Zadanie programistyczne nr 1
z Sieci komputerowych

1 Opis zadania

Napisz program `traceroute`, wyświetlający adresy IP routerów na ścieżce do docelowego adresu IP. Program powinien działać w trybie tekstowym i jego jedynym argumentem powinien być adres IP komputera docelowego.

Program powinien wysyłać pakiety ICMP *echo request* o coraz większych wartościach TTL (podobnie jak robi to wywołanie `traceroute -I`. Dla każdej wartości TTL $\in [1, 30]$ program powinien wykonać następujące operacje.

1. Wysłać 3 pakiety ICMP `echo request` z ustalonym TTL (jeden za drugim, bez czekania na odpowiedź).

2. Poczekać (co najwyżej) przez zdefiniowany w programie czas (np. 1 sekundę). Oczywiście można to czekanie połączyć z odbieraniem pakietów z następnego punktu.

3. Odebrać z gniazda pakiety, które nadeszły przez ten czas. Ewentualne odpowiedzi na wczesniejsze pakiety (z wcześniejszymi wartościami TTL) potraktować jak śmieci. Żeby nie zablokować się na odczytywaniu, można odczytywać w trybie nieblokującym, choć można też przyspieszyć oczekiwanie wykorzystując funkcję `select()`.

4. Wyświetlić adres IP routera, od którego nadejdą komunikaty i średni czas odpowiedzi w milisekundach. W przypadku braku odpowiedzi od jakiegokolwiek routera należy wyświetlić `*`. W przypadku odpowiedzi od więcej niż jednego routera należy wyświetlić wszystkie odpowiadające. W przypadku nie otrzymania trzech odpowiedzi w ustalonym czasie zamiast średniego czasu odpowiedzi należy wyświetlić `???`.

Po iteracji, w której otrzymamy odpowiedź od docelowego komputera, należy przestać zwiększać TTL i zakończyć program.

Przykładowy wynik działania programu może wyglądać następująco:

```
> ./traceroute 156.17.254.113
1. 156.17.4.254 40ms
2. 156.17.252.34 ???
3. *
4. *
5. 156.17.254.113 156.17.254.114 50ms
6. 156.17.254.113 65ms
```

Program powinien obsługiwać błędne dane wejściowe, zgłaszając odpowiedni komunikat.

1.1 Uwagi implementacyjne

1. Do wysyłania i odbierania komunikatów ICMP wykorzystaj gniazda surowe. Pamiętaj, że wymagają one uprawnień administratora (programy będą uruchamiane na maszynie wirtualnej w sali 109).

2. Wykorzystaj fakt, że na podstawie odpowiedzi można zidentyfikować do jakiego pakietu należą: komunikaty ICMP *echo reply* zawierają te same pola *identifier* i *sequence number*, zaś komunikaty ICMP *time exceeded* zawierają oryginalny nagłówek IP i 8 bajtów oryginalnego pakietu IP (czyli w w przypadku odpowiedzi na ICMP *echo request* cały oryginalny nagłówek ICMP).

3. Możesz wykorzystywać fragmenty kodu podane na wykładzie, w szczególności kawałek kodu obliczający sumę kontrolną nagłówka ICMP.

2 Uwagi techniczne

Pliki Swojemu ćwiczeniowcowi należy dostarczyć jeden spakowany plik zawierający katalog z programem. Katalog powinien zawierać:

- Kod źródłowy w C lub C++, czyli pliki `*.c` i `*.h` lub pliki `*.cpp` i `*.h`. Każdy plik `*.c` i `*.cpp` na początku powinien zawierać w komentarzu imię, nazwisko i numer indeksu autora.
- Plik `Makefile` pozwalający na kompilację programu po uruchomieniu `make`.
- Ewentualnie plik `README`.

W katalogu tym **nie** powinno być żadnych innych plików, w szczególności skompilowanego programu, obiektów `*.o`, czy plików źródłowych nie należących do projektu.

Kompilacja Kompilacja i uruchamianie przeprowadzane zostaną w 64-bitowym środowisku Linux.

Kompilacja w przypadku C ma wykorzystywać standard ISO C99 z ewentualnymi rozszerzeniami GNU (opcja kompilatora `-std=c99` lub `-std=gnu99`).

Kompilacja powinna wykorzystywać opcje `-Wall` i `-W`. Podczas kompilacji nie powinny pojawiać się ostrzeżenia.

3 Sposób oceniania programów

Poniższe uwagi służą ujednoliceniu uceniania w poszczególnych grupach. Napisane są jako polecenia dla ćwiczeniowców, ale studenci powinni **koniecznie się** z nimi zapoznać, gdyż będziemy się ściśle trzymać poniższych wytycznych. Programy będą testowane na zajęciach w obecności autora programu. Na początku program uruchamiany jest w różnych warunkach i otrzymuje za te uruchomienia od 0 do 10 punktów. Następnie obliczane są ewentualne punkty karne. Oceniamy z dokładnością do 0,5 punkta. Jeśli ostateczna liczba punktów wyjdzie ujemna wstawiamy zero. (Ostatnia uwaga nie dotyczy przypadków plagiatów lub niesamodzielnych programów).

Testowanie: punkty dodatnie Rozpocząć od kompilacji programu. W przypadku programu niekompilującego się, stawiamy 0 punktów, nawet jeśli program będzie ładnie wyglądał.

3 pkt. Uruchomić program na jednym z adresów należących do instytutu np. `156.17.4.1` a następnie na jakichś dwóch adresach zewnętrznych: np. `74.125.232.116` (google.com) i `213.180.146.27` (onet.pl). Porównać wyniki z wykonaniem polecenia `traceroute -I`. Do następnych punktów wybrać taki adres X, na którym program studenta działał poprawnie, jeśli nie działał nigdzie poprawnie, to za pozostałe punkty program również otrzymuje zero punktów.

Uruchomić na tej maszynie wiresharka i sprawdzić, czy program studenta faktycznie wysyła określone w zadaniu 3 pakiety na każdy TTL.

2 pkt. Uruchomić jednocześnie na tym samym komputerze (na tej samej wirtualnej maszynie) program `traceroute -I` X i program studenta na adresie X.

1 pkt. Uruchomić na tym samym komputerze (na tej samej wirtualnej maszynie) pinganie adresu X. Uruchomić program studenta na adresie X.

2 pkt. Uruchomić jednocześnie dwie instancje programu studenta, obie na adresie X.

1 pkt. Uruchomić jednocześnie dwie instancje programu studenta, jedną na adresie X, jedną na jakimś innym.

1 pkt. Uruchomić program studenta na adresie X i w połowie jego działania uniemożliwić odbieranie odpowiedzi na pakiety, np. wpisując `iptables -P INPUT DROP`. Program nie powinien zablokować się, lecz skończyć działanie po jakimś czasie.

Punkty karne Punkty karne przewidziane są za następujące usterki.

do -3 pkt. Zła / nieczytelna struktura programu: wszystko w jednym pliku, brak modularności i podziału na funkcjonalne części, w szczególności mieszanie funkcji które wysyłają pakiety z funkcjami które odbierają, mieszanie części wysyłającej pakiety z częścią wyświetlającą komunikaty dla użytkownika, niekonsekwetne wcięcia, powtórzenia kodu.

-1 pkt. Brak sprawdzania poprawności wywołania funkcji systemowych, takich jak `write()` czy `bind()`.

-1 pkt. Zły plik `Makefile` lub jego brak: program powinien się kompilować poleceniem `make`, polecenie `make clean` powinno czyścić program z tymczasowych obiektów (plików `*.o`), polecenie `make distclean` powinno usuwać skompilowane programy i zostawiać tylko pliki źródłowe.

-1 pkt. Niewłaściwa kompilacja: nietrzymanie się opcji podanych w zadaniu, ostrzeżenia wypisywane przy kompilacji, kompilacja bezpośrednio do pliku wykonywalnego bez tworzenia obiektów tymczasowych `*.o`.

-1 pkt. Nietrzymanie się specyfikacji wejścia i wyjścia. Przykładowo: wyświetlanie informacji diagnostycznych, wyświetlanie znaków zapytania zamiast gwiazdek, oczekiwanie na podanie adresu IP na standardowym wejściu zamiast jako argumentu.

-1 pkt. Brak sprawdzania poprawności danych na wejściu.

(-3/-6 pkt) Kara za wysłanie programu z opóźnieniem: -3 pkt. za opóźnienie do 1 tygodnia, -6 pkt. za opóźnienie do 2 tygodni. Programy wysyłane z większym opóźnieniem nie będą sprawdzane.

Marcin Bieńkowski