4. `DATA`: Pakiet z danymi (K->S)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 4,
    2. identyfikator sesji: 64 bity,
    3. numer pakietu: 64 bity; liczba w porządku sieciowym,
    4. liczba bajtów danych w pakiecie: 32 bity; liczba w porządku sieciowym,
    5. dane: długość zależna od pola 3, ciąg bajtów.
5. `ACC`: Potwierdzenie pakietu z danymi (S->K)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 5,
    2. identyfikator sesji: 64 bity,
    3. numer pakietu: 64 bity; liczba w porządku sieciowym.
6. `RJT`: Odrzucenie pakietu z danymi (S->K)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 6,
    2. identyfikator sesji: 64 bity,
    3. numer pakietu: 64 bity; liczba w porządku sieciowym.
7. `RCVD`: Potwierdzenie otrzymania całego ciągu (S->K)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 7,
    2. identyfikator sesji: 64 bity.

Programy do napisania

Należy napisać dwa programy: klienta i serwer.

Serwer pobiera dwa parametry: pierwszym jest określenie protokołu, napis `tcp` lub `udp`. Serwer UDP obsługuje wersję z retransmisją i bez. Drugim argumentem jest numer portu, na którym serwer ma nasłuchiwać. Po uruchomieniu serwer ma nasłuchiwać na podanym porcie i obsługiwać połączenia zgodnie z powyższym protokołem. Dane z otrzymanych pakietów `DATA` z protokołu mają być wypisywane na standardowe wyjście (`stdout`) bezpośrednio po otrzymaniu kompletnego pakietu. Tak, jak opisano powyżej, serwer obsługuje jedno połączenie jednocześnie. Serwer nie wypisuje na standardowe wyjście żadnych informacji poza przesłanym ciągiem bajtów.

Klient pobiera trzy parametry: określenie protokołu (`tcp` / `udp` / `udpr`), adres serwera w notacji liczbowej lub jako nazwę i numer portu. Po uruchomieniu klient wczytuje zawartość standardowego wejścia do bufora (aż do napotkania końca pliku), a następnie (i dopiero wtedy) zaczyna transmisję danych zgodnie z protokołem. Po przesłaniu danych i odebraniu potwierdzenia otrzymania całego ciągu klient kończy działanie.

Ewentualne błędy związane z komunikacją należy obsługiwać, wypisując na wyjście błędów (`stderr`) komunikat zaczynający się od tekstu "ERROR:" z dołączonym opisem błędu. Klient po napotkaniu błędów łączności kończy działanie. Serwer, o ile to możliwe, przechodzi do obsługi kolejnego połączenia.

Inne błędy, np. dotyczące wczytywania pliku, alokacji pamięci itp. powinny być obsłużone analogicznie, poprzez wypisanie informacji zaczynającej się od "ERROR:" na `stderr` i ewentualne zakończenie pracy programu.

Programy mają być dostarczone w archiwum `*.tgz`, po którego rozpakowaniu i wykonaniu polecenia `make` w korzeniu drzewa katalogów powstaną pliki `ppcbs` oraz `ppcbc` zawierające odpowiednio serwer i klienta.

Rozwiązania mają być napisane w języku C/C++. Powinny kompilować się bez ostrzeżeń i wykonywać poprawnie na serwerze students.

Testy i raport

Po napisaniu programu należy przetestować jego wydajność w różnych warunkach sieciowych. Do testów można wykorzystać połączone ze sobą maszyny wirtualne. Trzeba sprawdzić prędkość przesyłania danych, ilość przesłanych danych itd., gdy zmienia się przepustowość łącza, opóźnienie, procent zagubionych pakietów, a ustawienia programu obejmują wybór protokołu, długość pakietów, długość przesyłanego pliku.

Procedurę testowania, wyniki oraz wnioski z nich wynikające należy opisać w raporcie w formacie pdf umieszczonym w głównym katalogu archiwum z rozwiązaniem pod nazwą `raport.pdf`. Raport może zawierać wykresy i nie powinien zajmować więcej niż 3 stron A4 zapisanych czcionką o rozsądnym rozmiarze.

Uwagi

Programy powinny być napisane zgodnie ze sztuką. Brak oczywistych oczekiwań wobec programów (np. że nie sformatują dysku, czy że wartości zwracane przez funkcje systemowe są sprawdzane) w treści zadania nie oznacza, że program, który ich nie spełnia, będzie uznany za dobry. Również kod programu będzie podlegał ocenie.

Protokół będzie testowany rygorystycznie, również z programami wzorcowymi. Należy dołożyć starań, żeby obsługiwać wszystko poprawnie, biorąc pod uwagę specyfikę protokołów UDP i TCP.

Testy i raport z testów jest ważną częścią rozwiązania.

Stałe `MAX_WAIT` oraz `MAX_RETRANSMITS` należy zadeklarować w pliku `protconst.h`, w którym nie powinno być innych definicji ani deklaracji.

Pytania do zadania należy kierować na adres P.Parys@mimuw.edu.pl. Odpowiedzi do najczęściej zadawanych pytań będą pojawiać się na forum.

Protokół Przesyłania Ciągów Bajtów

Zadanie polega na zaimplementowaniu poniżej opisanego protokołu oraz na wykonaniu testów wydajności swojej implementacji w różnych warunkach sieciowych.

Protokół

PPCB służy do przesyłania ciągu bajtów pomiędzy klientem a serwerem. Jako protokołu niższej warstwy używa TCP lub UDP, a połączenia korzystające z UDP mogą obsługiwać prosty mechanizm retransmisji. Trzeba więc zaimplementować trzy wersje protokołu. Ciąg bajtów jest przesyłany w paczkach o wielkości od 1 do 64000 bajtów. Każda paczka może mieć inną długość.

Komunikacja przebiega następująco (szczegółowe opisy zawartości pakietów znajdują się poniżej):

1. Nawiązanie połączenia
    - Gdy używany jest protokół TCP, klient tworzy połączenie TCP do serwera. Jeżeli serwer obsługuje już inne połączenie, to nie odbiera nowego do czasu zakończenia obsługi poprzedniego. Jeśli klientowi nie uda się nawiązać połączenia, to kończy działanie.
    - Klient wysyła pakiet `CONN`.
    - Serwer odsyła pakiet `CONACC`, jeśli przyjmuje połączenie. Serwer korzystający z TCP zawsze odpowiada w ten sposób. Serwer korzystający z UDP może być w trakcie obsługi innego połączenia, w takiej sytuacji odsyła pakiet `CONRJT`. Klient po otrzymaniu `CONRJT` kończy działanie.
2. Przesyłanie danych

    Dopóki nie zostaną wysłane wszystkie dane:

    - Klient wysyła pakiet `DATA` o niezerowej długości.
    - Serwer po otrzymaniu pakietu `DATA` sprawdza, czy pochodzi on z aktualnie obsługiwanego połączenia, czy pakiet jest kolejnym pakietem danych i czy jest poprawny. Jeśli nie, to serwer odsyła pakiet `RJT` i przestaje obsługiwać to połączenie.

      Poprawny pakiet jest potwierdzany przez serwer UDP z retransmisją poprzez odesłanie pakietu `ACC`, a serwery TCP i UDP nie wysyłają potwierdzeń.

      Klient UDP z retransmisją czeka na otrzymanie `ACC` przed wysłaniem kolejnego pakietu.
3. Zakończenie

    Po odebraniu wszystkich danych serwer TCP odsyła potwierdzenie odebrania danych `RCVD`, zamyka połączenie i przechodzi do obsługi kolejnych. Serwery UDP i UDP z retransmisją odsyłają potwierdzenie `RCVD` i przechodzą do obsługi kolejnych połączeń.

    Po wysłaniu wszystkich danych i odebraniu potwierdzenia (`RCVD`) klient TCP zamyka połączenie i kończy działanie. Klient UDP i UDP z retransmisją kończy działanie.

W prawie każdym miejscu, w którym następuje oczekiwanie na odebranie pakietu, jeśli pakiet nie będzie otrzymany w trakcie `MAX_WAIT` sekund, program ma zakończyć obsługę danego połączenia bądź zażądać retransmisji. Nie dotyczy to oczekiwania przez serwer na nawiązanie nowego połączenia, które trwa dowolnie długo.

Mechanizm retransmisji działa następująco: jeśli w okresie `MAX_WAIT` nie zostanie otrzymane potwierdzenie odebrania wysłanych danych, to należy je wysłać ponownie. Czynność tę należy powtórzyć maksymalnie `MAX_RETRANSMITS` razy, a w przypadku niepowodzenia zakończyć obsługę połączenia. Potwierdzeniem odebrania wysłanych danych jest odebranie kolejnego pakietu wynikającego z przebiegu działania protokołu. Przykładowo, potwierdzeniem odebrania pakietu `DATA` jest pakiet `ACC`, a potwierdzeniem odebrania `ACC` jest kolejny pakiet z danymi. Potwierdzeniem `CONN` jest `CONACC`. Ostatni pakiet `DATA` jest potwierdzany pakietem `ACC`, który już nie jest potwierdzany. Serwer bezpośrednio po jednokrotnym wysłaniu tego `ACC` wysyła pakiet `RCVD`, który również nie jest potwierdzany. Serwer i klient ignorują ponownie otrzymane wcześniejsze pakiety (a więc `DATA` z wcześniejszym numerem pakietu lub `CONN` w przypaku serwera, natomiast `ACC` z wcześniejszym numerem pakietu lub `CONACC` w przypadku klienta).

Pakiety przesyłane przez protokół

Przesyłane pakiety składają się z pól o określonej długości występujących bezpośrednio po sobie, bez żadnych wypełnień pomiędzy polami.

Pakiety to:

1. `CONN`: Nawiązanie połączenia (K->S)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 1,
    2. losowy identyfikator sesji: 64 bity; można je traktować jako ciąg bajtów albo liczbę,
    3. identyfikator protokołu: 8 bitów; wartości to:
        - tcp: 1
        - udp: 2
        - udp z retransmisją: 3,
    4. długość ciągu bajtów: 64 bity; liczba w porządku sieciowym.
2. `CONACC`: Akceptacja połączenia (S->K)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 2,
    2. identyfikator sesji: 64 bity.
3. `CONRJT`: Odrzucenie połączenia (S->K)
    1. identyfikator typu pakietu: 8 bitów; dla tego typu pakietów 3,
    2. identyfikator sesji: 64 bity.