ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1999
Seria: INFORMATYKA z. 36 Nr kol. 1414

Dariusz AUGUSTYN, Bartłomiej DŁUGOSZ
Politechnika Śląska, Instytut Informatyki

SYSTEM KOMUNIKACJI MIĘDZYPROCESOWEJ W LOKALNYCH SIECIACH KOMPUTEROWYCH

Streszczenie. Artykuł opisuje model komunikacji miedzyprocesowej w lokalnych sieciach komputerowych. Opracowanie zawiera opis systemu przesyłania komunikatów NetBC, wykorzystującego sieciowe protokoły komunikacyjne IPX i UDP/IP. W artykule przedstawione jest przykładowe zastosowanie systemu rozpraszania komunikatów w zdalnym odświeżaniu wyświetlanych danych z wykorzystaniem mechanizmów aktywnej bazy danych.

INTERPROCESS COMMUNICATION SYSTEM IN LOCAL AREA NETWORKS

Summary. This article shows simple model of an interprocess communication spread on local area networks. The paper describes an architecture of NetBC - spraying messages system, based on IPX or UDP/IP communication protocols. Remote refreshing displayed data system, based on active database mechanisms was shown as an example of using NetBC system.

1. Wprowadzenie

Wiele nowoczesnych trendów informatycznych ukierunkowanych jest na problemy związane z integracją oprogramowania ze środowiskami sieci komputerowych. Jednym z problemów jest kwestia wzajemnej komunikacji programów uruchamianych na różnych stacjach roboczych i powiadamianiu o zajściu pewnych zdarzeń.

W niniejszym opracowaniu przedstawiono rozwiązanie problemu komunikacji między procesami oparte na systemie rozgłaszania komunikatów w lokalnej sieci komputerowej. Zaproponowany system NetBC umożliwia komunikację procesom pracującym pod kontrolą

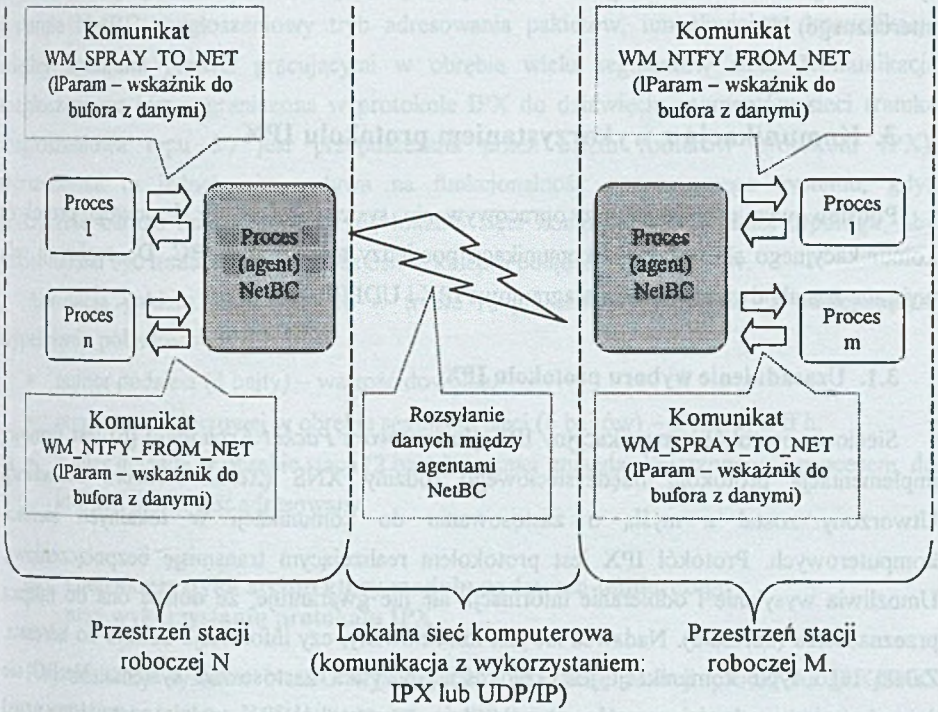
systemu operacyjnego Windows 3.x/95/NT. Program NetBC umożliwia komunikację procesom pracującym na stacjach roboczych połączonych w ramach lokalnej sieci komputerowej jedno- lub wielosegmentowej. Przesyłanie komunikatów odbywa się z wykorzystaniem sieciowych protokołów komunikacyjnych IPX lub UDP/IP.

2. Model komunikacji międzyprocesowej opartej na przesyłaniu komunikatów - zasada działania systemu NetBC

Opracowany system NetBC jest wykorzystany jako pośrednik w przekazywaniu komunikatów. Na jednej stacji roboczej uruchomiony jest tylko jeden program NetBC (zwany dalej agentem NetBC).

Komunikacja rozpoczyna się w momencie, gdy dowolny proces żąda od systemu NetBC przesłania komunikatu. Jego zadaniem jest przetworzenie informacji do odpowiedniego formatu, rozesłanie danych do wszystkich agentów NetBC, pracujących w lokalnej sieci komputerowej. Zadaniem agenta NetBC, który odebrał dane, jest poinformowanie procesów o nadejściu komunikatu. Następnie proces otrzymujący informacje o nadejściu komunikatu analizuje jego zawartość i wykonuje operacje, wynikające z interpretacji treści tego komunikatu.

W celu umożliwienia komunikacji pomiędzy dowolnymi procesami a systemem NetBC w obrębie jednego stanowiska wykorzystano mechanizm komunikatów systemu Windows 3.x/95. System operacyjny Windows (w wersjach 3.x/95/NT) oparty jest na idei przetwarzania komunikatów wysyłanych do okien aplikacji [5]. Komunikat systemu Windows składa się z następujących czterech elementów: identyfikatora okna, identyfikatora komunikatu oraz parametrów dodatkowych wParam, lParam, specyficznych dla danego rodzaju komunikatu. Aplikacje (a dokładnie okna aplikacji) otrzymują komunikaty i w odpowiedzi na nie wykonują odpowiednie akcje. Komunikacja pomiędzy dowolnymi procesami a systemem NetBC w obrębie jednego stanowiska odbywa się przez specjalnie do tego celu zdefiniowane komunikaty o nazwach: *WM_NTFY_FROM_NET* i *WM_SPRAY_TO_NET*. Rys. 1 przedstawia ideę komunikacji między procesami z wykorzystaniem programu NetBC.



Rys. 1. Schemat komunikacji między procesami z wykorzystaniem systemu NetBC
Fig. 1. Schema of interprocess communication using NetBC system

Komunikat *WM_NTFY_FROM_NET* jest wysyłany przez system NetBC do wszystkich procesów w obrębie stacji, na których jest uruchomiony. Odebranie komunikatu stanowi informacje dla uruchomionych aplikacji o nadejściu wiadomości od innego procesu. Parametr *lParam* tego komunikatu zawiera wskaźnik do bufora danych, zawierającego całą właściwą wiadomość. Aby komunikat ten został wysłany do wszystkich procesów, identyfikator odbiorcy komunikatu systemu Windows (uchwyt okna) musi mieć wartość FFFF h (zarezerwowany identyfikator dla wszystkich okien wszystkich uruchomionych aplikacji).

Komunikat *WM_SPRAY_TO_NET* będący zgłoszeniem żądania rozgłoszenia wiadomości może być wysyłany przez dowolny proces do agenta NetBC. Komunikat ten informuje agenta NetBC, że parametr *lParam* zawiera wskaźnik do bufora danych zawierającego wiadomość, którą należy dostarczyć do wszystkich procesów, pracujących w obrębie lokalnej sieci komputerowej (i w ramach danej stacji roboczej).

Jako opcję zaimplementowano również taką wersję systemu, w której komunikacja pomiędzy agentem i aplikacjami w obrębie stacji roboczej odbywa się za pośrednictwem

systemowego mechanizmu dynamicznej wymiany danych – DDE (ang. dynamic data interchanege) [5].

3. Komunikacja z wykorzystaniem protokołu IPX

Podstawowym problemem przy opracowywaniu systemu był wybór sieciowego protokołu komunikacyjnego służącego do komunikacji pomiędzy agentami NetBC. Do realizacji celu wybrane zostały dwa protokoły datagramowe IPX i UDP/IP.

3.1. Uzasadnienie wyboru protokołu IPX

Sieciowy protokół komunikacyjny IPX (*Internetwork Packet Exchange*) [6] jest firmową implementacją protokołu międzysieciowego rodziny XNS (*Xerox Network Standard*). Utworzony został z myślą o zastosowaniu do komunikacji w lokalnych sieciach komputerowych. Protokół IPX jest protokołem realizującym transmisję bezpołączeniową. Umożliwia wysyłanie i odbieranie informacji, ale nie gwarantuje, że dotrze ona do miejsca przeznaczenia (adresata). Nadawca nie jest informowany, czy informacja dotarła do adresata. Zaletą tego typu komunikacji jest szybkość. Specyfika zastosowań systemu NetBC nie wymusza niezawodności przesyłu, co powoduje, że protokół IPX spełnia postawione przed systemem wymagania. Ponadto umożliwia komunikację rozgłoszeniową poza obrębem jednego segmentu sieci.

Dodatkowym argumentem przemawiającym za wyborem tego protokołu sieciowego jest jego duża popularność ze względu na fakt, iż jest to podstawowy protokół komunikacyjny wykorzystywany w lokalnych sieciach komputerowych opartych na systemie Novell NetWare [6].

3.2. Tryb pracy systemu NetBC w protokole IPX

Mechanizmy i konstrukcja routerów protokołu IPX umożliwiają komunikację w trybie rozgłoszeniowym w obrębie pojedynczego segmentu sieci, jak i w innych jej segmentach (pakiety rozgłoszeniowe są routowane do sąsiednich podsieci). Tryb ten jest bardzo użyteczny w sytuacji, gdy trzeba przesłać jedną informację do dużej ilości stacji w sieci, gdyż pojedynczy pakiet rozgłoszeniowy dociera do wszystkich użytkowników. W protokole IPX tryb rozgłoszeniowy jest standardowo ograniczony tylko do jednego segmentu sieci. Istnieje jednak specjalny typ ramki IPX stworzony na potrzeby protokołu NetBIOS [6], którego działanie oparte jest w części na rozgłoszeniowym trybie adresowania. W protokole IPX ten typ ramki

jest oznaczony kodem 20. Wykorzystując właśnie ten rodzaj pakietu, zastosowano w systemie NetBC rozgłoszeniowy tryb adresowania pakietów, umożliwiający komunikację między agentami NetBC pracującymi w obrębie wielu segmentów sieci. Komunikacja rozgłoszeniowa jest ograniczona w protokole IPX do dziewięciu segmentów sieci (ramka rozgłoszeniowa typu 20 jest przepuszczana przez osiem routerów protokołu IPX). Ograniczenie to jednak nie wpływa na funkcjonalność opracowanego systemu, gdyż praktycznie bardzo rzadko spotyka się lokalne sieci komputerowe o takiej topologii, aby pakiet musiał być transmitowany przez osiem kolejno połączonych routerów.

Adresacja pakietu IPX typu 20 w trybie rozgłoszeniowym polega na następującym wypełnieniu pól adresowych:

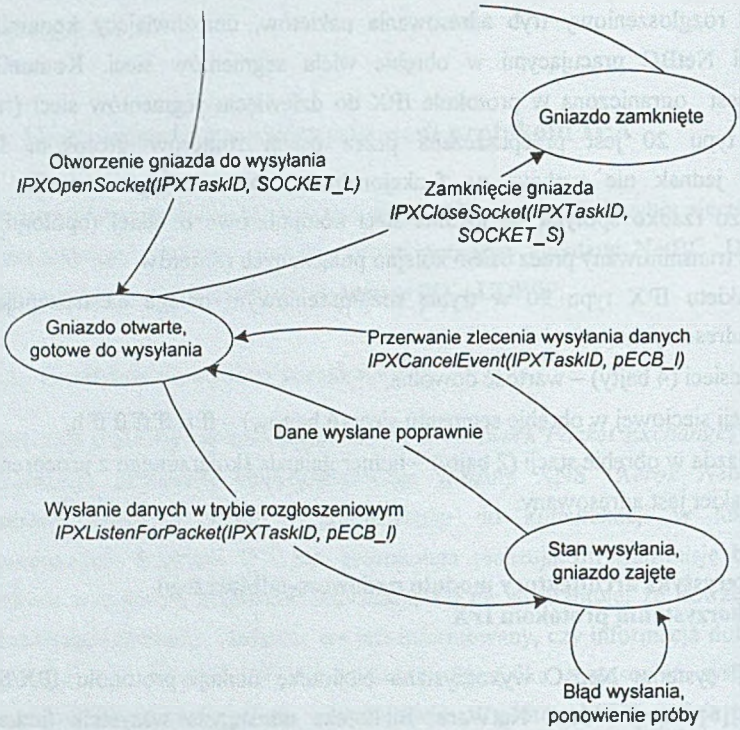
- numer podsieci (4 bajty) – wartość dowolna,
- numer stacji sieciowej w obrębie segmentu sieci (6 bajtów) – ff ff ff ff ff ff h,
- numer gniazda w obrębie stacji (2 bajty) – numer gniazda skojarzonego z procesem, do którego pakiet jest adresowany.

3.3. Charakterystyka architektury modułu nadawczo-odbiorczego przy wykorzystaniu protokołu IPX

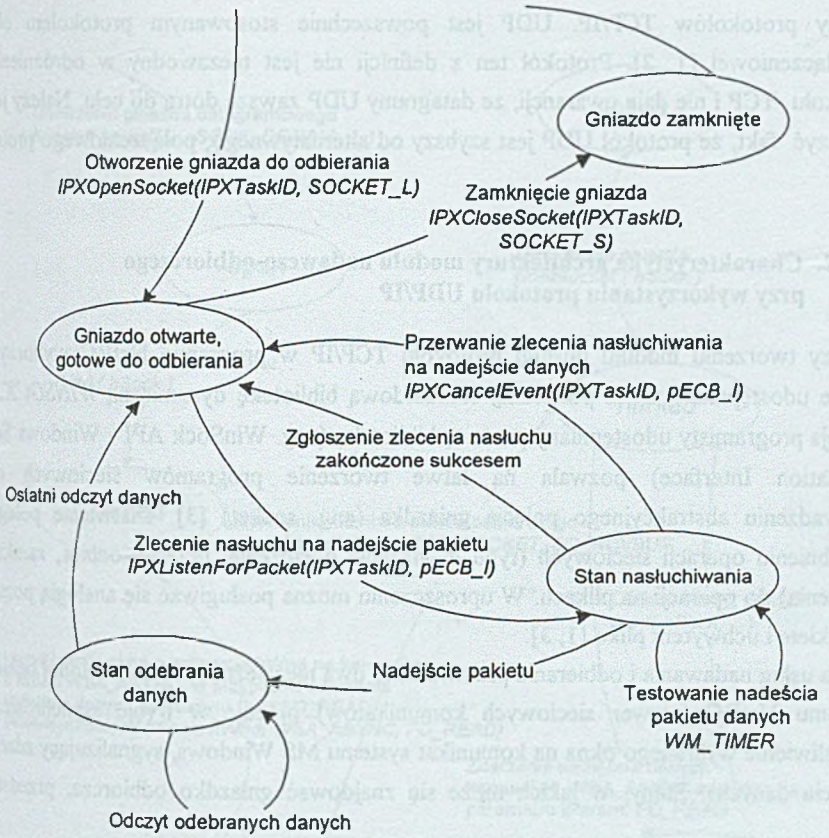
Przy realizacji systemu NetBC wykorzystano bibliotekę funkcji protokołu IPX/SPX: *NWIPXSPX.DLL* [6] firmy Novell NetWare. Biblioteka udostępnia wszystkie funkcje i definicje struktur pozwalające na komunikacje przez ten protokół. Moduł *NWIPXSPX.DLL* jest przeznaczony dla systemów operacyjnych obsługujących 16-bitowe aplikacje, co umożliwia wykorzystanie go nawet w systemach pracujących pod kontrolą systemu Windows 3.x.

Do komunikacji systemu z innymi agentami zarezerwowano dwa gniazda. Jedno przeznaczone jest tylko do wysyłania danych, natomiast drugie tylko do odbioru pakietów z danymi. Rozwiązanie to pozwoliło na rozdzielenie funkcji pomiędzy poszczególne moduły programu i poprawiło przejrzystość rozwiązania.

Zarówno moduł nadawczy, jak i odbiorczy pracują w trybie synchronicznym. Wysłanie danych następuje po zgłoszeniu żądania przez dowolny proces, natomiast nadejście pakietu jest testowane każdorazowo po nadejściu komunikatu *WM_TIMER* skojarzonego z głównym oknem aplikacji, pochodzącym od zegara systemowego. Stany, w których może się znajdować gniazdo nadawcze, przedstawia rys. 2, stany gniazda odbiorczego przedstawia rys.3.



Rys. 2. Diagram stanów gniazda nadawczego dla protokołu IPX
Fig. 2. Sending Socket States Diagram for IPX protocol



Rys. 3. Diagram stanów gniazda odbiorczego dla protokołu IPX
Fig. 3. Receiving Socket States Diagram for IPX protocol

4. Komunikacja z wykorzystaniem protokołu UDP/IP

4.1. Uzasadnienie wyboru protokołu UDP/IP

W związku z rosnącym zastosowaniem protokołu IP zarówno w sieciach rozległych, jak i lokalnych zdecydowano o wykorzystaniu tego protokołu w systemie NetBC. Specyfika zastosowań, takich jak rozgłaszanie krótkich komunikatów bez potwierdzenia, dla których program NetBC został zaprojektowany, nie wymusza niezawodności przesyłu. Stąd też idea

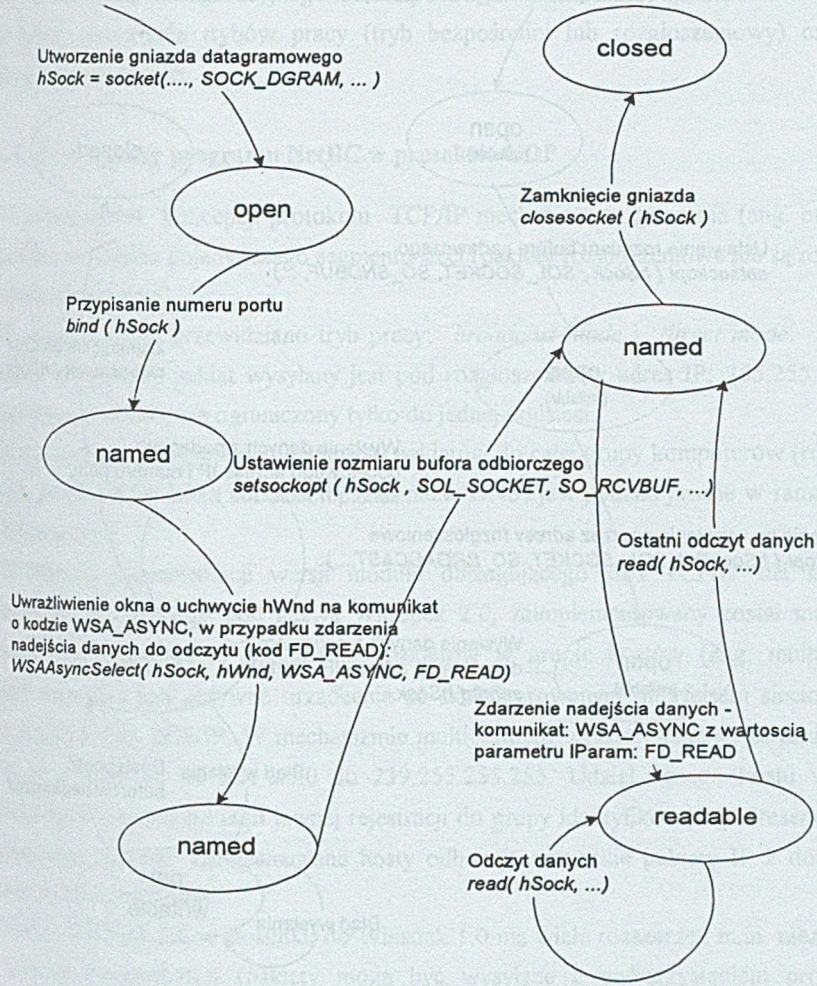
wykorzystania protokołu datagramowego UDP (ang. User Datagram Protocol) należącego do rodziny protokołów TCP/IP. UDP jest powszechnie stosowanym protokołem obsługi bezpołączeniowej [1, 2]. Protokół ten z definicji nie jest niezawodny w odróżnieniu od protokołu TCP i nie daje gwarancji, że datagramy UDP zawsze dotrą do celu. Należy jednak zauważyć fakt, że protokół UDP jest szybszy od alternatywnego, połączeniowego protokołu TCP.

4.2. Charakterystyka architektury modułu nadawczo-odbiorczego przy wykorzystaniu protokołu UDP/IP

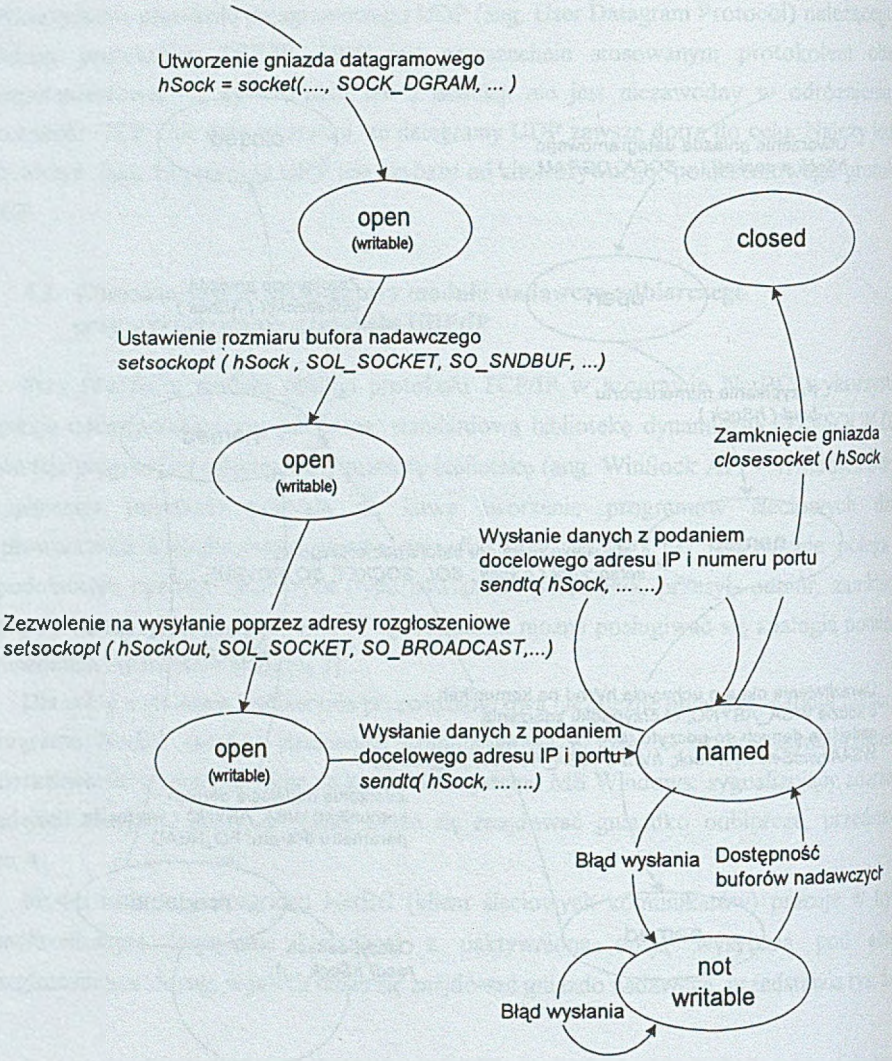
Przy tworzeniu modułu obsługi protokołu TCP/IP w programie NetBC wykorzystano funkcje udostępniane przez publiczną, standardową bibliotekę dynamiczną *WINSOCK.DLL*. Interfejs programisty udostępniany przez tę bibliotekę (ang. WinSock API - Windows Socket Application Interface) pozwala na łatwe tworzenie programów sieciowych dzięki wprowadzeniu abstrakcyjnego pojęcia gniazdka (ang. socket) [3]. Ułatwienie polega na upodobnieniu operacji sieciowych (typu nawiązanie połączenia, przesył, odbiór, zamknięcie połączenia) do operacji na plikach. W uproszczeniu można posługiwać się analogią pomiędzy gniazdkiem i uchwytem pliku [1, 3].

Dla usług nadawania i odbierania przewidziano dwa niezależne gniazdka. Moduł odbiorczy programu NetBC (serwer sieciowych komunikatów) pracuje w trybie asynchronicznym (uwrażliwienie wybranego okna na komunikat systemu MS Windows, sygnalizujący zdarzenie nadejścia danych). Stany, w jakich może się znajdować gniazdko odbiorcze, przedstawia rys. 4.

Moduł nadawczy programu NetBC (klient sieciowych komunikatów) pracuje w trybie synchronicznym (wysyłanie blokujące) z uaktywnioną opcją wysyłania pod adresy rozgłoszeniowe. Stany, w jakich może się znajdować gniazdo nadawcze, przedstawia rys. 5.



Rys. 4. Diagram stanów gniazdka odbiorczego dla protokołu UDP/IP
Fig. 4. Receiving Socket States Diagram for UDP/IP protocol



Rys. 5. Diagram stanów gniazdka nadawczego dla protokołu UDP/IP
Fig. 5. Sending Socket States Diagram for UDP/IP protocol

4.3. Parametry pracy systemu w protokole UDP

W ramach pliku konfiguracyjnego można poddać zmianom następujące parametry: numer portu UDP, ustawienia trybów pracy (tryb bezpośredni lub rozgłoszeniowy) czy tryb przezroczystości danych.

4.4. Tryby pracy programu NetBC w protokole UDP

W podstawowej koncepcji protokołu TCP/IP mechanizm rozgłaszania (ang. broadcast) działa tylko w obrębie pojedynczego segmentu sieci (pakiety rozgłoszeniowe nie są routowane do sąsiednich podsieci).

W związku z tym przewidziano tryb pracy: *broadcast mode* i *direct mode*. W trybie *broadcast* pojedynczy pakiet wysyłany jest pod rozgłoszeniowy adres IP: 255.255.255.255. Przesył taki jest szybki, ale ograniczony tylko do jednej podsieci.

W trybie *direct* następuje wysyłanie serii pakietów do całej grupy komputerów (również w różnych podsieciach). Lista adresów IP (lub nazw DNS) jest podana jawnie w ramach pliku konfiguracyjnego.

W nowszej implementacji wersji modułu obsługującego stos TCP/IP dla systemów Windows 95/NT, znanego pod nazwą WinSock 2.0, zaimplementowany został mechanizm pozwalający na routowalne rozgłaszanie pakietów w grupie hostów (ang. multicasting). Hostem nazywane jest aktywne urządzenie ze skonfigurowanym interfejsem sieciowym do pracy z protokołem TCP/IP. W mechanizmie multicastingu wykorzystane są zarezerwowane adresy IP z klasy D: od 224.0.0.0 do 239.255.255.255. Udział danego hostu w grupie multicastingowej zależy od jego jawnej rejestracji do grupy identyfikowanej adresem z wyżej wymienionego zakresu. Zarejestrowane hosty odbierają wszystkie pakiety IP z docelowym adresem multicastingowym.

Moduł WinSock 2.0 w stosunku do Winsock 1.0 ma wiele rozszerzeń m.in. niezależność od warstwy transportowej (pakiety mogą być wysyłane z wykorzystaniem protokołów TCP/IP, IPX/SPX, NetBEUI, AppleTalk). Opisany wcześniej mechanizm muticastingu nie został wykorzystany w systemie NetBC ze względu:

- na fakt, że routowanie pakietów multicastingowych zachodzi tylko wtedy, gdy routery obsługują specjalny protokół IGMP (ang. Internet Group Management Protocol). Niestety protokół ten nie jest implementowany w standardowych wersjach popularnych systemów operacyjnych, udostępniających funkcje routowania, takich jak: Novell Netware czy Windows NT.

- na założenie, że system rozgłoszeniowy powinien działać również na platformie 16 bitowej (moduł WinSock 2.0 może być wykorzystany na platformach systemowych Windows 95 i Windows NT).

4.5. Przezroczystość danych

Ze względu na zapewnienie ewentualnej przyszłej współpracy z różnymi platformami sprzętowo-programowymi (w szczególności różnymi od platformy Windows/INTEL) zastosowano opcjonalny, prosty mechanizm zapewnienia przezroczystości danych. Implementacje poszczególnych, podstawowych typów danych dla różnych platform sprzętowo-programowych mogą się różnić rozmiarem lub kolejnością bajtów. Przykładowo, kolejność bajtów w implementacji czterobajtowego typu całkowitego w systemie Windows NT/ INTEL jest odwrotna od kolejności w systemie Solaris/SPARC.

Pola typu *Double Word* wchodzące w skład komunikatu rozpraszanego w sieci komputerowej mogą być tłumaczone z postaci charakterystycznej dla lokalnej architektury systemowej (tutaj INTEL) z/na uniwersalny format sieciowy (poprzez wykorzystanie funkcji z WinSock API: *htonl*, *ntohl* - ang. host to network long, network to host long).

5. Zastosowania programu NetBC

5.1. Propozycja formatu komunikatu, wykorzystywanego do zdalnego odświeżania wyświetlanych danych

Mechanizm rozpraszania komunikatów został stworzony głównie z myślą o umożliwieniu odświeżania danych wyświetlanych na formatkach. Chodzi o zapewnienie uaktualnienia wartości danych na formatkach w momencie zdarzenia modyfikacji tych danych przez inne zadanie w innym węźle sieci komputerowej. Przyjęto, że komunikat składa się z pięciu pól *Double Word* (o nazwach: *Header, Command, Arg1, Arg2, Arg3, Arg4*) i jednego pola w postaci łańcucha znakowego o długości 256 znaków (o nazwie *Mess*).

Dla realizacji tych założeń dotyczących rozpraszania komunikatów o odświeżaniu przyjęto następujące ustalenia precyzujące zastosowanie poszczególnych pól komunikatu:

- *Header* - wartość 1 (oznaczająca komunikat, że mechanizm będzie stosowany do odświeżania danych),
- *Command* - wartość 1 (oznaczająca prezentowaną właśnie jedną z metod odświeżania),
- *Arg1* - numer systemu,

- *Arg2* - 32-bitowa mapa (każdemu z 32 bitów odpowiada jedna z tablic bazy danych, dla której należy uaktywnić mechanizm odświeżania),
- *Arg3* - dodatkowa 32-bitowa mapa (następne 32 bity, jeżeli liczba 32 bitów z pola *Arg2* jest niewystarczająca).

Arg 1 - określa identyfikator systemu informatycznego, a dokładniej identyfikator bazy danych. Zakłada się bowiem, że w ramach eksploatowanego systemu informatycznego może być wykorzystywanych wiele baz danych (obsługiwanych przez jeden lub wiele komputerów), a pojedyncze aplikacje mogą się łączyć jednocześnie z wieloma bazami danych (ang. multidatabase application).

Arg 2 - precyzuje zbiór tablic podlegających odświeżaniu. Aplikacja, w ramach której zrealizowano modyfikacje, wysyła do innych aplikacji informacje o zrealizowaniu modyfikacji zawartości pewnych tablic. Przyjęto, że tablice posiadają swoje unikalne numery w ramach bazy danych. Ustawienie „1" na odpowiednim bicie (o określonym numerze) w mapie bitowej oznacza żądanie odświeżenia odpowiedniej tablicy.

Na ogół tylko cześć tablic bazy danych zawiera dane krytyczne, tzn. takie, których obraz powinien być odświeżany na poszczególnych stacjach. Przykładowo, słowniki (tablice wykorzystywane do walidacji zawartości innych tablic) na ogół nie podlegają częstym zmianom i w przypadku ich zmian nie wymaga się od aplikacji natychmiastowego uaktualnienia ekranu. Jednak w sytuacji niewystarczającej długości pola *Arg2* wykorzystać można kolejne pola typu Double Word: *Arg3*, *Arg4* czy nawet pole 256-znakowe *Mess*, wykorzystując kolejne bajty tego ostatniego jako odpowiednie mapy bitowe.

Połączenie pól *Arg1* i *Arg2* precyzuje jednoznacznie tablicę w całym systemie informatycznym (nawet systemie heterogenicznym w sensie systemów zarządzania bazą danych).

5.2. Rozszerzenia architektury aplikacji klienta

W celu uaktywnienia mechanizmu rozpraszania komunikatów odświeżania danych należy nieznacznie zmienić budowę aplikacji. Oczywiście, celem jest minimalizacja zmian kodu w programu.

W uproszczeniu można przyjąć, że aplikacja dokonuje pewnych operacji na obiektach, będących pojęciem na wyższym poziomie abstrakcji niż tablica bazy danych (można tak założyć nawet, jeżeli w aplikacji nie ma jawnej implementacji obiektów z dziedziny przedmiotowej). Zawsze bowiem można określić obiekt poprzez skojarzenie z pewnym zbiorem tablic (obiekt określony na kolumnach z wybranych tablic). Zbiory tablic, na których określone są poszczególne obiekty, nie muszą być rozłączne.

Pojedyncza funkcja aplikacji na ogół związana jest z realizacją sekwencji pewnych okien-formatek, na których realizowane są odpowiednie wyświetlenia i edycje danych. W ramach danej formatki na ogół realizowana jest edycja danych związanych z pewnym obiektem (lub obiektami).

W proponowanym rozwiązaniu każdą formatkę wyposaża się we własne zmienne odpowiadające numerowi systemu (wartość stała na ogół dla całej aplikacji - odpowiednik *Arg1*) i mapie bitowej (wartość charakterystyczna dla formatki - odpowiednik *Arg2*). W przypadku obiektowej implementacji koszt takiej rozbudowy jest niewielki dzięki dziedziczeniu wielobazowemu, gdyż każdą formatkę rozszerza się o zmienne instancyjne (charakterystyczne tylko dla tej formatki), bez modyfikacji istniejącego kodu, związanego z daną formatką. Zadaniem programisty jest jedynie ustawienie właściwych wartości w mapie bitowej (*Arg2*), która opowiada obiektowi, na którym działa dana formatka, lub inaczej - odpowiada zbiorowi numerów tablic, których zawartość formatka modyfikuje. Takie ustawienie powinno mieć miejsce jedynie raz, w momencie otworzenia formatki (zdarzenie utworzenia okna). W przypadku modyfikacji danych na formatce aplikacja (za pośrednictwem agenta NetBC) automatycznie rozsyła w sieci komputerowej (i w obrębie lokalnej stacji roboczej) informacje o zmodyfikowanych tablicach (o zmodyfikowanym obiekcie). Inne aplikacje (w innych węzłach sieci) odbierają tę informację (za pośrednictwem swoich agentów NetBC), zapamiętują w zmiennych klasowych (tych samych, wspólnych dla wszystkich formatek całej aplikacji) i rozsyłają komunikaty systemu MS Windows do wszystkich swoich otwartych formatek. W ramach obsługi tego komunikatu wszystkie formatki automatycznie dopasowują swoją mapę bitową (zmienna instancyjna) do mapy bitowej informacji, która przyszła z innej stacji (zmienna klasowa). Jeżeli realizacja operacji „bitowego i" (ang. bitwise AND) na obu mapach bitowych daje „1" na którymkolwiek bicie, oznacza to pokrywanie się obiektów zmodyfikowanego w innej aplikacji i wyświetlanego w danej aplikacji, co w rezultacie powoduje odświeżenie danych na formatce.

Idea wiązania wielu tablic (w ramach danego obiektu) ma m.in. w konsekwencji prowadzić do minimalizacji przesyłów informacji między aplikacjami. Można oczywiście rozważyć wariant, w którym modyfikacja zawartości pewnej liczby tablic w ramach obsługi formatki pociąga za sobą przesył całej serii komunikatów z osobnymi żądaniami odświeżenia obrazu każdej z tablic.

5.3. Wykorzystanie mechanizmów aktywnej bazy danych

W rozwiązaniu przedstawionym w poprzednim rozdziale komunikacja odbywa się wyłącznie pomiędzy aplikacjami i z inicjatywy jednej z aplikacji. Można jednak rozważyć inny wariant, w którym do systemu wymiany informacji włączony będzie program serwera bazy

danych. W wariancie takim można wykorzystać wyzwalacze (ang. trigger) i procedury bazy danych (ang. stored procedures) [4]. W nowoczesnych implementacjach serwerów baz danych możliwe jest wykonywanie procedur w momencie wystąpienia zdarzeń modyfikacji typu: *INSERT, UPDATE, DELETE.* Niestety, w większości przestrzeń adresowa działania procedur jest ograniczona tylko do procesu serwera bazy danych i praktycznie nie można wykonać żadnych akcji wykraczających poza serwer bazy danych, w środowisko systemu operacyjnego. W eksperymentalnej instalacji wykorzystano serwer bazy danych SQLBase firmy CenturaSoftware w wersji 7.0 dla systemu operacyjnego Windows NT.

Ta najnowsza wersja serwera bazy danych umożliwia (poprzez mechanizm funkcji zewnętrznych - ang. external procedures) wywołanie z wnętrza procedury bazy danych funkcji pochodzącej z dowolnej dynamicznej biblioteki (ang. DLL – Dynamic Link Library). Dzięki temu np. poprzez wywołania funkcji systemowych (np. z systemowych bibliotek: *Kernel32.exe, User32.exe*) możliwe jest wykonywanie różnych akcji w środowisku systemu operacyjnego.

W instalacji eksperymentalnej poprzez mechanizm triggerów uwrażliwiono serwer bazy danych na zdarzenia modyfikacji wybranych tablic. Wybrana aplikacja realizując pewną operację (wykonując zadanie SQL) aktywuje jedno ze zdarzeń *INSERT, UPDATE, DELETE.* Wtedy serwer bazy danych wykonując procedurę związaną z triggerem komunikuje się z agentem NetBC i staje się inicjatorem przesyłu informacji o żądaniu odświeżenia danej tablicy. Aplikacje na stacjach roboczych są jedynie odbiorcami takiej informacji. Zaletą rozwiązania jest scentralizowany (łatwy do rekonfiguracji) rozsył komunikatów. Wady są następujące:

- rozsyłanie żądań odświeżenia dotyczących pojedynczych tablic (brak rozsyłu grupowego, dotyczącego obiektu złożonego z kilku tablic),
- możliwość wysyłania nadmiarowych informacji o odświeżeniu, w sytuacji gdy transakcja, w której były aktywowane triggery, została wycofana.

6. Podsumowanie

System NetBC ma stanowić rozwiązanie problemu rozsyłania komunikatów pomiędzy niezależnymi aplikacjami. Wykorzystany został przede wszystkim w celu efektywnego odświeżania uaktualnionych danych wyświetlanych na różnych stacjach roboczych w sieci komputerowej. Pracuje na platformach Windows 3.x/95 i wykorzystuje najpopularniejsze protokoły IP i IPX. Powstał ze względu na brak takiego mechanizmu w systemach komercyjnych. Zastosowany został również jako narzędzie pomocnicze w badaniach

efektywnościowych (przy pomiarach czasu wykonania) do synchronizacji startu procesów równocześnie uruchamianych na różnych stacjach roboczych.

LITERATURA

1. Quinn B., Shute D.: Windows Socket Network Programming. Addison-Wesley Publishing Company, Inc., Massachusetts 1995.
2. HeyWood D., Scimger R.: Networking with Microsoft TCP/IP. New Riders Publishing Indianapolis 1997.
3. Stevens W. R.: Programowanie zastosowań sieciowych w systemie UNIX. WNT, Warszawa 1995.
4. Boring J., Gerber D.: SQLBase Advanced Topic Guide. SQL Design, Inc. Lakeland 1995.
5. Petzold C., Yao P.: Programming Windows 95. Microsoft Press, A Division of Microsoft Corporation, Washington 1996.
6. Praca zbiorowa: NetWare Software Developers Kit 13. Novell Inc., Ohio 1996.

Recenzent: Dr inż. Maciej Bargieński

Wpłynęło do Redakcji 12 kwietnia 1999 r.

Abstract

This article presents network broadcasting system NetBC - simple system for an interprocess communication in local area networks. The principles of NetBC architecture based on datagram network protocols IPX and UDP/IP was described. The problem of routing broadcasting network packets through multiple network segments was discussed in the paper. Remote refreshing of application's screens was presented as an example of usage NetBC system. The paper shows a variant of refreshing screens system based on triggers and stored procedures which sprays messages by process of a database server.