

== Opis protokołu do obsługi aukcji internetowych ==

== wersja: 1.01 ==

Autor: Tomasz Bartosiak

== Streszczenie ==

Niniejszy dokument opisuje budowę protokołu do obsługi aukcji internetowych. Ma on na celu ukazanie celów, jaki ten protokół ma realizować oraz założeń, na jakich bazuje.

Prezentuje on także komunikaty, jakie wysyłają między sobą dwa programy, wykorzystujące ten protokół, oraz w jaki sposób mają być one interpretowane.

Opis protokołu ma być podstawą do implementacji tegoż protokołu.

== Cele ==

Protokół ma na celu obsługę rozproszonych aukcji internetowych. Ma on umożliwiać wystawianie przedmiotów na aukcji, oraz licytowanie przedmiotów wystawianych do licytacji przez innych użytkowników.

Ma on także umożliwić zapoznanie się z przedmiotami, które dany użytkownik wystawił do licytacji.

== Założenia ==

Zakładamy, że protokół działa w warstwie sesji.

Zakładamy, że domyślnym numerem portu dla protokołu będzie 8800.

Zakładamy, że podczas uruchamiania aplikacji będzie możliwość zmiany domyślnego numeru portu.

Zakładamy, że do przeglądania aukcji wykorzystywany będzie protokół bezpołączeniowy UDP.

Zakładamy, że pliki przesyłane będą za pomocą protokołu połączeniowego TCP.

Zakładamy, że typ pliku jest zgodny z rozszerzeniem w przesyłanej nazwie pliku.

Zakładamy, że do komunikatów realizujących drugą fazę (licytacja), wykorzystany będzie protokół połączeniowy TCP, ze względu na jego niezawodność.

Zakładamy, że każdy komunikat z fazy drugiej wymaga potwierdzenia.

Zakładamy, że połączenia TCP są zrywane po przesłaniu lub otrzymaniu potwierdzenia, bądź po przekroczeniu pewnego progu czasowego.

Zakładamy, że protokół działa bez centralnego serwera, w trybie P2P. Wszelka komunikacja odbywa się bezpośrednio między licytującymi a wystawiającym.

Zakładamy, że każdy przedmiot można licytować od momentu wystawienia go.

Zakładamy, że przy wystawianiu przedmiotu do licytacji podawany jest ostateczny termin końca licytacji. Licytacja może zostać zakończona wcześniej przez wystawiającego.

Zakładamy, że każdy może wystawić maksymalnie MAX_ITEMS przedmiotów do licytacji na raz.

Zakładamy, że każdy licytujący może dołączyć do każdej aukcji, ale tylko raz.

Zakładamy, że każda aukcja ma swój unikalny identyfikator.

Zakładamy, że każdy uczestnik aukcji ma swój unikalny identyfikator, jednoznacznie określający wystawiającego, licytującego i licytowany przedmiot.

== Unikalne identyfikatory ==

W drugiej fazie, protokół ma wykorzystywać unikalne identyfikatory dla aukcji i licytującego. Do konstrukcji tych identyfikatorów będziemy wykorzystywać dane o adresie karty sieciowej wystawiającego (dla identyfikatora aukcji i identyfikatora licytującego) oraz o adresie karty sieciowej licytującego (dla identyfikatora licytującego).

Identyfikator aukcji zajmuje AUCT_ID_LEN oktetów.

- * pierwsze MAC_LEN oktetów to adres karty sieciowej wystawiającego,
- * kolejne PORT_LEN oktetów to numer portu który zajmuje wystawiający,
- * a pozostałe oktety to numer identyfikujący przedmiot u wystawiającego.

Identyfikator licytującego zajmuje BID_ID_LEN oktetów.

- * pierwsze AUCT_ID_LEN oktetów to identyfikator aukcji,
- * kolejne MAC_LEN oktetów to adres karty sieciowej licytującego,
- * a ostatnie PORT_LEN oktetów to numer portu zajmowany przez licytującego.

== Daty ==

Ponieważ aukcje mają określony ostateczny termin zakończenia, należy poinformować o nim licytujących. Podczas przesyłania wszelkich informacji na temat terminów przyjmujemy następujący format dat:

- * daty przekazywane są jako cztero-elemenowa tablica liczb uint8.
- * pierwsze 3 liczby odpowiadają odpowiednio dniu, miesiącu i roku zakończenia aukcji
- * czwarta liczba mówi, w której minucie dnia aukcja zostanie (najpóźniej) zamknięta.

== Wersje ==

Realizacja protokołu o dowolnym numerze wersji powinna być w stanie komunikować się ze wszystkimi realizacjami wcześniejszych wersji protokołu. We wszystkich komunikatach znajduje się pole version jednoznacznie rozpoznające wersję protokołu. W wypadku komunikacji między dwoma protokołami o różnych wersjach, za standard uznaje się format komunikatów z wcześniejszej wersji.

Numery wersji przesyłane są jako dwuelementowa tablica liczb.

== Format komunikatów ==

Zakładamy sieciowy porządek oktetów we wszystkich liczbach. Wszelkie napisy uzupełniamy zerami.

Występują 3 podstawowe typy komunikatów:

- * Komunikaty fazy pierwszej przesyłane za pomocą UDP


```
FASE_ONE_MSG {
    uint8[2]                version;
                        // numer wersji protokołu
    uint8                  message_type;
                        // typ wiadomości
    uint8                  text_size;
                        // długość napisu text
    octet[text_size]      text;
                        // napis (np. opis przedmiotu)
    uint8                  it_count;
                        //ilość przedmiotów, których dotyczy wiadomość
```

```

octet[it_count][MAX_NAME]      items;
        // nazwy przedmiotów, których dotyczy komunikat
octet[it_count][AUCTION_ID_LEN] ids;
        // identyfikatory aukcji
uint16[it_count]                values;
        //obecne ceny przedmiotów
uint8[it_count][4]              expires;
        // czas planowanego końca licytacji
}

```

* Komunikat fazy pierwszej zawierający plik lub prośbę o przesłanie pliku

```

FILE_MSG {
    uint8[2]                version;
        // numer wersji protokołu
    uint8                  message_type;
        //prośba o plik lub plik
    octet[AUCTION_ID_LEN]  id;
        // identyfikator przedmiotu, którego dotyczy komunikat
    octet[MAX_NAME]        item;
        // nazwa tego przedmiotu
    octet[MAX_FILE_NAME]   file_name;
        // nazwa pliku wraz z rozszerzeniem
    uint64                 file_size;
        // rozmiar pliku (w oktetach)
    octet[file_size]       file;
        // plik
}

```

* Komunikaty fazy drugiej (licytacja)

```

FASE_TWO_MSG {
    uint8[2]                version;
        // numer wersji protokołu
    uint8                  message_type;
        // typ wiadomości
    uint16                 value;
        // wartość (proponowana lub najwyższa)
    octet[BID_ID_LEN]      bidder_id;
        //identyfikator licytującego
}

```

== Opis komunikatów ==

Ponieważ jest to pierwsza wersja protokołu, we wszystkich komunikatach należy ustawić wartość pola version:

```

* version[0] = 1
* version[1] = 1

```

We wszystkich komunikatach pomijane są tablice zero-elementowe.

Komunikaty fazy pierwszej typu FASE_ONE_MSG:

```

* ITEMS_Q_MSG - komunikat pytający o przedmioty wystawione przez użytkownika
na sprzedaż. Ustawione pola:
    message_type = ITEMS_Q_MSG
    text_size = 0
    it_count = 0

```

- * ITEMS_A_MSG - odpowiedź na komunikat ITEMS_Q_MSG. Ustawione pola:
 - message_type = ITEMS_A_MSG
 - text_size = 0
 - it_count = *ilość przedmiotów wystawionych na aukcję*
 - items = *tablica nazw tych przedmiotów*
 - ids = *tablica identyfikatorów aukcji*
 - values = *tablica aktualnych najwyższych cen*
 - expires = *tablica planowanych momentów końca aukcji*przy czym pola tablic o tym samym numerze odpowiadają tym samym przedmiotom.

- * PROPERTIES_Q_MSG - pytanie o własności przedmiotu. Ustawione pola:
 - message_type = PROPERTIES_Q_MSG
 - text_size = 0
 - it_count = 1
 - items[0] = '\0' (nieistotne)
 - ids[0] = *identyfikator aukcji*
 - values[0] = 0 (nieistotne)
 - expires[0] = 0 (nieistotne)

- * PROPERTIES_A_MSG - odpowiedź na komunikat PROPERTIES_Q_MSG. Ustawione pola:
 - message_type = PROPERTIES_A_MSG
 - text_size = *długość opisu przedmiotu*
 - text = *opis przedmiotu*
 - it_count = 1
 - items[0] = *nazwa przedmiotu*
 - ids[0] = *identyfikator aukcji*
 - values[0] = *aktualna najwyższa zalicytowana kwota*
 - expires[0] = *planowany czas końca aukcji*

Komunikaty fazy pierwszej typu FILE_MSG:

- * FILE_Q_MSG - prośba o przesłanie pliku. Ustawione pola:
 - message_type = FILE_Q_MSG
 - id = *identyfikator przedmiotu/licytacji*
 - item = '\0' (nieistotne)
 - file_name = '\0' (nieistotne)
 - file_size = 0

- * FILE_A_MSG - przesłanie pliku. Ustawione pola:
 - message_type = FILE_A_MSG
 - id = *identyfikator przedmiotu/licytacji*
 - item = *nazwa przedmiotu*
 - file_name = *nazwa pliku wraz z rozszerzeniem*
 - file_size = *rozmiar pliku w oktetach*
 - file = *plik*

Komunikaty fazy drugiej typu FASE_TWO_MSG:

- * POSITIVE_MSG - odpowiedź pozytywna na ostatnie zapytanie. Ustawione pola:
 - message_type = POSITIVE_MSG
 - value = *aktualna cena za przedmiot, jeśli jest to odpowiedź na JOIN_MSG, 0 w przeciwnym wypadku*
 - bidder_id = 0 (nieistotne)

- * NEGATIVE_MSG - odpowiedź negatywna na ostatnie zapytanie. Ustawione pola:
 - message_type = NEGATIVE_MSG

```
value = *powód odrzucenia, przedstawiony liczbowo*  
bidder_id = 0 (nieistotne)
```

Odpowiednie wartości pola value są wyszczególnione w rozdziale 'Opis stanów'.

- * JOIN_MSG - komunikat o chęci dołączenia do aukcji. Ustawione pola:
message_type = JOIN_MSG
value = 0
bidder_id = *identyfikator, jaki otrzyma licytujący po akceptacji*
- * CLOSED_MSG - komunikat o zakończeniu aukcji. Ustawione pola:
message_type = CLOSED_MSG
value = 0
bidder_id = 0
- * OUTBID_MSG - komunikat o zwiększeniu ceny. Ustawione pola:
message_type = OUTBIC_MSG
value = 0
bidder_id = 0
- * BID_MSG - zalicytowanie kwoty. Ustawione pola:
message_type = BID_MSG
value = *licytowana kwota*
bidder_id = *identyfikator licytującego*
- * SOLD_MSG - informacja o wygraniu aukcji. Ustawione pola:
message_type = SOLD_MSG
value = *za ile przedmiot został kupiony*
bidder_id = *identyfikator kupca*

== Opis stanów ==

-- Wystawiający: --

Dla każdej licytacji przechowujemy informacje o jej stanie oraz identyfikatorach wszystkich osób, które dołączyły do aukcji, ze szczególnym wyróżnieniem identyfikatora osoby z najwyższą zalicytowaną kwotą.

Kiedy do wystawiającego przyjdzie komunikat z fazy pierwszej, postępujemy zgodnie z tym, co jest napisane w części 'Wystawiający-Faza1'. Jeśli przyjdzie komunikat z fazy drugiej, postępujemy zgodnie z 'Wystawiający-Faza2'

W chwili wystawienia nowego przedmiotu uaktualniamy spis przedmiotów przeznaczonych do licytacji. Automat związany z fazą drugą znajduje się wtedy w stanie początkowym.

Jeśli w którymkolwiek momencie otrzymamy wiadomość, której nie jesteśmy w żaden sposób obsłużyć zgodnie z opisem protokołu, odpowiadamy NEGATIVE_MSG z value = OTHER (jeśli komunikat wymagał potwierdzenia) lub ignorujemy. W każdym wypadku należy poinformować użytkownika.

- Wystawiający-Faza1 -

Pasywne otwarcie. Odpowiadamy na przychodzące pytania zgodnie ze schematem:

- * ITEMS_Q_MSG - tworzymy komunikat typu ITEMS_A_MSG i przesyłamy go za pomocą protokołu UDP.
- * PROPERTIES_Q_MSG - tworzymy komunikat typu PROPERTIES_A_MSG dla odpowiedniego przedmiotu (na podstawie wartości pola ids[0]) i

NEGATIVE_MSG z value = WRONG_TARGET. Należy także poinformować o tym użytkownika.

- * DEAD_STATE - aukcja nie istnieje, została zakończona. Otwarcie pasywne. Komunikat BID_MSG powoduje wysłanie odpowiedzi NEGATIVE_MSG z value = CLOSED. Po wysłaniu komunikatu rozłączamy połączenie. Po otrzymaniu komunikatów POSITIVE_MSG lub NEGATIVE_MSG zrywamy połączenie.
- * OPEN_STATE - aukcja jest otwarta. Jeśli zostanie zamknięta przez użytkownika, należy przejść do stanu CLOSING_STATE. Pasywne otwarcie. Po otrzymaniu komunikatu BID_MSG należy przejść do stanu EXAMINING_STATE. Po otrzymaniu komunikatów POSITIVE_MSG lub NEGATIVE_MSG zrywamy połączenie.
- * CLOSING_STATE - aukcja została zamknięta, należy poinformować o tym wszystkich uczestników. Aktywnie inicjujemy połączenie po kolei z każdym uczestnikiem. Wysłaliśmy komunikat CLOSED_MSG i oczekujemy na przyjęcie potwierdzenia (POSITIVE_MSG), po czym kończymy połączenie. Nawet jeśli nie udało nam się uzyskać komunikatu POSITIVE_MSG, kontynuujemy (licytujący powinien się zsynchronizować podczas próby licytacji). Kiedy informujemy wszystkich uczestników o zakończeniu aukcji, wysyłamy zwycięzcy komunikat SOLD_MSG, a następnie przechodzimy do stanu DEAD_STATE. Jeśli w odpowiedzi na komunikat CLOSED_MSG otrzymaliśmy NEGATIVE_MSG, informujemy o tym użytkownika i kontynuujemy tak, jakbyśmy otrzymali POSITIVE_MSG.
- * EXAMINING_STATE - otrzymaliśmy ofertę, należy ją przeanalizować. Jeśli numer uczestnika jest niepoprawny, wysyłamy NEGATIVE_MSG z value = WRONG_ID i kończymy połączenie. Jeśli wartość licytowana jest mniejsza od aktualnej ceny, wysyłamy NEGATIVE_MSG z value = PRICE_HIGHER i kończymy połączenie. Jeśli któraś z powyższych sytuacji miała miejsce, wracamy do OPEN_STATE. Jeśli żadna z powyższych sytuacji nie miała miejsca, to wysyłamy wiadomość POSITIVE_MSG, kończymy połączenie i przechodzimy do stanu INFORMING_STATE.
- * INFORMING_STATE - nowa cena, należy poinformować uczestnika, że został przelicytowany. Jeśli była to pierwsza oferta za przedmiot, przechodzimy do stanu OPEN_STATE. Jeśli nie - aktywnie inicjujemy połączenie z uczestnikiem, który dotychczas prowadził w licytacji. Wysłaliśmy komunikat OUTBID_MSG z value ustawionym na aktualną cenę i oczekujemy na przyjęcie potwierdzenia (POSITIVE_MSG), po czym kończymy połączenie. Nawet jeśli nie udało się uzyskać potwierdzenia, kontynuujemy. Jeśli w odpowiedzi na OUTBID_MSG otrzymaliśmy NEGATIVE_MSG z value <> TIMEOUT, to informujemy o tym użytkownika i kontynuujemy tak, jakbyśmy otrzymali POSITIVE_MSG.

Kończenie licytacji:

W chwili gdy przekroczony zostanie limit czasowy, bądź użytkownik chce zakończyć aukcję. Wykonujemy akcję CLOSE_AUCTION (przedstawioną na rysunku) po najbliższym przejściu automatu do stanu OPEN_STATE.


```

      _\-----
     /  /|JOINED_STATE|-----
?OUTBID_MSG/ /-----
!POSITIVE_MSG/ / |!BID_MSG /|\
              / /  \|\ /      |?POSITIVE_MSG or ?NEGATIVE_MSG
              / /  \|\ /      |
----- |WAITING_STATE|-----
-----

```

Opis stanów dla licytującego:

- * DEAD_STATE - licytacja została zakończona, bądź nie zostaliśmy do niej dopuszczeni. Po otrzymaniu komunikatów NEGATIVE_MSG lub POSITIVE_MSG kończymy połączenie. Po otrzymaniu komunikatów OUTBID_MSG lub CLOSED_MSG odpowiadamy NEGATIVE_MSG z value = CLOSED.
- * WAITING_TO_JOIN_STATE - oczekujemy na dopuszczenie do licytacji. Jeśli otrzymaliśmy NEGATIVE_MSG, to idziemy do stanu DEAD_STATE. Jeśli otrzymaliśmy POSITIVE_MSG, to idziemy do stanu JOINED_STATE. W obu wypadkach kończymy połączenie.
- * JOINED_STATE - bierzemy udział w licytacji. Oczekujemy na komunikat z aukcji, bądź działanie użytkownika. Jeśli otrzymamy komunikat OUTBID_MSG, odpowiadamy POSITIVE_MSG, po czym kończymy połączenie. Jeśli otrzymaliśmy komunikat CLOSED_MSG, to odpowiadamy POSITIVE_MSG i kończymy połączenie, po czym przechodzimy do stanu DEAD_STATE. Jeśli użytkownik chce zalicytować, aktywnie otwieramy połączenie z wystawiającym, wysyłamy mu BID_MSG, po czym przechodzimy do stanu WAITING_STATE.
- * WAITING_STATE - zalicytowaliśmy, czekamy na rezultat. Jeśli otrzymamy komunikat CLOSED_MSG, to postępujemy tak, jakbyśmy byli w stanie JOINED_STATE. Zarówno komunikaty POSITIVE_MSG, jak i NEGATIVE_MSG powodują zakończenie połączenia i powrót do stanu JOINED_STATE.

Licytowanie:

Jeśli użytkownik chce zalicytować, jeśli znajdujemy się w stanie JOINED_STATE - wysyłamy komunikat BID_MSG, jeśli nie - robimy to, kiedy po raz pierwszy wejdziemy do stanu JOINED_STATE.

== Numery ==

Czasy:

TW = 1 sekunda

TL = 10 sekund

Stałe:

```

ITEM_LEN =          4
           // ilość oktetów przeznaczona na numer przedmiotu w identyfikatorze
MAC_LEN =           6
           // ilość oktetów przeznaczona na adres karty sieciowej w identyfikatorze
PORT_LEN =          2
           // ilość oktetów przeznaczona na numer portu w identyfikatorze
AUCTION_ID_LEN =    12
           // MAC_LEN + PORT_LEN + ITEM_LEN
BID_ID_LEN =        20
           // AUCTION_ID_LEN + MAC_LEN + PORT_LEN
MAX_ITEMS =         4294967296

```

```
    // 2^(8*ITEM_LEN)
MAX_NAME =          128
    // maksymalna długość nazwy przedmiotu
MAX_FILE_NAME =     128
    // maksymalna długość nazwy pliku wraz z rozszerzeniem
```

Numery wiadomości:

```
ITEMS_Q_MSG =       4
ITEMS_A_MSG =       6
PROPERTIES_Q_MSG =  8
PROPERTIES_A_MSG = 10
```

```
FILE_Q_MSG =        12
FILE_A_MSG =        14
```

```
JOIN_MSG =          1
POSITIVE_MSG =      3
NEGATIVE_MSG =      5
CLOSED_MSG =        7
BID_MSG =           9
NEW_PRICE_MSG =     11
SOLD_MSG =          13
```

Numery błędów:

```
UNABLE =            1
WRONG_TARGET =      2
CLOSED =             3
WRONG_ID =           4
TIMEOUT =            5
PRICE_HIGHER =      6
OTHER =              7
```