Trabalho Prático Nº2 – FolderFastSync: Sincronização rápida de pastas em ambientes *serverless*
Duração: 6 aulas (não consecutivas, ver planeamento das aulas práticas no e-learning)

Motivação

A maioria dos utilizadores usa no dia a dia um ou mais serviços de armazenamento remoto na nuvem. Quer por razões de segurança (cópia de segurança), quer por facilidade em ter os seus ficheiros disponíveis em mais que um computador, devidamente sincronizados e atualizados. Quando se trabalha em equipa, a necessidade de usar pastas partilhadas é também grande, sendo um dos serviços mais básicos e indispensáveis nos projetos e no trabalho em equipa. Embora existam já muitas e boas soluções, este é o nosso pretexto para desenvolvermos um novo serviço de sincronização de pastas espontâneo e ultra-rápido, que se baseie quase exclusivamente em protocolos de transporte não orientado à conexão, como por exemplo o UDP.

Objetivos

O objetivo deste trabalho é implementar uma aplicação de sincronização rápida de pastas sem necessitar de servidores nem de conetividade Internet, designada por *FolderFastSync* (**FFSync**). Ao executar a aplicação, será necessário indicar como parâmetros, quer a pasta a sincronizar quer o sistema parceiro (ou parceiros!) com quem se vai sincronizar. Corre depois em permanência dois protocolos: um de monitorização simples em HTTP sobre TCP e outro, a desenvolver de raiz para a sincronização de ficheiros sobre UDP. O cenário de uso mais simples, com apenas dois sistemas, está ilustrado na figura 1.

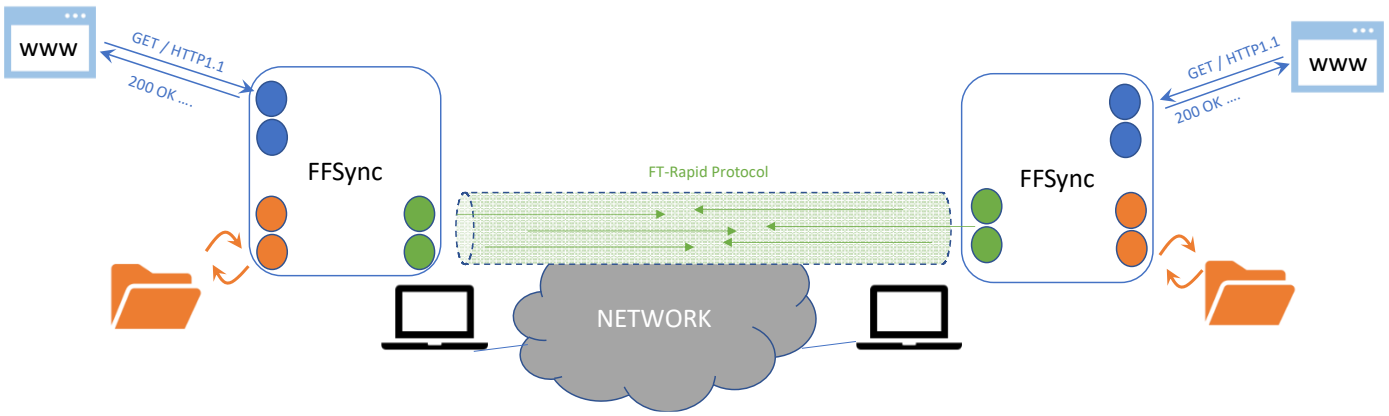

Figura 1: Esquema geral de funcionamento

Descrição

Cada nó que pretende sincronizar uma pasta com outros nós deverá colocar em execução uma instância da aplicação FolderFastSync (**FFSync**). Basta para tal executar na linha de comando a aplicação, indicando os parâmetros considerados fundamentais para o seu funcionamento. Por exemplo, será útil indicar a pasta a sincronizar (<folder>) e o endereço do parceiro com quem se sincronizar (<peer>). Um exemplo de execução seria **$ FFSync pasta1 10.1.1.1** que trataria de manter sincronizada a pasta designada por pasta1 com o sistema cujo endereço IP é 10.1.1.1

Depois de entrar em execução, a primeira tarefa será validar os parâmetros e verificar se há condições de funcionamento. Nomeadamente se possui acesso à pasta e à rede. Depois o objetivo é começar de imediato a atender pedidos em TCP na porta 80, e em simultâneo, em UDP também na porta 80. A porta 80, quer em TCP quer em UDP, é uma porta reservada, para a qual pode ser necessário ter privilégios de super user (root), mas as portas sugeridas podem ser trocadas por outras para maior comodidade e facilidade de testes como por exemplo 8888.

O atendimento em TCP serve apenas para receber pedidos muito simples HTTP, de apenas um url base /, para verificar o estado da aplicação. Deve ser possível abrir um browser qualquer e escrever http://10.1.1.1/ e obter de volta uma página simples e básica com informações internas de funcionamento da aplicação. Mas será sobre UDP que toda a sincronização de pastas deve ocorrer. Espera-se que cada grupo desenhe convenientemente um protocolo adequado para a tarefa, designado por protocolo **FT-Rapid**. Esta é a parte mais importante deste trabalho: desenhar como deve ser o protocolo e implementar um protótipo funcional! Devem ser definidos os formatos dos pacotes a serem trocados, a semântica associada e diagramas de sequência que mostrem como o protocolo opera. A figura 2 contém um esquema detalhado com identificação dos componentes.

O protocolo deverá em primeiro lugar trocar a lista de ficheiros. Depois cada instância fica responsável por pedir ao outro lado os ficheiros que lhe fazem falta e enviar de volta os que lhe pedirem. Deve também ser possível enviar múltiplos ficheiros ao mesmo tempo.

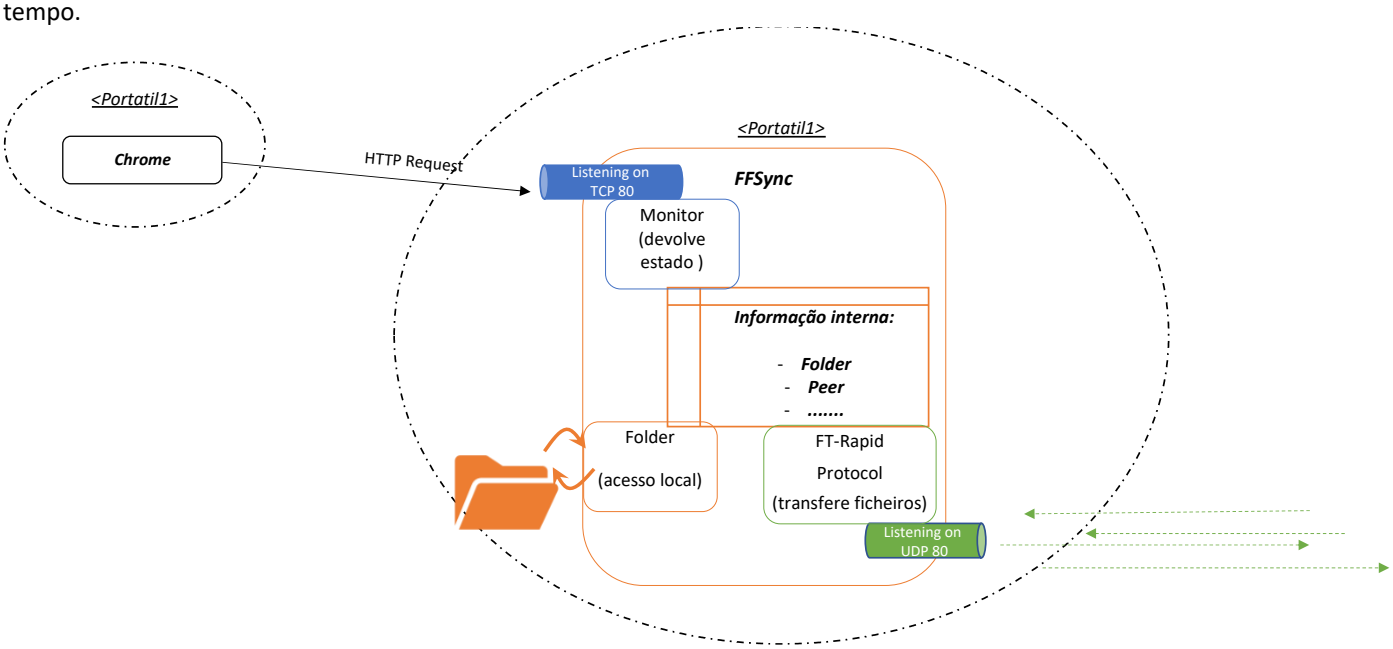

Figura 2: Visão detalhada

Pode-se usar como plataforma de teste o emulador CORE e a topologia CC-Topo-2022.imn (embora tal não seja obrigatório). Para o cenário de teste mínimo é necessário arrancar duas aplicações FFSync em nós diferentes:

```
Servidor1>  FFSync pasta1 10.3.3.1
...
Orca>  FFSync sharedfolder 10.2.2.1
...
Portatil1> wget http://10.3.3.1/
Portatil1> wget http://10.2.2.1/
```

Desenvolvimento

Requisitos obrigatórios:

- O sistema deve conseguir responder a pedidos HTTP GET devolvendo, no mínimo, o seu estado de funcionamento
- O sistema deve conseguir atender múltiplos pedidos em simultâneo nos dois sentidos (não é suposto enviar apenas um ficheiro de cada vez de forma sequencial, que pode ser ineficiente)
- O sistema deve ser eficaz (ou seja, deve conseguir cumprir corretamente a função de sincronização). O sistema diz-se eficaz se no final de uma operação de sincronização entre dois sistemas, as pastas de um e de outro forem exatamente iguais (pode-se comprovar por exemplo recorrendo à posteriori com o comando *diff*)
- O sistema de ter um sistema de registo (*logs*) que vá indicando as operações que estão a ser executadas. A quantidade de informação a ser exibida deve ser definida por cada grupo, podendo haver diferentes níveis de *logging*
- O sistema deve definir e obter duas métricas de eficiência: tempo de transferência e débito final em bits por segundo. Os valores devem ser registados no *log*.
- Segurança: o sistema deve ter implementado um mecanismo simples autenticação mútua (a definir, ex: uso de um segredo partilhado)

Requisitos opcionais:

- (Opcional) O sistema pode ter também uma interface CLI de linha de comando interativo, com algumas opções, semelhante à interface Web ou complementar.
- (Opcional) O acesso HTTP na porta 80 pode oferecer um conjunto adicional de funcionalidades, e não apenas o estado do funcionamento. Exemplos: ver lista de ficheiros na pasta, escolher ficheiros a sincronizar, registo das últimas transferências
- (Opcional) Requisitos extra de segurança: integridade e/ou confidencialidade dos pacotes trocados
- (Opcional) Sincronização entre mais que dois (ex: FFSync pasta1 10.3.3.1 10.3.3.2 10.3.3.3 ...)
- (Opcional) Sincronização de pastas e subpastas

Será interessante desenvolver um algoritmo de envio que permite enviar N ficheiros ao mesmo tempo de acordo com algum critério ou estratégia relevante. Uma sugestão, por exemplo, seria tentar dar mais largura de banda aos ficheiros maiores e menos aos mais pequenos, de modo que cheguem quase todos aos mesmo tempo. Para que se possam transferir vários ficheiros em simultâneo, nos dois sentidos (*full-duplex*) será necessário que cada pacote identifique sempre o ficheiro e a posição dentro do ficheiro a que pertence, multiplexando as transferências.

Será conveniente construir o protótipo de forma incremental, garantindo um conjunto básico de funcionalidades em cada etapa. O objetivo é ter sempre pronto um protótipo para demonstrar, caso o tempo termine. As etapas devem ser discutidas no planeamento no grupo e/ou com o docente.

Um conjunto de metas que podem ser definidas (estas podem não estar corretas, nem ser as mais adequadas são apenas exemplos):

1. Definir linguagem de programação a usar e fazer pequenos exemplos básicos de *sockets* TCP e UDP, com envio e receção em simultâneo de mensagens HELLO (só para testar conhecimentos mínimos de uso de *sockets*)
2. *FFSync* aceita parâmetros e interpreta-os corretamente, registando em *log* quais os parâmetros de funcionamento.
3. *FFSync* consegue obter uma lista dos ficheiros da pasta a sincronizar e listá-los no "*log*" ou na saída normal.
4. *FFSync* consegue trocar mensagens protocolares simples com o seu parceiro remoto (qualquer mensagem)
5. Definir quais as mensagens a trocar para obter a lista de ficheiros do sistema remoto (sintaxe, semântica e diagrama temporal)
6. *FFsync* implementa fase de obtenção da lista de ficheiros remoto e quais precisam ser pedidos ou enviados
7. Definir protocolo de transferência de ficheiro, aos pedaços, com garantia de entrega ordenada
8. *FFsync* implementa transferência de pelo menos um ficheiro num sentido. Validar.
9. *FFSync* atende pedidos HTTP na porta 80 dizendo o seu estado...
10. .... etc... (continuar)

Cenários de Teste

Para facilitar o teste e avaliação da solução estão disponíveis três pastas com conteúdos variados:

- **tp2-folder1**: pasta com apenas um ficheiro de texto de 232KB para ser sincronizado com o sistema de destino
  O zip da pasta pode ser obtido via **wget http://marco.uminho.pt/disciplinas/CC-LEI/tp2-folder1.zip**
- **tp2-folder2**: pasta com 3 ficheiros de diferentes tipos, que ocupa aproximadamente 6,2MB
  O zip da pasta pode ser obtido via **wget http://marco.uminho.pt/disciplinas/CC-LEI/tp2-folder2.zip**
- **tp2-folder3**: pasta com 6 ficheiros de diferentes tipos, alguns iguais aos da pasta tp2-folder2 e que ocupa 38MB
  O zip da pasta pode ser obtido via **wget http://marco.uminho.pt/disciplinas/CC-LEI/tp2-folder3.zip**

Sugerem-se os seguintes cenários de teste, sem prejuízo de que possam ser usados outros adicionalmente:

1. Sincronizar a pasta *tp2-folder1* do **Servidor1** com uma pasta vazia *orca1* no servidor **Orca**. O programa deverá transferir corretamente o ficheiro de texto em falta. Obter tempo de descarga e débito real em bits/s.
2. Sincronizar a pasta *tp2-folder1* colocada no **Servidor1** com a pasta vazia *grilo1* no portátil **Grilo**. No caminho há um link que introduz erros, nomeadamente pacotes perdidos e duplicados. O programa deverá transferir corretamente o ficheiro de texto em falta. Obter tempo de descarga e débito real em bits/s. Comprovar que os conteúdos foram transferidos corretamente.
3. Sincronizar a pasta *tp2-folder2* do **Servidor1** e a pasta vazia *orca2* no servidor **Orca**. O programa deverá transferir corretamente os múltiplos ficheiros, em simultâneo. Obter tempo de descarga e débito real em bits/s
4. Sincronizar a pasta *tp2-folder2* colocada no **Servidor1** e com a pasta *tp2-folder3* colocada no servidor **Orca**. Ambas as pastas possuem inicialmente o conteúdo predefinido. Alguns ficheiros são comuns, outros não. Um deles possui uma ligeira diferente de tamanho e de data. O programa deverá conseguir sincronizar as duas pastas transferindo apenas os ficheiros necessários nos dois sentidos. Obter tempo de descarga e débito real em bits/s

FASE 1: Desenho e teste do protocolo FT-Rapid

Etapas sugeridas para esta fase:

- Especificar o protocolo para funcionar sobre UDP
  - formato das mensagens protocolares (sintaxe)
  - função e significado dos campos (semântica)
  - diagrama temporal ilustrativo (comportamento)
- Implementação e teste do protocolo
  - exemplo de teste: pedir uma lista de ficheiros predefinida

FASE 2: Implementação do interface de monitorização

Etapas sugeridas para esta fase:

- Aceitar pedidos HTTP GET / na porta TCP 80
- Devolver uma resposta HTTP Response correta em formato txt ou html (ou ambos a pedido)

Referências

1. Secção 2.7, do Capítulo 2 do livro recomendado "*Computer Networking: A Top-Down Approach, 7th Edition*", J. Kurose and K. Ross
2. https://pt.wikibooks.org/wiki/Redes_de_computadores/Programação_com_sockets

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
  * formato das mensagens protocolares
  * 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-PLx-Gy-Rel.pdf** para o relatório
- **CC-TP2-PLx-Gy-Codigo.zip** ou **CC-TP2-PLx-Gy-Codigo.rar**) para a pasta com o código desenvolvido (devidamente documentado)

em que **x** é o identificador do turno PL e **y** o número do grupo (e.g. CC-TP2-PL2-G2-Rel.pdf para o relatório TP2 do grupo 2 do PL2)