Instrukcja obsługi The cake router

Instalacja

Przejdź do folderu, w którym chcesz zainstalować program, pobierz i rozpakuj

```
$ wget https://github.com/jacawaca/the-cake-router/releases/download/cake/v1.0-cake.zip
$ unzip v1.0-cake.zip
```

Skrypty i jary powinny znaleźć się w folderze. Skrypt może działać bez instalacji, ale dla ułatwienia celowym może być wykorzystanie krótkiego aliasu lub dodanie folderu ze skryptem do zmiennej `$PATH`.

Metoda pierwsza,

```
$ sudo echo "export cake-router=./path/to/file/CakeClient.sh" >> ~/.bash_aliases
$ cd ~
$ . .bashrc
```

Metoda druga,

```
$ sudo echo "export PATH=/path/to/dir/CakeClient:$PATH" >> ~/.bashrc
```

Działanie

Klient uruchamia program może zatem uruchamiać program poprzez skrypt lub alias. Załóżmy, że będzie to robił bezpośrednio wywołując skrypt.

```
$ ./CakeClient.sh
```

Dostępne opcje można wyświetlić przy pomocy komendy *help*

```
$ ./CakeClient.sh --help
```

Ważną opcją jest możliwość wyświetlania informacji nt. węzłów pośrednich. Użytkownik może ją uzyskać przy pomocy -l lub –list

```
$ ./CakeClient.sh -l # lub --list
# LUB
$ ./CakeClient.sh --list
```

Kluczową opcją, jest opcja *send*

```
$ ./CakeClient.sh -s adres msg.txt
# lub
$ ./CakeClient.sh --send adres msg.txt
```

Wysyła plik msg.txt na adres, który ma mieć formę adresu IPv4. Być może program będzie działał dla nieliczbowych nazw domen (DNS), ale nie przetestowaliśmy tego działania. Otrzymanie odpowiedzi kończy działanie programu klienta.

Program korzysta z pliku konfiguracyjnego *adressess.config*. Jego obsługę zapewnia następująca opcja

```
$ ./CakeClient.sh --set-trasa
```

Wygenerowywuje ona plik konfiguracyjny, który umożliwia wybór trasy. Obecnie można w ten sposób wybrać jedynie dwa węzły pośrednie. Program działa dla większej ilości (testowaliśmy dla 2), więc w celu zadeklarowania większej ilości węzłów pośrednich trzeba ręcznie edytować adressess.config. Np. tak

```
$ cd /path/to/cake
$ vim adressess.config
```

W pliku konfiguracyjnym znajdują się adresy kolejnych węzłów pośrednich. Plik tekstowy zawierający wiadomość powienien zostać uprzednio przygotowany. Plik tekstowy nie może zawierać średników `;`.

Obsługa serwera pośredniczącego

Przed wysłanie wiadomości wszystkie węzły pośrednie wraz z adresatem muszą uruchomić skrypt odbierający

```
$ ./CakeNode.sh
```

W przypadku gdy dana maszyna nie jest adresatem skrypt ten służy do przesłania komunikatu dalej, bez ingerencji użytkownika. Gdy paczka dotrze do adresata, wyświetli się komunikat informujacy o nadejściu paczki oraz możliwości jej zapisu do nowego pliku, którą należy potwierdzić/odrzucić wpisując wartość logiczną (*true* lub *false*). Po akceptacji nadaje się nazwę nowemu plikowi i jest on zapisay w katalogu, w którym znajduje się skrypt. Następnie możliwe jest wpisanie tekstowej odpowiedzi zwrotnej do klienta. Nie można w niej zawrzeć żadnych średników `;`.

Sprawdzenie poprawności przesłanych danych

. Sprawdzenie poprawności wysłania i odbioru możliwe jest przy użyciu *sha256sum*. W tym celu proponujemy następującą procedurę. Załóżmy, że chcemy przesłać plik msg.txt na adres 192.168.1.1

```
$ sha256sum msg.txt > wynik_msg.txt
$ ./CakeClient.sh -s 192.168.1.1 msg.txt
```

Następnie gdy uda się to uda

```
$ ./CakeClient.sh -s 192.168.1.1 wynik_msg.txt
```

Adresat odbiera oba pliki (wpisuje kolejno true, i nazwę pliku; dwa razy: dla wiadomości i jej sumy kontrolnej). Załóżmy, że zapisał sobie wiadomość jako msg.txt i sumę kontrolną jako *clientSum_msg.txt*. Wówczas wykonuje on

```
$ sha256sum msg.txt > mySum_msg.txt
```

i dalej sprawdza, czy pliki są identyczne. Np. przy użyciu *diff*

```
$ diff -sq mySum_msg.txt clientSum_msg.txt
```