Aleksander Strzeboński, Piotr Radecki

Końcowa dokumentacja projektu z przedmiotu Programowanie Sieciowe

1. **Temat projektu**
   Zrealizowany został temat nr 13: Implementacja protokołu: Synchronizing Internet Clock frequency protocol (SIC).
2. **Zawartość projektu**
   Do projektu dołączam folder plikami:
   - changing_time.c
   - check_time.c
   - correct_time.c
   - sic.c
   - sic_extended.c
   - udpserver.c
   - Makefile
3. **Opis protokołu**
   Protokół SIC ma za zadanie dosynchronizować zegar wewnętrzny klienta do zegara wewnętrznego serwera. Główne założenia protokołu są następujące. Klient pobiera znacznik czasowy t1 czyli aktualny czas liczony w mikrosekundach od 1 stycznia 1970 roku i wysyła go do serwera. Serwer zaraz po odebraniu wiadomości zaraz przed odesłaniem wiadomości pobiera znaczniki czasowe t2 i t3 i dopisuje je do wiadomości odsyłanej do klienta. Po otrzymaniu wiadomości klient pobiera znacznik czasowy t4. Na podstawie tych czterech znaczników czasowych można obliczyć różnicę phi pomiędzy zegarami wewnętrznymi serwera i klienta. Należy pamiętać, że różnica ta nie jest stała, gdyż częstotliwości zegara wewnętrznego klienta i serwera nie są jednakowe. Klient powtarza powyższy schemat obliczania phi wielokrotnie. Następnie przy pomocy regresji liniowej przybliża funkcję phi(t) funkcją liniową:
   phi(t[us]) = K + F*t[s]
   Obliczone współczynniki K i F zapisywane są w pliku. Na podstawie tych współczynników można w każdej chwili wyliczyć w kliencie aktualny czas serwera.
4. **Opis programów**
   Uwaga: przed skompilowaniem programów komendą *make* należy zainstalować bibliotekę GSL. W CentOSie robi się to komendą:
   yum install gsl-devel

   - **changing_time.c**
     Program changing_time.c służy do zmiany zegaru wewnętrznego komputera o bezwzględną wartość. Program należy wywołać w następujący sposób:

     ./changing_time sec usec

Gdzie sec oznacza liczbę sekund o jaką chcemy zmienić zegar, a usec liczbę mikrosekund o jaką chcemy zmienić zegar (liczba mikrosekund nie może przekroczyć miliona). Uwaga: program należy wykonać z uprawnieniami roota, gdyż tylko root może zmieniać zegar wewnętrzny.

- **udpserver.c**
  Program udpserver.c przy pomocy funkcji fork tworzy dwa serwery. Należy wysołać go w następujący sposób:

  ./udpserver adres_ipv4

  Czyli np.:

  ./udpserver 192.168.56.25
  adres_ipv4 to adres na którym będzie uruchomiony Serwer nr 2.

  Serwer nr 1: serwer multicastowy, który co 5 sekund na adres 224.0.0.1 z portu 5300 wysyła wiadomość o treści np. „192.168.56.25". Jest to adres na którym działa drugi serwer unicastowy. Serwer nr 2 jest serwerem UDP działającym na podanym adresie i porcie 4444. Jest to serwer implementujący protokół SIC po stronie serwera. Oczekuje on na wiadomość a następnie odsyła otrzymaną wiadomość z dodanymi znacznikami t2 i t3. Program działa w trybie demona. Wywołuje się go bez dodatkowych argumentów.
- **sic.c**
  Program sic.c implementuje protokół SIC po stronie klienta. Program należy wywołać w następujący sposób:

  ./sic 224.0.0.1 5300 scaler(opcjonalnie)

  Program najpierw czeka na adresie multicast 224.0.0.1 na porcie 5300 na wiadomość od serwera z adresem unicast na którym uruchomiony jest serwer SIC. Następnie klient UDP co sekundę przez 60 sekund wymienia wiadomości z serwerem SIC i oblicza różnice czasu phi oraz RTT (Round Trip Time). Na podstawie obliczonych phi klient oblicza współczynniki K oraz F i zapisuje je w pliku *sic.txt* w katalogu z którego został uruchomiony w formacie:
  K
  F
  Jeżeli program wywołany jest z argumentem scaler, to klient pobrane znaczniki t1 i t4 przemnaża przez scaler w celu zasymulowania niewłaściwej częstotliwości zegara wewnętrznego. Jeżeli w trakcie wymiany danych zmieni się ścieżka, co skutkuje dużą zmianą RTT, procedura wymiany danych z serwerem SIC rozpocznie się od nowa.
- **correct_time.c**
  Program correct_time.c należy wywołać w następujący sposób:
  ./correct_time scaler(opcjonalnie)

Program ten służy do poprawienia wewnętrznego zegara klienta przy użyciu współczynników K i F zapisanych wcześniej w pliku *sic.txt*. Wywołany bez argumentu zmienia zegar wewnętrzny klienta na czas serwera. Uwaga: w wyniku tej zmiany współczynniki K i F dezaktualizują się i nie powinny być już więcej wykorzystywane. Gdy program ten wywołany zostanie z argumentem scaler, to jedynie obliczy on czas serwera mnożąc wcześniej swój czas przez scaler w celu zasymulowania niepoprawnej częstotliwości zegara. Program należy uruchomić z poziomu użytkownika root.

- **check_time.c**
  Program check_time.c uruchamia klienta UDP który łączy się z serwerem SIC, wymienia jedną wiadomość i oblicza różnicę w czasach phi w celu sprawdzenia czy klient jest zsynchronizowany z serwerem. Program należy wywołać w następujący sposób:

  ./check_time adres_serwera scaler(opcjonalne)

  Program oblicza czas serwera na podstawie współczynników K i F zapisanych w pliku sic.txt i tak obliczone czasy t1 i t4 używa do komunikacji z serwerem i obliczenia phi. Jeżeli podany jest argument scaler to program przemnaża wewnętrzny czas klienta przez scaler. Program nie zmienia czasu wewnętrznego na czas serwera, dzięki czemu współczynniki K i F nie dezaktualizują się.
- **sic_extended.c**
  Program sic_extended jest programem demona, który spina powyższe programy klienta w całość. Wywołanie jest następujące:

  ./sic_extended 224.0.0.1 5300 scaler(opcjonalne)
  scaler działa analogicznie jak w powyższych programach.

  Program najpierw robi to samo co program sic.c: Odbiera wiadomość multicast, prowadzi komunikację z serwerem SIC i zapisuje obliczone współczynniki K i F w pliku *sic_ext.txt* w swoim katalogu roboczym tzn. */tmp*. Następnie program poprawia czas wewnętrzny klienta tak jak robi to program correct_time.c. Potem program po raz kolejny prowadzi w komunikację z serwerem SIC w celu zaktualizowania zdezaktualizowanych współczynników K i F. Potem program co 10 sekund przeprowadza pojedynczą komunikację z serwerem SIC w celu obliczenia różnicy czasów tak jak robi to program check_time.c. Jeżeli różnica czasów będzie większa niż jedna sekunda, to program ponownie oblicza współczynniki K i F, poprawia czas wewnętrzny klienta, oblicza współczynniki K i F i przechodzi z powrotem do regularnego sprawdzania czy zegary są zsynchronizowane. Jeżeli podany jest argument scaler, to program nie poprawia czasu wewnętrznego klienta i co za tym idzie nie musi obliczać współczynników K i F po dwa razy.

5. Prezentacja działania programów

Kompiluję programy i uruchamiam program udpserver.c po stronie serwera:

```
[root@centOS25 projekt]# make
gcc -o changing_time changing_time.c
gcc -o check_time check_time.c
gcc -o correct_time correct_time.c
gcc -o udpserver udpserver.c
gcc -o sic sic.c `gsl-config --cflags --libs`
gcc -o sic_extended sic_extended.c `gsl-config --cflags --libs`
[root@centOS25 projekt]# ./udpserver 192.168.56.25
[root@centOS25 projekt]# _
```

Zmieniam czas wewnętrzny klienta o 10000 sekund:

```
[root@centOS26 projekt]# ./changing_time 0 0
Current time is: 1611459125358486
Current time is: Sun Jan 24 04:32:05 2021
[root@centOS26 projekt]# ./changing_time 10000 0
Current time is: 1611469154592481
Current time is: Sun Jan 24 07:19:14 2021
[root@centOS26 projekt]# _
```

Jak widać u klienta była godzina 4:32. Z początku serwer i klient byli zsynchronizowani.

Uruchamiam program sic.c (uwaga, program wykonuje się około minutę) i wyświetlam współczynniki K i F:

```
[root@centOS26 projekt]# ./sic 224.0.0.1 5300
Received message from 10.0.2.4:
192.168.56.25
[root@centOS26 projekt]# cat sic.txt
18604862292.184196
-5.339761
[root@centOS26 projekt]# _
```

Sprawdzam czy synchronizacja się udała a następnie poprawiam czas klienta:

```
[root@centOS26 projekt]# ./check_time 192.168.56.25
K=18604862292.184196
F=-5.339761
t1 = 1611459627159583
t2 = 1611459627160187
t3 = 1611459627160241
t4 = 1611459627164173
RTT = 4536
phi1 = 1664
phi2 = 1664
[root@centOS26 projekt]# ./correct_time
K=18604862292.184196
F=-5.339761
Old time was: Sun Jan 24 07:27:25 2021
New time is: Sun Jan 24 04:40:45 2021
[root@centOS26 projekt]#
```

Jak widać różnica czasów wyniosła 1664us więc synchronizacja powiodła się (8 minut między 4:32 a 4:40 uciekło na robienie zrzutów ekranu i opisywanie).

Podobne kroki wykonuję używając opcji scaler:

```
[root@centOS26 projekt]# ./sic 224.0.0.1 5300 1.0001
Received message from 10.0.2.4:
192.168.56.25
[root@centOS26 projekt]# cat sic.txt
13751764799.349518
91.457127
[root@centOS26 projekt]# ./check_time 192.168.56.25 1.0001
K=13751764799.349518
F=91.457127
t1 = 1611460443372397
t2 = 1611460443373427
t3 = 1611460443373494
t4 = 1611460443374167
RTT = 1703
phi1 = -179
phi2 = -178
[root@centOS26 projekt]# ./correct_time 1.0001
K=13751764799.349518
F=91.457127
Old time was: 1611621596706615
Corrected time is: 1611460450660835
[root@centOS26 projekt]# _
```

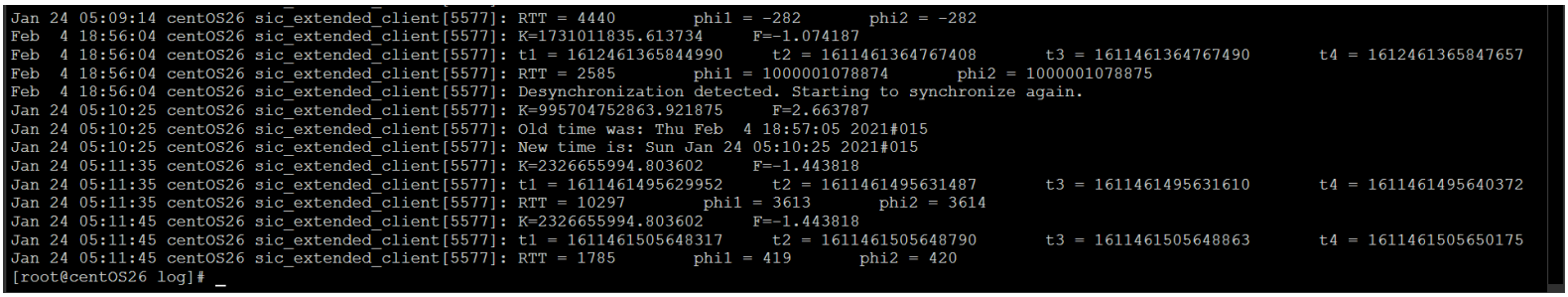
Teraz zmieniam czas klienta i uruchamiam program sic_extended.c

```
[root@centOS26 projekt]# ./changing_time -1000000 0
Current time is: 1610460777494925
Current time is: Tue Jan 12 15:12:57 2021
[root@centOS26 projekt]# ./sic_extended 224.0.0.1 5300
Received message from 10.0.2.4:
192.168.56.25
[root@centOS26 projekt]# _
```

```
Jan 12 15:14:13 centOS26 sic_extended_client[5577]:  STDIN =  0
Jan 12 15:14:13 centOS26 sic_extended_client[5577]: Sic client started by User 0
Jan 12 15:14:13 centOS26 sic_extended_client[5577]: Starting sync loop
Jan 24 05:01:54 centOS26 sic_extended_client[5577]: K=-986189087858.486938     F=-8.575751
Jan 24 05:01:54 centOS26 sic_extended_client[5577]: Old time was: Tue Jan 12 15:15:14 2021#015
Jan 24 05:01:54 centOS26 sic_extended_client[5577]: New time is: Sun Jan 24 05:01:54 2021#015
Jan 24 05:02:54 centOS26 sic_extended_client[5577]: K=1731011835.613734     F=-1.074187
Jan 24 05:02:54 centOS26 sic_extended_client[5577]: t1 = 1611460974463344     t2 = 1611460974465320      t3 = 1611460974465388      t4 = 1611460974465482
Jan 24 05:02:54 centOS26 sic_extended_client[5577]: RTT = 2070        phi1 = -941       phi2 = -941
Jan 24 05:03:04 centOS26 sic_extended_client[5577]: K=1731011835.613734     F=-1.074187
Jan 24 05:03:04 centOS26 sic_extended_client[5577]: t1 = 1611460984471306     t2 = 1611460984473136      t3 = 1611460984473219      t4 = 1611460984474021
Jan 24 05:03:04 centOS26 sic_extended_client[5577]: RTT = 2632        phi1 = -514       phi2 = -514
Jan 24 05:03:14 centOS26 sic_extended_client[5577]: K=1731011835.613734     F=-1.074187
Jan 24 05:03:14 centOS26 sic_extended_client[5577]: t1 = 1611460994477998     t2 = 1611460994479380      t3 = 1611460994479437      t4 = 1611460994479799
Jan 24 05:03:14 centOS26 sic_extended_client[5577]: RTT = 1744        phi1 = -510       phi2 = -510
[root@centOS26 log]#
```

Jak widać po godzinach logów demona program pomyślnie zsynchronizował się z serwerem i zaczął obliczanie czasów phi.
Następnie jeszcze raz zmieniam czas wewnętrzny klienta i patrzę jak zareaguje.

```
[root@centOS26 projekt]# ./changing_time 1000000 5000
Current time is: 1612461358257750
Current time is: Thu Feb  4 18:55:58 2021
[root@centOS26 projekt]# _
```



Desynchronizacja została wykryta i naprawiona.

6. Źródła

- https://tools.ietf.org/id/draft-alavarez-hamelin-tictoc-sic-02.html
- Materiały zamieszczone na stronie UPEL przedmiotu Programowanie Sieciowe