1 Teste automático remoto de servidores ES

O presente guia contém os procedimentos para execução remota de aplicações *user* na maquina 'tejo' para teste dos servidores ES desenvolvidos pelos alunos.

1.1 Introdução

No âmbito do projecto de comunicação usando a interface sockets oferece-se a possibilidade de testar as aplicações desenvolvidas pelos alunos em comunicação com aplicações que cumpram o protocolo especificado.
Tal é a finalidade do servidor concorrente ES em execução no porto 58000 da máquina 'tejo', o qual permite testar as aplicações *user* desenvolvidas pelos alunos.
Com a finalidade inversa de testar as aplicações ES desenvolvidas pelos alunos, existe em execução na máquina 'tejo' um servidor TCP concorrente (no porto 59000) que inicia localmente instâncias da aplicação *user* em resposta a acessos TCP por **netcat**, originados pelos alunos. Num acesso típico, os alunos especificam os parâmetros de execução remota da aplicação *user* na máquina 'tejo'.

Assim é possível executar em simultâneo na máquina 'tejo' instâncias da aplicação *user* que enviam mensagens a diferentes servidores ES alvo instalados na rede pública ou na rede do Técnico. Como por exemplo nas máquinas *sigma*.

Ambas as modalidades de teste acima referidas permitem aos alunos suportar tanto o desenvolvimento das suas aplicações como a componente de autoavaliação.

1.2 Instruções de activação dos testes

As aplicações *user* executadas remotamente na máquina 'tejo' vão ler as sequências de comandos a executar de *scripts* predefinidos, escolhidos pelos alunos para testar os seus servidores ES, mantendo separação de dados entre si não havendo assim qualquer interferência entre aplicações *user* que possam estar em execução simultânea.

A figura que se segue ilustra uma sequência de teste típica sendo a mesma executada automaticamente e sem interrupção em três fases.

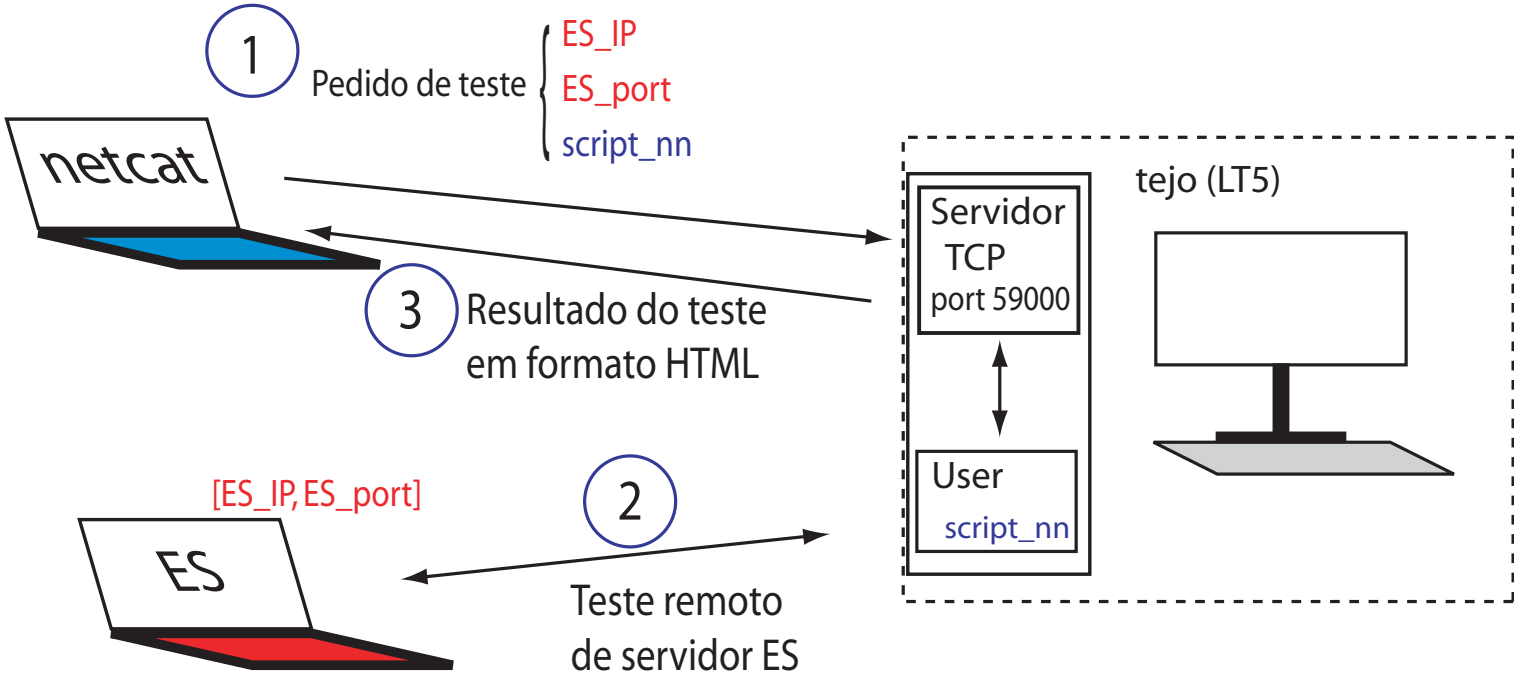

Figure 1: Teste remoto de servidores ES

A primeira fase do teste consiste na activação da aplicação *user* no 'tejo' - assinalada na figura com ①. Nessa activação são especificados o endereço IP (ES_IP) e o porto (ES_port) do servidor ES a testar. São também especificados o número do *script* (script_nn) que contém os comandos a executar pela aplicação *user*, e o nome do ficheiro de relatório a ser enviado para o computador que activa o teste.
No seguimento da activação, a aplicação *user* executa o *script* escolhido em acesso ao servidor ES dos alunos - segunda fase assinalada na

figura com ②.
A terceira fase, ③, consiste no envio pelo sistema de testes no tejo do relatório para o computador que activou o teste. Esse relatório segue em formato HTML, para ser visualizado usando um *browser* de internet.

Para executar um teste subordinado a um dado *script*, os alunos têm de compor e executar a seguinte linha de comando no terminal Linux:

echo "target_IP target_port script" | nc tejo.tecnico.ulisboa.pt 59000 > report.html

O texto entre "" constitui a **mensagem de comando** na qual, 'target_IP' é o endereço IP da máquina onde executam o seu servidor AS, 'target_port' é o porto da sua aplicação ES na referida máquina e 'script' é um inteiro com um ou dois dígitos que indica o número do *script* de teste que querem correr (ver *scripts* publicados na página da disciplina).
A linha acima descrita não deve ser segmentada. Isto é: Não deve ser executado primeiramente o comando **nc** e depois inserida a mensagem de comando já dentro do ambiente **nc**. Porque entre a aceitação da ligação TCP no servidor e a leitura da mensagem de comando existe uma reduzida tolerância de tempo para evitar o bloqueio de recursos no servidor.

Na mensagem com o formato apresentado acima, o servidor no tejo espera encontrar o caracter '\n' logo a seguir ao número do *script*, o qual é inserido pelo próprio comando 'echo' nas instalações *Linux* que serviram para elaborar este guia. Caso o comando 'echo' não insira o '\n' pode ser usado por exemplo o seguinte procedimento alternativo:

printf "target_IP target_port script\n" | nc tejo.tecnico.ulisboa.pt 59000 > report.html

Os espaços entre campos da mensagem de comando podem ser em qualquer número desde que o comprimento total da mensagem (incluindo '\n') não exceda os 25 bytes. Não serão aceites pedidos que definam como alvo os endereços IP público ou privado do tejo, ou começados por 127.

No final da execução, o servidor envia um ficheiro HTML com o resultado da referida execução obtido pela aplicação *user* cuja execução se solicitou. Esse ficheiro pode ser convertido em formato pdf para ser entregue no contexto da auto-avaliação do projecto.

O servidor ES alvo a testar pode ser executado no *sigma* cujo endereço IPv4 pode ser obtido executando o comando *hostname -i*. Note-se que o *sigma* pode ser endereçado por vários IPs. Caso queiram testar o servidor ES nos seus domicílios, os alunos devem proceder à configuração de *port forwarding* nos seus *routers* de acesso à rede.

Como exemplo, considere-se o servidor ES a testar no porto 58050 do *sigma* com IP=193.136.128.103 para execução do *script* número 6. A linha a executar numa máquina com Linux (podendo ser ela o próprio *sigma*) será:

echo "193.136.128.103 58050 6" | nc tejo.tecnico.ulisboa.pt 59000 > report.html

Ficando o relatório da execução guardado no ficheiro report.html na máquina na qual se executa o comando **nc**.

1.3 Teste de servidores ES usando *scripts* de comandos

Os *scripts* de comando da aplicação *user* no tejo invocada remotamente, estão em regra numerados de acordo com o grau de complexidade dos testes que produzem. No entanto, os alunos podem solicitar a execução dos *scripts* pela ordem que entenderem conveniente para testarem aspectos particulares do funcionamento dos seus servidores.

Aconselha-se o estudo pormenorizado dos *scripts* antes das respectivas execuções para que sejam percebidas de forma adequada as funcionalidades que são testadas pelos mesmos e assim se possam interpretar convenientemente os resultados dos relatórios. Os *scripts* encontram-se devidamente comentados no sentido de facilitar a sua interpretação.
Apresenta-se a seguir uma descrição sumária dos *scripts* pela ordem natural de numeração dos mesmos sugerindo-se o teste do servidor pela mesma ordem de numeração dos *scripts*.

1.4 Descrição dos *scripts* de teste

Pressupõe-se a execução sequencial dos *scripts* descritos a seguir, devendo a base de dados encontrar-se vazia de utilizadores e de eventos antes da execução do *script_01*.
Pressupõe-se também que no ES a testar se desactiva temporariamente a validação da data de ocorrência dos eventos. Para que seja possível criar eventos com datas anteriores à data de realização dos testes.

Os primeiros três *scripts* testam funcionalidades básicas do servidor ES exclusivamente através de comunicações UDP. Estas funcionalidades estão relacionadas com o registo e identificação dos utilizadores e validação de *passwords* na base de dados do ES.
O quarto *script* testa a funcionalidade de alteração de password que já é suportada em TCP.
Os *scripts* 5, 6, 7 e 8 testam a transferência de ficheiros com os comandos **create** (*upload*) e **show**, (*download*) para três utilizadores diferentes. O *script* 8 compara os ficheiros recebidos com os originais enviados com os comandos **create**.

Com os *scripts* numerados de 1 a 8 executados em sequência testam-se assim funcionalidades básicas do servidor baseadas em UDP e TCP.

Com os *scripts* numerados de 9 a 12 apresentados a seguir testam-se funcionalidades de gestão e consistência da base de dados do servidor. Para executar esta sequência de testes deve começar-se por inicializar de novo a base de dados removendo todos os utilizadores já registados e todos os eventos criados.

O *script* 9 preenche a base de dados criando 7 eventos de um utilizador. Dois dos eventos estão no passado devendo aparecer assinalados como tal nas listagens onde figuram.

O *script* 10 começa por registar quatro novos utilizadores. De seguida, cada um desses novos utilizadores reserva lugares nos primeiros quatro eventos criados pelo *script_09*, pedido sucessivamente listagens de reservas e finalmente detalhes sobre os quatro eventos em causa. A finalizar, o *script* 10 faz entrar em sessão o utilizador que criou os eventos para um pedido de listagem *myevents*.

O *script* 11 testa a acumulação de reservas nos eventos 001 a 004. Ficando no final do teste os eventos 003 e 004 no estado de esgotados. No final do *script*, o utilizador que criou os eventos fecha os eventos 003, 004 e 007 - este último um evento para o qual não foram efectuadas reservas.
O *script* 12 testa de novo a acumulação de reservas nos eventos 001 a 004 e 007. Ficando no final do teste os eventos 001 e 002 esgotados e devendo as reservas nos eventos 003, 004 e 007 ser rejeitadas.

1.5 Testes de concorrência

Os testes propostos nesta secção estão orientados para avaliar as capacidades de concorrência do servidor. Eles baseiam-se em dois conjuntos de quatro *scripts* cada. O primeiro conjunto, constituído pelos *scripts* numerados de 21 a 24, idênticos, serve para testar a concorrência do servidor com quatro aplicações *user* em acesso simultâneo para criação de 10 eventos cada um.
O *script_25* serve de base à constituição no servidor de uma base de dados para testar concorrência na descarga de ficheiros com o comando **show**. A concorrência será testada pelo segundo conjunto de *scripts* numerados de 26 a 29.

Para testar a concorrência em criação de eventos, deve garantir-se em primeiro lugar que a base de dados do ES está vazia, e que tem quatro utilizadores com os seguintes pares UID/password registados: 111111/aaaaaaaa, 222222/bbbbbbbb, 333333/cccccccc e 444444/dddddddd.
De seguida abrem-se quatro terminais e neles executam-se os comandos:

echo “TargetIP TargetPort 21” | nc tejo.ist.utl.pt 59000 > report1.html

echo “TargetIP TargetPort 22” | nc tejo.ist.utl.pt 59000 > report2.html

echo “TargetIP TargetPort 23” | nc tejo.ist.utl.pt 59000 > report3.html

echo “TargetIP TargetPort 24” | nc tejo.ist.utl.pt 59000 > report4.html

As aplicações *user* no ‘tejo’ “percebem” pela numeração dos *scripts* envolvidos que devem arrancar em simultâneo. Portanto, só ao quarto pedido de execução é que todas arrancam. Recomenda-se contudo rapidez entre as primeira e quarta execuções por forma a não ocupar recursos por tempo excessivo.

Os resultados dos quatro testes encontram-se nos relatórios produzidos os quais devem ser analisados pois contêm a informação relevante para aferir das capacidades de concorrência do servidor. Estes quatro testes simultâneos podem servir para aferir a capacidade do servidor para resolver colisões de acesso à base de dados do ES em operações de escrita.

Para testar a concorrência do servidor ES em operações de descarga de ficheiros, deve a base de dados encontrar-se limpa e com o utilizador UID/password==111111/aaaaaaaa registado.

A execução do *script_25* serve para povoar a base de dados com os mesmos 7 eventos do *script_09*.

Após a execução deste *script*, podem ser executados em simultâneo os restantes quatro numerados de 26 a 29 tal como foram executados os *scripts* numerados de 21 a 24, *mutatis mutandis*. Os *scripts* numerados de 26 a 29 são parecidos entre si embora com diferentes ordenações. Após a descarga de cada ficheiro, o mesmo é comparado com o original (comandos RCOMP) para aferir do bom funcionamento das operações de *upload* e *download*.

Recomenda-se compartimentar os testes rigorosamente tal como exposto acima. O primeiro teste de concorrência deve referir sempre todos os números de *script* 21 a 24 e apenas estes. Do mesmo modo se deve proceder relativamente ao segundo teste de concorrência. Proceder de outra forma conduz a testes que ocupam ou bloqueiam de forma improdutiva os recursos do ‘tejo’.