Projekt PSIR

Zespół

Adam Staciwa

Wojciech Polak

Radosław Szawłowski

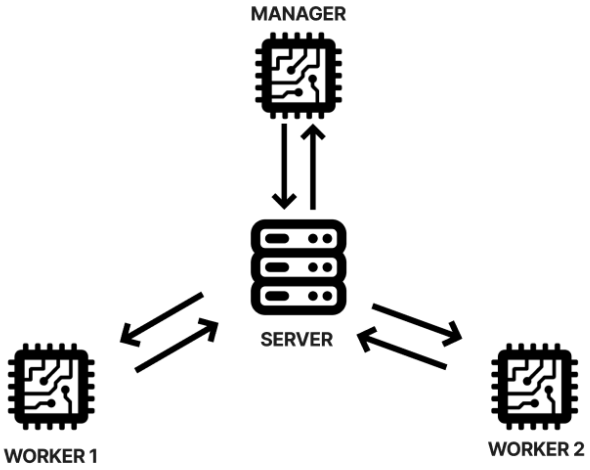
Politechnika Warszawska, Instytut Telekomunikacji

15 stycznia 2024

Spis treści

1. Wstęp

W implementacji projektu założyliśmy następującą strukturę. Manadżer, który przesyła krotki do serwera, z którego pobierane są zadania przez workerów, a następnie zwracane po ich wykonaniu. Komunikacja odbywa się przy pomocy protokołu ALP.



2. Specyfikacja ALP

Zaprojektowany przez nas protokuł przesyłu informacji, obsługujący przestrzeń krotek posiada następujący format:

— **Typ operacji** - 1 bajt
— **Długość wiadomości** - 1 bajt
— **Krotka** - n bajtów, maks. 253

Typy operacji i ich identyfikatory wykonywane przez protokół to:

— **0x01 - OUT** - dodawanie krotki do przestrzeni
— **0x02 - IN** - pobieranie konkretnej krotki z przestrzeni
— **0x04 - INP** - pobieranie konkretnej krotki z przestrzeni, bez blokowania
— **0x08 - RD** - odczytanie krotki z przestrzeni, ale krotka w niej pozostaje
— **0x10 - RDP** - odczytanie krotki z przestrzeni, bez blokowania
— **0x20 - ACK** - potwierdzenie odebrania pakietu
— **0x40 - ALP_HLL** - wiadomość z przywitaniem

Nasza wiadomość ALP może mieć maksymalnie 255 bajtów, wynika to z faktu że urządzenia dla których to projektujemy mają ograniczone zasoby. Z tego względu nasz protokół, ogranicza do minimum liczbę bitów zajmowaną przez poszczególne pola. Na typ operacji 1 bajt jest wystarczająco, ponieważ mamy kilka operacji, które są odpowiednio zapisane na różnych bitach. Dla parametru z długością wiadomości ustalono również 1 bajt, poniewaz maksymalny rozmiar naszej wiadomości to 255 bajtów , a do zapisania tej wartości wystarczy 1 bajt. Informacja o długości wiadomości, będzie nam potrzebna aby dzięki odczytaniu tego pola, łatwo rozróżniać wiadomości protokołu ALP Następnie krotka może zawierać od 0 do 253 bajtów. Rozmiar krotki jest zależny od rodzaju i ilości pól.

3. Implementacja protokołu ALP

Struktura protokołu ALP została zaimplementowana w pliku nagłówkowym **tuple_space.h**.

```
typedef struct {
    uint8_t op_type;
    uint16_t msg_len;
    Tuple *tuple;
} ALPMessage;
```

Rys. 1. Struktura wiadomości ALP

Definiujemy w niej 3 zmienne: op_type - określającą typ wiadomości, msg_len - długość wiadomości i nowa struktura zwana **Tuple**, która została zdefiniowana w następujący sposób.

```
typedef struct {
    uint8_t length;
    char* value;
} StringField;

// Tuple field structure: | TYPE | DATA |
typedef struct {
    uint8_t is_actual; // YES or NO
    uint8_t type;      // TS_INT, TS_FLOAT or TS_STRING
    union {
        int int_field;
        float float_field;
        StringField string_field;
    } data;
} Field;

// Structure for Tuple | Nr. of fields (1B) | Field 1 | Field 2 | ... | Field n |
typedef struct {
    uint8_t num_fields;
    Field *fields;
} Tuple;
```

Rys. 2. Struktura krotki

Struktura **Tuple** reprezentuje jedną krotkę. Posiada pole 'num_fields' zawierające informacje o ilości pól w krotce, oraz wskaźnik do tablicy struktur 'Field' oznaczonej 'fields.
Struktura **Field** zawiera pola: 'is_actual' zawierające informacje czy dane pole jest aktywne czy nie, pole 'type' zwraca nam informacje o typie danych. Mamy 3 możliwe typy zdefiniowane następująco: TS_INT 0x01, TS_FLOAT 0x02, TS_STRING 0x04, oraz unię 'date', która przechowuje rzeczywistą wartość danego typu danych.
Stworzono również strukturę tworzącą string potrzebny w naszej krotce o nazwie **StringField** składającego się z pola 'length' informującego o długości tego stringa oraz wskaźnika 'value' na wartość tekstową

3.1. Niezbędne funkcje w pliku nagłówkowym

W naszym przypadku, aby zgodnie z zaleceniami protokół ALP był protokołem binarnym stworzoną dwie funkcje: jedną do serializacji a drugą do deserializacji wiadomości ALP. Użyto znanych struktur i funkcji języka C, aby zrobić to jak najprosztszy i czytelny sposób.

4. Implementacja przestrzeni krotek API

Przestrzeń krotek to stworzona struktura służąca do przechowywania krotek z zadaniami. W przestrzeni krotek znajdują się wszystkie krotki niezależnie od rodzaju czy flag. Struktura przestrzeni jest wywoływana w kodzie serwera. Wygląda ona następująco.

```c
// --- TUPLE SPACE
typedef struct {
    char key[50];  // Identifier for the tuple
    Tuple *tuple;   // Tuple value
    int count;
} TupleSpaceEntry;

typedef struct {
    TupleSpaceEntry entries[MAX_TUPLES];
    int count;
    pthread_mutex_t mutex;
} TupleSpace;
```

Rys. 3. Struktura przestrzeni krotek

5. Implementacja managera

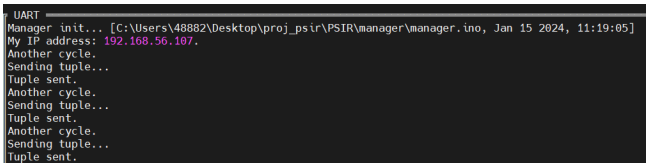
Manager to aplikacja zbudowana dla platformy emulującej Arduino, ktora buduje krotki i wysyła je do serwera z odpowiednim typem. Korzysta ona ze struktur zdefinowanych w pliku nagłówkowym tuple_space.h.
Manager na początku wysła wiadomość typu Hello do serwera. Następnie cyklicznie zaczyna wysyłać krotki z użyciem struktury ALP.
Ma także zaimplementowane odpowiednie metody do odbierania wysyłanych przez serwer wiadomości i wypisywania ich do terminala.

6. Komunikacja przy pomocy protokołu ALP

Serwer został zaimplementowany jako aplikacja TupleSpace.c. Plik zawiera metody do budowania struktury ALP, opisanej w powyższej specjalizacji, wiadomość składająca się z odpowiedniego typu, długości wiadomości i krotki jest serializowana i wysyłana do odpowiedniego adresata.
Podobna implemenctacja znajduje się w kodzie managera, gdzie budujemy wiadomość o odpowiedniej strukturze, deserializujmey je i wysyłamy przy pomocy protokołu UDP.

```
student@iot:~$ gcc -g TupleSpace.c -o Tuplespace
student@iot:~$ ./Tuplespace
[2024-01-15 11:31] TupleSpace has IP: 192.168.56.104
[2024-01-15 11:31] Socket created successfully!
[2024-01-15 11:31] Socket binded successfully!
[2024-01-15 11:31] Waiting for message...
[2024-01-15 11:31] Received message from 192.168.56.107:9545
Printing Tuple with 3 fields:
(otherTuple, 123, 3.140000)
Tuple (otherTuple, 123, 3.140000) added to Tuple Space.
Received ALP_OUT message. Added tuple to the tuple space.
[2024-01-15 11:31] Waiting for message...
[2024-01-15 11:31] Received message from 192.168.56.107:9545
Printing Tuple with 3 fields:
(otherTuple, 123, 3.140000)
Tuple (otherTuple, 123, 3.140000) added to Tuple Space.
Received ALP_OUT message. Added tuple to the tuple space.
[2024-01-15 11:31] Waiting for message...
^C
student@iot:~$
```

Rys. 4. Odbieranie przez serwer wiadomości typu ALP



Rys. 5. Wysyłanie krotek przez managera

W obecnej formie projektu, możliwe jest tylko korzystanie z funkcji nieblokujących, czyli: inp, rdt i out.

7. Implementacja i funkcje aplikacji Workerów

W naszej koncepcji dwie aplikacje workerów pobierają z serwera krotki z odpowiednimi zadaniami. W przypadku funkcji z blokowaniem, jeżeli nie znajdą odpowiadającej zadanej im krotki w przestrzeni, to czekają aż takie krotki się pojawią. Dla funckji nieblokujących, pomijają zadanie, jeśli jest niemożliwe do wykonania. Po wykonaniu zadania wysyłają krotke spowrotem do serwera.
Niestety w obecnej wersji projektu nie udało się zaimplementować funkcjonalności tych aplikacji.