

Specyfikacja The cake router

Ryszard Michalski

Paweł Polak

Jacek Strzałkowski

29 kwietnia 2021

The cake router

UWAGA Polecamy zapoznać się z HTMLową wersją niniejszej specyfikacji dostępnej w repozytorium Github [link](#) .

Trasowanie cebulowe

Trasowanie cebulowe jest techniką służącą do anonimowej komunikacji w sieci komputerowej. Komunikacja odbywa się przy wykorzystaniu trzech rodzajów jednostek: klienci, węzły pośredniczące i końcowe. Połączenie klienta z odbiorcą wygląda następująco. W pierwszej kolejności klient wybiera listę węzłów P_1, P_2, \dots, P_N, WK po których chce dotrzeć do adresata. Następnie nadaje paczkę informacji $\{dane, adres[P_2-ADRESAT]\}$ do P_1 . Dalej P_1 zapisuje adres klienta i wysyła paczkę $\{dane, adres[P_3-ADRESAT]\}$ do P_2 . Proces się powtarza aż węzeł końcowy - WK - wyśle pakiet do adresata.

| URZĄDZENIE | OPERACJA | PAMIĘĆ |
|------------|----------|--------|
| KLIENT | | |
| P1 | | |
| P2 | | |
| P3 | | |
| WK | | |
| ADRESAT | | |

Figure 1: Trasowanie cebulowe

Droga w kierunku przeciwnym jest następująca: adresat wysyła odpowiedź do WK. Z kolei z WK jest wysyłana ta odpowiedź do P_N , do WK zapisał adres. Dalej P_N przesyła odpowiedź do P_{N-1} itd., aż odpowiedź wróci do klienta.

Funkcjonalności

Nasz program będzie spełniał następujące wymagania: - [X] umożliwi przesyłanie komunikatów UDP (*user datagram protocol*, protokół bezstanowy, nie zapewnia retransmisji danych, umożliwia przesyłanie danych do wielu użytkowników). - [X] Użytkownik otrzymuje instrukcję dotyczącą tego, w jaki sposób może wykonać kontrolę danych. Np. tak

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

- [X] Klient będzie posiadał listę wszystkich węzłów pośredniczących (P_i oraz WK) zapisaną w pliku konfiguracyjnym.
- [X] Klient będzie mógł łatwo podejrzeć listę przy pomocy bez znajomości nazwy pliku konfiguracyjnego.
- [X] Klient będzie mógł wygenerować listę węzłów pośrednich za pomocą odpowiedniej funkcji.
- [X] Komunikaty będą zawierały paczkę złożoną z informacji przesyłanych oraz adresów rozdzielonych średnikiem ;.
- [X] Możliwe będzie przesłanie paczki do innego hosta (który również «oferuje» swoje usługi jako węzeł pośredni) i otrzymanie od niego odpowiedzi. W zasadzie, możliwe będzie wysłanie paczki do każdego hosta (węzła). Program węzła pośredniego i adresata jest ten sam.
- [X] Sieć będzie w stanie obsłużyć równolegle przynajmniej dwóch klientów oraz wielu klientów po sobie.
- [X] Wiadomość powrotna będzie wysyłana każdemu klientowi osobno.
- [X] Adresat nie będzie mógł bezpośrednio zidentyfikować skąd pochodzi wiadomość. Dlatego nadawca powinien się podpisywać w swojej wiadomości.
- [X] Adresat będzie mógł wpisać swoją odpowiedź w terminalu.
- [X] Adresat będzie mógł zapisać komunikat do pliku
- [X] Adresat będzie zabezpieczony przed wpisaniem wiadomości z ;
- [X] Przed uruchomieniem programu klienckiego, program (z poziomu powłoki) sprawdzi, czy jest zainstalowana java. ## Obsługa programu Klient będzie mógł uruchamiać program poprzez prosty alias. Na potrzeby robocze założymy, że będzie to cake-router. Aby to osiągnąć, wystarczy wykonać na przykład

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

Klient uruchamia program cake-router w bashu poprzez wpisanie nazwy skryptu np. `./CakeClient.sh` lub (Patrz instrukcja) za pomocą aliasu.

```
$ cake-router --help
```

Wyświetla mu się informacja o możliwych opcjach. Następnie uruchamia program z opcją list

```
$ cake-router -l
```

Dostaje informacje o węzłach pośrednich z wczytanych z pliku konfiguracyjnego.

Możliwość wygenerowania pliku z adresami pośrednimi będzie realizowana za pomocą opcji

```
$ cake-router --set-trasa
```

Wysyłka odbywać się będzie za pomocą opcji `send`. Założymy, że użytkownik chce wysłać `msg.txt` na adres `192.168.1.1`

```
$ cake-router -s 192.168.1.1 msg.txt
```

Po wysłaniu użytkownik programu oczekuje na odpowiedź od nadawcy.

Węzeł pośredni będzie uruchamiał swój program w konsoli za pomocą

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

Gdy paczka dojdzie do adresata, będzie miał możliwość zapisania wiadomości do pliku oraz odpowiedzi. ## Działanie

```
Plik Maszyna Widok Wejście Urządzenia Pomoc
jacek@ubuntuserver1:~/cake$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.110 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:febc:e9c8 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:6c:e9:c8 txqueuelen 1000 (Ethernet)
    RX packets 66860 bytes 83801106 (83.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8397 bytes 851167 (851.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 229 bytes 19390 (19.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 229 bytes 19390 (19.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

jacek@ubuntuserver1:~/cake$

Plik Maszyna Widok Wejście Urządzenia Pomoc
jacek@ubuntuserver3:~/cake$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.109 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe1d:7a2f prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:1d:7a:2f txqueuelen 1000 (Ethernet)
    RX packets 66890 bytes 84286703 (84.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 10074 bytes 906709 (906.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 255 bytes 21330 (21.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 255 bytes 21330 (21.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

jacek@ubuntuserver3:~/cake$

Plik Maszyna Widok Wejście Urządzenia Pomoc
jacek@ubuntuserver2:~/cake$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.111 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fe56:55ae prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:56:55:ae txqueuelen 1000 (Ethernet)
    RX packets 14038 bytes 4349055 (4.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3972 bytes 529535 (529.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 209 bytes 17683 (17.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 209 bytes 17683 (17.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

jacek@ubuntuserver2:~/cake$
```

Nadanie wiadomości i wybór trasy

```
Podgląd OBS Studio 28 maj 14:47
jacek@domowy: /media/jacek/pliki/uczelnia/6-semestr/sk/lab/the-cake-router/exe
jacek@domowy: /media/jacek/pliki/uczelnia/6-semestr/sk/lab/the-cake-router/exe$ cat message.txt
Cześć!
Piszę poprzez TOR.
Z poważaniem.
Marian
jacek@domowy: /media/jacek/pliki/uczelnia/6-semestr/sk/lab/the-cake-router/exe$
```

```
Plik Maszyna Widok Wejście Urządzenia Pomoc
student@student-VB: ~/cake
student@student-VB: ~/cake$ cat message.txt
Cześć!
Piszę poprzez TOR.
Z poważaniem.
Jan
student@student-VB: ~/cake$
```

Nadanie wiadomości poprzez dwa węzły pośrednie