

Universidade do Minho
Mestrado de Engenharia Informática
Tecnologias e Protocolos de Infraestrutura



Universidade do Minho

Projecto Integrado - RIPA/RMS

Redes IP Avançadas

Redes Multiserviço

Ano Lectivo de 2008/2009

Monitorização de níveis de QoS com recurso a métricas One-way

José Pedro Vilaça Novais
Miguel Craveiro Martins de Almeida

11 de Julho de 2009

Conteúdo

Conteúdo	i
1 Monitorização de QoS	1
1.1 Introdução	1
1.2 Motivação	1
1.3 Desenvolvimento do Sistema	2
1.3.1 Requisitos para a medição de métricas <i>one-way</i>	2
1.3.2 Funcionamento da aplicação de medição	4
1.3.3 Estrutura dos pacotes de prova	4
1.3.4 Estrutura dos dados armazenados	4
1.3.5 Ambiente de desenvolvimento	5
1.4 Utilização	5
1.4.1 Instalação e configuração	5
1.4.2 Configuração da sincronização do relógio por GPS	6
1.4.3 <i>Site</i> com medições dos níveis de QoS	6
1.5 Dificuldades de Implementação	8
1.6 Conclusões	9
Bibliografia	10

Capítulo 1

Monitorização de QoS

1.1 Introdução

Este projecto pretende integrar as áreas de estudo dos dois módulos da UCE de Tecnologias e Protocolos de Infraestrutura. Foi desenvolvida uma ferramenta de medição de níveis de QoS (*Quality of Service*) de uma rede com métricas num só sentido, permitindo conhecer-se com muito mais detalhe as características da rede.

1.2 Motivação

Na maior parte das situações, quando se pretende medir características de uma rede como atraso ou percentagem de pacotes perdidos no caminho, a utilização de métricas de dois sentidos é preferida, já que reflecte tanto o pedido como a resposta de uma ligação. A ferramenta deste género mais acessível é o comando `ping`, que analisa o *Round Trip Time* (RTT). No entanto, muitas vezes os administradores de redes pretendem diferenciar as medições num e noutro sentido da rede. Algumas das principais situações em que este tipo de medições são importantes:

- Por forma a avaliar o cumprimento de *Service Level Agreements* (SLA) acordados com clientes pode haver a necessidade de recolha de métricas *one-way*, tais como *delay*, *jitter* (a variação no *delay*) e a percentagem de pacotes perdidos.
- Um determinado fluxo de dados não tem necessariamente de seguir o mesmo caminho na rede em ambos os sentidos. A identificação de diferenças entre os dois caminhos pode ser essencial para se detectar falhas na rede, que doutro modo seriam muito difíceis de se identificar.
- Flutuações nas características da rede podem influenciar negativamente muitos protocolos adaptativos, como o caso de alguns *codecs* de *Voice over IP* (VoIP).

Um sistema em funcionamento que utiliza este tipo de medições é utilizado pelo **RIPE Network Coordination Centre** (<http://www.ripe.net/projects/ttm/Plots/>). No entanto, não existe uma alternativa gratuita, de simples instalação e utilização e adaptada a *firmwares open-source* como o `pfSense` ou `OpenWrt`. O sistema que foi desenvolvido pretende aproximar-se do utilizado pelo **RIPE**, mas com a vantagens enunciadas.

1.3 Desenvolvimento do Sistema

O sistema que foi desenvolvido consiste numa rede de *Measurement Points* (MP) interligados e que trocam constantemente pacotes de prova entre si (ver figura 1.1). Espera-se conhecer métricas como *delay*, *jitter* e *packet loss* entre cada um destes MP. Em ambientes de produção, a ferramenta desenvolvida poderá ser instalada tanto em máquinas dedicadas como nos próprios *routers* (baseados em Linux). Deverão ser instalados em pontos estratégicos da rede que permitam uma análise útil.

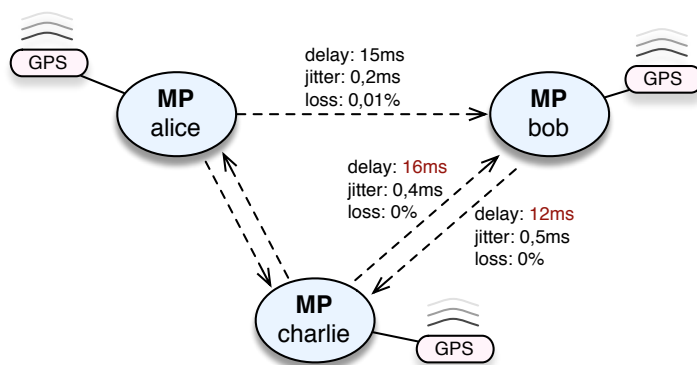


Figura 1.1: Três MP interligados, onde se destaca a diferença significativa entre o atraso nos dois sentidos de **bob** para **charlie**.

O acesso aos dados recolhidos da rede deverá ser facilmente acedida, mediante o acesso a um *site*. Este *site* poderá ser único, centralizando os dados recolhidos de todos os MP, ou acedido em cada um dos MP, reunindo apenas a informação relativa a esse MP. No caso do *firmware OpenWrt*, o site seria integrado na LuCI (<http://luci.freifunk-halle.net/>), um motor *web* já desenvolvido com o propósito de fornecer ao utilizador um ambiente simples de configuração dos *routers*. Note-se que este método de funcionamento pode ser particularmente útil em situações em que se deseja instalar um sistema simples entre apenas dois pontos da rede, sem obrigar a instalar um terceiro servidor que apenas reúna os valores analisados.

1.3.1 Requisitos para a medição de métricas *one-way*

Quando são utilizadas medições *two-way*, como o RTT não é necessário haver sincronização perfeita entre os dois pontos da rede, já que a referência é o relógio local. No entanto, a utilização de métricas *one-way* implica conhecer-se o tempo exacto de saída do emissor e o tempo exacto de chegada ao receptor. Naturalmente, estes deverão estar perfeitamente sincronizados para que não exista erro nas medições. Se o sistema pretende medir valores de atraso na ordem do milissegundo, é de se esperar que o erro de medição seja inferior a 1ms. Isto implica que a diferença entre os relógios dos vários MP seja inferior a este valor. No entanto, os protocolos existentes de sincronização de relógios pela rede (como o NTP) introduzem um erro que apesar de ser pequeno (na ordem dos 5ms) [1], é maior do que o desejado. Torna-se necessário recorrer a mecanismos mais precisos, como a sincronização por GPS. Este tipo de sincronização já permite um erro na ordem dos microsegundos [3, 4]. Existem no mercado vários equipamentos capazes de oferecer este tipo de precisão [7, 8]. No entanto são caros, de grande dimensão e de instalação pouco

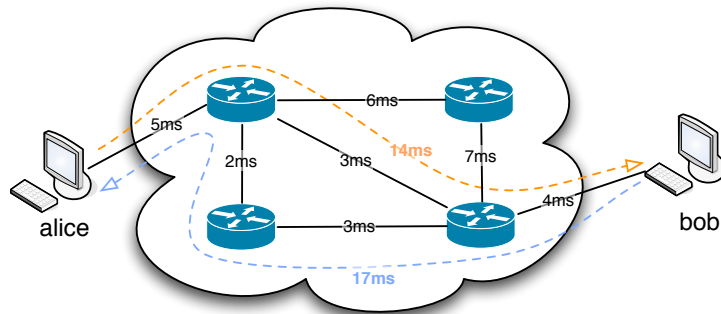


Figura 1.2: Ilustração de um caso em que o caminho percorrido pelo tráfego num sentido é diferente do sentido inverso.

prática. A solução acaba mesmo por ser a construção de um destes sistemas, com o apoio de várias ferramentas e mecanismos disponíveis para Linux:

- **ntp:** (*Network Time Protocol*) Este protocolo (e utilitário com o mesmo nome) permite manter a constante sincronização entre o relógio interno do sistema com vários relógios remotos. A sua utilização mais comum é através da rede, havendo uma eleição do servidor com menor *offset* (diferença entre o tempo do relógio interno e o recebido do servidor de referência).
- **gpsd:** Este serviço permite manter comunicação com um dispositivo de GPS (USB, Bluetooth ou RS232). É geralmente utilizado para se obter as coordenadas da localização actual. No entanto, permite também obter a hora exacta do satélite, que deverá ser mantida por um relógio atómico, permitindo assim tornar a máquina que receba este sinal num *stratum-0*¹.
- **LinuxPPS:** Apesar de o sinal que chega do satélite ao dispositivo GPS permitir um erro de microsegundos, o percurso desde o *kernel* à aplicação induzem um erro demasiado grande (pode chegar a milisegundos[3]). No entanto, é possível compilar o *kernel* do Linux com um *patch* (*LinuxPPS*) que oferece o suporte do *Pulse Per Second* (PPS). Com este *patch*, as aplicações já conseguirão ter precisão suficiente para recolher a hora exacta do dispositivo GPS, com precisão na ordem dos microsegundos. Infelizmente, muito poucos dispositivos suportam esta precisão².

Para além deste problema de precisão por parte das aplicações, foi importante garantir-se que as interrupções do sistema operativo ou outros factores não influenciavam as medições. O código abaixo, usado para testar esta situação, permitiu concluir que não se pode contar com menos do que 0,2ms de atraso na execução. Sendo <1ms, mostrou-se que o facto de se trabalhar com uma linguagem de *scripting* não seria um impedimento ao desenvolvimento desta aplicação.

```
$t0 = microtime(true);
usleep(7*1000); // 7ms
```

¹Na hierarquia de servidores NTP, um *stratum-0* é um servidor de tempo que oferece uma sincronização considerada perfeita e que poderá servir de referência para níveis hierárquicos abaixo.

²Aparentemente[3], o dispositivo mais compatível é o **Garmin GPS 18 LVC**.

```
$t1 = microtime(true);
printf("t0=%.6fs\nt1=%.6fs\n", $t0, $t1);
printf("t1-t0=%.6fms\n", ($t1-$t0)*1000);
```

1.3.2 Funcionamento da aplicação de medição

Cada MP é identificado por um nome (uma *string* de caracteres) que deverá ser único na rede. A aplicação³ que corre em cada um dos MP funciona tanto como servidor como cliente:

- Como **cliente** apenas envia constantemente pacotes de prova para os vários MP configurados, em cada 0,5s (tempo configurável e incrementado de um valor aleatório pequeno, de forma a evitar que os pacotes de prova se sincronizem com outros eventos da rede). Estes "pacotes" são, na verdade, datagramas UDP (protocolo de transporte não fiável), dada a necessidade de avaliar a perda de pacotes e de evitar retransmissões, que iriam deturpar as medições do *delay* e *jitter*. A estrutura destes pacotes é descrita mais adiante.
- Como **servidor** recebe os pacotes de prova, armazenando o seu conteúdo. A cada 60s (tempo configurável) são calculados o mínimo, média e a máximo do *delay* entre cada um dos MP, para além do *jitter* médio e da percentagem de pacotes perdidos. Neste momento, dependendo da configuração, estes valores são enviados para um repositório remoto via HTTP ou armazenados localmente. Neste caso, também são devolvidos aos clientes respectivos, através de uma ligação de controlo sobre TCP, para que os possa mostrar no seu site local.

1.3.3 Estrutura dos pacotes de prova

Os pacotes de prova são enviados sobre a forma de *strings* com a estrutura

```
nome:seqnumber:timestamp
```

Como exemplo: `alice:31:1246929565` . Um pacote deste género reúne toda a informação necessária para que o receptor consiga analisar a qualidade da ligação (mantendo um formato pequeno, para evitar que se prejudique o funcionamento da rede): através do `name` é identificado o cliente que o enviou, com o `seqnumber` (número sequencial) é possível detectar se existem perdas durante o percurso e com o `timestamp` local identifica-se o tempo de propagação do pacote na rede (valor que será subtraído ao *timestamp* do MP receptor).

1.3.4 Estrutura dos dados armazenados

Os dados armazenados localmente são guardados em ficheiros de texto com o nome (exemplo) `alice-bob.metrics` na directoria `/tmp/metrics/`. Cada linha do seu conteúdo tem a estrutura

```
time=1246929565 group=50 count=670 min=0.105 med=0.192 max=0.343 jitter=0.067 lost=0
```

onde se destacam os campos `group` (grupo 50 indica que se tratam de cálculos feitos de 50 em 50 segundos) e `count` que representa o número de pacotes recebidos desde o último cálculo.

Os dados armazenados remotamente são enviados via HTTP para um servidor remoto, com pedido do género

³A aplicação foi desenvolvida em PHP, usufruindo-se de muitos métodos e funções úteis para comunicação em rede.

```
http://192.168.1.1/input.php?from=alice&to=bob&time=1246929565&group=50&
&count=670&min=0.105&med=0.192&max=0.343&jitter=0.067&lost=0
```

e armazenados numa tabela MySQL com as mesmas colunas.

1.3.5 Ambiente de desenvolvimento

Conforme já descrito na secção 1.3.1, o sistema de sincronização por GPS é algo complexo de implementar para se conseguir a precisão desejada de <1ms de erro. Para que essa investigação e desenvolvimento pudesse decorrer paralelamente com o desenvolvimento da aplicação dos MP, foi importante criar-se um ambiente de testes que simulasse comportamentos idênticos a uma rede real. O módulo `netem` para a ferramenta `tc` (*Traffic Control*) do Linux aparenta [2] ser muito fidedigna para simulações de redes reais, para além de ser de muito acessível utilização. Por exemplo, configurações como

```
$ tc qdisc add dev lo root netem loss 0.3% 25% delay 30ms 2ms distribution normal
```

permitiram que durante a fase de desenvolvimento fosse testado o envio de pacotes de prova para o próprio MP (`localhost`), com probabilidade de perda de de pacotes de 0.3% (com uma correlação de 25%) e atrasos de 30ms com uma variação de 2ms que siga uma distribuição normal. Um cenário relativamente aproximado do real, com algum exagero quanto às perdas, mas interessante para serem visíveis, já que geralmente quase não acontecem. Verifica-se mais à frente que, de facto, as medições feitas pelo MP com esta configuração coincidem com estes valores, como seria de esperar.

1.4 Utilização

1.4.1 Instalação e configuração

. instalacao A instalação da aplicação nos MP é trivial, dado tratar-se de um *script* PHP. Apenas é necessário que se encontre pré-instalado o interpretador desta linguagem, com suporte para *forks*. O *daemon* de controlo (que funciona como servidor e cliente - ver 1.3.2) não necessita de receber qualquer parâmetro para arrancar, utilizando o ficheiro de configuração. No entanto, é possível executá-lo com opções específicas:

```
Usage:
$ ./qos-metrics
$ ./qos-metrics server
$ ./qos-metrics client
$ ./qos-metrics client <server_name>
```

A instalação em `OpenWrt` pode ser feita através do sistema gestor de pacotes interno do sistema, através do comando: `$ ipkg install qos-metrics.ipk`, onde o último parâmetro é o nome do pacote pré-compilado com a `build root` desta distribuição [5].

A configuração de cada MP é feita no ficheiro `config.php`, com a estrutura do seguinte exemplo:

```
$servers['braga'] = "78.12.11.313:33301";
$servers['porto'] = "78.12.15.313:33301";
$servers['coimbra'] = "79.200.66.67:33301";
$servers['lisboa'] = "79.200.62.172:33301";
$servers['faro'] = "86.121.11.72:33301";
```

```

$servers['viana'] = "193.133.172.82:33301";

$config['myname'] = 'aveiro';
$config['myport_control'] = 33300; // default: 33300 (TCP)
$config['myport_metrics'] = 33301; // default: 33301 (UDP)
$config['polltime'] = 500; // default: 500 (ms)
$config['random'] = 80; // to avoid sync with other events
$config['seqturn'] = 99999;

$config['showdots'] = false;
$config['showreceivedpackets'] = false;
$config['showstores'] = true;

```

Se a aplicação for instalada em OpenWrt será possível configurar estes parâmetros através da sua ferramenta de configuração uci [5].

1.4.2 Configuração da sincronização do relógio por GPS

O sistema de sincronização do relógio por GPS (já parcialmente descrito na secção 1.3.1) pressupõe a existência de um dispositivo GPS já ligado ao sistema, tal como a pré-instalação das ferramentas `ntp` e `gpsd`. Com o comando `$ gpsd /dev/rfcomm0` inicia-se o *daemon* que recolhe os valores recebidos do satélite (não requer configuração). Comandos como o `xgps` ou o `$ cgps localhost 2947` permitem visualizar os dados que estão a ser recebidos, tal como o número de satélites visíveis. A instalação do `ntp` e do `gpsd` em distribuições comuns é trivial com o uso dos gestores de pacotes. Em OpenWrt deverá ser feita a instalação do `ntpcient`.

Instalado e configurado conforme abaixo indicado (ficheiro `ntp.conf`) o `ntp` faz pedidos ao `gpsd` a cada 60 segundos, por forma a manter o relógio constantemente sincronizado:

```

# sem PPS:
server 127.127.28.0 minpoll 4 prefer
fudge 127.127.28.0 refid GPS flag3 1

# com PPS:
server 127.127.28.0 minpoll 4 prefer
fudge 127.127.28.0 refid PPS1 flag3 1

```

O estado do sincronismo do relógio pode ser consultado através do comando `ntpq -p [3]`. Note-se a diferença entre o *jitter* com e sem PPS (ver secção 1.3.1). Este valor simboliza a diferença entre a hora recebida e a hora actualmente definida no relógio físico local. Naturalmente, quanto mais próximo de zero se encontrar, mais exacto estará:

remote	refid	st	t	when	poll	reach	delay	offset	jitter
*SHM(0)	.PPS.	0	l	13	16	377	0.000	-0.015	0.004
+GPS_NMEA(0)	.GPS.	1	u	-	64	377	44.461	1.580	5.223

1.4.3 Site com medições dos níveis de QoS

O *site* disponível (tanto instalado localmente nos *routers* com *firmware open-source* como centralizado num único servidor - ver secção 1.3) reúne muita informação útil ao administrador da rede, por cada MP.

As três matrizes apresentadas (ver figura 1.3) apresentam as médias dos valores de *delay*, *jitter* e *packet loss* durante o último dia. Cada célula destas matrizes apresenta ainda um valor

avg delay	alice	bob	charlie
alice	0.27ms (0.01)	11.73ms (0.13)	53.15ms (1.33)
bob	13.88ms (0.26)	12.52ms (0.67)	66.94ms (1.17)
charlie			0.08ms (0)

jitter	alice	bob	charlie
alice	0.03ms (0.01)	0.36ms (0.13)	0.23ms (0.03)
bob	0.27ms (0.08)	3.49ms (0.24)	0.44ms (0.1)
charlie			0.02ms (0)

lost	alice	bob	charlie
alice	0% (0)	0% (0)	0% (0)
bob	0% (0)	6.44% (0.01)	0% (0)
charlie			1.43% (0.48)

Figura 1.3: Tabelas apresentadas no site, para três MP (alice, bob e charlie).

entre parêntesis, que representa a diferença existente entre o valor em causa durante a última hora e durante os últimos 5 minutos (valores configuráveis). Desta forma, consegue-se ter uma noção das flutuações recentes que tem havido na rede. Naturalmente, que estes valores poderão ser estendidos para valores como o último dia e os últimos 10 dias (valores utilizados pelo mesmo sistema do RIPE).

Os critérios para a escolha das cores seguem a lógica:

- Quando a diferença entre os dois valores (média do último dia e dos 10 últimos dias) é superior ao desvio padrão, é considerado que a flutuação se encontra acima do normal e o valor é mostrado a vermelho, indicando que existe uma degradação da qualidade da rede nesse parâmetro entre esses dois MP. Por outro lado, na situação inversa o valor é mostrado com a cor verde, indicando que existe uma melhoria na qualidade da rede. Nas restantes situações é simplesmente apresentado a preto.
- Ainda, quando a diferença entre o valor num sentido e o valor do sentido inverso é superior ao desvio padrão, a célula é apresentada com a cor azul indicando que existe uma diferença significativa entre os dois sentidos desse caminho.
- Por último, é possível definir limites para cada um dos valores, acima dos quais a cor do valor é apresentada a vermelho, independentemente da distância em relação ao desvio padrão. A definição deste parâmetro é especialmente útil para a definição de limites na percentagem de perdas, onde geralmente há muitas restrições, a ponto de considerar que

um nível de perdas superior a 0,01% é inadmissível. A título de exemplo, tanto na figura 1.3 como na 1.4, o limite definido foi de 1%. Tendo sido induzido um grande nível de perdas (com recurso ao sistema de simulação descrito na secção 1.3.5), verifica-se que de facto os valores são indicados a vermelho, alertando para uma falha grave.

Muitas vezes é útil consultar os valores de *delay*, *jitter* e *packet loss* apenas de um MP para todos os outros. Seleccionando o nome de cada um dos MP nas tabelas cima, é apresentada uma segunda tabela que apresenta os valores das métricas no sentido desse MP para todos os outros. O resultado apresentado será uma tabela com o aspecto da figura 1.4, seguindo a mesma lógica de valores e cores indicada acima.

from: bob

to:	delay	jitter	lost
alice	12.21ms (0.24)	0.28ms (0.02)	0% (0)
bob	12.48ms (0.38)	3.46ms (0.47)	4.98% (0.18)
charlie	19.44ms (1.36)	0.37ms (0.12)	0% (0)

Figura 1.4: Tabela que apresenta os valores de um MP para todos os outros.

1.5 Dificuldades de Implementação

Embora inicialmente se previsse que os maiores obstáculos ao desenvolvimento do projecto fossem centrar-se na análise dos dados recolhidos, rapidamente se percebeu que a grande dificuldade seria mesmo a sincronização dos relógios. O mecanismo PPS (funcionamento descrito nas secções 1.3.1 e 1.4.2) apenas está disponível em alguns tipos de GPS. Acabou por se descobrir que os dois dispositivos de que se dispunha para o desenvolvimento do projecto não disponham desta funcionalidade. Tendo sido projectado um ambiente de simulação fidedigno (ver secção 1.3.5) e assumindo que seria necessário adquirir-se dois dispositivos que suportassem PPS (geralmente mais caros), optou-se por desenvolver o sistema assumindo a sincronização e garantindo as condições (*software* instalado e totalmente configurado) pronto a receber um sinal PPS do dispositivo GPS.

Optou-se pela linguagem de *scripting* PHP por já apresentar embutida muitas classes, métodos e funções que facilitariam o desenvolvimento da aplicação. No entanto, apesar da programação ficar facilitada do ponto de vista da comunicação, o facto de a linguagem apenas suportar *forks*, mas não *threads* acabou por complicar muito o seu desenvolvimento. Reparou-se também que, quando a rede de MPs se encontra toda montada, a aplicação acaba por gastar mais recursos do que se esperava. Apesar de cumprir os requisitos para o papel que pretende desempenhar, se o projecto tiver continuidade no futuro, seria interessante começar por "portar" o código para uma linguagem diferente. Talvez o mais aconselhável fosse o *Python* por ser leve, também de objectos, mas mais virada para o desenvolvimento de aplicações. Para além disso, a linguagem é completamente suportada pelo *firmware* OpenWrt e pfSense.

1.6 Conclusões

Apesar dos imensos obstáculos que foram aparecendo durante o desenvolvimento deste projecto, a base implementada é mais que suficiente para se conseguir uma boa análise dos níveis de QoS numa rede em ambiente SOHO (*Small Office or Home*). No estado actual, provavelmente não será escalável a ponto de ser utilizada num ISP. No entanto, após afinação de certos parâmetro (especialmente na robustez e gestão de recursos computacionais) este poderá ser o ponto de partida para o desenvolvimento de uma solução final e até pronta para estar disponível no repositório *SourceForge*.

Os conceitos de medição de métricas de QoS integra-se completamente no âmbito do módulo de Redes Multiserviço, apesar de nem tanto no de Redes IP Avançadas. Apesar de não ser prevista diferenciação de tráfego, facilmente se adapta o sistema a um ambiente *DiffServ*, bastando classificar por classes os pacotes de prova enviados entre os MP.

Bibliografia

- [1] K. Marzullo, S. Owicki "Maintaining the Time in a Distributed System", 1985
- [2] S. Hemminger "Network emulation with NetEm", 2005
- [3] Using a Garmin GPS 18 LVC as NTP stratum-0 on Linux 2.6
<http://time.qnan.org/>
- [4] Microsecond Precision with PPS and NTP
<http://www.linuxquestions.org/linux/node/1624>
- [5] OpenWrt Documentation
<http://kamikaze.openwrt.org/docs/openwrt.html>
- [6] Network Time Protocol (Clock Strata)
http://en.wikipedia.org/wiki/NTP_server#Clock_strata
- [7] Tekron GPS Clock
<http://www.tekroninternational.com/>
- [8] TimeTools NTP Time Servers
<http://www.timetools.co.uk/>