Load balance com Packet Filter

Autoria de Daniel Bristot de Oliveira
16/11/2006
Última Atualização 18/11/2006

Continuando a série de artigos sobre load balance, irei mostrar como utilizar o Packet Filter para balanceamento de carga para servidores Web.

Este artigo tráz algumas novidades, como Qualidade de Serviço e um novo ambiente, mais flexível.

A parte de Alta disponibilidade fica por conta do artigo: FreeBSD HA: Alta disponibilidade com CARP, Ifstated e pfsync, formando um ambiente clusterizado, provendo Alto desempenho e Alta disponibilidade.

IntroduçãoPor que substituir o Pen?
Por questões de performance, o pen(8) é um programa userland, então ele está sujeito a paginação de memória, assim também a Swap, e todas as burocracias, e checagens que o kernel faz em relação as trocas de informações de I/O. O PF executa em nível do kernel, assim, retiramos problemas citados acima, e adicionamos um melhor acesso as informações de I/O de rede, integração com o firewall, o que possibilita um melhor tratamento de QoS, DoS.

O Packet Filter
O Packet Filter, ou pf(4), é o filtro de pacotes, que foi criado para o OpenBSD, em substituição ao ipf(4), isto por problemas de licença. O PF trás uma série de características junto com o controle de fluxo de pacotes, como NAT, controle de banda e priorização de pacotes, normalização de pacotes, tratamento de DoS, etc.

Foge do escopo deste artigo uma apresentação detalhada do PF, caso você não conheça o PF, sugiro que leia a FAQ, que está disponível na versão em português em: http://openbsd.org/faq/pf/pt/ e em inglês em: http://openbsd.org/faq/pf

O ambiente:

Temos dois balanceadores de carga, compartilhando o mesmo endereço, utilizando carp(4). E dentro da rede local, temos seis servidores, divididos em dois grupos, o grupo dos servidores Web, e dos servidores de Mídia.

Os servidores Web, serão responsáveis somente pelo processamento de requisições de páginas, e pelo processamento por trás destas páginas, como por exemplo, códigos PHP, Python, enfim, o processamento de código executável.

Os servidores de Mídia, serão responsáveis apenas pelo recebimento de requisições de imagens, vídeos, etc., enfim a parte de mídia que envolve uma página.

Bom, o porque disto é que podemos otimizar os servidores, para que eles sejam mais especializados e obtenham um melhor desempenho. Por exemplo, os servidores de mídia, deveremos ter discos velozes e um I/O de rede afinado para envio de grandes streams, já os servidores Web, podemos fazer um melhor afinamento em processamento, e cuidarmos melhor de cache de páginas já compiladas, como por exemplo, utilizando PHP, com o e-accelerator, e armazenando as páginas compiladas em um "memory disc", enfim, uma série de benefícios podemos tirar disto.

Este ambiente é apenas um dos possíveis ambientes, você provavelmente pensará em N tipos diferentes de se formar um cluster, mais adequado as suas necessidades, porém com este ja posso aplicar algumas técnicas.

Como as coisas devem acontecer.

Eu terei dois endereços validos na internet, www.fug.com.br e o midia.fug.com.br, que serão responsáveis pelo endereçamento dos meus dois grupos de servidores, os web e os de midia, respectivamente.

As requisições irão chegar na minha interface externa do balanceador de carga, neste ponto, iremos redirecionar as conexões destinadas aos servidores de midia, contidos na tabela <serv_midia>, e as conexões destinadas aos servidores Web, contidos na tabela <serv_web>.
ConfiguraçãoMacros e tabelas
Vamos começar a configuração, ela será feita dentro do arquivo de configuração do pf(4), o pf.conf(5), dentro de /etc.

Abaixo vamos criar as macros e tabelas para o nosso esquema de balanceamento.

```
web_addr= "200.18.15.17"          #Endereço do servidor de web
midia_addr= "200.18.15.18"        #Endereço do servidor de mídia
ext_if="xl0"                      # Interface ligada a Wan
internal_net="192.168.0.0/24"
```

#Tabela dos servidores Midia
table <serv_midia> { 192.168.0.20, 192.168.0.21 }

#Tabela dos servidores de Web
table <serv_web> { 192.168.0.10, 192.168.0.11, 192.168.0.12, 192.168.0.13 }

Nota: Estas tabelas podem ser gerenciadas em momento de execução, com o utilitário pfctl(8), damos aqui alguns exemplos: (Visualizando conteúdo da tabela serv_midia) # pfctl -t serv_midia -T show 192.168.0.20192.168.0.21 (Adicionando novo host à tabela serv_web)
pfctl -t serv_web -T add 192.168.0.22
1/1 addresses added. (Removendo um host da tabela serv_web) # pfctl -t serv_eb -T add 192.168.0.12
1/1 addresses deleted.

Bom, agora que temos as tabelas, podemos avançar, aqui vou seguir a ordem de como as regras devem aparecer no arquivo de configuração, então o próximo ponto é o NAT.
NAT (Network Address Translation)

Todos nossos servidores estarão em uma rede interna ou DMZ, eles não estarão ligados diretamente a internet, e na verdade responderão pelos ips externos dos balanceadores de carga. O NAT cuidará desta tradução de entrada e saída para internet. Vamos a configuração:
nat pass on $ext_if from $internal_net to any -> ($ext_if) O que esta regra faz é:

Redireciona, tudo o que chega na interface externa, com protocolo tcp, de todos, para o endereço dos servidores de mídia, na porta 80, irá
ser enviado para um dos servidores de mídia, escalonando em round-robin, isto é circularmente.

OBS: Ele sairá utilizando o mesmo endereço que ela chegou, por exemplo, se chegou para o midia.fug.com.br, ele vai sair com o endereço do midia.fug.com.br.

Agora vamos partir para a configuração de balanceamento de carga.

Temos dois tipos de balanceamento, um para os servidores de mídia e outro para os servidores web, vamos seguir esta ordem.

Balanceamento de carga - Servidores de Midia

Como falei anteriormente, os servidores de mídia irão servir apenas arquivos de mídia (obvio :-), então, partindo do pre-suposto que os dois servidores trabalham com o mesmo conjunto de dados, não importa de onde as imagens vem, elas apenas precisam ser acessadas. O método adotado para o escalonamento de requisições, tendo em consideração que ele deve ser estático, e que balanceadores nada sabem sobre como estão os servidores, vai ser o escalonamento circular (round-robin). Aqui está uma desvantagem do PF, não é possível criar prioridades para servidores, o que é possível no pen(8).

Partindo para a configuração:
rdr on $exit_if proto tcp from any to $midia_addr port 80 -> <serv_midia> round-robinO que a regra faz é: Redireciona, tudo o que chega na interface externa, com protocolo tcp, de todos, para o endereço dos servidores de mídia, na porta 80, irá ser enviado para um dos servidores de mídia, escalonando em round-robin.

Balanceamento de carga - Servidores Web

Agora iremos ver uma outra característica de balanceamento de carga. Neste exemplo, estamos fazendo um cluster distribuído, onde temos vários nós de processamento, que compartilham um meio de comunicação e endereçamento em comum, a rede e o endereço válido na internet, também devem compartilhar o mesmo conteúdo de arquivos, via NFS por exemplo, porém, eles não possuem um ambiente de compartilhamento de memória.

Uma vez que um cliente se conecte ao meu ambiente, ele será enviado para um servidor web, que armazenará algumas informações, como por exemplo, dados de uma sessão de WebMail. Por não existir um ambiente de compartilhamento de memória, somente um servidor web saberá sobre esta sessão, então, para não haver uma quebra de sessão, todas as próximas conexões deste cliente, devem ser enviadas para o mesmo servidor web.

Para isto utilizaremos o mesmo balanceamento de carga com algoritmo circular dos servidores de mídia, porém, com

rastreamento de cliente. Após uma primeira conexão, todas as proximas serão enviadas para o mesmo servidor Web, assim mantendo a sessão.

Partindo para a configuração:
rdr on $exit_if proto tcp from any to $midia_addr port 80 -> <serv_midia> round-robin sticky-address
O que a regra faz é: Redireciona, tudo o que chega na interface externa, com protocolo tcp, de todos, para o endereço dos servidores web, na porta 80, irá ser enviado para um dos servidores web, escalonando em round-robin, isto é circularmente, e fazendo o rastreamento de clientes.

Quando um cliente faz a primeira conexão, o PF checa em uma lista de clientes, a existência de uma conexão anterior, neste caso ela não existe, o escalonador vai dizer quem é o próximo servidor na lista, e o cliente será enviado para ele, após isto uma entrada será adicionada a tabela de clientes, dizendo que este cliente foi enviado para o servidor web X, por exemplo. Em uma próxima conexão, quando chegar, o PF irá verificar se o cliente está na lista de clientes, neste caso, existe uma entradas dizendo que ele foi enviado anteriormente para o servidor web X, então, ele será enviado novamente para este servidor X, mantendo a sessão.

As questões de balanceamento de carga estão ok, agora vamos dar mais uma olhada no PF.
Tabela Source, a tabela rastreamento de clientes.
Esta tabela de rastreamento de clientes, chama-se tabela Source, em uma referência a Source Track.

Para visualizar as entradas na tabela Source, use o pfctl(8)

```
# pfctl -s Source
```

Temos duas questões para tratar nesta tabela:
O tempo de vida de uma entrada
Quantos clientes poderemos rastrear
O tempo de vida de uma entrada vai definir por quanto tempo devemos armazenar a informação sobre o cliente, assim garantindo a sessão. Este valor é estipulado com a macro set timeout src.track, e este tempo é medido em segundos, veremos abaixo um exemplo de meia hora.

Sobre o número de clientes, como o PF executa em nível do kernel, esta tabela ocupa espaço dentro do endereçamento de memória do kernel. Por isto devemos ter um cuidado especial como ela, pois o crescimento desta lista, ao ponto de chegar ao fim da memória real do sistema, causará pânico de exaustão de pool(9). o que mataria meu sistema. Para resolver este problema, podemos adicionar um limite de entradas nesta lista, com a diretiva set limit src-nodes.

Para prevenir este problema, veja um exemplo destes dois limites adicionados ao arquivo de configuração do PF.

```
set limit src-nodes 20000    # limita em vinte mil o número de entradas na tabela
set timeout src.track 1800   # cada entrada permanece 1800 segundos ou meia hora na tabela.
```

Para visualizar valores de timeout e limites:

```
# pfctl -s timeouts
```
Bom, existe outra tabela muito importante, que é a tabela State, ela mantém o estado das conexões "state full", ela nos trás ótimos benefícios de performance, porém isto custa memória e podem causar o mesmo problema da tabela Source, ela foge um pouco do escopo deste artigo, mas informações podem ser vistas aqui:
http://openbsd.org/faq/pf/pt/filter.html#state

Qualidade de serviço

Seguindo a ordem que as coisas acontecem, um cliente se conecta ao meu servidor Web, e receberá uma página, que geralmente, tem imagens, que serão requisitadas pelo browser ao servidor de midia. Isto mostra que o servidor web será acessado antes dos servidores de mídia, e que os servidores de mídia serão acessados, na grande maioria das vezes, somente se for feita uma referência a eles no conteúdo dos servidores Web.

O conteúdo exportado do servidor de Web, geralmente possui um tamanho pequeno, pois páginas são texto puro, já os servidores de mídia exportarão imagens e vídeos, que tem um tamanho relativamente maior que o das páginas.

Porém, as páginas Web são um meio interativo com o usuários, o que leva a necessidade de uma maior prioridade no tratamento destas requisições.

Isto nos leva a seguinte conclusão:
Servidores de Mídia precisarão de maior largura de banda, e
Servidores Web precisarão de uma maior prioridade sobre as requisições aos servidores de Mídia.

Outra questão é o DoS:

Se vários clientes iniciarem download de grandes arquivos de mídia, por exemplo um filme, ao ponto de ocupar toda a minha banda, meus servidores Web se tornarão indisponíveis! Para resolver este problema, podemos garantir banda para os dois tipos de servidores, dividindo a minha banda total entre eles.

Estes problemas podem ser resolvidos com o PF, utilizando ALTQ, e CBQ. A FAQ do PF contém um conteúdo mais detalhado sobre isto, aqui apenas mostrarei a aplicação destas técnicas ao nosso ambiente.

Aqui neste exemplo, vamos supor que temos um link de 5Mb. Vou dividir o tráfego em duas partes, uma de 1,5Mb, dedicado aos servidores Web, e um de 3,0Mb para os servidores de mídia e 0.5Mb para outros protocolos, assim, garantimos maior banda para os servidores de mídia, e conseguimos alocar um espaço para os servidores Web.

Outra questão é a priorização dos pacotes destinados aos servidores Web, em relação aos outros pacotes.

Vamos a configuração:

```
altq on wan0 cbq bandwidth 5120Kb queue { http, default }
queue http bandwidth 4608Kb { http-web, http-midia }
queue  http-web bandwidth 1536Kb priority 2 cbq(borrow, red)
queue  http-midia bandwidth 3072Kb priority 1 cbq(borow, red)
queue default bandwidth 512Kb priority 1 cbq(default)
```

Acima, na primeira linha, adicionamos o controle de banda na interface "wan0", que estará ligada a internet, esta interface tem 5Mb de largura de banda, e a partir dela criamos duas filas, uma chamada http, que será responsável pelo tráfego dos servidores, e uma default para os demais protocolos. Na segunda linha eu defino a fila http, com largura de banda de 4,5 Mb, e divido ela em duas sub-filas, uma chamada http-web e outra http-midia. A fila http-web será reservada aos servidores Web, com uma largura de banda de 1.5Mb, e prioridade maior que as outras duas, ainda adicionei o parâmetro Borow, para empréstimo de banda e red para detecção aleatória prematura de congestionamento. A fila http-midia será reservada aos servidores de Mídia, com uma largura de banda de 3.0Mb, e prioridade menor que a fila http-web, também adicionei o parâmetro Borow, para empréstimo de banda e red detecção aleatória prematura de congestionamento.

A última fila a default, será o caminho para o tráfego de outros protocolos, com uma largura de 0,5Mb

Agora, que temos as filas criadas, basta aplicar o tráfego as filas, como no exemplo abaixo:

```
# abro a porta para o protocolo http
pass in quick on wan0 proto tcp from any to any port 80 keep state
pass in quick on lan0 proto tcp from any to <serv_web> port 80 keep state
pass in quick on lan0 proto tcp from any to <serv_midia> port 80 keep state

# aplico as filas aos servidores Web
pass out quick on lan0 proto tcp from any to <serv_web> port 80 keep state queue http-web
pass out quick on lan0 proto tcp from any to <serv_midia> port 80 keep state queue http-midia
```

A qualidade de serviço está feita.
FinalAcho que ele irá ficar um pouco complexo para pessoas que nunca trabalharam com o PF. Sugiro que leiam antes o FAQ do PF, ele é muito claro, e útil, o man do PF também é bem simples. O ambiente é somente de exemplo :-)

O PF se comportou muito bem ao load balance, porém, o estado da arte não esta completo:

Se um servidor web cair, podemos tirar ele da lista de servidores (table <serv_web>), mas não é possível retirar o seu rastro (Tabela Source), veja: http://www.fug.com.br/historico/html/freebsd/2006-05/msg00877.html

O pfsync sincroniza somente a tabela state, e não a source, assim, a troca de estados dos balanceadores de carga não fica transparente para os clientes, já que a tabela source é quebrada. O escalonamento sem prioridades limita um pouco o sistema.

Porém isto pode ser contornado.

Fico por aqui, Um grande abraço! :)