Systemy Komputerowe w Sterowaniu i Pomiarach

Laboratorium 4

Aleksander Kruk
Michał Sobiech

29 listopada 2023

Zadanie 1

Użyte komendy – w folderze ~/SKPS	
Ręczne pobranie i rozpakowanie `skps_lab4_student.tar.xz`	
`PATH="$PWD:$PATH"`	Poszerzamy zmienną path by program zadziałał
`make`	Kompilujemy w folderze src (bez sdk)
`./cw4a 3 100 10000 10000`	Uruchamiamy program dla przykładowych wartości
`make clean`	Czyścimy skompilowane pliki

Zadanie 2

Użyte komendy – w folderze ~/SKPS	
Korzystamy z openwrt zainstalowanego na poprzednich laboratoriach	
`wget https://downloads.openwrt.org/releases/ 21.02.1/targets/bcm27xx/bcm2711/ openwrt-sdk-21.02.1-bcm27xx-bcm2711_gcc-8.4.0_musl.Linux-x86_64.tar.xz`	Pobranie sdk openwrt
`tar -xf openwrt-sdk-21.02.1-bcm27xx-bcm2711_gcc-8.4.0_musl.Linux-x86_64.tar.xz`	Rozpakowanie sdk
`mv openwrt-sdk-21.02.1-bcm27xx-bcm2711_gcc-8.4.0_musl.Linux-x86_64.tar.xz openwrt-sdk`	Skrócenie nazwy
`make menuconfig`	Uruchomienie konfiguracji w folderze openwrt-sdk

Wybrane opcje	Status
`Global Build Settings::Select all target specific packages by default`	✗
`Global Build Settings::Select kernel module packages by default`	✗
`Global Build Settings::Select all userspace packages by default`	✗
`Global Build Settings::Cryptographically sign package lists`	✗
`Advanced configuration options::Automatic removal of built directories`	✗

Użyte komendy	
`export LANG=C`	Dodanie wymaganej zmiennej systemowej
`vi ~/SKPS/openwrt-sdk/feeds.conf.default`	Otwarcie pliku konfiguracyjnego
`src-link skps ~/SKPS/openwrt-sdk/cw4_owrt_pkg`	Dodanie do pliku konfiguracyjnego
`~/SKPS/openwrt-sdk/scripts/feeds update -a`	Aktualizacja pakietu
`~/SKPS/openwrt-sdk/scripts/feeds install -p skps -a`	Instalacja pakietu skps
`make menuconfig`	Uruchomienie konfiguracji

Wybrane opcje	Status
`Examples::cwicz4mak`	✓

Użyte komendy

`make ~/SKPS/openwrt-sdk/package/feeds/skps/cwicz4mak/compile`	Kompilacja pakietu
`cd ~/SKPS/openwrt-sdk/bin/packages/aarch64_cortex-a72/skps/`	
`python3 -m http.server`	Uruchomienie serwera do transferu plików
`wget http://10.42.0.1:8000/cwicz4mak_1_aarch64_cortex-a72.ipk`	Pobranie pliku pakietu
`opkg install cwicz4mak_1_aarch64_cortex-a72.ipk`	Zainstalowanie pakietu na płytce
`./cw4a 3 100 10000 10000`	Uruchamiamy program dla przykładowych wartości

Zadanie 3

Użyte komendy

`opkg install htop`	instalacja htop
`vi /boot/user/cmdline.txt`	Edycja pliku cmdline.txt
`maxcpus=1`	Dodanie na końcu pliku i wyjście
`reboot`	restart
`stress-ng --matrix 0 -t 1m & htop`	Sprawdzenie ilości pracujących rdzeni
`mkdir ~/lab_4/z3_test_1`	Folder dla pierwszej konfiguracji
`cd ~/lab_4/z3_test_1`	
`mkdir 250k 300k 325k 375k 500k`	
`...`	Szacowanie punktu granicznego
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 250000`	
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 500000`	
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 375000`	
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 300000`	
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 325000`	
`vi /boot/user/cmdline.txt`	Edycja pliku cmdline.txt
`maxcpus=2`	Zmiana wartości i wyjście
`reboot`	restart
`stress-ng --matrix 0 -t 1m & htop`	Sprawdzenie ilości pracujących rdzeni
`mkdir ~/lab_4/z3_test_2`	Folder dla drugiej konfiguracji
`cd ~/lab_4/z3_test_2`	
`mkdir 250k 375k 500k`	
`...`	Szacowanie punktu granicznego
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 250000`	
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 500000`	
`stress-ng --matrix 0 -t 1m & cw4a 3 100 10000 375000`	
`mkdir ~/lab_4/z3_test_3`	Folder dla trzeciej konfiguracji
`cd ~/lab_4/z3_test_3`	
`mkdir 250k 375k 500k 750k`	
`...`	Szacowanie punktu granicznego
`cw4a 3 100 10000 250000`	
`cw4a 3 100 10000 500000`	
`cw4a 3 100 10000 375000`	
`cw4a 3 100 10000 1000000`	
`cw4a 3 100 10000 750000`	
`vi /boot/user/cmdline.txt`	Edycja pliku cmdline.txt
`maxcpus=4`	Zmiana wartości i wyjście
`reboot`	restart
`stress-ng --matrix 0 -t 1m & htop`	Sprawdzenie ilości pracujących rdzeni
`mkdir ~/lab_4/z3_test_4`	Folder dla czwartej konfiguracji
`cd ~/lab_4/z3_test_4`	
`mkdir 250k 375k 500k 750k 1M`	
`...`	Szacowanie punktu granicznego
`cw4a 3 100 10000 250000`	
`cw4a 3 100 10000 375000`	
`cw4a 3 100 10000 500000`	

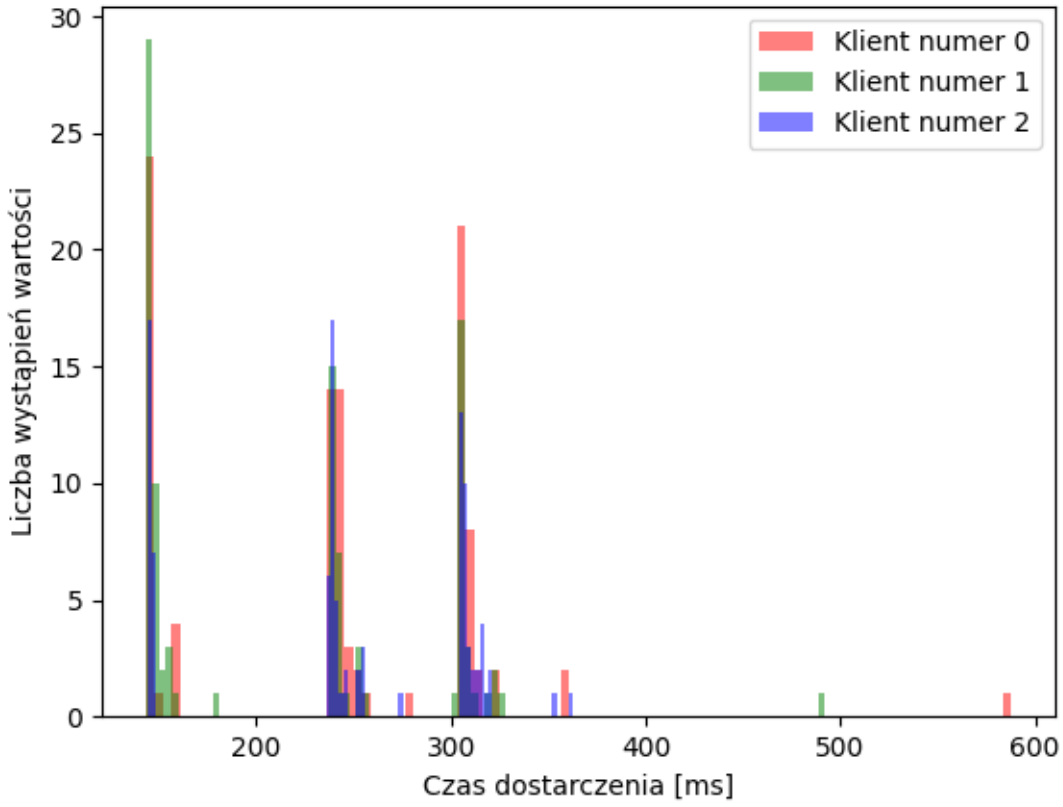
```
cw4a 3 100 10000 750000
cw4a 3 100 10000 1000000
```

Rezultaty	Ilości iteracji pętli
Konfiguracja 1	325 000
Konfiguracja 2	375 000
Konfiguracja 3	750 000
Konfiguracja 4	1 000 000

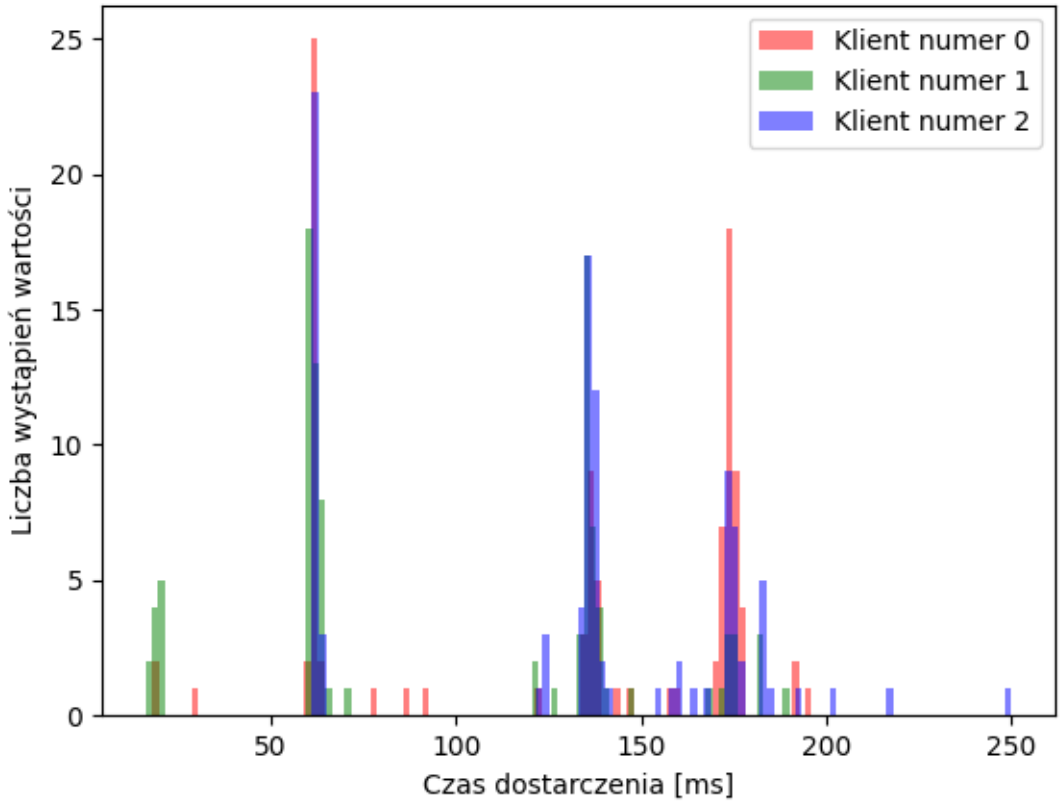
Zadanie 4

Pomiary wartości połowicznych wykonaliśmy przy okazji poprzedniego zadania. Dla każdych połowicznych wartości da się zauważyć nieproporcjonalny spadek czasu oczekiwania do różnicy wartości granicznej. Czas oczekiwania nie jest zależny liniowo od czasu obciążenia.

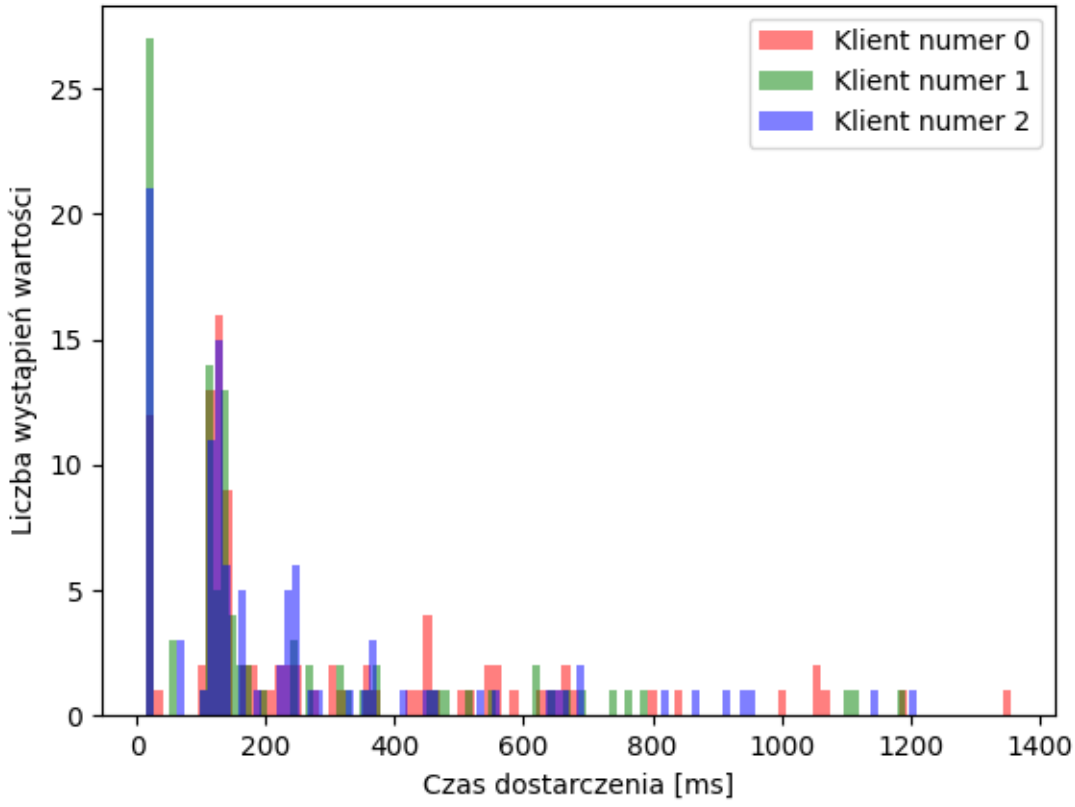
Test nr 1



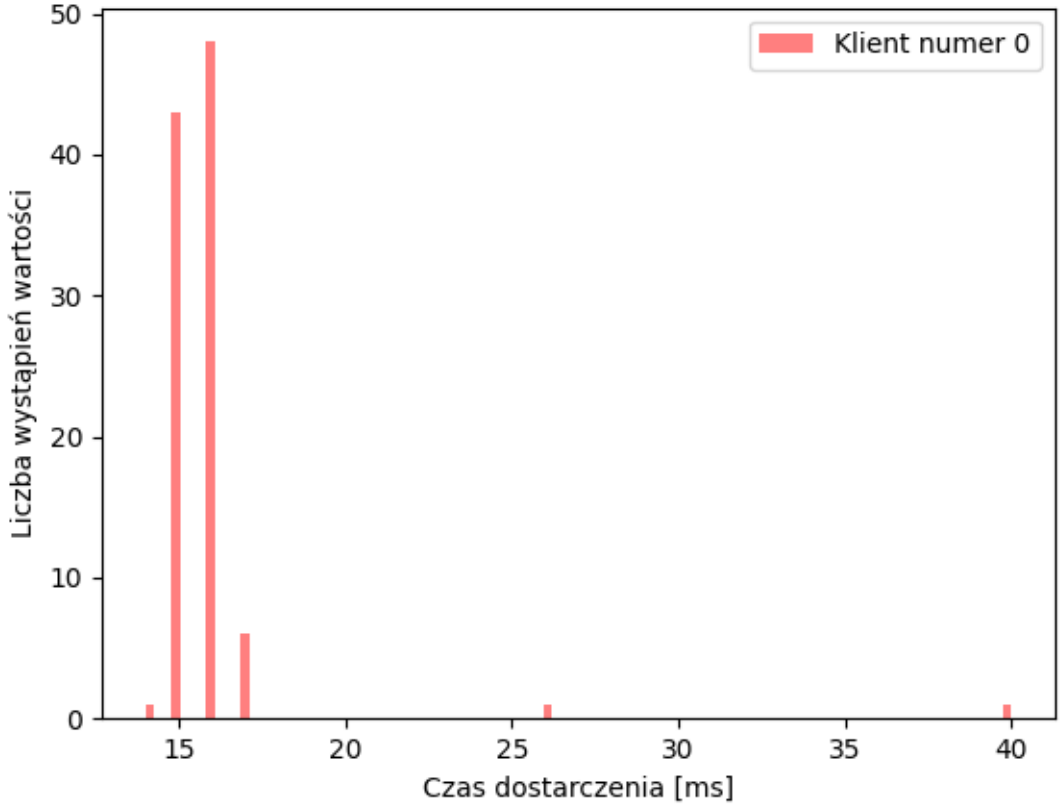
Test nr 2



Test nr 3



Test nr 4

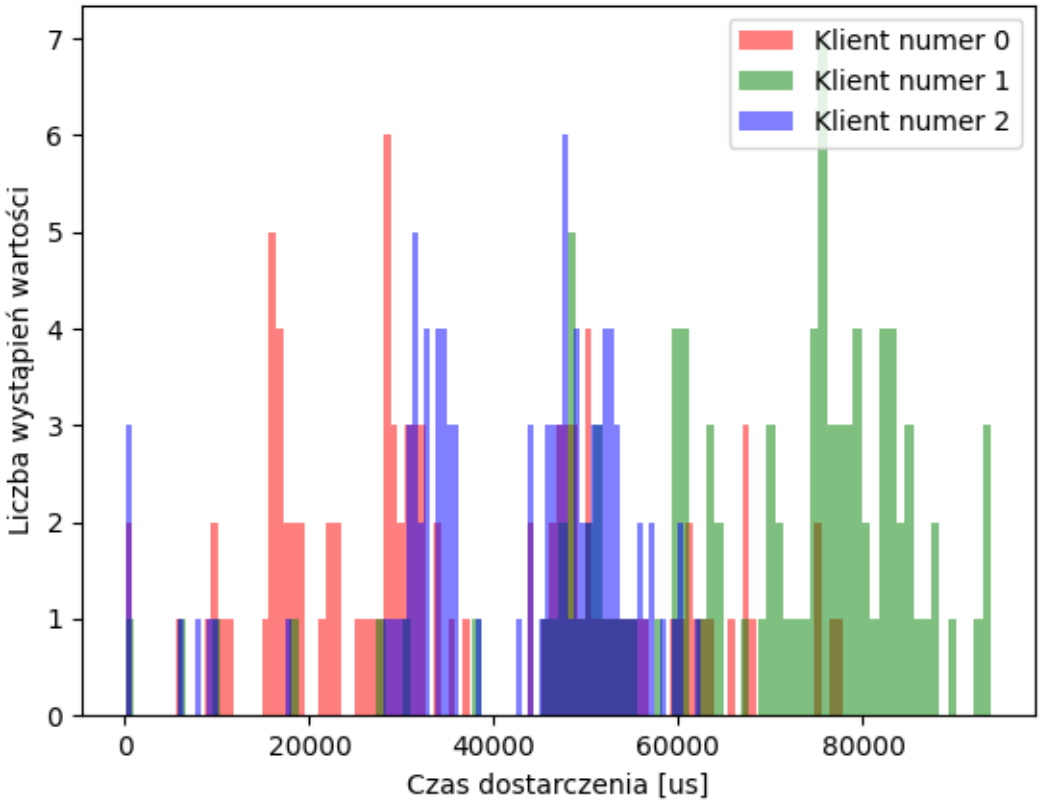


Zadanie 5

Program `cw4b.c` modyfikujemy w sekcji `else`. Po każdej zmianie rekompilujemy pakiet.

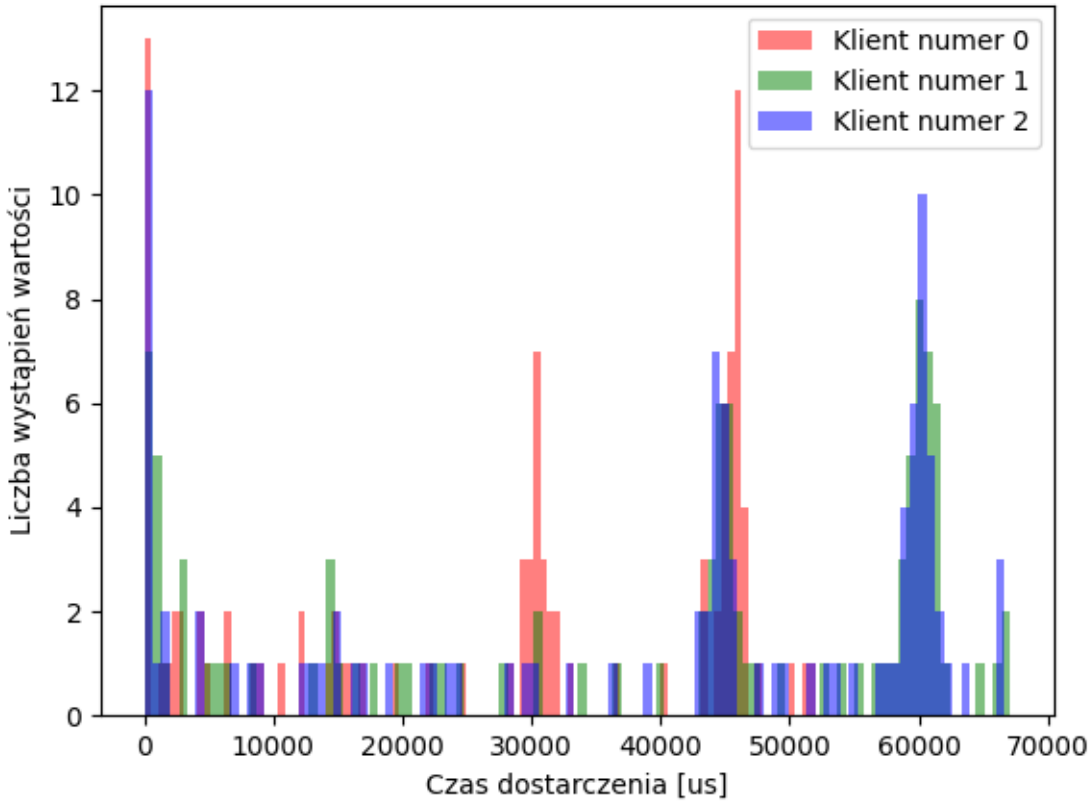
Wariant z aktywnym oczekiwaniem dla pierwszego

```
...
else {
  if (ncli != 0) {
    pthread_cond_wait(&rbuf->cvar,&rbuf->cvar_lock);
  }
  pthread_mutex_unlock(&rbuf->cvar_lock);
  ...
}
```



Wariant z aktywnym oczekiwaniem dla wszystkich

```
...
else {
  pthread_mutex_unlock(&rbuf->cvar_lock);
}
```


Klient numer 0
Klient numer 1
Klient numer 2
Liczba wystąpień wartości
Czas dostarczenia [us]
0
2
4
6
8
10
12
0
10000
20000
30000
40000
50000
60000
70000

Zadanie 6

Aby wyeliminować zaobserwowany efekt różnicy między pobraniami zestawów próbek zmodyfikowaliśmy plik `cw4a.c`. Dodaliśmy korekcję czasu oczekiwania w zależności od zaobserwowanej różnicy czasów pobrań.

```
unsigned long int getCurrentTime() {
  struct timeval currentTime;
  gettimeofday(&currentTime, NULL);
  return currentTime.tv_usec + currentTime.tv_sec * 1000000;
}
...
unsigned long prev_smptime = 0;
for(i=0; i<nsmp; i++) {
  int j;
  unsigned long smptime;

  unsigned long difference = getCurrentTime() - prevTime;
  if (udelsmp > difference) {
    usleep(udelsmp - difference);
  }
  else {
    usleep(udelsmp);
  }
  prevTime = getCurrentTime();
  ...
}
...
```