Análise de Desempenho de Estimadores de Largura de Banda Disponível para Utilização em Grades

Daniel M. Batista[1], **Luciano J. Chaves**[1], **Nelson L. S. da Fonseca**[1], **Artur Ziviani**[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Campinas – SP – Brasil

[2]Laboratório Nacional de Computação Científica (LNCC)
Petrópolis – RJ – Brasil

batista@ic.unicamp.br,luciano@lrc.ic.unicamp.br,nfonseca@ic.unicamp.br
ziviani@lncc.br

Abstract. *Modern large-scale grid computing systems for processing advanced science and engineering applications rely on geographically distributed clusters. In such highly distributed environments, estimating the available bandwidth between clusters is a key issue for efficient task scheduling. We analyze the performance of two well known available bandwidth estimation tools,* `pathload` *and* `abget`*, with the aim of using them in grid environments. Our experiments consider the accuracy of the estimation, the convergence time, their level of intrusion in the grid links, and the ability to handle multiple simultaneous estimations. Overall,* `pathload` *represents a good solution to estimate available bandwidth in grid environments.*

Resumo. *As grades computacionais atuais voltadas para o processamento de aplicações avançadas baseiam-se em clusters distribuídos geograficamente. Nestes ambientes largamente distribuídos, é necessário que antes do escalonamento das aplicações seja estimada a largura de banda disponível entre os clusters. Este artigo apresenta resultados obtidos em experimentos que analisam o desempenho dos estimadores de largura de banda disponível* `pathload` *e* `abget` *com o objetivo de utilizá-los em grades. Na avaliação, considerou-se as características desejáveis de um estimador ideal para grades: não ser intrusivo, executar rápido, poder ser executado simultaneamente por diversos usuários e fornecer a largura de banda disponível de forma precisa. Resultados indicam que o estimador* `pathload` *é o mais indicado para o uso em grades.*

1. Introdução

Diferente dos sistemas paralelos convencionais confinados em redes locais, e cujos enlaces são dedicados à comunicação das aplicações paralelas, as grades expandem-se por diversos domínios administrativos interligados por enlaces de rede compartilhados entre as aplicações da grade e aplicações convencionais [Foster 2002].

Em ambientes paralelos e distribuídos como as grades, as aplicações são divididas em programas menores chamados de tarefas. Para que as aplicações executem na grade é necessário que as tarefas sejam escalonadas para um conjunto dos recursos disponíveis. Pelo fato das grades serem formadas por recursos distribuídos geograficamente e interligados por enlaces de rede não exclusivos, as decisões do escalonamento

das aplicações são influenciadas não apenas por parâmetros computacionais locais (CPU disponível, memória disponível, etc...) mas também por parâmetros relacionados com a rede (atraso, largura de banda disponível, etc...).

Além de ser um item crítico para o escalonamento de tarefas, a rede também exerce papel importante para sistemas que reagem às flutuações da disponibilidade dos recursos [Batista et al. 2008] [Huedo et al. 2002]. A motivação para a utilização de tais sistemas vem do fato de que os recursos das grades não são exclusivos para as aplicações em execução. Nestes sistemas os recursos são monitorados e se uma alocação de recursos diferente da atual fornecer ganhos para as aplicações em execução, as tarefas dessas aplicações são migradas. Ter estimativas precisas da largura de banda disponível é de fundamental importância para avaliar os ganhos que possam ser alcançados com a migração de tarefas em execução.

Exemplos que reforçam que parâmetros da rede devem ser considerados nas decisões de escalonamento e migração para aplicações executadas em grades são encontrados em [Silvester 2005], [Ito et al. 2005] e [LCG 2008]. Em [Silvester 2005] são apresentadas medições em enlaces de rede de uma grade construída sobre uma rede de pesquisa dos Estados Unidos. Durante a execução de aplicações na grade, observa-se que a utilização destes enlaces aumenta de cinco a oito vezes com relação à sua utilização em um dia normal. [Ito et al. 2005] apresenta uma proposta de ajuste automático de parâmetros do protocolo `GridFTP` para se alcançar a utilização máxima dos enlaces de grades e, dessa forma, diminuir os tempos de execução das aplicações. Pelos resultados obtidos, mostra-se que os ajustes ideais dos parâmetros do protocolo estão diretamente relacionados com a largura de banda disponível dos enlaces bem como com o produto banda $\times$ atraso entre os nós da grade. Um exemplo recente que reforça a importância das redes nas decisões de escalonamento está presente nos números referentes à grade que processará os dados produzidos pelo *Large Hadron Collider* (LHC) do CERN. Espera-se que a *LHC Computing Grid* (LCG) distribua cerca de 15PB por ano [LCG 2008]. A importância de avaliar o estado dos caminhos de rede nas grades também pode ser confirmada pela existência de um grupo de trabalho específico para isso dentro do Open Grid Forum (OGF), uma comunidade formada por usuários, desenvolvedores e fabricantes voltada para a padronização de ferramentas e de métodos utilizados em grades [OGF 2009].

Dada a importância da rede para as aplicações em grades, a largura de banda disponível entre os recursos é um parâmetro importante a ser considerado no escalonamento e na migração de tarefas. Entretanto, diferente de informações estáticas como a capacidade máxima dos enlaces, a largura de banda disponível é uma métrica dinâmica que precisa ser freqüentemente medida. Dentro de uma certa janela de tempo, o valor que representa a largura de banda disponível é incerto dada a dinâmica do ambiente e as imprecisões das ferramentas utilizadas [Jain and Dovrolis 2004] [Jain and Dovrolis 2005]. Ignorar tais incertezas pode trazer como impacto o aumento no tempo de execução de aplicações em grades, como demonstrado nos experimentos em [Batista et al. 2009], onde o escalonamento realizado desconsiderando as incertezas envolvidas foi responsável pelo aumento do tempo de execução de aplicações em cerca de 20%. O ideal, portanto, é encontrar uma faixa de valores que represente a largura de banda disponível nos enlaces em um certo instante de tempo.

Existem diversas formas de obter a largura de banda disponível em enlaces de

rede. Três métodos bastante utilizados são: (i) fazer previsões baseadas no histórico de medições passadas; (ii) medir diretamente nos equipamentos de interconexão; e (iii) utilizar estimadores nos dispositivos finais. Previsões são realizadas através do emprego de ferramentais matemáticos como séries temporais e redes neurais. Baseando-se nas medições feitas em um intervalo de tempo passado, fornece-se uma previsão de disponibilidade do futuro. Neste contexto, algumas ferramentas para grades têm sido propostas e vem sendo utilizadas para cuidar do armazenamento do histórico de medições [Massie et al. 2004] bem como para fazer previsões [Wolski et al. 1999]. Entretanto, esta abordagem de monitoramento exige um conhecimento prévio de todos os recursos que compõem a grade. Grades formadas por recursos que entram e saem freqüentemente não podem utilizar tais métodos. Este mesmo problema de entrada e saída de recursos afeta o método de medição direta nos dispositivos de interconexão. O problema ocorre dada a necessidade de intervenção de administradores das organizações onde os dispositivos estão localizados para que as informações sejam disponibilizadas para os usuários e os escalonadores de tarefas via SNMP ou web services [Zanikolas and Sakellariou 2005]. O último método consiste em sondar o caminho entre os nós e, a partir de métricas coletadas, como variações no atraso entre pacotes, estimar a capacidade disponível no enlace [Liu et al. 2008] [Prasad et al. 2003]. A vantagem deste método vem do fato dele poder ser utilizado sem intervenção em recursos que não sejam os dispositivos finais da grade. Desse modo, torna-se possível descobrir a largura de banda disponível até mesmo para um nó que tenha sido recentemente agregado à grade pela primeira vez.

O objetivo deste artigo é comparar, através de medições, os estimadores de largura de banda passante disponível `pathload` [Jain and Dovrolis 2003] e `abget` [Antoniades et al. 2006] com relação a métricas relevantes para as suas utilizações em ambientes de grades. Além da precisão das estimativas, são avaliados o tempo de convergência das ferramentas, a intrusão nos enlaces da grade e o comportamento quando mais de uma instância é executada ao mesmo tempo para medição de disponibilidade de dois hosts distintos para um mesmo host alvo. O tempo de convergência é uma métrica importante pelo fato dela afetar o tempo de execução total da aplicação na grade. Assim como ocorre com o tempo de execução dos escalonadores de tarefas [Batista et al. 2008], o tempo de convergência dos estimadores deve ser relativamente pequeno quando comparado com o tempo de execução total da aplicação. A intrusão nos enlaces tem impacto na disponibilidade dos recursos para outros usuários, pois as grades utilizam recursos que são compartilhados por diversas aplicações. Finalmente, o comportamento quando ocorrem execuções simultâneas da ferramenta é importante visto que esse tipo de situação tende a ocorrer com freqüência em grades, dado o comportamento assíncrono dos usuários. Outros trabalhos que comparam estimadores de largura de banda passante disponível e também avaliam o `pathload` e o `abget` são encontrados em [R. S. Prasad and Dovrolis 2003] e [Shriram et al. 2005] (além de [Jain and Dovrolis 2003] e [Antoniades et al. 2006], onde as ferramentas são introduzidas). Entretanto, em nenhum desses quatro trabalhos são avaliadas todas as métricas que são avaliadas neste artigo. A maioria dos trabalhos restringem-se em comparar as ferramentas com relação à precisão das medições.

Conforme explicado anteriormente, a largura de banda disponível entre dois nós apresenta incertezas que exige a representação do seu valor como um intervalo, ao invés de uma simples média aritmética. Dentre os diversos estimadores que os autores pesquisaram na literatura ([CAIDA 2008] [Antoniades et al. 2006] [Ozturk and Kulkarni 2008]),

somente o `pathload` e o `abget` fornecem estimativas como um intervalo de valores e por este motivo os experimentos realizados restringem-se a eles. Observou-se que, com algumas pequenas modificações, o `pathload` representa uma boa solução para estimar a disponibilidade de recursos de redes em grades.

As próximas seções deste artigo estão organizadas da seguinte forma. A Seção 2 resume no que consiste o processo de estimativa de largura de banda. A Seção 3 apresenta as ferramentas `pathload` e `abget`. A Seção 4 descreve os experimentos realizados e discute os resultados obtidos. A Seção 5 conclui o artigo.

2. Estimativa de largura de banda

Estimar largura de banda, geralmente, inclui diversas métricas tais como a capacidade nominal do enlace, a capacidade do gargalo de um caminho, a largura de banda disponível em um caminho e a capacidade máxima de transferência (*Bulk Transfer Capacity* – BTC) entre dois hosts [Prasad et al. 2003].

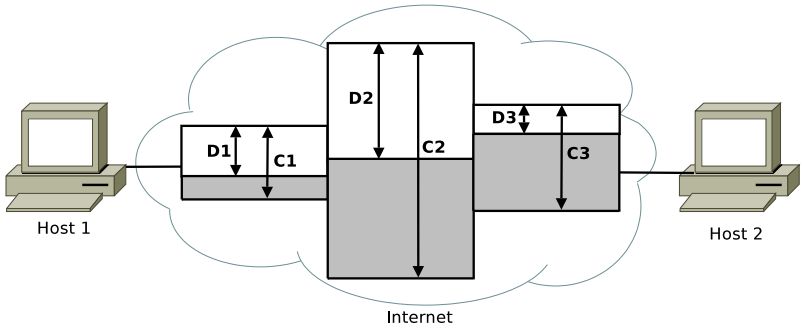


Figura 1. Métricas relacionadas com largura de banda (figura baseada em [Prasad et al. 2003]).

A Figura 1 ilustra as várias métricas relacionadas com largura de banda. Sejam os hosts Host 1 e Host 2 interligados por um caminho de rede formado por 3 enlaces/hops. Cada enlace na figura é simbolizado por um retângulo em que a parte cinzenta representa a capacidade ocupada do enlace, enquanto a parte em branco representa a largura de banda disponível. Neste exemplo, a capacidade nominal de cada enlace é $C1$, $C2$ e $C3$, a capacidade do gargalo do caminho é $C1$ e a largura de banda disponível fim a fim é $D3$.

A BTC é a máxima vazão obtida por uma conexão TCP no caminho. A BTC não pode ser representada na Figura 1 pois esta métrica depende do tipo de tráfego concorrente que esteja ocupando os enlaces do caminho no momento das medições. A dependência do tipo de tráfego concorrente deve-se ao fato do protocolo TCP implementar algoritmos de controle de congestionamento que tendem a compartilhar a capacidade dos enlaces entre os fluxos TCP existentes. Por exemplo, se em um caminho a largura de banda disponível é zero mas o tráfego concorrente consiste de um único fluxo TCP, uma ferramenta ideal para medir a BTC informaria um valor igual à metade da capacidade do enlace. Neste artigo nós focamos em ferramentas que estimam a largura de banda disponível fim a fim.

3. Estimadores de largura de banda disponível em grades

Em [CAIDA 2008], apresenta-se uma lista de diversas ferramentas para estimativas de largura de banda disponível, entretanto, com exceção do `pathload`, nenhuma delas

fornece a estimativa por meio de um intervalo. Uma ferramenta bastante utilizada por operadores e administradores de redes para estimar a largura de banda disponível é o `iperf` [NLANR 2008]. O `iperf` mede a disponibilidade do caminho entre dois hosts de uma forma bastante simples. Cada host executa um processo local e os processos comunicam-se entre si através do envio de dados (UDP ou TCP), a fim de saturar o caminho entre eles. Ao término da medição mede-se quantos bytes B foram recebidos durante o intervalo de tempo Δt, sendo a largura de banda disponível igual a $\frac{B}{\Delta t}$. A intrusão na rede gerada pelo `iperf` e o fato da estimativa ser fornecida como uma média fazem dele uma opção ineficiente para ser utilizado em grades. Em [Tirumala et al. 2003] é apresentada uma forma de diminuir a intrusão do `iperf` quando ele é executado com a opção de enviar dados via protocolo TCP. É utilizada uma pilha TCP modificada que permite a identificação do momento em que a fase de partida lenta termina e a estimativa do `iperf` passa a ser realizada somente partir desse momento. Como após a fase de partida lenta a vazão alcançada pelo TCP está mais estável, a estimativa pode ser realizada muito mais rápida e com menos intrusão. Apesar desta modificação resolver o problema da intrusão, a estimativa continua sendo fornecida através de uma média. O `DIChirp` [Ozturk and Kulkarni 2008] é um estimador de largura de banda disponível que utiliza acesso direto ao *hardware* das placas de rede para evitar estimativas incorretas causadas pelas variações nas trocas de contexto dos processos. O `DIChirp` estima a largura de banda disponível através do envio de conjuntos de pacotes chamados *chirps* entre os dois hosts envolvidos na estimativa. O processo tem início com o envio dos *chirps* à menor taxa possível. Caso o destinatário detecte aumento no atraso dos pacotes dentro de um *chirp* considera-se que a taxa de envio daquele *chirp* é maior do que a largura de banda disponível entre os dois hosts. A última taxa utilizada sem variações no atraso é considerada como a largura de banda disponível. Assim como o `iperf`, o `DIChirp` fornece a estimativa através de um único valor, o que também o torna ineficiente para ser utilizado em grades.

As ferramentas consideradas neste artigo, `pathload` e `abget`, fornecem a estimativa de largura de banda disponível por meio de um intervalo e esta é a principal característica que as torna ideais para estimarem os estados dos enlaces de grades. Ambas as ferramentas fazem as suas estimativas utilizando a técnica *Self-Loading Periodic Streams* (SLoPS). São transmitidos diversos fluxos de pacotes entre os nós finais a diferentes taxas e mede-se o atraso em um sentido (*One-Way Delay* – OWD) [1] de cada pacote para as diferentes taxas. Se for detectado aumento do OWD, isto significa que a taxa de transmissão é maior do que a largura de banda disponível, caso contrário, a taxa é menor do que a largura de banda disponível. As informações a respeito das variações no OWD são trocadas entre os nós finais a fim de que, através de um algoritmo iterativo, seja encontrado o intervalo que representa a largura de banda disponível no caminho.

O `pathload` e o `abget` estimam a largura de banda disponível fim a fim conforme explicado a seguir:

[1] Para medir o OWD, os hosts finais devem estar sincronizados. Possíveis soluções para o sincronismo são o uso de servidores NTP – Protocolo de Tempo em Rede (*Network Time Protocol*), a utilização de placas GPS em ambos os hosts finais ou um relógio por software para melhorar a precisão das medições sem usar as placas GPS, como proposto em [Pásztor and Veitch 2002].

3.1. `pathload`

Quando se deseja estimar a largura de banda do host A para o host B utilizando o `pathload` deve-se executar um processo "remetente" no host A e um processo "destinatário" no host B. As informações a cerca das variações no OWD são trocadas entre os hosts através de uma conexão TCP de controle. O "remetente" inicia o processo de estimativa enviando 1 fluxo de pacotes UDP a uma taxa inicial T_{start}. À medida que os pacotes chegam no "destinatário", o OWD é calculado. Caso não seja observado aumento nos OWD dos pacotes, o "destinatário" informa, pela conexão TCP de controle, ao "remetente" que aumente a taxa de envio dos pacotes. Vários algoritmos de ajuste são implementados no `pathload` para evitar que variações no OWD que não sejam causadas pela taxa maior do que a largura de banda disponível sejam interpretadas de forma incorreta. Ao final da estimativa o `pathload` fornece como saída um intervalo $[T^{min}, T^{max}]$ que corresponde à largura de banda disponível no caminho entre o host A e o host B.

3.2. `abget`

Para se estimar a largura de banda do host A para o host B utilizando o `abget`, basta que seja executado um processo no host B direcionado para o endereço do host A ou o endereço de um servidor TCP (por exemplo, um servidor HTTP) localizado na rede local do host A. O algoritmo implementado no `abget` simula o funcionamento do protocolo TCP de modo a controlar a taxa com a qual o host A envia pacotes para o host B. O `abget` ignora a implementação padrão do TCP do sistema operacional e manipula os ACKs enviados para o host A. Falsos ACKs são enviados para o host A informando o reconhecimento de somente 1 MSS. Ao receber cada ACK, o host A envia para o host B um pacote com o tamanho de 1 MSS. Portanto, se o host B envia os ACKs com um período P, ele induz o host A a enviar dados a uma taxa $T = MSS/P$. À medida que os pacotes do host A chegam ao host B o OWD é medido e a taxa com que os ACKs são gerados é ajustada a fim de se encontrar o intervalo $[T^{min}, T^{max}]$ que corresponde à largura de banda disponível.

O `abget` fornece duas opções para a convergência ao intervalo que corresponde à largura de banda disponível entre dois hosts. A primeira opção utiliza uma busca binária, enquanto a segunda realiza uma busca linear. Na busca binária, o `abget` multiplica ou divide por 2 a taxa de envio dos pacotes de estimativa, caso seja detectado que essa taxa é menor ou maior do que a largura de banda disponível, respectivamente. No caso da busca linear, o aumento ou diminuição da taxa é realizada através da adição ou subtração de 1 unidade dessa taxa, respectivamente.

3.3. Diferenças entre `pathload` e `abget`

A principal vantagem do `abget` está no fato dele dispensar a execução de processos nos dois hosts, o que representa uma desvantagem do `pathload`. Uma desvantagem do `abget` vem do fato dele necessitar ser executado com privilégio de super-usuário, já que somente dessa forma é possível desviar da implementação padrão do TCP do sistema operacional. Outra desvantagem deve-se ao fato de que muitos parâmetros devem ser informados manualmente pelo usuário a fim de que a convergência para o intervalo correspondente à largura de banda disponível seja realizada de forma rápida. O `pathload` dispensa esses parâmetros por conta da conexão TCP de controle que permite o ajuste fino do SLoPS em tempo de execução. Uma desvantagem do `pathload` vem do fato dele

utilizar pacotes UDP para fazer a sua estimativa, pacotes que geralmente são bloqueados em *firewalls* das organizações. O `abget` não possui esse problema pois a maioria das organizações já mantém liberada a passagem de pacotes para servidores HTTP, entretanto a exigência de haver um servidor TCP na rede de cada *cluster* que compõem a grade representa mais uma desvantagem do `abget`.

4. Análise de desempenho

O `pathload` e o `abget` foram avaliados em dois cenários distintos. Os experimentos no primeiro cenário têm o objetivo de avaliar as ferramentas em um ambiente com enlaces de baixa capacidade nominal (10Mbps), enquanto os experimentos no segundo cenário avaliam as ferramentas em um ambiente com enlaces de alta capacidade nominal (1Gbps). Avaliar as ferramentas em ambos os cenários é importante dada a heterogeneidade dos recursos de rede que interligam recursos em grades.

4.1. Primeiro cenário: enlaces com baixa capacidade nominal

O primeiro cenário, resumido na Figura 2, foi emulado no programa `NCTUns`, um emulador e simulador de redes largamente utilizado na literatura [Wang et al. 2007]. O `NCTUns` foi utilizado pela facilidade que ele fornece para se criar topologias de redes virtuais com as mais diversas características e por permitir a interação desta topologia com hosts reais. Os hosts reais podem executar aplicações que utilizem a rede sem necessidade de modificações. No caso dos experimentos realizados no primeiro cenário, o NCTUns foi necessário para que as aplicações acessassem enlaces com 10Mbps de capacidade nominal, ao invés dos 1Gbps da rede real. As estimativas de largura de banda disponível neste primeiro cenário foram realizadas do host cronos, e do host eolo, para o host mnemosyne. Todos esses 3 hosts são hosts reais localizados em uma mesma rede local e interligados por uma rede Gigabit Ethernet. O `NCTUns` foi executado na máquina urano, também localizada na mesma rede local. A fim de observar o comportamento dos estimadores sob diversas condições da rede, utilizou-se dois hosts virtuais no `NCTUns` para gerar tráfego de interferência que afeta a largura de banda disponível entre os hosts reais. A Tabela 1 resume as configurações de todos os hosts envolvidos nos experimentos.

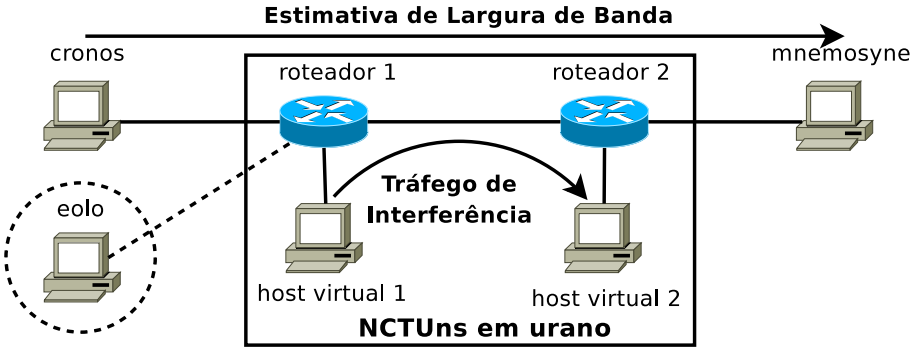


Figura 2. Ambiente utilizado para os experimentos com o NCTUns.

Três métricas foram avaliadas nos experimentos. A precisão, o tempo de execução e a intrusão das ferramentas. No primeiro cenário, avaliou-se as ferramentas sem a ocorrência de tráfego de interferência, com tráfego de interferência UDP CBR igual a 2, 4, 6, 8 e 10Mbps e com tráfego de interferência TCP. Como a capacidade nominal dos

Tabela 1. Características dos hosts utilizados nos experimentos.

Host	Processador/Memória	Sistema operacional (Linux)
eolo	Intel Core 2 Quad 2.66GHz / 4GB	Debian kernel 2.6.23.1
cronos	Intel Core 2 Quad 2.40GHz / 4GB	Debian kernel 2.6.23.1
mnemosyne	Dual Xeon 2GHz / 4GB	Debian kernel 2.6.23.1
urano	Intel Core 2 Duo 2.13GHz / 4GB	Fedora kernel 2.6.25.9

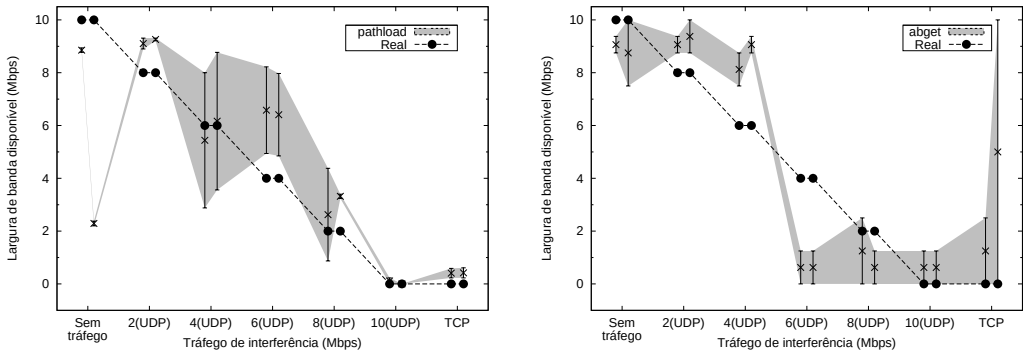
enlaces foi mantida fixa em 10Mbps durante todas as medições, uma simples subtração permite o cálculo da largura de banda disponível real entre os hosts reais. Numa primeira etapa as medições foram realizadas somente entre cronos e mnemosyne. Na segunda etapa, as medições foram realizadas simultaneamente entre cronos e mnemosyne e entre eolo e mnemosyne. Apenas os resultados mais relevantes são apresentados neste artigo. Em todos os experimentos o `abget` foi executado com os parâmetros referentes à utilização de uma busca binária. Experimentos utilizando a busca linear mostraram-se muito custosos em termos de tempo de execução quando comparados com a execução via busca binária (o tempo de execução foi cerca de 300% maior).

Um detalhe importante referente à configuração dos experimentos diz respeito à desabilitação da *Interrupt Coalescence* (IC). A IC é uma característica que algumas placas de rede possuem com o objetivo de diminuir a quantidade de interrupções que são geradas para o sistema operacional quando pacotes são recebidos ou enviados. Sem a IC habilitada, cada pacote gera 1 interrupção. Com a IC habilitada, interrupções são geradas somente após uma certa quantidade de pacotes ou após um certo intervalo de tempo. Como demonstrado em [Prasad et al. 2004], estimadores de largura de banda disponível baseados em medições do OWD fornecem estimativas incorretas caso sejam executados em máquinas com IC habilitada. O `pathload` possui códigos que garantem seu correto funcionamento em ambientes com a IC habilitada, mas somente se as placas de rede forem de fabricantes específicos. Já o `abget` não possui nenhum código implementado para lidar com a IC. Por conta disso, todas as máquinas utilizadas nos experimentos tiveram a IC desabilitada. Uma conseqüência da desabilitação da IC é a diminuição da vazão máxima suportada em redes de alta velocidade ($\geq$ 1Gbps) mas isso não influencia nos resultados que foram obtidos dado que o interesse é na precisão das ferramentas.

As Figuras 3 e 4 exibem os resultados obtidos para a primeira etapa do primeiro cenário, em que os estimadores foram executados somente entre cronos e mnmemosyne.

Os gráficos da Figura 3(a) e da Figura 3(b) apresentam, respectivamente, as estimativas de largura de banda disponível fornecidas pelo `pathload` e pelo `abget` em função do tráfego de interferência (Os valores no eixo horizontal estão ordenados de forma diretamente proporcional ao impacto na disponibilidade do caminho). A curva denominada “Real” apresenta a largura de banda disponível real entre os hosts. As áreas cinzentas denominadas “pathload” e “abget” apresentam os intervalos que foram devolvidos pelas ferramentas. Para cada configuração de tráfego de interferência, as ferramentas foram executadas duas vezes seguidas. Os resultados mostram que as estimativas do `pathload` ficaram mais próximas dos valores reais quando havia tráfego de interferência na rede. Ao contrário das estimativas do *pathload*, as estimativas do `abget` não seguiram o comportamento da curva da disponibilidade real do caminho entre cronos e mnemosyne. Com tráfego de interferência $\leq$ 4Mbps, o `abget` estimou que o caminho estava praticamente 100% disponível, enquanto que com tráfego de interferência

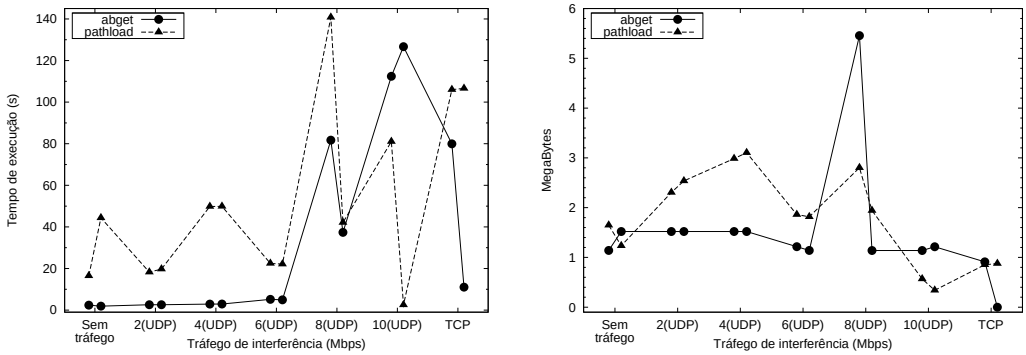
$\geq$ 6Mbps, as estimativas passaram a informar que o enlace estava praticamente 100% ocupado. É importante observar que com tráfego de interferência TCP os intervalos das estimativas do `abget` foram em média maiores do que as outras configurações de tráfego de interferência.



(a) `pathload` (b) `abget`

Figura 3. Estimativas fornecidas (1º cenário).

O tempo de execução dos estimadores está exibido no gráfico da Figura 4(a). Pelo fato do `abget` não possuir um ajuste inicial da taxa de transmissão como o `pathload` possui (É preciso fornecer como parâmetro para o abget um valor que seja próximo à capacidade nominal dos enlaces), ele precisa transferir uma mesma quantidade de dados no início da estimativa, independente da disponibilidade dos enlaces. Por conta disso, à medida que os enlaces mostram-se mais ocupados, o tempo de execução tende a aumentar, como observa-se na curva "abget". Entretanto, como o `abget` foi executado com a opção de se estimar o intervalo através de uma busca binária, ele consegue convergir para um resultado mais rápido do que o `pathload` quando a concorrência nos enlaces é baixa ($\leq$ 8Mbps). O baixo tempo de execução do `abget` em uma das execuções com tráfego de interferência TCP deve-se ao fato de que a ferramenta não conseguiu estabelecer conexão com o servidor web de mnemosyne por conta do intenso tráfego de interferência.



(a) Tempo de execução (b) Intrusão

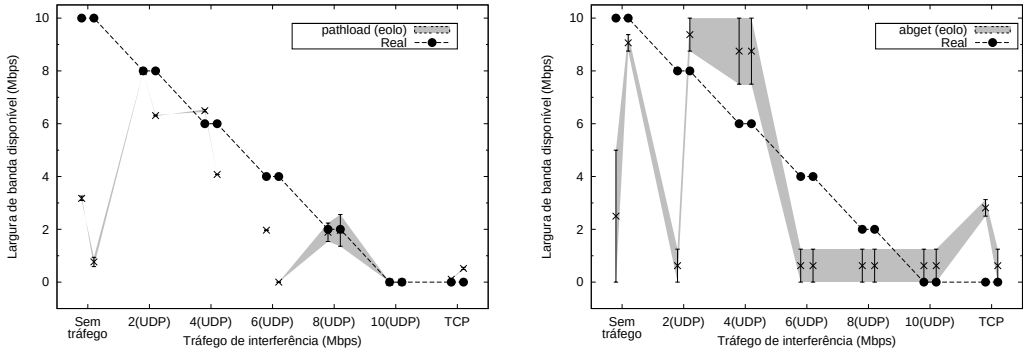
Figura 4. Desempenho (1º cenário).

As informações referentes à intrusão das ferramentas estão exibidas no gráfico da Figura 4(b). Neste gráfico é exibida a quantidade de bytes que foram injetados por cada ferramenta durante sua execução. Observa-se, na média, um comportamento mais estável do `abget`. O `pathload` tem uma tendência em diminuir o tráfego gerado à medida

que os enlaces vão se mostrando mais ocupados. Isso ocorre pelo fato de que as taxas de transmissão do `pathload`, a cada iteração do seu algoritmo, não são definidas de acordo com uma busca binária como é implementada no `abget`. Dessa forma, após 1 iteração o `pathload` pode diminuir a sua taxa e injetar menos tráfego na rede do que o `abget`.

Alguns dos resultados obtidos para a segunda etapa do primeiro cenário, quando os estimadores foram executados entre cronos e mnmemosyne e entre eolo e mnemosyne, estão exibidos nos gráficos das Figuras 5 e 6. Os resultados referentes à intrusão das ferramentas foram similares àqueles obtidos com apenas uma instância dos estimadores em execução e por conta disso não são exibidos. É importante observar que para ser possível realizar a execução simultânea do `pathload` foi necessário modificar o seu código-fonte, já que no código original as portas utilizadas pelo programa são definidas de forma estática.

Os gráficos das Figuras 5(a) e 5(b) exibem as estimativas do `pathload` e do `abget` executados entre os hosts eolo e mnemosyne. Os resultados obtidos entre os hosts cronos e mnemosyne foram semelhantes e por isso não são apresentados. O `pathload` continuou fornecendo melhores resultados do que o `abget` principalmente quando há tráfego de interferência. A principal diferença dos resultados do `pathload`, com relação aos resultados da primeira etapa, vem do fato dos intervalos serem menores nesta segunda etapa. A combinação dos tráfegos UDP do estimador torna a disponibilidade do enlace menor. Como esses tráfegos têm taxas fixas que tendem a ser produzidas nos mesmos intervalos de tempo, observa-se variações menores na disponibilidade dos enlaces, o que diminui os intervalos.



(a) `pathload` (b) `abget`

Figura 5. Estimativas fornecidas com execuções simultâneas em eolo (1º cenário).

O principal resultado referente aos tempos de execução (Figura 6) diz respeito ao crescimento mais acentuado do tempo de execução do `abget`. Assim como nos gráficos que apresentam as estimativas das ferramentas, somente é exibido o tempo de execução entre os hosts eolo e mnemosyne porque os resultados obtidos entre cronos e mnemosyne foram similares. A combinação dos tráfegos das duas instâncias em execução amplia o tempo de execução do estimador a partir de 4Mbps (UDP), ao contrário do crescimento mais intenso a partir de 8Mbps (UDP) quando apenas uma instância estava em execução.

De um modo geral, neste primeiro cenário observou-se estimativas melhores por parte do `pathload` do que do `abget`, embora o `abget`, na maioria das vezes, tenha

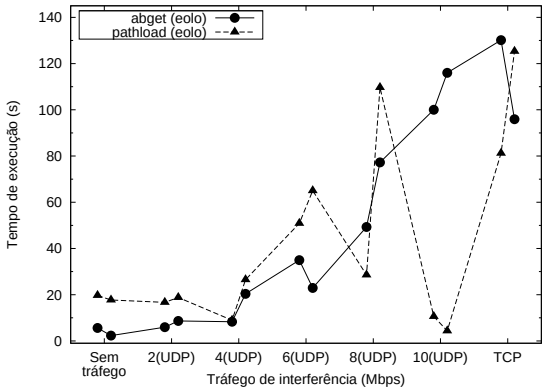


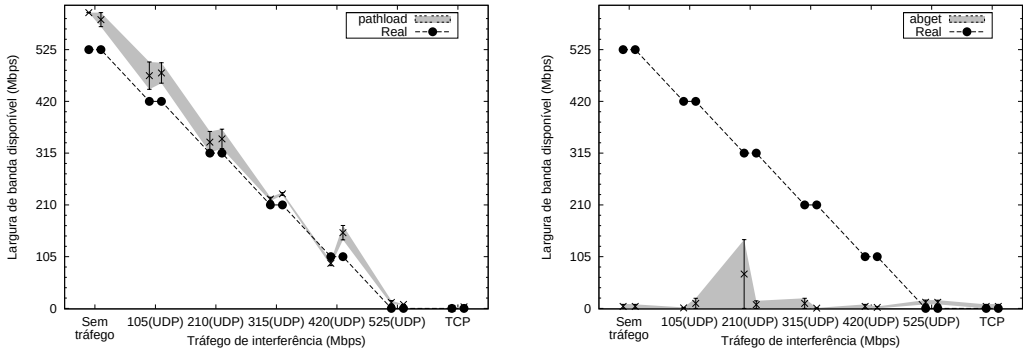
Figura 6. Tempo de execução com execuções simultâneas em eolo (1º cenário).

executado mais rápido e tenha apresentado um comportamento mais estável, na média, em termos de intrusão. Ambos os estimadores executaram relativamente rápidos ($<$ 2min20seg), considerando-se o tempo de vida de aplicações em grades que levem horas para executar, e injetaram poucos bytes na rede ($<$ 6MB).

4.2. Segundo cenário: enlaces com alta capacidade nominal

O segundo cenário construído para avaliar o `pathload` e o `abget` difere do primeiro cenário pelo fato do `NCTUns` não ser utilizado. As ferramentas foram avaliadas diretamente sobre os enlaces reais de 1Gbps da rede local que interliga os hosts. Sem o `NCTUns` para gerar o tráfego de interferência entre os hosts, utilizou-se o programa `iperf` [NLANR 2008] para este fim. Os hosts virtuais e os roteadores foram criados em urano através de interfaces de rede virtuais. As estimativas continuaram sendo realizadas entre os mesmos hosts descritos no cenário 1. Apesar dos enlaces terem capacidade nominal de 1Gbps, a capacidade real observada foi de 525Mbps, por isso considera-se que esta é a capacidade nominal dos enlaces. As mesmas métricas e as mesmas etapas do primeiro cenário valem para este segundo cenário. A diferença está nas taxas dos tráfegos de interferência UDP. Os valores utilizados foram 105, 210, 315, 420 e 525Mbps.

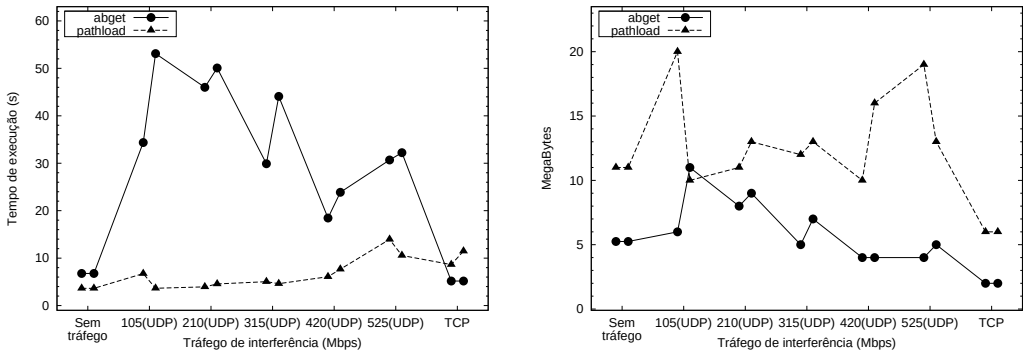
A precisão das ferramentas quando as estimativas foram realizadas somente entre cronos e mnemosyne está exibida nos gráficos da Figura 7. As estimativas do `pathload` (Figura 7(a)) continuaram melhores do que as estimativas do `abget` (Figura 7(b)). Diferente do primeiro cenário, as estimativas do `pathload` seguem claramente o padrão de disponibilidade dos enlaces. De forma semelhante ao primeiro cenário, as melhores estimativas do `pathload` foram fornecidas quando havia mais tráfego de interferência na rede. O `abget` apresentou um comportamento diferente com relação ao primeiro cenário. Apesar dos enlaces fornecerem mais capacidade de transmissão, o estimador informou que o enlace estava praticamente 100% ocupado independente do nível de tráfego de interferência. É importante observar que o parâmetro que define a maior taxa que o `abget` deve estimar foi modificado para 525Mbps, ao invés dos 10Mbps utilizado no primeiro cenário, entretanto não houve melhorias na estimativa fornecida. Diversos ajustes foram realizados nos demais parâmetros da ferramenta em busca de melhores resultados mas não se obteve sucesso. É importante destacar a quantidade de parâmetros do `abget` como um ponto negativo da ferramenta em grades nas quais novos usuários estejam constantemente chegando e recursos estejam sempre entrando e saindo.



(a) `pathload` (b) `abget`

Figura 7. Estimativas fornecidas (2º cenário).

Com relação ao tempo de execução, observa-se pelo gráfico da Figura 8(a) uma inversão com relação ao primeiro cenário. Os tempos de execução do `abget` foram em média bem maiores do que os tempos de execução do `pathload`. O `pathload` só executou por mais tempo que o `abget` quando havia tráfego de interferência TCP. Observando as saídas das execuções do `abget` nota-se que o intervalo de tempo despendido em cada uma das taxas dentro da busca binária é muito maior do que aquele despendido pelo `pathload`. Enquanto o `pathload` interrompe o envio de pacotes a uma certa taxa e passa para a taxa seguinte quando detecta-se estabilidade no atraso entre os pacotes iniciais, o `abget` sempre mantém uma quantidade constante de pacotes enviados para cada taxa. O aumento da quantidade de taxas a serem avaliadas pelo `abget` neste segundo cenário é responsável pelo maior tempo de execução da ferramenta em relação ao `pathload`.



(a) Tempo de execução (b) Intrusão

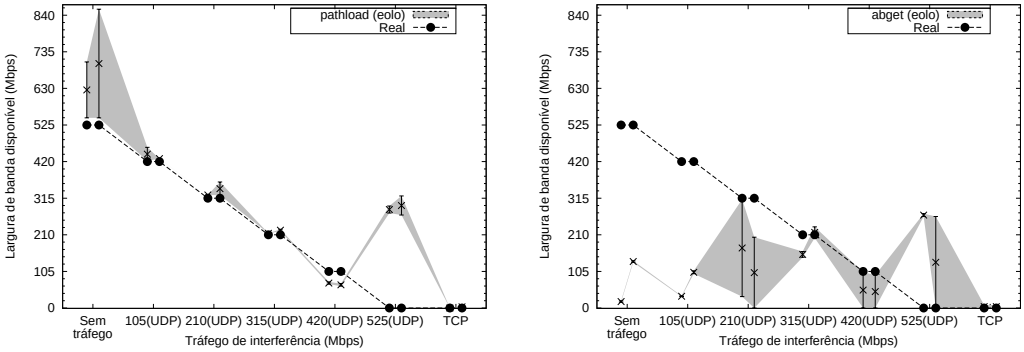
Figura 8. Desempenho (2º cenário).

Com relação à intrusão, pelo gráfico da Figura 8(b) observa-se que o `pathload` injetou mais tráfego na rede do que o `abget` e que o `abget` continuou apresentando uma variação menor em termos de bytes enviados na rede do que o `pathload`. Comparando os resultados da Figura 8(b) com aqueles da Figura 4(b), nota-se que o nível de intrusão aumentou proporcionalmente ao aumento da capacidade dos enlaces (10Mbps → 522Mbps).

As principais diferenças entre os resultados observados com a execução de duas instâncias simultâneas das ferramentas e os resultados observados com a execução de uma única instância podem ser observadas nos gráficos das Figuras 9 e 11. Assim como nos

experimentos realizados no NCTuns, os resultados obtidos entre os hosts eolo e mnemosyne e entre cronos e mnemosyne foram semelhantes e por isso somente os primeiros são exibidos.

Com relação à precisão (Figuras 9(a) e 9(b)), observa-se que os intervalos fornecidos pelo abget passaram a ser maiores e que o pathload forneceu estimativas bem diferentes da disponibilidade real quando os enlaces mostraram-se mais congestionados. Além disso, observou-se também o aumento do intervalo fornecido pelo pathload quando não houve tráfego de interferência na rede.



(a) pathload (b) abget

Figura 9. Estimativas fornecidas com execuções simultâneas em eolo (2º cenário).

O principal destaque referente ao tempo de execução dos estimadores (Figura 10) continua sendo o crescimento do tempo de execução do abget à medida que a rede mostra-se mais congestionada. O tempo de execução chega a aumentar cerca de 1 ordem de grandeza entre o caso onde o tráfego de interferência foi de 420Mbps UDP e o caso onde o tráfego de interferência foi de 525Mbps UDP.

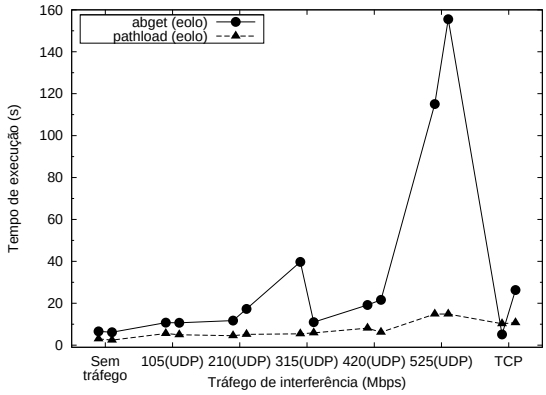


Figura 10. Tempo de execução com execuções simultâneas em eolo (2º cenário)

Com relação à intrusão, é importante destacar o comportamento do pathload executado em cronos quando o tráfego de interferência é de 105Mbps (UDP) (Figura 11). Em uma das execuções, o estimador gerou cerca de 300MB de dados, uma quantidade que não é irrelevante. Esse resultado registra que há a possibilidade das ferramentas mostrarem-se muito intrusivas, o que justifica a implementação de procedimentos que evitem tal comportamento. Uma solução simples consistiria em solicitar do usuário um valor máximo para o tempo de execução da ferramenta ou para a quantidade de bytes

que podem ser enviados. Apesar da possibilidade dessa condição de parada gerar estimativas piores, o resultado final pode ser melhor se forem consideradas todas as métricas envolvidas.

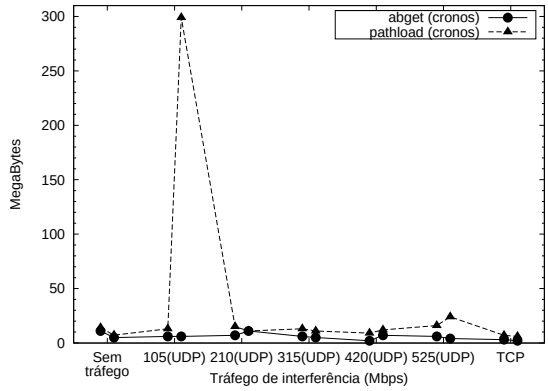


Figura 11. Intrusão com execuções simultâneas em cronos (2º cenário)

Em resumo, pelo observado nos resultados do segundo cenário, conclui-se que o `pathload` fornece bons resultados mesmo em um ambiente formado por enlaces com alta capacidade de transmissão. Nestes ambientes o tempo de execução do `pathload` mostrou-se também inferior ao do `abget`, apesar do `abget` em média ser menos intrusivo. Escalonadores que utilizassem o `abget` para estimar a largura de banda disponível neste cenário produziriam escalonamentos que subutilizariam a rede, já que poucas transferências seriam utilizadas devido às estimativas de congestionamento mesmo em situações onde os enlaces estavam praticamente 100% disponíveis. Observou-se ainda que o `pathload` tem o potencial de inundar a rede com tráfego que afetaria o funcionamento das demais aplicações em execução.

De um modo geral, considerando todos os cenários, o `pathload` mostrou-se a melhor ferramenta em termos do conjunto de métricas avaliadas. Entretanto, usuários e administradores de grades devem ter noção da possibilidade de interferência nos demais fluxos da rede durante as estimativas. Além disso, o fato do `pathload` ter que ser executado em ambos os hosts considerados nas estimativas é um ponto negativo que motiva modificações no `abget` para que suas estimativas sejam tão precisas quanto as do `pathload`. Também há a necessidade de modificação do código do `pathload` para que as portas utilizadas sejam definidas de forma dinâmica, permitindo assim a sua utilização simultânea entre diversos hosts da grade.

5. Conclusão

Por serem ambientes concorrentes, heterogêneos e descentralizados, as grades possuem disponibilidades que variam com o passar do tempo. Um dos processos essenciais para a execução de aplicações em grades é a escolha dos recursos que serão utilizados por cada aplicação. Para que a escolha seja feita da melhor forma possível, é importante que se tenha uma visão atualizada da grade em termos da disponibilidade de cada recurso, com destaque para os recursos de redes, que são críticos para aplicações de e-Science devido à quantidade maciça de dados transferida.

Este artigo comparou o `pathload` e o `abget`, duas ferramentas de estimativa de largura de banda disponível, no contexto dos requisitos que tais ferramentas devem

atender para a sua utilização em grades: precisão nas estimativas com informações referentes à incertezas, baixa intrusão, rapidez nas estimativas e corretude em caso de execuções concorrentes. Pelos experimentos realizados, observou-se melhores resultados do `pathload` do que do `abget` quando todos os requisitos foram avaliados em conjunto. Entretanto, modificações devem ser feitas no `pathload` para que ele possa ser executado em vários hosts da grade simultaneamente e para evitar que ele gere tráfego que interfira nos demais fluxos da grade durante suas estimativas. Devido à sua vantagem de ser executado em apenas 1 dos hosts durante as estimativas, é importante considerar modificações no `abget` a fim de melhorar a sua precisão e a fim de diminuir a quantidade de parâmetros que devem ser passados pelo usuário.

Referências

Antoniades, D., Athanatos, M., Papadogiannakis, A., Markatos, E., and Dovrolis, C. (2006). Available Bandwidth Measurement as Simple as Running wget. In *Proceedings of the Passive and Active Measurement Workshop 2006*.

Batista, D. M., da Fonseca, N. L. S., Miyazawa, F. K., and Granelli, F. (2008). Self-Adjustment of Resource Allocation for Grid Applications. *Computer Networks*, 52(9):1762–1781.

Batista, D. M., Drummond, A. C., and Fonseca, N. L. S. (2009). Robust Scheduler for Grid Networks. In *SAC '09: Proceedings of the 2009 ACM Symposium on Applied Computing*, pages 354–358, New York, NY, USA. ACM Press.

CAIDA (2008). CAIDA : tools : taxonomy. `http://www.caida.org/tools/taxonomy/performance.xml#bw`. Acessado em 6 de Janeiro de 2009.

Foster, I. (2002). What is the Grid? A Three Point Checklist. *GRIDToday*, 1(6). `http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf`. Acessado em 6 de Janeiro de 2009.

Huedo, E., Montero, R. S., and Llrorent, I. M. (2002). An Experimental Framework for Executing Applications in Dynamic Grid Environments. Technical Report 2002-43, NASA Langley Research Center.

Ito, T., Ohsaki, H., and Imase, M. (2005). On Parameter Tuning of Data Transfer Protocol GridFTP for Wide-Area Grid Computing. In *Proceedings of the BroadNets 2005*, volume 2, pages 1338–1344.

Jain, M. and Dovrolis, C. (2003). End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM TON*, 11(4):537–549.

Jain, M. and Dovrolis, C. (2004). Ten Fallacies and Pitfalls on End-to-end Available Bandwidth Estimation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 272–277, New York, NY, USA. ACM.

Jain, M. and Dovrolis, C. (2005). End-to-end Estimation of the Available Bandwidth Variation Range. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 265–276, New York, NY, USA. ACM Press.

LCG (2008). WLCG Worldwide LHC Computing Grid. http://lcg.web.cern.ch/LCG/public/. Acessado em 6 de Janeiro de 2009.

Liu, X., Ravindran, K., and Loguinov, D. (2008). A Stochastic Foundation of Available Bandwidth Estimation: Multi-Hop Analysis. *IEEE/ACM TON*, 16(1):130–143.

Massie, M. L., N.Chun, B., and Culler, D. E. (2004). The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, 30(7):817–840.

NLANR (2008). Iperf. http://sourceforge.net/projects/iperf/?abmode=1. Acessado em 7 de Janeiro de 2009.

OGF (2009). Network Measurements Working Group (NM-WG). http://forge.ggf.org/sf/projects/nm-wg. Acessado em 15 de maio de 2009.

Ozturk, Y. and Kulkarni, M. (2008). DIChirp: Direct Injection Bandwidth Estimation. *Int. J. Netw. Manag.*, 18(5):377–394.

Pásztor, A. and Veitch, D. (2002). PC-based precision timing without GPS. In *ACM SIGMETRICS*, Los Angeles, CA, USA.

Prasad, R., Dovrolis, C., Murray, M., and Claffy, K. (2003). Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. *Network, IEEE*, 17(6):27–35.

Prasad, R., Jain, M., and Dovrolis, C. (2004). Effects of Interrupt Coalescence on Network Measurements. In *Proceedings of the Passive and Active Network Measurement Workshop 2004*, pages 247–256.

R. S. Prasad, M. J. and Dovrolis, C. (2003). Evaluating Pathrate and Pathload with Realistic Cross-Traffic. Talk at Bandwidth Estimation Workshop 2003. http://www.caida.org/workshops/isma/0312/slides/rprasad-best.pdf. Acessado em 6 de Janeiro de 2009.

Shriram, A., Murray, M., Hyun, Y., Brownlee, N., and Broido, A. (2005). Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links. In *Passive And Active Network Measurement: 6th International Workshop, PAM 2005*. Springer.

Silvester, J. A. (2005). CalREN: Advanced Network(s) for Education in California. http://isd.usc.edu/~jsilvest/talks-dir/20051021-cudi-merida.pdf. Acessado em 6 de Janeiro de 2009.

Tirumala, A., Cottrell, L., and Dunigan, T. (2003). Measuring end-to-end bandwidth with Iperf using Web100. Technical Report SLAC-PUB-9733, SLAC.

Wang, S. Y., Choua, C. L., and Lin, C. C. (2007). The Design and Implementation of the NCTUns Network Simulation Engine. *Simulation Modelling Practice and Theory*, 15(1):57–81.

Wolski, R., Spring, N. T., and Hayes, J. (1999). The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768.

Zanikolas, S. and Sakellariou, R. (2005). A Taxonomy of Grid Monitoring Systems. *Future Generation Computer Systems*, 21(1):163–188.