

Karol Ziarek 318749

Jakub Kieruczenko 318669

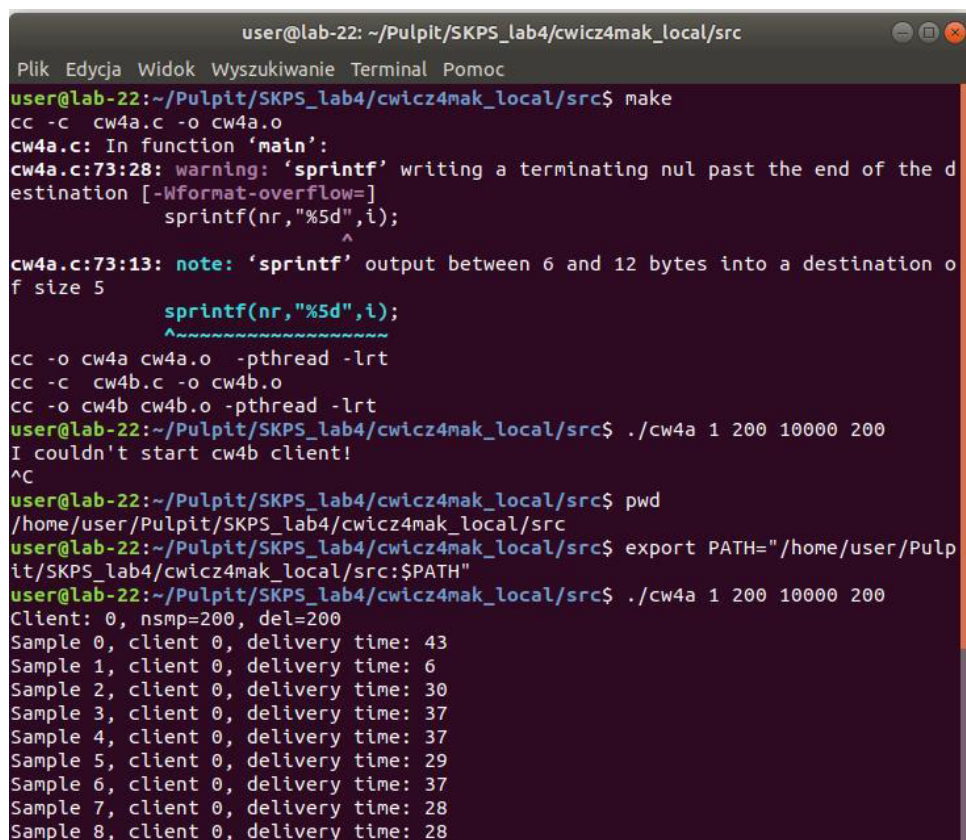
## SKPS - Laboratorium 4

Zespół korzysta z karty SD - 105e

### 1. Przetestowanie działania programów na "gospodarzu"

1. Pobrany został i rozpakowany plik skps\_lab4\_student.tar.xz
2. Program został zbudowany poleceniem *make* w folderze z plikami źródłowymi.
3. Po zmianie ścieżki w zmiennej PATH komendą:  
*export PATH="/home/user/Pulpit/SKPS\_lab4/cwicz4mak\_local/src:\$PATH"*

Program odpala się na maszynie "gospodarza"



```
user@lab-22: ~/Pulpit/SKPS_lab4/cwicz4mak_local/src
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
user@lab-22:~/Pulpit/SKPS_lab4/cwicz4mak_local/src$ make
cc -c cw4a.c -o cw4a.o
cw4a.c: In function 'main':
cw4a.c:73:28: warning: 'sprintf' writing a terminating nul past the end of the destination [-Wformat-overflow=]
    sprintf(nr,"%5d",i);
                           ^
cw4a.c:73:13: note: 'sprintf' output between 6 and 12 bytes into a destination of size 5
    sprintf(nr,"%5d",i);
    ~~~~~^~~~~~
cc -o cw4a cw4a.o -pthread -lrt
cc -c cw4b.c -o cw4b.o
cc -o cw4b cw4b.o -pthread -lrt
user@lab-22:~/Pulpit/SKPS_lab4/cwicz4mak_local/src$ ./cw4a 1 200 10000 200
I couldn't start cw4b client!
^C
user@lab-22:~/Pulpit/SKPS_lab4/cwicz4mak_local/src$ pwd
/home/user/Pulpit/SKPS_lab4/cwicz4mak_local/src
user@lab-22:~/Pulpit/SKPS_lab4/cwicz4mak_local/src$ export PATH="/home/user/Pulpit/SKPS_lab4/cwicz4mak_local/src:$PATH"
user@lab-22:~/Pulpit/SKPS_lab4/cwicz4mak_local/src$ ./cw4a 1 200 10000 200
Client: 0, nsmp=200, del=200
Sample 0, client 0, delivery time: 43
Sample 1, client 0, delivery time: 6
Sample 2, client 0, delivery time: 30
Sample 3, client 0, delivery time: 37
Sample 4, client 0, delivery time: 37
Sample 5, client 0, delivery time: 29
Sample 6, client 0, delivery time: 37
Sample 7, client 0, delivery time: 28
Sample 8, client 0, delivery time: 28
```

### 2. Zbudowanie pakietu dla OpenWRT

Pakiet został skompilowany przy pomocy SDK OpenWRT, analogicznie jak na poprzednim laboratorium i przesłany na RPi, gdzie został zainstalowany przy pomocy opkg:

```

8 -----
8 root@OpenWrt:/# wget http://192.168.9.117:8000/cwicz4mak_1_aarch64_cortex-a72.ip
8 k
8 Downloading 'http://192.168.9.117:8000/cwicz4mak_1_aarch64_cortex-a72.ipk'
8 Connecting to 192.168.9.117:8000
8 Writing to 'cwicz4mak_1_aarch64_cortex-a72.ipk'
8 cwicz4mak_1_aarch64_ 100% |*****| 4908 0:00:00 ETA
8 Download completed (4908 bytes)
8 root@OpenWrt:/# ls
8 bin                                lost+found
8 boot                               mnt
8 buggy_1.0-1_aarch64_cortex-a72.ipk overlay
8 cwicz4mak_1_aarch64_cortex-a72.ipk proc
8 demo1_1.0-1_aarch64_cortex-a72.ipk rom
8 demo1mak_1_aarch64_cortex-a72.ipk root
8 dev                                sbin
8 etc                                sys
8 gpio_in.py                         tmp
8 gpio_led_1.py                     usr
8 gpio_led_2.py                     var
8 gpio_snd.py                       worms_1.0-1_aarch64_cortex-a72.ipk
8 lib                                www
8 lib64
8 root@OpenWrt:/# opkg install cwicz4mak_1_aarch64_cortex-a72.ipk
8 Installing cwicz4mak (1) to root...
8 Configuring cwicz4mak.
8 root@OpenWrt:/# cw4a 1 1 1 1
8 Client: 0, nsmp=1, del=1
8 Sample 0, client 0, delivery time: 3500
8 waiting for childrenroot@OpenWrt:/#

```

### 3. Ustalenie granicznej wartości czasu przetwarzania

**Wariant 1. - 3 klientów, 1 rdzeń, pełne obciążenie**

Czas przetwarzania: **320000**

**Wariant 2. - 3 klientów, 2 rdzenie, pełne obciążenie**

Czas przetwarzania: **480000**

**Wariant 3. - 3 klientów, 2 rdzenie, bez obciążenia**

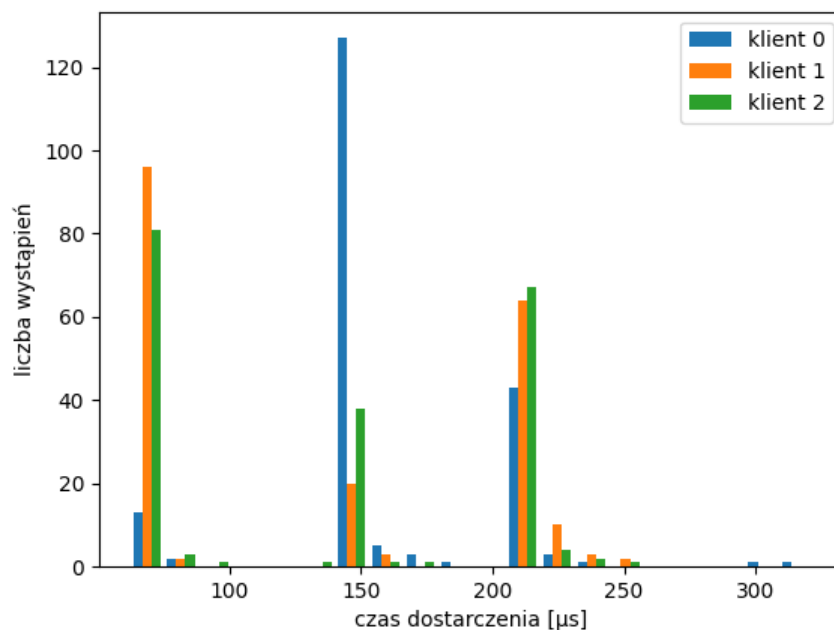
Czas przetwarzania: **600000**

**Wariant 4. - 1 klient, 4 rdzenie, bez obciążenia**

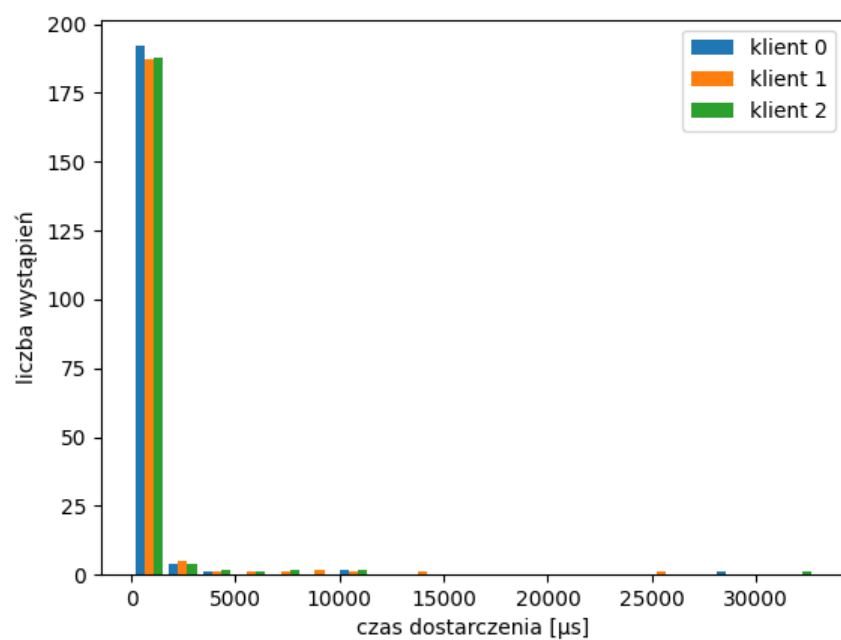
Czas przetwarzania: **840000**

### 4. Rozkład czasu dostarczenia danych

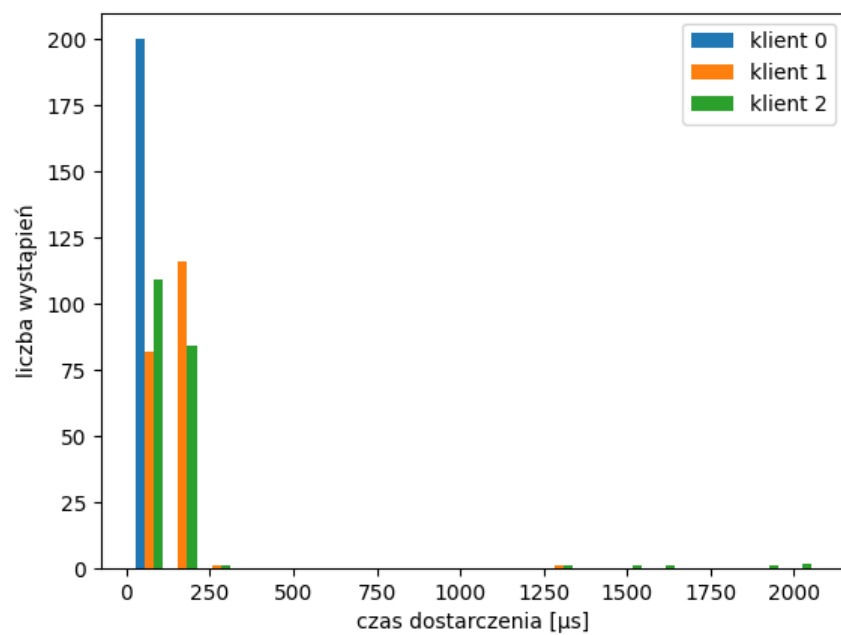
Wariant 1, czas przetwarzania = 160000:



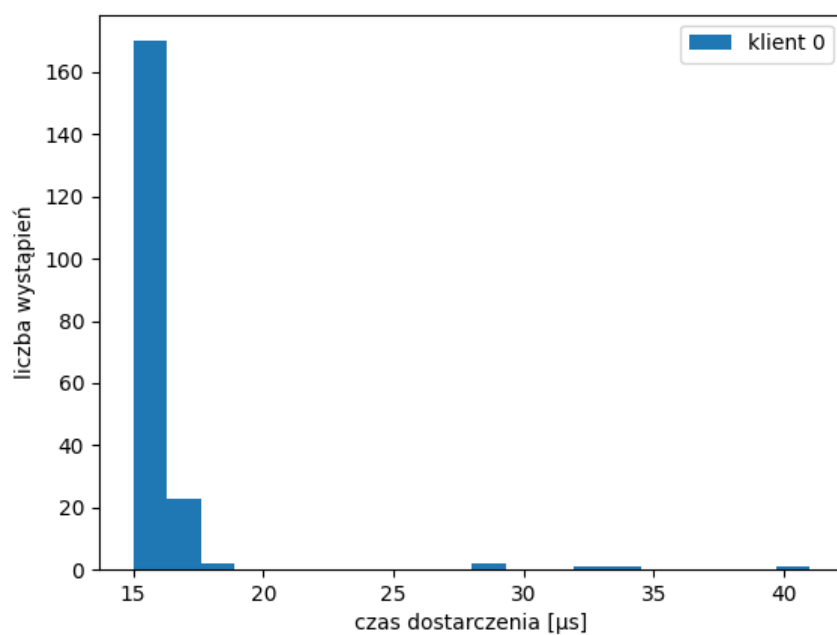
Wariant 2, czas przetwarzania = 240000:



Wariant 3, czas przetwarzania = 300000:



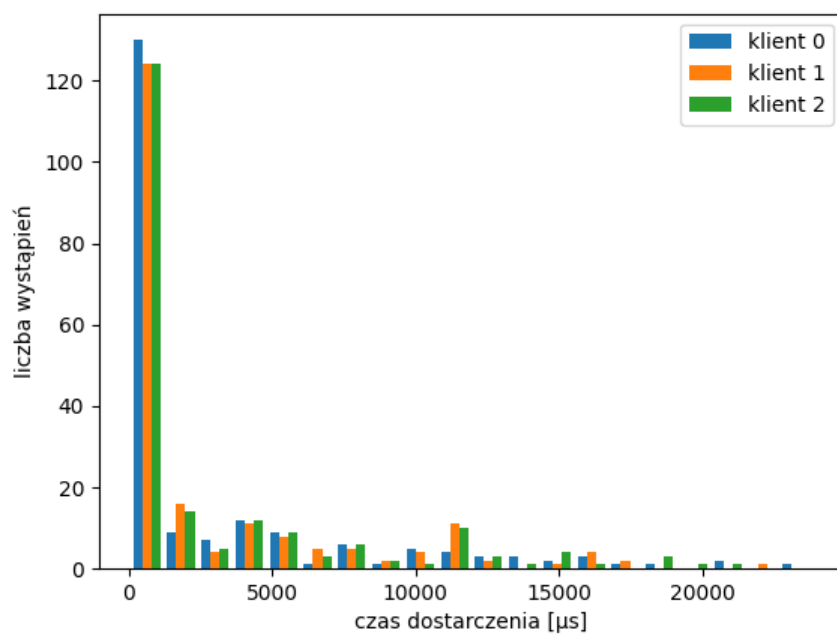
Wariant 4, czas przetwarzania = 420000:



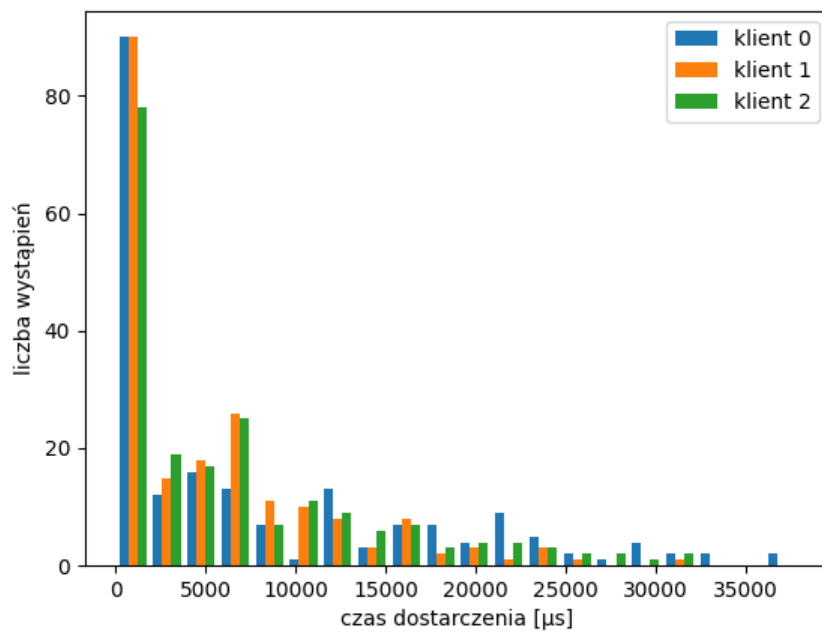
## 5. Aktywne oczekiwanie

### Aktywne oczekiwanie klienta 0:

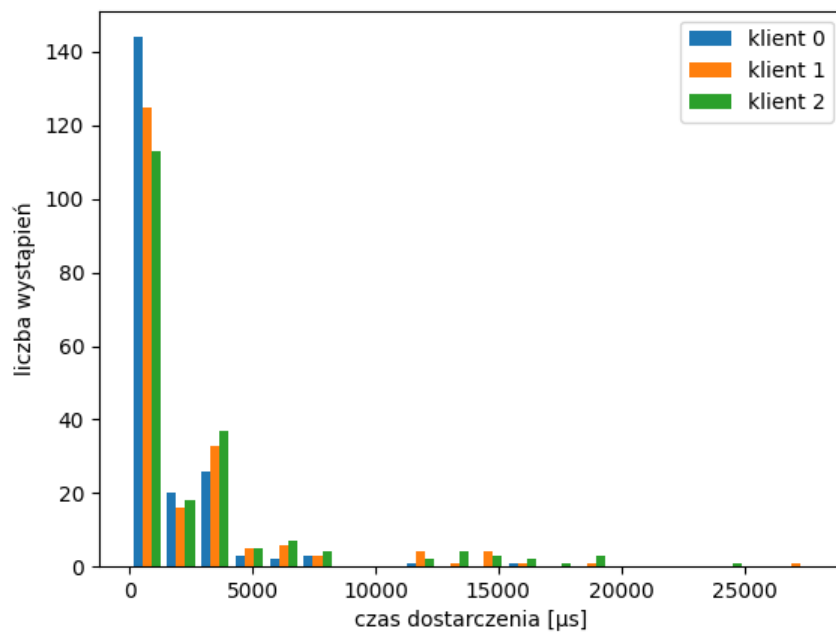
Wariant 1, czas przetwarzania = 160000:



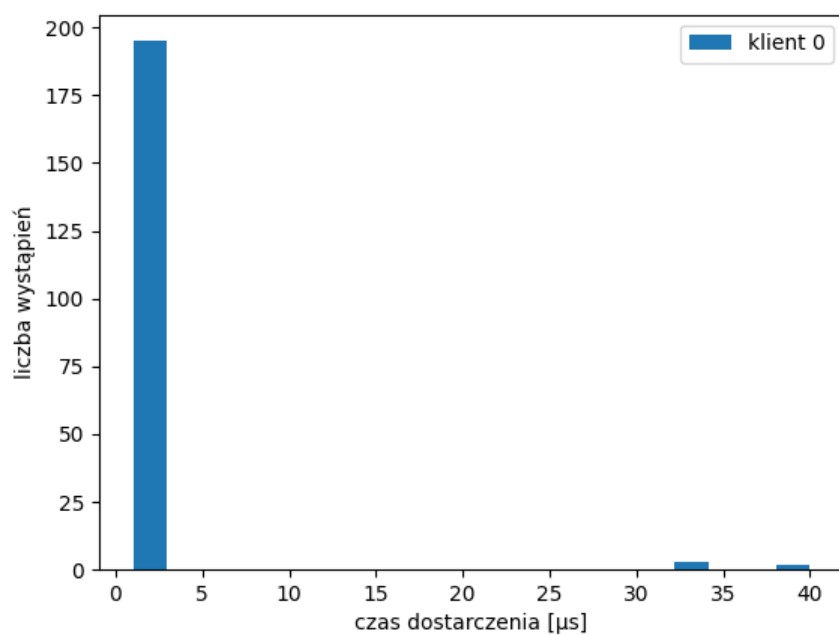
Wariant 2, czas przetwarzania = 240000:



Wariant 3, czas przetwarzania = 300000:

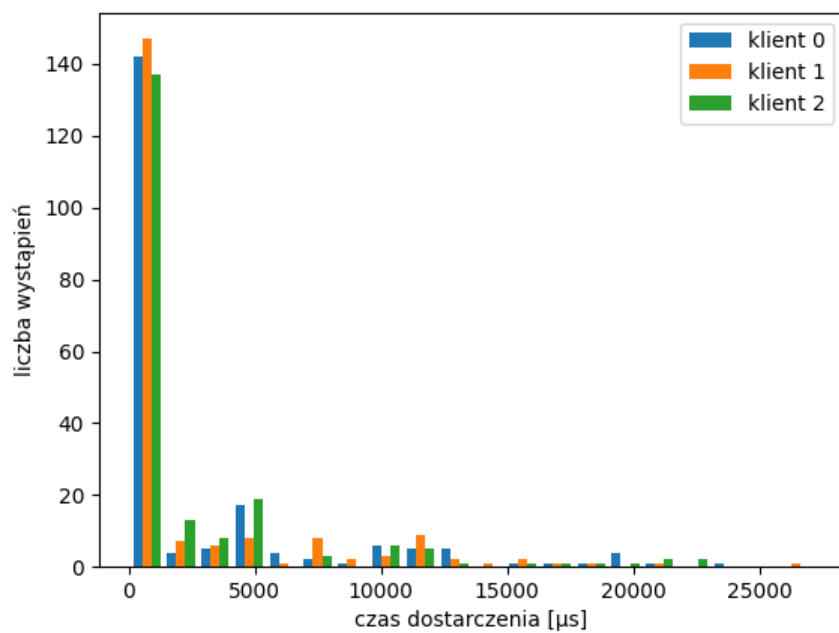


Wariant 4, czas przetwarzania = 420000:

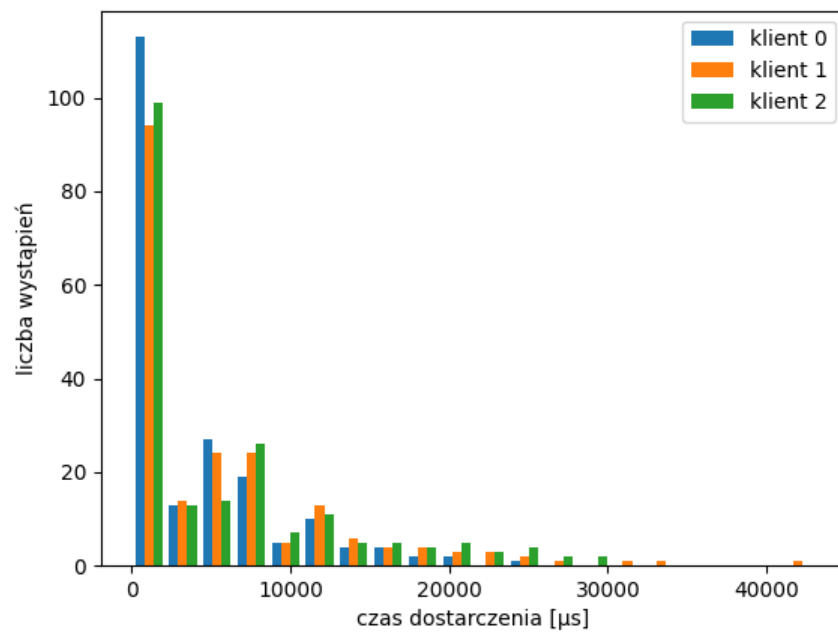


### Aktywne oczekiwanie wszystkich:

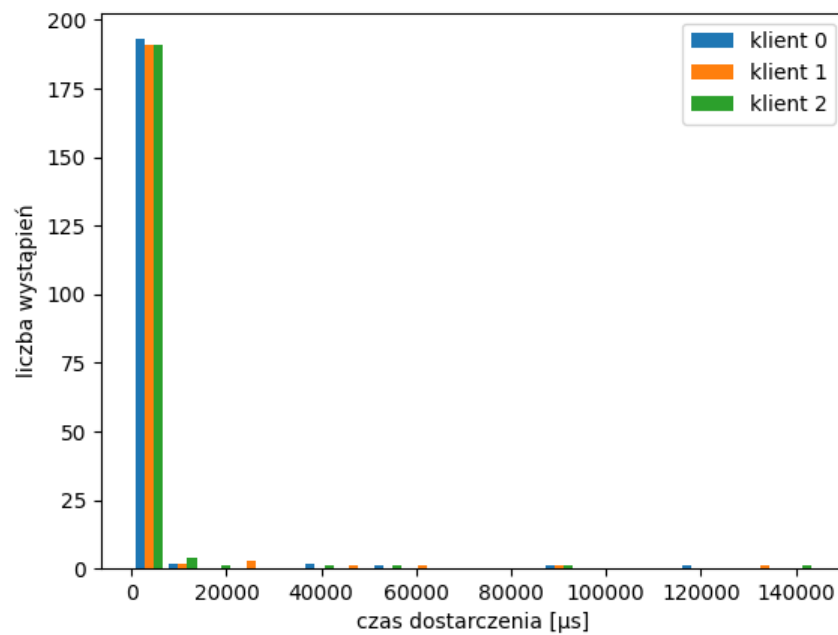
Wariant 1, czas przetwarzania = 160000:



Wariant 2, czas przetwarzania = 240000:

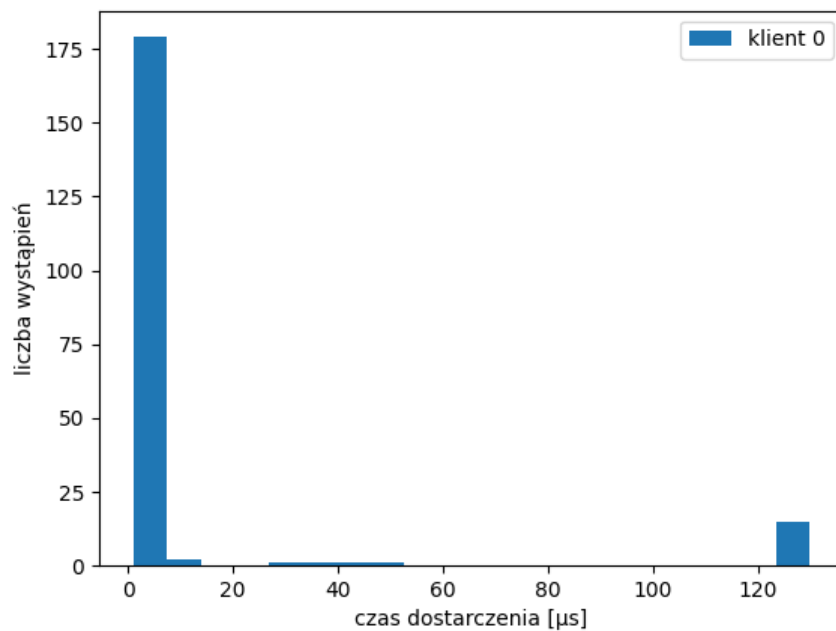


Wariant 3, czas przetwarzania = 300000:



Wariant 4, czas przetwarzania = 420000:





## 6. Właściwy pomiar czasu

Porównywane są pliki server.txt dla konfiguracji:

3 klientów, 1 rdzeń, pełne obciążenie, czas przetwarzania: 160 000

We właściwym rozwiązaniu program powinien używać timerów, ponieważ usleep nie jest dokładny - nie gwarantuje wybudzenia dokładnie wtedy, kiedy oczekujemy, mamy jedynie pewność, że proces zostanie uśpiony na co najmniej zadaną ilość czasu. Histogram przedstawia wersję korygującą niedokładności usleep poprzez zapamiętywanie czasów i dostosowywanie długości kolejnych przerw.

