Trabalho Prático Nº2 – Balanceamento de carga com servidor proxy invertido
Duração: 5 aulas (não consecutivas, ver calendário)

Motivação

Para muito serviços, um único servidor não é suficiente para dar vazão aos pedidos dos clientes. Nesses casos é necessário ter uma *pool* de servidores, com *N* servidores capazes de dar resposta aos pedidos. Ainda assim o serviço terá um único ponto de entrada para todos os clientes. Trata-se de um servidor de *front-end*, com um nome e um endereço IP únicos e bem conhecidos, cuja tarefa é atender as conexões dos clientes e desviá-las para um dos servidores de *back-end* disponíveis. Esse servidor designa-se normalmente por *Reverse Proxy*. Esta abordagem é comum nos serviços Web e está ilustrada na *figura 1*. 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 é possível fazer melhor, coligindo dados do estado do servidor e da rede, redirecionado em função de uma métrica dinâmica. Neste trabalho pretende-se desenhar e implementar um protótipo simples desta abordagem em duas fases: i) Recolha de informação dos servidores via UDP (*Multicast* e *Unicast*); ii) implementação de um *front-end* TCP que receba as conexões dos clientes, escolha um dos servidores disponíveis e intermedeie a conexão TCP para o servidor escolhido.

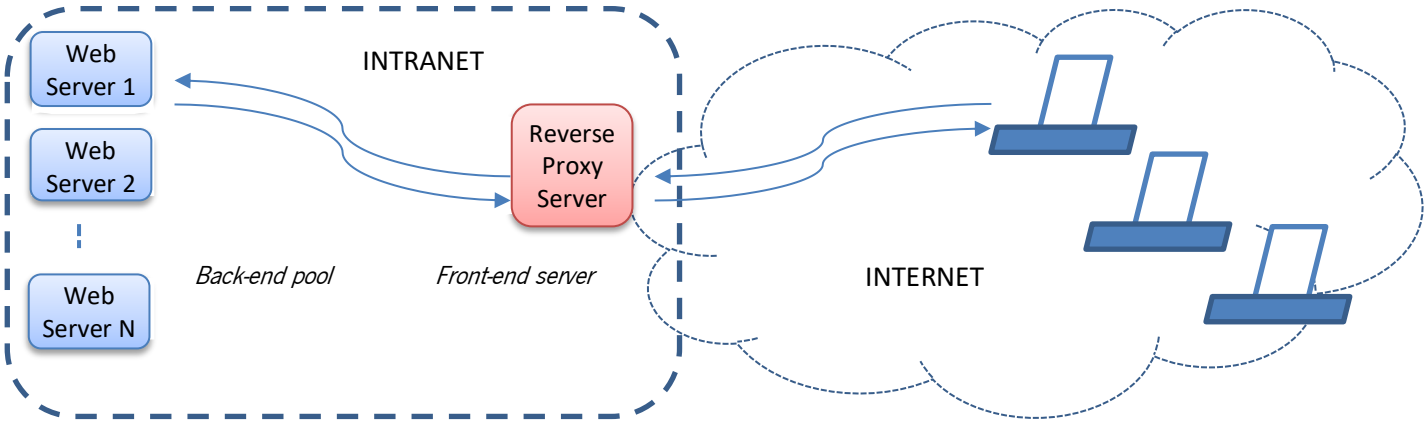


Figura 1

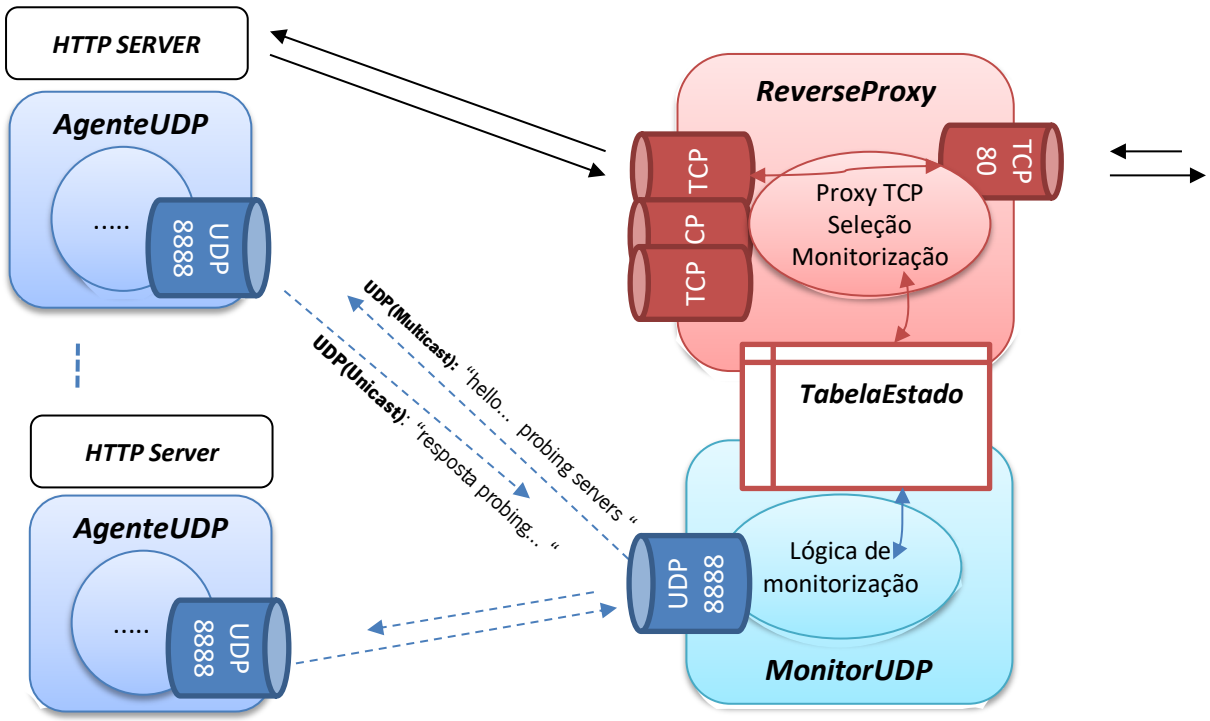
Objetivos

O objetivo deste trabalho é desenhar e implementar um serviço de balanceamento de carga com servidor proxy invertido. O serviço deve poder funcionar sem necessidade de qualquer configuração (configuração zero). A escolha do servidor a usar deve ser feita com base em métricas dinâmicas de dois tipos: 1) estado do servidor em termos de CPU e RAM; 2) estado da rede em termos de RTT e largura de banda consumida por servidor. Será necessário, em primeiro lugar, desenhar uma tabela de informação de estado por cada servidor de *back-end* que seja descoberto automaticamente. A tabela de estado deve conter a seguinte informação: i) identificação do servidor (IP e PORTA); ii) Carga do servidor: RAM e CPU; iii) RTT em milissegundos; iv) estimativa da largura de banda efetivamente usada pelo servidor. De seguida é necessário desenhar e implementar um protocolo sobre UDP capaz de descobrir automaticamente os servidores de *back-end* disponíveis e preencher os dados de identificação do servidor, carga do servidor e RTT da rede. Finalmente pretende-se implementar um servidor proxy TCP genérico, que fique à escuta na porta 80 e redirecione automaticamente cada conexão TCP na porta 80 que receber para um dos servidores disponíveis de acordo com um algoritmo de seleção bem definido. Cabe ao servidor proxy a tarefa de medir a largura de banda efetivamente usada por cada servidor.

Descrição

A *figura 2* mostra uma visão mais detalhada do sistema. Na arquitetura proposta podem identificar-se, sem perda de generalidade, 4 elementos: **1)** A tabela com informação de estado (***TabelaEstado***); **2)** Um agente UDP (***AgenteUDP***) a instalar em cada servidor; **3)** Um agente UDP para deteção e monitorização de servidores (***MonitorUDP***); e **4)** Servidor TCP a atender na porta 80 (***ReverseProxy***); O componente identificado como HTTP Server é um servidor externo (ex: Apache HTTP Server ou outro qulauqer) e não deve em caso algum ser implementado!

O servidor *ReverseProxy* atua apenas como mero intermediário. Recebe e aceita as conexões TCP dos clientes, escolhe um servidor disponível consultando a tabela de estado, e abre uma conexão TCP para o servidor escolhido. Tudo o que receber de um lado envia para o outro e vice-versa. Esta tarefa deve ser feita de forma totalmente transparente, qualquer que seja o tipo de dados (texto, imagem, som, vídeo), sem que seja feita qualquer tentativa de processamento dos dados recebidos de um ou outro lado.



Componentes:

1. ***TabelaEstado*** – uma estrutura de dados partilhada, que mantém toda a informação necessária para o algoritmo de seleção de servidor poder funcionar. Recomenda-se que contenha *i)* identificação do servidor (IP e Porta) e *ii)* carga do servidor (RAM e CPU) que lhe serão fornecidas diretamente pelo ***AgenteUDP***, na resposta a pedidos periódicos enviados pelo ***MonitorUDP***; *iii)* RTT medido diretamente pelo ***MonitorUDP***; e *iv)* estimativa da largura de banda medida pelo ***ReverseProxy*** ao enviar e receber dados;

2. ***AgenteUDP*** - um agente UDP que escuta pedidos de *probing* na porta 8888 enviados pelo ***MonitorUDP*** por *multicast* para o grupo Multicast com endereço IPv4 **239.8.8.8**; Antes de ficar à escuta em UDP na porta 8888 deve pois juntar-se ao grupo *multicast* referido); Por cada mensagem recebida, o ***AgenteUDP*** deve responder com uma mensagem UDP dirigida em *unicast* ao ***MonitorUDP***, incluindo na resposta a carga da máquina (RAM e CPU). As respostas podem ser artificialmente atrasadas, por um pequeno intervalo de tempo aleatório entre 0 e 10 ms, de modo a não serem enviadas todas ao mesmo tempo. Deve ser possível verificar a integridade e autenticação de origem das mensagens recebidas e enviadas com uma *assinatura digital simétrica*. Para tal pode-se usar, por exemplo, usando um algoritmo semelhante ao HMAC definido no RFC 2104, ou algo equivalente.

3. ***MonitorUDP*** – um agente UDP responsável pela descoberta de servidores; de *t* em *t* segundos deve enviar uma mensagem de *probing*, por UDP, dirigida ao grupo Multicast 239.8.8.8 para a porta 8888; deve escutar respostas UDP *unicast* na mesma porta e verificar a integridade de todas as mensagens recebidas antes de atualizar a ***TabelaEstado*** em conformidade.

4. ***ReverseProxy*** - um agente TCP capaz de atender pedidos na porta 80, vindos de qualquer cliente (tipicamente browsers Web); a cada pedido recebido deve consultar a ***TabelaEstado***, escolher um dos servidores disponíveis usando um algoritmo de seleção e abrir uma conexão TCP para o servidor escolhido; todos os dados recebidos do cliente serão enviados de forma totalmente transparente (sem nenhum processamento, nem nenhum pressuposto) para o servidor e vice-versa; durante a troca de dados o ***ReverseProxy*** pode ir atualizando a estatística de largura de banda usada, colocando os valores na ***TabelaEstado***; As conexões serão terminadas por iniciativa de uma das entidades em comunicação (servidor HTTP ou cliente).

Cenário de Teste

Pode-se usar como plataforma de teste o emulador CORE, embora tal não seja obrigatório. Para o cenário de teste mínimo são precisas três máquinas. Numa delas põe-se a correr o servidor ***ReverseProxy*** e o ***MonitorUDP***. Nas outras duas põe-se a correr um **AgenteUDP** e um servidor HTTP qualquer (ex: *Apache*, *mini-httpd*, *nginx*. etc). O servidor HTTP deve estar capaz de servidor uma página HTML com imagens. Pode-se assim usar um browser genérico para testar, uando um URL com o endereço IP do ***ReverseProxy*** e porta 80.

FASE 1: Implementação da TabelaEstado, AgenteUDP e MonitorUDP (3 aulas)

Etapas sugeridas para esta fase:

- Desenhar o PDU a ser enviado pelo ***MonitorUDP*** aos ***AgenteUDP***
- Desenhar o PDU de resposta a ser enviado pelos ***AgenteUDP*** ao ***MonitorUDP***
- Implementar a ***TabelaEstado***, o ***AgenteUDP*** e o ***MonitorUDP***
- Testar (ex: imprimindo a tabela de estado)

FASE 2: Implementação do ReverseProxy (2 aulas)

Etapas sugeridas para esta fase:

- Desenho e implementação do algoritmo de seleção do servidor
- Implementação do componente que intermedeia as ligações TCP de forma transparente
- Implementação de estimativa da largura de banda usada
- Teste no cenário proposto

Planeamento

O desenho e implementação deste serviço devem ser feitos ao longo das aulas práticas de CC, quer nas explicitamente dedicadas ao trabalho, como em todas as outras onde existir tempo livre. As fases de desenvolvimento são as seguintes:

FASE 1: As primeiras 3 aulas reservadas para o TP2
FASE 2: As últimas 2 aulas reservadas ao TP2

Relatório

O relatório deve ser escrito em formato de artigo com um máximo de 10 páginas (recomenda-se o uso do formato LNCS - *Lecture Notes in Computer Science*, instruções para autores em http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0). Deve descrever o essencial do desenho e implementação com a seguinte estrutura recomendada:
- Introdução
- Arquitetura da solução
- Especificação do protocolo UDP
  * formato das mensagens protocolares (PDU)
  * interações
- Implementação
  * detalhes, parâmetros, bibliotecas de funções, etc.
- Testes e resultados
- Conclusões e trabalho futuro

Regras de submissão

Cada grupo deve fazer a submissão (*upload*) do trabalho, usando a opção de "troca de ficheiros" associada ao seu grupo nos "Grupos" do Blackboard (*elearning.uminho.pt*), da seguinte forma:

- **CC-TP2-PLxx-Rel.pdf** para o relatório
- **CC-TP2-PLxx-Codigo.zip** ou **CC-TP2-PLxx-Codigo.rar** para a directoria com o código desenvolvido (devidamente documentado)

em que **xx** é o identificador de cada grupo (e.g. CC-TP2-PL21-Rel.pdf para o relatório TP2 do grupo PL21)