6 Podsumowanie

Celem mojej pracy magisterskiej było zweryfikowanie obiegowych opinii na temat jakości i wydajności mechanizmów komunikacji sieciowej w systemach Linux i FreeBSD. Chciałem też przygotować przejrzysty opis porównawczy implementacji protokołów TCP/IP w tych systemach. Choć obszerność kodu sieciowego w jądrach badanych systemów nie pozwoliła opisać go w szczegółach, opracowania zamieszczone w niniejszej pracy wyczerpują dużą część postawionych na wstępie zadań.

Znaczna część pracy jest poświęcona analizie kodu transmisji danych. Ten obszerny kod, składający się z funkcji o rozmiarach dochodzących do 1900 wierszy, został stworzony z myślą o maksymalizacji wydajności i jest dość trudny w czytaniu. Mam nadzieję, że moje opisy okażą się pomocne przy analizie kodu pokrewnych systemów lub nowych wersji przedstawionych systemów. Dodatkową pomoc może stanowić przewodnik po organizacji kodu źródłowego, która w wielu przypadkach bywa sprzeczna z intuicją i logiką. Aby zorientować się w mnogości struktur danych używanych przez implementacje protokołów sieciowych, można skorzystać z przygotowanych na potrzeby tej pracy zestawień, opisów i diagramów najważniejszych struktur, traktując je jako punkt odniesienia. Lokując inne struktury na tle tutaj przedstawionych, łatwiej będzie nie pogubić się w ich wzajemnych powiązaniach. Diagramy przepływu sterowania, podsumowujące wykonaną analizę, ukazują schematycznie architekturę najważniejszych części podsystemów sieciowych i pozwalają umieszczać analizowane funkcje na tle modeli struktury warstwowej protokołów. Mam nadzieję, że wybrane elementy niniejszej pracy mogą stanowić wartościową pomoc dydaktyczną.

Eksperymenty wykonane w pracy pozwoliły obalić opinie o słabej wydajności kodu sieciowego jednego bądź drugiego systemu. Przeciwnie: tak implementacja Linuksa, jak i FreeBSD wykazały się dobrą wydajnością. Teoretyczną wadą opisanych rozwiązań jest pośrednie kopiowanie danych na ścieżkach nadawczej i odbiorczej. Specjalizowane biblioteki komunikacyjne są zazwyczaj pozbawione tej wady. Badania przeprowadzone w pracy pozwalają jednak sądzić, że nawet dla szybkich mediów transmisyjnych opisane implementacje działające na współczesnych komputerach PC potrafią nasycić pakietami kanał transmisyjny

w co najmniej 80 procentach. Implementacja FreeBSD wykazuje wzrost wydajności przy wzroście stosunku mocy obliczeniowej komputera do przepustowości medium transmisyjnego. W tych warunkach, w których przeprowadzono badania, dla implementacji Linuksa zaobserwowano odwrotną zależność, ale poziom nasycenia kanału transmisyjnego nadal przekraczał 75%.

Kod sieciowy FreeBSD wywodzi się z cieszącego się dobrą opinią kodu z Berkeley. Wyróżnia się niewielkimi rozmiarami i przejrzystością stosowanych struktur danych, klarowną organizacją kodu źródłowego i niewielką liczbą dobrze zdefiniowanych funkcji. Niestety, rozmiary niektórych funkcji znacząco utrudniają analizę kodu i wykrywanie błędów. Pomiary przeprowadzone dla implementacji FreeBSD dają stabilne i przewidywalne wyniki.

Na kodzie Linuksa swoje piętno odcisnął model rozproszonego, ewolucyjnego rozwoju jądra. Organizacja kodu źródłowego jest w wielu przypadkach nielogiczna, struktury danych bywają bardzo duże i niespójne, a liczba funkcji i ich podział na pliki utrudniają śledzenie ich wzajemnych powiązań. Mimo braku projektu podsystemu sieciowego jako całości, w wielu miejscach można dostrzec, że wybrane struktury danych -- np. struktura buforów sieciowych sk_buff -- zostały zaprojektowane z dużą starannością. Duży nacisk położono na uzyskanie dobrej wydajności kodu. W chwili obecnej główne kierunki ewolucji to dostosowywanie kodu do wydajniejszej pracy na maszynach wieloprocesorowych i projektowanie bardziej przejrzystych struktur danych. Wyniki pomiarów dla Linuksa 2.4 są bardziej stabilne niż dla Linuksa 2.2, choć pomiary wykonane na maszynach jednoprocesorowych wykazały niewielki spadek maksymalnej wydajności.

Obie badane implementacje wzorują swoją architekturę na modelu warstwowym ISO OSI. Obie nie przestrzegają tego modelu w sposób ścisły, w obu występują warstwy administracyjne, nie zdefiniowane w modelu referencyjnym. W obu implementacjach stosuje się niektóre z usprawnień zaproponowanych przez Vana Jacobsona, ale w żadnej nie zrezygnowano ze struktury warstwowej, choć struktura kodu FreeBSD jest w niektórych aspektach mniej złożona niż struktura kodu Linuksa. Obie implementacje wykazują największą wydajność, gdy karta sieciowa pracuje w trybie pełnodupleksowym, co oznacza, że na zwiększenie równoległości przetwarzania reagują wzrostem wydajności, zgodnie z tezami Vana Jacobsona.

Kod omawianych systemów podlega ciągłym modyfikacjom. Zmiany mają czasem charakter lokalny, innym razem architekturalny. Przyszłe wersje Linuksa i FreeBSD mogą dawać odmienne wyniki w testach wydajnościowych. W swojej pracy wykazałem, że wpływ modyfikacji na wydajność może być trudny do przewidzenia: przyspieszenie działania ścieżek nadawczej i odbiorczej kodu Linuksa 2.2.16 spowodowało spadek wydajności implementacji. Przytoczyłem kilka propozycji dalszych modyfikacji kodu Linuksa, jednak implementacja tych rozwiązań wymagałaby gruntownych zmian w architekturze podsystemu sieciowego. Można przypuszczać, że takie zmiany nie nastąpią, a do zadań wymagających szczególnie szybkiej komunikacji przy małym obciążeniu procesora będzie się wykorzystywać specjalizowane biblioteki komunikacyjne, takie jak M-VIA.