

Protokół obsługi cytatów.
Sieci Komputerowe 2010/2011
Wersja 1.0

Mateusz Litwin

28 kwietnia 2011

Spis treści

1	Streszczenie	3
2	Cele protokołu	3
3	Założenia protokołu	3
3.1	Wersja protokołu	3
3.2	Warstwa działania protokołu	3
3.3	Protokół transportu	3
3.4	Model komunikacyjny	3
4	Format komunikatów	4
4.1	Porządek oktetów	4
4.2	Sposób interpretacji znaków	4
4.3	Kodowanie tekstu	4
4.4	Ogólna postać komunikatów	4
5	Opis komunikatów	5
5.1	Wiadomości od klienta	5
5.1.1	Prośba o podanie wersji	5
5.1.2	Prośba o przesłanie cytatu	5
5.2	Wiadomości od serwera	6
5.2.1	Wysyłanie wersji	6
5.2.2	Typowy cytat	6
5.2.3	Długi cytat	6
6	Opis stanów	7
6.1	Legenda	7
6.1.1	Oznaczenia występujące w diagramach	7
6.2	Opis stanów po stronie klienta - pobieranie wersji protokołu	8
6.2.1	Ogólny plan działania	8
6.2.2	Diagram stanów po stronie klienta - pobieranie wersji protokołu	8
6.3	Opis stanów po stronie klienta - pobieranie cytatu	8
6.3.1	Terminy występujące w opisie stanów	8
6.3.2	Ogólny plan działania	9
6.3.3	Uwaga o kolizjach kluczy komunikatów	9
6.3.4	Uwaga o niepoprawnych i nieoczekiwanych komunikatach	9
6.3.5	Uwaga o przekroczeniu czasu oczekiwania	9
6.3.6	Uwaga o asynchroniczności procesu pobierania cytatu	9
6.3.7	Uwaga o pobieraniu wielu cytatów	10
6.3.8	Uwaga o zakleszczeniach i wyścigach	10
6.3.9	Diagram stanów po stronie klienta - pobieranie cytatu	10
6.4	Opis stanów po stronie serwera	11
6.4.1	Terminy występujące w opisie stanów	11
6.4.2	Ogólny plan działania	11
6.4.3	Uwaga o niepoprawnych komunikatach	11
6.4.4	Uwaga o asynchroniczności procesu realizacji żądań	11
6.4.5	Uwaga o zakleszczeniach i wyścigach	11
6.4.6	Diagram stanów po stronie serwera	12
7	Wykaz stałych liczbowych	13

1 Streszczenie

Niniejszy dokument zawiera opis protokołu służącego do obsługi serwisu cytatów-sentencji. Na podstawie tej dokumentacji możliwa jest pełna implementacja tego protokołu.

2 Cele protokołu

Celem protokołu jest umożliwienie przesyłania losowo wybranych cytatów. Dzięki protokołowi możliwe będzie na przykład wyświetlanie losowych sentencji na stronie WWW klienta. Komunikacja odbywać się będzie według paradygmatu klient-serwer. Klient zgłasza prośbę do serwera o przesłanie losowego cytatu, a następnie serwer wysyła tekst.

3 Założenia protokołu

3.1 Wersja protokołu

Ten protokół jest w wersji 1.0.

3.2 Warstwa działania protokołu

Protokół działa w warstwie aplikacji.

3.3 Protokół transportu

Protokół będzie korzystał z UDP (zalecany port 23456).

Wybór ten jest uwarunkowany specyfikacją problemu:

- w typowym przypadku wysyłane cytaty będą krótkie i będą mieściły się w jednym komunikacie, np. nie będzie istotna kolejność przesyłanych komunikatów,
- obsługa cytatów nie jest kluczowa z punktu widzenia użytkownika, dlatego należy postawić na szybkość kosztem stabilności usługi.

3.4 Model komunikacyjny

Specyfikacja problemu narzuca model komunikacyjny klient-serwer.

4 Format komunikatów

4.1 Porządek oktetów

Porządek sieciowy.

4.2 Sposób interpretacji znaków

Brak - w protokole nie występują liczby ze znakiem.

4.3 Kodowanie tekstu

Łańcuchy znaków kodowane są za pomocą ISO 8859-2 (Latin-2).

4.4 Ogólna postać komunikatów

Wszystkie komunikaty zawierać będą pole `msg_type` typu `uint16`. Pole to występować będzie na początku komunikatu i będzie informować z jakim typem komunikatu mamy do czynienia. Dalej występować będą inne pola w zależności od typu komunikatu.

Pole-klucz `key` ustalamy, aby móc odróżniać komunikaty dotyczące naszego żądania od innych komunikatów. Jest to konieczne, ponieważ używamy protokołu UDP i możemy otrzymywać w dowolnej kolejności komunikaty dotyczące różnych żądań.

5 Opis komunikatów

5.1 Wiadomości od klienta

5.1.1 Prośba o podanie wersji

```
VERSION_REQUEST_MSG {  
    uint16 msg_type;  
    uint32 key;  
}
```

msg_type - typ wiadomości, ma wartość równą `VERSION_REQUEST_TYPE`;

key - klucz będący identyfikatorem żądania;

Zapytanie wysyłane do serwera z prośbą o podanie obsługiwanej wersji protokołu.

5.1.2 Prośba o przesłanie cytatu

```
QUOTE_REQUEST_MSG {  
    uint16 msg_type;  
    uint32 key;  
}
```

msg_type - typ wiadomości, ma wartość `QUOTE_REQUEST_TYPE`;

key - klucz będący identyfikatorem żądania `QUOTE_REQUEST_MSG`, kluczem będzie losowa, 32 bitowa liczba bez znaku;

Komunikat wysyłamy do serwera, aby otrzymać komunikat/komunikaty z cytatem.

5.2 Wiadomości od serwera

5.2.1 Wysyłanie wersji

```
VERSION_MSG {  
    uint16 msg_type;  
    uint32 key;  
}
```

msg_type - typ wiadomości, ma wartość większą równą `MIN_VERSION`;

key - klucz będący identyfikatorem żądania, na które `VERSION_MSG` jest odpowiedzią;

Odpowiedź serwera na zapytanie o obsługiwaną przez niego wersję protokołu.

Stała `MIN_VERSION` wynosi 1000.

Wersje protokołu mają postać: `"(msg_type / 1000).(msg_type mod 1000)"`.

Na przykład wersja 2.203 jest zapisana w polu `msg_type` jako 2203.

5.2.2 Typowy cytat

```
QUOTE_MSG {  
    uint16 msg_type;  
    uint32 key;  
    uint16 text_length;  
    uint8[text_length] text;  
}
```

msg_type - typ wiadomości, ma wartość `QUOTE_TYPE`;

key - klucz będący identyfikatorem żądania, na które `QUOTE_MSG` jest odpowiedzią;

text_length - pole zawierające długość pola `text`;

text - treść cytatu;

Cytaty o długości mniejszej lub równej `MAX_LENGTH` są wysyłane komunikatem `QUOTE_MSG`.

5.2.3 Długi cytat

```
LONG_QUOTE_MSG {  
    uint16 msg_type;  
    uint32 key;  
    uint16 parts_number;  
    uint16 part_id;  
    uint16 text_length;  
    uint8[text_length] text;  
}
```

msg_type - typ wiadomości, ma wartość `LONG_QUOTE_TYPE`;

key - klucz będący identyfikatorem żądania, na które `LONG_QUOTE_MSG` jest odpowiedzią;

parts_number - liczba komunikatów, w których został wysłany cytat;

part_id - numer danego fragmentu, liczba z przedziału `[0, parts_number - 1]`;

text_length - pole zawierające długość pola `text`;

text - fragment cytatu;

Jeżeli cytat długości N ma więcej znaków niż MAX_LENGTH , to zostanie podzielony na $\lceil N/MAX_LENGTH \rceil$ części. Części numerujemy kolejnymi liczbami całkowitymi, zaczynając od 0.

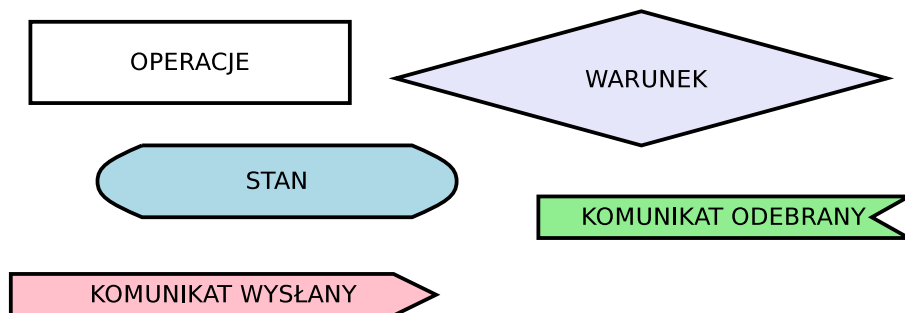
Część numer 'i' cytatu długości N , wysłanego w odpowiedzi na żądanie `QUOTE_REQUEST_MSG` o polu `key = K`, zostanie przesłana komunikatem postaci:

```
LONG_QUOTE_MSG {
    uint16 msg_type = QUOTE_TYPE;
    uint32 key = K;
    uint16 parts_number = (N + MAX_LENGTH - 1) / MAX_LENGTH;
    uint16 part_id = i;
    uint16 text_length = min(N - i * MAX_LENGTH, MAX_LENGTH);
    uint8[text_length] text = fragment cytatu od znaku numer
        (i * MAX_LENGTH) do (min((i + 1) * MAX_LENGTH, N) - 1), włącznie;
}
```

6 Opis stanów

6.1 Legenda

6.1.1 Oznaczenia występujące w diagramach



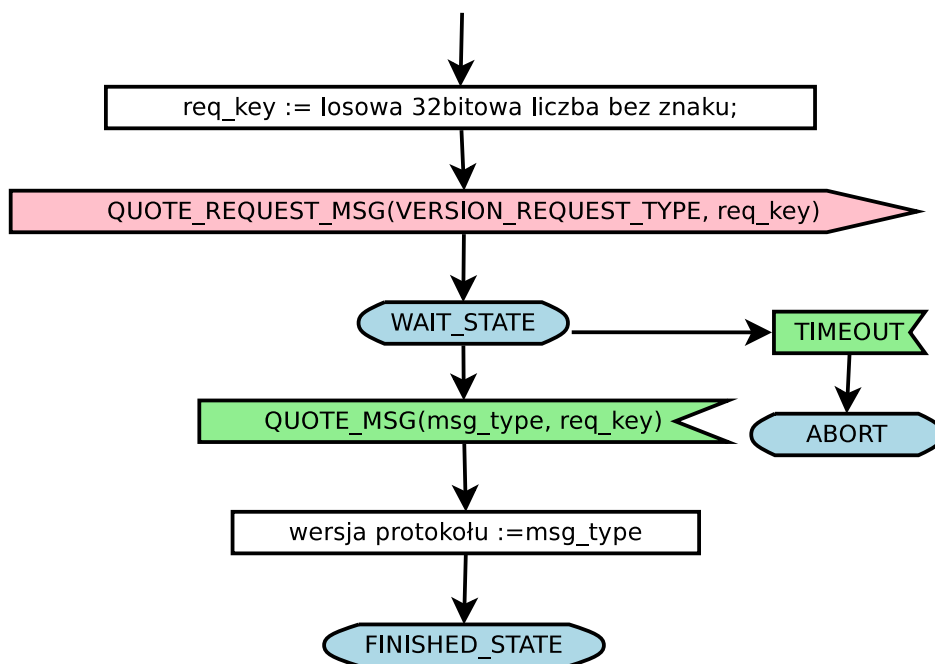
Diagramy nie mają zaznaczonych sytuacji związanych z odebraniem nieprzewidzianego/niepoprawnego komunikatu.

6.2 Opis stanów po stronie klienta - pobieranie wersji protokołu

6.2.1 Ogólny plan działania

1. Klient przygotowuje losowy klucz.
2. Klient wysyła żądanie o podanie wersji - VERSION_REQUEST_MSG.
3. Klient oczekuje na komunikat VERSION_MSG (WAIT_STATE).
4. Klient odbiera komunikat i kończy pobieranie wersji (FINISHED_STATE) albo w ciągu TIMEOUT milisekund nie otrzymuje oczekiwanego komunikatu i kończy działanie (ABORT).

6.2.2 Diagram stanów po stronie klienta - pobieranie wersji protokołu



6.3 Opis stanów po stronie klienta - pobieranie cytatu

6.3.1 Terminy występujące w opisie stanów

- `QUOTE_REQUEST_MSG(req_key)` - komunikat `QUOTE_REQUEST_MSG` o polu `key` równym `req_key`;
- `LONG_QUOTE_MSG(rcv_key, rcv_parts_number)` - komunikat `LONG_QUOTE_MSG` o polach `key` równym `rcv_key` i `parts_number` równym `rcv_parts_number`;
- `QUOTE_MSG(rcv_key)` - komunikat `QUOTE_MSG` o polu `key` równym `rcv_key`;

6.3.2 Ogólny plan działania

1. Klient wysyła żądanie o losowo wygenerowanym kluczu `req_key`.
2. Następnie oczekuje na odpowiedź serwera (`WAIT_STATE`).
3. Komunikaty o innym kluczu niż `req_key` są ignorowane.
4. Jeżeli w ciągu `TIMEOUT` milisekund nie zakończymy pomyślnie działania to anulujemy żądanie (`ABORT`).
5. Jeżeli odbieramy komunikat `QUOTE_MSG` o polu `key = req_key` to kończymy (`FINISHED_STATE`).
6. Jeżeli odbierzemy wszystkie `rcv_parts_number` komunikaty `LONG_QUOTE_MSG` o polu `key = req_key` to kończymy (`FINISHED_STATE`).

6.3.3 Uwaga o kolizjach kluczy komunikatów

W związku z użyciem UDP możliwa jest sytuacja, że klient otrzyma komunikaty dotyczące starych żądań o tym samym kluczu co aktualne żądanie. Sytuacja ta jest mało prawdopodobna z powodu używania losowych kluczy. Protokół uważa za dopuszczalne pomieszanie różnych cytatów o tych samych kluczach, dotyczy to także długich cytatów.

6.3.4 Uwaga o niepoprawnych i nieoczekiwanych komunikatach

Wszelkie nieprzewidziane w danym stanie komunikaty należy ignorować. Za niepoprawne komunikaty uznaje się m.in:

- komunikat `LONG_QUOTE_MSG` o polu `part_id` niemniejszym niż `parts_number`,
- komunikat, którego wartość pola `msg_type` jest różna od przewidzianego przez protokół,
- zduplikowany komunikat `LONG_QUOTE_MSG`.

6.3.5 Uwaga o przekroczeniu czasu oczekiwania

Jeżeli w ciągu `TIMEOUT` milisekund od momentu przejścia w stan `WAIT_STATE` klient nie otrzyma wszystkich oczekiwanych komunikatów, to przechodzi w stan `ABORT`.

6.3.6 Uwaga o asynchroniczności procesu pobierania cytatu

Pobieranie cytatu interpretujemy jako kolejne kroki, które wykonuje klient począwszy od `INITIALIZED_STATE`, aż po `FINISHED_STATE`. Faza odbioru, jak i wysyłania komunikatów są niezależne. Specyfikacja protokołu umożliwia klientowi wielokrotne i jednoczesne pobieranie cytatów.

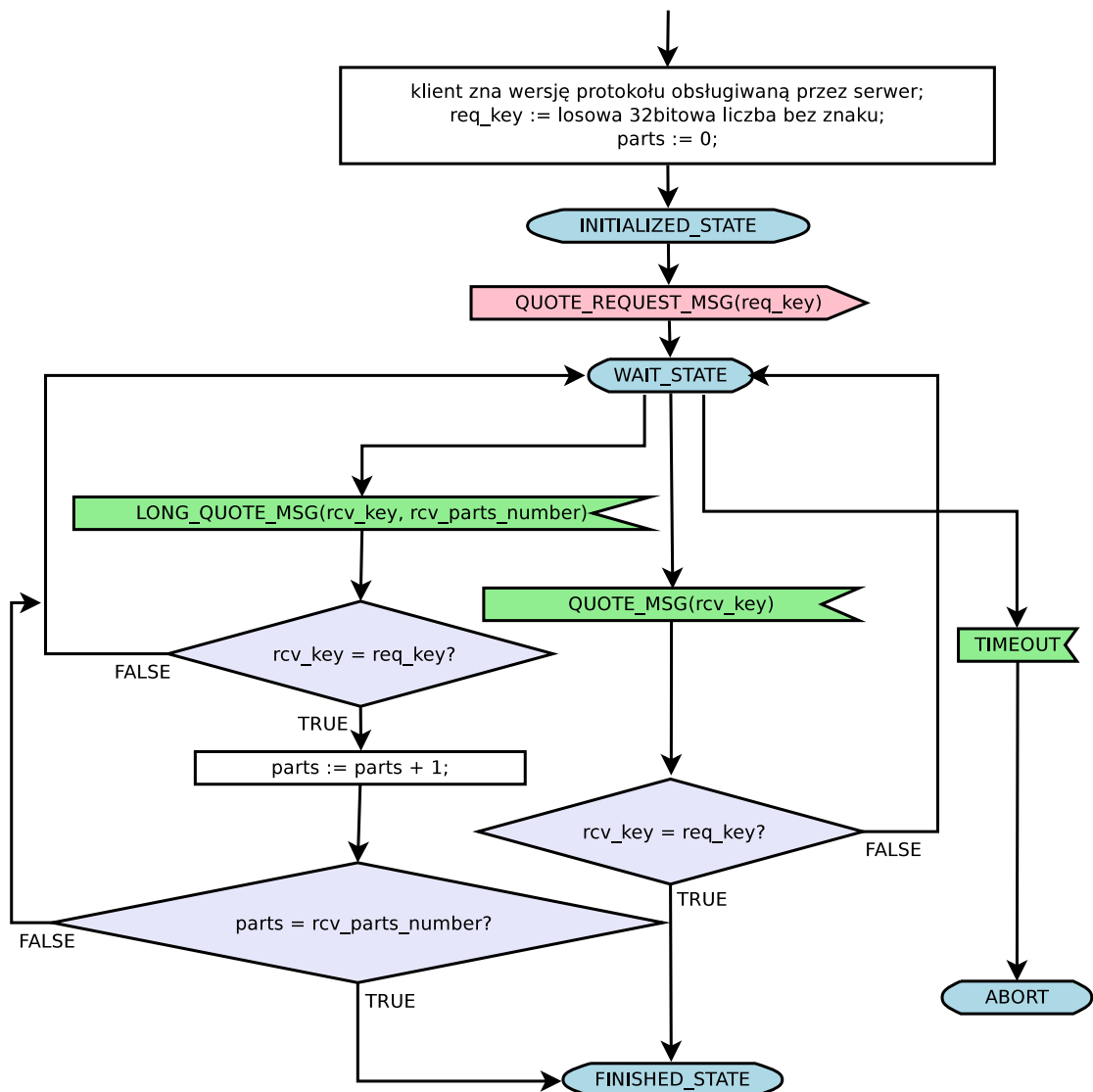
6.3.7 Uwaga o pobieraniu wielu cytatów

Pobieranie wielu cytatów zalecane jest poprzez wielokrotną realizację żądania o pojedynczy cytat.

6.3.8 Uwaga o zakleszczeniach i wyścigach

Poprawna implementacja protokołu zapewnia brak zakleszczeń i wyścigów.

6.3.9 Diagram stanów po stronie klienta - pobieranie cytatu



6.4 Opis stanów po stronie serwera

6.4.1 Terminy występujące w opisie stanów

- `QUOTE_REQUEST_MSG(req_key)` - komunikat `QUOTE_REQUEST_MSG` o polu `key` równym `req_key`;
- `QUOTE_MSG(req_key, quote_len, quote)` - komunikat `QUOTE_MSG` o polach `key`, `text_length`, `text` równych odpowiednio `req_key`, `quote_len`, `quote`;
- `LONG_QUOTE_MSG(req_key, parts_number, i, quote_len[i], quote[i])` - komunikat typu `LONG_QUOTE_MSG`, którego pola `key`, `parts_number`, `part_id`, `text_length`, `text` wynoszą odpowiednio `req_key`, `parts_number`, `i`, `quote_len[i]`, `quote[i]`. Dokładny opis zawartości komunikatu `LONG_QUOTE_MSG` znajduje się w punkcie 5.2.2;

6.4.2 Ogólny plan działania

1. Serwer inicjuje się, po czym przechodzi w stan `WAIT_STATE`.
2. Serwer czeka na komunikat `QUOTE_REQUEST_MSG` lub `VERSION_REQUEST_MSG`.
3. Jeżeli serwer odebrał `QUOTE_REQUEST_MSG`, to losuje cytat i w zależności od jego długości wysyła go pojedynczym komunikatem `QUOTE_MSG` lub paroma komunikatami typu `LONG_QUOTE_MSG`.
4. Jeżeli serwer odebrał `VERSION_REQUEST_MSG`, to wysyła pojedynczy komunikat typu `VERSION_MSG`.
5. Po wysłaniu komunikatu wraca do stanu `WAIT_STATE`.

6.4.3 Uwaga o niepoprawnych komunikatach

Podobnie jak po stronie klienta, także serwer powinien ignorować niepoprawne komunikaty.

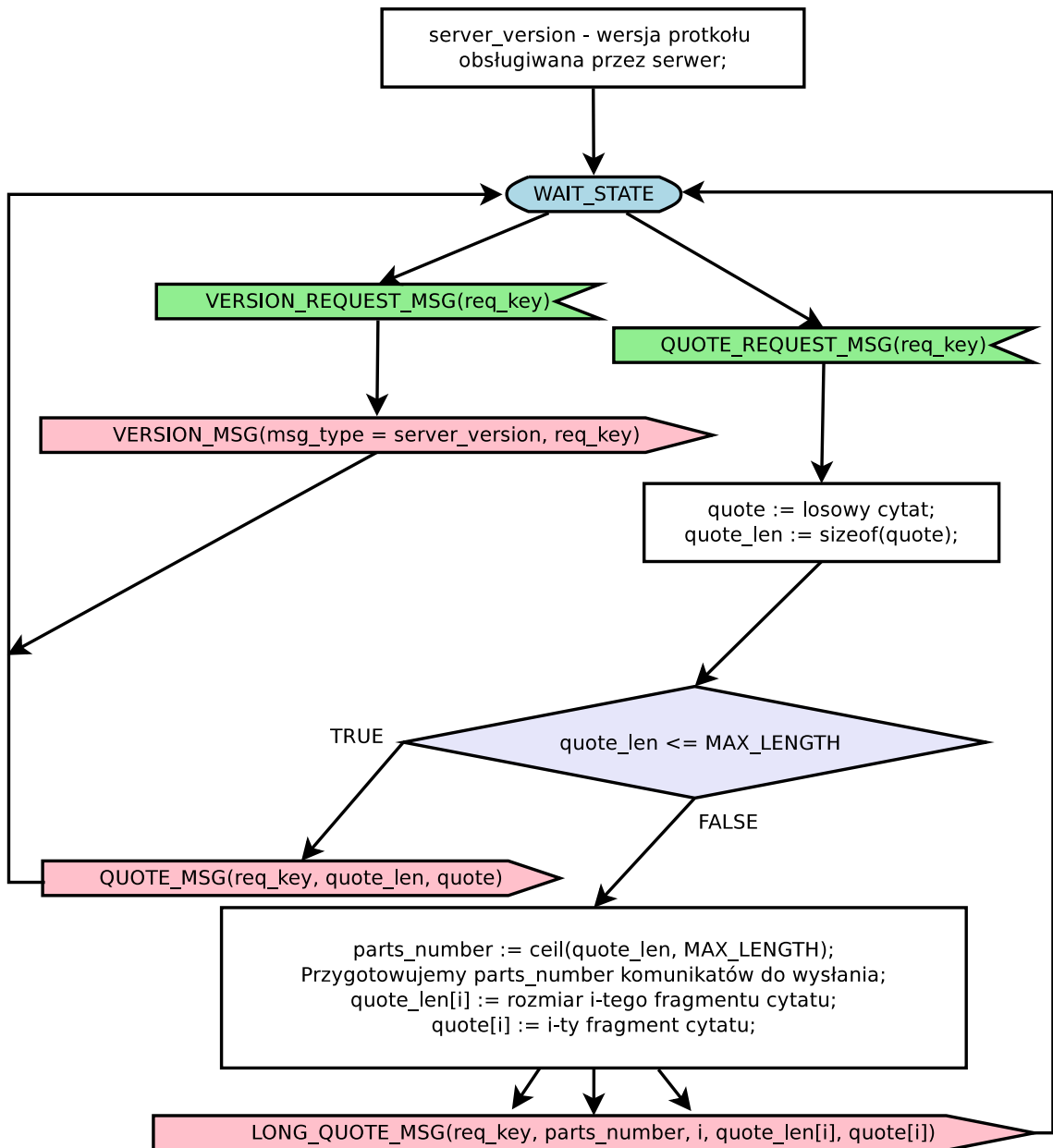
6.4.4 Uwaga o asynchroniczności procesu realizacji żądań

Realizowanie żądania interpretujemy jako kolejne kroki, które wykonuje serwer począwszy od odbioru żądania w stanie `WAIT_STATE`, aż po zakończenie wysyłania komunikatu do klienta. Specyfikacja protokołu umożliwia serwerowi jednoczesną realizację wielu żądań od dowolnych klientów.

6.4.5 Uwaga o zakleszczeniach i wyścigach

Poprawna implementacja protokołu zapewnia brak zakleszczeń i wyścigów.

6.4.6 Diagram stanów po stronie serwera



7 Wykaz stałych liczbowych

Stałe liczbowe użyte w dokumentacji:

- MAX_LENGTH = 1024
- MIN_VERSION = 1000
- VERSION_REQUEST_TYPE = 0
- QUOTE_REQUEST_TYPE = 1
- QUOTE_TYPE = 2
- LONG_QUOTE_TYPE = 3
- TIMEOUT = 5000
- ZALECANY PORT = 23456
- WERSJA PROTOKOŁU = 1.0