Instytut Teleinformatyki

Wydział Inżynierii Elektrycznej i Komputerowej
Politechnika Krakowska

programowanie usług sieciowych

„Protokoły IPv4 i IPv6”

laboratorium: 04

Kraków, 2014

Spis treści

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące protokołów internetowych IPv4 i IPv6. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Tematyka laboratorium

Tematyką laboratorium jest programowanie aplikacji klient-serwer wykorzystując protokoły trzeciej warstwy modelu OSI: IPv4 i IPv6. IP (ang. *Internet Protocol*) udostępnia usługi pakowania i adresowania. IP identyfikuje hosty lokalne i zdalne. Gdy trasa do sieci docelowej wymaga innych rozmiarów pakietu, IP dzieli pakiet na fragmenty, co pozwala na ich transmisję bez błędów, a następnie składa razem fragmenty w pakiet w hoście docelowym. Odrzuca też pakiety przeterminowane oraz przekazuje wyznaczone pakiety do protokołów w wyższych warstwach

1.2. Zagadnienia do przygotowania

Przed przystąpieniem do realizacji laboratorium należy zapoznać się z zagadnieniami dotyczącymi **IPv4**: [Stevens, Rozdział 10]

- Format nagłówka IPv4; [RFC 6734]
- Adresowanie IPv4; [RFC 791]
- Adresy internetowe zdalne i lokalne.

A także z zagadnieniami dotyczącymi **IPv6**: [Stevens, Rozdział 10]

- Format nagłówka IPv6;
- Nagłówki rozszerzeń IPv6 i ich kolejność;
- Opcje;
- Adresowanie w protokole IPv6.

Zapoznać się także należy z linuxowymi narzędziami takimi jak: `netstat, ifconfig`. A także z funkcjami `inet_pton i inet_ntop`.

Literatura:

[1] W.R. Stevens, *„Programowanie Usług Sieciowych”*, *„API: gniazda i XTI”*.
[2] R. Scrimger, P. LaSalle, C. Leitzke, M. Parihar, M. Gupta, *“TCP/IP.Biblia*
[3] RFC 2460, RFC 791.

2. Przebieg laboratorium

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

2.1. Przygotowanie laboratorium

Przed wykonaniem ćwiczenia należy sprawdzić czy jest wkompilowaną w jadro obsługa protokołu IPv6.Najprościej można to sprawdzić wykonując polecenie:

```
$ ifconfig
```

Do wydania tego polecenia wymagane są uprawnienia administratora. Jeżeli w opisie interfejsów znajdzie się linia podobna do tej:

```
inet6 addr: fe80::230:4fff:fe18:d8eb/64 Scope:Link
```

oznacza ona, iż jest aktywna obsługa protokołu IPv6. Jeżeli takowej linii nie będzie oznacza to że jądro nie obsługuje IPv6.

Do wykonania ćwiczeń potrzebne są programy które należy skopiować z katalogu `home/shared/pus/pus_04_IPv4_IPv6` do katalogu `pus` w swoim katalogu domowym.

Aby tego dokonać należy wykonać następujące polecenia:

```
$ mkdir –p ~/pus/pus_04_ipv4_ipv6
$ cp home/shared/pus/pus_04_ipv4_ipv6/* ~/pus/pus_04_ipv4_ipv6
```


2.2. Zadanie 1. Weryfikacja kolejności bajtów

Program wykorzystywany w tym ćwiczeniu umieszcza parę liczb w pamięci a następnie odczytuje komórki pamięci w określonej kolejności.
Program należy uruchomić w miarę możliwości na różnych komputerach (laboratorium, mars, serwer studencki)

Ćwiczenie wykonujemy w następującej kolejności:

1. Przechodzimy do katalogu
   ```
   $ cd ~/pus/pus_04_ipv4_ipv6
   ```
2. Kompilujemy program:
   ```
   $ gcc byteorder.c –o byteorder
   ```
3. Uruchamiamy program:
   ```
   $ ./byteorder
   ```

Proszę wyjaśnić działanie programu na różnych architekturach(Intel, IBM).
Jaki to ma wpływ na komunikację w sieci?

2.3. Zadanie 2. Przekształcanie adresu IP z notacji kropkowej na zrozumiałą dla maszyny

Program pokazuje w jaki sposób przekształca się adres IP z postaci „10.10.1.1” do postaci która może być przetwarzana przez maszynę.

Ćwiczenie wykonujemy w następującej kolejności:

1. Przechodzimy do katalogu
   ```
   $ cd ~/pus/pus_04_ipv4_ipv6
   ```
2. Kompilujemy program:
   ```
   $ gcc ip2int.c –o ip2int
   ```
3. Uruchamiamy program:
   ```
   $ ./ip2int
   ```

Zadanie polega na dopisaniu kodu zamieniającego adres IP z postaci dwójkowej na postać kropkowo-dziesiętną

Wskazówka: Do wykonania tego zadania pomocne będą funkcje:


```
#include <arpa/inet.h>

int inet_pton(int family, const char *strptr, void *addrptr);
```

przekazuje: 1, jeśli wszystko w porządku; 0, jeśli dane wejściowe maja niepoprawna postać prezentacji; -1 jeśli wystąpił błąd.

```
const char *inet_ntop(int family, const void *addrptr, char *strptr,
size_t len);
```

przekazuje: wskaźnik do wyniku,jeśli wszystko w porządku; NULL jeśli wystąpił błąd.

Pierwsza funkcja `inet_pton` próbuje przekształcić napis wskazywany przez argument `strptr` i zapamiętać wynik w pamięci pod adresem wskazywanym przez argument `addrptr`.
Funkcja `inet_ntop` wykonuje przekształcenie odwrotne – z postaci liczbowej wskazywanej przez argument `addrptr` do postaci prezentacji wskazywanej przez argument `strptr`.

2.4. Zadanie 3. Program nawiązujący połączenie

Program nawiązuje połączenie z zadanym portem oraz z zadanym adresem IPv4. Jeżeli połączenie zostało nawiązane zostaje wyświetlony odpowiedni komunikat.

Ćwiczenie wykonujemy w następującej kolejności:

1. Przechodzimy do katalogu
   `$ cd ~/pus/pus_04_ipv4_ipv6`
3. Edytujemy źródło programu i zmieniamy port oraz IP
   `$ nano ./test.c`
4. Kompilujemy program
   `$ gcc ./test.c -o ./test`
5. Uruchamiamy program
   `$ ./test`

Zadanie polega na przetestowaniu portów dobrze znanych oraz większych od 1024. Co zaobserwujemy ?
Następnie dopisujemy kod wysyłający dane do gniazda po nawiązaniu połączenia.
Wykorzystujemy w tym celu następujący kawałek kodu:


```
// nasza wiadomość
char *wiadomosc="Witaj";
.
.
.
// wysłanie wiadomości do gniazda
wyslane_bajty=send(gniazdo,wiadomosc,dlugosc,0);

printf("Wyslano %d bajtow",wyslane_bajty);
// zamknięcie gniazda
close(gniazdo);
```

Wszystko śledzimy przy pomocy polecenia tcpdump uruchomionego na innej konsoli. Obserwujemy jakie dane odbieramy od hosta do którego wysyłamy wiadomość.

2.5. Zadanie 4. Analiza przebiegu datagramów dla IPv4 i IPv6

Podana jest przykładowa implementacja serwera oraz klienta IPv4 oraz takiej samej pary dla IPv6. Zadanie polega na przeanalizowaniu działania i przebiegu datagramów podczas komunikacji dla współdziałających ze sobą programów: `serwera6` <-> `klienta6` dla IPv6 oraz `serwera4` <-> `klienta4` dla IPv4 i zaobserwowaniu różnic w komunikacji przy użyciu prostego snifera.

W tym celu należy:

1. Przejść do katalogu roboczego laboratorium nr 4 (przygotowany został wcześniej):
   `$ cd ~/pus/pus_04_ipv4_ipv6`
2. Uruchomić polecenie `netstat` i zapoznać się z aktualnym stanem połączeń TCP:
   `$ netstat -t -n`
3. Uruchomić skompilowany wcześniej program „serwer”
   `$ ./serwer <numer_portu>`
   - gdzie `<numer_portu>` to liczba z zakresu 1024-65536
4. Po ponownym uruchomieniu polecenia netstat i należy zaobserwować czy doszedł nowy wpis, odpowiadający uruchomionemu serwerowi.
5. Uruchomić aplikację kliencką: `klient6`**.** Program ten jest wywoływany z linii poleceń w następujący sposób:
   `$ ./klient <adres_ip_serwera> <numer_portu>`
   a. gdzie `<adres_ip_serwera>` to adres IP serwera podany w postaci zgodnej z IPv6 np.: ”**::ffff:127.0.0.1**”
   b. `<numer_portu>` to liczba z zakresu 1024-65536

W tym momencie serwer powinien przesłać do klienta wiadomość o treści: „Ala ma kota”, po wyświetleniu tej wiadomości klient kończy swoje działanie, a serwer wciąż nasłuchuje. Można wykonać kilka takich iteracji uruchamiając aplikację klienta. Należy przy tym zaobserwować przebieg poszczególnych pakietów przy użyciu narzędzia `tcpdump`.

6. Analogicznie należy przeprowadzić takie samo działanie dla pary programów serwer4 i klient4. Parametry wywołania są takie same z tą jedynie różnicą, że dla klienta4 należy podać `<adres_ip_serwera>` jako zwykły adres IPv4 np.: 127.0.0.1
7. Zaobserwować i podać różnice wynikające z komunikacji i przebiegu datagramów dla obydwu typów protokołów – z czego one wynikają? Wnioski należy umieścić w sprawozdaniu.

2.6. Zadanie 5. Współdziałanie klienta IPv6 i serwera IPv4 i odwrotnie

Zadanie to polega na sprawdzeniu możliwości współdziałania aplikacji klient<-> serwer, które używają różnych typów protokołu IP.

1. Uruchomić skompilowany wcześniej program „serwer”
   ```
   $ ./serwer1 <numer_portu>
   ```
   c. gdzie <numer_portu> to liczba z zakresu 1024-65536
2. Uruchomić aplikację kliencką: klient4**.** Program ten jest wywoływany z linii poleceń w następujący sposób:
   ```
   $ ./klient1 <adres_ip_serwera> <numer_portu>
   ```
   d. gdzie `<adres_ip_serwera>` to adres IP serwera podany w postaci zgodnej z IPv6 np.: ”**::ffff:127.0.0.1**”
   e. `<numer_portu>` to liczba z zakresu 1024-65536
3. Należy odpowiedzieć na następujące pytania:
   f. Czy istnieje komunikacja pomiędzy programem klienta IPv6 i serwera IPv4 ?
   g. Jakie warunki muszą zostać spełnione, aby taka komunikacja była możliwa?
   h. Jaką postać mają podczas komunikacji przepływające datagramy IPv4 czy IPv6?
4. Następnie należy zamienić protokoły w poszczególnych aplikacjach. Czyli uruchomić serwer oparty na protokole IPv6 a klient IPv4.
   W tym celu należy uruchomić:
   i. `$ ./serwer1 <numer_portu>`
   j. `$ ./klient1 <adres_ip_serwera> <numer_portu>`

k. gdzie <adres_ip_serwera> to adres IP serwera podany w postaci zgodnej z IPv4 np.: ”**127.0.0.1**”
l. <numer_portu> to liczba z zakresu 1024-65536

5. Należy odpowiedzieć na następujące pytania:
   m. Czy istnieje komunikacja pomiędzy programem klienta IPv4 i serwera IPv6?
   n. Jakie warunki muszą zostać spełnione, aby taka komunikacja była możliwa?
   o. Jaką postać mają podczas komunikacji przepływające datagramy IPv6 czy IPv4?
6. Zaobserwować i podać różnice wynikające z komunikacji i przebiegów datagramów dla obydwu typów protokołów – z czego one wynikają? Wnioski należy umieścić w sprawozdaniu.

3. Opracowanie i sprawozdanie

Realizacja laboratorium pt. *„Protokoły IPv4 i IPv6”* polega na wykonaniu wszystkich zadań programistycznych podanych w drugiej części tej instrukcji. Wynikiem wykonania powinno być sprawozdanie w formie wydruku papierowego dostarczonego na kolejne zajęcia licząc od daty laboratorium, kiedy zadania zostały zadane.
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,
- trudniejsze kawałki kodu, opisane tak, jak w niniejszej instrukcji,
- 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.

