

Os Sistemas Gerenciadores de Streams de Dados (SGSDs) [Carney et al. 2002] surgiram das necessidades de uma nova classe de aplicações para monitorar eventos em tempo real, tais como análise de dados de sensores, monitoramento de tráfego, análise contínua de ações da bolsa de valores e monitoramento de campos de batalha. Dado o volume de dados que tais aplicações geram e a necessidade de efetuar consultas sobre esses dados indicam que elas se beneficiariam do uso de um Sistema Gerenciador de Banco de Dados (SGBD). No entanto, os SGBDs tradicionais não são capazes de atender todos os requisitos dessa classe de aplicações, que possuem em comum a necessidade de processamento em tempo real, além do processamento de entradas de dados alimentadas continuamente e em grande volume.

Os primeiros protótipos de SGSDs foram desenvolvidos no início desta

década, como por exemplo o projeto Aurora [Carney et al. 2002], o projeto TelegraphCQ [Chandrasekaran et al. 2003], que foi baseado no projeto Telegraph, e o projeto STREAM [Arasu et al. 2003]. Esses primeiros esforços tinham como foco o projeto de novos operadores e novas linguagens, assim como o processamento de alto desempenho operando de forma centralizada.

Os SGSDs de segunda geração, como o Medusa [Cherniack et al. 2003] e o Borealis [Abadi et al. 2005], foram construídos com base na experiência adquirida com os sistemas de primeira geração, e evoluíram para se tornarem sistemas distribuídos e com recursos como tolerância a falhas, balanceamento de carga e processamento paralelo. O Borealis é um SGSD distribuído de segunda geração desenvolvido por um grupo de pesquisadores de três universidades: Brandeis, Brown e MIT [Abadi et al. 2005]. O Borealis estende e modifica funcionalidades presentes nos projetos Aurora [Carney et al. 2002] e Medusa [Cherniack et al. 2003].

De uma maneira simplificada, um sistema Borealis recebe um conjunto de streams como entrada, e continuamente processa os dados do stream, agregando, filtrando e correlacionando-os de forma a produzir uma saída de interesse de outra aplicação. Um exemplo de consulta está ilustrado na Figura 1. Nesta consulta é definida uma janela de 300 segundos, na qual registros *Netflow* que possuem o mesmo sistema autônomo (AS) de origem e destino são agrupados e para cada agrupamento é gerada a soma dos octetos. No Borealis, um operador que processe um stream pode estar distribuído por várias máquinas físicas, chamadas de nodos processadores, ou simplesmente nodos. Assim, no exemplo da Figura 1 é possível que as operações de agrupamento estejam sendo realizadas em um servidor a a operação de união em outro.

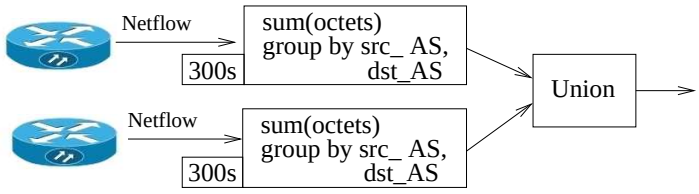

Figura 1. Consulta para gerar uma matriz de tráfego

Um sistema Borealis é composto pelos seguintes componentes: o *catálogo distribuído*, *nodos*, *aplicações clientes*, e *fontes de dados*, descritos a seguir. O *catálogo distribuído* guarda informações do sistema como um todo. Esta informação inclui o diagrama de consultas, ou seja, o conjunto de todos os operadores e todos os streams, e a informação de distribuição, que especifica a designação de operadores aos nodos processadores. Os *nodos* são os responsáveis pelo processamento dos streams. Cada nodo executa uma porção do diagrama de consulta e armazena informações sobre essa porção em um catálogo local. Os *nodos* também executam as tarefas necessárias para permitir a gerência de carga e garantir a tolerância a falhas. As *aplicações clientes* interagem com o Borealis produzindo dados ou consumindo os resultados produzidos pelo sistema. Essas aplicações podem também criar ou modificar o diagrama de consulta. O sistema Borealis vem com duas aplicações clientes por padrão. Um cliente com uma interface gráfica para a modificação de um diagrama de execução, e uma ferramenta de monitoramento capaz de mostrar as condições de carga e designação. As *fontes de dados* são aplicações cliente que produzem streams de dados e as enviam para o processamento pelos nodos.

4. Uma Estratégia para Monitoramento de Tráfego de Redes Baseada em SGSDs

A estratégia proposta neste artigo permite que um administrador de rede efetue consultas arbitrárias sobre os dados de tráfego obtidos de forma distribuída em todo um backbone. Foi desenvolvida uma ferramenta que pode ser aplicada a um backbone, com múltiplos equipamentos gerando fluxos em localidades diferentes. Para tanto é utilizada a capacidade de execução de consultas de forma distribuída do SGSD Borealis [Abadi et al. 2005].

A estratégia proposta utiliza nodos do SGSD localizados próximos às fontes de dados, e é constituída por três conjuntos de aplicações: aquisição de dados, controle do sistema e apresentação. As aplicações de aquisição são responsáveis por receber e processar os registros para que possuam o formato de entrada de um nodo Borealis. Uma vez convertido, o resultado é enviado para o nodo Borealis usando o protocolo de comunicação do sistema. Os resultados gerados pelos nodos Borealis devem ser recebidos por uma aplicação que seja capaz de entender o protocolo de comunicação do Borealis a fim de apresentá-los ao usuário. Esta é a finalidade das aplicações de apresentação. A Figura 2 apresenta uma visão geral da arquitetura. O sistema pode ser composto de vários nodos Borealis, vários nodos "alimentadores"(*aquisição*) e vários nodos de *apresentação*. Além destes, é necessário também um mecanismo de *controle* que permita a coordenação destes nodos. As aplicações de cada conjunto podem ser executadas em um nodo distinto, ou podem ser agregadas em um mesmo nodo.

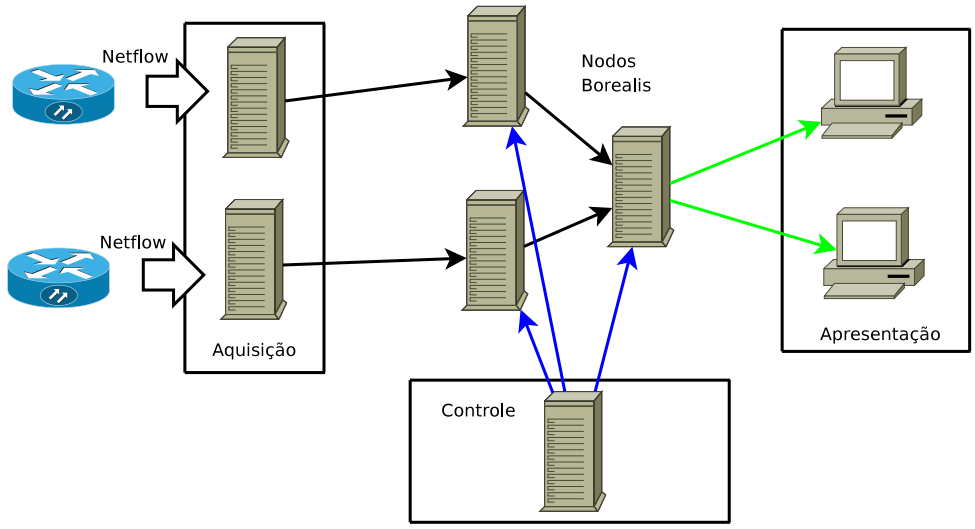

Figura 2. A estratégia de monitoramento de tráfego baseada no SGSD Borealis.

4.1. Aquisição de Dados

O módulo de aquisição de dados é responsável por fazer a interface entre as fontes de dados e os nodos Borealis. Dois grupos de tarefas são executadas por este módulo: a aquisição e conversão dos dados, além do envio dos dados para os nodos Borealis.

A ferramenta proposta utiliza os dados obtidos através do protocolo *Netflow*. No entanto também é possível utilizar outras fontes de dados, desde que as informações pro-

vidas possam ser transformadas no formato de entrada esperado. A fim de manter uma certa independência da fonte de dados, a aquisição e conversão dos dados são executadas por uma aplicação e o envio dos dados para os nodos Borealis é feito por outra aplicação.

Para suprir a função de fonte de dados utilizando o protocolo *Netflow* foi utilizada uma ferramenta livre chamada de *New Netflow Collector* (*nnfc*) [Klyachkin]. O *nnfc* é um coletor de fluxos *Netflow*, isto é, ele é capaz de receber fluxos *Netflow* enviados por um roteador, decodificá-los e armazená-los. Um *plugin* foi escrito para o *nnfc* para que os fluxos recebidos e decodificados pudessem ser enviados através de um mecanismo de IPC (*Inter Process Communication*) para a aplicação de envio de dados para o nodo Borealis.

A aplicação que faz o envio propriamente dito de informações para o Borealis recebeu o nome de *flowsender*, e foi implementado em C++ utilizando-se a API (*Application Programming Interface*) e bibliotecas do Borealis. O *flowsender* é capaz de ler registros de fluxo das filas em memória e enviá-los para um nodo Borealis.

Para fins de teste e validação uma pequena aplicação para a geração de carga sintética foi desenvolvida. O *dummysender* é capaz de gerar registros de fluxo em três modos: a partir de um arquivo texto, aleatoriamente, ou registros com conteúdo fixo. O número de registros a serem gerados, bem como a fila em memória na qual os fluxos devem ser armazenados podem ser definidos via linha de comando. Como os fluxos a serem gerados são conhecidos *a priori*, torna-se trivial a comparação com os resultados gerados pelo Borealis.

4.2. Controle

São duas as aplicações de controle: o *BigGiantHead* e o *BigMouth*. O *BigGiantHead* é uma aplicação que acompanha o Borealis e é utilizada para controlar os nodos Borealis. O *BigGiantHead* pode ser utilizado de duas maneiras: transiente ou persistente. No modo transiente, ele é chamado por uma aplicação cliente do Borealis e finalizado tão logo tenha lido o arquivo XML que contém a definição dos fluxos e consultas, e tenha feito o *deploy* dessas consultas, ou seja, tenha informado aos nodos Borealis o que cada um deve executar. No modo persistente, o *BigGiantHead* funciona como um *daemon*, e é capaz de se comunicar tanto com nodos Borealis quanto com outras aplicações clientes, permitindo o registro de fluxos e consultas.

Na arquitetura proposta, o *BigGiantHead* é executado em modo persistente. O usuário do sistema utiliza uma aplicação cliente, o *BigMouth*, para informar ao *BigGiantHead* o que o sistema como um todo deve executar.

O funcionamento do *BigMouth* é bastante simples, bastando informar via linha de comando o endereço e porta de destino do *BigGiantHead* e um arquivo XML contendo as informações de entradas, consultas e saídas. A comunicação entre os dois é feita pela rede, o que permite que sejam executados em máquinas distintas.

4.3. Apresentação

As informações processadas pelos nodos são enviadas utilizando o protocolo de comunicação do Borealis. A aplicação cliente do Borealis deve ser capaz de decodificá-las para poder exibi-las ao usuário.

Como o esquema de saída não é fixo, isto é, ele depende da consulta a ser realizada, a aplicação de visualização deve ser capaz de inferí-lo. Por este motivo a aplicação

foi chamada de *ureceiver* (*universal receiver*). Para executar o *ureceiver* é necessário informar via linha de comando a porta na qual a aplicação deve aceitar conexões dos nodos Borealis para receber os resultados, bem como o arquivo XML contendo a consulta a ser feita, para que a inferência do esquema de saída seja realizada.

Ao ser executado, o *ureceiver* aguarda conexões dos nodos Borealis, e mostra a cada intervalo de tempo definido na consulta, os resultados gerados pelo Borealis.

5. Validação e Avaliação

Para validar e avaliar o desempenho da ferramenta proposta, três cenários de testes foram criados, e são descritos a seguir.

5.1. Teste 1: Um Único Nodo Borealis e Carga Sintética

O primeiro teste executado tem como objetivo validar a ferramenta. Uma carga sintética foi gerada e o resultado retornado foi comparado com o resultado esperado. O teste permitiu também avaliar o desempenho do nodo Borealis. A topologia montada para o primeiro teste é mostrada na Figura 3.

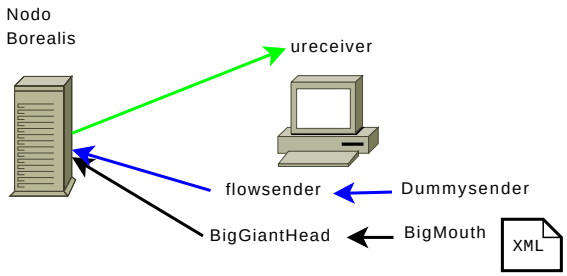

Figura 3. Teste 1: validação.

O ambiente de testes foi composto por duas máquinas: o nodo Borealis e o nodo cliente. A máquina utilizada como nodo Borealis possui uma CPU Athlon XP 2600+ e 1 GB de memória. A máquina cliente, na qual foi executado o *BigGiantHead*, *dummysender*, *flowsender* e *ureceiver*, possui uma CPU Celeron 900 e 2GB de memória. Ambas as máquinas estão conectadas em uma rede local *Ethernet* através de um switch de 100Mbps.

Para verificar se as respostas produzidas pelo sistema eram corretas foram gerados registros de fluxos a velocidades pré-determinadas, variando de mil a 40 mil registros por segundo em incrementos de mil registros. Uma consulta simples que contabiliza o número de pacotes de todos os registros a cada janela de 10 segundos foi empregada. Como o conteúdo de cada registro era igual e previamente conhecido, a verificação da precisão da resposta foi trivial.

Os testes foram repetidos 10 vezes. Foi observado que o ponto de saturação do sistema se situou em torno dos 35 mil registros por segundo. Ao se atingir essa taxa de geração de fluxos, o nodo Borealis atingia uma condição de erro e parava de contabilizar pacotes. O resultado obtido é compatível com um experimento similar descrito em [Ligocki et al. 2008]. Para todas as velocidades testadas abaixo do ponto do saturação, o sistema produziu respostas corretas.

Para investigar a influência da CPU nos resultados obtidos, o teste foi repetido utilizando-se o mesmo cenário, mas desta vez executando o nodo Borealis em uma

máquina com CPU Intel Core 2 Duo de 2Ghz e 1 GB de memória RAM. O ponto de saturação do sistema observado foi de aproximadamente 60 mil registros por segundo.

5.2. Teste 2: Múltiplos Nodos Borealis e Carga Sintética

O primeiro teste identificou um limite na quantidade de registros processados por segundo por um único nodo Borealis. O objetivo do segundo teste é determinar se múltiplos nodos Borealis podem ser combinados para processar um número maior de registros. A topologia utilizada no segundo teste é mostrada na Figura 4.

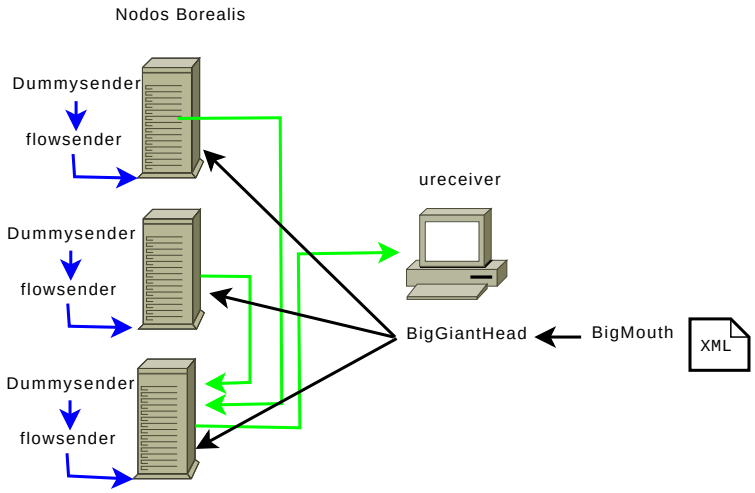

Figura 4. Teste 2: Múltiplos nodos Borealis.

O ambiente de testes foi composto por 3 nodos Borealis e um nodo cliente. As máquinas utilizadas como nodos Borealis possuem CPUs Athlon XP 2600+, Athlon XP 2400+ e Athlon XP 2200+, e 1GB de memória cada. A máquina cliente é a mesma empregada no teste anterior, isto é, uma CPU Celeron 900 com 2GB de memória.

Neste teste cada nodo Borealis executou também o *dummysender* e o *flowsender* para que os registros de fluxo fossem gerados localmente. Na máquina cliente foi executada a aplicação *BigGiantHead*, além do *ureceiver*.

A consulta de contabilização de pacotes utilizada no primeiro teste foi modificada para que cada nodo Borealis contabilizasse os seus pacotes. Os resultados parciais produzidos por cada nodo foram enviados para o primeiro nodo para que o resultado total fosse contabilizado por este.

Em cada nodo Borealis os registros de fluxo foram gerados a uma velocidade de 30 mil registros por segundo, e o resultado recebido pelo ureceiver apresentava a resposta correta esperada para o total de 90 mil pacotes por segundo em todas as 10 repetições do teste.

5.3. Teste 3: Carga Real

O terceiro teste tem como objetivo avaliar a utilização da ferramenta com uma carga real. O teste foi executado no Ponto de Presença no Paraná (PoP-PR) da Rede Nacional de Ensino e Pesquisa (RNP). No PoP-PR a contabilização de tráfego utilizando *Netflow* é

feita com o auxílio de uma máquina que observa de maneira passiva o tráfego em uma porta “espelhada” de um switch. Uma aplicação chamada *nprobe* [Deri] gera os registros no formato *Netflow* que são enviados para outras máquinas que coletam, armazenam e processam esses registros. A máquina que gera os registros *Netflow* foi configurada de forma a replicar os fluxos recebidos, enviando uma cópia para um nodo Borealis.

Como a máquina do PoP-PR que armazena os registros de *Netflow* o faz gravando os fluxos recebidos a cada intervalos de 5 minutos em arquivos, optou-se também por usar nas consultas feitas no nodo Borealis janelas de 5 minutos, pois isso permite que os resultados obtidos com o Borealis fossem confrontados com os resultados obtidos do processamento dos arquivos.

A carga real, ou seja o fluxo de registros *Netflow*, provida pela máquina que captura o tráfego, tem uma característica bem distinta da carga sintética usada até então. Enquanto que a carga sintética foi gerada de maneira contínua e a uma velocidade determinada, a carga real se apresenta na forma de rajadas. A cada intervalo de aproximadamente 30 segundos um conjunto de registros é enviado. A Figura 5 mostra a taxa de chegada de registros por segundo em um intervalo de tempo de 5 minutos.

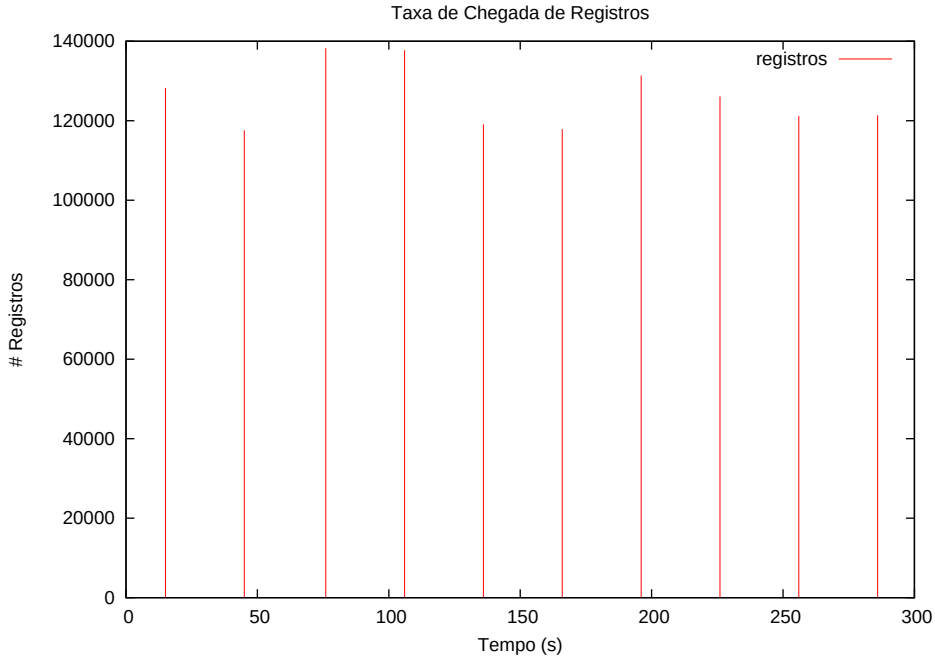

Figura 5. Teste 3: Chegada de novos registros por segundo.

O tráfego observado pela máquina que gera os registros de *Netflow* é da ordem de 1Gbps e 100 mil pacotes/s. Quando esse tráfego é sumarizado em registros *Netflow* ele gera uma quantidade de registros da ordem de 1 milhão de registros a cada intervalo de 5 minutos.

A consulta utilizada nos dois primeiros testes foi repetida nesse novo cenário, utilizando como nodo Borealis uma máquina com CPU Intel Core 2 Duo de 2Ghz e 1 GB de memória RAM. A Figura 6 mostra o número de registros contabilizados por esta máquina e o número de registros armazenados em arquivos para cada intervalo de 5 minutos du-

rante uma hora de execução do teste.

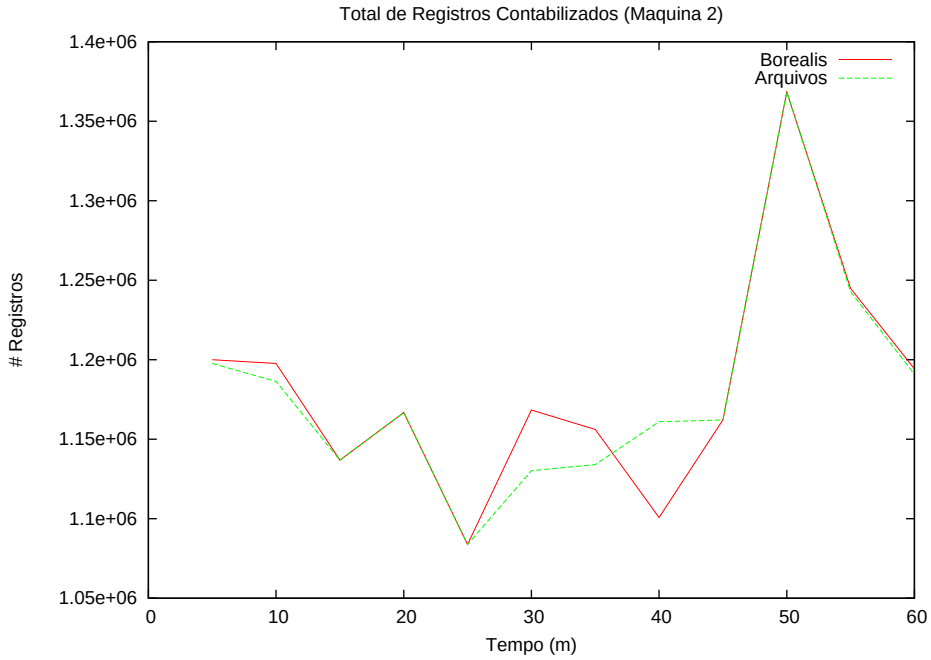

Figura 6. Teste 3: Número de registros contabilizados pelo Borealis (Máquina 2).

As pequenas diferenças observadas entre os valores são devido ao fato de que a ferramenta que armazena os registros em arquivos utiliza a hora local da máquina para determinar o conteúdo de cada arquivo, enquanto que o Borealis utiliza um valor de *timestamp* do próprio registro. Isto é, ao término de um intervalo de 5 minutos, contados pelo relógio da máquina local, o coletor *Netflow* armazena em um arquivo todos os registros recebidos até então. É possível que esse intervalo de 5 minutos aconteça durante uma rajada de pacotes. Já o Borealis não utiliza um relógio local, mas considera somente a informação de *timestamp* presente no registro. Sendo assim, é possível que o Borealis processe um pacote com um determinado *timestamp* X mesmo que a hora local já seja X+1.

6. Estudos de Caso

Como forma de demonstrar a funcionalidade da ferramenta proposta dois estudos de caso ou cenários de aplicação são descritos. Cada estudo de caso corresponde a tarefas comuns na gerência de backbones.

6.1. Matriz de Tráfego

Uma matriz de tráfego consiste de valores de tráfego medidos dois a dois dentre os participantes de uma determinada rede. A matriz de tráfego é particularmente útil dentro de um Ponto de Troca de Tráfego, uma estrutura na qual diferentes Sistemas Autônomos (AS) trocam tráfego entre si. As informações de quanto tráfego cada participante envia para cada outro participante é importante para a área de engenharia de tráfego de cada Sistema Autônomo e permite a definição de melhores políticas de roteamento. A obtenção da matriz de tráfego é bastante simples pois basta agregar os fluxos por AS de origem e destino contabilizando o número de octetos, como ilustrada na consulta da Figura 1.

Ao ser executada, a consulta gera, a cada janela de tempo, resultados contendo o *timestamp* da janela, o AS de origem, o AS de destino e a quantidade de octetos. A Figura 7 apresenta um exemplo de saída de resultados da consulta.

```
1217257500 65006 65001 16488787
1217257500 65000 65001 24356097
1217257500 65001 65000 572393822
1217257500 65009 65001 8272830
1217257500 65001 65009 279961720
1217257800 65000 65001 71098238
1217257800 65001 65000 1344944043
1217257800 65006 65001 34283545
1217257800 65009 65001 18086315
1217257800 65001 65009 627027711
```

Figura 7. Resultados da consulta matriz de tráfego.

6.2. Detecção de Anomalias

O segundo exemplo de utilização mostra como utilizar a ferramenta para detectar uma anomalia de tráfego conhecida como varredura de rede. Um varredura de rede acontece quando um host envia pacotes para vários outros hosts de uma rede com objetivo de descobrir quais respondem e quais não. A característica desses tipos de sonda é a grande quantidade de fluxos com apenas 1 pacote (em geral apenas um pacote UDP, ou um pacote ICMP é enviado). Para detectar esse tipo anomalia a consulta deve primeiro filtrar os fluxos com apenas 1 pacote. Em seguida ela agrega os fluxos por endereço IP de origem e destino, de forma a selecionar registros distintos de origem/destino. Uma nova agregação é feita agrupando apenas pela origem, contando assim os destinos distintos. Opcionalmente é possível estabelecer um valor de "corte"ou seja, hosts que produziram até um determinado número de fluxos com apenas 1 pacote e que podem ser considerados "normais".

Ao ser executada, a consulta gera, a cada janela de tempo, resultados contendo o *timestamp* da janela, o endereço IP de origem, e a quantidade endereços IP de destino distintos. A Figura 8 apresenta um exemplo de saída de resultados da consulta.

```
1217260500 10.36.11 1613
1217260500 10.107.209.1 577
1217260500 10.107.241.100 818
1217260500 10.107.230.63 747
1217260500 10.17.232.196 620
```

Figura 8. Resultados da consulta detecção de varreduras de rede.

7. Conclusão

Este trabalho descreve uma estratégia para a utilização de Sistemas Gerenciadores de Streams de Dados no monitoramento de tráfego de backbones em tempo real. Uma ferramenta que utiliza o SGSD Borealis e *Netflow* foi implementada, validada e o seu desempenho foi avaliado. Os resultados, tanto em desempenho, quanto em funcionalidade, demonstram a

viabilidade da proposta. Os resultados experimentais utilizando um ambiente de produção demonstram que é possível processar mais de um milhão de registros *Netflow* a cada intervalo de 5 minutos, e que os resultados são tão precisos quanto os obtidos pelo métodos tradicionais que utilizam o armazenamento dos fluxos em arquivos e processamento "off-line" dos mesmos. Trabalhos futuros incluem melhorias no SGSD Borealis, bem como na ferramenta, como por exemplo a criação de interfaces que permitam a visualização de resultados graficamente.

Referências

Abadi, D. J., Ahmad, Y., Balazinska, M., Çentintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A. S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S. (2005). The Design of the Borealis Stream Processing Engine. *Proceedings of the 2005 CIDR Conference*.

Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. (2003). STREAM: The Stanford Data Stream Management System. *IEEE Data Engineering Bulletin*, 26(1).

Bin, L., Chuang, L., Jian, Q., Jianping, H., and Ungsunan, P. (2008). A NetFlow based flow Analisys and Monitoring System in Enterprise Networks. *Computer Networks*, (52):1074–1092.

BRISA. *Gerencimanento de Redes - Uma abordagem de sistemas abertos*. Makron.

CAIDA (1998). cflowd tools. `http://www.caida.org/tools/measurement/cflowd`.

Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., and Zdonik, S. (2002). Monitoring streams: a new class of data management applications. *Procedings of the 27th internation conference on very large databases, Hon Kong, August 2002*, pages 215–226.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., and Shah, M. (2003). TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR'03)*.

Cherniack, M., Balakrishnan, H., Balazinska, M., Carney, D., Cetintemel, U., Xing, Y., and Zdonik, S. (2003). Scalable Distributed Stream Processing. *Proceedings of the 2003 CIDR COnference*.

Cisco (2007). NetFlow Services and Applications - White Paper.

Combs, G. wireshark. `http://www.wireshark.org`.

Cottrell, L. (2008). Network monitoring tools. `http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html`.

Cranor, C., Gao, Y., Johnson, T., Shkapenyuk, V., and Spatscheck, O. (2002). Gigascope: High Performance Network Monitoring with a SQL Interface. *ACM SIGMOD 2002*, pages 623–627.

Cranor, C., Johnson, T., and Spataschek, O. (2003). Gigascope: A Stream Database for Network Applications. *ACM SIGMOD 2003*, pages 647–651.

Deri, L. nProbe. http://www.ntop.org/nProbe.html.

Deri, L. and Suin, S. (2000). Effective Traffic Measurement Using ntop. *IEEE Communications Magazine*, pages 138–143.

Dubendorfer, T., Wagner, A., and Plattner, B. (2005). A Framework for Real-Time Worm Attack Detection and Backbone Monitoring. *Procedings of the 2005 First IEEE Internation Workshop on Critical Infrastructure Protection*.

Duffield, N., Lund, C., and Thorup, M. (2002). Properties and Prediction of Flow Statistics from Sampled Packet Streams. *Procedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, Marseille, France, 2002*, pages 159–171.

Fullmer, M. (2007). Flow-tools. http://www.splintered.net/sw/flow-tools/.

Fullmer, M. and Romig, S. (2000). The OSU flow-tools package and CISCO netflow logs. *Procedings of the Forteenth Systems Administration Conference (LISA-00)*, pages 291–304.

Jacobson, V., Leres, C., and MacCanne, S. tcpdump. http://ftp.ee.lbl.gov.

Klyachkin, A. New Netflow Collector. http://sourceforge.net/projects/nnfc.

Ligocki, N., Gomes, C. L., and Hara, C. (2008). A Flexible Network Monitoring Tool based on a Data Stream Management System. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'2008)*.

Munz, G. and Carle, G. (2008). Distributed Network Analysis Using TOPAS and Wireshark. *Network Operations and Management Symposium Workshops, 2008.*, pages 161–164.

Ntop Team (2007). NTop. http://www.ntop.org.

Phaal, P. and Panchen, S. (2007). Packet Sampling Basics.

Phaal, P., Panchen, S., and McKee, N. (2001). RFC 3176: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks.

Plagemann, T., Goebel, V., Bergamini, A., Tolu, G., Urvoy-Keller, G., and Biersack, E. W. (2004). Using Data Stream Management Systems for Traffic Analysis - A Case Study. *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM'04)*, pages 215–226.

Plonka, D. Flowscan. http://www.caida.org/tools/utilities/flowscan/.

Stallings, W. (1996). *SNMP, SNMPv2 and RMON - Pratical Network Management*. Addison-Wesley.

Zhou, A., Yan, Y., Gong, X., Chang, J., and Dai, D. (2008). SMART: A System for Online Monitoring Large Volumes of Network Traffic. *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 1576–1579.