NS-*Facilities* - Uma ferramenta de apoio ao desenvolvimento de simulações *

D. M. Oliveira[1], **R. dos S. Cruz**[1], **R. J. P. de B. Salgueiro**[1]

[1]Departamento de Computação – Universidade Federal de Sergipe
Avenida Marechal Rondon, s/n – São Cristóvão - SE - 49100-000

danilomo@dcomp.ufs.br, ruironaldi@gmail.com, salgueiro@ufs.br,

Abstract. *This paper presents a simulation support framework based on NS-2. NS- 2 is the most widely used computer network simulator in academic area. One of the disadvantages of the NS-2 is not offering an infrastructure for planning, configuration and simulation result extraction. The NS2 Facilities framework automates the tasks of configuration, implementation and result extraction in simulation experiments. The Java language and database HSQLDB were used for implementation issues. Tests with the framework indicate greater agility in setting up simulations, and results recovery flexibility.*

Keywords: modeling of systems, evaluation of performance, simulation

Resumo. *Este trabalho apresenta um framework de apoio ao desenvolvimento de simulações para o NS-2. O NS-2 é o software de simulação de redes de computadores mais utilizado na área acadêmica. Uma das desvantagens do NS-2 é não oferecer uma infraestrutura para o planejamento, configuração e extração dos resultados de simulações. O framework NS2 Facilities automatiza as tarefas de configuração, execução e extração dos resultados em experimentos. Para a implementação foram utilizados a linguagem Java e o banco de dados HSQLDB. Os testes realizados com o framework indicam uma maior agilidade na configuração de simulações, e flexibilidade na recuperação dos resultados.*

Palavras-chave: modelagem de sistemas, avaliação de desempenho, simulação

*A ferramenta foi desenvolvida com bolsa da Pyxis Tecnologia/IEL

1. Introdução

Diversos sistemas de Tecnologia de Informação e Comunicação (TIC) desenvolvidos não levam em consideração aspectos de performance do sistema, ou apenas fazem isso nos estágios finais de desenvolvimento. Isso frequentemente leva a frustrações para os gerentes e prejuízos para as empresas que fazem uso desses sistemas [Menasce et al. 2004]. Em estágios primários de desenvolvimento, a performance de um sistema pode ser predita através de técnicas de modelagem. Simulação é uma técnica de modelagem onde os modelos são construídos a partir de programas de computadores que emulam o comportamento dinâmico do sistema. Alguns autores, como [Issariyakul and Hossain 2009], consideram que simulações normalmente são mais precisas e detalhadas que modelos analíticos, desde que cada detalhe do sistema pode ser adicionado ao modelo de simulação para melhor descrever o sistema atual. Às vezes, o sistema é tão complexo que modelagem analítica não é adequada. Quando o sistema real não está disponível para testes, como frequentemente é o caso, modelos de simulação são usados para predizer a performance ou comparar diferentes alternativas.

Apesar de seus benefícios para a modelagem de sistemas, simulações possuem algumas desvantagens, sendo que a mais relevante é o tempo gasto. Na criação de um modelo de simulação, muito tempo é gasto com a verificação e validação do modelo. A verificação do modelo afirma que o modelo está corretamente implementado e a validação implica em garantir que o modelo implementado representa corretamente o sistema real. O uso de um pacote de simulação pode diminuir o tempo gasto com a verificação e a validação, porém, muito tempo ainda é gasto com a execução da simulação. Programas de simulação precisam ser executados durante um longo tempo para garantir que um certo grau de confiança seja alcançado. Além disso, geralmente é necessário haver várias replicações de uma execução em particular para diminuir os impactos da aleatoriedade. Cada replicação consiste na execução da simulação com os mesmos parâmetros de entrada, apenas mudando o valor da semente (valor dado como entrada para o gerador de números aleatórios que determina a sequência de números gerada).

Construir um modelo de simulação a partir do zero com uma linguagem de propósito geral pode ser muito custoso, desta forma, o uso de linguagem ou ferramenta de simulação é preferido. O *Network Simulator* (versão 2) [NS-2 2008], mais conhecido como ns-2, é o simulador de rede mais usado entre os simuladores de código aberto [Issariyakul and Hossain 2009]. Devido sua natureza modular e de código aberto, o ns-2 ganhou grande aceitação na área de pesquisa e muitas contribuições para o seu código. Com o ns-2 é possível estudar vários protocolos de rede existentes sobre redes cabeadas e sem fio e testar novos protocolos e aplicações. Apesar de ser uma boa ferramenta para simulação de redes, o ns-2 não fornece suporte para o planejamento e configuração de simulações e análise dos resultados, tudo isso é deixado a cargo do desenvolvedor.

Neste trabalho, é proposto o NS-*Facilities*, um *framework* para simulações no ns-2 que permite a organização, configuração, execução e extração de resultados de forma automatizada. O NS-*Facilities* é útil ao desenvolvedor de simulações para o ns-2 porque ajuda a reduzir o tempo gasto com simulações. Os resultados das simulações são armazenados em um banco de dados relacional e podem ser consultados através da linguagem SQL, o que oferece uma grande flexibilidade na recuperação desses resultados. Os testes confirmaram uma maior agilidade no desenvolvimento e configuração de simulações.

Este trabalho está organizado da seguinte forma. Na seção 2, são discutidos os aspectos relevantes de algumas ferramentas construídas com o propósito de automatizar tarefas de desenvolvimento de simulações. Na seção 3, é descrito o *framework* proposto: sua arquitetura, a API de simulação, a ferramenta de execução em batch e o módulo de persistência dos resultados. Na seção 4, é apresentado o estudo de caso usado para avaliação do *framework*. A seção 5 é dedicada à conclusão deste trabalho.

2. Trabalhos relacionados

O foco da equipe de desenvolvimento do ns-2 é o simulador de rede e os modelos de simulação para diversos protocolos. Como o ns-2 não oferece suporte ao desenvolvimento, configuração e análise dos resultados, muitos desenvolvedores constróem ferramentas para automatizar algumas dessas tarefas.

Dentre as ferramentas dispostas na literatura, destacam-se as que oferecem soluções para cenários particulares de uso, geralmente redes sem fio. Em [Kurkowski et al. 2005] é proposto o iNSpect, que é implementado com o intuito de avaliar a precisão do modelo que representa a rede sem fio (a topologia), validar novas versões do simulador ns-2 e analisar resultados das simulações. A ferramenta nsclick [Neufeld et al. 2004] facilita a construção e teste de protocolos para redes *Ad hoc*, visando a redução de erros na implementação de um mesmo protocolo desenvolvido em diferentes plataformas, aumentando assim sua portabilidade.

O ns-2 é bastante utilizado no apoio ao ensino de redes computadores. Em [Welzl et al. 2006], é mostrado que muitos alunos de graduação não gostam do ns-2 por causa da escrita de longos *scripts* de simulação. Neste trabalho, é proposta uma interface de desenvolvimento de simulações no ns-2 que permite o uso do mouse para criar as topologias de rede e depois gerar scripts TCL automaticamente, permitindo a utilização de agentes e protocolos.

Uma problemática tratada por alguns autores é com relação ao pós-processamento dos arquivos de *trace*. A ferramenta JTrana [Qian and Fang 2008] foi criada para a análise dos resultados de simulações e usa um banco de dados MySQL. Ela mostra informações sobre o estado geral da rede, nível de energia dos nós e estatísticas dos pacotes. JTrana trabalha com o formato antigo e o novo de *trace* do ns-2, assim como o formato para redes sem fio, gravando um registro na tabela para cada linha do arquivo de trace. O uso de um banco de dados relacional como o MySQL para guardar grandes volumes de dados do arquivo de trace implica em severas restrições de desempenho no JTrana. Uma melhor alternativa é o uso de um *data warehouse*, como no o que é usado pelo *framework* XAV [Ben-El-Kezadri et al. 2008].

Outra solução que facilita a organização dos resultados para posterior análise é o CostGlue [Savić et al. 2006], que é um repositório de dados para diferentes tipos de ferramentas de simulação. No seu núcleo possui uma aplicação, o CoreGlue, que atua convertendo formatos de saída de simulações diferentes em diferentes formatos de entrada e disponibilizando os dados da simulação em uma interface web a fim facilitar o acesso remoto à ferramenta.

Também existem trabalhos que oferecem uma infraestrutura de apoio em todas as fases do desenvolvimento de simulação. ANSWER (*Automated NS-2 Workflow ManagER*) [Andreozzi et al. 2009] é uma ferramenta de automação para o ns-2 projetada

para simulações em grande escala. O *workflow* de simulação é definido como o processo completo de desenvolvimento, que vai desde a definição dos objetivos até a geração de gráfico dos resultados. Outro trabalho similar é o *SwanTools* [Perrone et al. 2008], que foi projetado para o simulador de redes SWAN. O *SwanTools* possui a vantagem sobre o ANSWER de armazenar os resultados em banco de dados, porém a ferramenta para qual ele foi feito é menos usada do que o ns-2.

O NS2 *Facilities*, aqui proposto, está mais próximo da ferramenta ANSWER, que oferece suporte para o todo os estágios de simulações (com exceção da geração de gráficos). Uma vantagem do NS2 *Facilities* sobre ANWSER é o uso de um banco de dados relacional, que permite maior flexibilidade na recuperação dos resultados através da linguagem SQL. O uso de banco de dados para a recuperação dos resultados de simulações se mostrou eficiente nas ferramentas JTrana e XAV, porém estes armazenam os traces inteiros, exigindo grande espaço em disco. O módulo de persistência de resultados do NS2 *Facilities* (seção 3.3) grava os resultados de simulações, portanto, ocupa menos espaço em disco.

3. *Framework* de simulação

Por ser o simulador de redes mais utilizado na área acadêmica e de pesquisa, muitos modelos de simulação são desenvolvidos para esta ferramenta. Desta forma, ao se modelar algum cenário de rede de computadores com o uso de simulações, haverá uma boa chance de já existir algum modelo desenvolvido e disponível. Por causa disto, o ns-2 tem sido a primeira escolha como ferramenta de simulação de redes por muitos desenvolvedores.

Apesar de ser uma boa ferramenta de simulação de redes, e possuir muitos modelos prontos para diversos cenários de uso, o ns-2 não oferece infraestrutura para auxiliar o desenvolvimento de simulações. Várias tarefas no desenvolvimento de simulações podem ser automatizadas, como a configuração das simulações a serem executadas, a exploração do espaço de parâmetros do modelo, a extração e análise dos resultados, geração de gráficos, etc.

Com a utilização do ns-2, todas estas atividades são deixadas a cargo do desenvolvedor. Geralmente, para a automação destas tarefas são criados *scripts* em linguagens como *Perl*, AWK, *bash*, *Python*, etc. Além disso, não há um padrão a ser seguido para o desenvolvimento de simulações, assim, cada desenvolvedor emprega seu próprio modo *ad-hoc* para desenvolver suas simulações.

O *framework* proposto por este trabalho oferece uma API de simulação, configuração de execução e análise de resultados. Também é oferecida uma ferramenta de execução em *batch*, que recebe um ou mais modelos de simulação escritos em OTcl, um arquivo de configuração, e a partir da combinação dos parâmetros descritos no arquivo de configuração ele executa várias simulações, com repetições ou não. Após o término das simulações todas as métricas de desempenho coletadas são gravadas em um banco de dados pelo módulo de persistência dos resultados.

A arquitetura do NS2 *Facilities* é mostrada na figura 1.

- Na camada 1, está situado o ns-2, que engloba o simulador de rede e os *scripts* de simulação;

- A camada 2 é onde se situa a API, que modela os conceitos de simulação e faz a associação destes com o ns-2;
- Na camada 3 estão situados os módulos que fazem uso da API: o módulo de execução em *batch*, o módulo de persistência e módulos de *front-end* para configuração de simulações;
- Na última camada estão os módulos que fazem uso do módulo de persistência, que são módulos para geração de relatórios e gráficos sobre os resultados de simulação.

A ferramenta de execução em *batch* e *front-ends* de configuração de simulações são responsáveis pela configuração de um estudo de simulação. Através destes módulos os usuários definem os modelos utilizados, os parâmetros dos modelos e tempo de simulação. Esta camada faz uso da camada da API, instanciando objetos das classes definidas (seção 3.1) que modelam uma simulação em particular. Uma simulação no NS2 *Facilities* é composta por objetos pertencentes às classes da API. Cada instância de execução é representada por um objeto que está associado a um modelo descrito por um *script* OTcl. A API é reponsável por executar o ns-2 para realizar a execução de cada instância de simulação que compõe um estudo. Após o término de cada execução, a API faz a extração dos resultados da simulação e grava-os em arquivos separados.

O módulo de persistência faz o armazenamento em banco de dados de todos os objetos e seus atributos: parâmetros, datas de execução e métricas de performance coletadas nos arquivos de saída. Fazendo o uso do módulo de persistência, estão os módulos para geração de relatórios e gráficos que podem ser desenvolvidos.

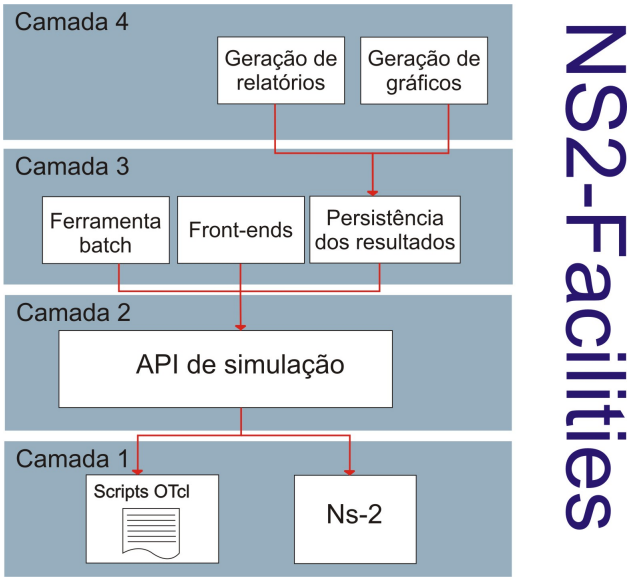

Figura 1. Arquitetura do *framework* de simulação

As subseções a seguir detalham os módulos do *framework*: a API de simulação, a ferramenta de execução em *batch* e a persistência dos estados da simulação.

3.1. API de simulação

A API é a parte central do *framework*. Ela modela os conceitos para a organização da simulação associando-os ao ns-2 [Varga and Hornig 2008]. A figura 2 exibe o modelo

conceitual adotado. Foram definidas seis classes descritas a seguir:

- **Modelo** – É a parte invariante da simulação, que representa algum sistema do mundo real. Na nossa API, um modelo é representado por um *script* TCL (topologia + comportamento);
- **Estudo** – Um estudo é uma série de experimentos para estudar algum fenômeno em um ou mais modelos;
- **Experimento** – Um experimento é a exploração do espaço de parâmetros de um *modelo* e consiste em várias mensurações;
- **Mensuração** – É um conjunto de execuções no mesmo modelo, com os mesmos parâmetros. Pode consistir de várias *replicações*;
- **Replicação** – Para maior confiabilidade, uma mensuração é executada algum número de vezes, com diferentes sementes para os geradores de números aleatórios. Cada execução dessa é denominada replicação. No final, é tomada a média dos resultados das replicações de uma mensuração.
- **Execução** – Uma instância de execução de uma simulação, em um dado horário e em um determinado computador;

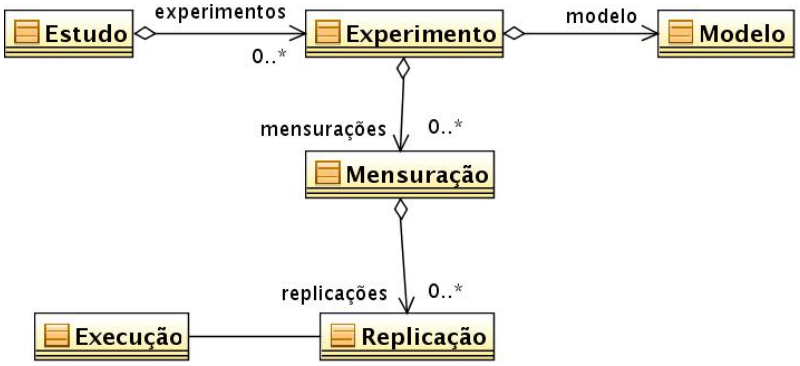

Figura 2. Modelo conceitual da API de simulação

A API foi implementada na linguagem Java e utiliza orientação a objetos para modelar os conceitos citados descritos anteriormente.

Para exemplificar, considere um cenário de simulação que possui um modelo com um certo número de nós servidores e clientes, que são os parâmetros de entrada do modelo. Admita também que é feito um experimento com esse modelo no qual os parâmetros de entradas sejam dados por:

```
number_clients = {10, 20, 30, 40};
number_servers = {1, 2, 3};
replications = 3;
```

Para o exemplo considerado, portanto, será feita uma mensuração para cada elemento do produto cartesiano dos dois conjuntos e cada mensuração será repetida três vezes, com diferentes sementes. Assim, serão feitas $4.3.3 = 36$ execuções.

3.2. Ferramenta de execução em *batch*

A ferramenta de execução em *batch* oferece facilidades para configuração de simulações no ns-2. Este módulo faz uso de um arquivo de configuração (.ini), no qual são declarados

os parâmetros do modelo que foram deixados em abertos. Para a leitura de arquivos .ini, foi utilizada a biblioteca Java inij4 [Szkiba 2010].

Cada parâmetro declarado no arquivo de configuração é atribuído no início do *script* de simulação, portanto pode ser redefinido depois, o que anula a atribuição feita no arquivo .ini. Os parâmetros podem tomar lista de valores, que são declaradas de duas formas:

- Com a sintaxe: <Valor inicial> .. <Valor final> step <Incremento>. Por exemplo numero_nos = 10 .. 100 step 10, que equivale a lista 10, 20, 30, ..., 100;
- Valores separados por vírgulas. Por exemplo numero_nos = 10, 15, 20, 25, 30, 40.

Um arquivo de configuração descreve um estudo. Arquivos de configuração são divididos em seções, no qual cada seção descreve um experimento. Um estudo é descrito da seguinte forma:

```
[<Nome do script TCL (modelo)>]
<parametro> = <valor ou lista de valores>
<parametro> = <valor ou lista de valores>
<parametro> = <valor ou lista de valores> ...
```

Se o nome da seção, que descreve o nome do *script* tcl que representa o modelo, for "*all*", a atribuição de parâmetros da seção será feita para todos os experimentos. Uma atribuição de um parâmetro de mesmo nome anulará a atribuição feita para a seção *all*.

3.3. Persistência dos resultados da simulação

Uma vez executadas todas as simulações previstas em um estudo, o próximo passo é extrair os resultados das execuções. Não existe nenhuma facilidade para esta atividade no ns-2, o desenvolvedor é forçado a escrever *scripts* para a realizar a tarefa de extração, sumarização e geração de gráficos. Portanto, o passo seguinte é construir um módulo de extração dos resultados das simulações de maneira organizada afim de permitir fácil recuperação dos resultados.

A solução solução adotada no NS2 *Facilities* para a extração dos resultados é o uso de banco de dados. O modelo conceitual descrito na figura 2 é mapeado em tabelas para que os resultados de um estudo sejam armazenados. É utilizado o HSQLDB [Toussi 2010], que é um banco de dados leve disponível para a linguagem Java. O uso da linguagem SQL possibilita grande flexibilidade para a recuperação dos resultados.

A adição de tabelas novas, além das tabelas de cada classe descrita anteriormente, foi necessária. A tabela *Parameter* armazena os nomes e valores atribuídos de parâmetros associados a uma simulação. Com essa tabela, o usuário realiza consultas de resultados para atribuições de parâmetros definidas na consulta. As outras duas tabelas representam dados extraídos de uma execução em particular: (a) *Scalar*, para dados escalares, (b) *Vector* para listas de valores.

Um valor escalar é uma atribuição de algum valor a um nome de variável, por exemplo, *atraso-medio = 0.5s*. Um vetor é uma lista de valores associada a um nome de variável. Por exemplo, um nó pode ser monitorado em um intervalo de tempo pré-definido e a vazão de saída em cada tempo pode ser medida e armazenada em um vetor.

Em uma execução particular, são gerados três arquivos: (a) arquivo-trace, (b) arquivo de valores escalares, (c) arquivo de vetores. O arquivo de *trace* é o gerado pelo simulador, que mantém o registro de todos os eventos acontecidos na simulação. O arquivo de valores escalares mantém associações do tipo *chave* = *valor* representando valores escalares relacionados à execução. Esse arquivo pode ser escrito durante a simulação, ou depois de um pós-processamento do arquivo *trace*. O arquivo de vetores armazena dados em forma de lista de valores. O módulo de persistência faz a leitura desses arquivos e grava os valores escalares e vetores em suas respectivas tabelas.

4. Estudo de caso

Para testar a ferramenta foi utilizado o estudo de caso desenvolvido com base nos interesses do trabalho apresentado em [Júnior et al. 2009], o qual avalia o impacto do tráfego de aplicações de voz e vídeo em redes WiMAX. O cenário de simulação utilizado é constituído de uma rede WiMAX com uma estação base, duas estações assinantes e um nó cabeado conectado à estação base. As duas estações assinantes enviam dados para o nó cabeado: uma faz um *streaming* de vídeo, enquanto a outra sobrecarrega a rede. Neste cenário, o parâmetro que é alterado é o valor da taxa de transmissão da estação que gera interferência. Este cenário tem como objetivo avaliar o mecanismo de provisão de QoS para aplicações prioritárias (a aplicação de vídeo) com a interferência de aplicações menos prioritárias (o gerador de interferência).

O arquivo de configuração para o estudo é descrito a seguir:

```
[all]
replications = 4
stop = 100
[simulacao.tcl]
TaxaTrans = 10, 20, 30, 40
```

A seção *all* indica os parâmetros que são definidos para todas as seções do arquivo. O parâmetro *replications* indica quantas replicações são feitas para cada mensuração. O parâmetro *stop* define o tempo de simulação (em segundos) de cada simulação. O arquivo define um estudo, que é descrito pelo modelo de simulação implementado pelo *script* "simulação.tcl" e a exploração do espaço de parâmetros dos parâmetros deixados em aberto do script.

Foi desenvolvido um *front-end* para recuperação de resultados, mostrado na figura 3. Este *front-end* permite a atribuição de valores que são dados como entrada a uma consulta. Após submeter a consulta, é exibida uma nova janela com uma tabela de resultados, mostrada na figura 4. Foi utilizada a função de agregação *avg* do SQL para computar a média dos valores das replicações de uma dada simulação. Estes resultados são armazenados pelo módulo de persistência e podem ser recuperados através de consultas SQL.

Um exemplo de consulta SQL é mostrado na figura 5, que mostra uma lista das métricas de desempenho calculadas e armazenadas como valores escalares. Esta consulta retorna os resultados para as mensurações que possuirem o parâmetro "Taxa de transferência" definidos com o valor 10.

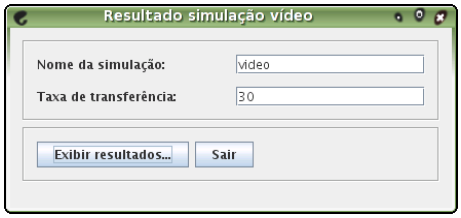

Figura 3. *Front-end* para consulta

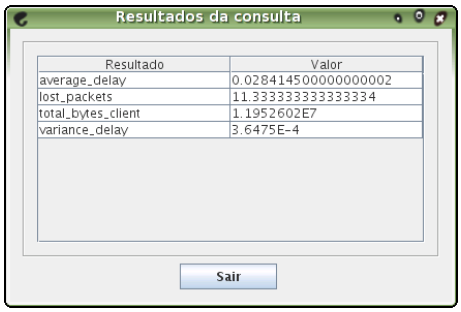

Figura 4. Resultados da consulta

5. Conclusão

Este trabalho propõe um *framework* de apoio a simulação de redes de computadores para a ferramenta ns-2. O *framework* oferece uma ferramenta que recebe um arquivo de configuração que descreve todos os experimentos a serem realizados em um modelo e todos os parâmetros a serem explorados. Depois da execução, todas as simulações são executadas e os resultados são armazenados em banco de dados, podendo ser manipulados através de SQL.

A ferramenta proposta introduz facilidades para o usuários do ns-2. Além de automatizar várias etapas na utilização de simulações ele induz o usuário a escrever simulações para o ns-2 de forma organizada, o que antes era feito de forma *ad-hoc* e variava de usuário para usuário. A criação e elaboração de gráficos estatísticos baseados em especificações do usuários e a criação de *front-ends* são facilidades do *framework* a serem implementadas.

```
select id_measurement, dsc_scalar, avg(value_scalar)

from parameter left join scalar
 on parameter.id_study = scalar.id_study and
 parameter.id_experiment = scalar.id_experiment and
 parameter.id_measurement = scalar.id_measurement

where id_study = 1 and dsc_parameter = 'TaxaTrans'
 and value_parameter = '10'
 group by dsc_scalar, id_measurement
 order by id_measurement
```

Figura 5. Consulta SQL para recuperação dos resultados

Referências

Andreozzi, M. M., Stea, G., and Vallati, C. (2009). A framework for large-scale simulations and output result analysis with ns-2. In *Simutools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, pages 1–7.

Ben-El-Kezadri, R., Kamoun, F., and Pujolle, G. (2008). Xav: a fast and flexible tracing framework for network simulation. In *MSWiM '08: Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 47–53, New York, NY, USA. ACM.

Issariyakul, T. and Hossain, E. (2009). *Introduction to Network Simulator NS2*. Springer.

Júnior, R. d. S. M., Freire, T. F., Silva, A. S. F. d., Oliveira, D. M., Salgueiro, R. J. P. d. B., and Salgueiro, E. M. (2009). Avaliação de desempenho de tráfego de vídeo em redes wimax. In *Congresso Internacional de Computación y Telecomunicaciones*.

Kurkowski, S., Camp, T., Mushell, N., and Colagrosso, M. (2005). A visualization and analysis tool for ns-2 wireless simulations: inspect. In *MASCOTS '05: Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 503–506, Washington, DC, USA. IEEE Computer Society.

Menasce, D. A., Almeida, V. A. F., and Dowdy, L. W. (2004). *Performance by Design: Computer Capacity Planning by Example: Computer Capacity Planning*. Prentice Hall International.

Neufeld, M., Jain, A., and Grunwald, D. (2004). Network protocol development with nsclick. *Wirel. Netw.*, 10(5):569–581.

NS-2 (2008). The network simulator - ns-2. Disponível em [http://nsnam.isi.edu/nsnam].

Perrone, L. F., Kenna, C. J., and Ward, B. C. (2008). Enhancing the credibility of wireless network simulations with experiment automation. In *WIMOB '08: Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication*, pages 631–637, Washington, DC, USA. IEEE Computer Society.

Qian, H. and Fang, W. (2008). Jtrana: A java-based ns2 wireless trace analyzer. Disponível em [http://sites.google.com/site/ns2trana/].

Savić, D., Pustišek, M., and Potortì, F. (2006). A tool for packaging and exchanging simulation results. In *valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 60. ACM.

Szkiba, I. (2010). The java .ini library. Disponível em [http://ini4j.sourceforge.net/].

Toussi, F. (2010). Hsqldb - 100% java database. Disponível em [http://hsqldb.org/].

Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pages 1–10.

Welzl, M., Ali, M., and Hessler, S. (2006). Network Simulation By Mouse (NSBM): A GUI Approach for Teaching Computer Networks with the ns Simulator. In *Proceedings of the International Conference on Interactive Computer Aided Learning (ICL 2006)*, pages 26–28. Citeseer.