

FolderFastSync: Sincronização rápida de pastas em ambientes serverless

Duarte Serrão¹, Mário Coelho², and Rita Gomes³

¹ a83630

² a42865

³ a87960

Resumo O presente relatório apresenta um resumo das principais etapas do desenvolvimento de um serviço de sincronização de pastas espontâneo e ultra-rápido. Este serviço, designado por FolderFastSync, foi concebido de modo a não necessitar de servidores nem de conectividade Internet. Trata-se de um serviço que opera ao nível da camada de aplicação da pilha protocolar, usando ao nível da camada de transporte o protocolo UDP de modo a ser o mais rápido possível. Este documento detalha a forma como o serviço foi concebido, em particular os detalhes de um protocolo definido para o efeito e designado FTRapid.

Keywords: Sincronização · Serverless · UDP · FTRapid.

1 Introdução

A ideia de base do serviço FolderFastSync é que, a partir de uma determinada máquina, se envie um pedido para uma outra máquina indicando o endereço IP da máquina de destino e ainda o nome de uma pasta que se pretende sincronizar. Este serviço pressupõe que a aplicação é iniciada na pasta raiz da partilha, à semelhança do que acontece com serviços de sincronização deste género existente (e.g. Onedrive). Assim, por exemplo a pasta `"/folder1"` estará na raiz e a pasta `"/folder1/folder2"` é uma sub-pasta da anterior.

Tal como nos serviços existentes, é possível existir uma abordagem passiva, em que a aplicação de sincronização não detecta modificação na pasta raiz e por isso fica só à escuta de solicitações do exterior (da serviço na cloud no caso do Onedrive), ou uma abordagem activa, na qual a aplicação detectou alterações na pasta raiz e trata de as comunicar às máquinas que lhe estão associadas. No serviço FolderFastSync procurou-se de certa forma replicar este comportamento.

Foi assumido que, para evitar confusões com a porta normal dos protocolos TCP e UCP, este serviço iria receber pedidos TCP e UDP na porta 8888.

De modo a simplificar a forma de autenticação no serviço, foi assumido que as máquinas em comunicação se conheciam. Nomeadamente, foi considerado que estas partilhavam um ficheiro de configuração da aplicação (designado `config.ftr`) que contém os pares IP - chave de acesso. Assim, sempre que uma máquina recebe um pedido inicial, tem primeiro de verificar que a máquina que enviou o pedido é de confiança, através da validação da chave desta na seu ficheiro de configurações local.

2 Arquitetura da solução

Para materializar o serviço FolderFastSync recorreu-se à linguagem de programação Java. Esta apresenta um conjunto de vantagens ao nível da gestão de sockets e threads comparativamente a outras linguagens, aspecto que é bastante relevante para o serviço implementado. Na Figura 1 apresentam-se as classes incluídas no desenvolvimento do serviço.

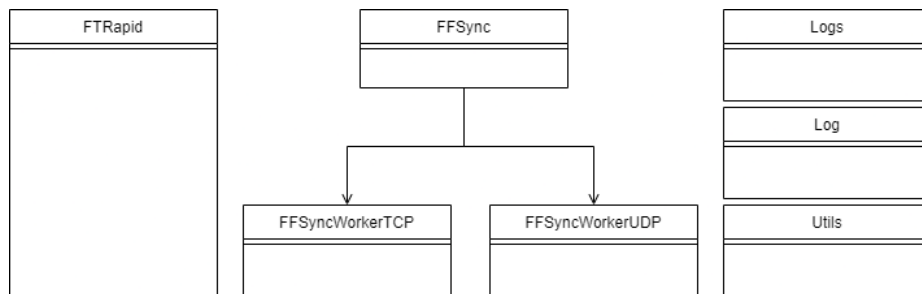


Figura 1. Classes incluídas no serviço FolderFastSync.

De uma forma resumida, a classe de entrada na aplicação é a FFSync. Esta permite o lançamento de duas threads de execução, além da thread principal. A lógica de ter partido a execução em três threads é a seguinte:

- numa thread é aberto um socket TCP e ficará, durante a execução da aplicação, continuamente à escuta de pedidos TCP que cheguem à aplicação. Estes pedidos serão depois processados pela classe FFSyncWorkerTCP, detalhada adiante;
- numa outra thread é aberto um socket UDP e ficará, durante a execução da aplicação, continuamente à escuta de pedidos UDP que cheguem à aplicação. Estes pedidos serão depois processados pela classe FFSyncWorkerUDP, detalhada adiante;
- a thread principal, apenas serve para enviar o pedido de sincronização a partir da máquina onde o serviço está a correr. Ou seja, enquanto as outras duas threads actuam de forma passiva, dando resposta a pedidos que cheguem, neste caso atua-se de forma activa.

A classe FFSyncWorkerTCP limita-se a receber o socket com a ligação à máquina que fez o pedido HTTP e colocar neste o conteúdo actual da classe Logs (detalhada adiante).

A classe FFSyncWorkerUDP trata de processar os pacotes UDP recebidos, solicitar à classe FTRapid a sua decodificação e, por fim, trata do envio de um novo pacote UDP com a resposta que a classe FTRapid definiu.

No lado direito da Figura 1 apresentam-se três classes auxiliares que foram necessárias para arrumar melhor a aplicação. De baixo para cima, a classe Utils

contém um conjunto de métodos necessários para input e output, bem como outras tarefas de manuseamento e comparação de estruturas de dados, que as classes principais necessitam. A classe Log contém apenas a estrutura de dados do log tipo. Este contém quatro parâmetros que são armazenados/atualizados a cada novo pacote recebido. A classe Logs contém um conjunto de entidades Log, uma por cada ficheiro solicitado por cada uma das máquinas que estejam ligadas ao serviço FolderFastSync e a comunicar com a máquina actual.

Do lado esquerdo da figura, está a classe principal da aplicação, FTRapid, classe esta que materializa o protocolo de comunicação entre instâncias diferentes do serviço FolderFastSync. A aplicação foi pensada desta forma de modo a permitir que, no futuro, seja possível alterar o protocolo de comunicação sem que isso cause grandes impactos no funcionamento global da aplicação. As principais funcionalidades desta classe são a decodificação dos pacotes de dados recebidos, interpretando correctamente as mensagens que estes transportavam, e a codificação de mensagens de resposta adequadas aos pedidos que vão chegando. Na secção seguinte detalha-se o protocolo FTRapid que esta classe pretende materializar.

3 Especificação do protocolo

De modo a ilustrar o protocolo definido, a presente secção divide-se em duas partes. Na primeira detalham-se as mensagens protocolares previstas, enquanto na segunda se ilustra o funcionamento do protocolo com diagramas de sequência representativos.

3.1 Formato das mensagens protocolares

De modo a tornar o serviço o mais eficiente possível, procurou-se ter mensagens protocolares com o menor overhead possível. Assim, definiu-se como mensagem protocolar de base a que se apresenta na Tabela 1. Esta possui um overhead total de cinco bytes e um comprimento total de 512 bytes. Estão previstos essencialmente três campos:

- Número de Sequência: indica a ordem relativa de cada mensagem dentro do conjunto de fragmentos em que a informação a transmitir foi dividida;
- Flag Último: indica se o fragmento de dados enviado na mensagem actual é o último ou não de modo a permitir a reconstrução no destino da informação inicial;
- Dados: conjunto de até 507 bytes contendo a informação da mensagem a transmitir.

A escolha do tamanho total deveu-se ao facto de se pretender um serviço rápido. Assim sendo, é importante minimizar as tarefas que podem introduzir demoras no serviço. Uma dessas tarefas diz respeito à eventual fragmentação e reconstrução de mensagens muito grandes. Assim, é importante perceber, ao

| | | | |
|---------------------|-------------|-------|-----|
| 0 | 3 | 4 | 512 |
| Número de Sequência | Flag Último | Dados | |

Tabela 1. Formato das mensagens protocolares.

longo da pilha protocolar, quais os potenciais pontos onde estas tarefas podem acontecer. Normalmente é ao nível da camada física que se encontram as maiores limitações, além de que quem desenvolve as aplicações desconhece os meios físicos que estarão disponíveis durante a utilização das aplicações.

Tendo em consideração os aspectos referidos acima, optou-se por um tamanho máximo de mensagem de 512 bytes. Note-se ainda que este é o tamanho dos pacotes de dados DNS, sendo este um dos serviços mais rápidos da internet.

Apesar de só existir um único formato para as mensagens protocolares, tendo em conta o conteúdo das mesmas, estão previstos vários tipos de pacotes distintos. São estes que o protocolo FTTRapid reconhece e implementa na classe FTTRapid mencionada anteriormente.

Mensagem Inicial

Na Tabela 2 apresenta-se a adaptação da mensagem protocolar tipo ao cenário em que se pretende enviar a mensagem inicial de ligação. Sendo esta a primeira interação entre a máquina que envia o pedido e a máquina que o recebe, o número de sequência destas mensagens foi definido como -3. A flag que indica se é o último fragmento foi definida como -1 pois trata-se de informação irrelevante nesta mensagem. Quanto ao campo dos dados, este encontra-se dividido em duas partes. Uma primeira parte com indicação da pasta a sincronizar. Uma segunda com indicação da chave da máquina de destino de modo a permitir que esta valide que ambas as máquinas se conhecem. No final de cada uma das strings que definem a pasta e chave de destino adicionou-se uma marca (`\0`) para permitir distinguir estes dois campos.

| | | | |
|---------------------|-------------|--------|---------------|
| 0 | 3 | 4 | 512 |
| Número de Sequência | Flag Último | Dados1 | Dados2 |
| -3 | -1 | Pasta | Chave Destino |

Tabela 2. Formato da mensagem inicial.

Mensagem Lista

Na Tabela 3 apresenta-se a adaptação da mensagem protocolar tipo ao cenário em que se pretende enviar a mensagem contendo a lista de ficheiros existentes na pasta solicitada anteriormente na mensagem inicial. Neste caso o número de sequência destas mensagens foi definido como -2. A flag que indica se é o último fragmento foi igualmente definida como -1, admitindo-se assim que a lista cabe dentro do espaço disponível. No campo dos dados apenas vai existir a lista dos ficheiros terminada com a marca (`\0`) para permitir saber onde esta termina no

caso de não ocupar a totalidade do espaço disponível para a mensagem. Note-se que, embora aqui se designe apenas por lista, na verdade o que é enviado é a lista com nome, tamanho e data da última modificação de cada ficheiro. Esta informação será necessária no destino para verificar se é ou não necessário pedir uma cópia de cada ficheiro.

| | | | |
|---------------------|-------------|--------|-----|
| 0 | 3 | 4 | 512 |
| Número de Sequência | Flag Último | Dados1 | |
| -2 | -1 | Lista | |

Tabela 3. Formato das mensagem lista.

Mensagens Pedido Ficheiro

Na Tabela 4 apresenta-se a adaptação da mensagem protocolar tipo ao cenário em que se pretende enviar a mensagem a solicitar o envio de um ficheiro específico. Neste caso o número de sequência será diferente consoante se trate de um pedido inicial ou de um pedido seguinte. No caso do pedido inicial, o número de sequência será -1, de modo a que, ao adicionar o tamanho do primeiro bloco de dados a receber, este número fique coerente com a indexação de arrays iniciada em 0. A flag que indica se é o último fragmento foi novamente definida como -1 por ser irrelevante neste contexto. No campo dos dados apenas vai existir o nome do ficheiros terminado com a marca (`\0`) para permitir saber onde este termina no caso de não ocupar a totalidade do espaço disponível para a mensagem.

| | | | |
|---------------------|-------------|----------|-----|
| 0 | 3 | 4 | 512 |
| Número de Sequência | Flag Último | Dados1 | |
| -1 ou N | -1 | Ficheiro | |

Tabela 4. Formato das mensagens pedido de ficheiro.

Mensagem Dados

Na Tabela 5 apresenta-se a adaptação da mensagem protocolar tipo ao cenário em que se pretende enviar a mensagem com um bloco de dados de um ficheiro específico. Neste caso o número de sequência será um número natural indicando o byte onde se inicia o bloco de dados a enviar. Em particular, no caso do envio inicial, o número de sequência será 0. A flag que indica se é o último fragmento tem, no caso da mensagem de dados, significado relevante. O seu valor poderá ser 0, no caso de o bloco de dados a enviar não ser o último daquele ficheiro, ou 1 no caso contrário. No campo dos dados apenas vai existir o bloco dos dados terminado com a marca (`\0`) para permitir saber onde este termina no caso de não ocupar a totalidade do espaço disponível para a mensagem.

| | | | |
|---------------------|-------------|----------|--------|
| 0 | 3 | 4 | 512 |
| Número de Sequência | Flag Último | Dados1 | Dados2 |
| N | 0 ou 1 | Ficheiro | Dados |

Tabela 5. Formato da mensagem de envio de dados.

3.2 Interações

Na Figura 2 apresenta-se o diagrama de sequência do funcionamento do protocolo FTTRapid. Nesta é possível ver o cenário de base que é expectável que aconteça.

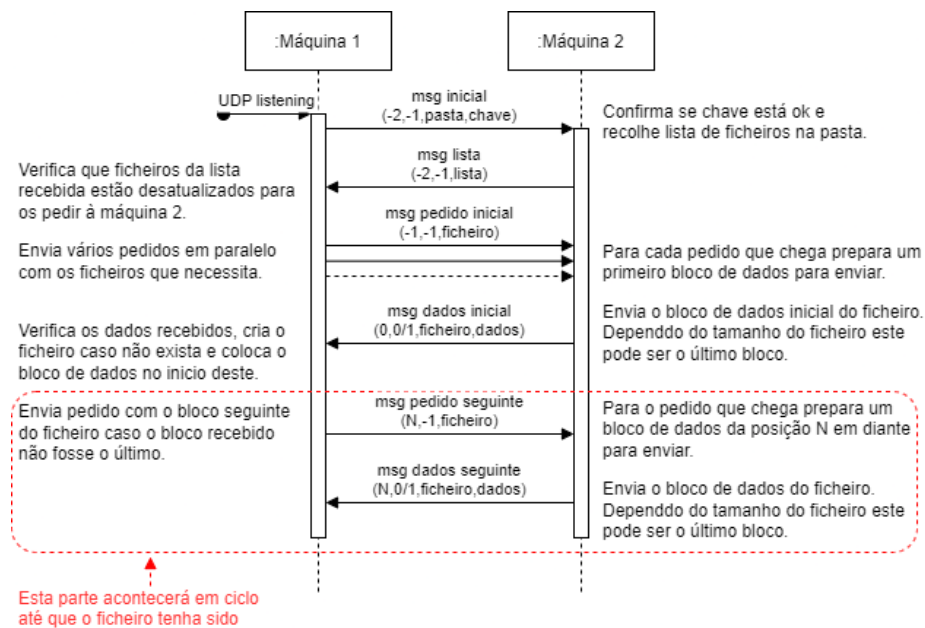


Figura 2. Diagrama de sequência do funcionamento do protocolo FTTRapid.

4 Testes e resultados

Para validar o serviço implementado foram realizados alguns testes conforme detalhado nos parágrafos que se seguem.

Pedidos HTTP GET

Era requisito da aplicação que esta fosse capaz de receber pedidos HTTP GET e devolve-se o estado do seu funcionamento. Na aplicação desenvolvida, existia uma thread apenas dedicada a receber pedidos TCP e que devolvia o

conteúdo da estrutura de logs mantida em memória durante o funcionamento das aplicação. Assim, usando a topologia core, foi simulada a sincronização da directoria tp2-folder entre a máquina Orca (efectuou o pedido) e a máquina Servidor 1 (recebeu o pedido). Durante este processo foi enviado a partir da máquina Portátil 2 um pedido HTTP GET para a máquina Orca. O resultado aparece na parte inferior da Figura 3.

Nesta é possível verificar dois instantes em o Portátil 2 envia o pedido. Na parte superior da figura, vê-se um primeiro pedido efectuado enquanto o processo de transferência do ficheiro ainda não tinha terminado. Por isso, em alguns dos parâmetros aparece a indicação "ongoing".

Na parte inferior da figura vê-se um segundo instante no qual o processo de transferência já tinha terminado. Neste caso, já todos os parâmetros têm valores. É possível verificar que a máquina Orca tinha recebido, no intervalo de tempo indicado, da máquina com IP 10.2.2.1 (Servidor 1) um bloco de informação relativo ao ficheiro tp2-folder/rfc7231.txt. Este envio demorou 1 segundo com um débito de 1,88MB/s.

```

vcmd
root@Portatil12:/tmp/pycore.39789/Portatil2.conf# wget 10.3.3.1:8888
--2021-12-23 23:31:35-- http://10.3.3.1:8888/
Connecting to 10.3.3.1:8888... connected.
HTTP request sent, awaiting response... 200 No headers, assuming HTTP/0.9
Length: unspecified
Saving to: 'index.html'

index.html          [ <=> ] 205 --KB/s in 0s

2021-12-23 23:31:35 (53,6 MB/s) - 'index.html' saved [205]

root@Portatil12:/tmp/pycore.39789/Portatil2.conf# ll
total 32
drwxrwxrwx 4 root root 4096 dez 23 23:31 ./
drwxrwxrwx 21 root root 4096 dez 21 21:40 ../
-rw-r--r-- 1 root root 139 dez 21 21:40 defaultroute.sh
-rw-r--r-- 1 root root 206 dez 23 23:31 index.html
-rw-r--r-- 1 root root 181 dez 21 21:40 preenche_resolvoconf.sh
-rw-r--r-- 1 root root 280 dez 21 21:40 stationroute.sh
drwxrwxrwx 2 root root 4096 dez 21 21:40 var.log/
drwxrwxrwx 3 root root 4096 dez 21 21:40 var.run/
root@Portatil12:/tmp/pycore.39789/Portatil2.conf# cat index.html
SenderIP, FileFullPath, TimeStart, TimeEnd, TimeTotal, SizeTotal, SizeTransferred, TransferRate
/10.2.2.1, tp2-folder1/rfc7231.txt, 2021-12-23T23:31:34.401934813, ongoing, ongoing, 235053, 129390, ongoing
root@Portatil12:/tmp/pycore.39789/Portatil2.conf# wget 10.3.3.1:8888
--2021-12-23 23:32:07-- http://10.3.3.1:8888/
Connecting to 10.3.3.1:8888... connected.
HTTP request sent, awaiting response... 200 No headers, assuming HTTP/0.9
Length: unspecified
Saving to: 'index.html.1'

index.html.1        [ <=> ] 221 --KB/s in 0s

2021-12-23 23:32:07 (34,5 MB/s) - 'index.html.1' saved [221]

root@Portatil12:/tmp/pycore.39789/Portatil2.conf# cat index.html.1
SenderIP, FileFullPath, TimeStart, TimeEnd, TimeTotal, SizeTotal, SizeTransferred, TransferRate
/10.2.2.1, tp2-folder1/rfc7231.txt, 2021-12-23T23:31:34.401934813, 2021-12-23T23:31:36.336963970, 1, 235053, 235172, 1890424
root@Portatil12:/tmp/pycore.39789/Portatil2.conf#

```

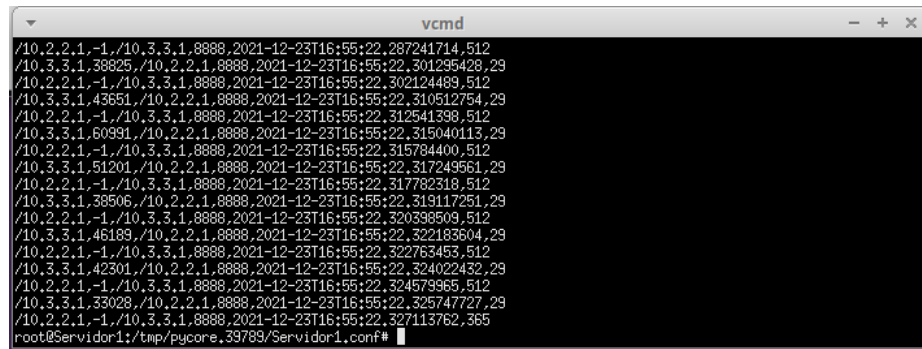
Figura 3. Resultado do teste HTTP GET.

Registo dos logs de utilização

Um outro requisito da aplicação era que armazenasse um registo dos eventos que fossem acontecendo durante a utilização da aplicação. Isso foi materializado

através de um ficheiro de logs que estaria armazenado na pasta raiz da aplicação. Este contém todas as trocas de mensagens registadas pela máquina em causa.

Na Figura 4 apresenta-se um exemplo do conteúdo do ficheiro de logs no cenário apresentado atrás para o teste de pedidos HTTP GET. Nesta é possível identificar os sucessivos pedidos de mais blocos do ficheiro (linhas terminadas em 29, sendo este o tamanho da mensagem) feitos pela máquina Orca (IP 10.3.3.1) à máquina Servidor 1 (IP 10.2.2.1). Nas restantes linhas verifica-se o envio de novos blocos do ficheiro pedido. Nas linhas terminadas em 512 (tamanho da mensagem) são enviados blocos do ficheiro intermédios. Na última linha, correspondendo ao último evento registado no teste, é possível ver que o tamanho da mensagem é de apenas 365 bytes uma vez que o último bloco não tinha tamanho suficiente para ocupar a totalidade do espaço disponível na mensagem.



```
vcmd
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,287241714,512
/10.3.3.1,38825,/10.2.2.1,8888,2021-12-23T16:55:22,301295428,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,302124489,512
/10.3.3.1,43651,/10.2.2.1,8888,2021-12-23T16:55:22,310512754,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,312541398,512
/10.3.3.1,60991,/10.2.2.1,8888,2021-12-23T16:55:22,315040113,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,315784400,512
/10.3.3.1,51201,/10.2.2.1,8888,2021-12-23T16:55:22,317249561,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,317782318,512
/10.3.3.1,38506,/10.2.2.1,8888,2021-12-23T16:55:22,319117251,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,320398509,512
/10.3.3.1,46189,/10.2.2.1,8888,2021-12-23T16:55:22,322183604,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,322763453,512
/10.3.3.1,42301,/10.2.2.1,8888,2021-12-23T16:55:22,324022432,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,324579965,512
/10.3.3.1,33028,/10.2.2.1,8888,2021-12-23T16:55:22,325747727,29
/10.2.2.1,-1,/10.3.3.1,8888,2021-12-23T16:55:22,327113762,365
root@Servidor1:/tmp/pycore.39789/Servidor1.conf#
```

Figura 4. Exemplo de ficheiro com logs de utilização.

5 Conclusões e trabalho futuro

O presente relatório apresentou os principais detalhes do desenvolvimento de um serviço de sincronização de pastas. Para tal, era também necessário pensar e desenvolver um novo protocolo de comunicação que suportasse esse serviço. Apesar de não ter sido possível atingir todos os objectivos propostos para o presente trabalho, o objectivo principal foi atingido - expor o grupo a este tipo de trabalhos e obrigar a pensar em soluções práticas para os conceitos teóricos que se abordaram ao longo do semestre.

Alguns dos principais aspectos a melhorar em futuras implementações da aplicação são:

- As questões relativas à forma de autenticação e segurança no acesso à aplicação terão de ser melhoradas;
- É possível reduzir o overhead das mensagens cujo número de sequência é apenas um número (-3, -2 ou -1). Neste caso, por simplificação, considerou-

se sempre 4 bytes para o número de sequência gerando-se desperdício nos casos referidos;

- A mensagem com a lista dos ficheiros existentes na pasta a sincronizar poderá exceder o espaço definido para o efeito. Assim, é preciso prever que a lista vá em mais do que uma mensagem, sendo necessária depois juntá-la no destino;