Instytut Teleinformatyki

Wydział Inżynierii Elektrycznej i Komputerowej
Politechnika Krakowska

programowanie usług sieciowych

„Protokół TCP”

laboratorium: 01

Kraków, 2014

Spis treści

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące warstwy transportowej modelu ISO/OSI a w szczególności **protokołu TCP**. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Tematyka laboratorium

Tematyką laboratorium jest programowanie aplikacji klient-serwer w oparciu o **protokół TCP** (over IP) oraz sprawdzenie jego właściwości. Protokół TCP (ang. *Transmission Control Protocol*) znajduje się w warstwie transportowej modelu ISO/OSI. Korzysta on z protokołu IPv4 lub IPv6 tworząc gniazda.

Jest to protokół zorientowany połączeniowo, czyli umożliwia zestawienie połączenia w którym efektywnie i niezawodnie przesyłane są dane. Połączenie to charakteryzuje się możliwością sterowania przepływem, potwierdzania odbioru, zachowania kolejności danych, kontroli błędów i przeprowadzania retransmisji.

TCP organizuje również dwukierunkową współpracę między warstwą IP, a warstwami wyższymi, uwzględniając przy tym wszystkie aspekty priorytetów i bezpieczeństwa. Musi prawidłowo obsłużyć niespodziewane zakończenie aplikacji, do której właśnie wędruje datagram, musi również bezpiecznie izolować warstwy wyższe - w szczególności aplikacje użytkownika - od skutków awarii w warstwie protokołu IP. Scentralizowanie wszystkich tych aspektów w jednej warstwie umożliwia znaczną oszczędność nakładów na projektowanie oprogramowania.

Pomimo związku z protokołem IP - **TCP** jest protokołem w pełni niezależnym i może zostać zaadaptowany do wykorzystania z innymi systemami dostarczania. Możliwe jest używanie go zarówno w pojedynczej sieci takiej jak Ethernet, jak i w skomplikowanej intersieci.

1.2. Zagadnienia do przygotowania

Przed przystąpieniem do realizacji laboratorium należy zapoznać się z zagadnieniami dotyczącymi:

- budowy modelu ISO/OSI
- znajomości budowy nagłówka protokołu TCP [RFC 793]
- obsługi programów **tcpdump** oraz **netstat**

A także z zagadnieniami dotyczącymi **gniazd TCP**:

- numeracja portów; [Stevens, 67]
- opcje protokołu TCP. [Stevens, 60]
- funkcje gniazd TCP; [Stevens, 110]

Literatura:

[1] W.R. Stevens, *„Programowanie Usług Sieciowych"*, *„API: gniazda i XTI"*.
[2] IETF (http://www.ietf.org/), RFC 793
[3] http://linux.about.com/od/commands/l/blcmdl8_netstat.htm

1.3. Cel laboratorium

Celem laboratorium jest poznanie elementarnych technik przesyłania prostych komunikatów między stacjami komputerowymi w sieci IP. Techniki te są bazą w tworzeniu zaawansowanych i wydajnych aplikacji klient-serwer. Podczas realizacji tego laboratorium zapoznasz się z:

- programowaniem warstwy transportowej przy użyciu protokołu TCP,
- z możliwościami protokołu TCP,
- z zaletami oraz wadami protokołu TCP,

Dzięki laboratorium nabędziesz praktycznych umiejętności w tworzeniu aplikacji klient-serwer za pomocą protokołu TCP.

2. Przebieg laboratorium

Druga część instrukcji zawiera zadania do praktycznej realizacji, które demonstrują zastosowanie technik z omawianego zagadnienia.

2.1. Przygotowanie laboratorium

W celu wykonania poniższych zadań laboratoryjnych należy:

1. Zalogować się w systemie Linux na swoim koncie.
2. Pobrać odpowiednie pliki źródłowe z katalogu
   `Home/shared/pus/pus_01_tcp;`
3. Zapoznać się lub przypomnieć informacje o programie tcpdump

2.2. Zadanie 1 Budowa nagłówka TCP

Zadanie polega na zaobserwowaniu budowy nagłówka TCP. W tym celu na jednej ze stacji roboczych uruchamiamy program **tcpdump**.

Tcpdump służy do wyświetlania nagłówków segmentów TCP pojawiających się na wybranym interfejsie sieciowym.

Składnia:

```
tcpdump [przełącznik] [wyrażenie]
```

Polecenie wywołane bez parametrów rozpoczyna nasłuch na pierwszym interfejsie w systemie. Najczęściej używane przełączniki:
-i: interfejs wybranie interfejsu do nasłuchu
-w: plik zapisz pakiety do pliku
***-dst host:* komputer** wyświetl pakiety skierowane do komputera, określonego za pomocą adresu IP lub nazwy domenowej
***-src host:* komputer** wyświetla pakiety pochodzące od podanego komputera

Następnie za pomocą dowolnej przeglądarki internetowej proszę połączyć się ze stroną http://mars.iti.pk.edu.pl i przeanalizować jak wygląda typowe połączenie TCP. W sprawozdaniu opisz swoje spostrzeżenia.

2.3. Zadanie 2. Prosta komunikacja klient-serwer

Zadaniem jest analiza przykładowego programu klienta i serwera. W tym przypadku jest to zapytanie serwera o czas.
Do tego ćwiczenia potrzebne będą źródła programu które należy ściągnąć z katalogu:

```
$ cd  home/shared/pus/pus_01_TCP
```

Następnie należy skopiować pliki *cw2_klient.c* oraz *cw2_serwer.c* do katalogu np. „tcp"
W celu uruchomienia programów należy wykonać następujące czynności:

1. Przejdź do katalogu ze źródłami serwera:
   `$ cd tcp/`
2. Skompiluj źródła do postaci binarnej:
   `$ gcc –o cw2_klient cw2_klient.c`
   `$ gcc –o cw2_serwer cw2_serwer.c`
3. Uruchom serwer na porcie 8500:
   `$ ./cw2_server ip nr_portu`
4. Uruchom klienta i przeanalizuj działanie programu
   `$ ./cw2_klient ip nr_portu`
5. Za pomocą programu tcpdump zaobserwuj trójfazowe połączenie tcp. Z wróć uwagę na numery jakie przesyłają flagi SYN oraz ACK.

2.4. Zadanie 3. Identyfikacja klienta

Proszę zmodyfikować kod źródłowy serwera, tak by wyświetlał on adres IP oraz numer portu klienta z którym nawiązał połączenie.
Wskazówka:

W tym ćwiczeniu pomocne mogą okazać sie funkcja inet_ntop() oraz ntohs().

Przykładowy wynik działania aplikacji powinien być następujący:

```
nawiazano polaczenie z 192.168.202.06, port 33654…
```


2.5. Zadanie 4. Komunikacja klient-serwer. Echo

W celu zapoznania się z kolejnymi funkcjami oprogramowania protokołu TCP stworzymy dwa programy komunikujące się ze sobą na zasadzie klient – serwer. Program klienta korzystając z protokołu TCP ustanawia połączenie z serwerem, a następnie przesyła do niego jeden znak (podany jako pierwszy argument wywołania programu) i oczekuje na odpowiedź serwera. Serwer zaś oczekuje na znak przesłany od klienta. Zaraz po otrzymaniu znaku przesyła go z powrotem do klienta - czyli jest to tzw. serwer echa.

1. Utwórz katalog na pliki źródłowe, np. TCP_echo:
   `$ mkdir TCP_echo`
2. Skopiuj pliki źródłowe: klient2.c oraz serwer2.c do nowo utworzonego katalogu z katalogu: `home/shared/pus/pus_01_TCP`
2. Skompiluj pliki źródłowe:
   `$ cc cw4_serwer.c –o serwer oraz cc cw4_klient.c –o klient`
3. Uruchom programy:
   `$ ./serwer &`
   `$ ./klient z`
4. Zaobserwuj przepływ danych za pomocą programu `ethereal`.

2.6. Zadanie 5. Modyfikacja serwera Echo

Na postawie programów z zadania 4 zmodyfikuj kod serwera w taki sposób aby serwer po otrzymaniu od klienta dowolnego tekstu odsyłał go w postaci odwróconej i dużymi literami.

2.7. Zadanie 6. Implementacja serwera wieloprocesowego.

Zadanie to zawiera przykład zaimplementowanego serwera oraz klienta ,którzy podobnie jak w zadaniu poprzednim używają do komunikacji w warstwie transportowej protokół TCP. Różnicą w porównaniu z poprzednim ćwiczeniem jest sposób implementacji serwera. W tym wypadku zadaniem głównego procesu serwera jest nasłuchiwanie nadchodzących połączeń oraz wypadku nawiązania połączenia przez klienta utworzenie potomnego procesu i przekazanie mu obsługi danego klienta. Taka implementacja serwera jest często spotykana ,gdy obsługa pojedynczego klienta jest bardzo długa i mogła by powodować na dany czas zajęcie serwera do tego stopnia, że nie akceptował by nowych połączeń. Aby osiągnąć opisany powyżej efekt użyliśmy funkcji fork()

która po nawiązaniu połączenia z klientem tworzy proces potomny i przekazuje mu obsługę nad klientem.

- ***fork*** – funkcja tworzy proces potomny procesu głównego. Oba procesy różnią się numerami pid i ppid. Funkcja ta zwraca -1 w wypadku błędu, 0 dla procesu potomnego oraz procesowi potomnemu numer pid syna.

```
#include <sys/types.h>
#include <unistd.h>

pid_t fork ();
```

Poniżej umieszczono przekład kodu serwera który pokazuje użycie funkcji fork().

```
for(;;)
      {
            len=sizeof(addr_k);
            acceptfd=accept(listenfd,(struct sock-
            addr*)&addr_k,&len);
            if(fork()==0)
            {
                  close(listenfd);
                  int pom=read(acceptfd,buf,5);
                  buf[2]=(buf[0]+buf[1])/2;
                  printf("Pid%d\n",getpid());
                  printf("Ppid%d\n",getppid());
                  buf1[0]=getpid();
                  buf1[1]=getppid();
                  write(acceptfd,buf,5);
                  write(acceptfd,buf1,4);
                  close(acceptfd);
                  return;
            }
      }
```

Widzimy że proces główny nasłuchuje tylko w nieskończonej pętli nowych żądań połączeń. Natomiast po pojawieniu się takiego żądania tworzony jest nowy proces potomny za pomocą funkcji fork(). Następnie jak wiemy funkcja ta zero zwraca procesowi potomnemu zatem instrukcje zawarte w instrukcji if będą wykonane tylko przez proces potomny . Zatem to on zajmuje się odebraniem danych i wyliczeniem średniej oraz przesłanie jej do klienta.

W celu uruchomienia tego programu należy najpierw pobrać źródła, które znajdują się w katalogu

```
$ cd  home/shared/pus/pus_01_TCP
```


Następnie należy skopiować pliki *cw6_klient.c* oraz *cw6_serwer.c* do katalogu np. „tcp" W celu uruchomienia programów należy wykonać następujące czynności:

1. Przejdź do katalogu ze źródłami serwera:
   ```
   $ cd tcp/
   ```
2. Skompiluj źródła do postaci binarnej:
   ```
   $ gcc –o cw6_klient cw6_klient.c
   $ gcc –o cw6_serwer cw6_serwer.c
   ```
3. Uruchom serwer
   ```
   $ ./ cw6_serwer
   ```
4. Uruchom klienta i przeanalizuj działanie programu
   ```
   $ ./ cw6_klient <ip>
   ```

Przykładowe wyniki generowane przez klienta:

```
$ ./cw6_klient 192.168.202.2
Podaj pierwsza liczbe <0,127>
12
Podaj druga liczbe <0,127>
18
Srednia wynosi 15
Pid wynosi 2114
Ppid wynosi 2113
```

Wskazówki : w czasie analizy działania programu zwróć szczególną uwagę na zmianę Pid i Ppid przy uruchamianiu klientów nawiązujących połączenie z tym samym serwerem oraz z różnymi serwerami. Wytłumacz zauważone prawidłowości, czym są one spowodowane.

3. Opracowanie i sprawozdanie

Realizacja laboratorium pt. *„Protokół TCP"* polega na wykonaniu wszystkich zadań programistycznych podanych w drugiej części tej instrukcji. Wynikiem wykonania powinno być sprawozdanie w formie wydruku papierowego. Sprawozdanie powinno zawierać:

- opis metodyki realizacji zadań (system operacyjny, język programowania, biblioteki, itp.),
- algorytmy wykorzystane w zadaniach (zwłaszcza, jeśli zastosowane zostały rozwiązania nietypowe),
- opisy napisanych programów wraz z opcjami,
- uwagi oceniające ćwiczenie: trudne/łatwe, nie/realizowalne podczas zajęć, nie/wymagające wcześniejszej znajomości zagadnień (wymienić jakich),
- wskazówki dotyczące ewentualnej poprawy instrukcji celem lepszego zrozumienia sensu oraz treści zadań.

