Uma Ferramenta de Monitoramento de Redes usando Sistemas Gerenciadores de Streams de Dados

Natascha Petry Ligocki[1], **Carmem Satie Hara**[1]

[1]Departamento de Informática
Universidade Federal do Paraná (UFPR) – Curitiba, PR – Brazil

{ligocki,carmem}@inf.ufpr.br

Abstract. *This paper describes the development of a network monitoring tool using a Data Stream Management System called Borealis. The goal is to provide a generic tool that covers the needs of several network topologies and configurations, considering also different monitoring purposes. Measurements can be defined by the user as queries, which can be done through a graphical user interface or an XML file. This approach provides an easy mechanism to implement, reuse and modify the tool according to the system needs. Moreover, this generic tool has a steep knowledge curve and allows one to obtain the desired information quickly.*

Resumo. *Este artigo descreve o desenvolvimento de uma ferramenta genérica de monitoramento de redes utilizando um Sistema Gerenciador de Streams de Dados chamado Borealis. O objetivo da ferramenta é atender às necessidades das diferentes configurações e topologias de rede, considerando também diversos objetivos de monitoramento. As medições são definidas pelo usuário através de consultas, as quais podem ser feitas utilizando uma interface gráfica ou um arquivo XML. Esta solução provê um mecanismo de fácil implementação, reuso e modificação da ferramenta de acordo com as necessidades do sistema. Além disso, ela possui uma rápida curva de aprendizagem, além de permitir que se obtenham resultados rapidamente.*

1. Introdução

Com a popularização da Internet, problemas na rede são bastante freqüentes nas companhias hoje em dia. Os usuários reclamam de diversos problemas, tais como disponibilidade da rede, acesso lento durante os horários de pico, problemas de *download* e de acessos em geral. Mesmo os administradores de rede mais experientes precisam ter uma visão do estado da rede como um todo antes de resolver tais problemas. A melhor forma de obter tais informações é através do monitoramento. Existem várias ferramentas disponíveis para este propósito. Porém, é difícil encontrar uma única ferramenta que atenda a todas as necessidades de uma empresa. Uma das soluções mais freqüentes é o uso de *scripts* implementados especificamente para cada cenário. Mas isto nem sempre é uma tarefa fácil. Além disso, quase sempre estes *scripts* são de difícil reuso e portabilidade.

Nos últimos anos, diversos Sistemas Gerenciadores de Streams de Dados (SGSD) foram propostos na literatura [Cranor et al. 2003, Abadi et al. 2003, Arasu et al. 2003,

Chandrasekaran et al. 2003, Balazinska et al. 2004, Abadi et al. 2005] para prover as funcionalidades dos Sistemas Gerenciadores de Banco de Dados (SGBD) tradicionais sobre fluxos contínuos de dados. Estes fluxos de dados (*streams*) podem ser, por exemplo, os pacotes trafegando em uma rede, ou dados de uma rede de sensores, ou de um sistema de monitoramento de chamadas. A característica principal destes sistemas é o grande volume de dados, o que impossibilita que eles sejam armazenados em sua totalidade para serem processados posteriormente. Assim, os SGSDs são sistemas que, além de outras facilidades, possuem uma linguagem de alto nível para expressar consultas, que são processadas à medida que os dados fluem pelo sistema.

Este artigo descreve o uso do SGSD Borealis [Abadi et al. 2005] para implementar uma ferramenta de monitoramento de redes, chamada *Packet Query Tool* (*PaQueT*). Ao contrário de outras ferramentas existentes para este propósito [SLAC 2007], que possuem um elenco pré-definido de métricas que podem ser geradas, a *PaQueT* é uma ferramenta de propósito geral, que permite que o administrador de uma rede defina as consultas de acordo com as suas necessidades específicas. Para isto, a *PaQueT* captura todos os pacotes de uma rede, particiona-os de acordo com um esquema pré-definido, e direciona esta informação para o SGSD. O administrador pode então utilizar o SGSD para executar as consultas e obter as informações desejadas. Estas consultas são expressas em uma linguagem de alto nível, que possui operações semelhantes à *Structured Query Language* (SQL), a linguagem de consultas padrão dos SGBDs relacionais. Uma vantagem desta abordagem é a facilidade de reuso, o que permite que as soluções sejam facilmente modificadas a fim de aperfeiçoá-las e adaptá-las conforme a necessidade.

Contribuições. As principais contribuições deste artigo são:

- o desenvolvimento da ferramenta *PaQueT*, que além de permitir uma análise detalhada de uma rede, é customizável pelo próprio usuário do sistema sem a necessidade de interferência de um desenvolvedor;
- um estudo experimental para comparar a *PaQueT* com outras ferramentas de monitoramento de redes.

Organização. O restante deste artigo está organizado da seguinte forma. Na Seção 2 são apresentados os SGSDs e trabalhos relacionados. A seção seguinte descreve a arquitetura e as peculiaridades da *PaQueT*. A Seção 4 descreve os experimentos feitos e mostra os resultados obtidos com a *PaQueT* em comparação com as ferramentas *Wireshark*[Cace 2007] e *Ntop*[Deri and Suin 2000], que são sistemas para monitoramento de redes de código aberto. Por fim, na Seção 5 são apresentados alguns trabalhos futuros, que incluem funcionalidades adicionais e melhorias para a ferramenta proposta.

2. Sistemas Gerenciadores de Streams de Dados e Trabalhos Relacionados

Os Sistemas Gerenciadores de Streams de Dados (SGSD) foram propostos para prover as funcionalidades de um Sistema Gerenciador de Banco de Dados (SGBD) sobre fluxos contínuos de dados, fornecendo respostas em tempo real, bem como resultados aproximados. Uma discussão sobre processamento de streams em tempo real pode ser encontrada em [Stonebraker et al. 2005]. A principal diferença entre os SGBDs e os SGSDs consiste em como os dados e as consultas persistem no sistema [Koudas and Srivastava 2003]. O primeiro contém informação estática e consultas dinâmicas, enquanto o último tem o comportamento inverso. Ou seja, os bancos de dados tradicionais normalmente executam consultas diferentes sobre o mesmo conjunto de dados. Já os SGSDs executam as

mesmas consultas sobre dados que chegam ao longo do tempo. Em algumas situações é interessante integrar os dois sistemas. Desta forma é possível armazenar os resultados obtidos com o processamento de streams para consultá-los posteriormente.

Os trabalhos de pesquisa envolvendo SGSDs são recentes, e a maioria dos sistemas desenvolvidos ainda são protótipos. Dentre eles podem ser citados: o Borealis [Abadi et al. 2005], o qual foi baseado em dois outros sistemas desenvolvidos pelo mesmo grupo: o Aurora [Abadi et al. 2003] e o Medusa [Balazinska et al. 2004]; o TelegraphCQ [Arasu et al. 2003], que foi implementado de forma a ser uma extensão do SGBD Postgres; o STREAM [Chandrasekaran et al. 2003], que foi um dos pioneiros nesta área de pesquisa; e por fim, o Gigascope [Cranor et al. 2003], que é um projeto comercial que apresentou resultados significativos no monitoramento de redes, mostrando inclusive vantagens sobre ferramentas como o Netflow [Cisco 2006].

Alguns estudos foram feitos sobre estes protótipos, e os resultados apresentados são encorajadores. Um exemplo é o estudo de caso feito sobre o SGSD TelegraphCQ, descrito em [Plagemann et al. 2004]. O objetivo deste trabalho era comparar as funcionalidades fornecidas por este SGSD com aquelas existentes na T-RAT [Zhang et al. 2002], uma ferramenta para analisar a dinâmica de uma rede. Este estudo serviu de inspiração para a implementação da ferramenta proposta neste artigo. Outros estudos de caso que demonstram a possibilidade de utilização do SGSD Borealis são: um jogo com suporte a múltiplos usuários [Ahmad et al. 2005] e sua utilização em um ambiente com diversas peculiaridades, como uma rede de sensores [Abadi et al. 2004].

O Borealis é o SGSD que foi escolhido para implementar a *PaQueT*. Dentre os SGSDs acima citados, ele é o único sistema distribuído. Além disso, ele possui características próprias e inovadoras tais como *registros de revisão*, *viagem no tempo* e *linhas de controle* [Ahmad et al. 2005]. Tal como em qualquer banco de dados distribuído [Lima et al. 2003], o Borealis também permite integração dos dados e compartilhamento de recursos e também implementa mecanismos de tolerância a falhas, processamento distribuído, escalabilidade, e balanceamento e dispersão de carga [Ahmad et al. 2005]. As características distribuídas do sistema são de extrema importância em termos de desempenho. Outra vantagem disto é que muitas aplicações para as quais os SGSDs foram projetados possuem entrada distribuída, o que facilita a captura dos dados. Uma vez que o Borealis apresenta diversas características desejáveis de uma ferramenta de monitoramento de redes, um dos objetivos deste trabalho é validar sua utilização como ponto de partida para o desenvolvimento deste tipo de aplicação.

3. Uma Ferramenta Genérica para Monitoramento de Redes

Nesta seção é descrita a *PaQueT*, uma ferramenta genérica de monitoramento de redes, implementada utilizando o SGSD Borealis. Ao contrário de outras ferramentas existentes, na *PaQueT* as métricas retornadas pelo sistema são arbitrárias e definidas pelo usuário através de consultas definidas sobre os pacotes que trafegam pela rede. Tais consultas podem ser feitas através de uma ferramenta gráfica ou através de arquivos XML. No restante desta seção são descritos a arquitetura da *PaQueT*, o esquema dos pacotes, a linguagem de consulta e por fim alguns detalhes relevantes da implementação.

Arquitetura. A Figura 1 mostra uma visão geral da arquitetura da *PaQueT*. A ferramenta consiste basicamente de dois módulos: o *IP Tool* e o SGSD Borealis, propriamente

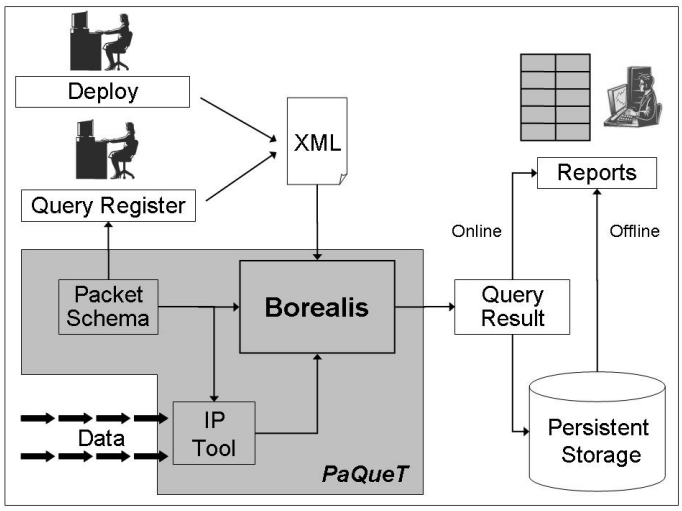

Figura 1. Arquitetura da *PaQueT*.

dito. De forma similar aos SGBDs, o Borealis requer que o esquema dos dados seja previamente definido para que eles possam ser processados. Assim, a *PaQueT* possui um conjunto de esquemas pré-definidos, que descrevem a estrutura dos pacotes que trafegam pela rede. O módulo responsável pela captura dos pacotes e sua decomposição em campos, de acordo com o esquema, é o *IP Tool*. Estes dados já decompostos são então enviados para o Borealis para que as consultas possam ser processadas.

Para utilizar a ferramenta, o usuário do sistema, normalmente o administrador da rede, registra as consultas (*Query Register*) para obter as informações desejadas. Opcionalmente, é possível também especificar um arquivo de distribuição dos recursos (*Deploy*), o qual deve conter informações sobre as responsabilidades de cada nó em um sistema distribuído. Ambas as especificações, de consulta e de recursos, podem ser feitas através de uma interface gráfica, chamada Borgui, que é fornecida juntamente com o Borealis. Elas são então traduzidas para arquivos XML, que são dados como entrada para o SGSD. Os resultados das consultas podem ser armazenados em uma tabela de um banco de dados tradicional (*Persistent Storage*) ou ser usados para gerar relatórios (*Reports*) a partir dos dados obtidos. Desta forma, os relatórios podem ser vistos à medida que os dados vão sendo gerados, ou podem ser obtidos do armazenamento persistente posteriormente. Com o resultado das consultas, o administrador é capaz de fazer diagnósticos sobre a rede e otimizar sua configuração. Outra facilidade fornecida pela *PaQueT* é a possibilidade de disparar eventos de acordo com o resultado de uma consulta, como por exemplo aquelas que detectam anomalias na rede.

O Esquema dos Pacotes. A *PaQueT* define o esquema dos pacotes sobre os quais o usuário pode definir suas consultas. Este esquema foi definido em XML como apresentado na Figura 2. Ele consiste de uma seqüência de elementos para cada campo (`field`) do cabeçalho dos pacotes, onde cada campo possui um nome (`name`), um tipo (`type`), e opcionalmente um tamanho (`size`). Para simplificar, o esquema mostrado contém informações apenas sobre os protocolos da camada de transporte TCP e UDP. No entanto, a *PaQueT* dá suporte também a outros protocolos da camada de rede IPv4. O nome de cada um dos campos foi escolhido de acordo com a definição que pode ser encontrada

em [Sans 2007]. Na seqüência é apresentada a linguagem de consulta do SGSD Borealis para ilustrar como o esquema definido pode ser utilizado para expressar consultas no sistema.

```
<schema name="TuplaPacote">
    <field name="captura"      type="timestamp"/>
    <field name="ether_dhost"  type="string"  size="6"/>
    <field name="ether_shost"  type="string"  size="6"/>
    <field name="ether_type"   type="string"  size="1"/>
    <field name="ip_vhl"       type="string"  size="1"/>
    <field name="ip_tos"       type="string"  size="1"/>
    <field name="ip_len"       type="int"/>
    <field name="ip_id"        type="string"  size="2"/>
    <field name="ip_off"       type="string"  size="2"/>
    <field name="ip_ttl"       type="string"  size="1"/>
    <field name="ip_p"         type="string"  size="1"/>
    <field name="ip_sum"       type="int"/>
    <field name="ip_src"       type="string"  size="4"/>
    <field name="ip_dest"      type="string"  size="4"/>
    <field name="tcp_sport"    type="int"/>
    <field name="tcp_dport"    type="int"/>
    <field name="tcp_seq"      type="long"/>
    <field name="tcp_ack"      type="long"/>
    <field name="tcp_off"      type="int"/>
    <field name="tcp_flags"    type="string"  size="1"/>
    <field name="tcp_win"      type="string"  size="2"/>
    <field name="tcp_sum"      type="int"/>
    <field name="tcp_urp"      type="string"  size="2"/>
    <field name="udp_sport"    type="int"/>
    <field name="udp_dport"    type="int"/>
    <field name="udp_len"      type="int"/>
    <field name="udp_sum"      type="int"/>
</schema>
```

Figura 2. Esquema dos dados de entrada da *PaQueT*.

Uma Interface para Construção de Consultas. O SGSD Borealis possui uma ferramenta chamada *Borealis Graphical User Interface* (Borgui), que é uma interface gráfica para o usuário construir suas consultas. Na Borgui, as consultas são expressas através de diagramas compostos de caixas, que representam os operadores, e flechas, que representam o fluxo da informação. Os operadores existentes na linguagem são baseados nos conceitos da álgebra relacional e do SQL. Estes operadores incluem a seleção, projeção, junção e união, além de operadores de agregação, como contadores, somas, médias, e outros. Novos operadores também podem ser definidos pelo usuário. Para alguns operadores, como os de agregação, é possível definir uma janela de tempo, ou seja, o intervalo que determina a periodicidade em que um novo valor agregado é gerado. A janela de tempo pode ser definida tanto por unidade de tempo, como pela quantidade de dados recebidos.

Para exemplificar como uma consulta é expressa no sistema, considere a Figura 3. A consulta retorna a quantidade de pacotes UDP e TCP que passaram pela interface de rede a cada intervalo de 60 segundos. Para isso, todos os pacotes são capturados pelo operador de `união`, criando um único fluxo de dados que é direcionado para o `filtro` (operador de seleção). Este operador passa para o próximo operador somente os pacotes cujo protocolo (`ip_p`) seja UDP ou TCP. Finalmente, o operador de agregação faz a contagem de quantos pacotes de cada tipo passaram pela interface a cada 60 segundos e retorna estes valores.

Para que uma consulta seja processada pelo Borealis, ela é primeiramente traduzida para um arquivo XML. Assim, uma forma alternativa de expressar uma consulta é através de um arquivo XML diretamente, sem a utilização da linguagem visual. A Figura 4 apresenta a consulta ilustrada na Figura 3 expressa em XML, porém sem o operador de união. Ou seja, ela representa a mesma consulta sobre apenas um ponto de monitoramento, já que o operador de união só precisa ser aplicado para criar um único fluxo de pacotes provenientes de múltiplos pontos.

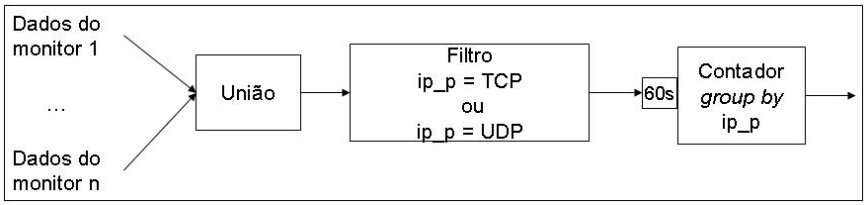

Figura 3. Exemplo de um diagrama de consulta.

Este exemplo de consulta mostra como elas podem ser facilmente construídas pelo usuário do sistema. Além disso, como todos os campos dos pacotes podem ser utilizados nas consultas, a *PaQueT* pode ser utilizada inclusive para o monitoramento de conteúdo.

IP Tool. Para construir a *PaQueT*, além de ser definido o esquema de entrada, foi desenvolvido em C++ o aplicativo *IP Tool*, responsável pela captura dos pacotes e sua quebra de acordo com os respectivos cabeçalhos e campos. No desenvolvimento de uma aplicação Borealis, uma vez definido o esquema de entrada, o sistema gera uma coleção de cabeçalhos de funções que devem ser implementadas para que os dados possam ser processados pelo SGSD. Estas funções são descritas em C++ e sua implementação deve ser feita pelo desenvolvedor da aplicação. Outras linguagens também podem ser usadas, desde que o gerenciamento da comunicação entre os módulos também seja desenvolvido. Para implementar o *IP Tool*, foi utilizada a biblioteca Pcap [Carstens 2002] para a captura e decomposição dos pacotes. Assim, as informações são encapsuladas dentro do esquema definido na Figura 2, compondo o fluxo de entrada do SGSD para o processamento das consultas registradas.

4. Estudo Experimental

Para explorar as funcionalidades disponíveis no Borealis, determinar a precisão dos resultados, e avaliar a carga de trabalho no sistema imposta pela *PaQueT*, foi realizado um estudo experimental. Os resultados obtidos foram comparados com duas ferramentas bastante populares de monitoramento de redes: *Ntop* [Deri and Suin 2000] e *Wireshark* [Cace 2007]. O objetivo da escolha das duas ferramentas foi a de tentar abranger tanto

```
<?xml version="1.0"?>
<!DOCTYPE borealis SYSTEM "/root/Borealis_Stuff/borealis/src/src/borealis.dtd">
<borealis>
    <input  stream="Filtro"          schema="TuplaPacote"/>
    <input  stream="Pacote"          schema="TuplaPacote"/>
    <outputstream="Agregacao"    schema="TuplaAgregacao"/>
    <schema name="TuplaPacote">
        <field name="tempo"        type="int"/>
        <field name="ip_p"         type="string"  size="4"/>
    </schema>
    <schema name="TuplaAgregacao">
        <field name="ip_p"         type="string"size="4"/>
        <field name="tempo"        type="int"/>
        <field name="numPacotes" type="int"/>
    </schema>
    <query name="NumPacotesUdpTcp">
        <box name="Filtro"  type="filter">
            <in stream="Pacote"/>
            <out stream="Filtro"/>
            <parameter name="expression.0"        value="ip_p='tcp' || ip_p='udp'"/>
            <parameter name="pass-on-false-port"  value="0"/>
        </box>
        <box name="Contagem"  type="aggregate">
            <in stream="Filtro"/>
            <out stream="Agregacao"/>
            <parameter name="aggregate-function.0"                value="count()"/>
            <parameter name="aggregate-function-output-name.0" value="numPacotes"/>
            <parameter name="window-size-by"                      value="VALUES"/>
            <parameter name="window-size"                         value="60"/>
            <parameter name="advance"                             value="60"/>
            <parameter name="order-by"                            value="FIELD"/>
            <parameter name="order-on-field"                      value="tempo"/>
            <parameter name="group-by"                            value="ip_p"/>
            <parameter name="independent-window-alignment"    value="1"/>
            <parameter name="drop-empty-outputs"                  value="1"/>
        </box>
    </query>
</borealis>
```

Figura 4. Consulta para contagem de pacotes por protocolo.

ferramentas de análise de protocolos de baixo nível, representadas pelo *Wireshark*, quanto ferramentas com foco nas estatísticas geradas, representadas pelo *Ntop*.

Existem diversas ferramentas de monitoramento de redes disponíveis. Uma lista bastante extensa destes sistemas pode ser encontrada em [SLAC 2007]. O *Wireshark* é uma ferramenta que possibilita verificar o conteúdo dos pacotes de diversos protocolos,

além de permitir a aplicação de filtros e a visualização de estatísticas sobre os dados obtidos. Apesar de dar suporte a dezenas de protocolos, ela não possui muitas opções de sumarização das informações. Como alternativa, os resultados podem ser exportados para outros formatos, podendo ser analisados por outros aplicativos. O *Ntop* é uma ferramenta para analisar o uso de uma rede, de forma similar ao que faz o comando *top* do Unix. Ela possui uma interface bastante amigável, sendo possível visualizar os resultados na *web*. Ela também dá suporte a diversos protocolos e interfaces de rede, e utiliza o conceito de *plugins* para adicionar novas funcionalidades à ferramenta. Vários gráficos mostrando o tráfego na rede podem ser gerados, e eles podem ser customizados pelo usuário, porém dentro de um escopo pré-definido.

Nesta seção são descritos dois experimentos realizados utilizando as ferramentas *PaQueT*, *Wireshark* e *Ntop*. Ambos foram realizados através de uma simulação com uma ferramenta de geração de pacotes randômicos, em um computador com processador Intel Celeron 1.46 GHz e 512 MB de memória RAM. Para realizar o estudo comparativo de carga de trabalho, as ferramentas *PaQueT*, *Wireshark* e *Ntop* foram executadas simultaneamente, sendo que a duração de cada experimento foi de mais de uma hora. Para melhor comparação, o *Wireshark* foi configurado para armazenar os resultados em arquivo e não mostrá-los na interface gráfica, assim como funcionam as demais ferramentas.

Durante os experimentos, o programa *top* do Linux foi utilizado para avaliar o tempo de CPU e o uso de memória física dos processos relativos a cada uma das ferramentas avaliadas. A taxa de atualização configurada foi de 3 segundos. Através destas informações foi possível obter o consumo de memória física e de tempo de CPU de cada um dos processos. Para análise, foram calculados para cada atributo, o pico, representando o valor máximo obtido, e a média, representando a média aritmética dos valores obtidos no período. No caso do uso de memória, visto que não houve diferença significativa entre os valores de pico e da média, os primeiros foram omitidos. Outro dado relevante analisado nos experimentos foi o tamanho dos arquivos de saída. Enquanto o *Ntop* armazena somente os resultados de métricas pré-definidas, o *Wireshark* armazena todos os pacotes e recalcula os resultados cada vez que o arquivo é reaberto. Já a *PaQueT* se preocupa apenas em armazenar as informações desejadas, ou seja, as que foram registradas nas consultas pelo usuário.

Nos experimentos, a *PaQueT* foi configurada para fazer a captura dos pacotes de modo promíscuo e para gerar os resultados em um arquivo. Porém, existem duas variações que poderiam ser utilizadas. A primeira refere-se à forma de captura. Se o modo promíscuo não for suportado, o monitoramento poderia ser feito em cada um dos pontos da rede isoladamente. Tal mudança requer apenas uma alteração na consulta, com a inclusão de um operador de união para capturar os pacotes de todos os pontos monitorados, como apresentado na Figura 3. Outra modificação que poderia ser feita é na forma de apresentação dos resultados. Ao invés de mostrá-los na tela e armazená-los em um arquivo, eles poderiam também ter sido inseridos em um banco de dados tradicional. Esta alteração também requer apenas a adição do operador *table* no final da consulta, o qual recebe como parâmetro comandos SQL.

Os experimentos descritos a seguir foram feitos separadamente para demonstrar a funcionalidade e a flexibilidade da *PaQueT*. No entanto, estas e quaisquer outras consultas poderiam ser registradas e monitoradas simultaneamente através de uma simples mescla

das consultas, ou até mesmo através de registros paralelos.

4.1. Monitoramento do número de pacotes por protocolo

Este experimento é o mesmo daquele descrito no diagrama da Figura 3 e no arquivo XML da Figura 4. O objetivo é fazer a contagem total de pacotes UDP e TCP que passaram pela rede durante um determinado período de tempo. A consulta é formada por apenas dois operadores: um operador de seleção (`Filtro`), o qual descarta todos os pacotes que não forem nem UDP, nem TCP, e outro operador de agregação (`Contagem`). Esse último é o responsável por fazer o agrupamento dos pacotes, de acordo com o seu tipo, para contabilizar o total de pacotes capturados. A janela de tempo da consulta é de 60 segundos, sendo que as janelas que não contêm pacotes dos tipos monitorados são descartadas.

A Tabela 1 mostra os resultados deste experimento com o *Wireshark* e *Ntop*. Neste experimento foram gerados mais de 210 mil pacotes durante um período de pouco mais de uma hora. Dos pacotes analisados, 77% eram TCP e 10% eram pacotes UDP. Os resultados obtidos pela *PaQueT* foram equivalentes àqueles gerados pelo *Wireshark* e pelo *Ntop*.

Tabela 1. Resultados obtidos com a monitoramento do número de pacotes.

Ferramenta	CPU (%) - Média	CPU (%) - Pico	Memória (%)	Arquivo
PaQueT	0,68	1,3	6,17	8,8 KB
Wireshark	1,2	4	5,9	1,8 MB
Ntop	0,15	1	6,65	39,7 MB

Com relação ao desempenho, tanto a *PaQueT*, quanto o *Wireshark* e o *Ntop* apresentaram resultados similares como mostra a Tabela 1. O consumo de memória foi melhor no *Wireshark*, enquanto o menor consumo de tempo de CPU foi do *Ntop*. A *PaQueT* teve os valores intermediários nos dois atributos. Quanto ao tamanho do arquivo gerado, a quantidade de dados gerados pela *PaQueT* é o menor, visto que ele contém somente a sumarização dos resultados desejados como mencionado anteriormente. Já o *Ntop* gera informações que são comumentemente utilizadas e o *Wireshark* armazena os pacotes propriamente ditos.

4.2. Monitoramento da taxa de transmissão por IP

O monitoramento da taxa de transmissão por IP é bastante útil para determinar os principais responsáveis do consumo da banda de uma rede. A consulta é feita de forma similar àquela apresentada na subseção anterior. Ela consiste de um agrupamento dos pacotes por IP do transmissor para fazer a soma do número total de bytes dos pacotes transmitidos. O objetivo é identificar a taxa de *upload* de cada usuário na rede. O mesmo poderia ser feito para obter a taxa de *download*.

Durante o monitoramento, trafegaram pela rede mais de 17 MB de pacotes TCP, representando 57% do tráfego total da rede. A Tabela 2 mostra os resultados obtidos no experimento com o *Wireshark* e o *Ntop*. Assim como no experimento descrito na subseção 4.1, em termos de funcionalidade as três ferramentas obtiveram o mesmo resultado. Também em termos de desempenho os resultados foram os mesmos com pequenas variações, visto que a forma de obtenção dos dados possuem complexidades equivalentes.

O tamanho dos arquivos possuem a mesma ordem de grandeza do experimento anterior, visto que são um reflexo das características de obtenção de dados de cada uma das ferramentas.

Tabela 2. Resultados obtidos com o monitoramento da taxa de transmissão.

Ferramenta	CPU (%) - Média	CPU (%) - Pico	Memória (%)	Arquivo
PaQueT	0,39	1,7	6,19	8,5 KB
Wireshark	1,17	2	6	32,4 MB
Ntop	0,18	2	6,53	1,8 MB

Os resultados apresentados pelos dois experimentos foram bastante positivos e indicam que, apesar do SGSD Borealis ser um sistema de propósito geral para o processamento de streams, o seu impacto no sistema, tanto em termos de memória como de utilização de CPU é bastante similar ao apresentado pelo *Wireshark* e *Ntop*. No entanto, a flexibilidade de customização da *PaQueT*, sem necessidade de conhecimentos mais avançados se torna uma grande vantagem. Portanto, os resultados indicam que a *PaQueT* é uma alternativa interessante para customização do monitoramento de redes. Um ponto a ser ressaltado é o tamanho do arquivo de saída. Ao contrário das ferramentas consideradas no estudo experimental, que armazenam todas as informações necessárias para gerar todas as suas métricas, que são pré-definidas, na *PaQueT* a filtragem é realizada durante o monitoramento, baseada nas consultas registradas no sistema. Isto diminui significativamente o volume de dados gerados e conseqüentemente a quantidade de armazenamento de informação. Vale ressaltar que os valores obtidos pela *PaQueT* foram precisos, o que demonstra a confiabilidade do sistema. Além disso, o *Wireshark* e o *Ntop* são freqüentemente utilizados para propósitos distintos na prática. Enquanto o primeiro faz o processamento dos pacotes propriamente ditos, o segundo é utilizado para a geração de estatísticas de alto nível. Devido a sua flexibilidade, a *PaQueT* pode ser utilizada para os dois propósitos, e os resultados do estudo experimental demonstram a sua eficácia.

5. Conclusão

Através dos estudos experimentais, foi possível validar a *PaQueT* como uma ferramenta genérica de monitoramento de redes. Por ser uma ferramenta de alto nível, é possível construir diferentes consultas sem a necessidade da ajuda de desenvolvedores para a implementação de programas específicos para cada cenário. Esta abordagem permite fácil reutilização e adaptação de consultas previamente existentes. Além disso, ela permite que apenas os dados sumarizados sejam armazenados, caso se deseje consultá-los posteriormente.

O estudo experimental demonstrou a eficácia da ferramenta proposta, tanto em termos de funcionalidade, como em desempenho. Além disso, uma das grandes vantagens da *PaQueT* é evitar o desperdício de armazenamento. Outro ponto importante é o fato de todas as consultas do Borealis passarem por um processo de otimização como nos bancos de dados tradicionais, permitindo melhorar seu desempenho de forma proporcional ao número de consultas registradas. Ou seja, quanto maior o número de consultas, melhor será a sua otimização, visto que alguns resultados parciais de uma consulta podem ser reaproveitados nas demais.

Na continuidade deste projeto, pretende-se aplicar os mesmos experimentos apresentados, não em um ambiente de simulação, mas em um ambiente real como a rede de uma companhia. Além disso, as características distribuídas da ferramenta também podem ser testadas, tais como o balanceamento de carga e a tolerância a falhas, podendo melhorar ainda mais o desempenho obtido pela *PaQueT*.

Uma análise também importante ainda a ser feita é identificar o ponto em que a *PaQueT* não consegue acompanhar a taxa de chegada dos pacotes. A captura depende da capacidade de processamento do sistema e da ferramenta, sendo que em momentos de pico é necessário descartar alguns dos dados de entrada. Para otimizar esta limitação, o Borealis utiliza um dispersor de carga configurável, permitindo a selecão dos pacotes a serem descartados através da aplicação de regras simples. Desta forma é possível gerar, mesmo nos momentos de pico, resultados aproximados sem afetar a confiabiliade e a importância da informação obtida.

Outros trabalhos futuros incluem a implementação de um analisador de resultados, bem como uma ferramenta para análise dos dados dos pacotes. Desta forma será possível gerar estatísticas sobre os sites acessados, facilitando por exemplo uma melhor configuração de *firewalls*. Por fim, o suporte a outros protocolos da camada de transporte ou de rede pode ser facilmente adicionados através da definição das estruturas dos pacotes e extensão do esquema de entrada.

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. In *Proceedings of the 2nd Conference on Classless Inter-Domain Routing (CIDR'05)*, pages 277–289.

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E. F., Hatoun, M., Hwang, J.-H., Maskey, A., Rasin, A., Singer, A., Stonebraker, M., Tatbul, N., Xing, Y., Yan, R., and Zdonik, S. (2003). Aurora: A data stream management system. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03)*.

Abadi, D. J., Lindner, W., Madden, S., and Schuler, J. (2004). An integration framework for sensor networks and data stream management systems. In *Proceedings of 30th International Conference on Very Large Data Bases (VLDB'04)*, pages 1361–1364.

Ahmad, Y., Berg, B., Çetintemel, U., Humphrey, M., Hwang, J.-H., Jhingran, A., Maskey, A., Papaemmanouil, O., Rasin, A., Tatbul, N., Xing, W., Xing, Y., and Zdonik, S. (2005). Distributed operation in the borealis stream processing engine. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD'05)*, pages 882–884.

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).

Balazinska, M., Balakrishnan, H., and Stonebraker, M. (2004). Load management and high availability in the medusa distributed stream processing system. In *Proceedings*

of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD'04), pages 929–930.

Cace (2007). *Wireshark*. Cace Technologies.

Carstens, T. (2002). Programming with pcap. `www.tcpdump.org/pcap.htm`.

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)*, pages 269–280.

Cisco (2006). *Introduction to Cisco IOS Netflow - A Technical Overview*. Cisco Systems Inc.

Cranor, C., Johnson, T., Spatscheck, O., and Shkapenyuk, V. (2003). The gigascope stream database. *IEEE Data Engineering Bulletin*, 26(1):27–32.

Deri, L. and Suin, S. (2000). Effective traffic measurement using ntop. *IEEE Communications Magazine*, 38(5):138–143.

Koudas, N. and Srivastava, D. (2003). Data stream query processing: A tutorial. In *Proceedings of 29th International Conference on Very Large Data Bases (VLDB'03)*, pages 1149–1149.

Lima, A. A. B., Matoso, M. L. Q., and Esperança, C. (2003). Efficient processing of heavy-weight queries in database clusters. Technical Report 001, UFRJ.

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. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement (PAM'04)*, pages 215–226.

Sans (2007). *TCP/IP and tcpdump. Pocket reference guide.* SANS Institute.

SLAC (2007). Network monitoring tools. `www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html`.

Stonebraker, M., Çetintemel, U., and Zdonik, S. (2005). The 8 requirements of real-time stream processing. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*.

Zhang, Y., Breslau, L., Paxson, V., and Shenker, S. (2002). On the characteristics and origins of internet flow rates. In *Proceedings of Conference on Applications, technologies, architectures, and protocols for computer communications (ACM SIGCOMM'02)*, pages 309 – 322.