

Sprawozdanie PSIR

Projekt

Jakub Sulikowski 318839
Michał Filipczyk 318761
Kacper Samuś 318829

Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych

19 lutego 2024



**Wydział Elektroniki
i Technik Informacyjnych**

POLITECHNIKA WARSZAWSKA

Spis treści

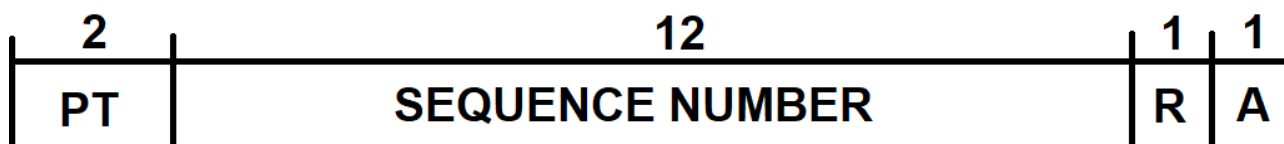
1. Zadanie projektowe	3
2. Specyfikacja ALP	3
2.1. Payload type	3
2.2. Sequence number (Nie zaimplementowano)	3
2.3. RDP_RESULT	3
2.4. Acknowledge	3
2.5. Zamiana końcówkowości	3
3. Opis implementacji serwera	4
3.1. Opis implementacji ALP na serwerze	4
3.2. Opis implementacji API przestrzeni krotek	4
3.3. Opis implementacji przestrzeni krotek na serwerze	4
4. Opis implementacji węzłów	6
4.1. Opis implementacji ALP na serwerze	6
4.2. Opis implementacji API przestrzeni krotek	6
5. Opis funkcjonalności oraz implementacji Aplikacji Manager - Worker	6
5.1. Manager	6
5.2. Worker	7
6. Opis funkcjonalności oraz implementacji Aplikacji Czujnik - Licznik	8
6.1. Czujnik	8
6.2. Licznik	9
7. Podsumowanie	9

1. Zadanie projektowe

Celem projektu było stworzenie warstwy programistycznej pośredniej między systemem operacyjnym, a aplikacjami, w postaci przestrzeni krotek. Ponadto, stworzenie protokołu ALP oraz dwóch aplikacji testujących.

2. Specyfikacja ALP

ALP jest to binarny protokół warstwy aplikacji działający na UDP, używany do komunikacji klienta z serwerem operującym przestrzenią krotek.



Rys. 1: Nagłówek pakietu protokołu ALP

Typ	ilość bitów
PAYLOAD TYPE	2
SEQUENCE NUMBER	12
RDP RESULT	1
ACKNOWLEDGE	1

2.1. Payload type

Zawiera informacje o typie przesyłanej wiadomości i wartości jakie przyjmuje odpowiednio:

- ack: potwierdź otrzymanie krotki o zadanych parametrach - wartość: 0b00
- out: umieść krotkę o zadanych parametrach w przestrzeni krotek - wartość: 0b01
- rdp: pobierz daną krotkę z przestrzeni krotek i zwróć RDP_RESULT = 1 jeżeli istnieje w przeciwnym razie zwróć RDP_RESULT = 0 - operacja nie blokująca - wartość: 0b10
- inp: pobierz a następnie wyjmij krotkę z przestrzeni krotek - operacja nie blokująca - wartość: 0b11

2.2. Sequence number (Nie zaimplementowano)

Numer sekwencji służy do identyfikowania czy nie utracono żadnego pakietu oraz czy pakiet należy do tego samego hosta w sesji.

2.3. RDP_RESULT

Jest wynikiem operacji rdp().

2.4. Acknowledge

Ta flaga ma wartość jeden gdy wysyłane jest potwierdzenie otrzymania wiadomości.

2.5. Zamiana końcówkowości

Alp ma zaimplementowaną zamianę końcówkowości danych z końcówkości hosta (Little Endian) na końcówkowość internetową (Big Endian) i odwrotnie.

3. Opis implementacji serwera

3.1. Opis implementacji ALP na serwerze

Najważniejszą cechą naszej implementacji jest wprowadzenie "rutyn" odbierania i wysyłania. Rutyna wysyłania zawiera wysłanie oryginalnej wiadomości oraz oczekiwanie na potwierdzenie przez określony czas, po którym jeżeli nie otrzyma potwierdzenia następuje ponowne przesłanie oryginalnej wiadomości. Rutyna wysyłania jest funkcją blokującą. W rutynie odbierania otrzymywana jest wiadomość oraz wysyłane jest potwierdzenie otrzymania wiadomości. Rutyna odbierania jest funkcją nieblokującą. Dodatkowo zaimplementowano mechanizm sprawdzania błędów protokołu.

3.2. Opis implementacji API przestrzeni krotek

API przestrzeni krotek jest swoistym łącznikiem między protokołem aplikacji, a przestrzenią krotek. W celu ułatwienia komunikacji zaimplementowano strukturę tuple_t odpowiedzialną za przechowywanie danych krotki - jej nazwę, liczbę pól oraz pola.

3.3. Opis implementacji przestrzeni krotek na serwerze

Przestrzeń krotek zaimplementowano jako linked_list co ułatwia operacje na przestrzeni. Na przestrzeni krotek można wykonać 3 operacje:

- Dodać krotkę na końcu listy
- Zwrócić krotkę i usunąć na podstawie indeksu
- Zwrócić krotkę bez usuwania na podstawie indeksu

```
ALP: Received Payload message
ALP: ACK message sent
SERVER: Removed tuple from Tuple Space
ALP: Payload message sent
ALP: Recived ACK message
SERVER: INP operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Removed tuple from Tuple Space
ALP: Payload message sent
ALP: Recived ACK message
SERVER: INP operation complete - going back to Idle
```

Rys. 2: Udana wykonanie operacji INP

```
ALP: Received Payload message
ALP: ACK message sent
SERVER: Given tuple does not exist in tuple space
ALP: Payload message sent
ALP: Recived ACK message
SERVER: INP operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Given tuple does not exist in tuple space
ALP: Payload message sent
ALP: Recived ACK message
SERVER: INP operation complete - going back to Idle
```

Rys. 3: Nieudane wykonanie operacji INP - krotka nie istnieje w przestrzeni - wtedy serwer zwraca pustą krotkę

```

ALP: Message resending timed out
ALP: Server going back to Idle
SERVER: INP operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Added tuple - name: check_if_prime field 1: 1 field 2: 2
SERVER: OUT operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Added tuple - name: check_if_prime field 1: 1 field 2: 3
SERVER: OUT operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Added tuple - name: check_if_prime field 1: 1 field 2: 4
SERVER: OUT operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Added tuple - name: check_if_prime field 1: 1 field 2: 5
SERVER: OUT operation complete - going back to Idle

ALP: Received Payload message
ALP: ACK message sent
SERVER: Added tuple - name: check_if_prime field 1: 1 field 2: 6
SERVER: OUT operation complete - going back to Idle

```

Rys. 4: Wyświetlane powiadomienia po stronie serwera

```

22:6:53 ALP: Received Payload message
22:6:53 ALP: ACK message sent
22:6:53 TUPLE_SPACE: Added tuple - name: change_tuple field 1: 2 field 2: 0
22:6:53 SERVER: OUT operation complete - going back to Idle

22:6:54 ALP: Received Payload message
22:6:54 ALP: ACK message sent
22:6:54 TUPLE_SPACE: Removed tuple from Tuple Space
22:6:54 ALP: Payload message sent
22:6:54 ALP: Recived ACK message
22:6:54 SERVER: INP operation complete - going back to Idle

```

Rys. 5: Dodanie implementacji wyświetlania czasu kolejnych wiadomości serwera

Mechanizm porównywania danej krotki z danym szablonem jest zrealizowany rekurencyjnie.

4. Opis implementacji węzłów

4.1. Opis implementacji ALP na serwerze

Podobnie jak w implementacji serwera działanie protokołu opiera się na rutynach. Jediną różnicą między implementacją serwerową, a węzłową jest to, że w węzłach nie zaimplementowano mechanizmu sprawdzania błędów, który figuruje w implementacji protokołu na serwerze.

4.2. Opis implementacji API przestrzeni krotek

W węzłach zaimplementowano odpowiednie funkcje reprezentujące operacje na przestrzeni krotek. Podobnie jak w implementacji na serwerze skorzystano ze struktury `tuple_t`.

5. Opis funkcjonalności oraz implementacji Aplikacji Manager - Worker

5.1. Manager

Manager zleca zadanie sprawdzania czy liczba jest liczbą pierwszą, aby to zrobić wysyłana jest krotka z liczbą do sprawdzenia.

```
UART
172.20.10.3
--OUT_PHASE--
Tuple with number 2 is out!!!
Tuple with number 3 is out!!!
Tuple with number 4 is out!!!
Tuple with number 5 is out!!!
Tuple with number 6 is out!!!
```

Po przesłaniu odpowiedniej liczby krotek, zaczyna pobierać krotki "is_prime" oraz "is_not_prime".

```
UART
Trying to remove tuple "is_not_prime"
Removed tuple from TS
Trying to remove tuple "is_prime"
Removed tuple from TS
```

Po odebraniu wszystkich krotek wypisuje je na port szeregowy.

```
--PRINTING_RESULTS_PHASE--  
  
Number: 2 - is_prime  
  
Number: 4 - is_not_prime  
  
Number: 3 - is_prime  
  
Number: 6 - is_not_prime  
  
Number: 5 - is_prime
```

5.2. Worker

Proces Workera próbuje cyklicznie pobrać krotkę "check_if_prime" z przestrzeni krotek.

```
UART  
  
--WORKER_CYCLE--  
Trying to remove tuple "check_if_prime"  
  
--WORKER_CYCLE--  
Trying to remove tuple "check_if_prime"  
  
--WORKER_CYCLE--  
Trying to remove tuple "check_if_prime"
```

Po pozyskaniu krotki Worker sprawdza czy dana liczba jest liczbą pierwszą i zwraca do przestrzeni krotek, krotkę z odpowiednim wynikiem.

```
--WORKER_CYCLE--  
  
Trying to remove tuple "check_if_prime"  
Tuple removed  
  
Checking if number is prime...  
Number is a prime!  
  
Sending result...  
Result sent
```

6. Opis funkcjonalności oraz implementacji Aplikacji Czujnik - Licznik

6.1. Czujnik

Zadaniem czujnika jest sprawdzanie wartości na cyfrowym pinie wejściowym oraz przesyłanie krotki w razie zmiany wartości na tym pinie.

```
UART
0.0.0.0

--SENSOR_CYCLE--
Sensing...
Change sensed!!!
Change tuple OUT!!!

--SENSOR_CYCLE--
Sensing...
Change sensed!!!
Change tuple OUT!!!

--SENSOR_CYCLE--
Sensing...
Change sensed!!!
Change tuple OUT!!!

--SENSOR_CYCLE--
Sensing...
Change sensed!!!
Change tuple OUT!!!
```

Gdy zczytana wartość się nie zmienia licznik wypisuje następujące wiadomości:

```
UART

--SENSOR_CYCLE--
Sensing...

--SENSOR_CYCLE--
Sensing...

--SENSOR_CYCLE--
Sensing...
```


6.2. Licznik

Zadaniem licznika jest pobieranie krotek z przestrzeni krotek oraz zliczanie danego typu zmian na cyfrowym pinie wężła czujnika. Po jakimś czasie licznik wypisuje ile razy nastąpiła zmiana danego typu.

```
--COUNTER_CYCLE--  
Trying to remove tuple "tuple_change"  
Tuple removed  
  
--COUNTER_CYCLE--  
Trying to remove tuple "tuple_change"  
  
Number of state changes from 0->1: 2  
Number of state changes from 1->0: 2  
  
--COUNTER_CYCLE--  
Trying to remove tuple "tuple_change"  
  
--COUNTER_CYCLE--  
Trying to remove tuple "tuple_change"
```

7. Podsumowanie

Udało się zaimplementować wszystkie wymagane komponenty.