Relatório TP2

Carlos Silva, João Coelho e Miguel Silva

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal
e-mail: {a75107,a74859,a74601}@alunos.uminho.pt

Resumo Rabanadas

1 Introdução

1.1 Contextualização

Em muitos serviços Web, um único servidor não é suficiente para dar vazão aos pedidos dos clientes. Nesses casos, é necessário ter uma pool de N servidores de back-end, capazes de dar resposta aos pedidos. Ainda assim, o serviço terá um único ponto de entrada, comum a todos os clientes, que se trata de um servidor de front-end, com nome e endereço IP únicos e bem conhecidos. A tarefa deste servidor, habitualmente designado por *Reverse Proxy*, é atender as conexões dos clientes e desviá-las para um dos servidores de back-end disponíveis.

A escolha do servidor pode ser cega, baseada por exemplo num algoritmo de Round-Robin, que faz uma distribuição equitativa das conexões pelos N servidores de back-end, mas será esta a melhor solução?

1.2 Exposição do problema

O principal objetivo deste trabalho é desenhar e implementar um serviço simples de proxy reverso TCP, em que a escolha do servidor a usar se faz com base em parâmetros dinâmicos, como por exemplo o RTT, as perdas e número de conexões TCP do servidor. Para tal, é necessário haver monitorização dos dados do estado do servidor e da rede, redirecionando em função de uma métrica dinâmica bem definida. Pretende-se desenhar e implementar um protótipo simples desta abordagem, separando-a em dois momentos: numa primeira fase temos um protocolo sobre UDP, para criar e manter atualizada uma tabela com dados recolhidos por agentes de monitorização; depois, numa segunda fase, pretende-se implementar um servidor proxy TCP genérico, que fique à escuta na porta 80 e redirecione, automaticamente, cada conexão TCP que recebe para a porta 80 de um dos servidores de back-end disponíveis (o que aparente estar em melhores condições para o fazer).

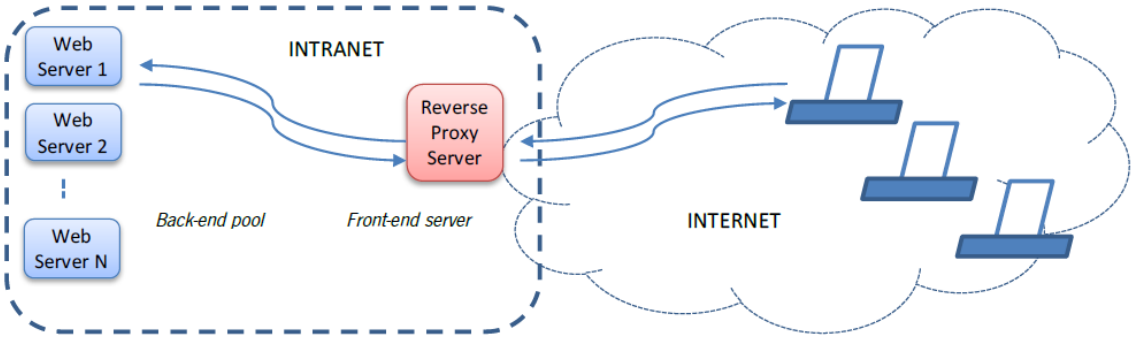
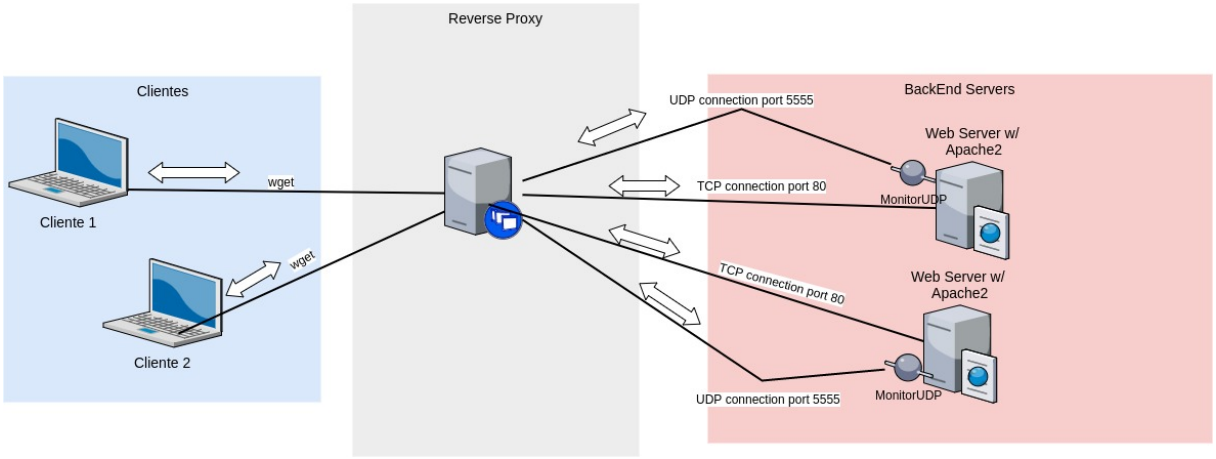


Figura 1. Esquematização da arquitetura do problema.

2 Arquitetura da solução implementada(em ambiente de teste)



3 Especificação do protocolo de monitorização

- **Primitivas de comunicação:** Sockets UDP.
- **Formato do PDU:** constituído pelo número do ACK do próximo pacote, que o servidor pretende receber, e pelo timestamp que permitirá calcular o RTT. Sobre o ACK, importa referir que é utilizado no cálculo do taxa de perdas e de duplicados. Já sobre o timestamp, nota para o cuidado especial que se teve para que contabilizasse apenas o tempo de deslocamento de um pacote entre servidores. Aquando da receção de um pacote, o monitor marca o momento. A esse valor temporal subtrai o timestamp recebido, obtendo assim o tempo da deslocação até si. Depois, no momento em que se prepara para enviar a resposta ao pedido de probing, volta a calcular o tempo atual, subtraindo o valor do tempo da deslocação. Assim, ao receber a resposta, o Reverse Proxy só tem de calcular o momento atual e subtrair o timestamp recebido na resposta, obtendo assim o RTT.
- **Interações:** Hellos de 5 em 5 segundos, com pedidos de probing smp que recebemos um estou disponivel

4 Implementação

detalhes, parâmetros, bibliotecas de funções, etc.

5 Testes e resultados

Para cenário de teste foi utilizado Apache2 em cada um dos servidores backend(dois), uma máquina a correr o ReverseProxy e ainda dois clientes a efetuar pedidos ao Reverse-Proxy. Os testes foram realizados na topologia core da unidade curricular, a máquina do ReverseProxy é o Servidor 1, as máquinas de backend são os portáteis 1 e 2 e por último os clientes que são o 1 e 2. Deixou-se a correr durante alguns segundos o ReverseProxy e os 2 servidores backend para que a tabela do ReverseProxy estabilizasse. Depois, efetuou-se o comando wget no Cliente 1 para o ReverseProxy obtendo o index.html do servidor backend com melhor score naquele instante. Verificou-se que o n o de conexões tcp(na tabela) do servidor backend naquele momento aumentou para 1, então testou-se um wget de cada cliente e o n o de conexões tcp do backend escolhido atualizou para 2. Como os pedidos dos clientes não foram efetuados exatamente ao mesmo tempo, também existiu a situação em que cada pedido foi atendido por um servidor backend diferente, ficando ambos com uma conexão tcp naquele momento.

6 Conclusões e trabalho futuro

No futuro não quero mais disto

Referências