Trabalho Prático II

Proxy TCP reverso com monitorização proactiva

Ana Esmeralda Fernandes, Bárbara Nadine Oliveira, and Miguel Dias Miranda

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a74321,a75614,a74726}@alunos.uminho.pt

1 Introdução

No presente relatório será descrito todo o processo para a implementação e estruturação do segundo trabalho prático da unidade curricular de Comunicações por Computador.

Tendo em mente que para muitos dos serviços utilizados hoje em dia não se torna suficiente a existência de apenas um servidor para dar resposta a todos os clientes, idealizamos uma arquitetura da aplicação baseada na existência de um servidor front-end, em que a sua tarefa é direcionar todas as conexões recebidas de clientes para servidores de back-end disponíveis. Um dos principais objetivos deste projeto passa então por implementar um servidor de proxy reverso TCP, *ReverseProxy*, que fará a escolha do melhor servidor a quem redirecionar o pedido de um cliente. Numa primeira fase será implementado o processo de monitorização dos servidores de backend e, posteriormente, implementado o atendimento de clientes.

Este trabalho foi implementado na linguagem de programação Java, com recurso bibliotecas de sockets, threads e controlo de concorrência.

2 Arquitetura da solução implementada

Para a estruturação da arquitetura do projeto desenvolvido, foram seguidos os objetivos e funcionalidades necessárias de implementar, referidas no enunciado do projeto.

Por uma questão de organização, esta estrutura pode ser divida em três subsistemas principais:

- O subsistema do Cliente, que engloba os pedidos do cliente via TCP e a respetiva resposta devolvida pelo servidor Reverso;
- O subsistema do ReverseProxy, onde se implementa a lógica de monitorização dos servidores de *backend* e os métodos de atendimento de Clientes;
- Por fim, o subsistema referente ao funcionamento dos servidores de *backend*, onde se implementam os mecanismos de resposta a pedidos de Clientes e o procedimento de manifestação de atividade do servidor de *backend*, para posterior monitorização no ReverseProxy.

Toda a implementação do projeto foi realizada utilizando a linguagem java com recurso a bibliotecas de sockets (sobre TCP e UDP), Threads e controlo de concorrência, na tentativa de representar o funcionamento da rede. A seguinte imagem procura esquematizar esta arquitetura, servindo assim como base para compreender a comunicação entre classes, métodos e processos implementados, que serão abordados nas seguintes secções.

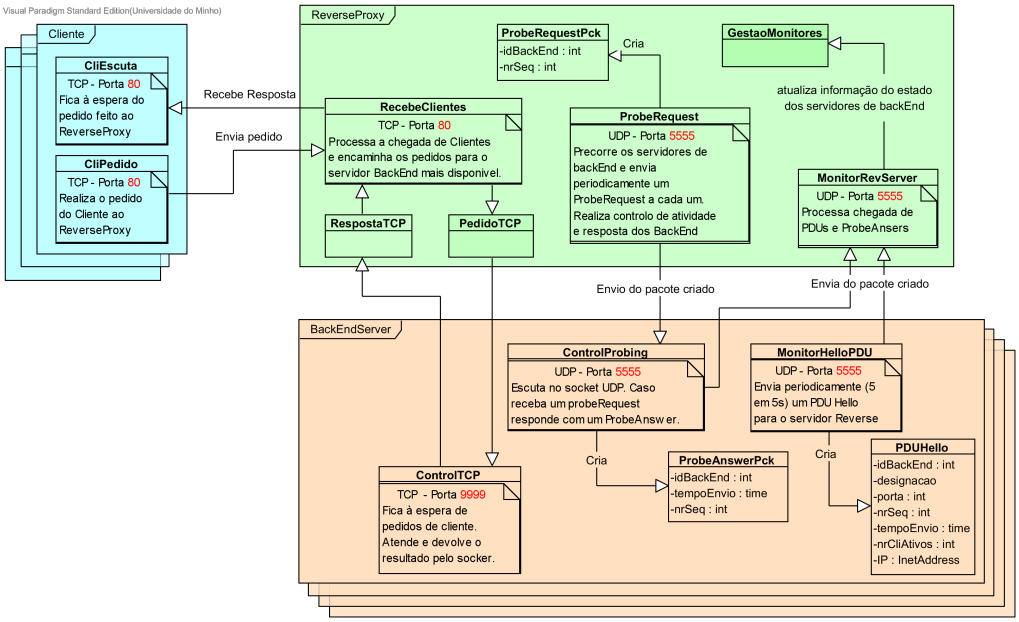

Figura 1. Esboço da arquitetura da solução implementada

3 Especificação do protocolo de monitorização

Como métodos de monitorização dos servidores de *backend*, por parte do servidor Reverso, foram implementados dois processos de controlo:

- O envio de pacotes periódicos, designados como *PDUHello*, pelos servidores de *Backend* para o servidor Reverso ao qual se ligaram. Estes pacotes não exigem a confirmação da sua receção por parte do servidor Reverso;
- A solicitação esporádica, por parte do servidor Reverso aos servidores *Backend* a si ligados, de pedidos *ProbeRequest*. Estes pedidos são numerados com um número de sequência e exigem uma resposta por parte do servidor *Backend* que os recebe. Para esta funcionalidade foram criados os pacotes de dados *ProbeRequestPck* e *ProbeAnswerPck*, respetivamente.

O conteúdo destes três pacotes de dados referidos serão especificados e abordados com detalhe nos seguintes tópicos desta secção. Para a implementação deste controlo e monitorização de servidores de *Backend* estão envolvidos os subsistemas do *ReverseProxy* e do *BackendServer*.

3.1 Primitivas de comunicação

A comunicação entre o servidor Reverso e os seus servidores de *Backend* foi realizada utilizando sockets baseados no protocolo UDP. Para a comunicação entre o servidor Reverso e os eventuais Clientes que a este solicitem pedidos, a comunicação entre as entidades é suportada pela implementação de um socket baseado no protocolo TCP.

Em qualquer um dos contextos de comunicação, é sempre realizado um processo de linearização da estrutura de dados antes de a enviar pelo socket. Por uma questão de simplificação, esta linearização/Marshalling tem como procedimento a passagem dos dados a enviar para uma string. Esta string terá os campos do pacote de dados a enviar pelo socket, separados pelo caractere espaço.

Do lado do recetor da mensagem, no processo de restauração/unmarshalling da informação vinda do socket, é feito o parsing pelo caractere espaço e, conforme o valor da

primeira posição, determina-se que tipo de pacote foi recebido. Este procedimento será abordado com mais detalhe no seguinte tópico onde se explica o formato dos pacote de dados criados.

A nível de endereçamento, e pelo facto dos sistemas estarem a operar em máquinas diferentes (simuladas na topologia CC-Topo-2017 no CORE), todos os sockets são encaminhados para o respetivo IP e porta a quem as mensagens devem ser entregues:

- No caso dos clientes, os seus eventuais pedidos serão sempre encaminhados para a porta 80, segundo um socket baseado em TCP orientado para o IP do servidor Reverso;
- Na manifestação de atividade por parte dos servidores de *Backend*, para o servidor Reverso a quem se ligam, o envio de pacotes PDUHellos e de respostas a possíveis pedidos de Probing são sempre realizados segundo um socket baseado em UDP, aberto na porta 5555 e orientado para o IP do servidor Reverso;
- Os pedidos de Probing, solicitados pelo servidor Reverso aos seus servidores de *Backend*, são novamente realizados por um socket UDP, aberto na porta 5555 e direcionado para o servidor de IP do *Backend* a quem o pedido de Probe é solicitado.
- Por fim, o reencaminhamento dos pedidos dos Clientes, por parte do servidor Reverso para um dos seus servidores de *Backend*, é feito enviando o pedido do Cliente para o socket UDP na porta 9999 e direcionando para o IP do servidor de *Backend* que irá tentar dar resposta ao pedido do cliente.

O tipo de socket utilizado, assim como a porta segundo a qual se estabelece a abertura do socket, podem ser revistas com mais algum detalhe na maqueta da arquitetura da solução, apresentada inicialmente. Quer na comunicação com clientes, como no controlo e monitorização dos servidores de *Backend*, todas as comunicações implementadas são assíncronas, não existindo nenhum processo a bloquear aguardando por uma resposta. Esta decisão permite simplificar a implementação da comunicação entre as diversas entidades e garantir que nenhum novo cliente ou servidor de *Backend* seja afetado pelo bloqueio de um dado canal de comunicação com o servidor Reverso principal.

Apesar de não ocorrer o bloqueio de processos à espera de uma resposta, no caso dos pedidos de Probing, que exigem confirmação e resposta por parte dos servidores de *Backend*, é utilizado o número de sequência dos pacotes enviados para garantir a chegada da resposta a um determinado pedido enviado.

A nível da disponibilidade e confiabilidade dos serviços, na tentativa de simular a realidade, foram alterados alguns links da topologia CORE para simular atrasos e perdas de pacotes, na comunicação entre as diferentes entidades.

3.2 Formato das mensagens protocolares

Formato das mensagem PDU:

O pacote PDU é utilizado como resposta ao protocolo de monitorização UDP.

Sempre que um servidor de *Backend* arranca, depois de uma fase inicial de configuração onde se determina o seu endereço IP e lhe é atribuído um *id* único, este servidor passa a enviar de forma regular (teoricamente de 5 em 5 segundos) um PDU Hello, apresentando assim dados sobre si e relativos à sua carga de utilização.

A seguinte imagem procura descrever os campos que existem no PDU:

O primeiro campo irá conter o valor "PDU"para que, no processo de tratamento dos dados recebidos pelo socket, seja possível saber que informações foram recebidas.

De seguida, como parâmetros estáticos e constantes temos:

- O número identificador do servidor de *Backend* que enviou o presente pacote de dados. Este *id* será único e é por ele que o servidor Reverso mapeia e gere a sua tabela de monitorização.

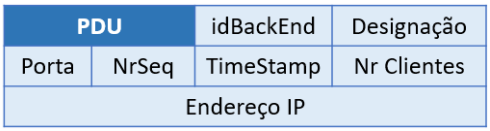

Figura 2. Campos do pacote PDU

- Designação do servidor.
- Especificação do número da porta (por omissão é a 5555, mas poderia ser outra).
- O IP da máquina onde o servidor de *Backend* está a operar e a enviar os PDUs

Estes parâmetros são úteis quando o servidor Reverso recebe pela primeira vez um PDU de um determinado servidor de *Backend*, pois usará estes campos para registar o servidor na sua tabela de monitorização.

Como parâmetros não estáticos, com principal relevo para o controlo e atualização dos dados de um servidor de *Backend* temos ainda na mensagem PDU os campos:

- O número de sequência do PDU Hello enviado.
  Apesar de não se exigir confirmação da receção de um PDU, é possível, no lado do servidor Reverso, controlar eventuais perdas na chegada destes Pacotes.
- Um timeStamp, em milissegundos, que representa a data/hora em que o pacote PDU foi gerado. Este campo permite estimar aproximadamente os RTT de receção de PDUs Hello, no lado do servidor Reverso.
  Sempre que é recebido um pacote por um socket, é tirado um timeStamp da hora de receção e comparado posteriormente com este timeStamp registado na mensagem de dados recebida.

Um exemplo do processo de linearização desta estrutura de dados, antes de a enviar pelo socket, seria o seguinte: "PDU 12 bakcend12 5555 4 TIMESTAMP 5 10.3.3.10 ".

Formato dos pacotes de Probing:

Os pacotes de Probing são utilizados no mecanismo de monitorização ativo, levado a cabo pelo servidor Reverso. Neste processo, o referido servidor envia, periodicamente, um pedido de Probe a cada um dos seus servidores de *Backend*, controlando assim a atividade e disponibilidade.

Como na solução desenvolvida, é no pacote PDU que o servidor de *Backend* se anuncia e partilha ao servidor Reverso as suas informações relevantes e mais recentes, o mecanismo de probing funciona apenas como uma forma de controlo da atividade dos servidores de *Backend*.

Por cada pacote ProbeRequest enviado a um determinado servidor de *Backend*, é exigido um respetivo pacote de resposta ProbeAnswer com o mesmo número de sequenciação. Através do controlo do número de sequências das respostas obtidas, foi estipulado que se o servidor de *Backend* não responder a cinco ProbeRequests consecutivos, então é assumido que o servidor se encontra indisponível ou offline, sendo assim removido da tabela de monitorização do servidor Reverso.

A remoção dos servidores de *Backend* da tabela de monitorização do servidor Reverso, relativa aos servidores disponíveis, é assim feita através deste controlo de respostas aos ProbeRequests. Contudo, se depois de apagado da tabela, o servidor Reverso receber um pacote PDU vindo do servidor de *Backend* removido, este é reconhecido como um novo servidor a estabelecer comunicação e é adicionado novamente à tabela de monitorização.

As seguintes imagens procuram descrever os campos que existem no pacote ProbeRequest e ProbeAnswer, respetivamente.

Para o pacote de pedido de Probe:

Figura 3. Campos do pacote Probe Request

- O primeiro campo contem o valor "ProbeRequest", para que no unmarshalling da mensagem recebida seja possível saber qual o conteúdo da mesma;
- O segundo argumento contem o identificador único do servidor de *Backend* a quem é enviada a solicitação de Probe;
- O último campo é o número de sequência do pacote enviado;

Sempre que este pacote é enviado, é marcada na entrada da tabela de monitorização, referente ao servidor de *Backend* que vai receber o pedido de Probe, um timestamp que representa a hora de envio do pedido e o valor do número de sequência do último ProbeRequest enviado e sobre o qual se espera uma resposta.

Um exemplo do processo de linearização desta estrutura de dados, antes de a enviar pelo socket, seria o seguinte: "ProbeRequest 12 5 ".

Figura 4. Campos do pacote ProbeAnswer

Sempre que um servidor *Backend* receber um pedido de Probing por parte do servidor Reverso e realizar uma resposta, será criada uma mensagem de dados ProbeAnswer com os seguintes campos:

- O primeiro campo contem o valor "ProbeAnswer", para que no unmarshalling da mensagem recebida seja possível saber qual o conteúdo a que a mesma se refere;
- O segundo argumento contem o identificador único do servidor de *Backend* que envia a resposta ao pedido de Probing;
- O número de sequencia do pacote enviado, que é igual ao número de sequência do pacote ProbeRequest que gerou esta resposta. Este campo é usado saber se o servidor de *Backend* responde regularmente a todos os Probes que lhe são solicitados;
- Um timeStamp, em milissegundos, que representa a data/hora em que o pacote ProbeAnswer foi gerado. Como quando o servidor Reverso envia um ProbeRequest, regista a hora em que o mesmo foi enviado, ao receber a resposta, com este campo é possível estimar, aproximadamente, os RTT de receção de Probes.

Um exemplo do processo de linearização desta estrutura de dados, antes de a enviar pelo socket, seria o seguinte:

"ProbeAnswer 12 5 TIMESTAMP ".

4 Interações

4.1 Envio de PDUs

Tal como solicitado, o envio de PDUs é iniciado automaticamente assim que um servidor de Backend seja ligado e configurado, com a atribuição ao mesmo de um identificador único. Este envio de PDUs é feito de forma regular, teoricamente de 5 em 5 segundos e permite ao servidor Reverso passar a conhecer um novo servidor de *Backend* que se anuncie

ou, caso já seja conhecido, atualizar na sua tabela de monitorização os parâmetros dinâmicos recebidos dentro do pacote de dados PDU. As funcionalidades associadas a este tipo de comunicação foram corretamente implementadas, sendo que a estrutura do pacote de dados PDU é abordada no capitulo anterior e a descrição da implementação algorítmica deste mecanismo será descrita no capitulo seguinte.

4.2 Envio de pedidos e Probing

Pelo facto dos campos dos pacotes de dados PDU já conterem todas as informações relevantes associadas ao servidor *Backend* que os envia, e sendo estes pacotes enviados com uma frequência constante e curta, foi determinado que no processo de probing estes parâmetros não seriam novamente inseridos nos campos do pacote de Probe.

Assim, o mecanismo de Probing é utilizado pelo servidor Reverso apenas para controlar a atividade de respostas explícitas por parte dos seus servidores de *Backend*, sendo assim possível remove-los, se um determinado número de pedidos de Probe consecutivos não receberem um pacote de resposta e confirmação.

Este limite de pedidos de probing sem resposta foi definido como sendo 5, ou seja, se não existir resposta por parte de um servidor *Backend* a cinco pedidos de Probe consecutivos, o servidor Reverso irá considera-lo como inativo e remove-lo da sua tabela de monitorização.

Se depois de removido um determinado servidor for porventura recebida uma resposta a um dos pedidos de Probing, este pacote será descartado e ignorado, pois o servidor terá fechado a sua conexão ("à escuta") com o servidor de *Backend*. Contudo, se depois de removido for recebido um pacote de dados PDU vindo do servidor *Backend* apagado, como neste pacote estão todos os dados relevantes o pacote será visto como a apresentação de um novo servidor de *Backend* e o servidor Reverso irá adiciona-lo à sua tabela de monitorização.

4.3 Monitorização da tabela de estado dos Backend

Assim que um servidor Reverso inicia a sua atividade, é também criada a sua tabela de monitorização, inicialmente sem nenhuma entrada relativa a servidores de *Backend* conhecidos.

À medida que a atividade do servidor Reverso avance e se anunciem e estabeleçam ligações com servidores de *Backend*, esta tabela passa a estar num estado dinâmico sobre constantes alterações. Por cada servidor conhecido, a entrada nesta tabela que o representa terá os seguinte campos:

- Como parâmetros estáticos:
  - O identificador (id) único do servidor *Backend*;
  - A porta sobre o qual se monitoriza o servidor;
  - O endereço de IP da máquina onde está a correr o servidor;
  - A designação pela qual é conhecido o servidor.
- Como parâmetros dinâmicos e em regular atualização:
  - O número de Cliente ativos, que representa o número de conexões TCP criadas pelo servidor de *Backend* para atender Cliente;
  - Número de pacotes PDU recebidos, perdidos e RTT médio de chegada de mensagens PDU, assim como o número de sequência do último pacote PDU recebido;
  - Número de pacotes de Probing (ProbeRequest) enviados, recebidos (ProbeAnswer), perdidos e RTT de chegada das respostas ProbeAnswer, assim como o número de sequência do último pedido de Probe enviado e sobre o qual se espera resposta.

Dos principais métodos sobre esta tabela, para dar resposta à monitorização e redirecionamento de Clientes por parte do servidor, destacam-se:

– Caso seja recebido um pacote de dados PDU:
  - Se o id do servidor de *Backend* for conhecido, são retirados os dados da mensagem PDU e atualizados os campos na entrada da tabela correspondente ao servidor que enviou o pacote;
  - Se o id do servidor de *Backend* for desconhecido, é criada uma nova entrada na tabela de monitorização para registar o novo servidor que se deu a conhecer.
– Caso seja recebido um pacote de ProbeAnswer, são atualizados na tabela os dados associados com o servidor *Backend* de quem se receberam as informações.
– Caso se determine que foram enviados cinco pedidos de Probe consecutivos sem obtenção de resposta, através de um pacote ProbeAnswer vindo do servidor *Backend*, a entrada relativa a este servidor será removida da tabela de monitorização.
– Caso seja estabelecida uma ligação com um novo Cliente, serão analisadas todas as entradas desta tabela de forma a determinar qual o "melhor"servidor disponível para atender o pedido solicitado pelo Cliente.

Esta noção de "melhor"servidor para atender um novo Cliente, é obtida através de uma métrica calculada com a atribuição de pesos aos parâmetros de controlo da atividade do servidor *Backend* considerados mais relevantes.

- O número de perdas de pacotes PDUs foi considerado o parâmetro mais importante pelo facto destes pacotes serem enviados/recebidos com maior regularidade que os pacotes de Probe. Este campo permite assim estimar a "garantia"de que na ligação entre o servidor Reverso e o de *Backend*, enviando um pacote, este será recebido;
- O RTT de chegadas de Probes e o RTT de chegada de PDUs entra também nesta métrica.
  Como os pacotes de Probing são as únicas mensagens sobre as quais se exigem respostas por parte dos servidor *Backend* que receba o pedido, este RTT é considerado mais relevante que o RTT de chegada de PDUs. Isto porque ele regista não só o tempo de RTT de chegada de um pacote de Probe como o tempo que aquela resposta ao pedido de Probe demorou a chegar, desde que o pedido de Probing foi solicitado. (medindo assim o tempo de resposta do servidor *Backend*).
  Como para atender um Cliente procuramos não só uma ligação rápida e fiável, mas também um servidor de *Backend* responsivo mal receba pedidos, esta métrica foi considerada a segunda mais relevante;
- Assumindo que os recursos físicos de hardware não sejam uma limitação, o número de clientes que o *Backend* esteja a atender foi considerado o parâmetro menos importante nesta métrica. Ainda assim foi tido em conta para evitar possíveis sobrecargas, ao encaminhar todos os pedidos de Cliente para o mesmo servidor de *Backend*.

A fórmula pode ser descrita da seguinte forma:

```
métrica = nrPDUsLost * 0.5
        + RTTChegadaProbes*0.3 + RTTChegadaPDUs*0.1
        +  nrClientesAtv * 0.1
```

5 Implementação

Nesta secção será abordada, de forma genérica, a metodologia usada para implementar em linguagem Java o funcionamento dos diferentes serviços do servidor Reverso, de *Backend* e do Cliente. Para uma melhor compreensão dos detalhes mais técnicos aqui referidos aconselha-se a revisão do esquema e descrição da arquitetura da solução implementada.

5.1 Servidor Reverso

O desenvolvimento das funcionalidades do servidor Reverso foram implementadas e estruturadas em cerca de quatro classes principais. Assim que o servidor seja iniciado:

- É criada a tabela de monitorização, sobre a forma de um Map.

Como foi definido que cada servidor de *Backend* seria identificado pelo seu id único, esta variável do tipo Map mapeia o id do *Backend* (chave) para um objeto que contem os dados relativos ao servidor *Backend* (Valor).

- São abertos os dois sockets necessários:

O socket baseado no protocolo UDP, utilizando a classe DatagramSocket, na porta 5555, para receber mensagens da dados PDU e enviar pedidos de Probing;

O socket baseado em TCP, utilizando a classe ServerSocket, na porta 80, para receber os pedidos dos Clientes.

- São iniciados três fios de execução (*Threads*), para executar o método run das classes MonitorRevServer, ProbeRequest e RecebeCliente, respetivamente.

As threads MonitorRevServer e ProbeRequest iniciadas irão utilizar mutuamente o socket UDP aberto na porta 5555. No fio de execução da classe MonitorRevServer é feito o processo de "escuta"do socket, à espera de pacotes PDU ou ProbeAnswer e, no fio de execução da classe ProbeRequest, o socket é utilizado para enviar os pedidos de ProbeRequest aos servidores *Backend*.

A thread da classe RecebeCliente terá controlo exclusivo sobre o socket TCP, pois só ela realiza o atendimento inicial dos pedidos dos Clientes. Sempre que um cliente chegar, determina-se qual o servidor de *Backend* mais "disponível", através da métrica já apresentada.

Depois de escolhido um servidor, é aberto para o seu IP e na porta 9999, um socket sobre TCP, para reencaminhar o pedido do Cliente para o *Backend* que efetivamente irá atender o Cliente.

5.2 Servidor Backend

O desenvolvimento das funcionalidades do servidor *Backend* foram implementadas de forma semelhante ao processo já descrito para o servidor Reverso. Assim que o servidor seja iniciado:

- É solicitada a atribuição do identificador numérico único que irá ser atribuído ao servidor de *backend* a iniciar;

- São abertos os dois sockets necessários:

O socket baseado no protocolo UDP, utilizando a classe DatagramSocket, na porta 5555, para enviar os pacotes PDU e responder aos pedidos de Probing do Reverso; O socket baseado em TCP, utilizando a classe ServerSocket, na porta exclusiva 9999, para receber o encaminhamento de pedidos dos Clientes e atende-los;

- São iniciados três fios de execução (*Threads*), para executar o método run das classes MonitorHelloPDU, ControlProbing e ControlTCP, respetivamente.

As threads MonitorHelloPDU e ControlProbing irão utilizar mutuamente o socket UDP aberto na porta 5555. No fio de execução da classe MonitorHelloPDU é feito o processo de envio regular de pacotes PDU pelo socket e, no fio de execução da classe ControlProbing, o socket é utilizado para receber os pedidos de Probe do servidor Reverso e lhe responder com o respectivo pacote ProbeAnswer.

A thread da classe ControlTCP terá controlo exclusivo sobre o socket TCP, utilizado-o para o atendimento efetivo dos pedidos dos Clientes e para lhes encaminhar a resposta ao pedido.

5.3 Cliente

Para o desenvolvimento do suporte a pedidos de Clientes, foi assumido que estes pedidos seriam representados por uma string.

Assim que o código do Cliente é executado, é aberto o socket TCP na porta 80 voltado para o servidor Reverso e são iniciados dois fios de execução, um para escrever o pedido no referido socket e outro para ler a eventual resposta ao pedido.

Assim que o socket é aberto, e no lado do servidor Reverso se identifica a conexão de um novo Cliente, é automaticamente determinado qual o servidor de Backend que irá servir o pedido e o conteúdo enviado pelo socket é reencaminhado para o socket aberto entre o servidor Reverso e o servidor *Backend* escolhido. Apesar de todo o código necessário ter sido implementado, apenas se conseguiu garantir e testar a comunicação entre o cliente e o servidor Reverso e a escolha do melhor servidor por parte do Reverso. A comunicação entre o Reverso de o *Backend* escolhido, assim como a devolução da resposta ao pedido por parte do *Backend* para o Reverso não se conseguiu corrigir e detetar a falha a tempo da data final do projeto.

6 Testes e Resultados

Para demonstrar o funcionamento da solução criada, foi utilizada a ferramenta CORE com a topologia *CC-Topo-2017*, correndo em diferentes nodos as várias entidades desenvolvidas.

Para estes testes, utilizaram-se como servidores de *Backend* os servidores Alter1, Alter2 e Alter3, podendo contudo ser qualquer um dos outros nodos. O servidor Reverso (principal) é por omissão o Servidor1 e os Cliente podem ser testados em qualquer um dos Nodos Portáti**l** ou Client**e**.

A nível dos servidores de *Backend*, fizeram-se ainda algumas alterações na topologia referida a nível dos links, de forma a simular perdas de pacotes e atrasos de forma aleatória. Á medida que se desenvolveu o código, criaram-se também alguns mecanismos de interação, para melhor ver a evolução do estado do sistema.

A nível dos Servidores de *Backend*:

- Assim que o servidor é colocado a funcionar, são apresentadas no seu terminal indicações quando este envia um pacote PDUHello ou quando recebe um pedido de Probing e responde ao mesmo.
- tecla 1 - Para o processo de envio de PDU Hello;
- tecla 2 - Para o envio de respostas a pedidos de Probing; (para testar a remoção no lado do Reverso);
- tecla 3 e 4 - Retomar o envio de PDU Hello e de respostas a pedidos de Probing, respetivamente.

A nível do Servidor Reverso, existe um menu de interação com as seguinte opções:

- tecla 1 - Lista os pacotes PDU Hello recebidos;
- tecla 2 - Listar pedidos de Probe enviados, assim como as respostas ao pedidos que recebe;
- tecla 3 - Listar toda a monitorização, quer associada com os PDU Hello como com os pedidos de Probing;
- tecla 4 - Para a listagem de qualquer uma das opções acima referida;
- tecla 5 - Lista o conteúdo da tabela de monitorização, que apresenta os dados relativos aos *Backend* conhecidos

Para o terminal do Cliente:

- Sempre que ele fizer um pedido, esse pedido é repetido no terminal e, quando recebida a resposta, a resposta devia também surgir no terminal. Como referido, a implementação desta parte foi realizada mas não está a funcionar.

As seguintes imagens procuram descrever a solução criada:

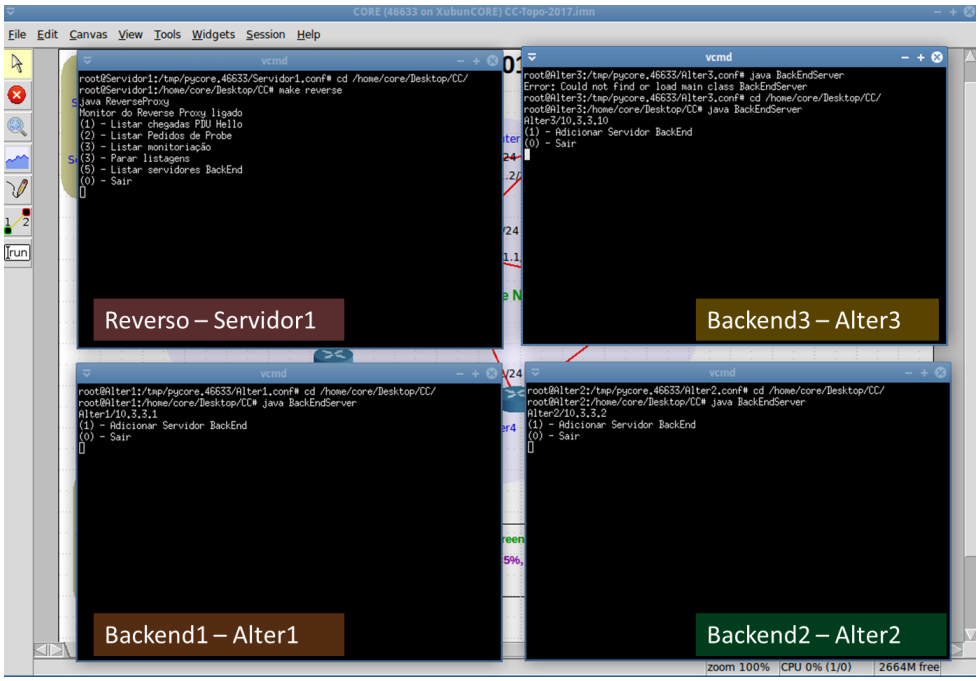

Figura 5. Estado inicial, correndo o código para o servidor Reverso e os seus *Backends*

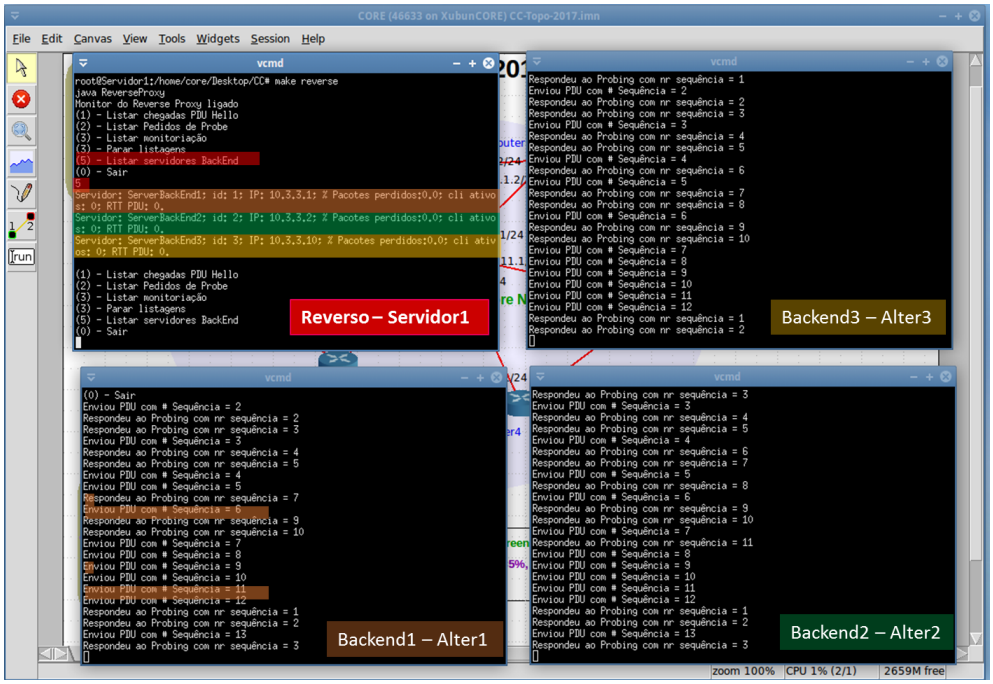

Figura 6. Estado da tabela de monitorização do servidor reverso Reverso, ao fim de alguns segundos de atividade recebendo pacotes PDU Hello dos servidores *Backend* iniciados e respostas aos pedidos de Probing solicitados

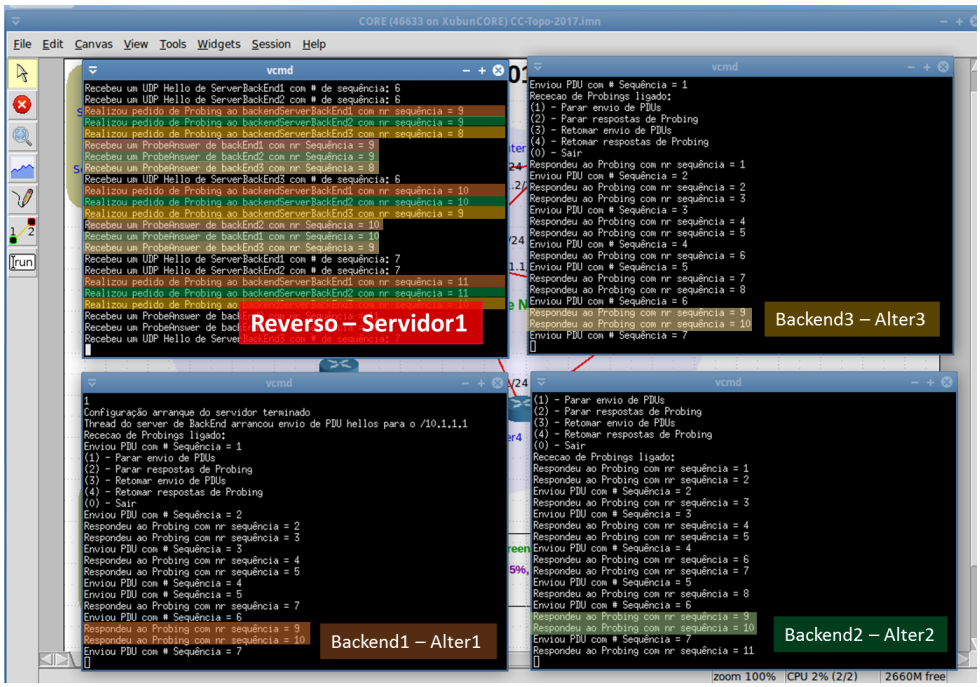

Figura 7. Exemplo da monitorização da atividade e tempos de resposta dos servidores *Backend*. Como o servidor Reverso conhece três servidores *Backend*, regularmente lhes envia pedidos de Proging numerados sobre os quais recebe respostas

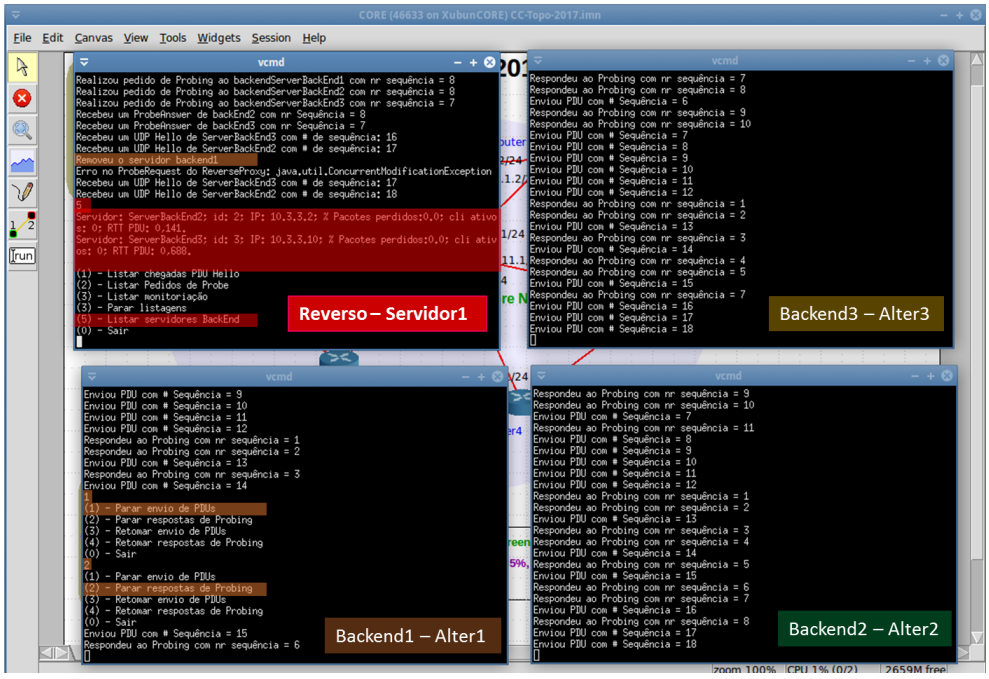

Figura 8. Parando o envio de pacotes PDU e respostas a pedidos de Probing no servidor *Backend*, ao fim de 5 pedidos de Probing consecutivos sem resposta no lado do servidor Reverso, este remove da sua tabela de monitorização a entrada associada com o servidor considerado inativo

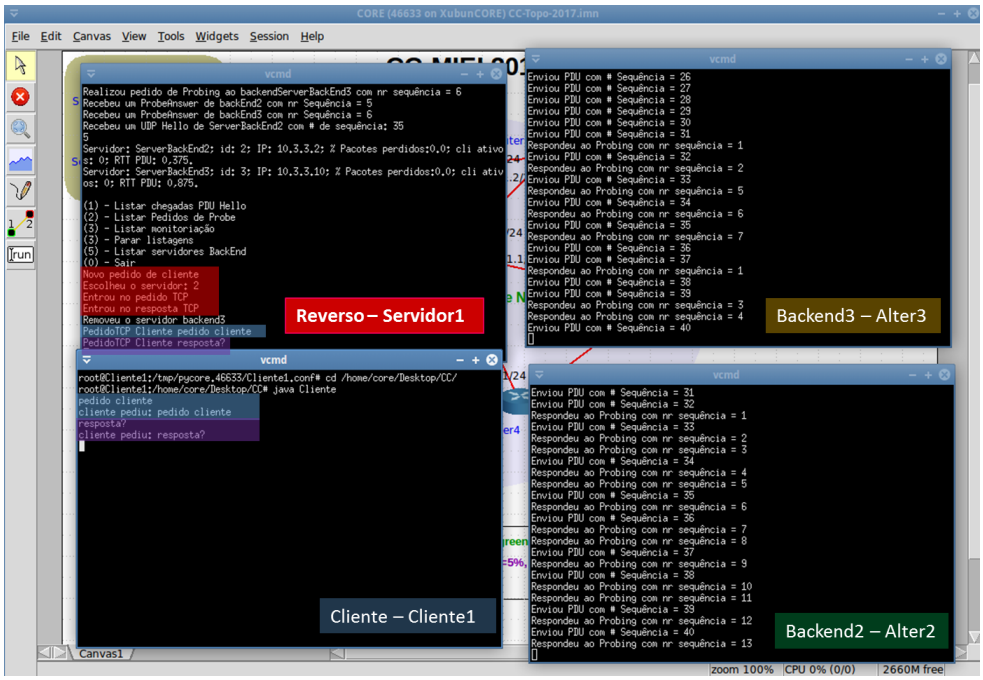

Figura 9. Exemplo da conceção de um novo Cliente ao servidor Reverso. O servidor reconhece o novo cliente, indica qual o servidor de *Backend* que o irá atender, mas a restante comunicação entre as entidades está implementada com falhas e não se desenrola, tal como referido

7 Conclusão

Com a resolução deste trabalho ficamos a compreender todo o conceito e implementação de um serviço proxy TCP reverso, cujo objetivo principal é conseguir dar resposta a pedidos efetuados por clientes.

De forma geral, para obter tal objetivo, o servidor proxy vai funcionar da seguinte maneira: todos os pedidos realizados pelos clientes entram por um único ponto (servidor front-end) e são reencaminhados para um dos vários servidores disponíveis (servidores *Backend*) que vão processar a resposta aos pedidos.

Durante a realização do projeto deparamos-nos com vários problemas a nível dos testes na plataforma CORE e implementação, tendo conseguido ultrapassar grande parte das dificuldades. Contudo, não conseguimos arranjar resolução para alguns destes problemas, mais concretamente na implementação das respostas aos pedidos dos clientes, por parte dos servidores *Backend*.

Apesar de não conseguir realizar com sucesso este requisito, que faria parte de um trabalho futuro, todas as funcionalidades de monitorização, escolha do servidor de *Backend* para atender o cliente e envio do pedido do cliente para o Reverso, estão corretamente implementadas.

Referimos ainda que o limite de página foi excedido exclusivamente para conseguir incluir imagens relevantes para demonstrar o funcionamento do sistema, esperando assim não ser prejudicados em qualquer sentido associado com este aspeto estrutural do relatório.