Reverse Proxy

COMUNICAÇÕES POR COMPUTADOR 2017

Trabalho Prático 2

André Rodrigues Freitas a74619
Paulo Jorge Machado Guedes A74411
Sofia Manuela Gomes de Carvalho a76658

Resumo

Para este projeto foi solicitado que se criasse um intermediador entre as conexões dos clientes e os servidores *back-end*, denominados monitores. Este intermediador tem o nome de "Reverse Proxy", que tem como objetivo receber todos os pedidos de clientes que queiram fazer pedidos aos monitores, sendo o *reverse proxy* a escolher qual monitor vai tratar do pedido de cada cliente.

O "Reverse Proxy" tem a função de receber qualquer mensagem por *socket* TCP do cliente, e enviar a respetiva mensagem para o monitor mais adequado a tratar do pedido do cliente. De seguida, recebe a resposta do monitor e reencaminha para o cliente, servindo como um intermediador na comunicação.

O *reverse proxy* terá que manter uma lista dos monitores disponíveis e das condições da rede desses monitores, assim como as conexões TCP já associadas a esse monitor, podendo realizar um julgamento mais correto de qual monitor escolher para tratar dos pedidos do cliente, tentando gerir ao máximo o trabalho dos monitores, não os sobrecarregando. Para tal, o *reverse proxy* terá que também comunicar com os monitores via UDP, de modo a verificar continuamente o estado da conexão dos respetivos monitores.

Conteúdo

Índice de Figuras

Introdução

Para responder aos serviços de vários clientes, um servidor não é o suficiente, sendo necessários vários servidores capazes de dar resposta aos pedidos. Para gerenciar a utilização dos vários servidores, todas as comunicações dos clientes terão um ponto de entrada, sendo este o servidor *front-end*, com um endereço IP único e bem conhecido, cuja tarefa é atender as conexões dos clientes e desviá-las para um dos servidores *back-end* disponíveis. Este servidor *front-end* chama-se "Reverse Proxy". A escolha dos servidores *back-end*, denominados monitores, pode ser cega, baseado num algoritmo de *Round-Robin*, mas optou-se por uma abordagem mais otimizada, verificando o estado da conexão dos vários monitores continuamente, via UDP, utilizando esses valores no momento da escolha do servidor para atender a conexão do cliente. Assim, o *reverse proxy* poderá estabelecer uma comunicação via TCP com os clientes e monitores, intermediando a comunicação entre estes, satisfazendo os pedidos do cliente da forma mais eficiente possível.

Tabela Monitor

Para obter o melhor monitor aquando a conexão de um novo cliente, teremos que aceder às informações sobre o estado da conexão de cada monitor. Para tal, foi criado uma classe para guardar todos os valores necessários para classificar o estado da rede.

É criada uma classe "Tabela Monitor" associada a cada monitor, onde se guardam os valores relevantes para a classificação do estado da rede, sendo eles:

- Rtt: *round trip time,* o tempo em milissegundos que demora um pacote a ser enviado desde o reverse proxy até ao monitor;
- PacketLoss: percentagem de pacotes que são perdidos na rede;
- TCPCon: número de conexões TCP que estão no momento ativas no monitor;
- Time: a hora na qual foi realizada a última notificação de que o monitor estava ativo;
- Duplicados: percentagem de pacotes que são duplicados da rede, congestionando esta mesma;

Assim, com estas variáveis, criamos uma fórmula matemática que calculava o estado da rede, utilizando o "Rtt", "PacketLoss", "TCPCon", "Duplicados", sendo ela:

```
PacketLoss + Rtt + 10*TCPCon + 0.5*Duplicados
```

Figura 1 - Classificação do Estado da Conexão

Esta fórmula foi criada através de tentativa erro, colocando exemplos de estados de conexões, e verificando qual seria o monitor escolhido, adaptando a fórmula.

Assim, quão menor for o "Status", resultado da fórmula, melhor será a conexão.

PDU

Tabela 1 - PDU Periódico

Tipo – 1 byte para determinar o tipo do pacote, neste caso, o byte 0.

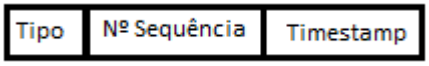

Figura 2 - PDU Probing

Tipo – 1 byte para determinar o tipo do pacote, neste caso, o byte é 1.

Nº Sequência – 1 byte para determinar o número de sequência do pacote.

Timestamp – Número de bytes indeterminado, normalmente entre 22 a 23, hora em que o pacote foi enviado no reverse proxy.

UDP

Definidos os parâmetros necessários para verificar o estado das conexões dos monitores, foi necessário criar uma conexão UDP entre cada monitor e o reverse proxy, para obter os dados da conexão.

Assim, cada monitor enviará de 10 em 10 segundos, um *datagrama UDP,* com o PDU Periódico, para informar o reverse proxy que o monitor está ativo e pronto a receber pedidos dos clientes.

Em adição, é enviado a cada monitor, desde o primeiro datagrama com o PDU Periódico recebido, de 2 em 2 segundos, dois pacotes com o PDU Probing, na qual o monitor apenas irá retornar o *datagrama* para informar ao reverse proxy que recebeu o *datagrama*. Caso o reverse proxy não receba o *datagrama* num certo intervalo de tempo (no primeiro pacote com o PDU Probing o intervalo é 100ms, a partir do primeiro pacote, o intervalo é 2*Rtt do monitor para o qual está a enviar o pacote), é considerado um pacote perdido, contabilizando no estado da conexão desse monitor. Caso receba um *datagrama* com o número de sequência igual a um *datagrama* já recebido, é contabilizado para os duplicados dessa conexão.

Antes de enviar os 2 pacotes, é verificado á quanto tempo foi enviado o último *datagrama* com o PDU Periódico desse monitor. Caso o último tenha sido á mais de 30 segundos, não é enviado mais nenhum pacote a esse monitor, considerando esse monitor morto.

Todos os *datagramas*, tanto no reverse proxy, como nos monitores, são enviados e recebidos na porta 80.

Esquema UDP

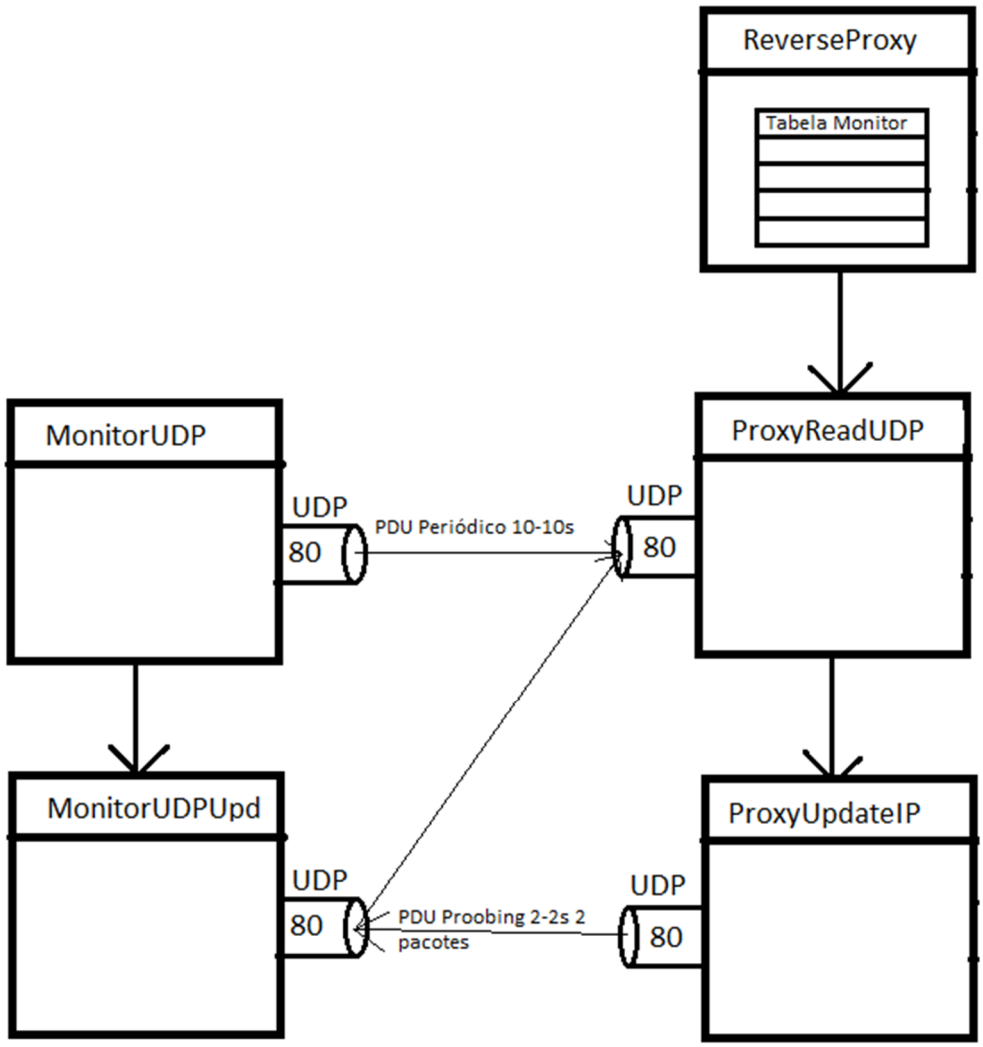

Figura 3 - Esquema UDP

TCP

Guardados os dados sobre as conexões dos monitores, falta implementar o reverse proxy a nível de conexões de TCP. Para tal, necessitamos de receber conexões de clientes a nível TCP e para cada cliente, realizar uma conexão a nível TCP com o monitor com o melhor *status*.

Assim, foi necessário estar á escuta de conexões a nível TCP por clientes, e quando é estabelecida uma conexão TCP com um cliente, é criada uma conexão TCP com um monitor, incrementando a variável "TCPCon" para informar o estado da conexão que o monitor já está em comunicação com um cliente. Assim, quando é estabelecida uma conexão com um cliente, é necessário criar uma *thread* que lê-se os dados enviados pela conexão TCP do cliente, e os reenvia-se para a conexão TCP do monitor, assim como uma *thread* que lê-se os dados enviados pelo monitor e os reenvia-se para a conexão TCP do cliente.

Quando a conexão com o cliente acaba, é fechada a conexão TCP com o monitor, decrementando a variável "TCPCon" desse monitor.

Para testes, foi criada um monitor e um cliente, para testar o estabelecimento das conexões de acordo com status dos monitores.

Esquema TCP

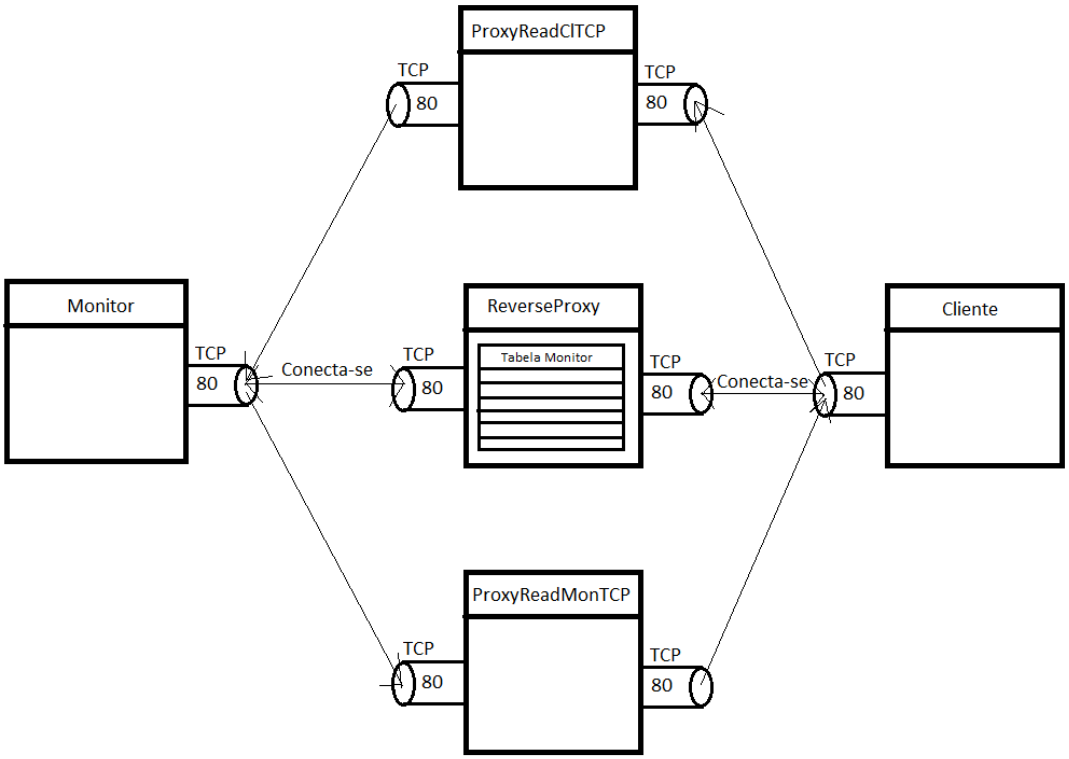

Figura 4 - Esquema TCP

Conclusão

Criado o reverse proxy, este conseguirá servir como um *front-end server* para responder aos pedidos dos clientes, providenciando também a melhor conexão possível aos clientes, de maneira a que os seus pedidos sejam concluídos o mais rapidamente possível. Com isto, os clientes não se necessitam de conhecer os endereços IP de todos os servidores, nem de verificar quais servidores estão disponíveis, visto que o reverse proxy já realiza essas verificações, sendo o seu IP o único conhecido pelos clientes. Com este método, é possível existir um número de servidores elevado, para dar resposta a vários clientes de uma maneira eficiente.