Computação por Computador

2º TRABALHO PRÁTICO

Grupo 32

Francisco Oliveira (a78416)
Raul Vilas Boas (a79617)
Vitor Peixoto (a79175)

Maio 2018

Mestrado Integrado em Engenharia Informática

Conteúdo

Capítulo 1

Introdução

No âmbito da unidade curricular de Computação por Computador, foi-nos pedido para desenvolver um *Reverse Proxy* capaz de monitorizar (MonitorUDP) e encontrar servidores (AgenteUDP), bem como criar um algoritmo de distribuição de carga pelos vários servidores.

O *Reverse Proxy* seria o nosso front-end utilizando um nome e endereço IP conhecidos e serviria como único ponto de entrada para todos os clientes. Este iria depois redirecionar a ligação para um dos nossos servidores de back-end.

Para a monitorização criamos o MonitorUDP responsável por manter um tabela com informação atualizada dos servidores (RTT, ligações TCP, percentagem de pacotes perdidos, etc), que iremos utilizar aquando da escolha do melhor servidor disponível para uma pedido.

Avançamos agora para uma explicação em mais detalhe.

Capítulo 2

Arquitetura da solução

Para este trabalho era necessário criar um *Reverse Proxy* com capacidades de monitorização e Servidores de back-end.

Para tal criamos:

- **AgenteUDP** - Servidor back-end. Responde a *probe requests* do MonitorUDP. Terá um servidor HTTP associado na mesma máquina.
- **MonitorUDP** - constantemente procura por novos servidores, bem como monitoriza os já existentes, mantendo uma tabela de servidores constantemente atualizada.
- **ReverseProxy** - capaz de tratar de toda a distribuição de carga utilizando um algoritmo de seleção, bem como redirecionamento de pedidos de clientes para o servidor back-end selecionado.

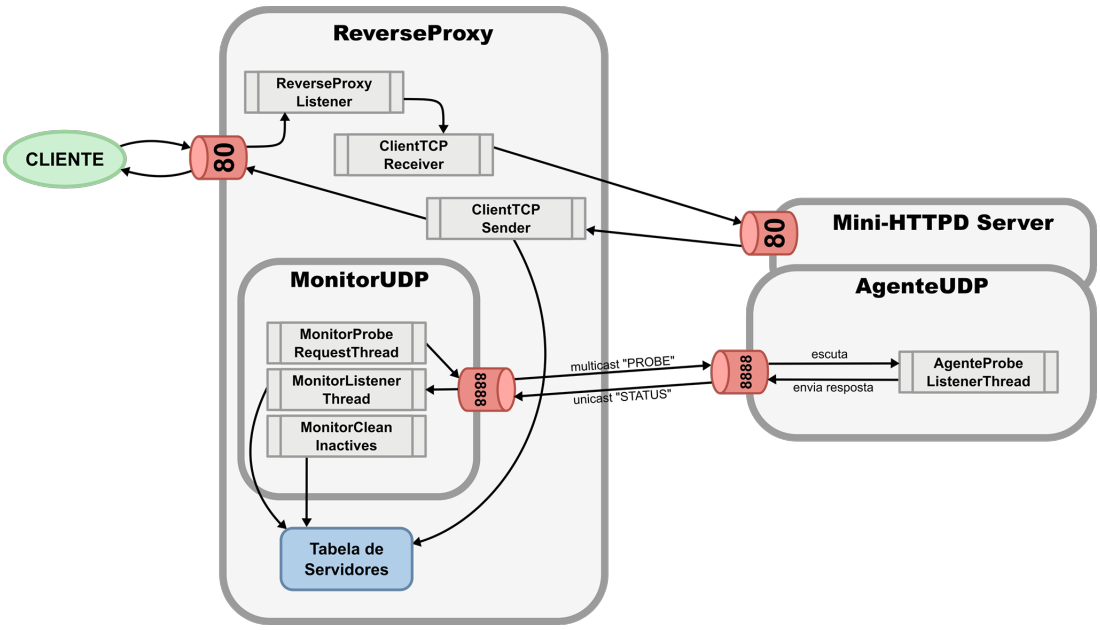

Figura 2.1: *Arquitetura simplificada*

2.1 AgenteUDP

O funcionamento do AgenteUDP baseia-se em 2 atividades. Uma thread AgenteProbeListenerThread escuta por qualquer *probe request* do MonitorUDP e responde com a sua informação e um servidor HTTP capaz de aceitar ligações TCP redirecionadas pelo ReverseProxy devolvendo-lhe as informações pedidas.

No que toca a escuta de *probe requests* (explicação em mais detalhe no próximo capitulo), o AgenteProbeListenerThread apenas escuta e responde a pedidos.

Sobre a ligações TCP estas não são de facto tratadas pelo AgenteUDP, mas sim pelo servidor mini-httpd a correr na mesma máquina. Este foi o servidor HTTP que escolhemos para testar a nossa arquitetura. Ele está responsável por receber pedidos TCP redirecionados pelo ReverseProxy e devolver a informação pedida.

2.2 MonitorUDP

O MonitorUDP também segue um funcionamento muito simples. A thread MonitorProbeRequestThread envia periodicamente, em Multicast, para a rede um *probe request* ao qual irá esperar respostas de servidores.

Esta resposta é recebida pela thread MonitorListenerThread, onde qualquer servidor ainda não conhecido é adicionado á tabela de servidores e se o servidor já for conhecido é atualiza a sua informação na tabela.

Temos também a thread MonitorCleanInactives, responsável por verificar a existência periódica (cada 20 segundos) de servidores que não respondam á demasiado tempo (inativos á mais de 30 segundos), eliminando-os da tabela.

2.3 ReverseProxy

Finalmente o ReverseProxy tem a thread ReverseProxyListener, responsável por receber pedidos TCP e posteriormente redireciona-los para um dos servidores back-end presentes na tabela de servidores, utilizando um algoritmo para seleção do melhor servidor no momento do pedido.

O algoritmo de seleção utilizado pelo ReverseProxy utiliza o RTT, pacotes perdidos e numero de conexões TCP ativas para decidir qual o melhor servidor para atender o pedido do cliente. O algoritmo será explicado em mais detalhe no próximo capitulo.

Depois de escolhido o melhor servidor o ReverseProxyListener cria 2 threads, que passam a ficar encarregues da ligação entre o Cliente e o Servidor escolhido, libertando o ReverseProxyListener para este continuar a escutar por mais pedidos.

A thread ClientTCPReceiver que fica encarregue de receber o input vindo do Cliente e sem o alterar envia para o Servidor escolhido e a thread ClientTCPSender que fica encarregue do percurso oposto, pegando nos dados retornados pelo Servidor e reenviando-os para o Cliente. Ambas as threads irão correr até que um dos lados feche a ligação.

Capítulo 3

Especificação do Protocolo PDU

3.1 Monitorização dos AgentesUDP

Para a devida monitorização dos AgentesUDP pelo MonitorUDP, foi necessário definir um Protocolo (PDU) para que toda a comunicação entre os elementos fosse possível e inequívoca.

Inicialmente o MonitorUDP envia um *probe request* a todos os membros do grupo Multicast. Esta mensagem é iniciado com um "PROBE"para indicar a sua intenção de receber uma resposta dos servidores com informações suas. Esta mensagem é enviada continuamente pelo MonitorUDP a cada 10 segundos.

O próximo passo é a resposta dos AgentesUDP a esta mensagem. Esta mensagem será iniciada por "STATUS"indicando que esta mensagem terá como conteúdo as condições atuais do AgenteUDP em causa, nomeadamente o estado do seu CPU e memória entre outros.

Este é um ciclo de monitorização do MonitorUDP e o seu PDU.

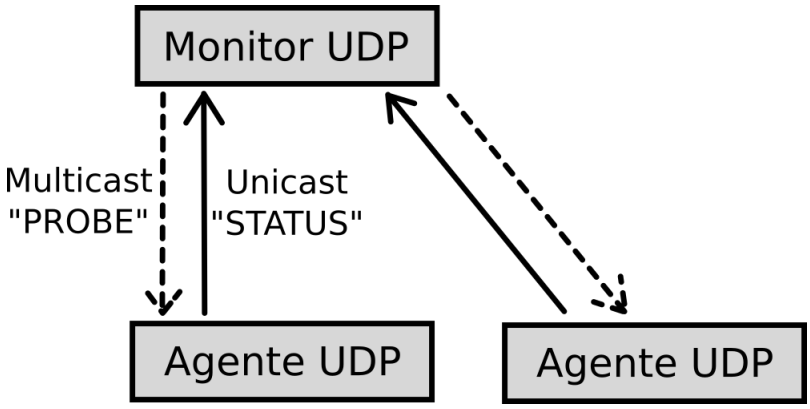

Figura 3.1: *Diagrama do PDU*

3.2 Escolha do melhor servidor

Para escolha do melhor servidor pelo ReverseProxy usa-mos 3 variáveis principais:

- RTT;
- Número conexões TCP;
- Pacotes perdidos.

Utilizamos o RTT porque ele tem uma grande importância na velocidade de resposta aos pedidos, sendo claramente a variável mais importante na escolha do melhor servidor.

Utilizamos também o número de conexões TCP porque apesar do RTT ser de elevada importância, é importante distribuir a carga por todos os servidores para melhor otimizar o uso de todos os recursos disponíveis, bem como evitar que o nosso servidor com melhor RTT seja inundado de conexões o que levaria a uma redução de performance.

Finalmente usa-mos o número de pacotes perdidos, porque um servidor pode ser muito rápido, contudo se perder demasiados pacotes na realidade será bem mais lento do que inicialmente parecia. Bem como leva a ligações instáveis, que não são de todo desejáveis.

3.3 Implementação

Aqui vamos referir algumas escolhas efetuadas relativas á implementação e funcionamento do nosso trabalho, que poderiam ser facilmente alteradas para melhor acolher diferentes ambientes de execução.

Os *probe requests* são realizados periodicamente a cada 10 segundos, aos quais se espera uma resposta dos AgentesUDP com o seu estado. A resposta com o estado do AgenteUDP deve ser enviada de imediato (e no nosso caso com um atraso aleatório entre 0 e 10ms para criar alguma dispersão nos RTT's dos agentes). Caso o AgenteUDP não responda até envio do próximo *probe request* o MonitorUDP irá considerar que o pacote foi perdido.

A thread MonitorCleanInactives, responsável por eliminar servidores inativos, corre periodicamente a cada 20 segundos e elimina servidores inativos 30 segundos.

O RTT médio de cada AgenteUDP é mantido na tabela, utilizando todos os valores registados em cada *probe request*, exceto na primeira resposta, visto esta ter como objetivo encontrar e adicionar o AgenteUDP á tabela de Servidores disponíveis. De seguida, recebida a resposta com a situação atual de um servidor, esta informação é guardada também na tabela.

Há que referir que a integridade e autenticidade da origem, de qualquer mensagem de monitorização entre o MonitorUDP e AgentesUDP, deveria ser confirmada através de uma assinatura digital simétrica, contudo não implementamos a anteriormente referida capacidade.

O nosso servidor HTTP de teste escolhido foi o mini-httpd. É um servidor HTTP fácil de implementar e que para pequena escala é perfeitamente aceitável em performance, o que permitiu um rápido teste das capacidades do nosso projeto.

Capítulo 4

Conclusão

Concluídas as tarefas propostas, damos por finalizado este trabalho prático.

Este trabalho permitiu consolidar a matéria lecionada nas aulas da unidade curricular, mas também obter um maior conhecimento acerca do funcionamento de um Servidor *Reverse Proxy*.

Como nota futura alguns aspetos podiam ser melhorados, como um algoritmo de escolha do melhor servidor mais capaz e completo, um Servidor HTTP mais elaborado e maior segurança implementando uma assinatura digital.