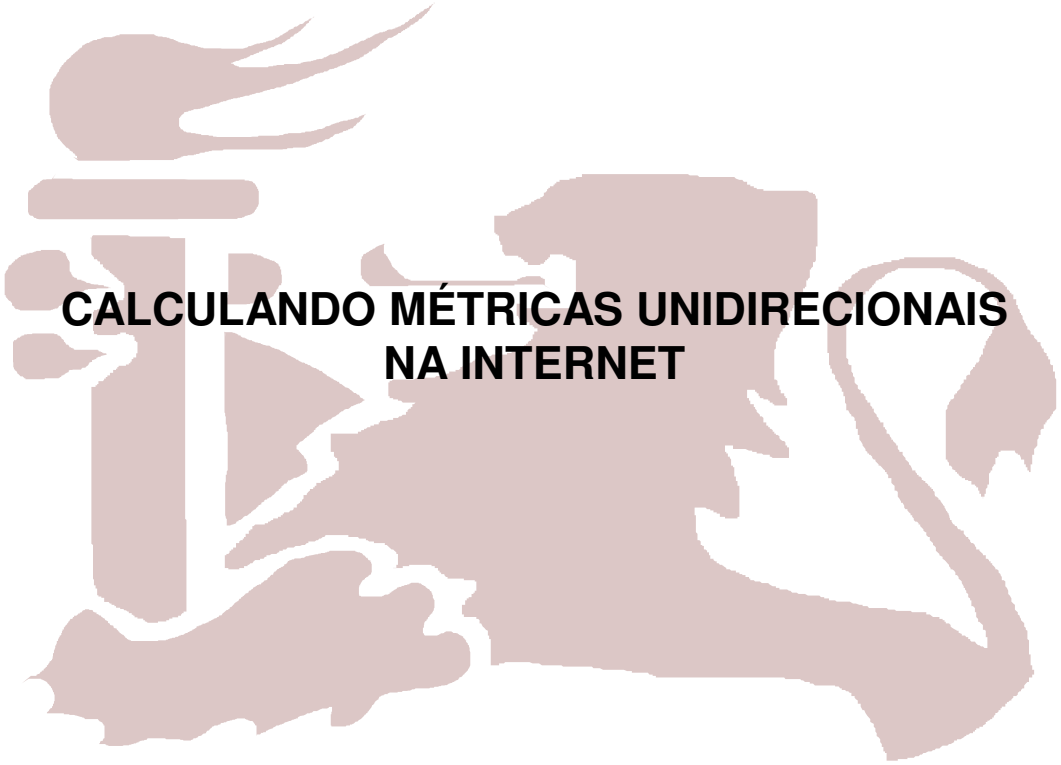
Universidade Federal de Pernambuco

Centro de Informática

Trabalho de Graduação em Redes de Computadores



CALCULANDO MÉTRICAS UNIDIRECIONAIS NA INTERNET

Aluno: Rodrigo dos Santos Bacelar Gouveia Barbosa

Orientador: Djamel Fawzi Hadj Sadok

Março de 2005

Resumo

O tráfego gerado por aplicações multimídia de tempo real é cada vez mais freqüente na Internet. Muitas dessas aplicações são sensíveis aos parâmetros unidirecionais de QoS, como atraso e taxa de perda unidirecionais. As técnicas e ferramentas existentes para extração dessas métricas geram resultados em função de seu próprio tráfego, não sendo capazes de avaliar o desempenho do serviço oferecido aos pacotes de uma aplicação alvo. Este trabalho introduz uma metodologia, discute suas restrições e apresenta ferramentas para medições unidirecionais na Internet. Três estudos de caso com aplicações de tempo-real foram realizados para demonstrar algumas características dessa metodologia.

Agradecimentos

Agradeço aos meus pais, Barbosa e Rita, pela confiança que depositaram em mim ao me deixarem sair de casa para estudar. Ao meu pai, por servir de exemplo a ser seguido e a minha mãe pelo colo nas situações de dificuldade. Agradeço a eles por financiarem minha estadia em Recife e por diversas outras coisas que se fosse citar não caberiam aqui.

Obrigado a Daniel, o irmão de afinidade, pela companhia. Obrigado a Rodrigo, o amigo das opiniões e do desabafo, a Pajé, o amigo dentro e fora da faculdade e a Bruno, o amigo das horas livres. Obrigado também as Anas.

Obrigado àqueles que contribuíram com esse trabalho: Arthur, Kamienski, Mouse e Stênio.

Agradeço aos professores Djamel e Judith por me oferecerem um ambiente de engrandecimento profissional e pessoal. Por me apoiarem e me ajudarem a concretizar vários planos.

Agradeço aos amigos da faculdade e do GPRT.

Sou grato aos professores e funcionários do CIn, da UFPE e ao governo federal por oferecerem uma graduação de ótima qualidade.

Agradeço em primeiro lugar a Deus, por nesses cinco anos, me dar paciência e principalmente perseverança em vários momentos. Finalmente agradeço também a ele pela conclusão de uma etapa da minha vida com bastante saúde.

Índice

Índice de Figuras

1.Introdução

A medição de parâmetros de QoS é importante para avaliar o desempenho de aplicações avançadas de tempo real na Internet, cujo comportamento assimétrico dificulta a obtenção de algumas métricas. Entre elas estão as métricas que são avaliadas em apenas um sentido, chamadas de unidirecionais. O conjunto das métricas unidirecionais é formado pelo atraso em um sentido, a sua variação e a perda de pacotes em um sentido.

O grupo de trabalho da Internet Engineering Task Force (IETF) que trata de métricas de desempenho para o protocolo IP (IP Performance Metrics - IPPM Working Group [12]), tem produzido documentos que especificam essas métricas e propõem metodologias para o seu cálculo. Existem algumas ferramentas ativas de monitoramento que foram criadas com base nesses documentos, injetando pacotes na rede para observar como eles se comportam em um caminho específico.

Os métodos propostos pelo grupo de trabalho IPPM para o cálculo de métricas unidirecionais e as ferramentas que o implementam apenas caracterizam o estado de um caminho da Internet e não avaliam o desempenho do serviço oferecido aos pacotes de uma aplicação alvo. Dessa forma, o método proposto por este trabalho possibilita a extração dos resultados através da geração e coleta dos pacotes gerados pela própria ferramenta, ao invés de um fluxo qualquer.

Motivado pela ausência de uma metodologia e uma ferramenta que avaliasse o desempenho dos serviços recebidos pelos pacotes de uma determinada aplicação em um caminho da Internet, propõe-se neste trabalho uma nova metodologia para o cômputo de métricas unidirecionais. A metodologia pode ser aplicada utilizando dois métodos distintos.

O primeiro método consiste em utilizar uma abordagem de medição passiva e *offline*, ou seja, o processo não injeta tráfego adicional na rede e o cômputo das métricas é realizado apenas após o termino do fluxo que se deseja estudar.

O segundo método propõe o cômputo das métricas unidirecionais com uma abordagem “semi-ativa” em tempo-real, permitindo um acompanhamento “ao vivo” das métricas através de um servidor HTTP. O método é chamado de semi-ativo, porque, ao contrário das ferramentas ativas, o tráfego injetado não é o diretamente avaliado.

O trabalho apresenta a metodologia e os métodos citados acima, assim como as ferramentas que os implementam: o IPstat-*standalone*, que utiliza o método passivo e *offline;* e o IPstat-*monitor*, que implementa o método semi-ativo para acompanhamento em tempo-real das métricas. Ambos são capazes de calcular atraso, variação no atraso (*jitter*) e taxa de perda de pacotes em uma única direção.

Na seqüência deste trabalho, a seção 2 discute as métricas unidirecionais em detalhes. A seção 3 apresenta trabalhos relacionados. A seção 4 descreve detalhadamente a metodologia e os métodos propostos. A seção 5 discute as restrições de implementação da metodologia proposta em função da precisão desejada. A seção 6 descreve o IPstat-*standalone* e a seção 7 o IPstat-*monitor*. A seção 0 mostra três experimentos realizados com o intuito de avaliar métricas de QoS em redes de computadores utilizando o IPstat. E finalmente na seção 9 apresentam-se as considerações finais e direções para trabalhos futuros.

2. Métricas IP Unidirecionais

A avaliação do desempenho de uma aplicação que troca informações pela rede pode depender do serviço recebido pelos seus pacotes. Para algumas aplicações, esse desempenho pode depender principalmente do serviço em cada direção ou mesmo em uma direção. Aplicações de tempo real, como vídeo sobre demanda e voz sobre IP são exemplos dessas aplicações.

Métricas IP unidirecionais são aquelas que medem o desempenho do protocolo IP em apenas um sentido, entre elas estão o atraso, o *jitter* e a perda de pacotes. Essas métricas são necessárias para caracterizar o estado dos caminhos entre os *hosts*, pois as rotas podem ser assimétricas [18], ou seja o caminho de um pacote que trafega num sentido pode diferir do caminho de um pacote que trafega no sentido inverso. Além disso, mesmo que os caminhos de ida e volta sejam simétricos, o enfileiramento dos pacotes pode sofrer variações.
O grupo de trabalho IPPM define e discute as métricas nos documentos [2], [3] e [7].

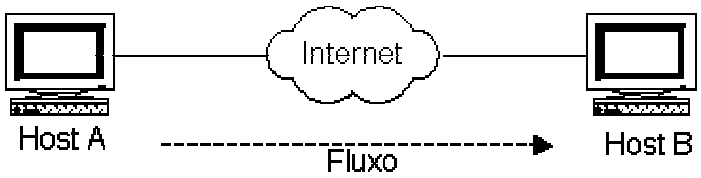


Figura 1: Fluxo no sentido do *Host A* para o *Host B*

Para melhor entendimento das métricas, considere um fluxo no sentido do host A para o host B (Figura 1). A perda unidirecional ocorre quando o pacote enviado por A não chega a B. Quando ocorre fragmentação de um pacote, a perda é caracterizada se pelo menos um dos fragmentos é perdido.

O atraso unidirecional (d^i) é definido como sendo o tempo que o pacote i demora a sair de A e chegar a B. Para obtenção dessa métrica, tanto A como B precisam ter os relógios sincronizados com uma boa precisão. Caso haja fragmentação do pacote analisado, o atraso é dado em função do último fragmento recebido, caso todos tenham sido recebidos. A questão da sincronização dos relógios será tratada com mais detalhes na seção 5.

A variação do atraso em um sentido pode ser definida sobre dois pacotes i e k quaisquer contidos no fluxo A=>B. O *jitter* (dv^{ik}) é calculado pela diferença entre os atrasos unidirecionais d^i e d^k. Ao contrário do atraso, o *jitter*, quando calculado para pacotes consecutivos (o que é o mais comum), independe da sincronização de fase dos relógios (i.e., se os relógios de ambos os *hosts* marcam a mesma hora). Ainda assim, o cálculo de *jitter* depende da sincronização de freqüência dos relógios (i.e., se um relógio se atrasa ou adianta em relação a outro com o passar do tempo). De qualquer forma, o erro de sincronização na freqüência é de várias ordens de magnitude inferior ao valor medido e pode ser ignorado. Isso permite fazer cálculos de *jitter* mesmo sem garantias de sincronização de relógios.

Para explicar porque a sincronização não é necessária, sejam T_A^i e T_A^k o instante de saída dos pacotes *i* e *k*, marcado pelo relógio do *host A*, e sejam T_B^i e T_B^k o instante de chegada dos mesmos pacotes, marcados pelo relógio do *host B*. Considere também que os relógios dos *hosts A* e *B* estão distantes, na fase, por **ε** unidades de tempo. Como já foi dito, o *jitter* é calculado pela diferença entre os atrasos:

$$dv^{ik} = d^k - d^i$$

Cada atraso é calculado pela diferença entre o momento de chegada no *host B* e o momento de saída do *host A*, mais um erro dado pela diferença de fase dos relógios no instante da chegada do pacote:

$$dv^{ik} = (T_B^k - T_A^k + \varepsilon^k) - (T_B^i - T_A^i + \varepsilon^i)$$

Admitindo-se que o cálculo é feito em pacotes adjacentes, tem-se que:

$$\varepsilon^k \cong \varepsilon^i$$

Portanto, a variação no atraso é dada pela equação:

$$dv^{ik} = (T_B^k - T_A^k) - (T_B^i - T_A^i)$$

3.Trabalhos Relacionados

Existe um grande número de técnicas e ferramentas baseadas em medição disponível na literatura. Em um escopo mais amplo, podem-se classificar os tipos de medição como passivas ou ativas. As ferramentas de medição passiva não injetam tráfego adicional na rede e analisam *traces* de fluxos capturados. As ferramentas ativas introduzem dados na rede para, a partir de seu próprio tráfego, inferir as métricas desejadas.

A maioria das ferramentas passivas são baseadas no *tcpdump/libpcap* [24], uma biblioteca de captura de pacotes que fornece mecanismos e facilidades para a implementação de aplicativos que necessitam capturar dados provenientes ou dirigidos à camada de enlace de uma interface de rede. Ela disponibiliza um mecanismo de filtragem de pacotes. Sua API, além de possibilitar um processamento imediato dos dados capturados, informa o momento da captura de cada pacote (*timestamp*), como também fornece meios para armazená-los e recuperá-los através de arquivos. Entretanto, a *libpcap* apenas coleta dados e não se propõe a realizar uma análise mais detalhada sobre as informações capturadas.

Motivados pelo poder e pela liberdade oferecidos pela *libpcap*, foram construídos diversos sistemas de medição e análise de tráfego baseados nessa ferramenta. Entre os mais conhecidos estão o *tcpstat* [25], *Tstat* [14] e *NTOP* [8]. O *tcpstat* fornece estatísticas extraídas de uma interface de rede ou de um arquivo criado pelo *tcpdump/libpcap*. As métricas reportadas por essa ferramenta dizem respeito à vazão (e.g. banda utilizada, número de pacotes por segundo, tamanho médio dos pacotes). O *Tstat* é semelhante ao *tcpstat* e além de reportar as métricas citadas acima, possui um analisador de fluxos TCP. O *NTOP* trabalha no nível IP e é capaz de caracterizar o tráfego por endereço IP, protocolo, etc. Entretanto, nenhuma dessas ferramentas consegue calcular as métricas unidirecionais.

A metodologia ativa, em geral, é usada para avaliar o comportamento de um caminho na Internet. O *ping* (*Packet Internet Groper*) é a ferramenta mais conhecida, que, através de mensagens ICMP, mede a conectividade entre dois

pontos, o atraso de ida e volta e a taxa de perda de ida e volta. Atualmente, existem alguns trabalhos que abordam medições ativas para inferir métricas unidirecionais. Esses trabalhos foram motivados pela possibilidade dos provedores de acesso à Internet saberem o comportamento de suas redes, aplicando melhorias em pontos onde fosse necessário.

Entre os trabalhos mais citados estão o *RIPE NCC* [10] e o *Surveyor* [23]. A infra-estrutura que ambos desenvolveram para realizar as medições é semelhante e contém três componentes: os dispositivos de medição, um servidor e um banco de dados. Os dispositivos de medição são máquinas equipadas com relógios GPS e que usam um sistema operacional capaz de realizar ajustes finos no relógio do sistema (da ordem de microssegundos). Esses dispositivos conseguem calcular as métricas unidirecionais a partir de informações trocadas entre eles. O servidor é responsável por controlar e configurar as medições, coletar e catalogar os resultados no banco de dados e finalmente realizar algum tipo de análise sobre os dados, que são disponibilizados e acessados através de um servidor *web*. A principal diferença entre ambos os trabalhos é que o *Surveyor* altera o *driver* da placa de rede (i.e., no nível do *kernel*) para gerar o *timestamp* dos pacotes de saída e de chegada, permitindo uma maior precisão da informação.

O *OWAMP* (*One-Way Active Measurement Protocol*) [21], é um documento recentemente criado pelo IPPM WG que, em resposta às diferenças nas implementações das plataformas citadas acima para extração das métricas unidirecionais, propõe um padrão e uma metodologia para o cálculo delas.

Os métodos propostos pelo IPPM WG e utilizados pelo *RIPE NCC* e *Surveyor* para descoberta de métricas IP unidirecionais se baseiam em medições ativas, ou seja, injetam seu próprio tráfego na rede. Essa abordagem possui dois aspectos desfavoráveis: o primeiro é que o tráfego gerado pode influenciar a medição; o segundo e principal aspecto negativo é que os resultados são obtidos através da coleta dos pacotes gerados pela ferramenta de medição. Essa ferramenta não é capaz de medir diretamente os pacotes de uma determinada aplicação que se deseja avaliar ou estudar.

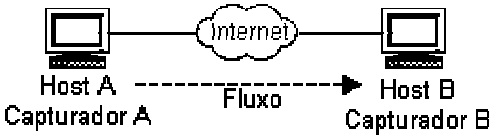
Motivado pelo segundo problema, foi proposta uma metodologia de medição de parâmetros unidirecionais para Internet, que pode ser aplicada

utilizando dois métodos distintos: um método passivo (que também resolve o primeiro problema citado no parágrafo anterior) e um método semi-ativo (não resolve o problema de gerar tráfego concorrente) que possui a característica de oferecer resultados em tempo-real.

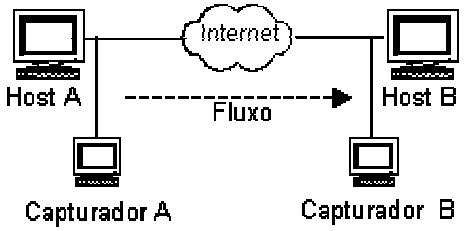
4. Metodologia Para Medir Parâmetros Unidirecionais

A metodologia proposta por este trabalho de pesquisa propõe a realização de medições unidirecionais, com o objetivo de analisar o serviço recebido pelos pacotes gerados por uma determinada aplicação. Vamos supor que se deseje avaliar um fluxo que vai do *host A* para o *host B*.

O primeiro passo é realizar a captura dos pacotes do fluxo desejado nos extremos da rede, conforme mostrado na Figura 2. O motor de captura pode tanto estar na mesma máquina que gera o fluxo (Figura 2a) como em máquinas diferentes (Figura 2b). Chamaremos as informações capturadas no lado do *host A* de *trace A* e as informações capturadas no lado do *host B* de *trace B*. Note que a captura deve ser filtrada para mostrar somente pacotes do fluxo desejado, originados no *host A* e destinados ao *host B*.



(a) *Capturadores* instalados nas máquinas que geram os fluxos



(b) *Capturadores* em máquinas diferentes

Figura 2: Fluxo sendo capturado no sentido *host* A para o *host* B

O segundo passo é a transferência parcial ou completa (dependendo do método utilizado: passivo ou semi-ativo) do *trace A* e do *trace B* para uma máquina, que finalmente, irá fazer o cômputo das métricas.

A técnica proposta neste trabalho é baseada na seguinte informação (contida na RFC que descreve o protocolo IP [19]): dado um protocolo e um mesmo par origem-destino, a unicidade do valor presente no campo *identification* do cabeçalho IP (de agora em diante, neste documento referido como ID), prevalece enquanto o datagrama permanecer ativo na internet. Esse campo é usado para fins de fragmentação e pode ser usado como identificador único de um pacote.

O algoritmo para o cálculo das métricas consiste em percorrer seqüencialmente o *trace A*. Para cada datagrama presente, é buscado no *trace B* o seu equivalente (com mesmo ID). Em seguida, monta-se um par ordenado de *timestamps* – o primeiro elemento do par representa o tempo marcado pelo relógio do *host A* na hora da saída do pacote e o segundo representa o tempo marcado pelo *host B* no momento de recepção do pacote. Esses *timestamps* devem estar presentes nos traces capturados, e para isso deve-se usar uma biblioteca de captura de pacotes que os gere.

Uma perda de pacote é detectada quando o segundo elemento do par inexiste, ou seja, quando para um pacote com um determinado ID no *trace A* não existe um pacote com um mesmo ID no *trace B*.

O cálculo de atraso é obtido subtraindo-se o segundo elemento do par pelo primeiro. Caso o segundo elemento inexista, o atraso é caracterizado como indefinido. Porém, a avaliação do atraso só oferece resultados confiáveis quando os relógios estão sincronizados.

O *jitter* é calculado como a diferença entre duas medidas de atraso consecutivas. Caso uma das medidas seja indefinida, o *jitter* também será. Como apresentado anteriormente, o cálculo de *jitter* independe da sincronização na fase dos relógios.

A metodologia pode ser tanto aplicada usando um método passivo, como usando um método semi-ativo e em tempo-real.

4.1 Método Passivo e *Offline*

O método consiste em armazenar o *trace A* e o *trace B* em dois arquivos distintos para processamento posterior. A análise não é em tempo-real e poderá ser feita somente após a transferência completa dos dois arquivos para uma mesma máquina.

Segue abaixo, uma descrição detalhada do processo:

1. Os *capturadores* são iniciados em ambos os lados (no lado do *host A* e do *host B*). Os *capturadores* devem ser capazes de armazenar as informações em arquivo utilizando algum formato (e.g. *tcpdump/libpcap*).

2. O fluxo é transmitido por completo.
3. Os *capturadores* são interrompidos e geram como saída dois arquivos contendo os *traces A* e *B.*
4. Os dois arquivos são reunidos e passados como entrada para uma ferramenta que realiza o cômputo das métricas e gera os resultados.

A ferramenta que é capaz de implementar esse método é apresentada na Seção 6.

4.2 Método Semi-Ativo em Tempo-Real

Esse método propõe o cômputo das métricas unidirecionais com uma abordagem "semi-ativa" em tempo-real. Ao contrário do método anterior, que tem o objetivo primordial de fornecer uma análise detalhada dos resultados, este método tem o foco em monitoramento, permitindo um acompanhamento das métricas por meio de um servidor HTTP*,* ao mesmo tempo em que o fluxo está sendo transmitido. O método é chamado de semi-ativo, porque, ao contrário das ferramentas ativas, o tráfego injetado não é o diretamente avaliado.

Diferentemente do método passivo, que armazena os quadros capturados em arquivo para uma análise *offline*, o método semi-ativo consiste em enviar a um servidor, de tempos em tempos, as partes do *trace A* e do *trace B* já capturadas, à medida que o fluxo é transmitido.

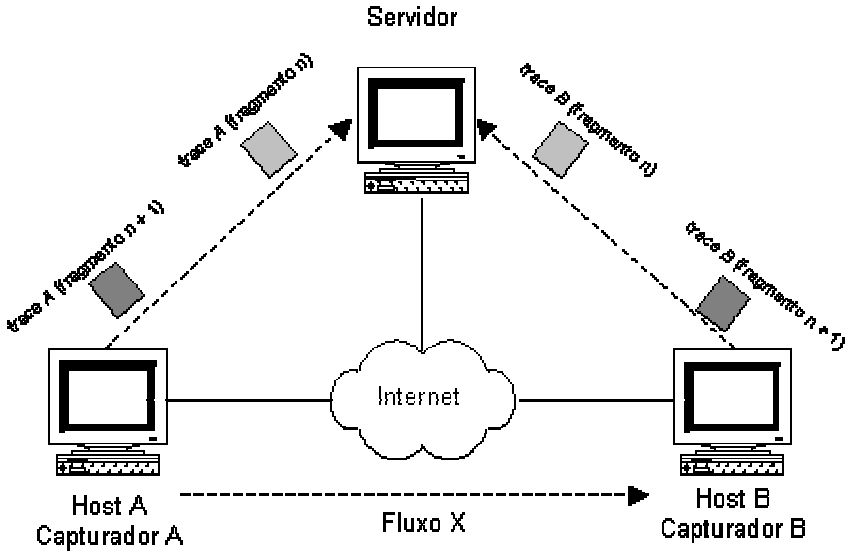


Figura 3: Método Semi-Ativo em Tempo-Real

O servidor é responsável por processar os fragmentos dos *traces* já recebidos e tornar as informações disponíveis em tempo-real

Uma implementação do IPstat-*monitor* é apresentada na seção 7.

5. Precisão da Metodologia

Esta seção discute a precisão da metodologia apresentada acima para estimar cada uma das métricas de interesse. Para o cálculo de taxa de perda de pacotes, o mecanismo utilizado não tem implicações práticas na efetividade da medição, que não há como ser alterada devido a sua simplicidade, então: basta dividir o número de pacotes não recebidos por um *host* pelo número de pacotes enviados ao mesmo.

Para o cálculo das métricas que são em função do tempo, nomeadamente atraso e *jitter*, é necessário ater-se à qualidade dos mecanismos de operação do relógio nas máquinas usadas para capturar o tráfego. A princípio, dado que os atrasos da rede são imprevisíveis, supõe-se que é absolutamente necessário garantir que os relógios das máquinas capturadoras de tráfego estejam sincronizados. Para analisar até onde vai essa necessidade, vamos avaliar os mecanismos que são usados pelos computadores para computar o tempo e os mecanismos usados para sincronizar os relógios de computadores.

Finalmente, para a metodologia mencionada na seção anterior, uma forma interessante de garantir que as capturas de tráfego estejam com relógios sincronizados é fazer as capturas com a mesma máquina. Essa forma será abordada posteriormente, na subseção 5.4.

5.1 Relógios em PCs

Cada relógio físico tem algumas características que influenciam bastante a qualidade do tempo servido [10]. Uma característica importante é a resolução do tempo fornecido. Em sua maioria, PCs possuem um relógio físico (hardware) barato, com uma resolução de 10ms (100Hz). Tipicamente, o relógio físico só é acessado pelo sistema em duas ocasiões: para leitura quando o sistema é ligado (boot) e para gravação quando o sistema é desligado. Paralelamente, os sistemas operacionais provêem um relógio, chamado de relógio do sistema, baseado em código (software) e na capacidade do processador principal, para fornecer a hora às aplicações. A resolução do relógio do sistema tem uma resolução que varia em função do sistema operacional usado. Em PCs, essa

resolução comumente é de 1μs, através do uso da estrutura *timeval*, composta de duas variáveis: *tv_sec* (segundos) e *tv_usec* (microsegundos).

Outra característica importante é a precisão do relógio que define a granularidade efetiva da informação de tempo, mesmo que ela tenha uma resolução mais alta. Assim, um relógio que tem uma resolução de tempo de 1μs pode estar fornecendo a hora com uma precisão de 10ms, e.g., embora use a estrutura *timeval*, com resolução de 1μs, a hora fornecida por um determinado relógio corre em saltos de 10ms (nesse caso, várias leituras do relógio do sistema em um intervalo de 10ms trarão no máximo 2 valores diferentes). A precisão varia bastante entre sistemas operacionais, tipicamente de 10ms (Windows) a poucos microssegundos e em alguns sistemas operacionais ela é ajustável (*FreeBSD*, *Linux*), com recompilação do *kernel*. Existem alguns *drivers* de relógio de sistema (sistemas *Unix*), também chamados de *nanokernels*, que permitem a um relógio trabalhar com a resolução e precisão de poucos nanosegundos.

Além disso, cada relógio possui um erro intrínseco, que é como o comportamento do relógio difere de um relógio de referência (ideal). Esse erro pode ser em fase (e.g., um determinado relógio está 2,54ms atrasado em relação a um relógio de referência, em um dado instante) ou em freqüência (e.g., a cada dia um determinado relógio atrasa 450ms em relação a um relógio de referência).

Para se fazer a comparação de relógios, existem 3 métricas na literatura [15] que podem ser usadas para avaliar a qualidade de um relógio em relação a um relógio de referência. A primeira é a *diferença* entre um relógio e o de referência em um dado instante, medida em segundos. Baseado nessa métrica pode-se ajustar a fase do relógio. A segunda métrica é o *desvio* de um relógio em relação ao relógio de referência, que é o aumento ou diminuição da diferença entre dois relógios em um intervalo de tempo, comumente medida em segundos/dia, ou em partes por milhão (PPM). Esta métrica está diretamente relacionada ao erro em freqüência. A terceira métrica é a *tendência*, que é a variação do desvio em relação a um relógio de referência, em função do tempo. Essa última métrica é desprezada em [15].

Baseado nisso, um bom mecanismo de sincronização de relógios tem 2 objetivos principais: ajuste de fase e ajuste de freqüência. Para isso, é necessário não apenas “ajustar” o relógio local, mas alterar o seu comportamento. Os

métodos mais utilizados para sincronização de relógios são utilizar um relógio físico de alta precisão ou utilizar o Network Time Protocol (NTP) [16] para sincronizar com um PC que usa um relógio físico de alta precisão. O NTP é um protocolo de sincronização de relógios em PCs que utiliza relógios de alta precisão disponíveis na Internet para sincronizar. Além disso, a implementação de referência do NTP vem com um disciplinador de relógios para garantir uma informação de tempo mais confiável.

5.2 Relógio Físico de Alta Precisão

É possível utilizar um relógio físico de alta precisão para corrigir o comportamento do relógio de sistema de um PC. Tipicamente, são usados relógios dos tipos:

- O Global Positioning System (GPS) é baseado em uma rede de satélites mantida pelo governo americano. Este sistema usa uma antena externa que pode se conectar diretamente a uma placa receptora instalada no PC (esquema com maior precisão) ou então se conectar a um receptor que irá transmitir a informação ao PC através da porta serial (menor precisão). As diversas implementações de GPS têm precisão entre dezenas microssegundos até poucos nanosegundos.
- O Relógio Atômico é baseado em um átomo oscilador de alta precisão (ao contrário dos osciladores baseados em cristais, dos PCs). Os mais comuns são os baseados em átomos de Césio, Rubídio ou Hidrogênio.
- O Receptor de avisos de hora por ondas de rádio interpreta sinais de hora provenientes de estações de disseminação de hora, existentes principalmente nos Estados Unidos e na Europa. A interconexão com um PC é feita através da interface serial (menor precisão) ou de uma placa de som (maior precisão).

Qualquer que seja o tipo de relógio usado para interconexão com um PC, existe a possibilidade de uso dos algoritmos embutidos no NTP para disciplinar o relógio. O NTP pode usar o relógio físico de alta precisão para substituir a informação proveniente da rede e no NTP já existem *drivers* para inúmeros relógios físicos. Tipicamente, as interfaces dos relógios físicos podem oferecer uma informação de hora completa e alguns também fornecem a informação no

esquema Pulsos Por Segundo (PPS), muito interessante para integração com o disciplinador de relógio do NTP.

5.3 Sincronização de Relógios com o Network Time Protocol

O NTP [16] faz os ajustes de fase e freqüência através de um disciplinador de relógio, que usa um *loop* de ajuste de fase (PLL – *Phase-Lock Loop*) e um *loop* de ajuste de freqüência (FLL – *Frequency-Lock Loop*), que fazem ajustes mínimos no relógio o tempo todo (com o objetivo de melhorar a qualidade da hora fornecida pelo relógio de sistema em qualquer instante). Ambos são alimentados por dados obtidos através de requisições a relógios de referência na Internet e por dados obtidos do próprio relógio de sistema local, para comparação. Tendo-se uma boa conexão de rede até um relógio de referência, o NTP pode ser configurado para garantir um erro inferior a 1ms.

Mesmo com relógios locais de baixa qualidade, e sem nenhum mecanismo de sincronização, é possível ter garantias de erro máximo no cálculo de *jitter* entre pacotes adjacentes, já que o erro na sincronização da freqüência (desvio) de um relógio local típico em relação a outro, para um intervalo de tempo suficientemente pequeno, (e.g., pacotes adjacentes), é desprezível.

Um relógio comum tem um desvio inferior a 50 PPM, o que dá 50μs a cada segundo. Supondo-se que estamos avaliando pacotes adjacentes cujos intervalos de saída são inferiores a 100ms (e.g., *streaming* de mídia), e supondo que não teremos perda de pacotes adjacentes, teremos que o intervalo máximo entre dois pacotes analisados é de 200ms e assim o erro máximo possível na medição do *jitter* é de $0{,}2x0{,}00005 = 0{,}00001$, ou seja, 10μs.

Se tivermos a certeza de que os relógios das máquinas capturadoras de tráfego têm desvio inferior a 50 PPM e admitirmos que 10μs de erro no cálculo do *jitter* é desprezível, em uma aplicação que envia pacotes em intervalos máximos de 100ms, não há necessidade de acrescentar nenhum mecanismo de sincronização de relógios para fazer o cálculo do *jitter* no experimento.

Para o cálculo do atraso, não há como escapar da necessidade de sincronização dos relógios. Porém, em alguns casos o uso do NTP é suficiente para garantir um erro máximo aceitável. Obviamente, o uso de alguns relógios de

alta confiabilidade para fazer experimentos de atraso é desejável, mas seu custo é muitas vezes proibitivo. Assim, foi desenvolvido um esquema para realizar experimentos com um nível de erro aceitável em situações onde o NTP consegue nos fornecer um bom desempenho.

Um servidor NTP deve ser instalado em um servidor dedicado na mesma rede da máquina que faz a captura dos pacotes. Este servidor deve ser configurado com 3 relógios de referência remotos de alta precisão, todos com o menor atraso possível em relação ao servidor local. A máquina de captura de tráfego deve usar NTP para sincronizar com o servidor local. Em ambas o processo do NTP deve ser executado com prioridade mais alta que quaisquer processos de usuário, para garantir que não haverá perda de qualidade na sincronização pelo contingenciamento do acesso à CPU pelo NTP, causando atrasos excessivos no processamento das informações de tempo necessárias ao processamento dos pacotes do NTP e ao disciplinador de relógio local. Assim, é possível obter um cenário com erro máximo de relógios inferior a 1ms, o que garante um erro aceitável no cálculo do atraso, dependendo dos requisitos.

5.4 Captura em Experimentos com Emulação de Rede

Há, ainda, uma forma de se resolver o problema da sincronização quando se está fazendo um experimento em uma rede emulada. Esta forma consiste em usar a mesma máquina para capturar o tráfego nas duas pontas de rede, assim os dados de ambas as capturas estarão automaticamente sincronizados. Sendo uma solução bastante simples e barata, esse cenário não permite muita interação com um ambiente real, já que os segmentos das duas extremidades da rede devem estar próximos o suficiente para ser possível fazer a captura em ambos a partir da mesma máquina. Utilizá-lo com emulação de rede, entretanto, permite a análise de uma vasta gama de possibilidades, até mesmo algumas possivelmente sem correspondentes no mundo real, mas que podem ser, futuramente, utilizadas. Dessa forma, esse cenário pode despertar o interesse da comunidade científica.

6. Descrição da Ferramenta IPstat-*standalone*

O IPstat-*standalone* é a ferramenta que implementa a metodologia proposta neste trabalho usando medição passiva. Esta ferramenta utiliza a biblioteca de captura de pacotes *libpcap* e produz resultados de métricas IP unidirecionais, que são atraso, variação do atraso e perda de pacotes. O IPstat-*standalone* foi implementado em C/C++, na plataforma Linux, sendo capaz de processar grandes quantidades de pacotes de forma rápida e eficiente. Para obter resultados com o IPstat, é necessário primeiramente efetuar medições com o *tcpdump* ou *ethereal* [9] (utilizando os filtros ou não) no segmento de rede da origem e do destino do fluxo de dados e gerar dois arquivos, que serão posteriormente processados.

As medições devem estar sincronizadas (manualmente) para que todos os pacotes (exceto os perdidos) constem nos dois arquivos gerados. O IPstat se caracteriza como uma ferramenta de medição e análise de tráfego passiva e apresenta os resultados somente após a captura e análise dos fluxos que se deseja analisar.

A forma de utilização da aplicação é semelhante à do *tcpstat*, e para viabilizar a criação de scripts no Linux (necessária quando o volume de dados a ser processado é grande), a interface do IPstat é através da linha de comando. Possui dois argumentos obrigatórios, que são o arquivo contendo o fluxo originado pela interface de rede de envio (de agora em diante, referido como *arquivo A*) e o fluxo recebido pela interface de rede de chegada (referido como *arquivo B*).

O IPstat é configurável e a fim de viabilizar a construção de gráficos há uma opção que personaliza a saída gerada à conveniência do usuário (mesma sintaxe do *printf* presente na biblioteca *stdio* da linguagem C). A ferramenta também disponibiliza um mecanismo de filtragem de pacotes, sendo possível processar apenas pacotes que seguem uma determinada expressão. A sintaxe da expressão segue o formato do *tcpdump*. Finalmente, o IPstat permite especificar se o resultado será mostrado para cada pacote contido em um trace ou se será agrupado em intervalos de tempo, através de um cálculo de média. Instruções sobre compilação, instalação, manual de utilização da ferramenta e exemplos de execução se encontram no Apêndice A.

7. Descrição da Ferramenta IPstat-*monitor*

O IPstat-*monitor* é a ferramenta que implementa o método semi-ativo em tempo real descrito na seção 4. O IPstat-*monitor* possui duas entidades: o cliente e o servidor.

Os clientes são sempre executados aos pares e enviam periodicamente ao servidor informações sobre um conjunto de pacotes que passaram pela interface de rede em um determinado intervalo de tempo. Eles também realizam o papel das máquinas *capturadoras* (Figura 2). Pode haver mais de um par de clientes e cada par é identificado unicamente pelo servidor através de um nome.

O servidor coleta dados provenientes dos clientes, processa-os e disponibiliza através de um serviço HTTP gráficos de atraso, *jitter* e perda, discriminados por fluxo. O IPstat-*monitor* tem a intenção de realizar um papel semelhante ao MRTG (*Multi Router Traffic Grapher*) [17], que é uma ferramenta bastante conhecida para monitorar a carga de tráfego em *links* através de páginas HTML contendo gráficos que fornecem uma representação visual do tráfego. A Figura 4 é um exemplo de página HTML gerada e disponibilizada pelo servidor do IPstat-*monitor*.

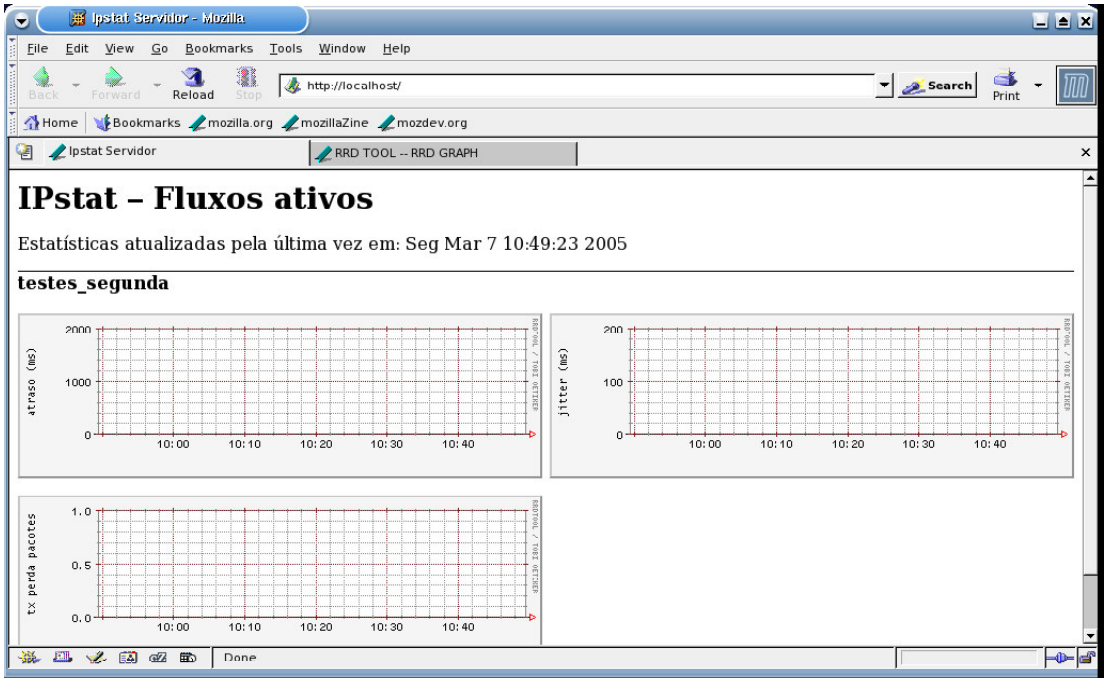


Figura 4: Exemplo de página HTML disponibilizada pelo IPstat-*monitor*

O IPstat-*monitor* foi implementado em C/C++, no *Slackaware Linux*, e utiliza a *libpcap* para capturar os pacotes. O Qt [27], uma biblioteca de classes de C++ foi usada para auxiliar a programação da comunicação (classes Qsocket e QServerSocket), a programação concorrente (classes QThread e QMutex) e a programação da interface gráfica (classes derivadas de QWidget).

A Figura 5 mostra a tela do cliente. Obrigatoriamente, o usuário deve fornecer ao programa o **Nome do Fluxo**. O campo **Filtro** representa os pacotes que o cliente irá filtrar. A sintaxe da expressão segue o formato do *tcpdump*. A **Interface** é a interface de rede usada para capturar os pacotes. O campo **Servidor** é auto-explicativo.

Figura 5: Tela de um cliente do IPstat-*monitor*

Um protocolo foi especificado para comunicação entre os clientes e o servidor. O Apêndice B mostra os tipos e os formatos das mensagens. O TCP foi escolhido para transportar as mensagens principalmente porque um requisito da aplicação é a confiabilidade no transporte da informação.

Vamos descrever de forma breve o processo de conexão, registro e troca de informações de dois clientes com o servidor utilizando o cenário apresentado na Figura 6. Vamos supor que se deseje monitorar o serviço oferecido aos pacotes de um *Fluxo X*. Não há comunicação entre dois clientes, com exceção, é

claro, do fluxo que é gerado pela aplicação que o IPstat deseja estudar. Ou seja, todas as mensagens são trocadas no sentido cliente⇔servidor.

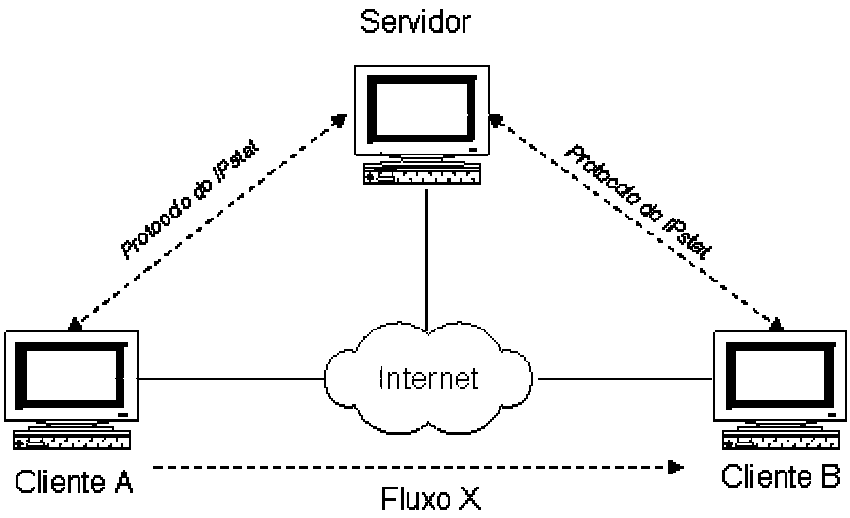


Figura 6: Cenário de execução do IPstat-*monitor*

Todos os clientes são iniciados em um estado de DESCONECTADO. O *cliente A* inicia uma conexão TCP com o servidor e após o estabelecimento da mesma, ele passa para um estado de CONECTADO. Em seguida, o *cliente A* se registra no servidor fornecendo um nome para o fluxo (no exemplo da Figura 6, o nome de registro é “Fluxo X”) e passa para um estado de latência chamado de AGUARDANDO_PEER.

O *cliente B* se conecta ao servidor e logo após se registra utilizando o mesmo nome “Fluxo X”. O servidor percebe que existe um cliente registrado com o mesmo nome e que está no estado AGUARDANDO_PEER (*cliente A*). O servidor envia uma mensagem de OK aos dois clientes. Finalmente, os clientes mudam para o estado de TRABALHANDO. O diagrama de estados completo do cliente pode ser encontrado no Apêndice C.

Após isso, os clientes se tornam aptos a reportar ao servidor informações sobre pacotes capturados. Quando ocorre um pedido de requisição, o cliente deve responder com dados referentes aos pacotes capturados entre essa e a última requisição feita. Para cada pacote capturado no cliente, são enviadas as informações necessárias para computar as métricas unidirecionais, ou seja, o ID do quadro e o instante em que ele foi aprisionado seguindo a estrutura *timeval*.

Como já discutido anteriormente, essa estrutura é composta de duas variáveis: *tv_sec* (segundos) e *tv_usec* (microsegundos)

O servidor guarda os resultados das métricas calculadas em uma base de dados circular disponibilizada livremente pela CAIDA, denominada de RRDTool (*Round Robin Database Tool*) [20]. O RRDTool pode ser definido como uma implementação de alguns recursos disponibilizados pelo MRTG, como por exemplo, a capacidade de criar gráficos e de armazenar informações. O IPstat-*monitor* utiliza o RRDTool para criar os gráficos a serem posteriormente disponibilizados através de uma página HTML. Após a sua criação, a página HTML é finalmente copiada para o diretório de exibição do servidor HTTP. Esse trabalho usou o Apache HTTP Server [4] como servidor.

7.2 Testes com a ferramenta

A fim de testar o IPstat-*monitor*, criou-se o cenário apresentado na Figura 7, e esse teste utilizou três máquinas:

O *Host A* gera o fluxo a ser monitorado. O teste utilizou a ferramenta TG [26] para geração de tráfego reproduzindo um fluxo UDP constante de 10 pacotes por segundo, dirigidos à porta 10001 do *Host B*. O *Host A* também hospeda o servidor do IPstat, assim como um dos clientes. O *Host B* possui o outro cliente IPstat executando. O *Host C* é responsável por emular alguns comportamentos da Internet. A emulação de rede permitiu fazer o estudo de clientes conectados em redes WAN virtuais, bastando implementar no emulador as características desejadas de variação de atraso e taxa de perda a serem aplicados para os pacotes. Foi utilizado o *NIST.Net* para emulação de rede

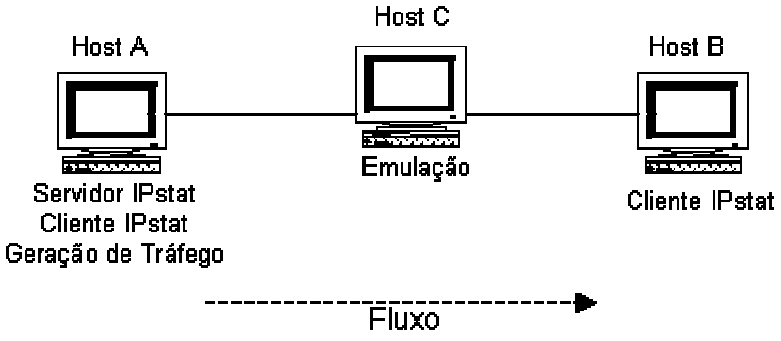


Figura 7: Cenário de Testes do IPstat-*monitor*

O Fluxo foi iniciado às 10:53. No *Host C* foi realizada uma configuração para que os pacotes do fluxo tivessem uma taxa de perda de 5 % e um *jitter* de 100 ms. Às 11:23, as configurações do *Host C* foram alteradas para provocar no fluxo uma taxa de perda de 30 % e um *jitter* de 50 ms. Às 11:45 o teste foi interrompido.

A Figura 8 mostra um acesso ao servidor HTTP logo após o fluxo ser interrompido. Nota-se que o gráfico de taxa de perda de pacotes e do *jitter* são um reflexo das configurações da emulação realizadas no *Host C*. Os valores apresentados no gráfico de atraso não têm valor significativo porque os relógios dos *Hosts A e B* não estavam sincronizados.

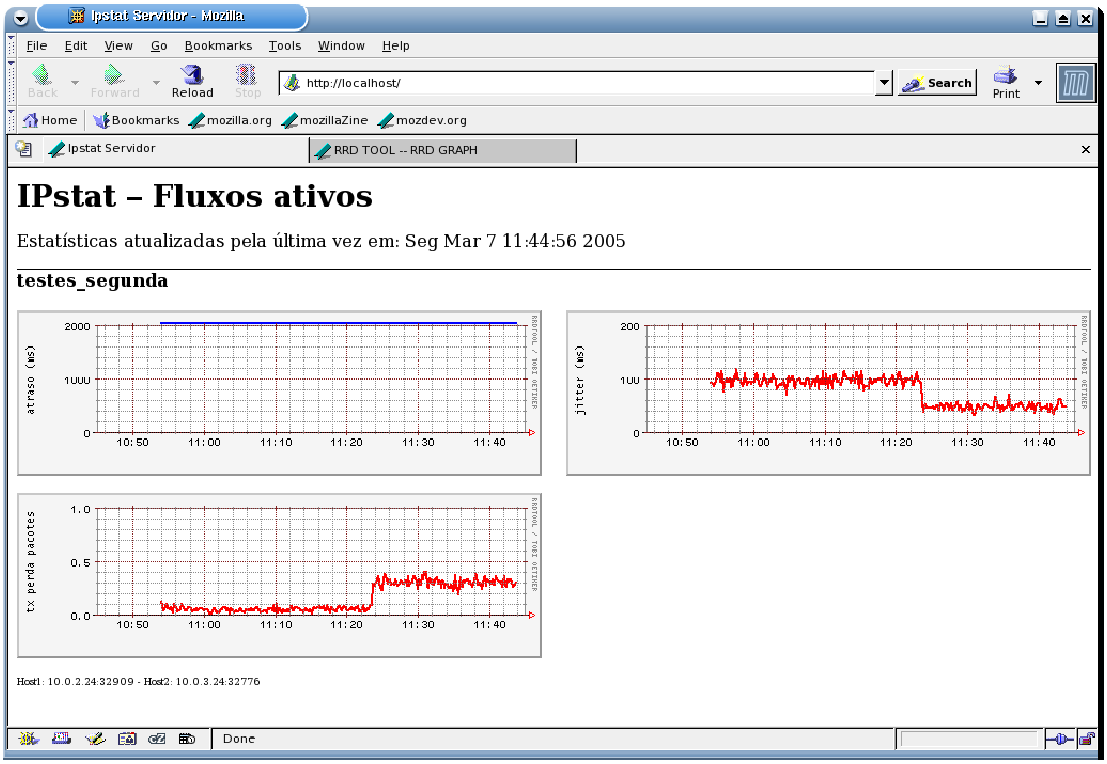


Figura 8: Acesso ao Servidor HTTP do IPstat-*monitor*

Os testes correram normalmente, com exceção que a aplicação não funcionou bem quando o TG foi configurado para gerar poucos pacotes por segundo. Identificou-se que o comportamento curioso se deve pelo fato da comunicação ser realizada através de TCP. Com isso, os clientes só enviavam informações sobre os quadros capturados ao servidor quando o *buffer* do TCP, gerando um comportamento não esperado pela aplicação. Portanto, nas próximas versões, a comunicação em TCP deverá ser substituída por “UDP confiável”.

8. Estudos de Caso

Esta seção mostra três experimentos realizados com o intuito de avaliar métricas de QoS em redes de computadores utilizando o IPstat. Ambos apresentam exemplos práticos do uso de redes em que a qualidade do serviço oferecida influi diretamente na qualidade da experiência do usuário.

8.1 Análise do tráfego de uma sessão de voz sobre IP na Internet

O primeiro estudo de caso do IPstat-*standalone* foi realizado para avaliar a qualidade de serviço percebida pelo tráfego de uma sessão interativa de voz na Internet. O experimento envolveu dois usuários (Usuário 1 e Usuário 2), que utilizaram o software *Skype* [22] para a conversação. O Usuário 2 estava conectado ao backbone da RNP, através de uma rede de acesso Ethernet, pelo PoP de Pernambuco. O Usuário 1 estava conectado ao backbone da Telemar, utilizando uma rede de acesso ADSL.

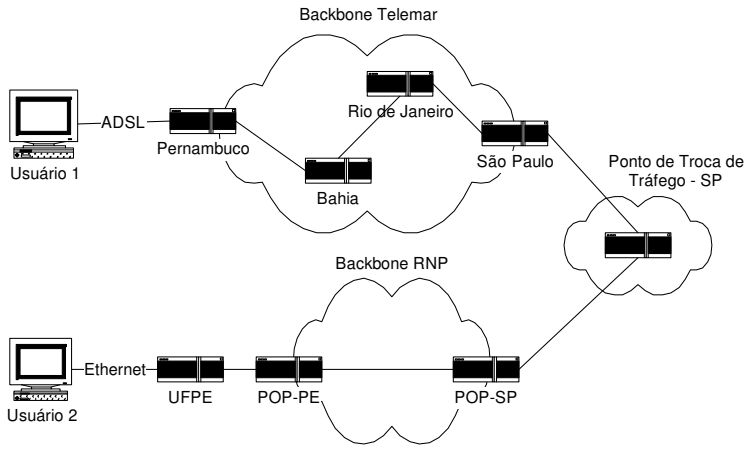


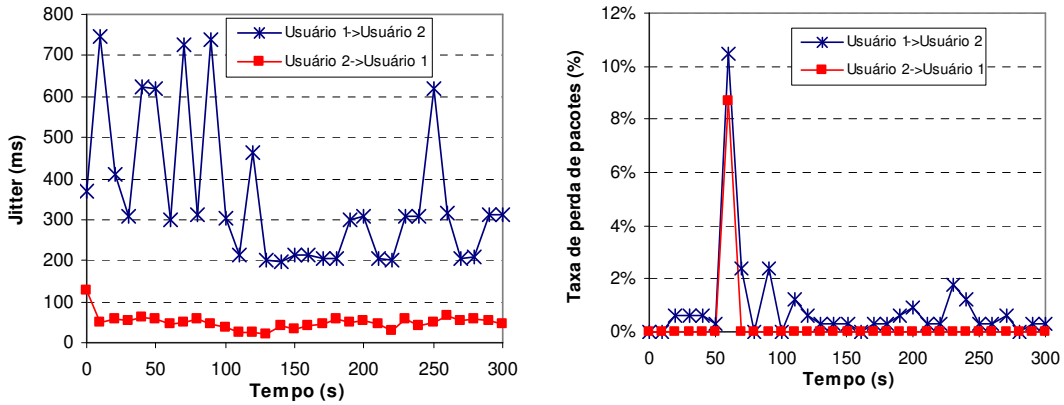
Figura 9: Topologia da rede usada em um experimento de VoIP na Internet

A Figura 9 apresenta a topologia utilizada e como pode ser visto, as duas redes estão interligadas no PTT (Ponto de Troca de Tráfego) em São Paulo, o que resulta em um longo caminho percorrido pelos pacotes, embora os dois usuários estivessem localizados na mesma cidade. Entre os dois usuários, existem 12 roteadores no caminho, o que pode ocasionar eventuais deteriorações no desempenho. Por este motivo, é importante avaliar a qualidade baseada em

parâmetros objetivos, como as métricas de QoS coletadas pelo IPstat. Neste estudo de caso, os relógios não estavam sincronizados e não foi possível calcular a métrica de atraso.

Para efetuar o estudo, primeiramente as capturas (com o *tcpdump*) foram manualmente iniciadas nas máquinas dos dois usuários e após isso, estabeleceu-se uma sessão de voz sobre IP (VoIP) de 5 minutos. Em seguida, detectaram-se as portas que eram utilizadas para o envio e recepção de voz, a fim de filtrar os pacotes e analisar apenas o fluxo gerado pelo *Skype*. Então, o IPstat foi utilizado para calcular o *jitter* e a perda de pacotes. Os resultados foram agrupados em amostras de 10 segundos pela ferramenta e são apresentados na Figura 10.

Através dos resultados gerados pelo IPstat pode-se quantificar a assimetria do caminho percorrido pelos pacotes em relação às métricas de *jitter* (Figura 10a) e taxa de perda (Figura 10b). Nota-se que no sentido Usuário 1=>Usuário 2, o *jitter* máximo observado e a taxa de perda observada alcançaram valores maiores do que no sentido Usuário 2=>Usuário 1, o que mostra assimetria no fluxo de dados, apesar de não haver assimetria nos caminhos de ida e volta.



(a) Jitter (b) Taxa de perda de pacotes

Figura 10: Resultados do IPstat para uma sessão de voz

8.2 Análise de tráfego de uma sessão de VoIP com Sincronização via NTP

O segundo estudo de caso da ferramenta IPstat-*standalone* foi realizado para testar o uso da ferramenta *Skype*. O NTP foi usado como mecanismo de

sincronização de relógios. Dessa forma, foi possível avaliar a métrica de atraso (em um sentido) com um erro inferior a 1ms.

Esse experimento foi realizado com dois usuários, um deles conectado ao *backbone* da Telemar e outro conectado ao *backbone* da Bell Canada, ambos com redes de acesso ADSL. Através do *traceroute*, descobriu-se que as rotas eram assimétricas. A Figura 11 ilustra a topologia utilizada. Os pacotes que o Usuário 1 enviava para o Usuário 2 passavam por 18 roteadores e por quatro *backbones*: Telemar, AT&T, TeleGlobe e finalmente pelo *backbone* da Bell Canada. Os pacotes do Usuário 2 para o Usuário 1 davam 21 saltos e passavam por três *backbones*: Bell Canada, Telefônica e Telemar.

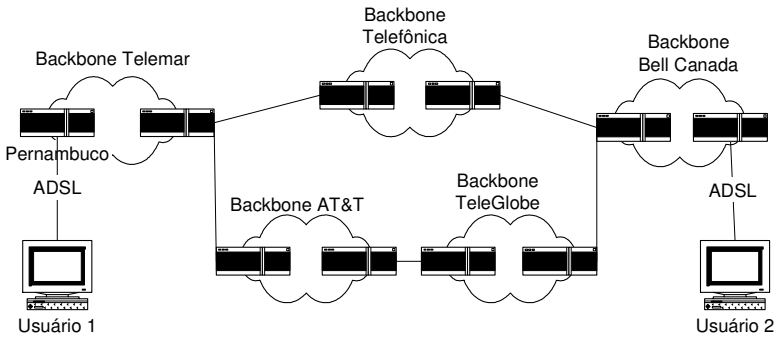


Figura 11: Topologia da rede utilizada no experimento com NTP

Foram gerados 8 minutos de tráfego de voz, e da mesma maneira do experimento anterior, identificaram-se as portas usadas pelo *skype* e capturou-se o fluxo.

Os resultados foram gerados pelo IPstat para cada pacote presente no trace e são apresentados na Figura 12. Nota-se que o atraso medido no sentido Usuário 1=>Usuário 2 é mais que o dobro do que o atraso medido no sentido contrário, refletindo a assimetria dos caminhos. O valor elevado do atraso no sentido Usuário 1=>Usuário 2, de aproximadamente 300ms, causou um impacto na qualidade do serviço. De acordo com [1], atrasos superiores a 300ms prejudicam a conversa, que torna-se inaceitável com atrasos acima de 450ms. A qualidade do serviço quantificada pela Figura 12, foi coerente com a avaliação da qualidade serviço feita subjetivamente pelos Usuários 1 e 2. De acordo com eles, quando falavam simultaneamente, por exemplo, a conversa tornava-se confusa, causando um mal-entendido.

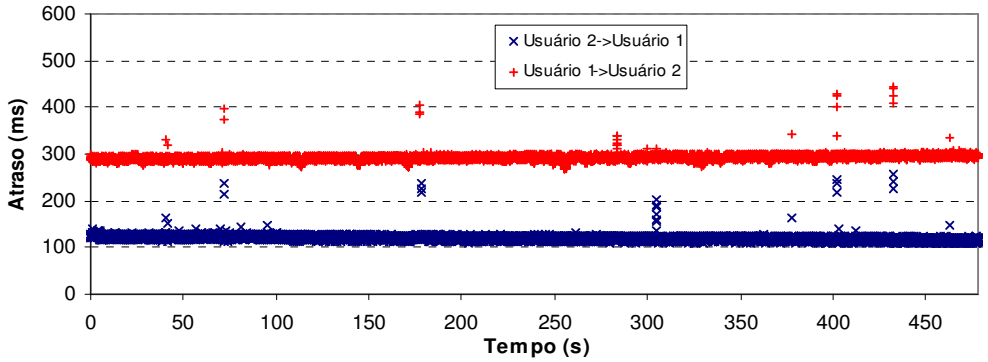


Figura 12: Resultados do IPstat para uma sessão de voz usando NTP

8.3 Avaliação do Middleware QoSWare

Uma versão reduzida da ferramenta IPstat-*standalone* foi inicialmente desenvolvida para ser aplicada em um experimento do projeto QoSWare [6] que necessitava de uma análise de métricas de QoS. Tal projeto utilizou uma rede emulada para realizar os experimentos de aplicações com requisitos de QoS. Isso permitiu o uso de uma única máquina (com duas interfaces de rede) para capturar os pacotes em ambos os extremos da rede.

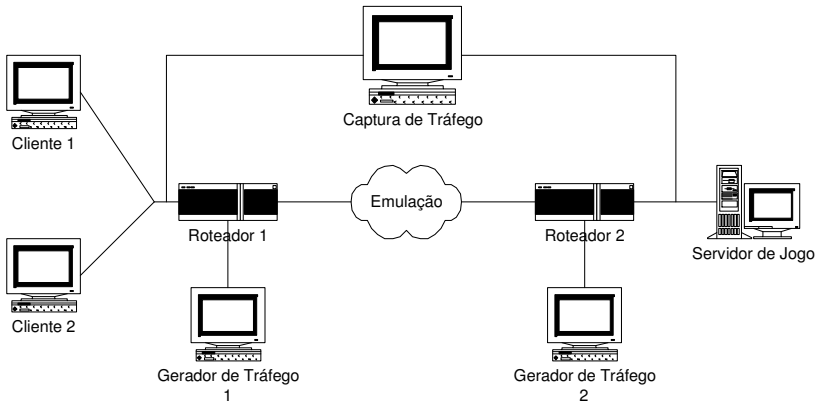


Figura 13: Topologia da rede emulada em estudo

A topologia estudada foi a de uma rede em halteres (*dumbbell*), com um emulador de rede conectando os roteadores (Figura 13). A emulação de rede permitiu fazer o estudo de clientes conectados em redes WAN virtuais, bastando implementar no emulador as características desejadas de capacidade, atraso e variação de atraso a serem aplicados para os pacotes. Foi utilizado o *NIST.Net* [5] para emulação de rede, o *TC* [13] para configurar duas máquinas como

roteadores com QoS e o *TG* [26] para gerar tráfego sintético na rede, repetindo o comportamento da Internet.

O objetivo foi avaliar um *middleware* de QoS, o QoSWare, e uma aplicação que o utilizou, o jogo Multisoccer. O Multisoccer, desenvolvido para avaliar o *middleware*, é um jogo de futebol em tempo real entre dois jogadores, usando um servidor. Por ser de tempo real, este jogo tem seu desempenho bastante dependente das condições de atraso e *jitter*. Foram especificados três cenários diferentes para avaliação e cada configuração única do experimento teve 5 replicações. No cenário A, a aplicação foi avaliada sem o *middleware*. No cenário B, a aplicação foi avaliada com o *middleware*, mas sem QoS, enquanto que no cenário C foi com QoS.

Cenário	Métrica		
	Atraso médio (ms)	Perda média de pacotes (%)	Jitter máximo (ms)
A	40,54	36,07%	10,88
B	40,74	49,35%	9,36
C	40,26	0,75%	0,88

Tabela 1: Resultados do IPstat para o MultiSoccer

A Tabela 1 mostra um exemplo dos resultados obtidos com o uso do IPstat. Vê-se claramente o benefício do uso de QoS tanto em relação à perda de pacotes quanto em relação ao *jitter* observados no tráfego do jogo.

9. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma metodologia, bem como ferramentas para medir parâmetros de QoS unidirecionais na Internet, que incluem as métricas de atraso, variação do atraso e taxa de perda de pacotes em um sentido. O trabalho é relevante no sentido de que a metodologia sugerida, ao contrário das outras propostas da literatura, realiza uma avaliação de desempenho do serviço recebido pelos pacotes de aplicações que possuem requisitos de tempo real. A partir de estudos de casos foi demonstrada a utilidade do IPstat.

Como trabalhos futuros, sugere-se a implementação de mais uma facilidade para a construção de gráficos através da integração do IPstat-*standalone* com o *gnuplot* [11]. O IPstat-*monitor* também pode ser aperfeiçoado com o objetivo de torná-lo um produto comercial, a ser utilizado pelos provedores de Internet e por alguns aplicativos, onde deverão ser implementadas questões de segurança. Por motivos de eficiência, a comunicação feita em TCP deverá ser substituída por uma comunicação usando “UDP confiável”.

10.Referências

[1] Alencar, M. S., "Telefonia Digital", 1a. edição, Editora Erica, 1998.

[2] Almes, G., et al, “A One-way Delay Metric for IPPM”, RFC 2679, Setembro de 1999.

[3] Almes, G., et al, “A One-way Packet Loss Metric for IPPM”, RFC 2680, Setembro de 1999.

[4] Apache HTTP Server Project, http://httpd.apache.org/, último acesso em 02/03/2005.

[5] Carson, M. & Santay, D., “NIST Net - A Linux-based Network Emulation Tool”, ACM SIGCOMM Computer Communication Review, Julho de 2003.

[6] Dantas, S. et al, “Middleware QoSWare: Provendo Suporte à Qualidade de Serviço para Aplicações Avançadas”, XXII Simpósio Brasileiro de Redes de Computadores, Gramado, Maio de 2004.

[7] Demichelis, C., et al, “IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)”, RFC 3393, Novembro de 2002.

[8] Derl, L., Suin, S., “Effective traffic measurement using ntop”, IEEE Communications Magazine 38, Maio de 2000.

[9] Ethereal: A Network Protocol Analyzer, http://www.ethereal.com/, último acesso em 02/03/2005.

[10] Georgatos, F. et al, “Providing Active Measurements as a Regular Service for ISP’s”, Proceedings of the Passive and Active Measurements Workshop, PAM2001, Amsterdam, Abril de 2001.

[11] gnuplot homepage, http://www.gnuplot.info, último acesso em 02/03/2005.

[12] Internet Engineering Task Force, IP Performance Metrics Working Group, http://www.ietf.org/html.charters/ippm-charter.html, último acesso em 02/03/2005.

[13] Linux Advanced Routing & Traffic Control HOWTO, http://lartc.org/howto, último acesso 02/03/2005.

[14] Mellia, M. et al, “Tstat: TCP STatistic and Analysis Tool”, Proceedings of the Second International Workshop on Quality of Service in Multiservice IP Networks, 2003

[15] Mills, D. L., “Network Time Protocol (Version 3) Specification, Implementation and Analysis”, RFC 1305, Março de 1992.

[16] Mills, D. L., “Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI”, RFC 2030, Outubro de 1996.

[17] MRTG: The Multi Router Traffic Grapher, http://people.ee.ethz.ch/~oetiker/webtools/mrtg/, último acesso em 02/03/2005.

[18] Paxson, V., “End-to-End Routing Behavior in the Internet”, IEEE/ACM Transactions on Networking, Outubro de 1997.

[19] Postel, J., “Internet Protocol”, RFC 791, Setembro de 1981.

[20] RRDTool, http://www.rrdtool.org, último acesso em 02/03/2005.

[21] Shalunov, S., et al, “OWAMP – One-Way Active Measurement Protocol”, Network Working Group Internet Draft “http://www.ietf.org/internet-drafts/draft-ietf-ippm-owdp-12.txt”, Dezembro de 2004.

[22] Skype: Free Internet telephony, http://www.skype.com, último acesso em 02/03/2005.

[23] Surveyor Home Page, http://www.advanced.org/surveyor/, último acesso em 02/03/2005.

[24] TCPDUMP public repository, http://www.tcpdump.org, último acesso em 02/03/2005.

[25] tcpstat Home Page, http://www.frenchfries.net/paul/tcpstat/, último acesso em 02/03/2005.

[26] Traffic Generator tool, http://www.postel.org/tg/tg.htm, último acesso em 02/03/2005.

[27] Trolltech – QtProduct Overview, http://www.trolltech.com/products/qt/index.html, último acesso em 02/03/2005.

Apêndice A - Manual do IPstat-*standalone*

A.1 Compilação e Instalação

Para compilar e instalar o IPstat no sistema, basta executar a seguinte seqüência de comandos no diretório base da distribuição do IPstat:

```
% ./configure
% make
% make install
```

A.2 Informações Importantes Antes de Usar

A ferramenta gera resultados a partir de dois arquivos que contém traces do fluxo que se deseja estudar ou avaliar. Portanto, o arquivo1 (possuindo o fluxo originado pela interface de rede de envio) e o arquivo2 (contendo o fluxo recebido pela interface de rede de chegada) devem ser gerados. Um exemplo de geração desses arquivos, utilizando o *tcpdump* encontra-se abaixo:
No host1:

```
% tcpdump -w arquivo1.dump
```

No host2:

```
% tcpdump -w arquivo2.dump
```

O IPstat analisa apenas arquivos contendo quadros provenientes de interfaces Ethernet e ADSL.

A.3 Usando o IPstat

Alguns argumentos utilizados são semelhantes ao do *tcpstat* (http://www.frenchfries.net/paul/tcpstat), e possui interface baseada em linha de comando, escolhida para viabilizar a criação de scripts no Linux (a criação de scripts é necessária quando o volume de dados a ser processado é grande). Os resultados são exibidos em notação científica com valores em segundos.

O IPstat possui dois argumentos obrigatórios, compostos pelos arquivo1 e arquivo2. Abaixo está a sintaxe completa do IPstat:

```
% ipstat [-?he] [-f filter expr] [-o output] [-i interval] [-s string] filename1 filename2
```

Opções de Uso:

`-f filter expr` A ferramenta disponibiliza um mecanismo de filtragem de pacotes, sendo possível processarem apenas pacotes que seguem uma determinada expressão. A sintaxe da expressão segue o formato do *tcpdump* (http://www.tcpdump.org) e está detalhada em seus manuais.

Por exemplo, para obter informações apenas sobre pacotes endereçados ao IP 192.168.0.1, contendo a porta 4444, *`filter expr`* será igual a "`dst 192.168.0.1 and port 4444`".

```
% ipstat -f "dst 192.168.0.1 and port 4444" arquivo1.dump arquivo2.dump
```

Recomenda-se utilizar a opção **–f** no IPstat ou no *tcpdump*, já que em uma interface de rede, trafegam quadros provenientes de diversos *hosts* e diversas aplicações

`-i interval` Através dessa opção, o IPstat permite mostrar o resultado por intervalos de tempo. O parâmetro *interval* é do tipo *float* e representa os segundos. Por exemplo, caso se deseje obter a métrica de atraso e se a opção "`-i 10`" for utilizada, os dados são agrupados por amostras de tempo e o resultado será reportado relacionado ao conjunto dos pacotes contidos em intervalos de 10 segundos. Se a métrica desejada for atraso, então será feita uma média. Se a métrica for taxa de perda, será informada a razão da quantidade de pacotes perdidos pela quantidade total de pacotes contidos no intervalo. Se a métrica for *jitter* máximo, será informado o maior valor de *jitter* encontrado no intervalo, e assim por diante.

Caso a opção não seja especificada, o resultado será informado para todos os pacotes do *trace.* Caso a opção "`-i -1`" seja utilizada, será informada uma média de todos os pacotes, sendo o resultado exibido em apenas uma linha.

Abaixo, tem-se um exemplo da utilização do parâmetro `-i`. O parâmetro `-o` será detalhado mais adiante.

```
% ipstat -i 10 -o "Time: %t\tLoss: %l\n" arquivo1.dump arquivo2.dump

Time: 0.000000e+00          Loss: 0.000000e+00
Time: 1.000000e+01          Loss: 0.000000e+00
Time: 2.000000e+01          Loss: 5.988024e-03
Time: 3.000000e+01          Loss: 6.006006e-03
Time: 4.000000e+01          Loss: 5.970149e-03
Time: 5.000000e+01          Loss: 3.003003e-03
Time: 6.000000e+01          Loss: 1.047904e-01
Time: 7.000000e+01          Loss: 2.395210e-02
Time: 8.000000e+01          Loss: 0.000000e+00
Time: 9.000000e+01          Loss: 2.402402e-02
Time: 1.000000e+02          Loss: 0.000000e+00
Time: 1.100000e+02          Loss: 1.197605e-02
Time: 1.200000e+02          Loss: 5.988024e-03
Time: 1.300000e+02          Loss: 2.994012e-03
Time: 1.400000e+02          Loss: 2.985075e-03
Time: 1.500000e+02          Loss: 3.003003e-03
```

Apesar do arquivo1.dump conter bastantes pacotes, nota-se que o resultado foi agrupado em amostras de 10 segundos. A informação contida no segundo zero, por exemplo, mostra o resultado relacionado aos pacotes capturados entre o tempo de captura do primeiro pacote e o tempo de captura do primeiro pacote após transcorrem 10 segundos.

`-o output` O IPstat é totalmente configurável. A fim de viabilizar a construção de gráficos, a opção `-o` ajusta a saída gerada à conveniência do usuário, além de definir as métricas a serem exibidas. Para facilitar sua utilização, o *output* possui a mesma sintaxe do *printf*, contido na biblioteca *stdio.h*, de C.

Considere o exemplo acima, onde o parâmetro *output* contém a *string* "`Time: %t\tLoss: %l\n`". O item `%t` é um caractere especial que especifica o tempo (já explicado acima). O caractere `\t`, assim como na sintaxe do *printf*, representa uma tabulação. O valor `%l` é outro caractere especial, representando a taxa de perda de pacotes. O caractere `\n`, como também na sintaxe do *printf*, representa uma quebra de linha. Abaixo, há uma tabela informando todas as métricas reportadas pelo IPstat, juntamente com todos os caracteres especiais.

`%T`	Timestamp
`%t`	interval (seconds)
`%l`	loss rate
`%d`	delay (seconds)
`%s`	min delay (seconds)
`%f`	max delay (seconds)
`%j`	jitter (seconds)
`%h`	min jitter (seconds)
`%k`	max jitter (seconds)

Tabela 2: Métricas reportadas pelo IPstat com o respectivo caractere especial

Caso a opção `-o` não seja especificada, é utilizado um *output* padrão: `"%T\t%d\t%j\t%l\n"`. Há também uma ajuda *on-line* (opção `-h`), caso o usuário ainda não esteja familiarizado com as métricas.

`-e` Alguns valores podem ser indefinidos. Por exemplo, quando um pacote é perdido, as métricas de atraso e *jitter* são indefinidas. Caso a opção `-e` seja utilizada, informação que contenha apenas valores indefinidos não é reportada.

Para ilustrar, considere o exemplo seguinte:

```
% ipstat -o "Timestamp: %T\t Jitter: %j\n" arquivo1.dump arquivo2.dump
...
Timestamp: 1.0747720262219510e+09          Jitter: 1.636600e-02
Timestamp: 1.0747720262499411e+09          Jitter: 1.635909e-02
Timestamp: 1.0747720262636432e+09          Jitter: UNDEFINED
Timestamp: 1.0747720262841220e+09          Jitter: UNDEFINED
Timestamp: 1.0747720263162551e+09          Jitter: 1.928799e-01
...
```

Note que para alguns pacotes, a métrica de *jitter* não está definida. Provavelmente porque alguns pacotes foram perdidos. Abaixo, há um exemplo de utilização da opção `-e`:

```
% ipstat -e -o "Timestamp: %T\t Jitter: %j\n" arquivo1.dump arquivo2.dump
...
Timestamp: 1.0747720262219510e+09          Jitter: 1.636600e-02
Timestamp: 1.0747720262499411e+09          Jitter: 1.635909e-02
Timestamp: 1.0747720263162551e+09          Jitter: 1.928799e-01
...
```

`-s string` Essa opção substitui a palavra `UNDEFINED`, utilizada quando a métrica é indefinida, pela palavra especificada em *string.* Por exemplo:

```
% ipstat -o "Timestamp: %T\t Jitter: %j\n" -s NAO_EXISTE arquivo1.dump
arquivo2.dump
...
Timestamp: 1.0747720262219510e+09          Jitter: 1.636600e-02
Timestamp: 1.0747720262499411e+09          Jitter: 1.635909e-02
Timestamp: 1.0747720262636432e+09          Jitter: NAO_EXISTE
Timestamp: 1.0747720262841220e+09          Jitter: NAO_EXISTE
Timestamp: 1.0747720263162551e+09          Jitter: 1.928799e-01
...
```

`-h` Exibe informações sobre as métricas disponíveis no IPstat, assim como seu caractere especial, utilizado no *output*. Quando especificado, exibe a seguinte saída:

```
IPstat version 1.0
Information about output formats:
       %T - timestamp
       %t - interval (seconds)
       %l - loss rate
       %d - delay (seconds)
       %s - min delay (seconds)
       %f - max delay (seconds)
       %j - jitter (seconds)
       %h - min jitter (seconds)
       %k - max jitter (seconds)
       default output: "%T\t%d\t%j\t%l\n"
```

–? Utilizado para exibir um *help on-line*. Quando especificada mostra a seguinte saída:

```
IPstat version 1.0
usage: ipstat [-?he] [-f filter expr] [-o output] [-i interval] [-s string]
filename1 filename2
       -? - display this help
       -h - display information about output formats
       -e - do not print undefined values
       -f filter expr- packet filter expression (like in tcpdump)
       -o output - format for the output of stats (see manpage or use -h option
       for details)
       -i interval - time interval (double seconds) for taking samples
       -s string - string displayed when the metric is undefined
       filename1 - read data from filenameOut, usually the file that has shorter
       timestamps
       filename2 - read data from filenameIn, usually the file that has larger
       timestamps
```

Apêndice B - Tipos e Formato das Mensagens Usadas na comunicação do IPstat-*monitor*

Todas as mensagens trocadas entre o cliente e o servidor são encapsuladas em um pacote TCP. Uma mensagem possui um cabeçalho de tamanho fixo (7 bytes) e dados de tamanho variável (vide Figura 14).

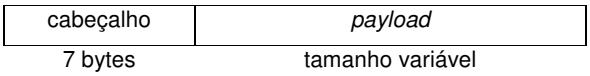


Figura 14: Formato das mensagens trocadas entre cliente e servidor

O cabeçalho possui 3 campos (vide Figura 15) :

- versão: versão do protocolo usada;
- código: especifica o tipo da mensagem;
- tamanho *payload*: informa o tamanho em bytes dos dados que seguem o cabeçalho.

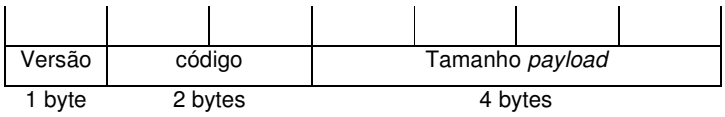


Figura 15: Estrutura do cabeçalho

Abaixo, estão descritos todos os tipos de mensagem usados no protocolo, com o respectivo código.

Mensagens de registro do cliente para o servidor:

`CLI_SERV_LOGIN_REQUEST - Código 0x0001`

Deve ser enviada logo após o cliente estabelecer uma conexão com o servidor, e tem a finalidade de registrar um fluxo que se deseja monitorar.

O *payload* é preenchido com uma string, que é a identificação do fluxo.

Mensagens de registro do servidor para o cliente:

`SERV_CLI_LOGIN_OK_AGUARDANDO_PEER - Código 0x0001`

Enviada quando o cliente realizou o registro com sucesso e informa que o seu parceiro de fluxo (*peer*) **não está** conectado ao servidor. Mandada em resposta a uma mensagem CLI_SERV_LOGIN_REQUEST.

A mensagem é composta apenas pelo cabeçalho e não possui dados (*payload*).

SERV_CLI_LOGIN_OK_PEER_LOGADO – Código 0x0002

Pode ser enviada em duas situações:

- Quando o cliente realizou o registro com sucesso e seu parceiro de fluxo (*peer*) já **está** conectado ao servidor. Nesse caso, é uma resposta a uma mensagem CLI_SERV_LOGIN_REQUEST;
- Quando o cliente que recebeu a mensagem já estava conectado, porém o seu parceiro de fluxo (*peer*) acabou de conectar.

A mensagem é composta apenas pelo cabeçalho e não possui dados (*payload*).

SERV_CLI_LOGIN_FALHOU_NOME_EXISTENTE – Código 0x0003

Enviada quando o cliente é incapacitado de se registrar ao servidor porque já existem dois clientes registrados com o nome de fluxo especificado.

A mensagem é composta apenas pelo cabeçalho e não possui dados (*payload*).

Mensagens de dados do servidor para o cliente:

SERV_CLI_DATA_REQUEST – Código 0x0010

Solicita todos os quadros que foram capturados entre o momento de recebimento dessa mensagem e o momento de recebimento da última mensagem SERV_CLI_DATA_REQUEST.

A mensagem é composta apenas pelo cabeçalho e não possui dados (*payload*).

Mensagens de dados do cliente para o servidor:

CLI_SERV_DATA_RESPONSE – Código 0x0010

É uma resposta a uma mensagem SERV_CLI_DATA_REQUEST, contendo informações sobre os quadros capturados.

O *payload* é preenchido com uma seqüência de “quadros”, e cada “quadro” possui 3 campos:

- id: valor obtido do campo *identification* do cabeçalho IP de um pacote capturado.

- ts_sec: momento da captura do pacote no formato de tempo do Unix (ISO 8601). Ou seja, esse campo marca os segundos decorridos desde 01 de Janeiro de 1970 (1970-01-01T00:00:00Z) no momento da captura.
- ts_usec: Como o campo anterior tem a precisão apenas de segundos, esse campo adiciona uma precisão de microssegundos.

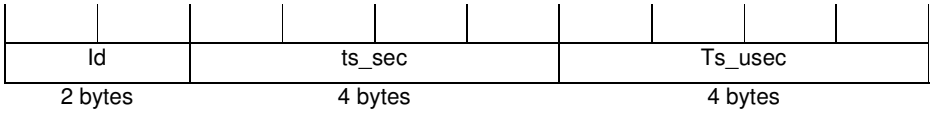


Figura 16: Estrutura de um quadro

Portanto, para capturar o momento exato da captura do pacote (em segundos), basta fazer: ts_sec + ts_usec * 10^{-6}.

Apêndice C - Diagrama de Estados do Cliente IPstat-*monitor*

A Figura 17 mostra o diagrama de estados completo do cliente.

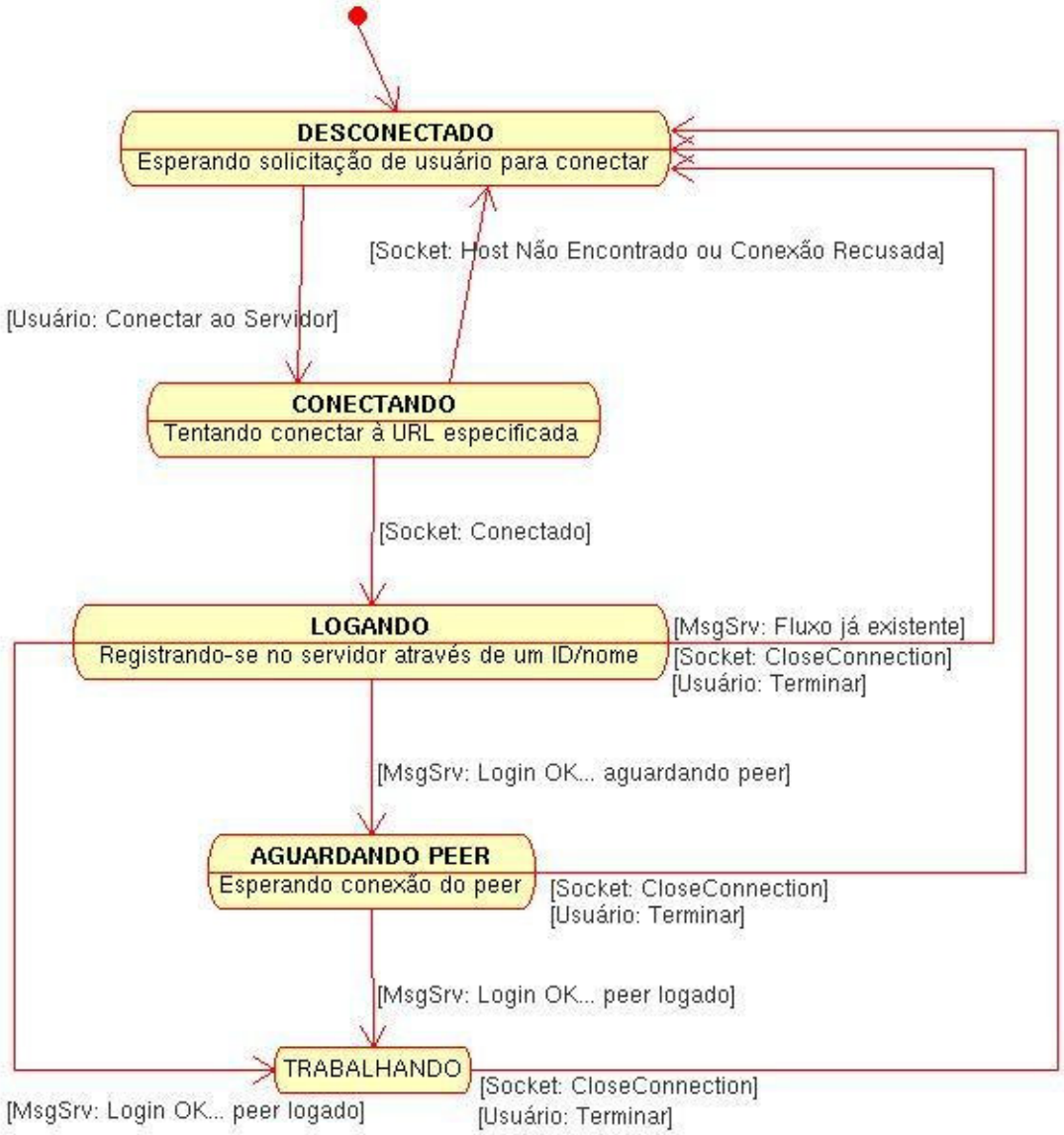


Figura 17: Diagrama de estados do cliente IPstat-*monitor*

_____	_____
Autor (Rodrigo dos Santos B. G. Barbosa)	Orientador(Djamel F. H. Sadok)