Instytut Teleinformatyki

Wydział Inżynierii Elektrycznej i Komputerowej
Politechnika Krakowska

programowanie usług sieciowych

„Opcje IP i gniazda surowe”

laboratorium: 10

Kraków, 2014

Spis treści

Spis treści 2
1. Wiadomości wstępne 3
1.1. Tematyka laboratorium 3
1.2. Zagadnienia do przygotowania 3
1.3. Cel laboratorium 4
2. Przebieg laboratorium 5
2.1. Przygotowanie laboratorium 5
2.2. Zadanie 1. Ustawienie opcji IP – stream identefier 5
2.3. Zadanie 2. Wyznaczanie trasy przez nadawcę 7
2.4. Zadanie 3. Gniazda surowe – protokół UDP 8
2.5. Zadanie 4. Gniazda surowe – protokół TCP 9
2.6. Zadanie 5. Odbieranie ECHO REPLY 9
2.7. Zadanie 6. Prosty program ‘myping’. 10
2.8. Zadanie 7. Modyfikacja programu ‘myping’ 10
3. Opracowanie i sprawozdanie 12

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące **opcji IP** oraz **gniazd surowych**. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Tematyka laboratorium

Tematem laboratorium są gniazda surowe i opcje protokołu IP. Idea programowania surowych gniazd (ang. *Raw Sockets*) opiera się na ręcznej modyfikacji nagłówków poszczególnych protokołów. W takim przypadku obudowanie datagramu odpowiednimi nagłówkami, wypełnienie ich pól oraz wysłanie tak spreparowanej paczki staje się obowiązkiem programisty. Dzięki stosowaniu gniazd tego rodzaju nie jesteśmy ograniczeni stosowaniem tylko protokołów TCP czy UDP, ale mamy także dostęp do warstwy sieciowej (IP) i warstwy transportowej (TCP, UDP, ICMP). Możemy implementować własne protokoły, jak również obsługiwać te, do których normalnie nie mielibyśmy dostępu stosując gniazda SOCK_STREAM lub SOCK_DGRAM.

Opcje IP (ang. *IP Options*) to dodatkowe pole nagłówka protokołu IP. Wszystkie opcje są definiowane przez jeden podzielony na trzy części bajt. Pierwsza część jest jednobitowa i określa zachowanie opcji w przypadku fragmentacji, druga część (dwubitowa) definiuje klasę opcji, a trzecia określa numer opcji. Dzięki opcjom nagłówka IP możliwe jest ustalenie m.in. dokładnego routingu źródłowego, zapisywanie trasy datagramu czy też sterowanie opcjami bezpieczeństwa.

1.2. Zagadnienia do przygotowania

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

- Postać ramki IP; [RFC 0791]
- Nazwy i przeznaczenie opcji IP; [RFC 0791]
- Stosowanie opcji IP. [Stevens, Roz.27]

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

- Postać datagramu TCP oraz UDP; [RFC 0793]
- Stosowanie gniazd w systemie Linux; [Stevens, Roz.3]
- Stosowanie gniazd surowych. [Stevens, Roz.28]
- Atak typu SYN flood

Literatura:

[1] W.R. Stevens, *„Programowanie Usług Sieciowych”*, *„API: gniazda i XTI”*.
[2] IETF (http://www.ietf.org/), [RFC 0791], [RFC 0793], [RFC 2113], [RFC 1108].

1.3. Cel laboratorium

Celem laboratorium jest poznanie podstawowych opcji IP, które mogą zostać dołączone do nagłówka protokołu IP. Ponadto zostanie przećwiczone tworzenie i korzystanie z gniazd surowych wykorzystywanych do wysyłania i odbierania datagramów. Podczas realizacji tego laboratorium zapoznasz się z:

- z budową pola opcji dla wybranych opcji IP(IPv4, IPv6),
- z technikami modyfikacji opcji IP,
- z techniką tworzenia gniazd surowych,
- z technikami modyfikacji ramek IP, ICMP,
- z zaletami oraz wadami gniazd surowych.

2. Przebieg laboratorium

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

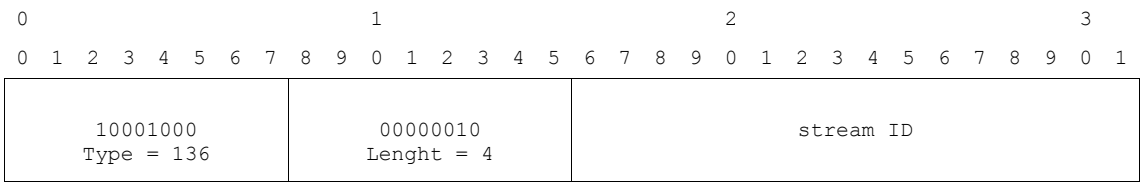
2.1. Przygotowanie laboratorium

W laboratorium wykorzystywany jest jeden komputer z zainstalowanym systemem Linux Fedora oraz oprogramowaniem Ethereal.
Przed przystąpieniem do wykonywania zadań:

1. Uruchom system Linux.
2. Zaloguj się jako root.
3. Skopiuj plik z katalogu
   `Home/shared/pus/pus_10_ip_layers`
   do katalogu `~/pus/pus_10_ip_layers` wykonując polecenie:
   `$ cp home/shared/pus/pus_10_ip_layers ~/pus/pus_10_ip_layers`
4. Uruchom program Ethereal i zapoznaj się z jego działaniem.
5. Wejdź do katalogu ~/pus/pus_10_ip_layers wykonując polecenie:
   `$ cd ~/pus/pus_10_ip_layers`

2.2. Zadanie 1. Ustawienie opcji IP – stream identefier

Celem zadania jest zapoznanie się ze sposobem i funkcją służącą do ustawiania opcji do pakietu IP na przykładzie opcji identyfikator strumienia (*ang. stream identefier*). Opcja ta wykorzystywana jest przez pakiet IPv4 do przenoszenia w sieci identyfikatora strumienia. Poniżej został przedstawiony wygląd pola opcji: identyfikator strumienia w pakiecie IP.



Opcja identyfikator strumienia(źródło: RFC 793)

Pierwszy bit opcji ustawiony na 1 powoduje powielenie opcji we wszystkich fragmentach podczas fragmentacji pakietu. Kolejne dwa bity ustawione na 0 wskazują,

że opcja dotyczy datagramu. Pięć ostatnich bitów to numer opcji, w tym przypadku wynosi 8. Zatem pierwsze 8 bitów określa typ opcji, który w tym przypadku w postaci dziesiętnej wynosi 136. Identyfikator strumienia może być 16 bitową liczbą.

Powyżej opisane ustawienia można zaobserwować w przykładowym pliku `zadanie1.c` zwracając uwagę na linię:

```
unsigned char ip_options[4] ={136,4,0,1};
```

pokazującą w jaki sposób ustawić wartości opcji o danym typie 136, długości 4 i identyfikatorze strumienia o wartości 1. Poniższy fragment przedstawia sposób ustawienia opcji:

```
setsockopt(gniazdo,IPPROTO_IP,IP_OPTIONS,            \
(unsignedchar*)&ip_options[0],4*sizeof(unsigned char));
```

W programie `zadanie1.c` stworzone zostało gniazdo surowe, poprzez które nadawany jest pakiet IP. Dla pakietu IP ustawiana jest opcja identyfikator strumienia. Pakiet ten za pośrednictwem protokołu ICMP przenosi żądanie ECHO REQUEST do wybranego hosta(w przykładzie www.onet.pl).

W celu przeprowadzenia ćwiczenia wykonaj następujące czynności:

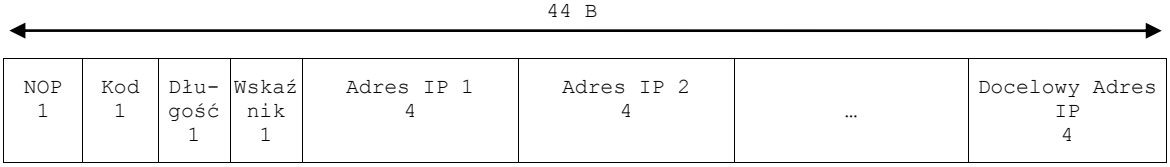
1. Włącz program Ethreal.
2. Z menu [Capture] wybierz [Options]
3. W polu [Capture Filter] wpisz:
   `icmp and dst www.onet.pl`
   co spowoduje, że przechwytywane będą wyłącznie pakiety ICMP wysyłane do www.onet.pl.
4. Uruchom przechwytywanie pakietów klikając na przycisk [Start]
5. Jeżeli nie jesteś w katalogu, w którym znajduje się źródło `zadanie1.c` to wejdź do niego wykonując polecenie:
   `$ cd ~/pus/pus_10_ip_layers/zad1`
6. Skompiluj program `zadanie1.c` poleceniem:
   `$ gcc zadanie1.c –o zadanie1`
7. Uruchom program poleceniem:
   `$ ./zadanie1 www.onet.pl`
8. Przejdź do programu Ethereal. W programie Ethereal powinien zostać przechwycony segment ICMP. Rozwijając drzewo protokołów (ramka poniżej okna przechwytywania) [Internet Protocol, src…, dst…] -> [Options(4bytes)] należy zauważyć iż ustawiona jest opcja [Stream identifier] na wartość 1.
9. Otwórz plik `zadanie1.c` programem pico lub gedit i zastąp linię:
   `unsigned char ip_options[4] ={136,4,0,1};`
   linią:
   `unsigned char ip_options[4] ={136,4,1,1};`
10. Ponownie skompiluj i uruchom program `zadanie1.c` wykonując polecenia z punktów 6 i 7.

11. Zaobserwuj przechwycony w Ethereal segment ICMP tak jak zostało to opisane w punkcie 8. Zwróć uwagę na zmienioną wartość [Stream identifier]. Spróbuj wytłumaczyć dlaczego aktualna wartość [Stream identfier] w tym przypadku wynosi 257. Ponadto zwróć uwagę na rozmiar nagłówka IP.

2.3. Zadanie 2. Wyznaczanie trasy przez nadawcę

W tym ćwiczeniu wykorzystamy program z ćwiczenia pierwszego, zmodyfikowany tak, aby pakiet ICMP był wysyłany do komputera z którego został wysłany, przechodząc przez zdefiniowaną trasę. Do przesyłania pakietów wzdłuż zdefiniowanej trasy służy opcja IP o nazwie SSRR **-** rygorystyczne trasowanie według nadawcy (*ang. strict source and record route*). Opcja ta wskazuje drogę pakietu do stacji docelowej. W przypadku tej opcji datagram IP musi przejść przez wszystkie węzły i tylko przez nie. Zatem kolejne węzły muszą być swoimi sąsiadami.

Poniżej został przedstawiony wygląd pola opcji SSRR w pakiecie IP. Zwróć uwagę na rozmiar pola opcji SSRR i zastanów się nad maksymalną możliwą do wprowadzenia w opcji ilością węzłów jakie może przejść datagram IP.



Opcja: Rygorystyczne trasowanie według nadawcy(SSRR)

Pole opcji SSRR przedstawione zostały w poniższej ramce:

Parametr/opcja	**Opis**
NOP	Nieobowiązkowe, powoduje wyrównanie położenia adresów do wielokrotności czterech bajtów
Kod	Kod opcji IP
Długość	Rozmiar pola opcji w bajtach
Wskaźnik	Wskazuje na numer bajtu adresu, który powinien być przetwarzany przez kolejny router, początkowo ustawiany na pierwszy adres IP
Adres IP 1 - …	Kolejne adresy IP przez które ma przejść pakiet
Docelowy adres IP	Adres IP docelowego hosta

1. Przejdź do katalogu zad2:

```
$ cd ~/pus/pus_10_ip_layers/zad2
```

2. Otwórz plik zadanie2.c i zapoznaj się z kodem źródłowym.
3. Zwróć szczególną uwagę na funkcję `setsockopt` i jej ostatni parametr.

```
setsockopt(gniazdo, IPPROTO_IP, IP_OPTIONS,(unsigned
char*)&ip_options[0],16*sizeof(unsigned char));
```

4. Odpowiedz na pytanie jaki powinna być wartość gdy trasa składałaby się z 6 węzłów.

```
unsigned char ip_options[16] =
{1,9,15,4,127,0,0,1,127,0,0,1,127,0,0,1};
```

5. Powyższą linię w programie `zadanie2.c` należy zmodyfikować w taki sposób, aby wysyłany pakiet dotarł do komputera docelowego (tutaj localhost), za pośrednictwem dwóch innych komputerów. Powyższa linia po modyfikacji powinna wyglądać następująco:

```
unsigned char ip_options[16] =
{1,9,15,4,x,x,x,x,y,y,y,y,127,0,0,1};
```

gdzie `x,x,x,x` oraz `y,y,y,y` są adresami IP komputerów pośredniczących w transmisji.
6. W sposób analogiczny jak w ćwiczeniu pierwszym należy skompilować i uruchomić program, a także sprawdzić poprawność jego działania przy użyciu programu Ethereal.

Opcja rygorystycznego trasowania według nadawcy w przeciwieństwie do opcji swobodnego trasowania charakteryzuje się tym, że brak możliwości podążania wskazaną trasą powoduje odrzucenie pakietu. Spróbuj zmienić adresy IP w taki sposób aby kolejne węzły nie były swoimi sąsiadami. Zaobserwuj przy pomocy programu Etehreal wynik działania programu.

2.4. Zadanie 3. Gniazda surowe – protokół UDP

W zadaniu tym należy zmodyfikować plik *example_udp.c* tak, aby utworzyć gniazdo surowe pakietu UDP oraz wypełnić poprawnie pola nagłówka UDP.
Plik znajduje się w katalogu:

```
$ ~/pus/pus_10_ip_layers/zad3
```

Po skompilowaniu programu poleceniem:

```
gcc example_udp.c –o example_udp
```


należy, przechwytując pakiety aplikacją ethereal, uruchomić program, a następnie przeanalizować jeden z interesujących nas pakietów.

2.5. Zadanie 4. Gniazda surowe – protokół TCP

W tym zadaniu należy utworzyć gniazdo surowe protokołu TCP, nagłówek IP oraz nagłówek TCP tak, aby program SYN flood'ował zadany adres IP na wybranym porcie. W trakcie ćwiczenia należy użyć pliku *example_tcp.c,* który należy odpowiednio zmodyfikować. Po skompilowaniu programu poleceniem:
Plik znajduje się w katalogu:

```
$ ~/pus/pus_10_ip_layers/zad4
```

2.6. Zadanie 5. Odbieranie ECHO REPLY

Celem zadania jest przeanalizowanie i modyfikacja programu `zadanie5.c` w taki sposób, aby możliwe było odebranie odpowiedź od serwera na żądanie echo (ang. *Echo Reply*) przy wykorzystaniu protokołu ICMP z użyciem gniazd surowych. W poniższym zadaniu zostanie przećwiczone czytanie i tworzenie pakietów ICMP.

Protokół ICMP zapewnia rodzinie TCP/IP mechanizmy informowania o błędach oraz przesyłania komunikatów sterujących. Jego pierwotnym przeznaczeniem było informowanie stacji nadawczej o zakłóceniu trasowania. Istnieje wiele funkcji pełnionych przez protokół ICMP jednakże najważniejszą z nich jest zapewnienie przesyłania sygnałów Echo Request/Echo Reply. Funkcja ta pozwala na sprawdzenie niezawodności połączenia między dwiema stacjami. Zwykle jest to realizowane za pomocą polecenia PING.
W kodzie pliku zadanie5.c brakuje definicji funkcji listener(). Zadaniem będzie napisanie definicji tej funkcji. Funkcja ta ma za zadanie obierać segmenty ICMP ECHO reply które są odpowiedzią na wysłane żądania ECHO request wysyłane przez funkcję ping().

Do implementacji tej funkcji należy wykorzystać funkcję:

```
int socket(int family, int type, int protocol);
```

za pomocą której należy stworzyć gniazdo surowe. Następnie w pętli (uwaga na ilość przejść pętli!)należy użyć funkcji:

```
int recvfrom(int s, void *buf, size_t len, int flags, struct sock-
addr *from, socklen_t *fromlen);
```

która pobiera komunikaty wysłane do gniazda. Wyniki działania programu należy wyświetlić na ekran przy pomocy zaimplementowanej w programie funkcji:


```
void display(void *buf, int bytes);
```

Przejdź do katalogu z programem:

```
$ cd ~/pus/pus_10_ip_layers/zad5
```

Skompiluj program poleceniem

```
$ gcc zadanie5.c –o zadanie5
```

a następnie uruchom

```
$ ./zadanie5 www.onet.pl
```

W kolejnej części zadania spróbuj przekształcić program taka by wyświetlał dodatkowe informacje związane ze strukturą nagłówka ICMP a więc: typ, identyfikator i numer sekwencyjny. W tym celu zapoznaj się ze struktura icmphdr (neti-net/ip_icmp.h). Dodaj także informacje ze struktury iphdr: wersja protokołu, rozmiar nagłówka, adres docelowy pakietu. Skompiluj przekształcony program i wykonaj test jego działania.

2.7. Zadanie 6. Prosty program ‘myping’.

Kolejnym zadaniem będzie przetestowania działania programu, który jest uproszczoną implementacją programu *ping*. W tym celu należy skompilować program `myping.c`, który znajduje się wśród programów wzorcowych.

1. Przejdź do katalogu z programem:

```
$ cd ~/pus/pus_10_ip_layers/zad6
```

2. Kompilacja odbywa się za pomocą polecenia:

```
$ gcc myping.c –o myping
```

Aby przetestować działanie aplikacji należy uruchomić program myping z jednym parametrem, którym jest adres docelowego hosta. Przykładowo:

```
$ ./myping onet.pl
```

Wynik działania programu prezentuje się następująco:

```
Pinging onet.pl:
Reply from 213.180.130.200: bytes=84 TTL=64
Reply from 213.180.130.200: bytes=84 TTL=64
Reply from 213.180.130.200: bytes=84 TTL=64
Reply from 213.180.130.200: bytes=84 TTL=64
```

2.8. Zadanie 7. Modyfikacja programu ‘myping’

Ostatnim zadaniem jest zmodyfikowanie znanego z poprzedniego zadania programu *myping* w taki sposób, aby wyświetlał on więcej informacji dotyczących odpytywanego hosta.

Należy skorzystać ze struktur odpowiadających za nagłówki IP i ICMP:

- `struct iphdr`
- `struct icmphdr`

Zmodyfikować należy funkcję `void display(void *buf, int bytes)`. Ponieważ bufor przekazywany do funkcji jest tablicą typu char dlatego podane wyżej struktury należy zainicjalizować w poniższy sposób:

```
struct iphdr   *ip = buf;
struct icmphdr *icmp = buf+ip->ihl*4;
```

Zadaniem jest wyświetlenie na ekranie informacji pobranych ze struktur nagłówka ICMP i IP takich jak:

- struct iphdr
  - wersja protokołu IP,
  - rozmiar nagłówka,
  - protokół,
  - adres docelowy pakietu.
- struct icmphdr
  - typ,
  - suma kontrolna,
  - identyfikator,
  - numer sekwencji.

Po przetestowaniu programu powinniśmy uzyskać następujący wynik:

```
Pinging onet.pl:

Reply from 213.180.130.200: bytes=84 TTL=64
IPv4, hdr_size=20 dest_addr=192.168.16.44
ICMP: type[0/0] checksum=49364 id=14850 seq=1
Reply from 213.180.130.200: bytes=84 TTL=64
IPv4, hdr_size=20 dest_addr=192.168.16.44
ICMP: type[0/0] checksum=49108 id=14850 seq=2
Reply from 213.180.130.200: bytes=84 TTL=64
IPv4, hdr_size=20 dest_addr=192.168.16.44
```

ICMP: type[0/0] checksum=48852 id=14850 seq=3

3. Opracowanie i sprawozdanie

Realizacja laboratorium pt. *„Opcje IP i gniazda surowe”* 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.

