UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES

10257750 – IVANLIS NASCIMENTO DA COSTA

RELATÓRIO EP 1 – REDES DE COMPUTADORES

 # DistRunner

O programa serve para rodar um mesmo commando em diferentes maquinas ao mesmo tempo.

Como Funciona

O programa servidor fica ouvindo na(s) maquina(s) alvo. O cli tenta abrir uma conexão com cada um dos servidores configurados no arquivo de configuração `hosts.conf`.

Uma vez que a conexão é feita, o cliente e o servidor conversam utilizando um protocolo para o cliente informar algumas informações como possíveis opções e o comando que o servidor vai ter que rodar.

O Protocolo

Uma mensagem no protocolo consiste em um token (3 bytes) informando o tipo da mensagem, um inteiro (4 bytes) informando o tamanho N da mensagem, e N bytes contendo o conteúdo da mensagem.

A conexão começa com quaisquer numero de mensagens do cliente com o token `OPT` informando as opções que o cliente vai querer ativar no servidor. Atualmente tem apenas uma opção (a de conectar o `stdin` do processo sendo rodado com o cliente para esse poder enviar inputs através do socket). Mais uma opção foi pensada, mas não conseguiu ser implementada devido ao tempo: conectar o `stderr` com o socket para o cliente poder receber mensagens de erro que possam ter sido enviadas pelo programa sendo rodado.

Após as opções, o cliente manda uma mensagem com o token `CMD` informando ao servidor qual comando rodar.

Depois de o cliente informar o comando, o servidor roda ele. A partir desse ponto, qualquer uma das duas partes podem iniciar um envio de mensagem. O cliente pode enviar uma mensagem com o token `INP` para dar um input ao comando, e o servidor pode enviar ao cliente uma mensagem com o token `OUT` com o output do programa. Quando qualquer uma das duas partes receber um EOF (tanto do socket quanto do stdin ou do programa sendo rodado), ele termina a conexão com o outro programa, fechando o socket.

O Código

As funções que implementam o baixo nível do protocolo (a comunicação entre o cliente e servidores) estão localizada no arquivo `comm.py`. Essas são utilizadas por ambos os programas.

Os códigos do cliente e do servidor ambos começam na função `main` (localizada no final dos arquivos).

Ambos os códigos tem um loop no `main` que gera uma série de threads.

Cada thread no programa o cliente (`cli.py` de command line interface) cuida de um servidor da lista de servidores no `hosts.conf`. Essa thread cuida de identificar e ler inputs vindos do stdin e outputs vindos do socket. Os inputs são enviados ao socket que essa thread controla, já o output é impresso no terminal e salvo em memória.

Cada thread no servidor cuida de uma nova conexão com um novo cliente. E possível haver vários clientes se conectando em um mesmo servidor. Essas threads cuidam de identificar e ler inputs vindos do socket e outputs vindos do programa.

Como Roda

A configuração mais básica segue assim:

No seu computador, rode o servidor (com a configuração padrão, ele vai estar escutando na porta `6789`):

```
bash
python3 ./server.py
```

Agora configure o arquivo `hosts.conf` da seguinte forma (não escolher nenhuma porta é a mesma coisa que escolher a porta padrão):

```
localhost
```

Por ultimo, rode o cliente:

```
bash
python3 ./cli.py whoami
```

Para rodar em mais maquinas ao mesmo tempo, rode o servidor nessas maquinas e configure essas no arquivo `hosts.conf`.

Para ver mais opções, rode qualquer um dos programas com `-h` ou `--help`.

O Que Não Funciona

* Quando rodar um programa que requer input (como `bash`), mas não rodar o cliente com a flag `-t` o programa simplesmente não faz nada, é preciso terminar o programa com Ctrl + C.
* O programa não consegue interpretar argumentos com espaço. Por exemplo: Tentando rodar o programa `bash -c 'Ola mundo'` normalmente rodaria o programa com esses argumentos [`bash`, `-c`, `Ola mundo`], mas em vez disso roda [`bash`, `-c`, `Ola`, `mundo`]. Isso se deve a mensagem com o token `CMD` que, ao enviar o comando para o servidor, separa os argumentos com espaço.
* O output de programas que andam com o cursor e enviam comandos especiais ao terminal não funcionam muito bem. Ex.: `top` e `vi`.