ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996

Seria: INFORMATYKA z.30 Nr. kol. 1315

Robert CHODOREK
Andrzej PACH

PROBLEMY IMPLEMENTACYJNE SZYBKICH PROTOKOŁÓW TRANSPORTOWYCH

Streszczenie. W artykule przedstawiono problemy implementacyjne szybkich protokołów transportowych oraz podano przegląd istniejących rozwiązań. Dodatkowo omówiono podstawowe zagadnienia przetwarzania protokołowego, mające wpływ na wydajność implementacji, podając przykład szybkiego protokołu transportowego XTP.

SOME ASPECTS OF IMPLEMENTATION OF HIGH SPEED TRANSPORT PROTOCOL

Summary. In this paper some aspects of high speed transport protocol implementation are presented. We present a survey of protocol implementations for high-speed networks. We describe our implementation, too.

ПРОБЛЕМЫ РЕАЛИЗАЦИИ ПРОТОКОЛОВ БЫСТРОДЕЙСТВИХ СЕТИ

Резюме. В работе представлено главные проблемы реализации протоколов быстродействих сети. Представлено тоже интересные реализации протоколов.

Pracę wykonano w ramach projektu badawczego nr 3 P406 027 06 finansowanego przez Komitet Badań Naukowych w roku 1995/96.

1. Wstęp

Szybki postęp technologii, widoczny w ostatniej dekadzie, umożliwił znaczne zwiększenie prędkości transmisji w warstwie fizycznej sieci [10]. Szybkie sieci wykorzystujące światłowody, kabel koncentryczny czy ekranowaną skrętkę przekroczyły prędkości transmisji 100 Mb/s [11][15][20][21] i osiągają prędkości rzędu 2,4 Gb/s [1]. Dużą prędkość transmisji należy jednak osiągnąć na poziomie aplikacji, lecz aby to osiągnąć, nie wystarczy tylko zastosować szybką warstwę fizyczną [2]. Sieć teleinformatyczna musi być wydajna jako całość, co wymaga zwiększenia wydajności pozostałych komponentów sieciowych. W chwili obecnej prowadzone są szeroko zakrojone prace nad szybkimi realizacjami pozostałych warstw i efektywnym przetwarzaniem informacji dokonywanym w tych warstwach. W przypadku warstwy transportowej zmierzają one w dwóch kierunkach [25]: tworzenia szybkich protokołów dostosowanych do wymagań sieci o dużych prędkościach transmisji oraz tworzenia wydajnych implementacji szybkich protokołów.

W warstwie transportowej wydajność protokołu w bardzo istotny sposób zależy od rozwiązań zastosowanych przy jego implementacji. Protokół transportowy powinien nie tylko posiadać wydajne mechanizmy sterowania przepływem, korekcji błędów czy zarządzania kontekstami i połączeniami. Musi również efektywnie współpracować z otaczającym go środowiskiem sprzętowo-programowym.

W artykule przedstawiono problemy implementacyjne szybkich protokołów transportowych. W rozdziale 2 omówiono podstawowe zagadnienia przetwarzania protokołowego, mające wpływ na wydajność implementacji, podając przykład szybkiego protokołu transportowego XTP. W rozdziale 3 przedstawiono metody implementacji protokołów transportowych: *on-host* oraz *off-host*. W rozdziale 4 przeanalizowano ograniczenia sprzętowe implementacji na przykładzie rozwiązania *on-host* lub *off-host*, zrealizowanego z zastosowaniem dodatkowego procesora RISC lub CISC uniwersalnego przeznaczenia. W rozdziale 5 przeanalizowano ograniczenia programowe implementacji, z uwzględnieniem zarówno procesów funkcjonujących w protokole, jak i wymagań narzuconych przez środowisko sprzętowo-programowe, a w rozdziale 6 podano przegląd istniejących rozwiązań.

2. Przetwarzanie protokołowe a wydajność transmisji

Wydajność transmisji w protokole transportowym w dużym stopniu zależy od wydajności odbiornika [4][16], gdyż to właśnie odbiornik musi wykonać wiele operacji związanych z testowaniem poprawności przesyłanych danych i ich integralności. Wydajność nadajnika ma w tym przypadku mniejsze znaczenie. Uproszczenie implementacji odbiornika i, co za tym idzie, znaczne zwiększenie jego wydajności można uzyskać na etapie projektowania protokołu. Przykładowo, w protokole XTP uproszczono znacznie funkcje odbiornika, przez co część operacji kontrolnych jest realizowana na zlecenie nadajnika. Nadajnik podejmuje decyzje między innymi związane z retransmisją i sterowaniem przepływem.

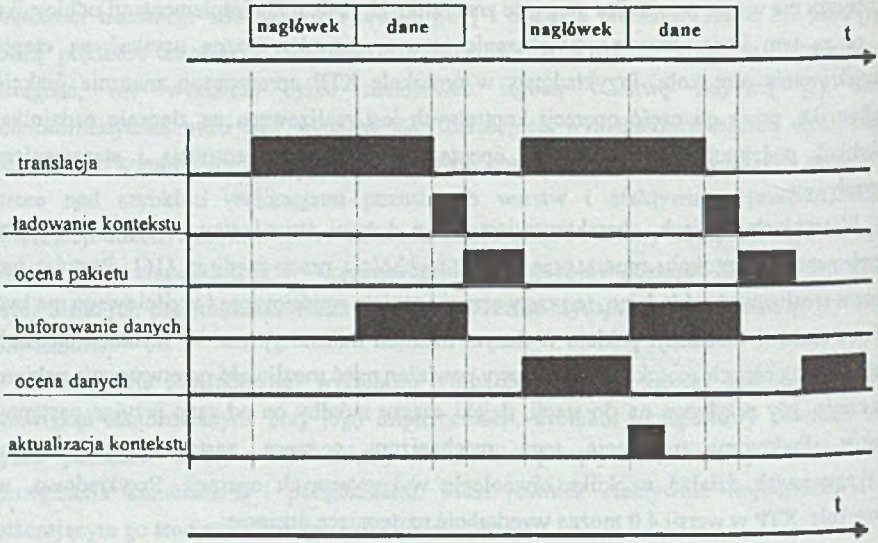
W szybkich sieciach, charakteryzujących się dużymi strumieniami danych, elementy implementacji protokołu muszą pracować z szybkością pracy medium [16]. Postulat ten można zrealizować zakładając, że przetwarzanie pakietu wejściowego i wyjściowego ma być równe czasowi transmisji pakietu w danym medium transmisyjnym. W szybkich sieciach teleinformatycznych protokół transportowy powinien mieć możliwość przetwarzania pakietu w czasie, gdy przybywa on do stacji, dzięki czemu mógłby on od razu przyjąć następny pakiet. Praktyczna realizacja tego mechanizmu wymaga zastosowania podziału wykonywanych działań na kilka równolegle wykonywanych operacji. Przykładowo, w protokole XTP w wersji 4.0 można wyodrębnić następujące operacje:

- *translacji* - poszukującej po odbiorze nagłówka odpowiedniego kontekstu,
- *ładowania kontekstu* - znalezionego w fazie translacji.
- *oceny pakietu* - na podstawie stanu kontekstu podjęta zostaje decyzja o przyjęciu lub odrzuceniu pakietu (np. pakietu powielonego),
- *buforowania danych* - w oczekiwaniu na decyzję o przyjęciu lub odrzuceniu danych,
- *oceny danych* - sprawdzenie sumy kontrolnej, przyjęcia lub odrzucenia danych,
- *aktualizacja kontekstu* - przepisanie informacji do kontekstu.

Zależności czasowe pomiędzy poszczególnymi operacjami przedstawiono na rys. 1. Jak widać na wykresach, kolejność przetwarzania operacji protokołowych pozwala na zrealizowanie w implementacji protokołu: (1) przetwarzania potokowego, (2) równoległego wykonywania części operacji:

- buforowania danych równolegle z częścią operacji translacji i ładowania kontekstu,
- operacji oceny danych równolegle z pozostałą częścią operacji translacji,
- operacji aktualizacji kontekstu równolegle z buforowaniem danych i translacją.

Przedstawione tutaj zadania można realizować w postaci specjalizowanych bloków funkcjonalnych VLSI. Przykład takiej implementacji protokołu XTP zamieszczony został w [4].



Rys. 1. Wykres czasowy kolejności zdarzeń podczas przetwarzania pakietu XTP
Fig. 1. Time diagram of XTP sequence of events

Podział zadań i ich równoległe przetwarzanie ułatwione może być także dzięki odpowiedniej budowie pakietu protokołu. W przypadku protokołu XTP pakiet posiada stałą długość nagłówka, niezależnie od opcji protokołu. Jest to konieczne w przypadku przetwarzania potokowego, ponieważ pozwala od razu, bez konieczności interpretowania nagłówka, wyznaczyć podział na nagłówek i segment informacyjny lub kontrolny.

3. Metoda implementacji

Definicja protokołu transportowego precyzuje wprawdzie zasady jego funkcjonowania, nie definiuje natomiast sposobu implementacji. Z jednej strony stwarza to pewną swobodę, pozwalającą ten sam protokół realizować w różnych środowiskach sprzętowo-programowych, posiadających różną wydajność, organizację czy system operacyjny. Z drugiej jednak strony, każda nowa implementacja wymaga zarówno osobnych testów na zgodność implementacji ze specyfikacją (ang. *conformance testing*), jak i każdorazowej optymalizacji pod kątem wydajności w danym środowisku sprzętowo-programowym.

W początkowym okresie rozwoju sieci teleinformatycznych warstwę fizyczną realizowano za pomocą specjalizowanych układów scalonych, natomiast warstwy wyższe (od 2 do 7) były realizowane programowo. Postęp w budowie układów VLSI spowodował, że obecnie popularnym rozwiązaniem jest realizacja funkcji podwarstwy MAC[1)] także za pomocą specjalizowanych układów VLSI. W realizacji wyższych warstw sieci spotyka się natomiast dwa trendy:

- realizację *on-host*, gdzie przetwarzaniem protokołowym zajmuje się procesor stacji roboczej,
- realizację *off-host*, gdzie przetwarzanie protokołowe odbywa się za pomocą: dedykowanego procesora, specjalizowanych układów VLSI lub transputerów, a procesor centralny inicjuje jedynie transmisję.

W rozwiązaniu *on-host* stosuje się jeden procesor, którego czas jest dzielony pomiędzy przetwarzanie protokołowe, system operacyjny i aplikacje.

W rozwiązaniu *off-host* przetwarzanie protokołowe przeniesione zostało do inteligentnego adaptera sieciowego, gdzie może być realizowane: (1) w postaci układów VLSI, np. układów LCA (*Logic Cell Array*) lub FPGA, (2) dodatkowego, uniwersalnego procesora jednopłytowego[2)]; spotykane rozwiązania wykorzystują jako procesor SBC procesory RISC lub CISC uniwersalnego przeznaczenia. Tego typu systemy posiadają często układy DMA i pamięć RAM o dostępie dwustronnym DP-RAM lub VRAM, co znacznie przyspiesza pracę systemu i upraszcza algorytmy przetwarzania, (3) za pomocą

[1)] dolna podwarstwa warstwy 2 modelu odniesienia OSI.
[2)] ang.: *single-board computer*, SBC.

transputerów, (4) w sposób hybrydowy, z wykorzystaniem specjalizowanych układów VLSI i procesora RISC.

Istnieją także rozwiązania hybrydowe, jak np. system z dodatkowym układem obliczającym sprzętowo sumę kontrolną i całym procesem przetwarzania realizowanym przez procesor główny.

Implementacja *off-host* będzie niezbędna w sieciach miejskich i sieciach o dużej prędkości transmisji. W sieciach lokalnych, zbudowanych w oparciu o sieć Ethernet o prędkości transmisji 10 Mb/s, wystarczające może okazać się rozwiązanie typu *on-host*, wykorzystujące nowoczesne, 32-bitowe procesory o wewnętrznej architekturze RISC i dużej częstotliwości zegara [14].

4. Ograniczenia sprzętowe

Rozwiązanie procesu przetwarzania protokołowego *on-host* lub *off-host* determinuje strukturę sprzętową implementacji. Poniżej, na przykładzie rozwiązania *on-host* lub *off-host*, zrealizowanego z zastosowaniem dedykowanego procesora RISC lub CISC, zostanie przedstawiona analiza ograniczeń sprzętowych implementacji, spośród których najważniejsze są:

- wydajność procesora; w rozwiązaniu typu *on-host* będzie to procesor stacji, w rozwiązaniu *off-host* - procesor dedykowany,
- pojemność pamięci systemu; w rozwiązaniu *on-host* - pamięci operacyjnej, w rozwiązaniu *off-host* - pamięci inteligentnego adaptera sieciowego,
- wydajność magistrali systemowej,
- wydajność adaptera sieciowego.

Zbyt mała wydajność procesora może stanowić podstawowe ograniczenie implementacji. Na wydajność procesora składa się między innymi: szybkość pracy zegara procesora, architektura wewnętrzna, interfejs z otoczeniem oraz wydajna i efektywna lista instrukcji. W przypadku oprogramowania protokołowego krytycznymi grupami instrukcji są: operacje przesłań danych, testowanie warunków, operacje logiczne i arytmetyka stałopozycyjna.

Przy wykonywaniu oprogramowania protokołowego występuje wiele operacji przesyłania danych pomiędzy wewnętrznymi strukturami danych protokołu oraz pomiędzy protokołem a jego otoczeniem - aplikacją i oprogramowaniem niższych warstw sieci. Dlatego też korzystne jest, aby procesor posiadał wydajne instrukcje przesłań blokowych, z

wydajnymi i efektywnymi trybami adresowania. W wielu sytuacjach do transferu danych stosuje się układy DMA. Jednak użycie ich jest nieefektywne w przypadku stosunkowo niewielkich struktur danych, takich jak np. rekordy kilkunasto bajtowe, ponieważ wówczas samo zaprogramowanie układu DMA będzie trwało dłużej niż operacja przekopiowania danych przez sam procesor.

Kolejnymi ważnymi instrukcjami są instrukcje logiczne i testujące pojedyncze bity czy pola bitowe. Umożliwiają one efektywne testowanie flag protokołu przy odbiorze oraz ich ustawianie w przypadku nadawania.

Drugim ograniczeniem sprzętowym jest pojemność pamięci systemu. Pamięć systemu jest dzielona pomiędzy: oprogramowanie systemowe, aplikację i oprogramowanie protokołowe z obszarem przeznaczonym na buforowanie pakietów. Z tego względu pojemność pamięci determinuje maksymalną wielkość okna i określa strategię zarządzania buforami.

Trzecie ograniczenie ze strony sprzętu to wydajność magistrali systemowej. Magistrala ma zapewnić wydajność konieczną do: transferu danych z oprogramowania protokołowego do adaptera sieciowego, aplikacji oraz, przede wszystkim, umożliwiać normalną, wydajną pracę systemu. Obecnie stosowane rozwiązania magistral systemowych posiadają wydajności:

- VME (32 bity) 40MB/s,
- ISA (16 bitów) 8MB/s,
- EISA (32 bity) 33MB/s,
- VL-bus (32 bity) 133MB/s,
- PCI (32 bity) 132MB/s

lub

- PCI (64 bity) 264MB/s.

Z tego względu konieczne jest właściwe dobranie typu magistrali systemu komputerowego, tak, aby zapewnić przepustowość odpowiednią dla danego typu adaptera sieciowego. Przy stosowaniu adapterów sieciowych typu Ethernet 10Mb/s wystarczy każdy z wymienionych wariantów magistrali. Dla szybszych adapterów sieciowych, np. FDDI, konieczna jest magistrala EISA, VME czy PCI.

Ostatnim z omawianych ograniczeń sprzętowych jest wydajność adaptera sieciowego. Adapter sieciowy jest to zestaw układów VLSI realizujących nadawanie i odbiór ramek przez sieć. Posiada on zazwyczaj własną pamięć, służącą do gromadzenia ramek przychodzących oraz ramek przygotowanych do nadania. Zawiera on także odpowiednie układy, służące do zapewnienia transmisji przez właściwe medium fizyczne. Adapter może

posiadać także własny układ DMA oraz umożliwiać scalanie i podział wiadomości. Takimi możliwościami charakteryzuje się np. sterownik sieci Ethernet oparty na koprocesorze sieci lokalnej Intel 82586 [22].

5. Ograniczenia programowe

Oprogramowanie protokołowe powinno być wydajne i powinno spełniać ściśle funkcje określone przez specyfikację protokołu. Dlatego, tworząc nową implementację protokołu transportowego, należy dokonać dokładnej analizy procesów funkcjonujących w protokole oraz uwzględnić wymagania narzucone przez środowisko sprzętowo-programowe.

Struktura oprogramowania stacji roboczej ma wpływ na inne zagadnienia implementacyjne niż miało to miejsce w przypadku sprzętu. Są to przede wszystkim zagadnienia związane ze współpracą protokołu transportowego z systemem operacyjnym stacji oraz zagadnienia zarządzania pamięcią i buforowania.

Zagadnieniem implementacyjnym, występującym zarówno w rozwiązaniu *on-host*, jak i *off-host*, jest współpraca protokołu komunikacyjnego z systemem operacyjnym. System operacyjny określa sposób organizacji procesu przetwarzania, zarządzania pamięcią i procesami, zarządzania urządzeniami we/wy [12][13]. W wielu stosowanych obecnie systemach operacyjnych zarówno oprogramowanie niższych warstw sieci, jak również sam protokół transportowy zintegrowane są z jądrem systemu operacyjnego. Takie rozwiązanie zmusza projektanta systemu transportowego do dokładnego uwzględnienia specyfiki funkcjonowania danego systemu, analizy jego struktur danych i zasad przydziału zasobów.

System operacyjny świadczy wiele typów usług. Zasadniczo korzystanie z nich jest wskazane ze względu na integralność systemu. Jednak w wielu przypadkach wywołania systemowe mogą powodować zbyt duży narzut czasowy, np. konieczne może okazać się przy tym przełączenie zadań. Najczęściej problem ten pojawia się w związku z użyciem funkcji systemowych zarządzających pamięcią. Rozwiązaniem może być wówczas przydział pewnego, dużego bufora, samodzielnie zarządzanego przez oprogramowanie protokołowe w sposób uproszczony i efektywny.

Przy tworzeniu implementacji protokołów transportowych pracujących w szybkich sieciach teleinformatycznych istotnym zagadnieniem jest problem buforowania danych. Nawet dla sieci Ethernet o prędkości transmisji 10Mb/s i obciążeniu 25% w ciągu 8 sekund zapełni się bufor o pojemności 2.5MB. Wynika stąd konieczność zastosowania odpowiednich metod buforowania, tak aby było ono szybkie i wprowadzało jak najmniejsze opóźnienie w transmisji. Aby ten cel osiągnąć, w miarę możliwości ogranicza się

czasochłonne operacje kopiowania danych pomiędzy poszczególnymi strukturami danych systemu operacyjnego a strukturami danych protokołu. Kluczową sprawą w badaniu wydajności buforowania są zagadnienia przekształcania N-wiadomości w N-ramkę i przekazanie jej do warstwy N-1. Możliwe są tutaj trzy rozwiązania:

- przepisywanie wiadomości,
- uzupełnienie N-wiadomości o kopertę N-warstwy w uprzednio zarezerwowanym miejscu,
- zapamiętywanie w niespójnym obszarze N-wiadmości i N-koperty.

Kopiowanie całej N-wiadomości do obszaru, gdzie umieszczamy N-kopertę jest rozwiązaniem modelowym. Zapewnia dużą separację warstw i niezależność implementacyjną, wymaga jednak dużego nakładu czasowego na kopiowanie. Nawet najbardziej wydajne kopiowanie z wykorzystaniem układów DMA zajmuje czas magistrali systemu komputerowego, blokując w ten sposób pracę lub zmniejszając wydajność pozostałych bloków systemu.

Drugie rozwiązanie wymaga znajomości wielkości N-kopert i rezerwowania pamięci przez użytkownika od razu z uwzględnieniem wszystkich warstw. Jest to, podobnie jak poprzednie, rozwiązanie mało efektywne, gdyż wymaga albo pełnej wiedzy o wielkościach wszystkich kopert używanych w systemie[3)], albo wywołuje specjalne funkcje rezerwujące taką pamięć.

Ostatnie rozwiązanie jest optymalne pod względem prędkości przetwarzania pakietu. Wymaga ono jednak inteligentnych adapterów sieciowych, mogących realizować scalanie wiadomości z niespójnych obszarów (np układ I82586) [22] lub posiadać zaimplementowane mechanizmy jednokrotnego scalania wiadomości w momencie ostatecznego przesyłania jej do adaptera sieciowego. Taki podział wiadomości i umieszczanie ich w niespójnych obsza- rach dobrze odpowiada strukturom systemów wielozadaniowych, gdyż każdy z takich obszarów może być oddzielnie chroniony i można zabezpieczyć poszczególne obszary przed ingerencją procedur innych warstw.

6. Przegląd istniejących rozwiązań

W szybkich sieciach teleinformatycznych zagadnieniem równie ważnym co zastosowanie szybkiego protokołu jest stworzenie jego szybkiej implementacji. W literaturze

[3)] Nie wszystkie wielkości kopert są stałe, część zależy od przesyłanych danych.

spotyka się wiele rozwiązań implementacyjnych. Najciekawsze z nich zostały zamieszczone poniżej.

Implementację w postaci układów VLSI dla protokołu XTP zaproponował Greg Chesson w ramach projektu Protocol Engine [3]. W rozwiązaniu tym wykorzystano specjalizowany procesor RISC, wykonujący część zadań protokołowych, oraz kilka specjalizowanych układów VLSI. Obliczanie sumy kontrolnej oraz układ zarządzający dostępem do pamięci DRAM zrealizowano sprzętowo. W rozwiązaniu Chessona do komunikacji z procesorem centralnym wykorzystano szybki układ DMA, współpracujący z magistralą VMEbus lub Sbus. Zaletą tej implementacji jest podział procesu przetwarzania danych protokołowych na dwa strumienie: wejściowy i wyjściowy, co pozwoliło na zrealizowanie znacznej części operacji przetwarzania w sposób potokowy.

Implementacja opracowana przez Kanakia i Cheritona dla protokołu VMTP (implementacja NAB - *VMP Network Adapter Board*) wykorzystuje bezpośrednią łączność z warstwą liniową, z pominięciem warstwy sieciowej. Została ona zaprojektowana do współpracy z siecią o prędkości 100 Mb/s. Implementacja NAB składa się z kilku komponentów. Do najważniejszych z nich należą: (1) blok kopiujący zbudowany z szybkiego układu DMA, (2) potokowy układ przetwarzania danych, (3) szybka pamięć VRAM współpracująca z blokiem kopiującym, (4) procesor protokołowy MC 68020 synchronizujący pracę modułów.

W latach 1992-1993 na University of Virginia powstała implementacja SAFENET (*Survivable Adaptable Fiber Optic Embeded Network*) [8][9], posiadająca unikalną, dwuprotokołową organizację. Zaimplementowano w niej protokoły: ISO TP4 oraz XTP. W rozwiązaniu SAFENET funkcje protokołowe realizuje wydzielony procesor MC 68040. Posiada on własną szybką pamięć DRAM i układy DMA. Komunikuje się on z procesorem centralnym Sparc za pośrednictwem magistrali VMEbus. Przez tę samą magistralę komunikuje się także z adapterem sieci FDDI.

Protokół XTP zaimplementowano na University of Virginia także dla systemu multimedialnego [18]. Jest to realizacja *on-host*, pracująca pod kontrolą systemu operacyjnego DOS. W tym rozwiązaniu protokół XTP współpracuje z kartą Ethernet lub kartą FDDI. Implementacja ta posiada stosunkowo dużą wydajność, umożliwiającą transmisję obrazów wideo w czasie rzeczywistym.

Implementacja protokołów OSI za pomocą transputerów została zaprezentowana w pracy [25]. Protokoły te zostały zrealizowane z wykorzystaniem czterech transputerów T414 z pamięcią własną 1MB. W ramach prowadzonych badań ustalono podział przetwarzania na poszczególne transputery, mający zapewnić ich równomierne obciążenie. Rozwiązanie to realizuje podstawową ideę przetwarzania równoległego procesów protokołowych.

Rozwiązania implementacyjne protokołów transportowych można także znaleźć w literaturze krajowej. Przykładowo, w [23][24] został przedstawiony opis architektury *off-host* opracowany w AGH, wykorzystujący procesor Z80 do realizacji działań protokołowych i współpracujący z koprocesorem komunikacyjnym Intel 82586.

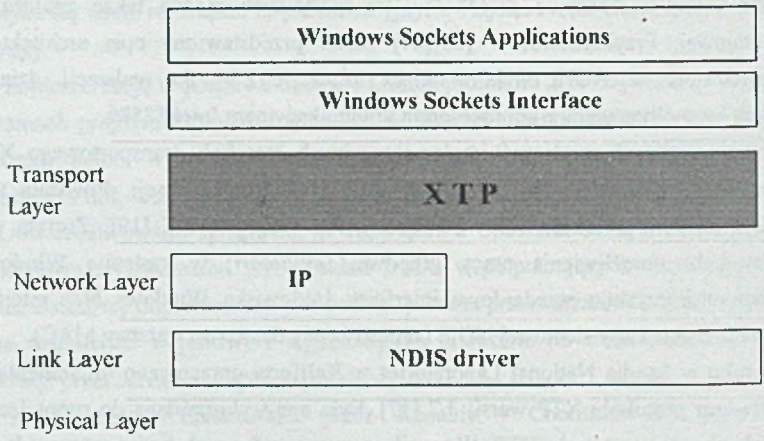
Jednym z najnowszych rozwiązań implementacyjnych protokołu transportowego XTP jest opracowanie Koreańskiego Instytutu Elektroniki i Telekomunikacji. Powstała tam implementacja *on-host* protokołu XTP dla środowiska Windows NT [19]. Została ona stworzona w celu umożliwienia pracy grupowej (*multicast*) w systemie Windows. Implementacja wykorzystuje standardowe interfejsy środowiska Windows NT: interfejs użytkownika i interfejs karty sieciowej NDIS (zapewniający dostęp do warstwy MAC).

W 1994 roku w Sandia National Laboratories w Kalifornii opracowano implementację obiektową *on-host* protokołu XTP wersji 3.7 [17]. Była ona wykorzystana do sprawdzenia nowych mechanizmów protokołu XTP. W wyniku przeprowadzonych badań wprowadzono szereg zmian do definicji protokołu i w rezultacie do opracowania wersji 4.0 protokołu XTP.

Badania protokołu XTP prowadzone są także w Katedrze Telekomunikacji Akademii Górniczo-Hutniczej w Krakowie pod kierunkiem dra hab. inż. Andrzeja R. Pacha. W latach poprzednich analizowany był tam protokół XTP w wersji 3.6, a obecnie prowadzone są prace nad wersją 4.0 [5][6][7]. Zaowocowały one powstaniem implementacji protokołu XTP w wersji 4.0, przeznaczonej do pracy w środowisku systemu operacyjnego DOS oraz Windows [6]. Jest to implementacja typu *on-host*, stworzona z wykorzystaniem techniki programowania obiektowego. Wykorzystanie programowania obiektowego pozwoliło na stworzenie wielu modułów (obiektów), które są odpowiedzialne za poszczególne procesy protokołowe, zapewniając przy tym ich dobrą integralność i separację. Daje to możliwość szybkiego i efektywnego testowania nowych mechanizmów protokołowych.

W systemie Windows stworzona implementacja odpowiada funkcjonalnie firmowej implementacji protokołu TCP. Współpracuje ona z protokołem IP, umożliwiając użycie protokołu XTP zamiast protokołu TCP. Istnieje także możliwość bezpośredniej współpracy protokołu XTP ze sterownikiem NDIS karty sieciowej (rys. 2). Aplikacje użytkownika współpracują z protokołem XTP, wykorzystując specyfikację *Windows Socket*. Pozwala to realizować zarówno bezpośrednią transmisję pomiędzy stacjami, jak i szereg usług sieciowych (służą do tego predefiniowane typy i identyfikatory gniazdek) bez konieczności znacznego modyfikowania aplikacji.

Wykorzystanie protokołu XTP znacznie zwiększyło możliwości funkcjonalne środowiska Windows przy pracy grupowej. Umożliwiło stworzenie efektywnych aplikacji typu klient-serwer, transmisji cyfrowej dźwięku czy wideo.



Rys. 2. Implementacja protokołu XTP w środowisku Windows
Fig. 2. The XTP implementation architecture

7. Zakończenie

W artykule przedstawiono podstawowe problemy występujące przy implementacji szybkich protokołów transportowych. Przeprowadzona analiza obejmuje uwarunkowania sprzętowe i programowe implementacji, z uwzględnieniem procesów funkcjonujących w protokole i wymagań narzucanych przez środowisko sprzętowo-programowe.

W artykule przeanalizowano także kilka charakterystycznych implementacji protokołów transportowych, w tym również własną implementację szybkiego protokołu transportowego XTP.

LITERATURA

[1] Biersack Ernst W., Cotton Charles J., Feldmeier David C., McAuley Anthony J., Sincoskie W. David: *Gigabit Networking Research at Bellcore.* IEEE Network Magazine, vol. 6, no. 2, pp. 42-48, March 1992.

[2] Boden Nanette J., Cohen Danny, Felderman Robert E., Kulawik Alan E., Seitz Charles L., Seizovic Jakov N., Su Wen-King: *Myrinet: A Gigabit-per-Second Local Area Network.* IEEE Micro, vol. 15, no. 1, pp. 29-36, February 1995.
[3] Cheriton D.: *VMTP - VERSATILE MESSAGE TRANSACTION PROTOCOL.* RFC 1045, Stanford University, February 1988.
[4] Chesson G.: *XTP/PE Design Considerations.* IFIP WG6.1/6.4 workshop on Protocols for High-Speed Networks, May 1989.
[5] Chodorek R.: *Sterowanie przepływem danych w protokole transportowym XTP.* I Konferencja Naukowa "Wpomaganie komputerowe w zarządzaniu", Kielce 1994.
[6] Chodorek Robert R., Pach Andrzej R.: *An implementation of the XTP transport protocol in the Windows for Workgroups environment.* PSTV'95, IFIP WG 6.1 Fifteenth International Symposium on PROTOCOL SPECIFICATION, TESTING AND VERIFICATION, Warsaw, 13-16 June 1995.
[7] Chodorek Robert R.: *Symulator protokołu XTP.* Krajowe Sympozjum Telekomunikacji KST'95, Bydgoszcz 1995.
[8] Dempsey B., Fenton J., Michel J., Waterman A., Weaver A.: *SAFENET Internals.* Computer Networks Laboratory, University of Virginia, January 7, 1993.
[9] Dempsey Bert J., Michel Jeffrey R., Weaver Alfred: *Tutorial on UVA SAFENET Lightweight Communications Architecture.* Computer Networks Laboratory, University of Virginia, January 7, 1993.
[10] Filipiak J.: *Rozwój usług i technik zintegrowanych sieci szerokopasmowych.* Krajowe Sympozjum Telekomunikacji KST'92, Bydgoszcz 1992.
[11] Goldberg L.: *100Base-T4 Chip Brings Speed To Today's LANs.* Electronic Design, pp. 180-182, Feburary 6, 1995.
[12] Madnick Stuart E., Donovan John J.: *Systemy operacyjne.* PWN, Warszawa 1983.
[13] Martyna J.: *Wstęp do projektowania systemów operacyjnych.* Uniwersytet Jagielloński, Skrypt Nr 674, Kraków 1993.
[14] Minnich R., Burns D., Hady F.: *The Memory-Integrated Network Interface.* IEEE Micro, vol. 15, no. 1, pp. 11-20, February 1995.
[15] Pach Andrzej R., Lasoń A.: *Nowoczesne sieci miejskie.* Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1994.
[16] Strayer W. T., Dempsey Bert J., Weaver Alfred C.: *Making XTP Responsive to Real-Time Needs.* Computer Science Report No. TR-89-18, December 1, 1989.
[17] Strayer W. T., Gray S., Cline R. E.: *An Object-Oriented Implementation of the Xpress Transfer Protocol.* Proc. of the Second International Worshop on Abdvanced Communications and Applications for High-Speed Networks, (IWACA'94), Heidelberg, Germany, September 26-28, 1994.
[18] Street F., Weaver Alfred C.: *A Video Mail Distibution System for Networked Personal Computers.* 18th Conference on LANs, pp. 320-326, Mineapolis, September 1993.
[19] Su-Yeon Kim: *Experience with an XTP implementation on Windows NT.* Proc. Australian Telecommunication Networks & Applications Conference, Melbourne, 5-7 December 1994.
[20] Wajda K.: *Sieci szerokopasmowe.* Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1994.
[21] Watson G., Albrecht A., Curcio J., Dove D., Goody S., Grinham J., Spratt Michael P., Thaler Patricia A.: *The Demand Priority MAC Protocol.* IEEE Network Magazine, vol. 9, no. 1, pp. 28-34, January/Feburary 1995.

[22] Wideł S.: *Konstrukcja sterownika sieci Ethernet opartego na koprocesorze sieci lokalnej Intel 82586.* Zeszyty Naukowe Politechniki Śląskiej, Seria Informatyka z.20, ss. 33-48, Gliwice 1993.
[23] Zieliński K., Indulska J., Krasowski R.: *Lokalna sieć komputerowa UMMLAN-2.* Informatyka nr 11-12, ss.8-11, 1986.
[24] Zieliński K., Indulska J., Magura-Witkowski P., Walasek T.: *Architektura oprogramowania lokalnej sieci komputerowej UMMLAN-2.* Informatyka nr 4, ss.9-12, 1988.
[25] Zitterbart M.: *High-Speed Transport Componets.* IEEE Network Magazine, vol. 5, no. 1, pp. 54-63, January 1991.

Recenzent: dr inż. Wojciech Mielczarek

Wpłynęło do Redakcji 3 stycznia 1996 r.

Abstract

New applications, such as real-time video, remount object databases or real time systems, require a new structure of the transport layer and a new type of services. The services provided by networks varies widely. Protocols, which meet these requirements, as VMTP, MMTP or XTP, are referred to as *light weight protocols*.

In this paper we present some aspects of a light weight transport protocols implementation. We discuss some design problems:

- the protocol processing - the protocol engine must accept, evaluate and buffer one packet in the time it takes for a second packet to arrive; to achieve the high data transfer rates needed, accepting a packet consist of a series of steps; the example of this sequence of events for the XTP transport protocol is shown in Fig. 1,
- the transport protocol implementation method - on-host or off-host,
- hardware and software limits for the on-host implementation and the off-host implementation based on a RISC or CISC processor.

We survey the related work and describe how each resolves the design issues presented above. Next, we describe ours implementation of the XTP transport protocol in the Windows environment (Fig.2). In this system we implemented the protocol like a firmware implementation of the TCP/IP. The implementation uses the Windows Socket Specification and protocol IP and NDIS driver for a network card.