Adaptação de um Protocolo de Medição Ativa para Redes Tolerantes a Atrasos/Desconexões

Carlos H. Lamb[1], **Jéferson C. Nobre**[1]

[1]Escola Politécnica
Universidade do Vale do Rio dos Sinos (UNISINOS)
São Leopoldo, RS – Brasil

carloslamb@gmail.com, jcnobre@unisinos.br

Resumo. *Redes Tolerantes a Atrasos/Desconexões (Delay/Disruption Tolerant Networks - DTNs) podem ser caracterizadas por altos atrasos e desconexões entre os nós participantes. Apesar dessas características, as DTNs necessitam de funcionalidades de gerenciamento de redes semelhantes àquelas encontradas nas redes convencionais, como, por exemplo, monitoramento da latência na troca de mensagens. Uma das abordagens mais utilizadas para tal monitoramento é o emprego de protocolos padronizados para medições ativas, como, por exemplo, o Two-Way Active Measurement Protocol (TWAMP). No entanto, protocolos como o TWAMP não estão preparados para as DTNs, já que consideram ambientes de rede convencionais, como a Internet. O presente trabalho descreve o Delay/Disruption Two-Way Active Measurement Protocol (DTWAMP), uma proposta para a realização de medições ativas em DTNs. Além disso, experimentos foram realizados a fim de avaliar a solução proposta. Os resultados obtidos demonstram que o DTWAMP possui propriedades desejáveis para a execução de tarefas de monitoramento em DTNs.*

Abstract. *Delay/Disruption Tolerant Networks (DTNs) can be characterized by high delays and disconnections between the participating nodes. Despite these characteristics, DTNs need network management features similar to those found in conventional networks such as, for example, latency monitoring in message exchanges. One of the approaches commonly used for such monitoring is the use of standardized protocols for active measurements, such as, for example, the Two-Way Active Measurement Protocol (TWAMP). However, protocols like TWAMP are not prepared for DTNs because they consider conventional network environments, such as the Internet. This paper describes the Delay/Disruption Two-Way Active Measurement Protocol (DTWAMP), a proposal for the execution of active measurements in DTNs. In addition, experiments were performed to evaluate the proposed solution. The results demonstrate that DTWAMP has desirable properties to perform monitoring tasks in DTNs.*

1. Introdução

Redes de computadores empregam diferentes arquiteturas para realizar a comunicação entre os nós participantes. Essas arquiteturas consideram algumas premissas sobre a troca de mensagens. Por exemplo, a arquitetura TCP/IP, utilizada na Internet, foi desenvolvida para ambientes com baixa latência e pequena perda de pacotes [Oliveira et al. 2008]. Entretanto, essa arquitetura não apresenta um bom desempenho para todos os tipos de redes de comunicação.

Existem ambientes de rede, considerados desafiadores, que possuem a dificuldade de manter uma comunicação fim-a-fim com baixa latência e pequena perda de pacotes [Oliveira et al. 2008]. Uma das arquiteturas mais utilizada para esses ambientes é denominada de Redes Tolerantes a Atrasos/Desconexões (*Delay/Disruption Tolerant Networks* - DTNs) [Cerf et al. 2007]. Essa comunicação acaba influenciando em como os serviços da DTN são executados, entre eles, o seu gerenciamento.

Em uma DTN é fundamental saber o tempo que uma informação pode levar para chegar ao seu destino, assim como o retorno de sua resposta. Entretanto, devido a características como alta latência e desconexões frequentes, as ferramentas existentes para o gerenciamento de redes, principalmente de redes TCP/IP, dificilmente obteriam êxito no gerenciamento de uma DTN [Salvador et al. 2011].

O emprego de um protocolo de medição *two-way*, como o TWAMP (*Two-Way Active Measurement Protocol* - TWAMP) [Hedayat et al. 2008], pode nos fornecer informações detalhadas de métricas do *round-trip* entre dois nós. O TWAMP possui sua arquitetura e metodologia baseado no OWAMP (*One-way Active Measurement Protocol* - OWAMP) [Shalunov et al. 2006] para definir uma métrica de avaliação do desempenho *round-trip* entre os nós de uma rede. Esta avaliação é realizada a partir dos resultados obtidos durante a troca de mensagens de teste entre dois nós. O TWAMP também possui uma versão reduzida, denominada de TWAMP *Light*, em que não é necessária a troca de mensagens para o estabelecimento da conexão entre os nós. Nesta versão é executada apenas as trocas de mensagens de teste entre os nós. No entanto, esses protocolos foram projetados para operar em uma rede TCP/IP e seu uso direto em uma DTN dificilmente apresentariam os resultados esperados.

Neste trabalho é proposto uma extensão do TWAMP para fornecer medições *two-way* entre dois nós DTN. Para isso, foi desenvolvida uma aplicação para adaptar o TWAMP *Light* em um ambiente DTN que informa os tempos de cada direção no *round-trip* existente entre dois nós DTN. A avaliação desta aplicação comprovou a eficácia da mesma em ambientes normais e em ambientes com atrasos.

O presente artigo é organizado da seguinte forma. Na Seção 2 é apresentado o referencial teórico. A solução proposta é descrita na Seção 3. Na Seção 4 é descrita a avaliação da solução proposta. A Seção 5 apresenta os trabalhos relacionados e por fim, a Seção 6 traz as conclusões e os trabalhos futuros.

2. Referencial Teórico

Esta seção apresenta os principais conceitos teóricos necessários ao desenvolvimento deste trabalho. Estes conceitos estão divididos em duas subseções. Na subseção gerenciamento de rede em DTN e seus desafios são apresentados os fundamentos teóricos sobre DTNs, os conceitos sobre o *bundle protocol* e também os desafios de gerenciamento em DTNs. Esta seção é encerrada com a apresentação sobre medições ativas.

2.1. Gerenciamento de Rede em DTN e seus Desafios

As DTNs se diferenciam de redes tradicionais, como a Internet, devido a características atípicas que podem ser encontradas em seu ambiente. Essas características acabam influenciando em como os serviços de uma DTN são executados.

As características mais comuns encontradas em uma DTN são a alta latência durante a comunicação entre os nós e as desconexões durante a comunicação dos nós. Com

isso, as mensagens encaminhadas entre os nós DTN podem levar horas ou dias até chegarem ao seu destino, podendo inclusive serem descartadas antes da entrega ao seu destinatário devido ao tempo de expiração que a mensagem possuir [Fall 2003].

A arquitetura DTN é proposta na RFC 4838 [Cerf et al. 2007]. A arquitetura DTN combina a utilização da técnica de comutação das mensagens e o armazenamento persistente dos dados, definindo uma sobrecamada (*overlay*) denominada de *bundle layer*, localizada entre a camada de transporte e a camada de aplicação. Os dispositivos que empregam a *bundle layer* são considerados como nós DTN, sendo que as mensagens transportadas pelos nós são chamadas de *bundles* [Cerf et al. 2007].

A *bundle layer* possui um protocolo de agregação (*Bundle Protocol* - BP) especificado na RFC 5050 [Scott and Burleigh 2007] que define o formato dos *bundles* assim como sua utilização. Esse protocolo deve ser executado em todos os nós pertencentes à DTN. O formato dos *bundles* é definido por um bloco primário obrigatório, um bloco opcional de *payload* e um conjunto de blocos de extensão opcionais [Cerf et al. 2007]. Um *bundle* deve ser uma sequência concatenada que contenha pelo menos dois desses blocos, sendo o bloco primário o primeiro dessa sequência [Scott and Burleigh 2007].

A implementação de uma DTN pode ser um desafio para um ambiente de testes de estudos práticos. Para isso, o uso de implementações de referência, como o DTN2 [DTNRG 2014], auxiliam no estudo das DTNs. O DTN2 implementa o *Bundle Protocol* definido pela RFC 5050 [Scott and Burleigh 2007]. O agente do *bundle protocol* e todo seu código de suporte é implementado como um *daemon* chamado de dtnd. As aplicações DTN interagem com o dtnd através de uma API, que é um mecanismo *Sun* RPC. A comunicação entre os nós DTN no DTN2 é configurada através do arquivo *dtn.conf* de cada nó [DTNRG 2014].

Assim como em qualquer ambiente de rede, uma DTN também possui a necessidade de gerenciamento/monitoramento de rede. Entretanto, as soluções de gerenciamento existentes, como o SNMP (*Simple Network Management Protocol* - SNMP) e o NETCONF (*Network Configuration Protocol* - NETCONF), dificilmente apresentarão os mesmos resultados em uma DTN do que em uma rede tradicional, como a Internet. Ambos são exemplos de protocolos que não podem ser empregados em ambientes em que não exista uma comunicação fim-a-fim contínua, como uma DTN. Embora existam trabalhos, como de [Isento et al. 2012] e de [Kumar et al. 2010], que apresentam soluções para o gerenciamento de DTNs baseadas nesses protocolos, os mesmos foram desenvolvidos para cenários DTN específicos ou ainda necessitam de implementação.

Em uma DTN não é possível estabelecer sessões de transporte fim-a-fim e não é possível para a aplicação de gerenciamento estar totalmente ciente das operações locais das condições de rede do dispositivo remoto a ser gerenciado. O SNMP e o NETCONF também necessitam estabelecer *loops* de controle através da infraestrutura de rede, sendo que a alta latência e a conectividade intermitente de uma DTN dificultam o estabelecimento desses *loops*, assim como na utilização de ferramentas de gerenciamento [Nobre et al. 2013]. Essas restrições são fundamentais para definir como as ferramentas de gerenciamento devem definir, transmitir e receber seus dados [Birrane and Cole 2010].

2.2. Medições Ativas

O TWAMP é um protocolo especificado na RFC 5357 [Hedayat et al. 2008]. Sua arquitetura e metodologia são baseadas no OWAMP, porém, enquanto que o OWAMP mede

características unidirecionais, como atraso de uma direção, o TWAMP define a medição de ida e volta (*two-way*), através de *timestamps*, entre a comunicação de dois *hosts*. Geralmente o TWAMP é composto de apenas dois *hosts* com funções específicas para cada um, sendo o nó cliente denominado de *controller* e o nó servidor denominado de *responder*.

O TWAMP é formado por dois protocolos, o TWAMP-*Control* e o TWAMP-*Test*. No TWAMP-*Control* são realizadas no mínimo oito trocas de mensagens entre o *controller* e o *responder* para o estabelecimento da conexão entre os nós e a definição de uma sessão de teste entre os mesmos. Após a conexão, a sessão de teste é iniciada e o TWAMP-*Test* é responsável pela transmissão de mensagens de testes do mesmo tamanho entre os nós em ambas as direções. Os resultados obtidos durante esta troca de mensagens serão utilizados para avaliar o desempenho de cada direção no *round-trip* entre os nós [Hedayat et al. 2008].

O TWAMP também possui uma versão reduzida, denominada de TWAMP *Light*, na qual não é necessária a troca de mensagens para o estabelecimento da conexão entre os nós pelo TWAMP-*Control*. Conforme a RFC 5357 [Hedayat et al. 2008], o TWAMP *Light* não especifica como é realizado o estabelecimento da conexão entre o *controller* e o *responder* e o início de uma sessão de teste entre os mesmos. O TWAMP *Light* assume que já exista uma conexão estabelecida entre os mesmos. Nesta versão é executada apenas as trocas de mensagens de teste entre os nós através do TWAMP-*Test* [Hedayat et al. 2008].

3. DTWAMP

O presente trabalho propõe uma extensão do TWAMP para fornecer medições *two-way* em DTNs. Nessas redes, a entrega de uma mensagem ao seu destino pode levar horas, existindo a possibilidade de inclusive exceder o tempo de vida dessa mensagem. Isso acaba influenciando em como os nós DTN se comunicam entre si, assim como em serviços como o gerenciamento de rede.

A extensão do TWAMP para DTNs pode ser útil para calcular o desempenho de métricas como atraso, perda de *bundles*, duplicação de *bundles*, etc. entre dois nós DTN. Com o conhecimento, por exemplo, do atraso existente de cada direção na comunicação entre os nós DTN, as demais aplicações DTN podem definir o tempo de vida que uma mensagem deve possuir.

Este trabalho propõe como solução para calcular o desempenho *round-trip* entre dois nós DTN e informando o tempo de cada direção, a adaptação do TWAMP *Light* para o ambiente de uma DTN. Esta solução é chamada de DTWAMP e, assim como o TWAMP *Light*, o mesmo é composto de um nó chamado *controller* que envia mensagens de teste para um nó chamado *responder*.

As seguintes premissas são consideradas durante este trabalho: O fato de o TWAMP *Light* não possuir o TWAMP-*Control* para estabelecer a conexão entre os nós, como a versão completa do TWAMP, impede a utilização das mensagens de teste do modo autenticado ou do modo criptografado. A RFC 5357 [Hedayat et al. 2008] apenas informa que o TWAMP *Light* deveria possibilitar a utilização destes modos, mas não cita como fazê-los. Por isso, o DTWAMP trabalha com as mensagens de teste do modo sem autenticação do TWAMP, não aplicando nenhum controle de segurança nas mensagens de teste trocadas entre os nós. Com relação às métricas de desempenho, o atraso, por ser

uma característica típica de uma DTN, é a métrica escolhida para ser demonstrada pelo DTWAMP neste trabalho.

O TWAMP apresenta desafios maiores para sua adaptação em um ambiente DTN do que o TWAMP *Light*. Em um ambiente que tem por característica altos atrasos e desconexões, as ferramentas de gerenciamento são incapazes de estabelecer efetivamente os *loops* de controle nesse ambiente [Birrane and Cole 2010]. Da mesma maneira, podemos comparar que a troca de mensagens utilizada pela versão completa do TWAMP para apenas o estabelecimento da conexão entre os nós, pode dificultar sua adaptação para uma DTN.

A versão reduzida do TWAMP, TWAMP *Light*, não apresenta a mesma dificuldade de adaptação do estabelecimento de conexão entre os nós do que o TWAMP. Como o TWAMP *Light* assume que já exista uma conexão estabelecida entre o *controller* e o *responder*, o mesmo realiza apenas as trocas de mensagens de teste entre os nós para avaliar o desempenho *round-trip*. Como a RFC 5357 [Hedayat et al. 2008] não especifica como deve ser essa conexão, este trabalho assume que a configuração para a comunicação dos nós DTN no DTN2 seja suficiente para tal conexão no DTWAMP. Além disso, a fim de realizar também o mínimo possível de troca de mensagens entre os nós, o DTWAMP baseia-se no TWAMP *Light*.

O transporte das mensagens de teste do DTWAMP é outra premissa considerada neste trabalho. Os nós DTN transportam todas as mensagens que são trocadas entre os mesmos através de *bundles*. Desta forma, para realizar a troca de mensagens de teste do TWAMP-*Test* em uma DTN é necessário que essas mensagens sejam adaptadas para o formato de *bundles*. Como os dados de um *bundle* são transmitidos através do campo *block body data* do bloco de *payload*, é necessário que os dados das mensagens de teste do TWAMP-*Test* sejam enviados apenas através deste campo. Por esse motivo, não é preciso realizar nenhuma alteração no bloco primário do *bundle*. Entretanto, no DTWAMP, cada mensagem de teste trocada entre os nós DTN necessita ser enviada através de um *bundle* exclusivo.

O TWAMP estabelece que as mensagens de teste trocadas entre o *controller* e o *responder* durante o TWAMP-*Test* utilizem campos com tamanhos fixos [Hedayat et al. 2008]. O motivo dos campos terem tamanhos fixos se deve pelo fato de o TWAMP determinar que as mensagens de teste utilizadas durante a sessão de teste entre os nós possuam tamanhos iguais, garantindo que mensagens com o mesmo tamanho sejam utilizadas nas duas direções para o cálculo do desempenho do *round-trip*. Enquanto isso, o *Bundle Protocol* não especifica o tamanho do campo *block body data* do bloco de *payload* do *bundle*, apenas informa dizendo que o mesmo é de tamanho variável [Scott and Burleigh 2007]. Por esse motivo, o DTWAMP utiliza tamanhos fixos nos dados do *controller* e do *responder* que serão transmitidos pelo *block body data* do bloco de *payload* do *bundle*. Dessa forma, como o *responder* possui uma quantidade maior de informações para fornecer, o *controller* possui um campo específico (*packet padding*) para igualar o seu *payload* com o *payload* do *responder*. Ambos *payloads* destes *bundles* são comparados pelo DTWAMP para verificar se os mesmos possuem tamanhos idênticos.

3.1. Estrutura das Mensagens

O *controller* troca mensagens com o *responder* para fornecer o tempo de cada direção do *round-trip* entre dois nós DTN. Essas mensagens são transportadas através de *bundles*

como se fossem as mensagens de teste utilizadas pelo TWAMP *Light*. No ambiente DTN, essas mensagens de teste, que são encaminhadas pelo bloco de *payload* do *bundle*, podem ser visualizadas na Figura 1.

A mensagem de teste especificada na Figura 1 é o bloco de *payload* do *bundle* encaminhado pelo nó *controller* e pelo nó *responder* do DTWAMP. Neste bloco, os campos *Block Type*, *Proc. Flags* e *Block Lenght* (representados em fundo verde na Figura 1) são os campos originais do bloco de *payload* de um *bundle* conforme a RFC 5050 [Scott and Burleigh 2007]. Esses campos são utilizados pelo *controller* e pelo *responder* em conjunto com a implementação DTN2 .

Os campos *Sequence Number*, *Timestamp* e *Packet Padding* pertencentes ao *controller* (representados em fundo cinza na Figura 1(a)) são transmitidos como os dados de *payload* do *bundle*. Estes campos foram adaptados para o ambiente DTN com base no TWAMP e são preenchidos automaticamente pelo nó *controller* do DTWAMP. O campo *Sequence Number* é um número inteiro começando por 1 que é incrementado conforme a quantidade de *bundles* enviados. O campo *Timestamp* possui o tempo registrado antes do envio do *bundle* para o nó *responder*. O campo *packet padding* possui o valor fixo de um *char* de [20] para poder igualar o *payload* do *controller* ao *payload* do *responder*, que são de 32 *bytes*.

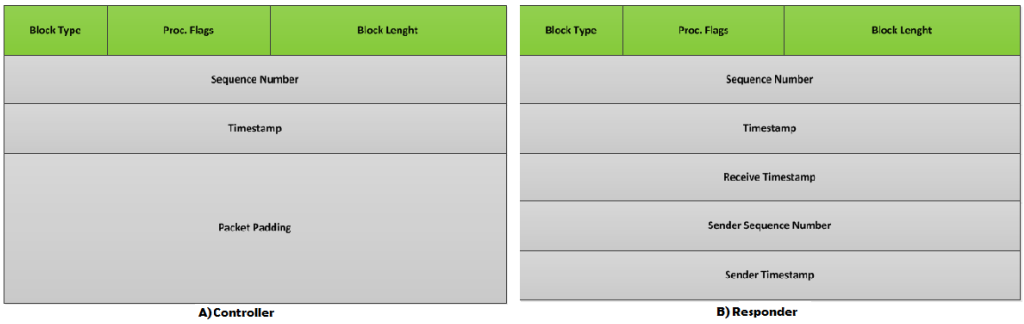

Figura 1. Mensagem de teste encaminhada pelos nós *controller* e *responder*

Os campos *Sequence Number*, *Timestamp*, *Receive Timestamp*, *Sender Sequence Number* e *Sender Timestamp* pertencentes ao *responder* (representados em fundo cinza na Figura 1(b)) também foram adaptados com base no TWAMP, sendo transmitidos como os dados de *payload* do *bundle*. Assim como no *controller*, estes campos são preenchidos automaticamente pelo nó *responder* do DTWAMP. O campo *Sequence Number* copia o mesmo valor do *Sequence Number* do *controller*. O campo *Timestamp* possui o tempo registrado antes do envio do *bundle* para o nó *controller*. O campo *Receive Timestamp* possui o tempo registrado quando o *bundle* do *controller* é recebido. Os campos *Sender Sequence Number* e *Sender Timestamp* possuem os valores informados pelo nó *controller*.

Os campos utilizados pelo *controller* e pelo *responder* como dados de *payload* do *bundle* são baseados nas mensagens de teste do modo sem autenticação do TWAMP. Para o ambiente DTN não foram utilizados todos os campos dessas mensagens de teste do TWAMP. Os campos *error estimate*, *Sender* TTL e MBZ não são utilizados pelo DTWAMP.

O campo *error estimate* não é utilizado, tanto no *controller* como no *responder*, em virtude deste campo ser um cálculo que mostra a estimativa de erro e de sincronização

entre as mensagens de teste. Como em um ambiente DTN pode haver oscilações constantes durante a comunicação entre os nós, como alta latência e desconexões, no momento o DTWAMP não utiliza este campo para o cálculo do desempenho *round-trip*.

O *Sender* TTL, utilizado pelo *responder*, é outro campo que não foi utilizado no DTWAMP. Este campo é definido conforme o *time to live* do pacote recebido do cabeçalho IP pelo nó *responder* do TWAMP [Hedayat et al. 2008]. Como o bloco primário de cada *bundle* possui um campo (*lifetime*) que determina o tempo de vida útil do *payload* do *bundle* [Scott and Burleigh 2007], o *Sender* TTL não foi adaptado para o DTWAMP.

Os campos MBZ, utilizados pelo *responder*, por serem campos compostos por zeros e ignorados pelo *controller*, também não são utilizados pelo DTWAMP para o cálculo do desempenho *round-trip*. Com relação ao *packet padding*, o mesmo é utilizado apenas pelo *controller* a fim de igualar seu *payload* com o *responder* e de se transmitir o mínimo de dados necessários para o cálculo do desempenho *round-trip*.

O DTWAMP foi projetado tendo o nó *controller* como o responsável por iniciar a troca de mensagens através de *bundles* com o nó *responder*, além de calcular o *round-trip* e exibir os resultados obtidos de cada direção. O nó *responder* é responsável por enviar um *bundle* de resposta para o nó *controller* a cada *bundle* recebido do mesmo.

A arquitetura DTN, por padrão, também utiliza *timestamps*, pois ela depende da sincronização entre os nós DTN para executar atividades como o roteamento entre os nós e a exclusão de *bundles* de acordo com seu tempo de expiração [Cerf et al. 2007]. Embora tanto o TWAMP [Hedayat et al. 2008] como o OWAMP [Shalunov et al. 2006] definem os *timestamps* como o número de segundos a partir do início do ano 1900, o DTWAMP, por se tratar de uma aplicação para DTNs, utiliza o *timestamp* definido pela RFC 5050 [Scott and Burleigh 2007] que emprega o número de segundos a partir do início do ano 2000.

3.2. Implementação

O DTN2 é o ambiente para o qual foram desenvolvidas as aplicações para avaliar o desempenho do DTWAMP. Este trabalho realizou modificações na aplicação *dtnperf* para implementar o DTWAMP. O *dtnperf* é uma aplicação desenvolvida em C, de código aberto e é disponibilizada através da implementação DTN2, sendo constituída por um *server* e um *client*, que enviam *bundles* entre os mesmos para fornecer o *goodput* de uma DTN. A versão 2.8 do *dtnperf-client* e a versão 2.5.1 do *dtnperf-server* foram utilizadas como base para as aplicações do DTWAMP que foram desenvolvidas (*controller* e *responder*) para calcular o *round-trip* entre dois nós DTN da implementação DTN2 baseado nos conceitos do TWAMP *Light*.

A aplicação *controller* do DTWAMP fornece dois modos de operações para informar os tempos de cada direção do *round-trip* entre dois nós DTN. A primeira opção é denominada como *time mode*. Nesse modo é especificado, através do argumento –t, por quantos segundos são trocados *bundles* entre os nós. Caso o tempo informado se encerre durante a transmissão de um *bundle*, a mesma continua até que o nó *controller* receba este *bundle*. O segundo modo de operação é chamado de *bundle mode*. Neste modo é informado, através do argumento –b, quantos *bundles* são trocados entre os nós. As aplicações desenvolvidas exigem a interação do usuário apenas para determinar o modo de operação a ser usado na aplicação *controller*, assim como a definição de um valor para o modo escolhido.

No final de cada troca de *bundle* entre os nós, independentemente do modo de operação escolhido, é informado na tela do *controller* o tempo em milissegundos de cada direção, além do *round-trip total* do *controller*. Em resumo, o cálculo do desempenho *round-trip* entre dois nós DTN realizado pelo DTWAMP pode ser visualizado através da Figura 2.

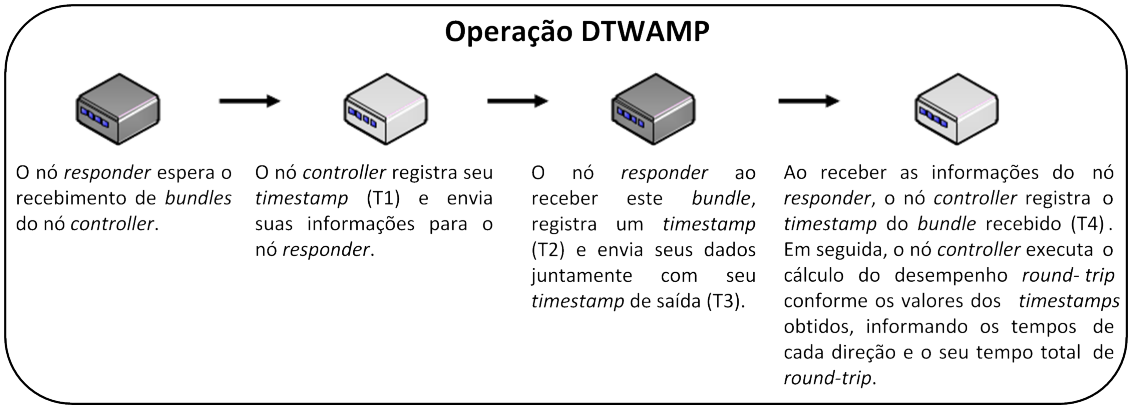

Figura 2. Operação do DTWAMP

A operação do DTWAMP termina quando o número de *bundles*, ou o tempo informado, se encerra. Após é calculado uma média dos resultados obtidos durante toda a operação do DTWAMP, informando a média dos tempos de cada direção e do *round-trip* total do *controller*. Além disso, é informado também o tempo total da operação em segundos e o número de *bundles* enviados durante a operação.

4. Avaliação

O objetivo da avaliação apresentada nesta seção é demonstrar a eficácia da implementação do DTWAMP desenvolvida em cenários distintos ao realizar o cálculo *round-trip* entre dois nós DTN, informando a latência existente de cada direção entre os mesmos. Inicialmente é descrito o ambiente utilizado durante a avaliação. Em seguida, são realizados experimentos com o objetivo de avaliar a eficácia da solução proposta em cenários com latências distintas. Por fim, são analisados os experimentos que foram realizados com tráfego entre os dois nós DTN.

4.1. Ambiente

O ambiente DTN que foi utilizado na avaliação é a implementação DTN2. Na avaliação deste trabalho foi utilizada a versão dtn-2.9.0 do DTN2, e para seu *framework* foi utilizada a versão oasys-1.6.0. Ambos foram instalados em duas VMs (máquinas virtuais) com o *Ubuntu* 11.10. Estas duas VMs são os *hosts* do ambiente DTN utilizado. A configuração realizada em cada uma das VMs segue a configuração padrão disponibilizada nos tutoriais do DTNRG [DTNRG 2014].

Os experimentos com latências foram realizados através da ferramenta *netem*[1], especificando atrasos na interface de rede *eth1* das VMs. Com estas latências, foram realizados experimentos com atrasos a partir da VM1, atrasos a partir da VM2 e atrasos em ambas as VMs. Para garantir a sincronização entre os nós DTN do ambiente, foram instaladas em ambas as VMs o NTP (*Network Time Protocol* - NTP).

[1]http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

4.2. Experimentos e Resultados

Os dois primeiros experimentos realizados determinaram o modo de operação da aplicação *controller* a ser utilizado para os experimentos com latências. Estes dois experimentos foram realizados utilizando as configurações padrões do DTN2 e das VMs. A Figura 3 apresenta uma comparação entre o *bundle mode*, identificando quantos segundos passaram para serem enviados os *bundles* especificados, e o *time mode*, informando quantos *bundles* foram enviados até o término da operação do DTWAMP.

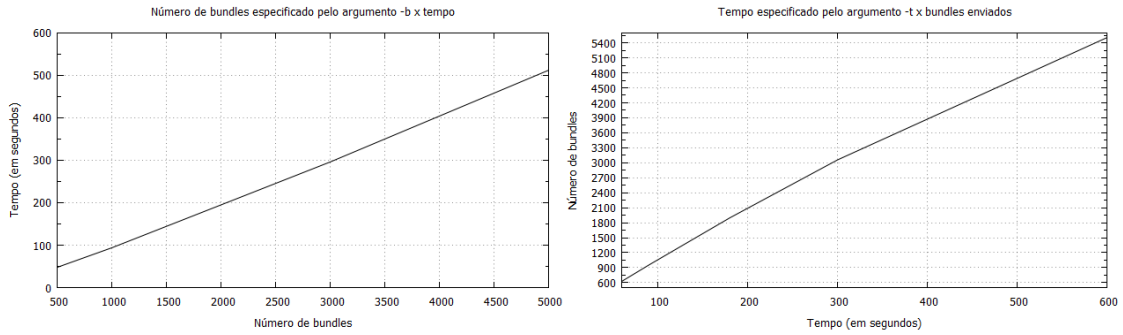

Figura 3. Comparação entre os modos de operações do DTWAMP

A análise dos resultados apresentados pela Figura 3 demonstra que o modo de operação utilizado não afeta no desempenho da operação do DTWAMP, pois os mesmos podem ser considerados como equivalentes. Tanto para estes dois primeiros experimentos, como para os demais, o nó *controller* foi executado na VM1 enquanto que o nó *responder* foi executado na VM2.

Os experimentos seguintes foram realizados utilizando o *time mode* com 600 segundos. O *time mode* foi escolhido pelo fato de os experimentos serem feitos com atrasos nas interfaces de rede das VMs. Dessa forma, é possível determinar o tempo de duração do experimento, enquanto que se fosse especificado uma determinada quantidade de *bundles* a serem enviados, não haveria como estimar o tempo de execução da operação.

O primeiro experimento do DTWAMP com *time mode* em 600 segundos foi executado sem nenhuma latência no ambiente entre os dois nós DTN a fim de comparar com os experimentos posteriores envolvendo latência. As médias dos resultados obtidos deste experimento podem ser visualizadas na Figura 4, na coluna indicada como 0. Neste experimento, assim como nos demais experimentos com latências, os mesmos foram repetidos dez vezes, sendo que a variabilidade dos resultados obtidos foi mínima.

Os experimentos posteriores do DTWAMP com *time mode* em 600 segundos foram executados com a existência de latência entre os nós DTN. Os atrasos utilizados nos experimentos começaram com 500 milissegundos (ms). Após, foi incrementado em cada experimento seguinte mais 500 ms, até a realização do último experimento com latência, que foi com 3.500 ms. Durante estes experimentos com latências não havia nenhum outro tipo de tráfego na rede entre as VMs.

A Figura 4 apresenta um gráfico com os resultados obtidos dos experimentos que foram realizados com atrasos especificados na interface de rede *eth1* da VM1 e da VM2. Nesta figura, a sigla C-R representa o tempo em milissegundos do nó *controller* até o nó *responder*. A sigla R-C representa o tempo em milissegundos do nó *responder* até o nó

controller. Por fim, a sigla C-C apresenta o tempo do total do *round-trip* do *controller* em milissegundos.

Nos experimentos com latências podemos verificar que os tempos apresentados estão conforme o esperado. Se não considerarmos a latência existente durante os experimentos entre os nós DTN, pode-se dizer que os tempos apresentados são similares ao encontrados na execução normal.

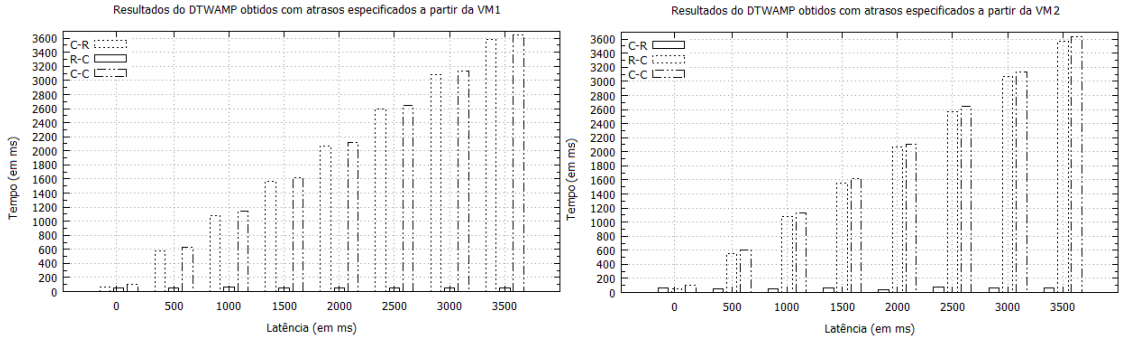

Figura 4. Resultados do DTWAMP obtidos com atrasos especificados a partir da VM1 e da VM2

A comparação entre os resultados da Figura 4 podem ser considerados como equivalentes, visto que a diferença entre os atrasos das VMs é mínima. A maior diferença entre o tempo de saída do *bundle* do *controller* até o *responder* do experimento a partir da VM1 com o tempo de saída do *bundle* do *responder* até o *controller* do experimento a partir da VM2 é encontrado no experimento com 2.500 ms de latência. Entretanto, mesmo neste experimento, os tempos obtidos na VM1 (2594 ms) possuem uma variabilidade pequena, sendo apenas 1,21% maiores do que os encontrados na VM2 (2563 ms). Com relação ao *round-trip* do *controller*, a maior diferença é encontrada no experimento com latência de 500 ms. Neste experimento, os tempos obtidos na VM1 (630 ms) são somente 3,45% maiores do que os encontrados na VM2 (609 ms).

Para analisar o comportamento do DTWAMP em ambientes com atrasos nas duas direções foram realizados os mesmos experimentos com latências nas duas VMs. Os resultados destes experimentos estão representados na Figura 5(a).

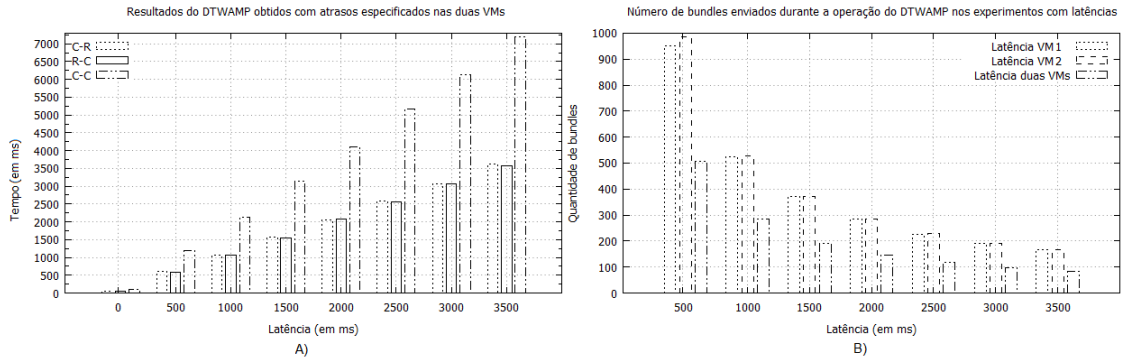

Figura 5. Resultados do DTWAMP obtidos com atrasos especificados nas duas VMs e a quantidade de *bundles* enviados durantes os experimentos com atrasos

Os resultados da Figura 5(a) apresentam tempos dentro do esperado para cada direção. Se compararmos os tempos de saída do *bundle* do *controller* até o *responder* da Figura 5(a) com o experimento a partir da VM1, a diferença é mínima entre os mesmos. A maior diferença é encontrada no experimento com 3.500 ms de latência. Neste experimento, os tempos obtidos na Figura 5(a) (3628 ms) são apenas 1,09% maiores do que os encontrados no experimento com latência apenas na VM1 (3589 ms). Com relação aos tempos de saída do *bundle* do *responder* até o *controller* do experimento a partir da VM2, a maior diferença, comparando com a Figura 5(a), é encontrada no experimento com 500 ms de latência. Neste experimento, os tempos obtidos na Figura 5(a) (577 ms) são 4,15% maiores do que os encontrados no experimento com latência apenas na VM2 (554 ms).

A Figura 5(a) também apresenta os tempos esperados para o *round-trip* do *controller*. Como existe latência nas duas direções, podemos comparar o *round-trip* do *controller* nos experimentos com latências de 500 ms, 1.000 ms e 1.500 ms da Figura 5(a) com o *round-trip* encontrado nos experimentos com latência de 1.000 ms, 2.000 ms e 3.000 ms respectivamente da Figura 4. Nessas comparações, a maior diferença encontrada é com relação ao experimento com 500 ms de latência da Figura 5(a) (1182 ms) e 1.000 ms de latência do experimento com latência apenas na VM2 (1135 ms). Nesta comparação, o tempo do *round-trip* do *controller* na Figura 5(a) é 4,14% maior do que o encontrado no experimento da VM2.

O número de *bundles* enviados durante os experimentos com atrasos também foi diminuindo devido ao aumento da latência. A Figura 5(b) apresenta o número de *bundles* enviados durante a operação do DTWAMP em cada experimento com latência realizado.

Na Figura 5(b) é possível observar que o número de *bundles* enviados durante os experimentos da Figura 4 são equivalentes. Apenas no experimento com 500 ms de latência, a VM2 enviou 32 *bundles* a mais (3,36%) que a VM1. Entretanto, se compararmos com os experimentos da Figura 5(a) (atrasos nas duas VMs), o número de *bundles* enviados diminuiu 48,37% (476 *bundles* a menos). Este número é esperado, visto que existe latência nas duas direções.

Os resultados demonstrados nas Figuras 4 e 5, em resumo, comprovam que o DTWAMP pode ser executado em ambientes com latências, desde que sejam suportadas pela configuração padrão do DTN2. Para latências maiores, é preciso verificar a necessidade de configurações adicionais do DTN2. Além disso, com o conhecimento do tempo estimado de envio e recebimento de uma mensagem entre dois nós DTN, as demais aplicações DTN podem definir o tempo de vida que essa mensagem deve possuir. Caso não seja possível alterar o tempo de vida da mensagem, as aplicações DTN também podem analisar se a mensagem pode ser enviada e/ou recebida dentro do tempo estimado, evitando assim tráfego desnecessário.

Os experimentos apresentados até o momento foram realizados em condições normais ou com simulações de atraso. Os experimentos apresentados a seguir foram realizados com o objetivo de demonstrar a operação do DTWAMP quando há tráfego na rede. O tráfego de rede considerado para os experimentos a seguir foi a transferência de um arquivo de 1 GB entre os dois nós DTN. Antes da realização dos experimentos foi calculado o tempo médio de transferência deste arquivo em duas situações: transferência do arquivo através da aplicação *dukto*[2] pela rede TCP/IP e a transferência do arquivo como *bundle*,

[2]http://www.msec.it/dukto

através das aplicações *dtncp* e *dtncpd*, disponibilizadas com o DTN2. O tempo médio da transferência do arquivo pela aplicação *dukto* foi de 40 segundos, enquanto como *bundle* foi de 70 segundos. Uma vez que a transferência do arquivo seja feita através de *bundles*, é esperado que o tempo fosse maior e a quantidade de *bundles* enviados fosse menor do que a transferência do arquivo pela rede TCP/IP.

Os dois primeiros experimentos foram realizados com o DTWAMP no *time mode* com 40 e 70 segundos respectivamente, em condições normais, sendo que os seus resultados podem ser visualizados na Figura 6. Os experimentos seguintes foram executados da seguinte maneira: Na VM1, foi realizada a transferência do arquivo a partir da mesma para a VM2 através do *dukto* e executado o DTWAMP com *time mode* de 40 segundos. Em seguida foi realizada a transferência do arquivo através do *dtncp* e executado o DTWAMP com *time mode* de 70 segundos. Em ambos os experimentos o DTWAMP foi executado paralelamente com a transferência do arquivo entre os nós. Posteriormente foram realizados os mesmos experimentos a partir da VM2. Os resultados encontrados podem ser visualizados na Figura 6.

Nos experimentos realizados tanto a partir da VM1 como a partir da VM2, pode-se observar que os tempos médios apresentados possuem diferenças conforme o modo de transferência utilizado em cada experimento. Além disso, pode-se notar que a transferência por *bundle* a partir da VM2, em comparação com a VM1, teve o tempo total do *round-trip* do *controller* 86,93% maior. Além disso, a quantidade de *bundles* enviados foi 40,22% menor.

Resultados obtidos durante a transferência de arquivo da VM1 para a VM2

	Time mode 40	Transferência pelo *dukto*	Transferência por *bundle*	*Time mode* 70
Controller-Responder	46 ms	93 ms	511 ms	47 ms
Responder-Controller	35 ms	258 ms	261 ms	33 ms
Controller-Controller	83 ms	352 ms	773 ms	80 ms
Bundles enviados	487 *bundles*	115 *bundles*	92 *bundles*	867 *bundles*

Resultados obtidos durante a transferência de arquivo da VM2 para a VM1

	Time mode 40	Transferência pelo *dukto*	Transferência por *bundle*	*Time mode* 70
Controller-Responder	46 ms	253 ms	472 ms	47 ms
Responder-Controller	35 ms	71 ms	972 ms	33 ms
Controller-Controller	83 ms	325 ms	1445 ms	80 ms
Bundles enviados	487 *bundles*	124 *bundles*	55 *bundles*	867 *bundles*

Figura 6. Comparação entre os experimentos de transferência de arquivo a partir da VM1 e da VM2

Os experimentos realizados durante a transferência de arquivo como *bundle* demonstram que o nó que está enviando o arquivo possui seu tempo mais afetado do que aquele que está recebendo o arquivo como *bundle*. Enquanto que na transferência pelo *dukto*, através da rede TCP/IP, ocorre o contrário, o nó que está recebendo o arquivo possui seu tempo mais afetado do que aquele que está enviando o arquivo. Este experimento pode-se considerar também como um experimento em que houve um aumento de processamento pelos nós, visto que o arquivo não foi transferido pela rede DTN.

5. Trabalhos Relacionados

Nesta seção são apresentadas brevemente duas aplicações que são disponibilizadas com a implementação DTN2. Estas aplicações podem ser caracterizadas por utilizarem medições ativas para suas métricas.

O *dtnping* possui as características do comando *ping* das redes TCP/IP. Ambos são considerados como uma ferramenta OWAMP para medição de características

de rede, podendo enviar um *bundle* para um nó de destino válido ou para o *localhost* [Davies and Doria 2010].

O *dtnperf* é uma aplicação para DTN desenvolvida por Piero Cornice similar ao *iperf*, fornecendo o *goodput* existente entre dois nós DTN. O *dtnperf* é constituído de duas partes: *dtnperf-client* e *dtnperf-server*. Enquanto que o *dtnperf-client* realiza o cálculo do *goodput* e disponibiliza as opções, como o modo a ser usado, o *dtnperf-server* apenas repassa as informações solicitadas pelo *dtnperf-client*.

As aplicações apresentadas nesta seção descrevem brevemente suas funcionalidades. Essas aplicações também possuem suas limitações, não possibilitando a medição de novas métricas por exemplo. Além disso, com relação a medição de tempos, tanto o *dtnping* como o *dtnperf* trabalham com a *flag* de recebimento de um *bundle* enviado a um determinado nó DTN, enquanto que o DTWAMP trabalha com *timestamps* durante a comunicação de dois nós DTN.

6. Conclusão

O presente trabalho apresentou uma proposta de extensão do TWAMP para fornecer medições *two-way* para DTNs, através da adaptação dos conceitos do TWAMP *Light*. Inicialmente foram apresentados os conceitos para o entendimento das DTNs e os desafios existentes no gerenciamento dessas redes. Um dos principais desafios no gerenciamento de DTNs é a falta de uma conexão fim-a-fim entre os nós, sendo este um dos principais motivos para as ferramentas tradicionais de gerenciamento não serem empregadas diretamente em uma DTN.

As aplicações de monitoramento existentes para DTN, como o *dtnping*, fornecem medições de determinadas características da rede. Entretanto, essas aplicações existentes são consideradas como OWAMP. As aplicações para demonstrar a solução proposta deste trabalho foram desenvolvidas com base na aplicação *dtnperf* para a implementação DTN2. O DTWAMP integra os conceitos do TWAMP *Light* para calcular os tempos de cada direção no *round-trip* entre dois nós DTN.

Para demonstrar a eficácia da solução proposta, foram realizados experimentos com simulações de atrasos a fim de comprovar o funcionamento do DTWAMP em ambientes característicos de uma DTN. Além disso, foram realizados experimentos para verificar como o DTWAMP reagiria quando houvesse outros tipos de *bundles* sendo transportados pelos nós DTN. Após os experimentos realizados foi constatado que o DTWAMP apresentou de maneira precisa os tempos de cada direção no *round-trip* existente entre dois nós DTN. Desta forma, o DTWAMP pode auxiliar as demais aplicações DTN. Se as demais aplicações DTN souberem o tempo estimado de uma mensagem, as mesmas podem definir o tempo de vida dessa mensagem, ou ainda, analisar se a mensagem pode ser enviada e/ou recebida, evitando assim tráfego desnecessário.

Apesar de o DTWAMP apresentar resultados satisfatórios nas avaliações realizadas, novas funcionalidades podem aprimorá-lo. Por exemplo, em trabalhos futuros, é possível incluir a adaptação das mensagens de teste do TWAMP-*Test* do modo autenticado e do modo criptografado para o DTWAMP. O modo autenticado possibilitaria que apenas os nós que fossem autenticados trocassem mensagens de teste entre si, enquanto que o modo criptografado possibilitaria a segurança do conteúdo transferido entre os nós. Com relação ao transporte das mensagens de teste, o DTWAMP envia as mesmas através

de um *bundle* exclusivo. Entretanto, pode haver determinados cenários DTN onde não seja possível realizar este transporte exclusivo. Para isso, o DTWAMP poderia adaptar suas mensagens de teste para serem transmitidas juntamente com outros dados de *bundles* prioritários. O DTWAMP também não foi avaliado em ambientes que tenham desconexões frequentes. Um experimento que mostrasse os resultados do DTWAMP nessas condições poderia comprovar que a aplicação pode ser adaptada para utilização em diversos cenários DTN. Por fim, a solução proposta comprovou que o TWAMP *Light* pode ser adaptado para uma DTN. Com isso, pode ser estudada uma maneira de se adaptar o TWAMP nestas redes.

Referências

Birrane, E. and Cole, R. G. (2010). Management of disruption-tolerant networks: A systems engineering approach. In *SpaceOps 2010 Conference*.

Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H. (2007). Delay-tolerant networking architecture. rfc 4838 (informational).

Davies, E. and Doria, A. (2010). Functional specification for dtn infrastructure software comprising rfc 5050 bundle agent and associated components. version 1.2.

DTNRG (2014). Grupo de pesquisa sobre redes dtn do irtf. disponível em: http://www.dtnrg.org.

Fall, K. (2003). A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34. ACM.

Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and Babiarz, J. A. (2008). Two-way active measurement protocol (twamp). rfc 5357 (request for comments).

Isento, J. N., Dias, J. A., Canelo, F., Rodrigues, J. J., and Proenca, M. (2012). Moni4vdtn: A monitoring system for vehicular delay-tolerant networks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1188–1192. IEEE.

Kumar, S., Mishra, A., and Cole, R. (2010). Configuration management for dtns. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 589–594. IEEE.

Nobre, J. C., Bertinatto, F. J., Duarte, P. A. P. R., Granville, L. Z., and Tarouco, L. M. R. (2013). Gerenciamento oportunístico em redes tolerantes a atrasos/desconexões através da utilização de tecnologia par-a-par na previsão de encontros entre nós. In *31 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Oliveira, C. T., Taveira, D. M., Braga, R. B., and Duarte, O. C. M. B. (2008). Uma proposta de roteamento probabilístico para redes tolerantes a atrasos e desconexões. In *26 Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Salvador, E. M., Macedo, D. F., and Nogueira, J. M. S. (2011). Gerência autonômica de redes dtn.

Scott, K. and Burleigh, S. (2007). Bundle protocol specification. rfc 5050 (request for comments).

Shalunov, S., Karp, A., Teitelbaum, B., Zekauskas, M. J., and Boote, J. W. (2006). A one-way active measurement protocol (owamp).