Opis protokołu do obsługi serwisu cytatów-sentencji

Wojciech Żółtak (wz292583)

24 maja 2011

Niniejszy dokument opisuje specyfikację protokołu do serwisu cytatów-sentencji.

1 Cele

Głównym celem protokołu jest przesyłanie losowo wybranych cytatów z bazy danych do aplikacji klienckiej.

2 Założenia

- Opisywany protokół działa w **warstwie aplikacji** na zasadach paradygmatu **klient-serwer**.
- Umożliwia przesyłanie **losowo wybranych** cytatów z bazy danych udostępnionej przez serwer.
- Zakładamy działanie w środowisku, w którym działanie w ramach określonego okna czasowego jest istotniejsze od pewności transmisji, tzn. przedkłada prędkość obsługi żądania klienta nad gwarancję dostarczenia do niego wiadomości. Z tego powodu komunikacja będzie odbywać się za pośrednictwem **UDP**.
- Znakomita większość cytatów wysyłanych do klienta nie będzie zbyt duża (powinna zmieścić się w jednym komunikacie), co nie znaczy, że protokół nie będzie umożliwiał przesyłania dłuższych sentencji.
- Serwer dysponuje bazą danych cytatów (pozostawioną w gestii implementacji), która pozwala jednoznacznie zidentyfikować każdy cytat za pomocą 4 bajtowego identyfikatora.

3 Format komunikatów

3.1 Założenia

- liczby zapisywane są w porządku sieciowym
- ciągi znaków kodowane są w standardzie UTF-8, z UNIXowymi znakami końca linii, bez znaku zero na końcu

3.2 Żądanie cytatu

```
REQ_MSG_T {
    uint8_t type; /* := RANDOM_MSG */
}
```

```
REQ_MSG_CHUNK_T {
    uint8_t  type; /* := SPECIFIC_MSG */
    uint32_t q_id;
    uint8_t  q_chunk;
}
```

Opis pól:

- type - determinuje typ komunikatu, `RANDOM_MSG` dla `REQ_MSG`, `SPECIFIC_MSG` dla `REQ_MSG_CHUNK`
- `q_id` - liczbowy identyfikator cytatu
- `q_chunk` - numer części cytatu

3.3 Odpowiedź serwera

```
RESP_MSG_T {
    uint32_t q_id;
    uint8_t  q_size;
    uint8_t  q_chunk;
    uint8_t  q_len;
    char     q_con[MAX_LENGTH];
}
```

Opis pól:

- `q_id` - liczbowy identyfikator cytatu
- `q_size` - rozmiar cytatu w komunikatach
- `q_chunk` - numer części cytatu, **numerowane od 1**
- `q_len` - długość przesyłanego tekstu w bajtach (pola `q_con`), nie większa niż `MAX_LENGTH`
- `q_con` - treść przesyłanego cytatu (bądź jego fragment)

4 Wymiana komunikatów

4.1 Założenia

- Serwer działa na pojedynczym, określonym porcie, którego numer znany jest klientowi.
- Klient wysyła żądania na znany port serwera, korzystając z dowolnego portu.
- Serwer wysyła odpowiedź na adres nadawcy żądania, uzyskany podczas odbierania wiadomości.

4.2 Stany serwera

Serwer działa bezstanowo. W każdym momencie przyjmuje żądanie od klienta i odsyła mu odpowiedź w postaci jednej lub więcej struktur typu `RESP_MSG`.

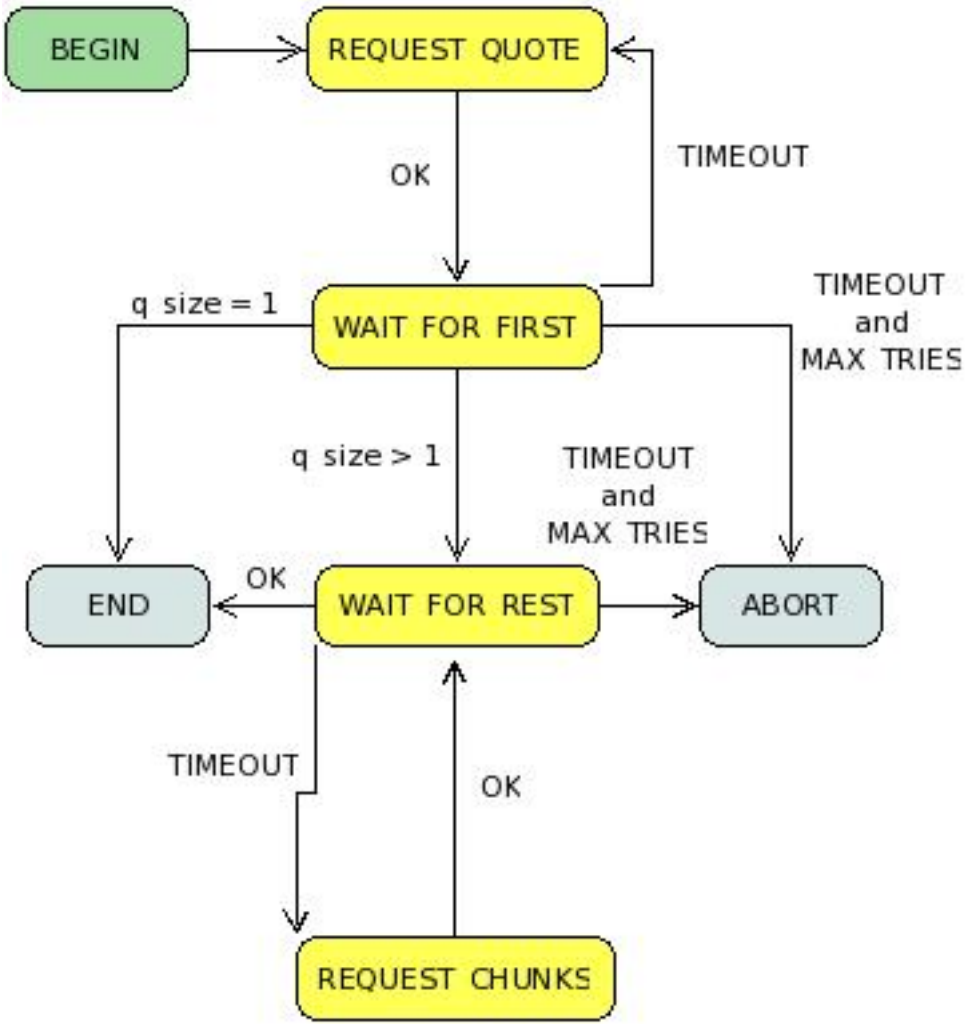
4.3 Stany klienta

Klient może znajdować się w następujących stanach:

- `BEGIN` - stan wejściowy. Nie posiadamy żadnych wiadomości.
- `REQUEST_QUOTE` - wysyła do serwera zapytanie o losowy cytat, w postaci struktury `REQ_MSG` z ustawionym polem `type` na `RANDOM_MSG`.
- `WAIT_FOR_FIRST` - oczekiwanie na pierwszą wiadomość od serwera, w postaci dowolnej struktury `RESP_MSG`. Jeśli otrzymana wiadomość zawiera kompletny cytat (pole `q_size` zawiera 1), klient przechodzi do stanu końcowego `END`. Jeśli wiadomość jest tylko fragmentem cytatu (pole `q_size` zawiera liczbę większą niż 1), klient zeruje licznik prób i przechodzi do stanu `WAIT_FOR_REST`. W przypadku gdy nastąpił `TIMEOUT`, ale nie `MAX_TRIES`, klient ponawia prośbę wiadomość i powraca do stanu `REQUEST_QUOTE`. W przypadku, gdy przekroczono limit prób i wciąż nie otrzymano wiadomości, klient przechodzi do stanu `ABORT`.
- `WAIT_FOR_REST` - oczekiwanie na pozostałe wiadomości od serwera w postaci struktur `RESP_MSG` zawierających takie samo `q_id` jak wiadomość otrzymana w stanie `REQUEST_QUOTE` oraz `q_chunk` z zakresu $[1, \texttt{q\_size}]$. Jeśli klient odbierze wiadomości dla wszystkich poprawnych `q_chunk` przechodzi do stanu `END`. W przypadku gdy nastąpił `TIMEOUT`, ale nie `MAX_TRIES`, klient stara się uzyskać brakujące części cytatu przechodząc do stanu `REQUEST_CHUNKS`. W przypadku gdy przekroczono limit prób i wciąż nie uzyskano kompletu wiadomości, klient przechodzi w stan `ABORT`.
- `REQUEST_CHUNKS` - wysyła do serwera zapytanie o brakujące części w postaci struktur `REQ_MSG_CHUNK` (po jednej dla każdego brakującego fragmentu) - wypełniając pole `type` na `SPECIFIC_MSG`, `q_id` na wartość otrzymaną w stanie `WAIT_FOR_FIRST` oraz `q_chunk` kolejno na wszystkie wartości, których nie otrzymano w stanie `WAIT_FOR_REST`. Następnie, klient wraca do stanu `WAIT_FOR_REST`.
- `ABORT` - operacja z jakichś powodów zakończyła się niepowodzeniem i nie będzie dalszych prób jej realizacji.
- `END` - stan wyjściowy. Klient posiada wszystkie potrzebne kawałki cytatu, w postaci pól `q_con`, w otrzymanych strukturach `RESP_MSG`,

które złącza w jeden ciąg znaków według porządku po zawartości pól `q_chunk`.

Poniższy schemat opisuje przejścia między stanami:



5 Używane stałe i etykiety

- `RANDOM_MSG` := 0
- `SPECIFIC_MSG` := 1
- `MAX_LENGTH` - maksymalna długość tekstu przesyłanego w jednym komunikacie; protokół nie wymusza konkretnej wartości, lecz sugerowane są nie za duże liczby (w okolicach 1000, np. 1018 octetów, tak żeby cały komunikat miał ich 1024)

- `TIMEOUT` - zdarzenie przekroczenia limitu oczekiwania na odpowiedź serwera; długość oczekiwania w gestii implementacji
- `MAX TRIES` - zdarzenie przekroczenia ilości prób pobierania brakujących fragmentów wiadomości; ilość prób w gestii implementacji, może być różna dla różnych stanów (np. stan `WAIT_FOR_REST` może pozwalać na większą ilość prób, niż `WAIT_FOR_FIRST`)