Rozproszona baza danych – dokumentacja

Sposób organizacji sieci

Każdy węzeł sieci nasłuchuje na jednym porcie (zadanmym parametrem `--tcpport`) w nieskończonej pętli, aż do otrzymania komunikatu `terminate` od klienta. Połączenia z innymi węzłami są realizowante poprzez posiadanie słownika `HashMap<String, NodeConnectionHandler>` przez każdy z węzłów. Jedna para `<String, NodeConnectionHandler>` reprezentuje połączenie z innym węzłem – klucz przyjmuje wartość`adres:tcpport` aby jednoznacznie identyfikować inny węzeł w seici. Przy terminacji, węzeł wysyła innym węzłom informacje o zakończeniu swojego działania (`srv__disconnect`) aby inne węzły mogły usunąć go ze swojego słownika połączeń. Słowniki realizują tym sposobem nieskierowany (połączenia są zawsze w obie strony – po jednym `NodeConnectionHandler` z każdej strony) graf połączeń węzłów.

Aby uniknąć cyklów w realizowaniu zadań rozproszonych – zadaniom są nadawane unikalne `TASK_ID`, w postaci `TASK_ID:nodeTaskIdCounter:adres_węzła:tcpport`. W ten sposób węzły mogą zwrócić wiadomość zwrotną`"ERROR"` w przypadku napotkania prośby o realizacje tego samego zadania (np. `srv__get-min`).

Przesyłane komunikaty w komunikacji klient-węzeł

Komunikaty są takie jak opisane w treści zadania. Komunikacja odpywa się poprzez protokół TCP, przy użyciu klas `ServerSocket` oraz `Socket` z biblioteki `java.net`.

Przesyłane komunikaty w komunikacji węzeł-węzeł (definicja protokołu)

Wszystkie komunikaty węzeł-węzeł zawierają parametr `<myAddress:myTcpPort>` – przekazuje on adres oraz port nasłuchiwania węzła nadającego, służy on do jednoznaczej identyfikacji węzła nadającego oraz aby węzeł zapamiętał jego adres oraz port nasłuchiwania w przypadku polecenia `srv__connect`. Unikalny `<taskId>` jest nadawany w niektórych komunikatach, aby uniknąć nieskończonego oczekiwania w przypadku gdy graf węzłów zawiera cykl.

Pełna lista komunikatów węzeł-węzeł:

- `srv__connect <myAddress:myTcpPort>` – Nawiązanie połączenia z nowym węzłem który odpowiada `"OK"`. Jako parametr są podane adres oraz port serwera ubiegającego o połączenie.
- `srv__disconnect <myAddress:myTcpPort>` – Informacja o terminacji siebie – oczekiwana odpowiedź to `"OK"`.
- `srv__get-value <taskId> <key> <myAddress:myTcpPort>` – Realizacja operacji `get-value` (części związanej z komunikacją węzeł-węzeł) opisanej w zadaniu. Parametr `key` odpowiada parametrowi komunikatu `get-value` klienta. Oczekiwana odpowiedź to `"ERROR"` w przypadku

wykrycia cyklu komunikatów dla danego `<taskID>` lub w przypadku nie odnalezienia danego klucza w sieci węzłów.

- `srv__set-value <taskId> <key:value> <myAddress:myTcpPort>` – analogicznie jak `srv__get-value`, tylko realizuje komunikat `set-value` jak opisane poleceniu. Odpowiedź to szukane `adres:tcpport` lub `"ERROR"`.
- `srv__find-key <taskId> <key> <myAddress:myTcpPort>` – analogicznie, z tym że realizuje komunikat `find-key` jak opisane w poleceniu.
- `srv__get-min <taskId> <myAddress:myTcpPort>` – analogicznie, z tym że ma tylko 2 argumenty jako że `get-min` nie ma dodatkowego argumentu.
- `srv__get-max <taskId> <myAddress:myTcpPort>` – analogicznie jak `srv__get-min`.