Zadanie programistyczne:
Sieć proxy do agregacji danych
SKJ (2025)

Wstęp

Dawno temu, w odległej galaktyce, była sobie planeta nazywana Numerlandią. Planeta była podzielona na dwie części: jedną z nich rządzili Tecepowie, drugą zaś Udepowie. Obie części różniły się od siebie prawie wszystkim, włącznie z językiem tak, że ich mieszkańcy nie potrafili ze sobą rozmawiać. Czas jednak nie stał w miejscu i po latach planeta zdołała się zjednoczyć pod rządami mądrego Numeryka. Władca ten postanowił przeprowadzić operację mającą na celu połączenie obu części pomimo różnic - rozpoczął od unifikacji wiedzy. Cała wiedza na planecie pozostawała pod opieką strażników kluczy - ponieważ każda informacja składała się z klucza (nazwy) oraz przypisanej mu wartości. Każdy strażnik sprawował pieczę nad jedną parą klucz-wartość. Każdy z nich zamieszkiwał jedną wieżę Mieszkańcy mogli udać się do strażnika i zapytać go, jaki klucz przechowuje oraz poprosić o jego aktualną wartość. Mogli też poprosić o zmianę aktualnej wartości na inną. Niestety, zarówno mieszkańcy jak i strażnicy potrafili mówić wyłącznie własnym językiem, co uniemożliwiało dostęp do wszystkich informacji na zjednoczonej planecie. Mądry Numeryk chciał ułatwić mieszkańcom życie nie tylko poprzez danie im możliwości dostępu do wiedzy bez względu na używany język, ale również przez jej połączenie aby nie trzeba było przemierzać krainy od strażnika do strażnika. W tym celu postanowił zorganizować sieć pośredników. Każdy pośrednik mógł rozmawiać ze strażnikami, którzy byli w jego okolicy. Mógł też rozmawiać z innymi pośrednikami i wymieniać z nimi informacje. Potrafił przy tym posługiwać się obydwoma językami używanymi przez mieszkańców dzięki czemu każdy, kto zwrócił się do niego z prośbą o pośredniczenie, mógł być obsłużony. A mieszkańcy mogli żyć długo i szczęśliwie.

Sformułowanie stanu początkowego

Dane są programy implementujące strażników kluczy (serwery) oraz mieszkańców (klientów), każdy w dwóch wersjach obsługującej wyłącznie jeden z protokołów TCP lub UDP. Archiwum z kodami tych programów w języku Java jest załącznikiem do zadania (w sekcji z materiałami kursu). Procesy serwerów TCP i UDP uruchamiane są w następujący sposób:

`java [TCPServer|UDPServer] -port <numer> -key <nazwa> -value <wartość>`

gdzie `<numer>` jest liczbą całkowitą określającą numer portu TCP/UDP na którym dany serwer będzie oczekiwał na żądania od klientów, `<nazwa>` to string (bez znaków białych) stanowiący nazwę dla wartości przechowywanej w serwerze, a `<wartość>` to liczba całkowita równa wartości przechowywanej przez serwer, skojarzonej z danym kluczem. Należy założyć, że wśród wszystkich serwerów istnieje maksymalnie jeden opiekujący się kluczem o danej nazwie.

Procesy klientów dla TCP i UDP uruchamia się następujaco:

`java [TCPClient|UDPClient] -address <adres> -port <numer> -command <komenda>`

gdzie `<adres>` jest adresem, na którym pracuje strażnik (serwer), z którym dany klient chce się skomunikować, `<numer>` jest liczbą całkowitą określającą numer portu TCP/UDP na którym te serwer pracuje, `<komenda>` to ciąg słów zależny od sytuacji stanowiący komendę, która ma zostać przesłana do serwera (następny rozdział).

Podstawowy protokół komunikacyjny

Komunikacja między klientem a serwerem prowadzona jest według Ściśle określonego protokołu, w którym komunikaty mają formę ściśle określonych ciągów tekstowych. Każda komenda od klienta zajmuje wiersz zakończony znakiem końca linii, każda odpowiedź od serwera to również jeden wiersz tekstu zakończonego znakiem końca linii. W przypadku protokołu UDP należy założyć, że cała wiadomość mieści się w jednym datagramie.

- `GET NAMES`

  Jest to żądanie zwrócenia nazwy (nazw) kluczy przechowywanych w danym serwerze. Odpowiedź zawsze ma postać:

  `OK <liczba> [nazwa]+`

  gdzie `<liczba>` określa liczbę przesyłanych nazw kluczy, a `[nazwa]+` to niepusta lista nazw kluczy rozdzielonych znakami spacji. Bazowe serwery przechowyją wyłącznie pojedyncze klucze, możliwość przesyłania większej liczby wynika z potencjalnej dalszej współpracy z pośrednikami.

- `GET VALUE <nazwa>`

  Jest to żądanie zwrócenia wartości przypisanej do danego klucza, gdzie `<nazwa>` to nazwa tego klucza. Jeśli dany serwer przechowuje klucz o takiej nazwie, odpowiedź ma postać:

  `OK <wartość>`

  gdzie `<wartość>` to liczba przypisana do żądanego klucza. Jeśli nazwa klucza jest niewłaściwa dla danego serwera, odpowiedź ma postać:

  `NA`

- `SET <nazwa> <wartość>`

  Jest to żądanie przypisania nowej wartości do klucza przechowywanego na danym serwerze, przy czym `<nazwa>` to nazwa tego klucza a `<wartość>` to liczba będąca przypisywaną wartością. Jeśli serwer przechowuje klucz o zadanej nazwie, zapamiętuje podaną wartość oraz zwraca odpowiedź postaci:

  `OK`

  W przeciwnym wypadku odpowiedzią jest ciąg:

  `NA`

- `QUIT`

  Komenda służąca do zatrzymania serwera. Po jej otrzymaniu serwer kończy pracę bez wysyłania jakiegokolwiek komunikatu zwrotnego.

W każdym przypadku, gdy serwer otrzyma komendę niezgodną z powyższym schematem, odpowiedzią jest wiersz postaci:

`NA`

Klient TCP umie rozmawiać wyłącznie z serwerem TCP, a klient UDP - tylko z serwerem UDP. W obydwu przypadkach po wymianie jednej pary komenda–odpowiedź klient kończy pracę (następuje rozłączenie połączenia TCP). Serwery natomiast nie utrzymują połączeń - rozłączają się i oczekują na kolejne żądania, które są obsługiwane sekwencyjnie.

Sformułowanie zadania

Zadanie polega na stworzeniu **aplikacji pośrednika** (*proxy*), która będzie w stanie komunikować się z dowolnie wieloma serwerami (bez względu na wykorzystywany przez nie protokół), z dowolnie wieloma klientami (bez względu na wykorzystywany przez nich protokół) oraz z innymi pośrednikami. **Nie wolno zmieniać aplikacji klientów i serwerów dostarczonych wraz z zadaniem.** Należy opracować i opisać protokół komunikacji między pośrednikami mający na celu propagowanie wiedzy oraz przekazywanie żądań klientów tak, aby mogły one docierać do właściwych serwerów strażników, których dotyczą.

Sposób działania aplikacji

Na rozwiązanie składa się jeden program implementujący klasę `Proxy` uruchamiany poleceniem:

`java Proxy -port <port> -server <adres> <port> ...`

gdzie pierwsza wartość `<port>` to numer portu (zarówno TCP jak i UDP), z którego będą mogli korzystać klienci, a para `<address> <port` to adres oraz port na którym działa albo serwer strażnika albo inna instancja proxy, z którą należy się połączyć. Zadaniem pośrednika jest detekcja protokołu komunikacyjnego, którego używa dany serwer (w

przypadku proxy można wybrać dowolnie zgodnie z decyzją programisty). Ciąg `-server <adres> <port>` występuje co najmniej raz, ale może zostać podany wielokrotnie i w takim przypadku pośredni musi utrzymywać kontakt z każdym z zadanych węzłów. Połączenie takie może być utrzymywane stale (w przypadku innych proxy) lub tylko wtedy, gdy jest to konieczne (dla serwerów). Taka sama para `<address> <port>` może zostać przekazana do wielu węzłów proxy. Dane proxy może rozmawiać bezpośrednio tylko z serwerami, które zostały mu przekazane jako parametry, do wszystkich innych może dotrzeć tylko przez inne węzły proxy.

- Po uruchomieniu proxy otwiera porty komunikacyjne (zarówno TCP jak i UDP), które posłużą mu do komunikacji z klientami lub innymi proxy.
- Następnie musi nawiązać kontakt ze wszystkimi zadanymi przez parametry węzłami, określić, które z nich to serwery a które to proxy. Następnie należy zgromadzić informację o przechowywanych w sieci kluczach – trzeba odpowiednio odpytać węzły oraz zapamiętać, jakimi kluczami dysponuje sieć (wszystkimi) oraz jak się ewentualnie do nich dostać (na których są serwerach w przypadku połączeń bezpośrednich lub przez które proxy można się do nich dostać).
- Podczas pracy proxy musi być gotowe na przyjmowanie zleceń od klientów i przekazywać je do odpowiednich serwerów. W szczególności ma umożliwić komunikację par używających różnych protokołów komunikacyjnych (np. TCPClient ¡-¿ UDPServer).
  - Odpowiedzią na komendę `GET NAMES` jest zgodna z protokołem komunikacyjnym klient–serwer lista wszystkich nazw kluczy dostępnych w sieci (uwaga na licznik na drugiej pozycji odpowiedzi).
  - W przypadku `GET VALUE <nazwa>` oraz `SET <nazwa> <wartość>` komendy te należy przekazać do sieci tak, aby mogły ostatecznie dotrzeć do odpowiednich serwerów opiekujących się zadanym kluczem oraz następnie przekazać do klienta odpowiedź z serwera. **W węzłach proxy nie przechowujemy wartości kluczy.** Jeśli sieć nie przechowuje klucza o zadanej nazwie, klient może otrzymać odpowiedni komunikat (`NA`) bezpośrednio z proxy.
  - Otrzymanie komendy `QUIT` oznacza zlecenie zakończenia pracy. W takim przypadku należy przesłać tę komendę dalej do sieci i zadbać o zakończenie pracy podległych sobie serwerów.

Należy opracować i opisać protokół służący węzłom proxy do komunikacji między nimi. Do komunikacji z klientami oraz serwerami proxy **muszą** wykorzystywać wyłącznie opisany powyżej protokół. Opracowany w zadaniu protokół może być zupełnie autorski, ale może też rozszerzać powyższy. Klient może łączyć się zarówno bezpośrednio z serwerem jak i z proxy, zależnie od przekazanego mu adresu i portu ale jego praca w obu przypadkach jest taka sama (**nie wolno modyfikować klientów**).

Wymagania i sposób oceny

1. W celu realizacji zadania należy zaprojektować i napisać proces implementujący protokół komunikacyjny, umożliwiający realizację opisanej powyżej funkcjonalności. Kwestię tego, jakiej treści pakiety są przesyłane między poszczególnymi procesami pozostawia się autorowi.

2. Proces jest uruchamiany z parametrami zgodnymi z powyższą specyfikacją. Jeśli liczba parametrów jest niepoprawna lub nie są one poprawne, proces ma zgłosić błąd i zakończyć pracę.

3. Poprawny i pełny projekt wart jest **500 punktów**. Za zrealizowanie poniższych funkcjonalności można otrzymać punkty do podanej wartości:

   - **maksymalnie 150 punktów** za program dla węzła proxy, który potrafi łączyć zbiór serwerów z klientami (bez innych węzłów proxy) i korzystać z wyłącznie jednego protokołu komunikacyjnego (TCP lub UDP).
   - **maksymalnie 300 punktów** za program dla węzła proxy, który potrafi łączyć zbiór serwerów z klientami (bez innych węzłów proxy) i korzystać jednocześnie z obydwu protokołów komunikacyjnych (TCP i UDP).
   - **maksymalnie 300 punktów** za program dla węzła proxy, który potrafi łączyć zbiór serwerów i innych proxy z klientami przy założeniu, że struktura połączeń to drzewo (nie ma cykli) i korzystać wyłącznie jednego protokołu komunikacyjnego (TCP lub UDP).
   - **maksymalnie 400 punktów** za program dla węzła proxy, który potrafi łączyć zbiór serwerów i innych proxy z klientami przy założeniu, że struktura połączeń to drzewo (nie ma cykli) i korzystać jednocześnie z obydwu protokołów komunikacyjnych (TCP i UDP).
   - **maksymalnie 500 punktów** za program dla węzła proxy, który potrafi łączyć zbiór serwerów i innych proxy z klientami przy założeniu, że struktura połączeń jest dowolna (mogą wystąpić cykle) i korzystać jednocześnie z obydwu protokołów komunikacyjnych (TCP i UDP).

4. Aplikację piszemy w języku Java zgodnie ze standardem Java 8 (JDK 1.8). Do komunikacji przez sieć można wykorzystać jedynie podstawowe, standardowe klasy do komunikacji dla danych protokołów.

5. Projekty powinny zostać zapisane do odpowiednich katalogów w systemie GAKKO w nieprzekraczalnym terminie **19.12.2025 23:55** (termin może zostać zmieniony przez prowadzącego grupę).

6. Spakowany plik projektu powinien obejmować:

- Plik *Dokumentacja_(nr indeksu).pdf*, opisujący, co zostało zrealizowane, co się nie udało, jak zainstalować, gdzie ewentualnie są błędy, których nie udało się poprawić. W szczególności, plik musi zawierać szczegółowy opis zaprojektowanego i zaimplementowanego protokołu (brak opisu protokołu lub jego fragmentaryczność może spowodować znaczące obniżenie oceny rozwiązania zadania).
- Pliki źródłowe (dla JDK 1.8) (włącznie z wszelkimi bibliotekami nie należącymi do standardowej instalacji Javy, których autor użył) - aplikacja musi dać się bez problemu skompilować na komputerach w laboratorium w PJA oraz musi działać zgodnie ze specyfikacją na dowolnym systemie operacyjnym na komputerze obsługującym język Java.

UWAGA: PLIK Z DOKUMENTACJĄ JEST WARUNKIEM KONIECZNYM PRZYJĘCIA PROJEKTU DO OCENY.

7. Prowadzący oceniać będą w pierwszym rzędzie poprawność działania programu i zgodność ze specyfikacją, ale na ocenę wpływać będzie także zgodność wytworzonego oprogramowania z zasadami inżynierii oprogramowania i jakość implementacji.

8. JEŚLI NIE WYSZCZEGÓLNIONO INACZEJ, WSZYSTKIE NIEJASNOŚCI NALEŻY PRZEDYSKUTOWAĆ Z PROWADZĄCYM ZAJĘCIA POD GROŹBĄ NIEZALICZENIA PROGRAMU W PRZYPADKU ICH NIEWŁAŚCIWEJ INTERPRETACJI.