

Programowanie sieciowe

Instrukcja do laboratorium

LAB02

Połączenie TCP

Zadanie 1. Skopiować programy `daytimetcpcliv6.c` i `daytimetcpsrvv6.c` do katalogu domowego użytkownika student na komputerze PC, a następnie skompilować poleceniem `gcc` lub `make`.

Uwaga: Punkty od a do e wykonać w grupach dwuosobowych: komputer PC - komputer PC

a) Uruchomić serwer na jednym komputerów

b) Uruchomić klienta z adresem IPv6 serwera jako parametrem i sprawdzić działanie programu.

c) Programem `ss` znaleźć gniazda utworzone w procesie serwera (`ss -6ta sport = :daytime`).

c) Dołożyć opóźnienie w serwerze przed i/lub po funkcjach sieciowych np. `:socket`, `bind`, `listen`, `accept`, `close`, tak aby zaobserwować różne stany TCP programem `ss` (ewentualnie `netstat` i `Isof`) podczas uruchamiania serwera i komunikacji z klientem. Pomocne opcje polecenia `watch` np:

```
watch -d -n 1 "ss -6ta '( sport = :daytime or dport = :daytime )'"
```

Jakie stany TCP można zaobserwować?

e) Dołożyć opóźnienia w kliencie przed i po funkcjach, np. `socket()`, `connect()`, `close()/exit()`, tak aby zaobserwować różne stany TCP programem `ss` (`Isof`, `netstat`), podobnie jak w punkcie c).

Zadanie 2:"Podglądanie" sesji TCP - Można wybrać punkt a) lub b):

a) Za pomocą komendy `"tcpdump -i eth0 -n port 13"` prześledzić wymianę pakietów pomiędzy serwerem i klientem (programy `daytimetcpsrvv6.c` i `daytimetcpcliv6.c`). Dzielimy się na grupy dwuosobowe: serwer uruchomić na jednym komputerze i łączyć się z drugiego komputera (można także użyć programów dla IPv4: `daytimetcpsrvv4.c` i `daytimetcpcliv4.c`). W logu odnaleźć adresy, parametry oraz fazy połączenia TCP.

b) Wykonać zadanie 2a za pomocą programu `Wireshark`.

Zadanie 3:W katalogu z materiałami znajdują się cztery pary programów:

- dtc6_A i dts6_A
- dtc6_B i dts6_B
- dtc6_E i dts6_E

Pary tworzą programy klienta i serwera dla usługi daytime dla protokołu IPv6 z Zadania 1, jednak z pewnymi modyfikacjami. Dla każdej pary w jednym z programów umieszczony jest błąd, który uniemożliwia poprawną komunikację. Dla każdej pary znaleźć, czy błąd znajduje się w kliencie, czy w serwerze i ewentualnie wywnioskować, jaki to jest błąd (w jakiej sekcji programu, jakie funkcje sieciowe nie zostały uaktywnione). Do wykrycia błędu użyć narzędzi testowanych na LAB01 (np. ss i tcpdump).

Opcje gniazd

Zapoznanie się z opcjami SO_REUSEADDR, IP_TTL, IPV6_V6ONLY. Sprawdzenie jak parametry protokołu TCP są ustawiane w trakcie zestawiania połączenia w zależności od opcji gniazd i parametrów ścieżki (PATH MTU).

Zadanie 4:

Opcja SO_REUSEADDR:

- **SO_REUSEADDR** - Normalnie funkcja **bind()** zwraca błąd jeśli spróbujemy związać gniazdo z adresem, który jest aktualnie w użyciu. Załóżmy, że gniazdo o adresie 127.0.0.1 i porcie 1111 jest aktualnie w stanie TCP_FIN. Jeśli chcielibyśmy z tym adresem skojarzyć jakieś inne gniazdo nie czekając na zakończenie czasochłonnego procesu zamykania połączenia to musimy włączyć właśnie tą opcję. Ważna uwaga: **nawet SO_REUSEADDR nie pomoże jeśli jakieś gniazdo nasłuchuje (stan LISTEN) na danym adresie.**

a. Uruchomić serwer(daytimetcpsrvv6_02.c) i połączyć się z serwerem programem klienta (daytimetcpcliv6_02.c) z dowolnej lokalizacji. Zamknąć serwer. Spróbować uruchomić serwer ponownie - serwer nie powinien dać się uruchomić, dopóki w systemie gniazdo z poprzedniego połączenia, które wykorzystywało port 13, będzie w stanie TIME_WAIT.

b. Następnie ustawić opcję **SO_REUSEADDR**dla gniazda w serwerze i powtórzyć sekwencję kroków z punktu a). Czy udało się ponownie uruchomić serwer, gdy gniazdo z poprzedniego połączenia wciąż było w stanie TIME_WAIT?

Zadanie 5.Opcja **IPV6_V6ONLY** – umożliwia odbieranie i wysyłanie pakietów IPv4 na gnieździe IPv6 jeśli ustawiona na FALSE (blokuje port zarówno dla protokołu IPv4 jak i IPv6). Sprawdzenie działania opcji IPV6_V6ONLY:

a. W programie daytimetcpsrvv6_02.c odblokować w kodzie źródłowym ustawianie opcji gniazd IPV6_V6ONLY.

W grupach dwuosobowych:

b. Uruchomić serwer daytimetcpsrvv6_02.c na PC z **wyłączoną** opcją IPV6_V6ONLY (wartość **równa zero**). Sprawdzić czy można się połączyć z serwerem za pomocą interfejsu loopback(lo) na adresie 127.0.0.1 i komputera kolegi za pomocą programu daytimetcpcliv4_02.c. Sprawdzić, czy da się jednocześnie na PC uruchomić program daytimetcpsrvv4_02.c, który jest serwerem czasu dobowego dla protokołu IPv4.

c. Uruchomić serwer daytimetcpsrvv6.c lokalnie na PC z **włączoną** opcją IPV6_V6ONLY (wartość **różna od zera**). Sprawdzić czy można się połączyć z serwerem za pomocą interfejsu loopback na adresie 127.0.0.1 i z komputera kolegi za pomocą programu daytimetcpcliv4_02.c. Sprawdzić, czy da się jednocześnie na PC uruchomić program daytimetcpsrvv4_02.c, który jest serwerem czasu dobowego dla protokołu IPv4.

(Nieobowiązkowe) Zadanie 6. Opcja IP_TTL

a. Zaimplementować ustawianie opcji gniazd **IP_TTL** na wartość 16 w programie daytimetcpcliv4_02.c (**uwaga na obsługę błędów funkcji setsockopt()**) - sprawdzić komunikację pomiędzy klientem na PC i serwerem pod adresem 129.6.15.30 (ewentualnie można wybrać inny serwer z listy na stronie <http://tf.nist.gov/tf-cgi/servers.cgi>), podejrzeć programem tcpdump (lub wireshark) czy opcja została ustawiona (na jakie pola w nagłówku protokołu IPv4 opcja **IP_TTL** ma wpływ?) i sprawdzić czy komunikacja ma miejsce.

b. Powtórzyć punkt a) dla opcji IP_TTL ustawionej na 1 i 0. **W tym ćwiczeniu obsługa błędów funkcji setsockopt() jest niezbędna.** Jakie błędy pojawiają się dla TTL=0, a jakie dla opcji IP_TTL=1. Dla IP_TTL=1 sprawdzić, czy do komputerów są wysyłane pakiety ICMP związane z połączeniem TCP (np. komendą tcpdump -i eth0 -n -vv icmp or port 13). Powtórzyć przykład dla opcji IP_TTL ustawionej na wartość mniejszą od liczby ruterów na ścieżce do serwera 206 129.6.15.30 (liczbę ruterów na ścieżce sprawdzić komendą traceroute).

c. Zadanie a) i b) powtórzyć dla protokołu IPv6 - zmiany dokonywać w programie daytimetcpcliv6.c i skorzystać z serwerz listy pod adresem <http://tf.nist.gov/tf-cgi/servers.cgi>, np. **2610:20:6F15:15::27**.

Opcje SO_RCVBUF i SO_SNDBUF:

- **Opcja SO_RCVBUF** ustawia lub pobiera maksymalny rozmiar bufora odbiorczego. Przy ustawianiu (setsockopt()) jądro podwaja wartość ustawianą (wykorzystuje dodatkową przestrzeń do przechowywania struktur danych jądra). Podwójna wartość tej opcji jest

zwracana przez funkcję `getsockopt()`. Domyślna wartość tej opcji jest ustawiana z pliku `/proc/sys/net/core/rmem_default`, a maksymalna wartość tej opcji jest ograniczona przez wartość ustawioną w pliku `/proc/sys/net/core/rmem_max`. Minimalna wartość tej opcji wynosi 256. Dla protokołu TCP (zarówno dla IPv4 jak i IPv6) obowiązują wartości z pliku `/proc/sys/net/ipv4/tcp_rmem`.

- **Opcja `SO_SNDBUF`** ustawia lub pobiera maksymalny rozmiar bufora nadawczego. Przy ustawianiu (`setsockopt()`) jądro podwaja wartość ustawianą (wykorzystuje dodatkową przestrzeń do przechowywania struktur danych jądra). Podwójna wartość tej opcji jest zwracana przez funkcję `getsockopt()`. Domyślna wartość tej opcji jest ustawiana z pliku `/proc/sys/net/core/wmem_default`, a maksymalna wartość tej opcji jest ograniczona przez wartość ustawioną w pliku `/proc/sys/net/core/wmem_max`. Minimalna wartość jej opcji wynosi 2048. Dla protokołu TCP (zarówno dla IPv4 jak i IPv6) obowiązują wartości z pliku `/proc/sys/net/ipv4/tcp_wmem`

(Nieobowiązkowe) Zadanie 7. Skompilować program `daytimetcpcliv6_03.c`, w którym znajduje się przykład na odczyt i ustawianie wartości opcji `SO_RCVBUF`. Sprawdzić w kodzie źródłowym jak powinien program działać. Następnie skompilować i uruchomić serwer czasu dobowego `daytimetcpsrvv6_03.c`. Na komputerze, na którym uruchomiono serwer połączyć się programem `daytimetcpcliv6_03.c` z serwerem za pomocą adresu IPv6 na karcie sieciowej i za pomocą adresu `::1`. Następnie połączyć się z serwerem czasu dobowego na PC kolegi. Zadanie powtórzyć dla IPv4 (przykłady `daytimetcpcliv4_03.c` i `daytimetcpsrvv4_03.c`)

Sprawdzić jakie parametry gniazd są negocjowane z kilkoma serwerami daytime w sieci Internet (np. z listy pod adresem <http://tf.nist.gov/tf-cgi/servers.cgi> - używać adresów numerycznych). Zaobserwować jak (i jakie) opcje zmieniają się podczas łączenia się procesów. Jak zmieniają się parametry **`SO_RCVBUF`** i **`TCP_MAXSEG`** w zależności od parametrów połączenia (MSS)?

Sprawdzić na jakie parametry komunikacji TCP (MSS, Okno) ma wpływ zmiana parametrów gniazd: **`SO_SNDBUF`**, **`SO_RCVBUF`**, **`TCP_MAXSEG`** (zmieniać wartości w programie klienta i obserwować zmiany).

(Nieobowiązkowe) Zadanie 8. Używając programów: `daytimetcpcliv4_04.c`, `daytimetcpcliv6_04.c`, `daytimetcpsrvv6_04.c` sprawdzić, jak rozmiar bufora nadawczego w aplikacji serwera (liczba bajtów wysyłanych za jednym wywołaniem funkcji wysyłającej) wpływa na szybkość transmisji danych pomiędzy serwerem i klientem (zmieniać wartości w programie i sprawdzać uzyskiwaną przepływność). Programy zostały tak przerobione, że serwer wysyła dane przez pięć sekund do klienta z maksymalną szybkością, a klient raportuje ile przesłano danych i z jaką szybkością. Dodatkowo można sprawdzić jak parametry: **`SO_SNDBUF`**, **`SO_RCVBUF`**, **`TCP_MAXSEG`** wpływają na szybkość transmisji danych.

Do przygotowania na następne zajęcia (LAB03):

1. Wiadomości z LAB01, LAB02.

2. Wiadomości z wykładów 2, 3:

- zestawianie sesji TCP
- stany protokołu TCP
- zamykanie sesji TCP
- zamykanie sesji TCP segmentem RESET
- kiedy powstaje stan TIME-WAIT w sesji TCP i jakie ma znaczenie?
- która strona połączenia TCP przechodzi przez stan TIME-WAIT?
- jak wygenerować z programu segment RESET protokołu TCP?
- zasady działania opcji: `SO_REUSEADDR`, `IP_TTL`, `IPV6_V6ONLY`, `SO_NODELAY`, `SO_LINGER`
- na jakie główne parametry połączenia może mieć wpływ zastosowana warstwa transportowa w aplikacji sieciowej?
- jakiego typu adresowanie wspierają protokoły transportowe TCP, UDP i SCTP?
- jakie parametry określają adres aplikacji w sieci IP?
- jakie parametry opisują połączenie TCP w sieci IP?
- funkcje sieciowe: `socket()`, `bind()`, `connect()`, `listen()`, `read()` i `write()`, `recv()` i `send()`, `close()`, `shutdown()`
- różnice w działaniu serwerów iteracyjnych i współbieżnych
- obsługa sygnału `SIGCHLD`