

Mag

Mikrokomputery w sieci KASK

Wrocław 1989

1749472/1



PRACE NAUKOWE POLITECHNIKI WROCŁAWSKIEJ

No. 7	Scientific Papers of the Computer Centre of the Technical University of Wrocław	No. 7
Studies and Research	No. 4	1989

Microcomputers in the KASK network

Prace Naukowe Centrum Obliczeniowego
Politechniki Wrocławskiej

7

Seria:
Studia i Materiały

4



Mikrokomputery w sieci KASK



Wydawnictwo Politechniki Wrocławskiej · Wrocław 1989

Recenzenci

Marian BUDKA

Czesław DANIŁOWICZ

Redaktorzy naukowci

Elżbieta KOSMULSKA-BOCHENEK

Witold KOMOROWSKI

Opracowanie redakcyjne

Maria KOPEĆ

Korekta

Barbara WACHOWSA

© Copyright by Wydawnictwo Politechniki Wrocławskiej, Wrocław 1989

WYDAWNICTWO POLITECHNIKI WROCŁAWSKIEJ

Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

ISSN 0860-1623

Nakład 200+75 egz. Ark. wyd. 5. Ark. druk. 4^{1/2}. Papier offset. kl. III, 70 g, B1.

Oddano do druku w październiku 1989 r. Druk ukończono w listopadzie 1989 r.

Zakład Graficzny Politechniki Wrocławskiej. Zam. nr 1461/89. Cena zł 300,-

PRZEDMOWA

Zeszyt zawiera przegląd prac prowadzonych w Centrum Obliczeniowym Politechniki Wrocławskiej nad wdrożeniem mikrokomputerów klasy IBM PC/XT/AT do Krajowej Akademickiej Sieci Komputerowej KASK. W pierwszych dwóch artykułach omówiono sposób wykorzystania mikrokomputera klasy IBM PC jako terminala innego komputera, jako mikrohosta sieci WAN oraz jako stacji roboczej lokalnej sieci komputerowej.

Dwie kolejne prace są poświęcone pomiarom w sieci i stosowanej przy tym aparaturze oraz problemowi testowania zgodności implementacji protokołu ze standardami.

Osobne zagadnienie stanowi projektowanie instalacji lokalnej sieci komputerowej w instytucji, któremu poświęcony jest następny artykuł.

Ostatnia praca porusza problemy związane z konstrukcją oprogramowania sieciowego.

Elżbieta Kosmulska-Bochenek

Wrocław, marzec 1988

*Mikrohost,
LAN, WAN*

Krzysztof JANCZEWSKI*

Elżbieta KOSMULSKA-BOCHENEK*

FUNKCJE MIKROKOMPUTERÓW W SIECI KASK

Na tle sieci WAN i LAN KASK dokonano przeglądu zastosowań mikrokomputerów w sieciach. Przedstawiono funkcje jakie mogą one w nich spełniać oraz wymagania dotyczące podłączenia ich do systemu. Omówiono zastosowania mikrokomputera IB/PC jako: terminala, mikrohosta WAN, stacji w sieci LAN i bramy (gateway) łączącej sieci WAN i LAN.

1. WSTĘP

Do roku 1986 w uczelniach wyższych w kraju zainstalowane były przede wszystkim komputery ODRA 1305 i R-32 jako systemy wielodostępne. Na podstawie tych systemów powstało wiele zasobów programowych i baz danych, często specjalistycznych, wykorzystywanych w badaniach, dydaktyce i zarządzaniu szkołą.

Począwszy od 1986 roku w uczelniach zaczęły pojawiać się mikrokomputery typu IBM/PC i ich liczba stale wzrasta. Łatwość instalowania, duża niezawodność, możliwość rozszerzeń i bogate oprogramowanie sprawiają, że sprzęt ten jest atrakcyjny. W kraju dostępne są mikrokomputery klasy IBM/PC (Personal Computer) typu XT (procesor INTEL 8088), AT (procesor INTEL 80286) oraz 32-bitowe z procesorem INTEL 80386.

Każdy z omówionych mikrokomputerów może być zainstalowany w taniej i ubogiej konfiguracji lub też w drogiej i rozbudowanej (większe pamięć-

* Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

ci RAM, dyski typu Winchester, drukarki od prostych do laserowych, koprocesory, dodatkowe karty rozszerzeń itp.). Mikrokomputery IBM/PC, eksploatowane na ogół w systemie operacyjnym PC(MS)-DOS, są z założenia jednostanowiskowymi stacjami roboczymi. Organizowanie z zastosowaniem tych komputerów wielodostępu jest rozwiązaniem sztucznym i może być wykorzystane tylko do wybranych zastosowań.

W sytuacji współistnienia różnego typu komputerów (dużych, mini- i mikro-) wyłonił się problem ekonomicznego wykorzystania ich zasobów, zarówno sprzętowych, jak i programowych. Najlepszym rozwiązaniem jest łączenie pojedynczych komputerów w lokalne sieci komputerowe LAN (Local Area Network), a te z kolei w sieci rozległe WAN (Wide Area Network).

Atrakcyjność sieci komputerowych rośnie w miarę rozwoju świadczonych przez nie usług. Rozwój tych ostatnich jest jednak ściśle uzależniony od sprzętu, jakim dysponuje sieć i sposobu jego wykorzystania.

2. SIECI KOMPUTEROWE W OŚRODKACH AKADEMICKICH

Ośrodki akademickie są naturalnym miejscem rozwoju sieci komputerowych zarówno w sferze zastosowań, jak i ich budowy. Wynika to z dwóch podstawowych funkcji spełnianych przez uczelnie: prowadzenia zajęć dydaktycznych i badań naukowych oraz faktu, że wykorzystanie nowoczesnej technologii i rozwiązań do realizacji tych zadań jest gwarancją spełnienia celów postawionych przed nauczaniem i pracami naukowymi.

Zadaniem specjalistycznych ośrodków informatycznych na uczelni jest śledzenie prac renomowanych firm produkujących sprzęt i oferujących nowe rozwiązania systemów mikrokomputerowych oraz współpraca z przemysłem przez prowadzenie odpowiednich badań i rozwijanie nowych produktów.

Sieci komputerowe najszerszej rozwinęły się w USA [6], gdzie nadal funkcjonuje i modyfikowana jest najstarsza akademicka sieć ARPA. O ile początkowe prace związane były z poszerzeniem konfiguracji o nowe komputery obliczeniowe, o tyle w ostatnich kilku latach sieć rozrasta się przez dołączanie sieci lokalnych, które wyposażone w odpowiednie usługi stanowią narzędzie do prac badawczych i dydaktycznych poszczególnych jednostek uczelni.

Tradycja sieci LAN w Europie jest skromniejsza, lecz i tu zauważa się duży postęp w ostatnich latach [6]. W Europie Zachodniej każdy kraj posiada akademicką sieć komputerową, której elementami są sieci lokalne np. UNA (Austria), FUNET (Finlandia), HEANET (Irlandia), SUNET (Szwecja), CHUNET (Szwajcaria). Sieci te połączone są we wspólną sieć EARN (European Academic and Research Network), która jest obecnie jedyną badawczą siecią ogólnoeuropejską.

W krajach socjalistycznych na uwagę zasługują sieci na Węgrzech (LOCHNESS [1]), NRD (ROLANEX, DELTA), ZSRR (AKADEMSIET [7]).

W Polsce pierwszą eksperymentalną siecią była MSK [2], następnie podjęto prace nad siecią SKJS2 [4],[5] oraz KASK.

Prace nad siecią MSK rozpoczęto w 1979 roku w ramach problemu re-sortowego MNSzWiT pt. "Komputeryzacja szkół wyższych", koordynowanego przez Politechnikę Wrocławską. Sieć MSK jest siecią heterogeniczną, w której rolę hostów pełnią komputery ODRA 1305 i R-32. Została ona opisana w licznych publikacjach (patrz literatura w pracy [4]) i dlatego nie będzie tu dokładniej omawiana.

Sieć SKJS2, wersja 1 jest siecią homogeniczną, łączącą komputery R-32. Jej cechą charakterystyczną jest połączenie w procesorze EC8371 (PTD) funkcji procesora czołowego i węzła podsieci transmisji danych.

3. SIEĆ WAN I LAN KASK

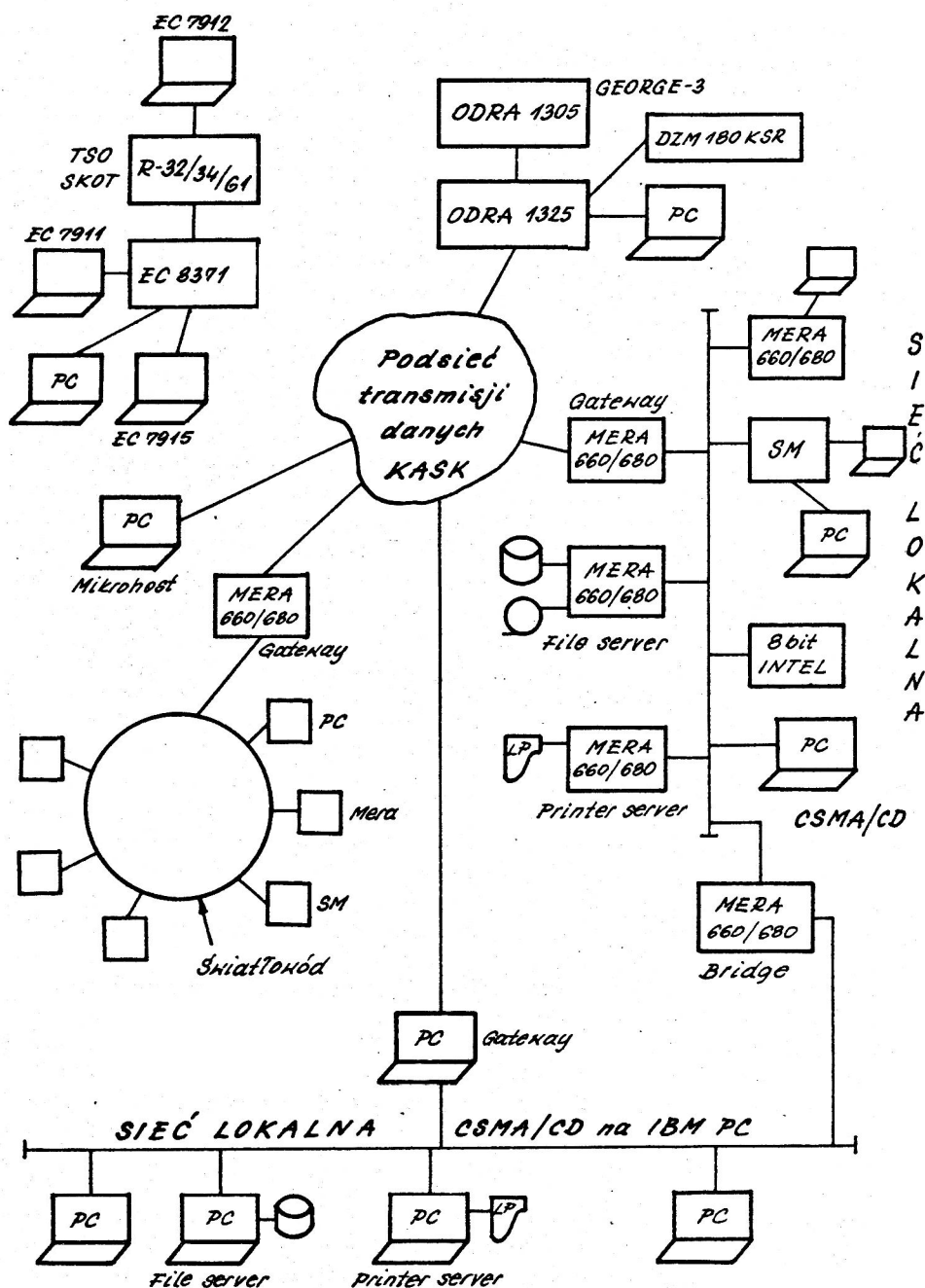
Budowę sieci KASK rozpoczęto w 1986 roku w ramach CPER nr 8.13 pt: "Budowa Krajowej Akademickiej Sieci Komputerowej. Rozwój metod i środków informatycznych w procesach nauczania i badaniach naukowych", którego głównym wykonawcą jest Politechnika Wrocławska. W realizacji CPER nr 8.13 bierze udział wiele ośrodków akademickich w kraju.

Bazą wyjściową do budowy sieci WAN KASK jest sieć MSK. Powstałe już rozwiązania i ciągle powstające nowe są stopniowo wdrażane w poszczególnych ośrodkach akademickich, tworząc tzw. regionalne sieci akademickie. Połączenie sieci regionalnych stworzy sieć WAN KASK.

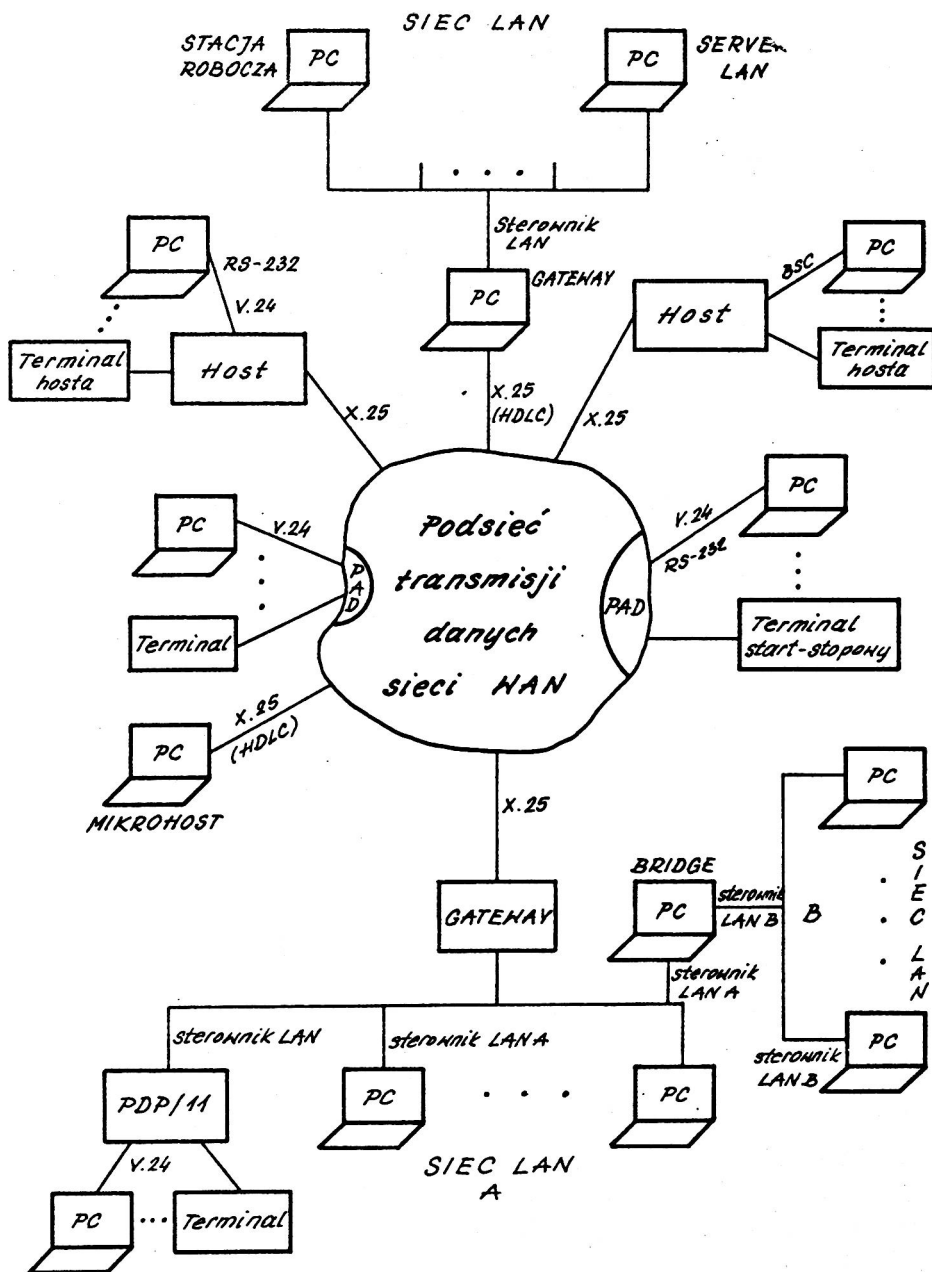
W planach realizacyjnych programu KASK przewiduje się, że oprócz komputerów ODRA 1305 i R-32, do sieci KASK włączone zostaną komputery R-34, R-61, mikrokomputery IBM/PC i sieci LAN KASK. W komputerach serii Riad przystosowane mają być do pracy sieciowej systemy TSO i SKOT oraz wiele baz danych opartych na systemie ISIS. Główne komponenty sieci KASK przedstawiono na rys. 1.

Znaczącym przedsięwzięciem w programie KASK jest budowa lokalnych sieci komputerowych. Zbudowane mają być sieci LAN o strukturze szynowej i pierścieniowej, do których jako stacje będą mogły być włączone mini-komputery SM, mikrokomputery IBM PC, MERA 660 lub 680 oraz 8-bitowe mikrokomputery z rodziny INTEL. Mają powstać sieci lokalne zawierające komputery różnego typu oraz sieci zawierające tylko mikrokomputery IBM PC.

Zgodność standardów protokołów górnych warstw (od warstwy 3 wg modelu OSI/ISO) w sieciach WAN i LAN KASK umożliwi łączenie tych sieci za pomocą bram (IWU). Zapewniony ma być dostęp z każdej stacji LAN do kom-



Rys. 1. Główne komponenty sieci KASK
Fig. 1. Main components of KASK network



Rys. 2. Mikrokomputery klasy IBM/PC w sieciach komputerowych

Fig. 2. Microcomputers IBM/PC in computer network

puterów obliczeniowych i ich terminali sieci WAN. Możliwy ma być również transfer zbiorów między stacjami LAN i komputerami WAN, inicjowany ze stacji LAN w obu kierunkach.

4. MIKROKOMPUTERY TYPU IBM/PC W SIECIACH

Podłączenie mikrokomputerów do sieci LAN lub WAN stwarza wiele problemów natury technicznej, programowej i użytkowej. Z punktu widzenia użytkownika mikrokomputer może spełniać różne funkcje, poczynając od prostego terminala, a kończąc na samodzielnym mikrohoście (DTE sieciowe) w sieci. W zależności od funkcji jakie ma on spełniać w sieci, może być do niej włączony na wiele sposobów. W punkcie tym omówione zostaną tylko niektóre najbardziej typowe zastosowania (rys. 2).

4.1. Mikrokomputer IBM/PC jako terminal innego komputera

W podstawowej konfiguracji mikrokomputer PC zawiera port do transmisji szeregowej według standardu RS-232C (V.24). Pozwala to na podłączenie go do innego komputera jako jego terminala. Komputerami obsługującymi terminale, przez port RS-232C, są np. ODRA 1305, minikomputery SM i MERA 660 lub 680. Tak podłączony mikrokomputer PC musi być jeszcze wyposażony w program emulujący dany typ terminala. Jeśli z kolei dany komputer jest włączony do sieci LAN lub WAN, to również i z PC dostępne są wszystkie usługi sieciowe, tak jak z każdego terminala danego komputera.

Często oprogramowanie IBM/PC, oprócz emulatora terminala, realizuje dodatkowe funkcje, takie jak przesyłanie zbiorów między IBM/PC i komputerem lub kompilację skrośną.

Komputery typu IBM 360 lub 370 (również R-32, R-34, R-61) współpracują ze swymi terminalami według protokołu BSC transmisji synchronicznej. Stąd przyłączenie IBM/PC wymaga dodatkowego wyposażenia mikrokomputera w kartę transmisji synchronicznej BSC i oczywiście w odpowiedni program emulujący terminal danego typu (np. IBM 3275).

Mikrokomputer IBM/PC może również pełnić rolę terminala startstopowego, włączonego do podsieci transmisji danych za pośrednictwem modułu PAD (Packet Assembly Disassembly). Moduł PAD jest na ogół elementem podsieci transmisji danych i umożliwia włączenie PC za pomocą standardowego portu RS-232C. W sieci KASK PAD nie jest on realizowany.

4.2. Mikrokomputer IBM/PC jako MIKROHOST WAN

W mikrohoście realizowane są określone protokoły komunikacyjne. W terminologii ISO/OSI będą one dotyczyć warstw do transportowej włącz-

nie oraz wybranych elementów warstw wyższych, np. określonych aplikacji. W tym przypadku IBM/PC staje się w mniej lub więcej ograniczonym zakresie (zależy od konkretnej realizacji) hostem sieciowym.

Z technicznego punktu widzenia podłączenie mikrokomputera jako mikrohosta wymaga zainstalowania w mikrokomputerze dodatkowej karty realizującej transmisję synchroniczną, najczęściej według protokołu X.25 HDLC. Karta ta może również zawierać własny procesor i program w swojej pamięci ROM/RAM, realizujący protokoły do transportowego włączenia. Rozwiązanie to odciąża główny procesor PC od realizacji funkcji transmisji danych i tym samym udostępnia większą moc dla funkcji użytkowych.

4.3. Mikrokomputer IBM/PC w sieci lokalnej

Mikrokomputery w sieci lokalnej spełniają na ogół rolę stacji roboczych lub specjalizowanych stacji, takich jak obsługi zbiorów (file server) lub obsługi drukarki (printer server). Istnieją sieci zbudowane tylko z IBM/PC, w których wszystkie mikrokomputery pracują pod systemem PC(MS) DOS, np. PC-Network, D-Link, jak również sieci, w których stacje robocze pracują pod PC DOS, a stacja obsługi zbiorów zrealizowana jest na IBM PC/AT pod innym systemem operacyjnym, np. sieć NETWARE firmy NOVELL.

Przyłączenie IBM/PC do sieci LAN wymaga wyposażenia go w sterownik tej sieci. Na ogół są to specjalizowane karty realizujące protokoły komunikacyjne do warstwy transportowej włącznie.

4.4. Mikrokomputer IBM/PC jako brama (gateway) łącząca sieci LAN i WAN

W roli bramy IBM/PC może być elementem sprzęgającym sieć LAN z siecią WAN. Jego zadaniem jest dopasowanie protokołów obu sieci. Technicznie podłączenie bramy wymaga zastosowania dwóch dodatkowych elementów: sterownika LAN i karty realizującej transmisję synchroniczną, zgodną z protokołem sieci WAN. Spotykane jest również połączenie w jednym mikrokomputerze kilku funkcji, np. bramy stacji roboczej.

W artykule dokonano przeglądu funkcji, jakie spełniać mogą mikrokomputery w sieciach, a zwłaszcza w sieci KASK. Przegląd ten byłby niepełny, gdyby nie wspomnieć o wykorzystaniu mikrokomputera do testowania i weryfikacji oprogramowania sieciowego. Szczegółowe omówienie tak przedstawionych zagadnień jest tematem dalszych artykułów.

LITERATURA

- [1] ARATO A., SERKADI-NAGY, SULYAN J., TELBISZ F., A software Architecture of a local Area Network, Computer Network Usage: recent experiences, North Holland 1986.
- [2] BATTEK J. i in., Podsumowanie prac nad Międzyuczelnianą Siecią Komputerową MSK, CO PWr., Raport SPR 16/86, Wrocław 1986.
- [3] BATTEK J. i in., Program budowy Międzyuczelnianej Sieci Komputerowej MSK w latach 1980-1985, część I i II, Wyd. PWr., Wrocław 1980.
- [4] HUZAR Z., KALIŚ A., KOCIATKIEWICZ P., Projekty i realizacje sieci komputerowych. W: Sieci komputerowe. Teoria, technika, zastosowania. Pr. nauk. C. O. PWr., Wrocław 1987, s. 82-99.
- [5] KOCIATKIEWICZ P. i in., Założenia techniczno-ekonomiczne: Sieć komputerowa SKJS2 wersja 1, IKSAiP Nr 033-IPBK-82, Wrocław 1982.
- [6] SADLAK J., The future of academic networks in Europe, CRE-INFORMATION Nr 76/1986, s. 163-167.
- [7] YAKUBITIS E. A., Architecture of Computer Networks for Large offices, Computer Network Usage: recent experiences, North Holland 1986.

FUNCTIONS OF MICROCOMPUTERS IN COMPUTER COMMUNICATION NETWORK KASK

In the paper a survey of the application of microcomputers in computer network WAN and LAN KASK is presented. Especially their function and the way of connection to the system are shown. The microcomputer IBM/PC is described as a terminal, as a microhost WAN, and as a working station of LAN.

ФУНКЦИИ МИКРОКОМПЬЮТЕРОВ В СЕТИ KASK

Представлен обзор применений микрокомпьютеров в вычислительных сетях, особенно в сети WAN и LAN KASK. Перечислены функции, которые они могут выполнять и требования относящиеся к их присоединению к сети. Описываются применения микрокомпьютера IBM PC в качестве терминала, рабочей микро IBM WAN, станции в сети LAN и IWU (gateway) соединяющего сети WAN и LAN.

Praca wpłynęła do Redakcji w marcu 1988 r.
w ostatecznej formie w lipcu 1989 r.

*Sieć komputerowa,
mikrokomputer, KASK,
mikrohost*

Krzysztof JANCZEWSKI*

Andrzej STANISZ*

MIKROKOMPUTERY KLASY IBM PC/XT/AT JAKO MIKROHOSTY W SIECI KASK

Artykuł prezentuje ogólną koncepcję i propozycję realizacji włączenia mikrokomputerów klasy IBM PC/XT/AT do sieci KASK jako mikrohostów, tj. samodzielnych urządzeń końcowych (DTE) o funkcjach analogicznych do funkcji komputerów głównych (hostów). Omówiono wymagania dotyczące sprzętu i programu, funkcje mikrohosta, a następnie proponowaną strukturę oprogramowania.

1. WPROWADZENIE

Pilotowa sieć komputerowa MSK, zbudowana w latach 1980-1985, doprowadziła do eksperymentalnego połączenia komputerów, zainstalowanych w wybranych ośrodkach uczelnianych, w sieć rozległą (typu WAN) o stosunkowo prostych usługach. Podstawowym założeniem przyjętym przez jej projektantów było połączenie dużych (jak na warunki krajowe), wielodostępnych systemów komputerowych ODRA 1300 i R-32, jakimi dysponowały wyższe uczelnie.

Szybki postęp techniki komputerowej, a zwłaszcza rozpowszechnienie minikomputerów, a następnie mikrokomputerów, spowodował gwałtowny wzrost zainteresowania problemami komunikacji komputerów różnych typów i mocy. Ze względu na dużą i stale rosnącą liczbę mikrokomputerów zainstalowanych na wyższych uczelniach szczególnego znaczenia nabrały działania w kierunku zapewnienia połączenia mikrokomputerów z "dużymi" komputera-

* Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

mi, a także łączenia mikrokomputerów w sieci lokalne (typu LAN) [1]-[4].

Połączenie mikrokomputera z dużym komputerem najczęściej realizuje się przez programową emulację terminala odpowiedniego typu [1],[2]. Na rynku oprogramowania znane są i dostępne różne produkty programowe do obsługi tego typu połączeń (m. in. CrossTalk XVI Kermit). Oprócz emulacji typowych terminali umożliwiają one na ogół także wzajemną komunikację komputerów w celu przesyłania zbiorów (zgodnie z jednym z ustalonych protokołów, zapewniających poprawną transmisję w obecności zakłóceń). Rozwiązania te zorientowane są na połączenia liniami telefonicznymi, stałymi i komutowanymi, przez modemy.

W nowszych rozwiązaniach jest możliwe wykorzystanie usług publicznych sieci transmisji danych (z komutacją łącz lub pakietów). Mikrokomputer spełnia wtedy rolę urządzenia końcowego (DTE - ang. Data Terminal Equipment), zdolnego do komunikacji z dowolnym innym tego typu urządzeniem. Sposób tej komunikacji określony jest przez odpowiedni protokół komunikacyjny. Protokół ten może być ustalony przez producenta i zawarty w kompletnym produkcie programowym, albo - w produktach otwartych - może być ustalony przez użytkownika przez dopisanie odpowiedniego programu.

W sieci KASK, wykorzystującej wyniki i doświadczenia z budowy sieci MSK, a także uwzględniającej tendencje rozwojowe, przewidziano opracowanie i wypróbowanie produktu programowego tego typu dla rozpowszechnionych mikrokomputerów klasy IBM PC/XT/AT. Mikrokomputery te, po dość prostym rozszerzeniu konfiguracji w stosunku do konfiguracji standardowej, mogą być podłączone do sieci KASK jako tzw. terminale pakietowe (tzn. urządzenia DTE, współpracujące z siecią wg Zalecenia CCITT X.25) i spełniać (w zakresie ograniczonym do jednego użytkownika) funkcje analogiczne do podstawowych funkcji komputerów głównych sieci (hostów). Rozwiązanie to nazwano mikrohostem [5],[6].

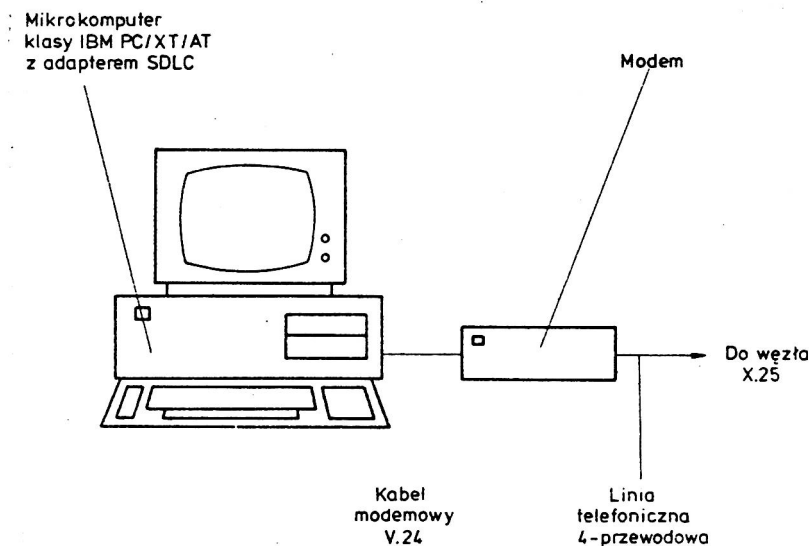
2. WYMAGANIA DOTYCZĄCE SPRZĘTU I PROGRAMU

Mikrokomputer klasy IBM PC/XT/AT, wykorzystywany jako mikrohost, musi być wyposażony w urządzenie zapewniające szeregową, synchroniczną transmisję danych, bitowo-zorientowaną (tzn. według protokołu HDLC), w trybie duplexowym, z szybkością 2400-9600 bit/s. Standardowe konfiguracje mikrokomputerów nie zawierają takiego urządzenia, ale mogą być w prosty sposób uzupełnione przez zamontowanie odpowiedniej karty rozszerzającej i modemu.

Na światowym rynku komputerowym oferuje się wiele kart, nadających się do takiego zastosowania, od prostych adapterów, realizujących jedy-

nie transmisję danych, do złożonych sterowników komunikacyjnych, z własnym mikroprocesorem, przejmujących na siebie realizację protokołów komunikacyjnych i odciażających procesor główny.

Dla mikrohosta przyjęto rozwiązanie najprostsze: zainstalowanie karty SDLC, będącej odpowiednikiem IBM Synchronous Data Link Communication Adapter. Za takim rozwiązaniem przemawia przede wszystkim stosunkowo niski koszt karty i względna łatwość jej uzyskania.

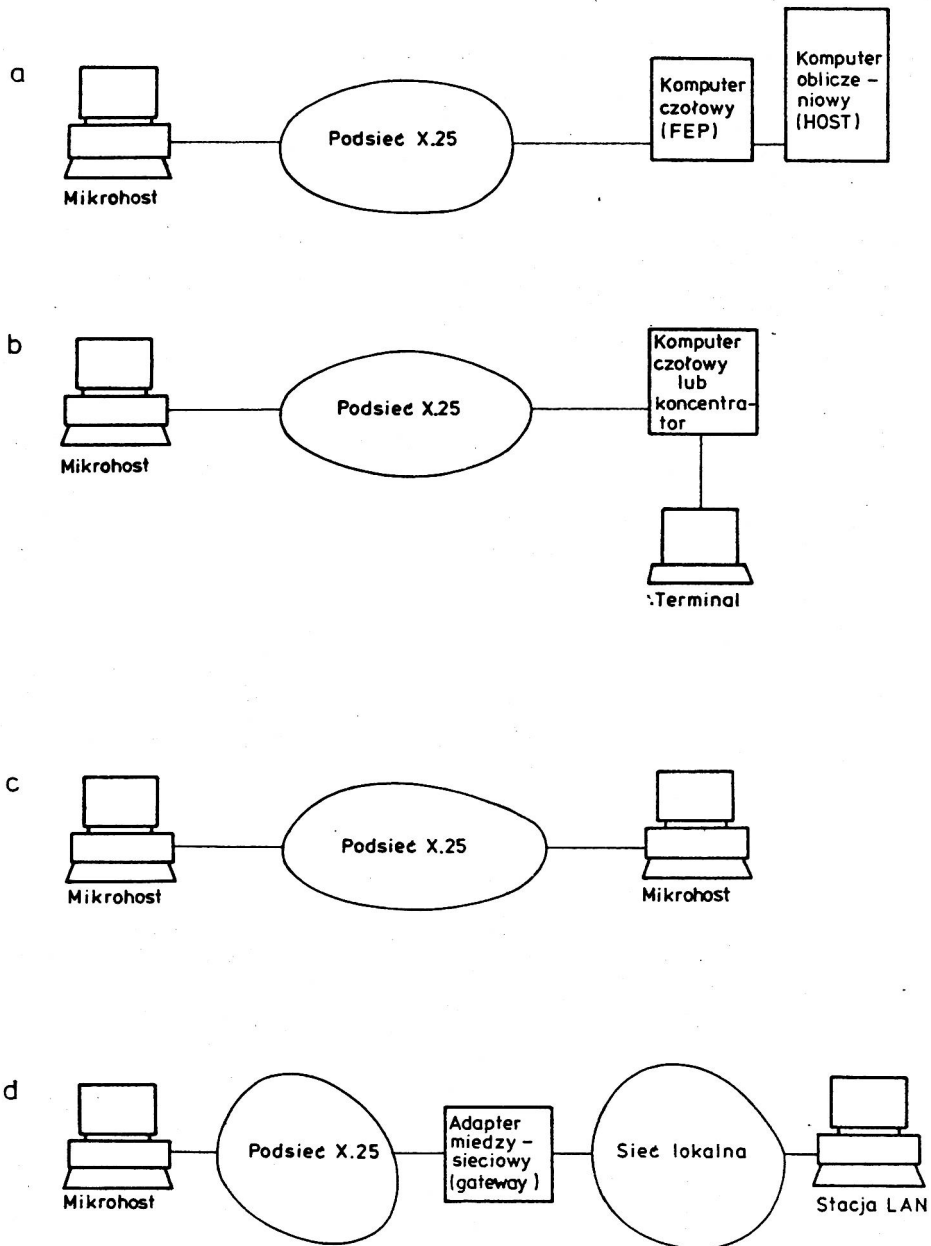


Rys. 1. Zestaw urządzeń mikrohosta

Fig. 1. Hardware configuration of microhost

Tak rozszerzony mikrokomputer, dołączony przez modem i linię telefoniczną do węzła sieci KASK (rys. 1), może być - przy odpowiednim oprogramowaniu - używany bądź jako normalny mikrokomputer osobisty, bądź jako mikrohost. Jego użytkownik może łączyć się z dowolnym komputerem lub terminalem, dostępnym w sieci rozległej KASK/WAN, a także przez podsieć komunikacyjną - z użytkownikami podłączonych do niej sieci lokalnych LAN/KASK (rys. 2).

Oprogramowanie mikrohosta powinno być zrealizowane tak, by praca w sieci nie wiązała się z dodatkowymi utrudnieniami dla użytkownika, zwłaszcza ze względu na powszechne stosowanie systemu operacyjnego PC-DOS (MS-DOS), traktowanego jako standard, wymaga się, by oprogramowanie mikrohosta mogło być eksploatowane pod kontrolą tego właśnie systemu.



Rys. 2. Możliwe połączenia sieciowe mikrohosta: a - mikrohost-host, b - mikrohost-terminal, c - mikrohost-mikrohost, d - mikrohost-stacja LAN

Fig. 2. Possible network connections of microhost: a - microhost-host, b - mikrohost-terminal, c - mikrohost-mikrohost, d - mikrohost LAN station

3. FUNKCJE MIKROHOSTA

Mikrokomputer klasy IBM/PC/XT/AT jest komputerem zorientowanym na jednego użytkownika, z jednoprogramowym na ogół systemem operacyjnym, pozbawionym ochronnych mechanizmów sprzętowych wymaganych podczas pracy wielodostępnej. Dlatego też w doborze funkcji mikrohosta preferowane powinny być przede wszystkim usługi świadczone jego bezpośredniemu użytkownikowi, a w mniejszym stopniu uwzględnione usługi świadczone przez mikrohosta na rzecz jego zdalnych użytkowników sieciowych.

Rozwój sieci KASK w kierunku budowy wielu sieci lokalnych (sieci LAN/KASK) i ich połączenia z siecią rozległą (sieć WAN/KASK) wymaga, by mikrohost, będący elementem sieci rozległej, umożliwiał swemu użytkownikowi także łączenie się z wybranymi stacjami roboczymi sieci lokalnych.

Uwzględniając takie założenie, przyjęto następujący zestaw podstawowych funkcji mikrohosta:

- połączenie terminal-host, w którym mikrohost emuluje zdalny terminal interakcyjny dowolnego hosta (rys. 2a)
- połączenie terminal-terminal, tzn. połączenie mikrohosta z dowolnym terminalem sieci, w tym także z innym mikrohostem, umożliwiające wzajemną wymianę komunikatów (rys. 2b,c)
- przesyłanie zbiorów między mikrohostem a innymi hostami sieci KASK, w tym również innym mikrohostem (rys. 2a,c)
- połączenie terminal-terminal oraz przesyłanie zbiorów między mikrohostem a dowolną stacją sieci lokalnej LAN/KASK (rys. 2d).

W odróżnieniu do typowego hosta sieci rozległej nie przewiduje się możliwości zdalnej pracy terminalowej w mikrohoście pod jego systemem operacyjnym.

3.1. Połączenie ze zdalnym hostem

W połączeniu tym mikrohost występuje w roli inteligentnego zdalnego terminala komputera i tej jego aplikacji, z którą nastąpiło połączenie. Użytkownik mikrohosta uzyskuje w ten sposób zdalny dostęp do zasobów programowych dowolnego komputera obliczeniowego (hosta) w sieci (ale nie innego mikrohosta). W danej chwili może być realizowane tylko jedno połączenie tego typu. Na obecnym etapie rozwoju sieci KASK dostępne są połączenia z:

- systemem GEORGE 3/MOP w komputerze ODRA 1305
- systemem TSO w komputerze R-32.

Przewiduje się możliwość łączenia z innymi komputerami, które mają być dołączane do sieci KASK (m. in. R-34, R-61, także z aplikacjami opartymi na systemie SKOT).

Połączenie terminalowe z hostem, zgodnie z przyjętą w sieci KASK architekturą, realizowane będą za pomocą usług wirtualnego terminala [7],[9]. Stacje wirtualnego terminala, zaimplementowane we wszystkich hostach sieci MSK, traktują terminale jako proste terminale wierszowe (jednowymiarowe). Dla wielu istniejących aplikacji takie podejście jest wystarczające. Jednakże większość zastosowań SKOT-a wymaga terminala ekranowego (dwuwymiarowego), np. typu MERA 7950 (IBM 3275).

Dlatego w sieci KASK zdecydowano się na pewną modernizację stacji wirtualnego terminala, dopuszczając tzw. przezroczysty tryb pracy. W trybie tym terminal sterowany jest przez aplikację i wszystkie dane bez żadnych przekształceń przesyłane są z aplikacji do terminala i na odwrót. Oczywiście taki tryb pracy wymaga, aby terminal był takiego typu, jakiego oczekuje aplikacja.

Wychodząc naprzeciw tym potrzebom, przewiduje się wbudowanie do mikrohosta emulatora programowego terminala IBM 3275. Użytkownik będzie miał do dyspozycji dwa typy terminali: terminal wierszowy i terminal ekranowy IBM 3275. Wybór typu terminala będzie dokonywał się automatycznie, w zależności od żądań aplikacji, z którą nastąpiło połączenie. O wybranym typie terminala użytkownik będzie poinformowany.

Jednocześnie z pracą terminalową ze zdalną aplikacją, w mikrohoście dostępne będą następujące udogodnienia:

- a) możliwość wydawania komend dotyczących transferu zbiorów,
- b) możliwość obsługi połączeń typu terminal-terminal i żądań transferu zbioru zgłaszanych z sieci,
- c) realizacja komend typu: dir, type, erase, chdir, copy, space,
- d) symulowanie wysyłania komunikatów ze zbioru dyskowego zamiast z klawiatury,
- e) przechwytywanie komunikatów lub ekranów i bezpośredni zapis do zbioru dyskowego,
- f) przechwytywanie całych ekranów lub wybranych ich części do bufora w pamięci RAM,
- g) przechwytywanie komunikatów dialogu terminal-aplikacja do bufora z odpowiednio ustawioną selekcją:
 - tylko komunikaty przychodzące
 - tylko komunikaty wychodzące
 - pełny dialog
 - filtracja przechwytywanych komunikatów ze znaków sterujących
- h) edycja bufora (przeglądanie i usuwanie fragmentów),
- i) zapis bufora do zbioru dyskowego,

j) możliwość redefiniowania wybranych klawiszy, którym można będzie przypisać dowolny tekst,

k) udogodnienie typu "help" dotyczące emulatora i komend mikrohosta.

Dokładniejszy opis realizacji połączenia z hostem oraz opis realizacji udogodnień przedstawiono w pracy [5].

3.2. Połączenie ze zdalnym terminalem

W połączeniu mikrohost-inny terminal (może nim być również inny mikrohost) mikrohost reprezentuje zawsze terminal wierszowy. Połączenie to umożliwia wzajemną wymianę komunikatów między terminalami. Każdy wiersz tekstu wprowadzony na jednym terminalu wyprowadzany jest na drugi terminal. Wymiana tekstów prowadzona jest w trybie duplexu, tzn. oba terminale mają równocześnie prawo nadawania.

Połączenie terminal-terminal może być inicjowane przez użytkownika mikrohosta lub też zgłoszone z sieci, tzn. inicjatorem połączenia z mikrohostem może być użytkownik zdalnego terminala. Wirtualny terminal mikrohosta będzie miał tylko dwa punkty obsługi połączeń terminalowych: jeden - do obsługi połączeń własnych (inicjowanych przez użytkownika mikrohosta) i drugi - do obsługi połączeń zgłaszanych z sieci. Jeśli ten drugi punkt będzie zajęty obsługą połączenia, to w tym czasie nie będzie przyjmował innych zgłoszeń. Tak więc w danej chwili mogą istnieć tylko dwa połączenia terminalowe: jedno własne (terminal-host albo terminal-terminal) i drugie terminal-terminal zgłoszone z sieci.

Użytkownik mikrohosta może zablokować punkt obsługi połączeń zgłaszanych z sieci, tzn. zabronić przyjmowania jakichkolwiek zgłoszeń. Jeśli ten punkt nie będzie zablokowany i będzie wolny, to zgłoszenie połączenia z sieci zostanie wstępnie przyjęte, a użytkownik mikrohosta zostanie o tym powiadomiony sygnałem dźwiękowym. Użytkownik może to zgłoszenie przyjąć lub odrzucić. Przyjmując zgłoszenie użytkownik może zawiesić swoje własne połączenie (jeżeli takie istniało) oraz przełączyć ekran i klawiaturę do obsługi połączenia zgłaszanego.

W połączeniu terminal-terminal dostępne będą wszystkie udogodnienia opisane w p. 3.1.

3.3. Transfer zbiorów

W sieci KASK implementowany jest system transferu zbiorów, który umożliwia przesyłanie zbiorów między pamięciami zbiorów różnych systemów komputerowych. System ten oparty jest na wspólnym dla wszystkich użytkowników i komputerów modelu pamięci zbiorów, nazywanym wirtualną pamięcią zbiorów [8],[9]. Pozwala to na zamaskowanie różnic między rzeczy-

wistymi pamięciami zbiorów przez funkcje odwzorowania lokalnej, rzeczywistej pamięci zbiorów na opis zgodny z jednolitym, rozumianym przez wszystkich modelem i umożliwia współpracę różnych systemów w standardowy sposób.

Przesyłanie zbiorów wymaga współpracy dwóch stacji transferu zbiorów, zlokalizowanych w różnych komputerach i komunikujących się ze sobą za pomocą specjalnego protokołu [8],[9]. Użytkownik kieruje swoje polecenia do stacji znajdującej się w tym komputerze, w którym pracuje, używając języka komend tej stacji. Stacja ta staje się wykonawcą zlecenia. Steruje ona jego realizacją i przekazuje użytkownikowi wszystkie informacje, związane z wykonaniem zlecenia.

Z punktu widzenia użytkownika, minimalny zakres usług, który musi być realizowany przez wszystkie stacje transferu zbiorów, sprowadza się do następujących zasad:

1. Można dokonywać transferu prostych sekwencyjnych zbiorów tekstowych, tj. zbiorów składających się z ciągu drukowalnych znaków, podzielonych na rekordy.

2. Stacja, przyjmująca zlecenie użytkownika, może realizować transfer zbioru w dwóch kierunkach: z własnego komputera do odległego lub z odległego do własnego.

3. Stacja przyjmująca zbiór, może pracować w dwóch trybach:

a) zbiór o podanej nazwie ma zostać utworzony jako nowy zbiór: jeśli zbiór o tej nazwie już istnieje, to transfer zostaje odrzucony;

b) jeśli zbiór o podanej nazwie istnieje, to przyjmowany zbiór ma być wpisany jako jego nowa zawartość; jeśli zbiór nie istnieje, to ma być utworzony nowy.

4. Stacja wysyłająca zbiór powinna go tylko przeczytać (skopiować).

5. Obie współpracujące stacje przesyłają między sobą zbiory w kodzie IA5 (równoważnym kodowi ASCII). Każda ze stacji (jeżeli to konieczne) dokonuje konwersji kodu lokalnego na IA5 i odwrotnie.

6. Identyfikację zbioru i autoryzację dostępu do nich określają następujące atrybuty:

- nazwa zbioru
- nazwa konta rozliczeniowego za transfer
- hasło konta rozliczeniowego
- nazwa użytkownika zbioru
- hasło użytkownika.

7. Transfer zbioru odbywa się w jednym seansie. Nie ma mechanizmów umożliwiających wznowienie przerwano (np. wskutek awarii) transferu.

Podany zakres usług realizowany będzie również przez stację transferu zbiorów mikrohosta. W stosunku do stacji, zaimplementowanych na komputerach ODRA 1305 i R-32, które mogą realizować jednocześnie kilka

transferów, a także oferują pewne dodatkowe usługi, stacja transferu zbiorów mikrohosta będzie rozwiązaniem prostszym [5]. Ze względu na ograniczone możliwości mikrokomputera, będzie ona mogła w danej chwili realizować tylko jeden transfer - zlecony przez użytkownika lub zgłoszony przez inną stację.

Transfer zbioru inicjowany będzie na ogół przez użytkownika mikrohosta. Aby umożliwić przesyłanie zbiorów między dwoma mikrohostami, przewidziano także przyjmowanie żądań transferu sieci. Użytkownik będzie miał możliwość blokady przyjmowania zgłoszeń transferów z sieci (tj. z innych stacji). Przy wolnej i niezablokowanej stacji zgłoszenie transferu z sieci będzie sygnalizowane sygnałem dźwiękowym. Użytkownik będzie mógł wyrazić zgodę na transfer lub go odrzucić.

Obsługa zlecenia przez stację rozpoczyna się od próby nawiązania dialogu ze stacją zdalną (faza negocjacji). Jeżeli transfer zdefiniowany przez użytkownika nie jest wykonalny albo został odrzucony przez partnera, to zlecenie transferu zostanie odrzucone, a informacja o przyczynie odrzucenia - przekazana użytkownikowi. Pomyślne zakończenie fazy negocjacji rozpoczyna właściwy transfer danych.

Jeśli w fazie negocjacji zostanie nawiązany dialog z partnerem (np. z powodu awarii lub zajętości partnera), to realizacja danego zlecenia transferu zostanie odłożona na pewien czas, po czym nastąpi ponowna próba realizacji transferu. Użytkownik będzie miał możliwość sprawdzenia stanu realizacji swego zlecenia i ewentualnie skasowania go.

Przyjmuje się, że transfer zbioru będzie realizowany współbieżnie z innymi funkcjami mikrohosta - jako zadanie drugoplanowe.

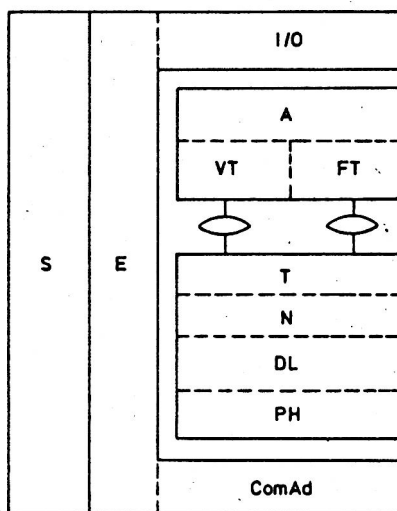
4. STRUKTURA PROGRAMOWA

Oprogramowanie mikrohosta funkcjonować powinno w typowym środowisku programowym mikrokomputerów klasy IBM PC/XT/AT, za które powszechnie uważa się system operacyjny PC DOS. Dlatego przyjęto, że będzie je stanowić pojedynczy program, przeznaczony dla tego systemu. Minimalna konfiguracja sprzętowa wymagana do jego pracy jest następująca:

- mikrokomputer klasy /PC/XT/AT z pamięcią min. 512 K bajtów
- monitor dowolnego typu
- 2 jednostki dyskietek lub 1 jednostka dysku twardego i 1 jednostka dyskietek
- adapter komunikacyjny SDLC.

4.1. Struktura funkcjonalna programu

Program mikrohosta zaprojektowano zgodnie z wymaganiami architektury ISO/OSI, z uwzględnieniem aktualnego stanu standardów oraz ograniczeń przyjętych w sieci MSK/KASK. Jednocześnie starano się nadać mu charakter rozwiązania otwartego na przyszłe modyfikacje i rozszerzenia, a także rozwiązania przenośnego, nadającego się do wykorzystania w innych produktach programowych o przeznaczeniu sieciowym, również w innych komputerach i innych systemach operacyjnych.



Rys. 3. Struktura oprogramowania mikrohosta

Fig. 3. Software structure of microhost

Strukturę funkcjonalną programu mikrohosta pokazano na rys. 3. Jako podstawę dekompozycji programu na moduły przyjęto podział na warstwy według modelu OSI/ISO. W strukturze wyodrębniono moduły realizujące 4 najniższe warstwy (Ph - fizyczną, DL - liniową, N - sieciową i T - transportową), a także moduły realizujące usługi wirtualnego terminala (VT) oraz transferu zbiorów (FT), tworzące razem z modułem jądra aplikacji (A) zintegrowaną aplikację.

Wymienione moduły współpracują ze sobą, a także z systemem operacyjnym (S), korzystają z usług modułu środowiska wewnętrznego (E).

Przyjęta zasada modularyzacji w naturalny sposób narzuca styki międzywarstwowe: sąsiednie warstwy komunikują się przez tzw. punkty dostępu do usług i przekazywanie prymitywów usługowych, zdefiniowanych w do-

kumentach ISO i CCITT (m. in. [10]). Ich implementacja w oprogramowaniu mikrohosta polega na zdefiniowaniu odpowiednich struktur danych (odpowiadających punktom dostępu) i podprogramów (odpowiadających przekazywanym prymitywom). Zasadę tę zastosowano także do styków wewnątrz zintegrowanej aplikacji (tzn. do usług wirtualnego terminala i transferu zbiorów), dla których brak odpowiednich dokumentów standaryzacyjnych.

4.2. Wewnętrzne środowisko programowe

Wewnętrzne środowisko programowe wyodrębniono z kilku powodów [5], [11]:

- a) w celu zapewnienia udogodnień programowych związanych ze specyfiką oprogramowania komunikacyjnego, i potrzebnych we wszystkich modułach programowych (np. zarządzanie pamięcią dynamiczną - gospodarka buforami),
- b) w celu dostarczenia zunifikowanych mechanizmów umożliwiających quasi-współbieżną realizację wielu akcji w różnych warstwach oprogramowania, a także standardowych środków komunikacji i synchronizacji tych akcji,
- c) w celu stworzenia szansy przenośności oprogramowania przez zlokalizowanie bezpośrednich kontaktów z systemem operacyjnym w modułach środowiska.

Wewnętrzne środowisko programowe mikrohosta [5] zakłada podział funkcji programu na fragmenty, realizowane w procedurach. Wprowadzono 4 klasy procedur:

- klasa I - procedury obsługi przerw
- klasa S - procedury-podprogramy (do realizacji zasadniczych funkcji przetwarzania)
- klasa T - procedury opóźnione (do realizacji czynności związanych z upływem limitów czasowych)
- klasa P - procedury-procesy (do realizacji funkcji przetwarzania o charakterze sekwencyjnym).

Wymienione procedury różnią się sposobem uaktywnienia: procedury klasy I uaktywniane są z przerw zewnętrznych (przede wszystkim adaptera komunikacyjnego i klawiatury), pozostałe są wywoływane przez wewnętrzny koordynator, zgodnie z żądaniami zgłaszanymi z innych procedur.

Działanie koordynatora jest następujące: żądania uaktywnienia procedur klasy S i P umieszczane są we wspólnej kolejce zgodnie z zasadą FIFO. Koordynator pobiera z niej kolejne żądanie i wywołuje wskazaną procedurę. Podczas wywołania procedury przekazywany jest do niej pojedynczy parametr (np. adres bufora z komunikatem), związany z żądaniem uaktywnienia. Realizacja procedury przebiega aż do jej zakończenia (dla klasy S) lub zawieszenia (dla klasy P), po czym sterowanie wraca do koordynatora.

Podczas realizacji procedury mogą generować żądania uaktywnienia dalszych procedur, zwłaszcza mogą one zgłaszać żądania uaktywnienia procedur klasy T, a także anulować wcześniejsze żądania tego typu. Procedury klasy T wywoływane są przez koordynator z opóźnieniem wskazanym w żądaniu. Zgłoszenia uaktywnienia procedury klasy T rejestrowane są w oddzielnej kolejce zegarowej. Koordynator na podstawie sygnałów odbieranych z przerwów zegarowych co jednostkę czasu przegląda tę kolejkę, aktualizuje liczniki czasu i wybiera kolejno do realizacji wszystkie procedury, dla których wyczerpał się limit czasowy.

Procedury klasy S, P i T wykonywane są zgodnie z zasadą wzajemnego wykluczania (tzn. nie jest możliwe ich wzajemne przerywanie), dzięki czemu problemy współużytkowania wspólnych zasobów (np. danych) znakomicie się upraszczają.

Procedury klasy P stanowią implementację prostej formy quasi-współbieżności, opartej na idei współprogramów (korutyn). Środowisko dostarcza mechanizmów umożliwiających:

- dynamiczne kreowanie i likwidację procesów
- synchronizację procesów z wykorzystaniem semafora
- komunikację i synchronizację procesów opartą na przekazywaniu komunikatów
- zawieszanie procesów z limitem czasowym.

Opisane środowisko oferuje stosunkowo rozbudowane udogodnienia. Wprowadzono je celowo, tak by zapewnić znaczną swobodę podczas realizacji poszczególnych warstw. Wydaje się, że oprócz przedstawionej tu pełnej wersji, możliwe jest zbudowanie uproszczonej wersji środowiska, w której nie występują procedury klasy P.

Zaproponowane mechanizmy mają określone zalety i wady, ujawniające się w różnych sytuacjach. Obszerniejszą dyskusję na ten temat zawiera artykuł [11].

4.3. Implementacja usług i protokołów

Ze względu na przeznaczenie oprogramowania mikrohosta zakres implementacji usług i protokołów komunikacyjnych musi odpowiadać ustaleniom dla sieci MSK/KASK [9]. Dotyczy to przede wszystkim protokołów działających w warstwie aplikacji, tzn. wirtualnego terminala i transferu zbiorów, które nie są zgodne z aktualnymi opracowaniami ISO.

W zakresie warstw komunikacyjnych (do transportowej włącznie) ustalenia dla MSK są podzbiorem istniejących standardów lub nieznacznie od nich odbiegają. Dlatego przyjęto, że w celu ułatwienia przyszłego rozwoju tej części oprogramowania opracowana zostanie tzw. wersja bazowa, odpowiadająca stanowi standaryzacji według stanu z roku 1984 (CCITT Red Book [10]) i zawierająca jedynie elementy i mechanizmy, określane w tych

dokumentach jako obowiązkowe. W wersji tej zawarte będą - jako opcje - wszystkie rozwiązania specyficzne dla sieci MSK/KASK. Po włączeniu tych opcji podczas generowania oprogramowania uzyska się tzw. wersję KASK, potrzebną w oprogramowaniu mikrohosta.

Wersja bazowa będzie tak skonstruowana, by mogła być łatwo uzupełniana i rozszerzana zgodnie z obowiązującymi standardami międzynarodowymi. Włączenie tych nowych elementów do wersji bazowej jako opcji doprowadzi w przyszłości do przekształcenia jej w tzw. wersję rozszerzoną. Powstanie w ten sposób baza programowa, podatna na stałą modernizację i umożliwiająca generację różnych wariantów oprogramowania sieciowego, przeznaczonych zarówno dla sieci KASK, jak i dla przyszłych zastosowań w publicznej sieci danych.

Opisana konstrukcja oprogramowania i język C, przyjęty jako główny język programowania, powinny spowodować, że powstanie oprogramowanie przenośne, którego zastosowanie nie ograniczy się do mikrohosta.

5. WNIOSKI KOŃCOWE

W artykule przedstawiono koncepcję i niektóre elementy projektu mikrohosta dla sieci KASK według stanu prac pod koniec etapu projektu technicznego (na początku 1988 roku). Gotowy produkt programowy ma być dostępny pod koniec 1989 roku. Zastosowanie tego rozwiązania w sieci KASK będzie raczej ograniczone: liczba mikrokomputerów klasy IBM PC/XT/AT, które będzie można włączyć do sieci KASK, będzie stosunkowo niewielka, zwłaszcza z powodu ograniczonej liczby linii w węzłach, które można by wydzielić do tego celu.

Projekt mikrohosta, oprócz celu czysto użytkowego, ma także i cel badawczy: sprawdzenie możliwości budowy przenośnego oprogramowania sieciowego dla mikrokomputerów osobistych. Przewiduje się przede wszystkim możliwość budowy pokrewnych produktów programowych dla sieci KASK: prostej stacji LAN/KASK i tzw. adaptera międzysieciowego (ang. gateway) LAN/WAN/KASK dla sieci lokalnej budowanej w Centrum Obliczeniowym Politechniki Wrocławskiej [1],[3],[5].

Prawdopodobnie najistotniejszym efektem przedsięwzięcia pod nazwą "mikrohost" będzie stworzenie modularnej bazy przenośnego oprogramowania sieciowego, opartego na filozofii modelu OSI/ISO, z której w łatwy sposób można będzie otrzymać odpowiednie oprogramowanie dla pewnej rodziny produktów sieciowych, instalowanych na różnych komputerach.

LITERATURA

- [1] BIELENINIK E., CHYŁA R., JANCZEWSKI K., ŁANOWSKA B., MARKO K., RYCHLIKOWSKI E., STANISZ A., Włączanie mikrokomputerów kompatybilnych z IBM PC i Elwro 800 do sieci KASK. Wstępne założenia funkcjonalne CO PWr., Raport SPR nr 30/86, Wrocław 1986.
- [2] CHYŁA R., JANCZEWSKI K., RYCHLIKOWSKI E., Mikrokomputer kompatybilny z IBM PC jako inteligentny terminal procesorów czołowych sieci KASK. Założenia funkcjonalne, PGM-15/01/87, CO PWr., Raport SPR nr 3/87, Wrocław 1987.
- [3] BICZ T., BIELENINIK E., KOSMULSKA-BOCHENEK E., HUDYMA E., Mikrokomputer kompatybilny z IBM PC jako gateway LAN-WAN/KASK. Założenia funkcjonalne, PGM-15/03/87. CO PWr., Raport SPR nr 5/87, Wrocław 1987.
- [4] JANCZEWSKI K., ŁANOWSKA B., STANISZ A., Oprogramowanie minikomputera SM-4 jako minihosta sieci MSK. Projekt techniczny CO PWr., Raport SPR nr 24/85, Wrocław 1985.
- [5] JANCZEWSKI K., STANISZ A., Mikrokomputer kompatybilny z IBM PC jako mikrohost w sieci KASK. Założenia funkcjonalne, PGM-15/02/87, CO PWr., Raport SPR nr 4/87, Wrocław 1987.
- [6] Mikrokomputer kompatybilny z IBM PC jako mikrohost w sieci KASK. Projekt techniczny. PGM-15/02/88, (pod red. A. Stanisza), CO PWr., Raport SPR nr 12/88, Wrocław 1988.
- [7] WIETRZYCH J., Usługi i protokoły terminala interakcyjnego w Międzyuczelnianej Sieci Komputerowej. Wersja 1, Inst. Cybern. Techn. PWr., Raport SPR nr 3/83, Wrocław 1983.
- [8] KASZUBA R., Protokół transferu zbiorów dla MSK, CO PWr., Raport SPR nr 16/83, Wrocław 1983.
- [9] Protokoły w międzyuczelnianej sieci komputerowej (MSK), (pod red. E. Bilskiego), Biblioteka Informatyki Szkół Wyższych, Wyd. PWr., Wrocław 1987.
- [10] CCITT Recommendations X.200-X.250, Red Book, Volume VIII, Fascicle VIII.5, Geneva 1985.
- [11] STANISZ A., Elegancja czy efektywność, czyli jak projektować oprogramowanie sieciowe (w niniejszym zeszycie).

IBM PC/XT/AT COMPATIBLE MICROCOMPUTERS
AS MICRO-HOSTS IN THE KASK COMPUTER NETWORK

The paper contains a presentation of a general concept and proposed realization for including IBM PC/XT/AT compatible microcomputers into the KASK computer network as micro-hosts, i.e. as self-contained terminals (DTE) with functions similar to those of main computers (hosts). The hardware and software requirements, the functions of the micro-host, and the proposed software structure are discussed.

МИКРО-ЭВМ СОВМЕСТИМЫЕ С IBM PC/XT/AT КАК "МИКРО-РАБОЧИЕ МАШИНЫ"
В ВЫЧИСЛИТЕЛЬНОЙ СЕТИ KASK

В статье представлены общая концепция и предложение разработки для решения проблемы включения в вычислительную сеть KASK микро-ЭВМ совместимых с IBM PC/XT/AT в качестве "микро-рабочих машин", т.е. самостоятельных оконченных устройств (ООД) с функциями сходными с функциями главных ЭВМ. Описываются аппаратные и программные требования, функции "микро-рабочей машины" и предлагаемая структура программного обеспечения.

Praca wpłynęła do Redakcji w marcu 1988 r.
w ostatecznej formie w lipcu 1989 r.

*Sieć komputerowa, pomiar,
tester, protokół*

Iwona DUBIELEWICZ*, Zbigniew FRYŻLEWICZ*, Krystyna KOLEŚNIK*,
Witold KOMOROWSKI*, Leszek MISIURA*

ZASTOSOWANIE APARATURY POMIAROWO-DIAGNOSTYCZNEJ W SIECI KASK

Opisano urządzenie SPS umożliwiające monitorowanie transmisji na styku DTE-DCE oraz testowanie wybranych urządzeń i linii transmisji danych. SPS jest systemem mikrokomputerowym, dla którego przygotowano specjalizowany system operacyjny. Operator, korzystając ze zbioru prostych komend, może w trybie dialogowym przygotować i przeprowadzić wybrany proces. Wyniki są pokazywane na ekranie lub mogą być wyprowadzane na drukarkę. Wstępna eksploatacja SPS pozwala sformułować wnioski odnośnie do jego przydatności i zaproponować zmiany wprowadzane w nowej wersji.

1. WSTĘP

Cyfrowe urządzenia do analizy strumieni danych są obecnie niezbędnym narzędziem pomiarowym stosowanym do uruchamiania elementów sieci komputerowych i w czasie ich eksploatacji. Urządzenia te umożliwiają diagnostykę elementów sieci dostępnych przez styk cyfrowy DTE-DCE łącza transmisyjnego. Metody takiej diagnostyki można podzielić na trzy grupy:

- testowanie w czasie normalnej pracy (monitorowanie strumienia danych),
- testowanie w warunkach symulacji DTE (wymuszanie strumienia danych),
- testowanie sygnałów styku DTE-DCE.

^x Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

Bierne monitorowanie danych w wybranym punkcie sieci pozwala uzyskać charakterystyki ilościowe, takie jak np. liczba odrzuconych lub powtórzonych bloków, czas odpowiedzi danego urządzenia końcowego, sekwencje znaków występujące w badanym okresie itp. Z kolei obserwacja styku cyfrowego na poziomie sygnałów binarnych daje informacje o stanie jego obwodów w różnych fazach transmisji.

Testowanie czynne polega na generowaniu strumienia danych, odpowiadającego wybranemu protokołowi w celu obserwowania odpowiedzi badanego urządzenia. Tego rodzaju testowanie może być przydatne np. podczas uruchamiania nowych urządzeń, kiedy wytworzenie specyficznych i powtarzalnych sytuacji ułatwia wykrycie ewentualnych błędów.

Skonstruowany na Politechnice Wrocławskiej w latach 1985-1986 system SPS [1], oparty na mikroprocesorze 8085, spełnia wszystkie podstawowe wymagania stawiane tego rodzaju aparaturze. Jego cechy charakterystyczne to:

- połączenie w jednym urządzeniu funkcji monitorowania i testowania
- implementacja transmisji asynchronicznej i wszystkich powszechnie stosowanych protokołów, takich jak: BSC, LAPB, SDLC, X.25
- możliwość analizy strumieni danych na poziomie liniowym i pakietowym w czytelnej postaci (tzw. proceduralnej)
- możliwość łatwego zadawania testów symulacyjnych pisanych w prostym języku komend
- wygodna obsługa urządzenia za pomocą zestawu komend w trybie dialogowym z podpowiedziami systemu
- możliwość powielania uzyskanych na ekranie raportów wprost na drukarkę.

Użytkownik SPS określa wybrany typ pracy za pomocą komend, których pełna definicja wprowadzana jest do SPS w trybie dialogowym. Dla wszystkich typów monitorowania (z wyjątkiem monitorowania styku) warunki te definiują:

- sposób monitorowania dwóch strumieni danych (do/z DCE) łącznie lub rozdzielnie
- rodzaje tzw. pułapek określających warunki zakończenia monitorowania
- szczegółowe parametry transmisji podlegającej monitorowaniu, zależne od typu monitorowania
- dodanie znacznika czasu odbioru rejestrowanych jednostek protokołu.

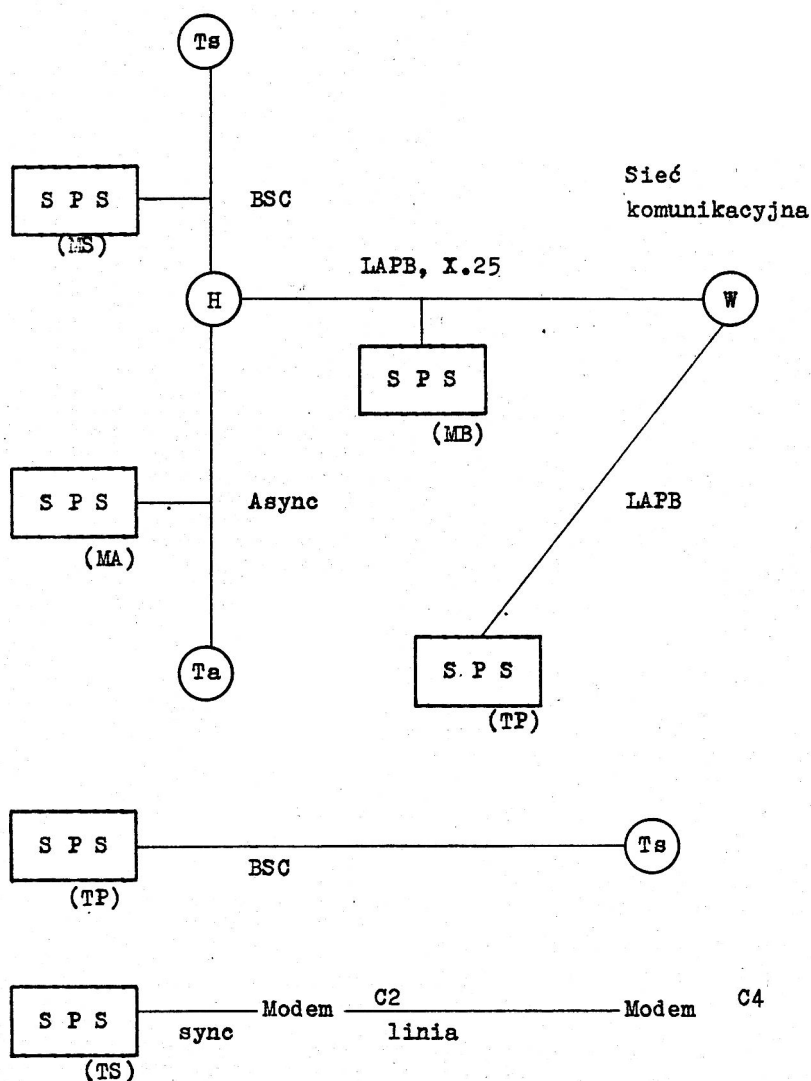
Do pułapek, które mogą być zdefiniowane dla wszystkich typów monitorowania, należą: upływ czasu trwania pomiaru, n-krotne wystąpienie zdefiniowanego wzorca (sekwencji 1-6 bajtów) i zapełnienie bufora rejestrującego. W czasie prowadzenia pomiaru są aktualizowane liczniki zdarzeń zdefiniowanych dla danego typu monitorowania, a obserwowane jednostki danych protokołu są wprowadzane do bufora cyklicznego. Uzyskane wyniki mogą być wyprowadzone na ekran w postaci raportu w trakcie wykonywania pomiaru i automatycznie są wyprowadzane po zakończeniu pomiaru.

Zainicjowany pomiar może być w każdej chwili zatrzymany z wyprowadzeniem dotychczasowych wyników, a następnie powtórzony dla tych samych warunków pomiaru. Po zakończeniu pomiaru zawartość bufora cyklicznego może być wyświetlana na ekranie za pomocą komend redagujących. Za ich pomocą można zdefiniować wyświetlany strumień (RX, TX, RX+TX), sposób wyświetlania (proceduralny, nieproceduralny), długość wyświetlanych jednostek (krótka, długa) i ich liczbę. Zawartość bufora jest wyprowadzana na ekran w postaci wydzielonych jednostek protokołu, z których każda opatrzona jest standardowym nagłówkiem określającym: rodzaj strumienia, stan (wskaźnik poprawności) jej odebrania oraz - opcjonalnie - czas odbioru jednostki.

Wykrycie warunku umożliwiającego wykonanie komendy w zadany sposób powoduje wyprowadzenie odpowiedniego komunikatu i zaniechanie akcji.

2. FUNKCJE SYSTEMU SPS

Możliwe zastosowania systemu SPS i miejsca jego włączenia w sieci komputerowej przedstawiono schematycznie na rysunku. Przy liniach transmisyjnych zaznaczono rodzaj (protokół) prowadzonej transmisji, a w nawiasach podano tryb pracy SPS właściwy dla danego pomiaru (MS - monitorowanie synchroniczne, MA - monitorowanie asynchroniczne, MB - monitorowanie protokołów bitowych, TS - testowanie standardowe, TP - testowanie programowane). Monitorowanie stanu linii styku S2 (MI), czego nie pokazano na rysunku, może być prowadzone w dowolnym miejscu sieci. Wyróżniono następujące elementy sieci: Ta - terminal asynchroniczny, Ts - terminal synchroniczny, H - host, W - węzeł sieci komunikacyjnej.



Rys. Możliwe zastosowania analizatora SPS w sieci KASK
 Fig. Applications of the SPS data analyzer in the KASK network

2.1. Monitorowanie obwodów styku S2

Procedura monitorowania, wywoływana komendą MI, powoduje monitorowanie 14 obwodów styku S2 o numerach 103-115 i 125 przy użyciu pakietu sprzęgającego A. Monitorowanie odbywa się za pomocą rejestrowania stanu logicznego obwodu (ON, OFF) w zadanych odstępach czasu. Po zakończeniu próbkowania (75 próbkowań) jest wyprowadzany raport końcowy w postaci wykresu stanu linii.

2.2. Monitorowanie transmisji według protokołów asynchronicznych

Za pomocą pakietu sprzęgającego B (z układem 8251) oraz komendy MA monitorowana jest transmisja półdupleksowa, której szczegółowe parametry, tj. szybkość transmisji (50-9600), długość znaku (5-8), długość bitu "stop" (1, 1.5, 2), rodzaj parzystości (EVEN, ODD, NONE) są definiowane w parametrach komendy. Oprócz wymienionych poprzednio, pułapką może być n-krotne wystąpienie błędu parzystości. Podczas monitorowania jest liczona liczba znaków, liczba błędów parzystości i liczba wystąpień wzorca.

2.3. Monitorowanie transmisji według protokołu synchronicznego BSC

Transmisja według protokołu synchronicznego BSC (komenda MS) również jest monitorowana za pomocą pakietu sprzęgającego B. Szczegółowe parametry, tj. kod transmisji (ASCII, EBCDIC), szybkość transmisji (600-9600) i rodzaj parzystości, są ustalane podczas definiowania komendy. Istnieje możliwość zdefiniowania pułapek tak, jak dla komendy MA. Podczas monitorowania mierzona jest liczba błędów parzystości, liczba wystąpień wzorca, liczba komunikatów rozpoczynających się od określonego znaku (ENQ, EOT, STX, SOH, ACKO, ACK1, NACK, RVI, WACK, TTD). Przeglądanie bufora cyklicznego może się odbywać w trybie nieproceduralnym (wyświetlanie komunikatów określonego strumienia opatrzonych nagłówkiem w postaci szesnastkowej i w kodzie ASCII) lub w trybie proceduralnym (znaki sterujące są wyświetlane w postaci ich mnemoników, pozostałe zaś w postaci szesnastkowej).

2.4. Monitorowanie transmisji

według protokołów synchronicznych zorientowanych bitowo

Transmisje prowadzone według protokołów LAPB, SDLC i X25 mogą być monitorowane z zastosowaniem pakietu sprzęgającego C (z układem 8273) za pomocą komendy MB. Szczegółowe parametry transmisji podawane przy definiowaniu komendy to typ protokołu (LAPB, SDLC, X25), szybkość transmisji (600-9600), miejsce włączenia SPS w system teleprzetwarzania (lokalnie, zdalnie) i sposób kodowania (NRZI, NRZ). Pułapkami charakterystycznymi dla tego rodzaju monitorowania są: limit dopuszczalnych błędów FCS, limit sekwencji ABORT, liczba jednostek protokołu określonej grupy. Dla protokołów LAPB i SDLC wyróżniono trzy grupy ramek: sterujące, nadzorcze i informacyjne, a dla protokołu X.25 - cztery grupy pakietów nawiązywania połączenia, informacyjne, sterujące i restartu.

Po zakończeniu pomiaru przeglądanie bufora cyklicznego może odbywać się w trybie nieproceduralnym w postaci jak dla monitorowania synchronicznego, oraz proceduralnym, zależnym od typu protokołu. Dla protokołów LAPB i SDLC w trybie proceduralnym ramka jest rozpakowywana i osobno wyświetlany jest: adres ramki (szesnastkowo), typ ramki (rozkaz, odpowiedź) i wartość bitu P/F, rodzaj ramki (I, S, U), mnemonik ramki, a dla ramek numerowanych numer ramki i potwierdzenia odpowiednio po komentarzu $N(S)=$ i $N(R)=$.

Dla protokołu X25 jest rozpakowywany pakiet i osobno wyświetla się GPI, numer kanału logicznego, mnemonik pakietu, numer pakietu i numer potwierdzenia z komentarzami odpowiednio $P(S)=$ i $P(R)=$.

2.5. Testowanie standardowe

Prowadzenie testowania standardowego (inicjowane komendą TT) wymaga podania parametrów testowanego układu, tj. rodzaju łącza (dupleks, półdupleks) i rodzaju układu (dla półdupleksu - nadawanie, odbiór, nadawanie i odbiór, dla duplexu - jednomaszynowy, dwumaszynowy). Długość testowania może być określona przez podanie czasu (nie więcej niż 4 h) lub liczby powtórzeń testu. Podczas testowania mierzona jest liczba błędnych bitów i błędnych bloków. Różne rodzaje zakończeń transmisji (poprawne i błędne) są sygnalizowane wyprowadzeniem odpowiedniego komunikatu w raporcie końcowym.

2.6. Testowanie programowane

Testowanie programowane wymaga wprowadzenia zestawu danych stanowiących test wysyłany do sieci oraz programu nadzorującego generację testu. Zdefiniowanie buforów zawierających wysyłane dane odbywa się przed określeniem warunków testowania i programu testującego, co realizowane jest w komendzie TP. Konfiguracja testowanego układu określona jest przez podanie szybkości transmisji (600-9600), rodzaju łącza (dupleks, półdupleks), klasy protokołu (bitowy, znakowy). Dla protokołów znakowych dodatkowo określa się długość znaku (5-8 bitów), liczbę znaków synchronizacyjnych (1; 1,5; 2), wzorzec znaku synchronizacji oraz rodzaj parzystości. Dla protokołów bitowych określa się typ protokołu (HDLC, SDLC) oraz sposób kodowania (NRZI, NRZ).

Program testujący wprowadzany jest po zdefiniowaniu warunków testowania i może być redagowany. Opisuje on sekwencję działań wykonywanych przez SPS.

Program testowy jest pisany w specjalnym języku programowania testów TPL. Jest to język, który operuje na następujących strukturach danych:

- buforach (zawierających komunikaty/ramki/pakiety)

- licznikach
- stoperach.

Bufory stanowią podstawowe struktury danych, a zawarte w nich informacje są transmitowane do/z urządzenia, porównywane z wzorcami i analizowane. Bufory mogą być zapisywane przez użytkownika SPS (do wysyłania w linię) lub wypełniane danymi odbieranymi z linii. W aktualnej wersji urządzenia wygenerowano 16 buforów, których rozmiary może ustalić użytkownik (4-2400 bajtów).

Liczniki i stopery są pomocniczymi strukturami danych, które umożliwiają organizację pętli (liczniki) oraz odmierzanie czasu (stopery). W aktualnej wersji istnieje 8 liczników dwubajtowych oraz 8 stoperów do odmierzania czasu z zakresu od 20 ms do 5 s.

W języku TPL istnieje 21 instrukcji, w tym:

- 4 instrukcje licznikowe
- 8 instrukcji skoków
- 2 instrukcje dla stoperów
- 2 instrukcje transmisyjne (nadawanie/odbior bufora)
- 3 instrukcje operacji na buforach
- 1 instrukcja zawieszenia programu (na podany czas)
- 1 instrukcja stopu.

Możliwość realizacji transmisji dwupiętowej w programie sekwencyjnym uzyskano przez przyjęcie zasady komunikacji bezpotwierdzeniowej. Po zainicjowaniu nadawania i odbioru program może sprawdzić (w dowolnej chwili) wykonanie każdej z tych operacji (nie trzeba czekać w pętli na ich wykonanie).

Program testujący nie może modyfikować treści buforów przygotowanych przez użytkownika SPS. Po zakończeniu testowania wyprowadzany jest raport końcowy, który podaje warunki testowania, typ zakończenia testu oraz wyprowadza liczbę wykorzystywanych buforów, stoperów i liczników, a także stan liczników. Użytkownik SPS może oglądać treść wysyłanych lub odebranych buforów, które mogą być wyświetlane w kodzie szesnastkowym, ASCII i dwójkowym.

Podano przykładowy program do komunikacji SPS z węzłem sieci MSK na poziomie protokołu liniowego. Bufory B0, B2 i B4 zostały wypełnione przez operatora SPS odpowiednio ramkami SABM, UA i I (z pakietem RESTART INDICATION):

1	10	SEND	B0	;nadaj SABM
2	50	WAIT	5	;czekaj 0.1 ms
3	43	JNS	15	;błąd nadaw
4	20	RCEV	B1	;odbieraj do B1
5	50	WAIT	20	;czekaj 0.4 s
6	44	JNR	1	;próbuj od nowa

7	30	GMP	B1,BO	;czy odebrano UA?
8	42	JNEO	14	;brak UA
9	10	SEND	B4	;nadaaj RESTART
10	50	WAIT	5	;czekaj
11	43	JNS	15	;błąd nadaw
12	20	RCEV	B3	;odbieraj
13	50	WAIT	50	;czekaj 1 s
14	00	HAIT		;stop poprawny
15	00	HAIT		;stop niepoprawny

Użytkownik pisze program wykorzystując prosty edytor tekstu wbudowany w system. Edytor ten wykrywa podstawowe błędy składniowe wprowadzanego programu. Napisany program wykonywany jest interpretacyjnie, przy czym stwierdzone błędy semantyczne powodują zakończenie wykonywania programu i sygnalizację błędu.

3. WNIOSKI

Wstępna eksploatacja systemu SPS, zarówno przez konstruktorów w czasie uruchamiania przyrządu, jak i przez pierwszych profesjonalnych użytkowników w warunkach normalnej pracy, pozwala sformułować wnioski odnośnie do jego przydatności i zaproponować zmiany, które powinny być wprowadzone w nowej wersji dla sieci KASK.

Ogólna koncepcja przyrządu wydaje się właściwa, a zakres spełnianych funkcji wystarczający. Wniosek taki potwierdza fakt, że SPS spełnia wszystkie funkcje realizowane przez analizatory danych (nazywane też analizatorami protokołów), oferowane przez renomowane firmy. Analizatory te różnią się między sobą rozwiązaniami konstrukcyjnymi, zakresem parametrów transmisji, liczbą implementowanych protokołów i dostępnych styków, jak również sposobem komunikacji z operatorem. Reprezentatywnymi przykładami są tu DA-10 (Wandel-Goltermann) [2] czy K1195 (Siemens) [3].

Największym błędem SPS z punktu widzenia użytkownika jest rozwiązanie konstrukcyjne przyrządu, którego prototyp zbudowano ze standardowych modułów systemu mikroprocesorowego MSM (Impol 1). Cała aparatura składa się z czterech części połączonych kablami i osobno zasilanych:

- kasety (standard 19 cali) zawierającej pakiety mikrokomputera i specjalizowane pakiety sprzęgające
- monitora ekranowego
- stacji dysków elastycznych
- klawiatury,

co sprawia, że urządzenie jest niewygodne w transporcie i instalowaniu.

Pewnym ograniczeniem funkcjonalnym, szczególnie w perspektywie przyszłych zastosowań, jest graniczna szybkość transmisji wynosząca w SPS 9600 b/s. Ograniczenie to wynika z zastosowania procesora 8085 (3 MHz) dostępnego w czasie budowy prototypu.

Dużym ułatwieniem w pracy z analizatorem byłoby wykorzystanie pamięci dyskowej zarówno do przechowywania definicji pomiaru, jak i do pamiętania wyników, np. pamiętania przebiegu transmisji w dłuższym okresie, niż pozwala na to ograniczony bufor w pamięci operacyjnej. Obecnie można w SPS przechowywać na dysku jedynie parametry komend pomiarowych, nie można natomiast przygotować tam np. programu testującego, gdy właśnie redagowanie testu (przy testowaniu programowym) jest czasochłonne, a przy niebanalnych testach programy i towarzyszące im struktury danych są złożone.

Wszystkie procesy monitorowania transmisji powinny mieć możliwość rozpoczynania rejestracji danych dopiero po wystąpieniu zdarzeń charakterystycznych - definiowanych przez operatora, analogicznie do obecnie zrealizowanych "pułapek" kończących pomiar. Takie rozszerzenie funkcji pozwoli operatorowi rejestrować interesujące go fragmenty transmisji, np. po błędzie określonego typu, po wystąpieniu określonej sekwencji znaków w danym strumieniu itp.

Realizacja ulepszeń operatorskich jest możliwa przez modyfikację obecnego prototypu SPS. Jego system operacyjny, zbudowany na zasadzie komunikujących się procesów [4], umożliwia dodanie nowych modułów lub przeprogramowanie istniejących, a pewna rezerwa pamięci operacyjnej wskazuje na wykonalność takiej rozbudowy.

Nie da się tego powiedzieć o dwóch pierwszych wymienionych cechach konstrukcji mechanicznej i szybkości pracy. Nowa wersja analizatora powinna umożliwiać transmisję z szybkością 72 kb/s lub przynajmniej 48 kb/s, aby mogła być przydatna w najbliższych latach. Taka szybkość wymaga zastosowania elementów nowej generacji (np. rodziny 8086 lub M68000) i prawdopodobnie zbudowania systemu wielomikroprocesorowego z układowym rozdzieleniem funkcji sterowania transmisją i obsługi operatora.

Oprogramowanie tego typu systemu, ze względu na swą złożoność, powinno być napisane w języku wyższego poziomu niż assembler, aby ułatwić proces projektowania i uruchamiania. Te przesłanki sugerują przyjęcie jako punktu wyjściowego jakiegoś gotowego mikrokomputera, który po wyposażeniu w specjalizowany moduł sprzęgu z torem transmisyjnym mógłby zostać oprogramowany z wykorzystaniem funkcji istniejącego systemu operacyjnego.

LITERATURA

- [1] DUBIELEWICZ I. i in, Aparatura pomiarowo-diagnostyczna dla sieci SKJS/2, CO PWr., Raport SPR nr 12/86, Wrocław 1986.
- [2] Data Analyzer DA-10. Prospekt firmy Wandel & Goltermann.
- [3] Protocol Tester K1195. Product Description. Siemens, 1986.
- [4] FRYŻLEWICZ Z., Mikrokomputerowy analizator protokołów. Informatyka 2/88.

APPLICATION OF THE SPS DATA ANALYZER AND TESTER IN THE KASK NETWORK

The SPS is a measuring and diagnostic system connected with DTE-DCE interface in communication network. It can be used for monitoring data streams and for testing the equipment in the network.

The paper presents basic monitor modes of the SPS and relevant commands for various transmission protocols. The principles of programmed testing are described. It is shown how SPS can be programmed in an easy-to-learn TPL language to fulfill requirements of various protocols.

ПРИМЕНЕНИЕ ИЗМЕРИТЕЛЬНО-ДИАГНОСТИЧЕСКОЙ АППАРАТУРЫ SPS В СЕТИ KASK

Представлено функции измерительно-диагностической системы названной SPS, которая предназначена для анализа потоков информации в сети передачи данных, а также для тестирования самой сетевой аппаратуры.

Обсуждается измерительные команды и их параметры определяющие условия измерения и вид получаемых результатов.

Описывается простой язык программирования тестов TPL (структуры данных и инструкции), на котором можно формулировать тесты для разных типов протоколов и разных параметров передачи.

Praca wpłynęła do Redakcji w marcu 1988 r.
w ostatecznej formie w lipcu 1989 r.

*Network protocols,
conformance testing*

Iwona DUBIELEWICZ*
Krystyna KOLEŚNIK*

SYSTEM TESTOWANIA ZGODNOŚCI IMPLEMENTACJI PROTOKOŁÓW W SIECI KASK

Przedstawiono koncepcję systemu testowania zgodności implementacji protokołu z jego standardem. System ten ma być przeznaczony do testowania protokołu transportowego. W pracy omówiono: typy testów, systemy testowania, strukturę testów zgodności i jej elementy oraz scharakteryzowano kombinowaną tablicowo-drzewową metodę notacji testów. Zaproponowana konfiguracja systemu testowania zgodności dla sieci KASK to system bazujący na metodzie testowania rozproszonego, który składa się z 2 części: testera aktywnego, działającego jako interpreter scenariuszy i testera-respondera, który działa jako automat skończony. Przedstawiono również strukturę testera aktywnego oraz omówiono funkcje jego modułów. Zaproponowano realizację systemu opartą na mikrokomputerze kompatybilnym z IBM PC/AT oraz omówiono specjalistyczne rozwiązania sprzętowe, wymagane do realizacji tego systemu.

1. WPROWADZENIE

Pojęcie testowania zgodności (ang. conformance testing) pojawiło się w początkach lat 80. w związku z potrzebą oceny i weryfikacji poprawności implementacji protokołów sieciowych z przyjętymi przez ISO ich standardami.

Prace podkomitetu SC 21 ISO zakładają, że standaryzacją obejmie się testy zgodności protokołów warstw 4, 5, 6 [6], [13]. Obecnie większość prac dotyczy testowania protokołu warstwy 4 (transportowej), ponieważ dla protokołu tej warstwy istnieje standard (IS 8073) [7], [8], [10], [12].

* Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

Celem prezentowanej pracy jest przedstawienie koncepcji systemu testowania zgodności dla warstwy transportowej sieci KASK. System ten będzie realizowany na komputerze kompatybilnym z IBM PC/AT. Zakłada się, iż projektowany system będzie uniwersalny w tym sensie, że pozwoli nie tylko na wykorzystywanie standardowych testów zgodności, lecz również na definiowanie przez użytkownika własnych testów, co umożliwi zastosowanie tego urządzenia do różnego rodzaju testowania.

W pierwszej części pracy przedstawiono podstawowe pojęcia testowania zgodności i metodologii testowania. Część druga zawiera opis koncepcji systemu testowania dla sieci KASK (konfiguracja, oprogramowanie, sprzęt). W zakończeniu oceniono prezentowaną koncepcję oraz podano uwagi dotyczące stosowania projektowanego systemu.

2. TESTOWANIE ZGODNOŚCI - POJĘCIA PODSTAWOWE, METODOLOGIA, TESTY

Zasadniczym celem testowania zgodności jest rozstrzygnięcie czy implementacja podlegająca testowaniu (ang. Implementation Under Test - IUT) wykazuje zgodność ze specyfikacją odpowiedniego standardu. Aspekty techniczne i ekonomiczne prowadzenia testowania ograniczają tak postawiony cel. Z tego względu wyróżnia się 3 rodzaje testów:

- bazowe testy współdziałania
- testy zgodności
- testy rozstrzygające.

Bazowe testy współdziałania są podstawą do wykrycia podstawowej niezgodności oraz wstępnej oceny zgodności przed podjęciem bardziej kosztownych etapów testowania.

Testy zgodności są narzędziami do praktycznego sprawdzenia, czy implementacja spełnia pełny zakres wymagań określonych w standardzie. Jest oczywiste, że testowanie takie nie może być kompletne ze względu na nieograniczoną liczbę zdarzeń, parametrów i zależności czasowych. Pomyślne przejście testowania zgodności jest podstawą do sprzedaży i nabywania produktu, o którym stwierdza się, że jest zgodny ze standardem.

Testy rozstrzygające dostarczają odpowiedzi diagnostycznej i rozstrzygają, czy rozważana implementacja spełnia pewne szczegółowe wymagania (dotyczą one bardzo wąskiego fragmentu implementacji). Są one również przydatne na etapie testowania niektórych elementów implementacji, zależnej od systemu, z którym ona współpracuje.

W dalszej części pracy przedmiotem rozważań są testy zgodności, ponieważ mają podlegać one standaryzacji.

2.1. Metodologia testowania

Testowanie implementacji protokołu wyrażone w języku stosowanym do opisu norm protokołów i usług to testowanie stacji realizującej protokół warstwy N (w skrócie N-stacji). Testowanie to można prowadzić wysyłając/odbierając do/z tej stacji prymitywy usług (ang. Abstract Service Primitive - ASP) z warstw (N+1) i (N-1), przy czym zakłada się, że warstwy (N-1), (N-2), itd. stanowią implementację wzorcową. Ze względu na sposób i miejsce wysyłania prymitywów do N-stacji, wyróżnia się 3 metody testowania: lokalne, zdalne, rozproszone.

W testowaniu lokalnym [13], [14] system testujący jest umieszczony w tym samym środowisku programowym co IUT i steruje oraz obserwuje zachowanie się N-stacji wysyłając lokalne (N-1)-ASP i (N+1)-ASP.

Metoda druga [6], [13] dopuszcza możliwość wyłącznie zdalnego sterowania N-stacją przez zdalne wysyłanie prymitywów (N-1). Obserwacja zachowania sprowadza się do analizy uzyskanych zwrotnie (zdalnych) (N-1)-ASP.

Testowanie rozproszone stanowi pewne połączenie obu metod. System testujący składa się z dwu elementów, tzw. testera aktywnego (ang. active tester lower tester lub test driver) oraz respondera testu (ang. test responder lub upper tester). Tester aktywny (nazywany dalej również testerem) steruje i obserwuje zachowanie N-stacji przez wysyłanie/odbiór zdalnych (N-1)-ASP. Natomiast responder testu (nazywany również dalej responderem) jest umieszczony w tym samym środowisku co IUT i steruje N-stacją przez lokalne (N+1)-ASP. Istotny problem, jaki powstaje przy tej metodzie testowania, to koordynacja pracy obu elementów. Trudność tę rozwiązuje się przez definiowanie specjalnych protokołów (ang. Test Driver - Responder Protocol-TDRP), które służą uzyskaniu synchronizacji pracy obu elementów.

Istnieją też mutacje wymienionych metod. Możliwe jest bowiem testowanie pojedynczej N-warstwy (ang. single layer) lub (N+a) warstw (ang. multi layer) czy też N-warstwy zanurzonej we wszystkich pozostałych warstwach (ang. embedded layer) [13].

Najpeźniejszą metodą testowania jest metoda testowania rozproszonego [1], [10], [11], [14], w której sterowanie IUT jest realizowane przez wykorzystywanie usług (N-1) warstwy. Pozwala to na wbudowanie w test błędnych sekwencji usług i jednostek protokołu (ang. Protocol Data Unit-PDU), jak również kodowanie błędnych PDU (ostatnie dwie możliwości są realne przy budowie Testera jako kodera/dekodera PDU).

2.2. Struktury testów zgodności

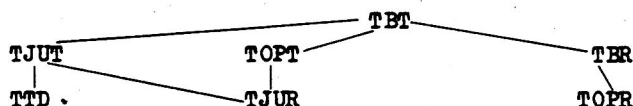
Testy zgodności mają strukturę hierarchiczną [4]-[6]. Komplet testów zgodności (ang. conformance test suit), służący do testowania obowiązkowych i opcjonalnych parametrów protokołów w ich pełnym zakresie wartości, składa się z grup testów (ang. test group). Można np. wyróżnić następujące, przykładowe grupy testów poziomu transportowego:

- testy transmisji danych w jednym połączeniu transportowym
- testy równoczesnych transmisji danych w wielu połączeniach transportowych
- testy stymulowanych sytuacji nie przewidzianych protokołem
- testy kolizji.

Każda grupa testów składa się z podgrup testów (ang. test subgroup) w wersjach inicjowanych przez Tester lub Responder. Podgrupy testów służą do szczegółowego testowania ustalonych grup parametrów różnych faz protokołów i muszą być wykonywane w określonej sekwencji. Kolejność ich wykonywania określa struktura drzewiasta. Na przykład grupa testów prostych połączeń transportowych może zawierać następujące podgrupy:

- bazowe testy nawiązywania połączenia inicjowane przez tester (TBT)
- bazowe testy nawiązywania połączenia inicjowane przez responder (TBR)
- testy jakości usług inicjowane przez Tester (TJUT)
- testy jakości usług inicjowane przez Responder (TJUR)
- testy transferu danych (TTD)
- testy odrzucania połączeń inicjowane przez Tester (TOPT)
- testy odrzucania połączeń inicjowane przez Responder (TOPR).

Strukturę drzewa testów dla tej podgrupy testów pokazano na rys.1.



Rys. 1. Struktura grupy testów transmisji danych

Fig. 1. The structure of the data transmission tests group

Elementem składowym podgrupy testów jest tzw. test abstrakcyjny, zwany też szkieletem testu. Składa się on z możliwych sekwencji zdarzeń zachodzących podczas działania fazy protokołu, której szczegółowe badanie jest celem podgrupy. Test abstrakcyjny ma strukturę sieciową, będącą uogólnieniem struktury drzewiastej w tym sensie, że niektóre fragmenty drzewa mogą się powtarzać. Szkielec testu nie zawiera wartości para-

metrów podanych podczas specyfikacji testu. Jedna ścieżka takiej struktury nazywa się abstrakcyjnym egzemplarzem testu (ang. abstract test case). Innymi słowy, podgrupa testów składa się z abstrakcyjnych egzemplarzy testów, będących ścieżkami w strukturze sieciowej każdego testu podgrupy. Uzupełnienie abstrakcyjnego egzemplarza testu wartościami parametrów przekształca go w rzeczywisty egzemplarz testu (ang. executable test case), nazywany też scenariuszem testu. Każdemu abstrakcyjnemu egzemplarzowi testu odpowiada wiele testów rzeczywistych pokrywających ustalone zakresy wartości parametrów testu.

Test rzeczywisty jest opisany za pomocą sekwencji zdarzeń inicjowanych przez obiekt testujący lub spodziewanych od obiektu testowanego. Do badania poziomu transportowego zdarzeniami w teście dokonywanym przez tester są sekwencje usług protokołu sieciowego, których parametrami są jednostki danych protokołu transportowego. Sekwencja zdarzeń powtarzających się w wielu testach rzeczywistych, nazywana krokiem testu, może być wydzielona w postaci odrębnej ścieżki opatrzonej nazwą, na którą można powoływać się w tych testach bez potrzeby ponownego specyfikowania jej treści.

2.3. Metody opisu testów

Najbardziej dopracowaną w tej chwili i zalecaną przez ISO metodą specyfikacji testu zgodności jest tzw. kombinowana technika tablicowo-drzewowa (ang. Tree and Tabular Combined Notation - TTCN). Technika ta jest standaryzowana [7].

Opis testu według TTCN składa się z 3 części:

- sekcji deklaracji opisującej elementarne zdarzenia i ich parametry wykorzystywane w treści testu
- sekcji ograniczeń zawierającej zestawy wartości parametrów wykorzystywanych w sekcjach deklaracji i części dynamicznej
- sekcji części dynamicznej zawierającej właściwy opis zachowania się testera w postaci prymitywów usług generowanych lub odbieranych: część dynamiczna ma także przypisany tekstowy, nieformalny opis celu testu i wymagań synchronizacyjnych.

Prymitywy usług i jednostki danych protokołu wykorzystywane w opisie zdarzenia testu są zdefiniowane osobno przez zestaw danych obejmujący: identyfikator, postać testową (PDU i ASP) i sposób kodowania, pełny komentarz objaśniający wykorzystywanie, wykaz parametrów, ich specyfikację i zestaw wartości. Dodatkowo dla PDU należy określić ich format.

W sekcji ograniczeń podaje się listy parametrów poszczególnych ASP i PDU dla testów rzeczywistych. Listy te tworzy się przez kombinacje możliwych wartości wszystkich obowiązujących dla danego poziomu testowania parametrów, będących atrybutami ASP lub PDU, które powinny być obje-

te testowaniem (ale nie dla wszystkich możliwych wartości tych parametrów!). Na przykład dla parametrów wielowartościowych testowaniem powinny być objęte wartości ekstremalne i jedna wartość pośrednia.

W sekcji części dynamicznej postać zdarzenia może przyjąć format:

<nazwa kroku>	<identyfikator inicjatora>	<identyfikator usługi>	<etykieta>	<odsyłacz do PDU i ASP>
np:	S!	N_CON_RQ		CONNECT

Zastąpienie nazwy usługi nazwą kroku prefiksowaną przez + oznacza miejsce włączenia podanego kroku testu, zaś dołączenie znaku $\rightarrow$ z etykietą w obrębie testu pozwala wskazywać miejsce kontynuacji testu. Cykliczne n-krotne powtarzanie zdarzenia zaznaczane jest jako wykładnik potęgi 'n' po nazwie zdarzenia, natomiast powtarzany fragment testu jest umieszczony w nawiasach '()', przy czym po nawiasie zamykającym podaje się liczbę powtórzeń. Parametry wykorzystywane w zapisie zdarzenia są oznaczane identyfikatorami.

Przewiduje się także możliwość wykorzystywania w języku [4]:

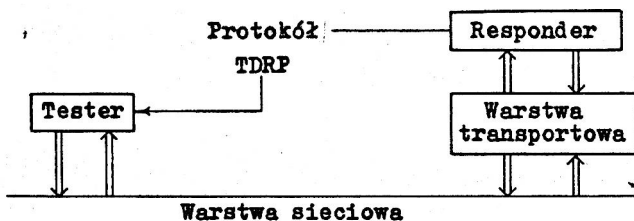
- tzw. pseudozdarzeń, np. WAIT do obsługi problemów związanych z zapisem w tekście upływu czasu
- wyrażeń służących do ustalania wartości parametrów
- przekazywania listy parametrów aktualnych podczas wywoływania poddrzewa.

Ze względu na prostotę języka opisu testu wykorzystywanego w TTCN, jego implementacja jest stosunkowo łatwa. Wykorzystanie do opisu testów jednej z technik formalnych stosowanych do specyfikacji usług i protokołów sieciowych, tj. LOTOS czy ESTELLE [2],[3], wymagałoby znacznie większych nakładów pracy w ich implementacji.

3. KONCEPCJA TESTOWANIA ZGODNOŚCI W SIECI KASK

3.1. Konfiguracja systemu testowania zgodności

Przyjęto, że testowanie zgodności w sieci KASK będzie prowadzone w systemie rozproszonym. Testowanie to będzie realizowane dla warstwy 4



Rys. 2. Konfiguracja systemu testowania zgodności sieci KASK

Fig. 2. Configuration of conformance testing system for KASK network

(transportowej), jakkolwiek zakłada się, że w przyszłości zostanie rozszerzone o możliwość testowania warstw wyższych. Prowadzenie tego testowania zakłada, że warstwa sieciowa i niższe zarówno w systemie testera, jak i w systemie komputerowym warstwy testowanej, stanowią implementację wzorcową.

Schemat projektowanego systemu przedstawiono na rys. 2.

TESTER umieszczony w środowisku programowym mikrohosta [9] i realizowany na mikrokomputerze typu IBM PC/AT będzie implementowany jako kodier/dekoder jednostek protokołu transportowego (taki tryb pracy według pracy [11] pozwala na prowadzenie najbardziej wszechstronnego testowania).

Responder umieszczony na poziomie warstwy sesji, będzie współpracował z testowaną warstwą transportową odpowiedniego komputera (RIAD, Odra, IBM, PC) wysyłając/odbierając prymitywy usług transportowych. Responder będzie implementowany jako automat skończony, którego tabele przejść i wyjść są definiowane zgodnie z wymaganiami prowadzonych testów.

Synchronizacja obu testerów zostanie zapewniona przez przyjęcie prostego protokołu TDRP. Protokół ten zawiera dwie podstawowe grupy komend:

- ustawiania kontekstu (stanu) początkowego Respondera
- odczytywania wartości parametrów/danych, informujących o aktualnym stanie Respondera.

Komendy wysyłane przez Tester wymuszają określone działania w Responderze, umożliwiając w ten sposób sterowanie i synchroniczną pracę obu elementów.

W dalszej części pracy rozważa się wyłącznie realizację Testera jako podstawowego komponentu systemu testowania. O Responderze zakłada się, że zostanie zrealizowany jako program łatwo przenośny na systemy komputerowe, które działają w sieci KASK. W związku z tym należy w nim wyróżnić dwie części:

- pierwsza, niezależna od systemu, stanowiąca jądro Respondera
- druga, zapewniająca styk do dowolnej realizacji warstwy transportowej.

Część pierwszą można zrealizować jednokrotnie w dowolnym języku wysokiego rzędu, natomiast druga musi być wymieniana przy każdej zmianie systemu komputerowego, jest bowiem od niego zależna.

3.2. Architektura programowa Testera

Zbudowanie testera uniwersalnego w zakresie możliwości badania różnych warstw protokołów oraz definiowania przez użytkownika własnych tes-

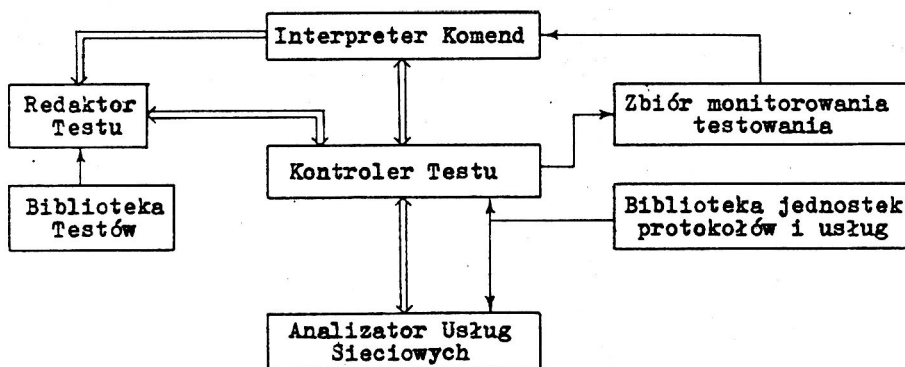
tów wymaga przyjęcia odpowiedniego rozwiązania programowego (programy sterowane danymi).

W związku z tym przyjmuje się, że:

- istnieje biblioteka testów, którą można modyfikować
- struktury jednostek danych protokołów i usług oraz wartości ich parametrów będą przechowywane w oddzielnych bibliotekach
- jest wyspecyfikowany precyzyjnie styl z warstwy sieciowej
- istnieje język opisu testu i jego interpreter, który pozwala wprowadzić do biblioteki testów nowo sformułowany test.

Przy tych założeniach strukturę oprogramowania przedstawiono na rys. 3. W strukturze tej wyróżniono 4 jednostki funkcjonalne:

- Interpreter Komend
- Redaktor Testu
- Kontroler Testu
- Analizator Usług Sieciowych



Rys. 3. Struktura funkcjonalna testera

Fig. 3. TESTER functional structure

Interpreter Komend pełni funkcję styku użytkownik-system. Komendy, które wprowadza użytkownik, umożliwiają:

- wprowadzenia wartości istotnych parametrów konfiguracji testowej (np. adresu punktu dostępu do usług transportowych, sesyjnych)
- wybór kategorii testu (bazowy, zgodności, rozstrzygający)
- wybór egzemplarza testu lub podgrupy testów
- definicję testu niestandardowego (jest udostępniany prosty edytor i analizator składni)
- wyprowadzenie i edycję uzyskanych wyników.

Redaktor Testu wykonuje funkcje wprowadzania, przechowywania i udostępniania testów do/z biblioteki testów. Dwie pierwsze funkcje są wykonywane w trybie off-line, tzn, przed uruchomieniem testowania, natomiast trzecia jest realizowana w trakcie testowania i dlatego istotny

jest sposób organizacji i dostępu do elementów biblioteki. Zakłada się, że w bibliotece będzie zapewniony bezpośredni dostęp do grupy i podgrupy testów. Natomiast pojedynczy test będzie generowany jako ścieżka wybranej podgrupy. Wygenerowany egzemplarz testu abstrakcyjnego zostanie wypełniony treścią wynikającą z aktualnie znanych i/lub ustalonych wartości parametrów testu (podanych przez użytkownika oraz przechowywanych w zbiorze parametrów konfiguracji). Redaktor testu generuje również zestaw list wartości parametrów, które w wybranym teście są przedmiotem sprawdzania. Uzyskany w ten sposób test rzeczywisty jest przekazywany do interpretacji Kontrolerowi Testu. Obecnie zakłada się, że test przekazany przez Redaktora Testu będzie zapisany w języku zbliżonym do notacji TTCN 4.

Kontroler Testu interpretuje test przekazany przez Redaktora Testu wraz z parametrami. Interpretację stanowią następujące kroki:

- generacja jednostki protokołu transportowego do wysłania
- przekazanie utworzonej PDU oraz żądania generacji odpowiedniej usługi sieciowej do Analizatora Usług Sieciowych
- odbiór danych od Analizatora Usług Sieciowych (PDU usługi)
- rejestrowanie wysyłanych/odbieranych jednostek protokołu do zbioru monitorowania testowania
- sprawdzanie rodzaju i wartości odebranych danych z danymi oczekiwanymi
- sygnalizacja końca interpretacji testu z podaniem przyczyny do Redaktora Testu.

Kontroler Testu jest asementyczny. Wszelkie dane sterujące jego pracą powinny być zawarte w opisie testu.

Analizator usług sieciowych realizuje "uogólniony" styk do komunikacji Kontrolera Testu z potencjalnie różnymi realizacjami warstwy sieciowej. Jego funkcją jest kodowanie i dekodowanie jednostek usług sieciowych, które są przekazywane do/z Kontrolera Testu.

W celu zapewnienia przenośności oprogramowania Testera, jak również możliwości współpracy ze środowiskiem mikrohosta, zakłada się, że zostanie ono napisane w języku C. Opisane komponenty systemu Testera zostaną zaimplementowane jako procesy w tym środowisku.

3.3. Rozwiązanie sprzętowe Testera

Przedstawiony system testowania zgodności będzie implementowany na mikrokomputerze kompatybilnym z IBM PC/AT. Jednak możliwość testowania protokołów sieciowych (pracujących w trybie synchronicznym, dwukierunkowym lub półdwukierunkowym) jest realna wyłącznie po wyposażeniu tego komputera w specjalizowany pakiet. Zaprojektowany pakiet, opisany w pracy [2], jest zgodny pod względem programowym i sprzętowym z pakietem IBM SDLC

Adapter oraz elektrycznie i mechanicznie z modelem EC8073 (styl V.24/RS 232C). Analogicznie jak w pakiecie oryginalnym jako sterownik transmisji został wykorzystany układ serii Intel 8073. Zaprojektowany pakiet zapewnia możliwość pracy z programowo wybieraną prędkością z zakresu 110-9600 Bd.

Zaproponowana w pracy koncepcja systemu testowania zgodności dla sieci KASK spełnia wymienione we wstępie założenia. System ten jest projektowany na tyle elastycznie, by oprócz swej podstawowej funkcji testowania zgodności, umożliwić również testowanie urządzeń sieciowych na etapie ich uruchamiania, jak i później w trakcie eksploatacji. Wydaje się, że jest on również uniwersalny w tym sensie, że modyfikacje polegające na zmianie biblioteki testów, biblioteki definicji PDU, usług i Respondera oraz definiowaniu nowego protokołu TDRP, pozwolą na prowadzenie testowania innej warstwy niż warstwa transportowa.

System ma szanse szerokiego rozpowszechnienia pod warunkiem zapewnienia dostępności specjalizowanego pakietu sterowania transmisją.

LITERATURA

- [1] ANSART J. P., GENEPI A., A Protocol Independent System for Testing Protocol Implementation w [16].
- [2] DUBIELEWICZ I. i in., Koncepcja testowania zgodności implementacji protokołów ze standardami. Założenia funkcjonalne, CO PWr., Raport SPR 8/87, Wrocław 1987.
- [3] ISO/TC 97/SC 6/WG 4 No 76, Formal Specification of IS8073 (Transport Protocol) in LOTOS, December 1985.
- [4] ISO/TC 97/SC 21/WG 1 No 215, Testing and Verification of Data Communication Protocol, November 1985.
- [5] ISO/TC 97/SC 21/WG 16-1 No 464, Conformance Testing and FDTs, December 1984.
- [6] ISO/TC 97/SC 21/WG 1 No 909, Revised Working Draft for OSI Conformance Testing Methodology and Framework, Toronto 1985.
- [7] ISO/TC 97/SC 21 No 1066, Conformance Test Suites for IS8073.
- [8] ISO/TC 97/SC 16 No 1994, Possible Test Suites for Class 0 Transport Protocol Conformance Testing. June 1984.
- [9] JANCZEWSKI K., STANISZ A., Mikrokomputer kompatybilny z IBM PC jako mikrohost w sieci KASK - założenia funkcjonalne, CO PWr., Raport SPR 4/87, Wrocław 1987.
- [10] LINN R. J., An Evaluation of IGST Test Architecture after Testing Class 4 Transport Protocol w [17].

- [11] PAVEL J. R., DWYER D. J., Some Experiences of Testing Protocol Implementations w [17].
- [12] RAFIQ O., HADDAD J., Description of Protocol Testing Scenarios w [15].
- [13] RAYNER D., Standarizing Conformance Testing for OSI w [15].
- [14] RAYNER D., A System for Testing Protocol Implementation w [16].
- [15] VISSER Ch. A., Standarization of Formal Description Techniques for Communication Protocols. W: Informat. Processing '86 . Proc. of IFIP 10 World Computer Congress 1986, North Holland 1986.
- [16] CSABA I., TARNAY K., SZENTIVANYI T.(eds.), Computer Network Usage, Recent Experiences, Proc. of the IFIP TC6 Working Conf. COMNET '85 North Holland 1986.
- [17] SUNSHINE C. A. (ed), Protocol Specification, Testing and Verification. Proc. of the IFIP WG 6.1 Second Int. Workshop North Holland 1982.

SYSTEM OF PROTOCOL IMPLEMENTATION CONFORMANCE TESTING IN KASK NETWORK

In the paper the proposal of solution of the system for conformance of the transport layer implementation for KASK network is presented.

At first, the basic notations of the conformance testing, testing methodology, test structures and their formal specifications are described. In the remaining part of the paper the configuration, software and hardware conceptions of the solution are presented. It is proposed to implement this conformance system as distributed one, the Test Driver (Lower Tester, Active Tester) of which is placed in the microcomputer IBM PC/AT compatible. To operate properly in this system this computer should be equipped with the IBM SDLC Adapter-like card. The software of the Test Driver is designed to operate in the scenario interpreter mode. The Test Responder (the second part of this system) will be placed in all the computer systems, which exist in KASK network i.e. RIAD, ODRA. It is suggested to realize it as finite automaton the transition table of which is defined according to protocol state transition table.

СИСТЕМА ТЕСТИРОВАНИЯ СООТВЕТСТВЕННОСТИ ИМПЛЕМЕНТАЦИИ ПРОТОКОЛОВ СЕТИ KASK

В работе представлена концепция системы тестирования соответственности имплементации протокола с его стандардом. Для этой системы предусматривается назначение для тестирования транспортного протокола.

В работе охарактеризовано типы тестов соответственности, системы тестирования, структуру тестов и ее элементы, а также характеризуется комбинированную таблично деревообразный метод номинации тестов.

Предлагаемая конфигурация системы тестирования соответствия для сети KASK это система базирующая на методе рассредоточенного тестирования, которая состоит из двух частей: активного тестера, действующего как интерпретатор сценариев и тестера респондера, который действует как конечный автомат.

В дальнейшей части работы представляется структура активного тестера и охарактеризованы функции его модулей. Предложена реализация системы основанной на микрокомпьютере совместимом с IBM PC/AT и охарактеризованы аппаратные решения необходимые для реализации этой системы.

Praca wpłynęła do Redakcji w marcu 1988 r.
w ostatecznej formie w lipcu 1989 r.

*Sieci komputerowe, lokalne
sieci komputerowe, projektowanie*

Elżbieta KOSMULSKA-BOCHENEK*

Elżbieta HUDYMA*

PROJEKTOWANIE SIECI LAN NA PRZYKŁADZIE CENTRUM OBLICZENIOWEGO

Przedstawiono problemy związane z projektowaniem lokalnej sieci komputerowej (LAN) w ośrodku. Rozróżniono dwie sytuacje: gdy ośrodek nie dysponuje żadnym sprzętem komputerowym oraz, gdy już taki posiada. Etapy projektowania zilustrowano na przykładzie Centrum Obliczeniowego.

1. WSTĘP

Projektowanie konfiguracji jakościowej i ilościowej lokalnej sieci komputerowej zależy od specyfiki ośrodka, w którym dana sieć ma pracować. Prawidłowe usytuowanie stacji sieciowych ma wpływ na efektywność pracy sieci rozumianą przede wszystkim jako ekonomiczne wykorzystanie zasobów, a także szybkość przesyłania komunikatów i zbiorów oraz niezawodność tego przesyłania [6].

Zadanie projektowania sieci LAN może być postawione w dwóch różnych sytuacjach wyjściowych istniejących w ośrodku. Pierwsza to sytuacja, gdy przedsiębiorstwo nie dysponuje żadnym mikrokomputerem i zakupiona konfiguracja LAN, wynikająca z potrzeb ośrodka, ma być tu jedynym źródłem usług informatycznych. W takim przypadku punktem wyjściowym projektowania jest analiza stanowisk pracy ośrodka pod kątem możliwości (i potrzeby) ich komputeryzacji, z czego wynikać będzie m. in. fizyczne

^x Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

rozmieszczenie stacji LAN, ich liczba i zasoby. Częściej jednak zachodzi sytuacja, że sieć LAN jest tworzona na istniejącej bazie sprzętowej i programowej przez uzupełnienie o elementy sieciowe. W tym drugim przypadku punktem wyjściowym do projektowania sieci jest stan faktyczny ośrodka, który musi być poddany wnikliwej analizie pod kątem potrzeb użytkowników, z uwzględnieniem ich topologii, a efektem powinien być optymalny rozdział zasobów w poszczególnych stacjach.

Projektowanie sieci, mającej być pierwszym efektem komputeryzacji ośrodka, daje liczne możliwości podejmowania decyzji zakupu sprzętu i oprogramowania oraz pozwala na zastosowanie odpowiednich kryteriów optymalizacyjnych w stosunku do lokalizacji poszczególnych stacji i rozmieszczenia w nich zasobów. W tym przypadku należy jednak liczyć się z możliwą niedoskonałością projektu, wynikającą z braku doświadczenia przyszłych użytkowników w specyfikacji potrzeb środowiska informatycznego.

Inaczej problem wygląda w zdeterminowanym środowisku informatycznym, gdzie użytkownicy znają swoje potrzeby i są w stanie przewidzieć kierunek ich wzrostu. W tym przypadku realizacja projektu może być obciążona narzuconym już rozmieszczeniem stacji LAN i związanymi z tym ewentualnymi opóźnieniami na liniach.

W dalszej części artykułu zostanie przedstawiony proces projektowania sieci LAN w sytuacji, gdy ośrodek dysponuje już określonym sprzętem informatycznym.

1. ETAPY PROJEKTOWANIA LAN W OŚRODKU

Zasadniczym celem łączenia mikrokomputerów w sieci lokalne jest zwiększenie mocy obliczeniowej istniejących stanowisk oraz ekonomiczne wykorzystanie zasobów (istniejących oraz planowanych). Stąd proces projektowania musi opierać się na stanie faktycznym ośrodka w sensie posiadanych zasobów informatycznych [4]. W procesie tym wyróżnić można następujące etapy:

- wyodrębnienie grup użytkowników i inwentaryzacja związanych z nimi zasobów

- zestawienie potrzeb wyodrębnionych grup użytkowników
- sporządzenie modelu użytkownika sieci
- dostosowanie i wybór usług sieciowych
- rozmieszczenie stacji i planowanie wyposażenia sieciowego stacji.

Wyodrębnienie grup użytkowników ma na celu analizę aktualnego stanu wykorzystania mikrokomputerów w pracach prowadzonych w ośrodku. Wynikiem pierwszego etapu projektowania powinna być lista grup użytkowników

wraz z posiadanymi aktualnymi konfiguracjami sprzętowymi oraz sprecyzowanymi zadaniami realizowanymi w każdym z nich. Należy na tym etapie uwzględnić ośrodki istniejące oraz planowane.

Celem drugiego kroku projektowania jest zestawienie potrzeb określonego w pierwszym etapie użytkownika grupowego pod kątem ilości i jakości zasobów. Potrzeby te wynikają z charakteru zadań realizowanych przez grupę, liczby potencjalnych użytkowników w grupie oraz planowanego rozwoju prac [3]. Ośrodki wyodrębnione w pierwszym kroku zostają zróżnicowane ze względu na zasoby. W wyniku tego etapu projektowania poszczególnym ośrodkom zostają przypisane wagi odzwierciedlające wielkość posiadanych (i planowanych) zasobów.

Modelem użytkowania sieci jest graf skierowany, którego węzły obrazują wyodrębnione grupy użytkowników, a łuki przepływ informacji, tzn. łączą użytkowników poszczególnych zasobów. Jednocześnie należy określić tu sposób komunikacji pomiędzy użytkownikami (komunikacja terminal-terminal, terminal-pamięć masowa, czy inne). Jest to podstawa do realizacji następnego kroku, jakim jest określenie typu usług realizowanych przez sieć [1],[2] oraz implementacja ich w odpowiednich punktach sieci. Przykładowo, poczta elektroniczna jest usługą, która wykorzystuje komunikację terminal-terminal, a transfer zbiorów komunikację terminal-pamięć zbiorów.

Następny etap, na podstawie wartości wag przypisanych w drugim kroku projektowania, weryfikuje aktualne rozmieszczenie mikrokomputerów w różnych punktach sieci pod kątem wzbogacenia stanowisk o dodatkowe zasoby oraz konieczne wyposażenie sieciowe dla każdej stacji LAN. Należy tu uwzględnić ewentualne zwiększenie liczby stacji.

Przykładem zastosowania przedstawionej procedury projektowania sieci LAN jest projekt lokalnej sieci komputerowej w Centrum Obliczeniowym Politechniki Wrocławskiej.

3. SIEĆ LAN W CENTRUM OBLICZENIOWYM (CO)

Istotnym celem wprowadzenia sieci LAN do Centrum Obliczeniowego jest zwiększenie mocy obliczeniowej mikrokomputerów IBM PC, rozmieszczonych w różnych budynkach i na różnych piętrach w obrębie budynku. Sieć lokalna w CO ma służyć dydaktyce, pracom badawczym oraz administracji.

W CO wyodrębniono pięć grup użytkowników:

- laboratorium dydaktyczne (w1)
- laboratorium badawcze (w2)
- pracownicy naukowo-dydaktyczni (w3)
- administracja 1 (sekretariat) (w4)
- administracja 2 (księgowość, zaopatrzenie) (w5).

Podział ten wynika zarówno z charakteru prac prowadzonych przez poszczególne grupy, jak i z ich geograficznego rozmieszczenia. Nadanie wag poszczególnym grupom użytkowników wynika z ilości już posiadanych zasobów oraz planów rozwoju.

Dla poszczególnych grup użytkowników podano w tabeli ilościowe zestawienie obecnie posiadanych sprzętowych zasobów informatycznych oraz oszacowanych potrzeb w tym zakresie. W kolumnie mk podano liczbę standardowych zestawów składających się z mikrokomputera IBM PC/XT lub IBM PC/AT z pamięcią RAM 640 kB, z dwoma stacjami dysków elastycznych każda po 360 kB, monitorem monochromatycznym współpracującym z kartą typu Herkules i drukarką mozaikową. W kolumnie PO określono dodatkowe zasoby pamięci operacyjnej, a w kolumnie UZ dodatkowe zasoby takie, jak pamięć na dysku twardym, specjalne typy drukarek i monitorów, streamery, plottery itp. wyposażenie mikrokomputerów.

T a b e l a

Zestawienie posiadanych i planowanych zasobów komputerowych dla poszczególnych grup użytkowników

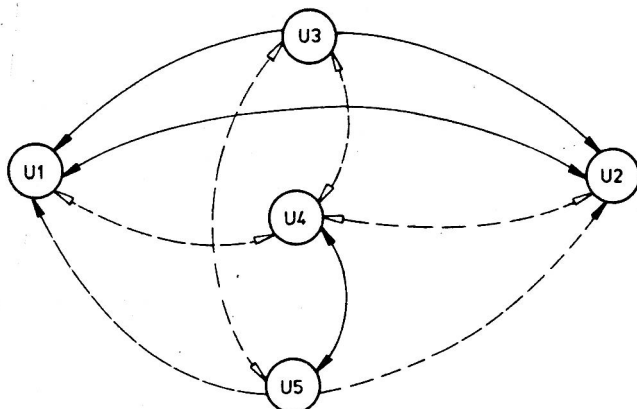
Grupa użytkowników	Posiadane zasoby			Planowane zasoby		
	mk	PO	UZ	mk	PO	UZ
w1	3	2 MB	-	8	16 MB	4 monitory kolorowe 4 drukarki kolorowe plotter 8 x 20 MB
w2	5	6 MB	-	8	24 MB	8 x 40 MB
w3	3	4 MB	-	5	20 MB	5 x 20 MB
w4	-	-	-	1	-	20 MB
w5	-	-	-	1	-	20 MB

Laboratorium dydaktyczne stanowi największy ośrodek mikrokomputerowy i tam potrzebna jest największa różnorodność zasobów ze względu na rodzaj prowadzonych zajęć dydaktycznych. Po analizie aktualnego stanu pozostałych użytkowników ustalono następujące wagi poszczególnych węzłów grafu; (w1)-5, (w2)-4, (w3)-3, (w4)-2, (w5)-1. Na podstawie tych danych powstał model użytkowania sieci jak na rys. 1.

W grafie tym linią ciągłą przedstawiono komunikację między węzłami z uwzględnieniem kierunku przepływu informacji pomiędzy użytkownikami, np. u3 korzysta z zasobów stacji u1 i u2, co pokazują strzałki. Linią przerywaną zaznaczono komunikację pomiędzy rzadko komunikującymi się użytkownikami.

Sieć LAN, ze względu na powszechne zastosowanie w CO, powinna mieć następujące możliwości:

- dostępność oprogramowania systemowego (systemy operacyjne, kompilatory, edytory itp.)



Rys. 1. Model użytkowania sieci

Fig. 1. Model of using of the network

- przechowywanie programów i danych
- dostarczanie informacji o zasobach sieci
- nadawanie i odbieranie komunikatów terminalowych
- świadczenie usług specjalnych, np. korzystanie z szybkiej drukarki, pamięci o dużej pojemności, urządzeń graficznych.

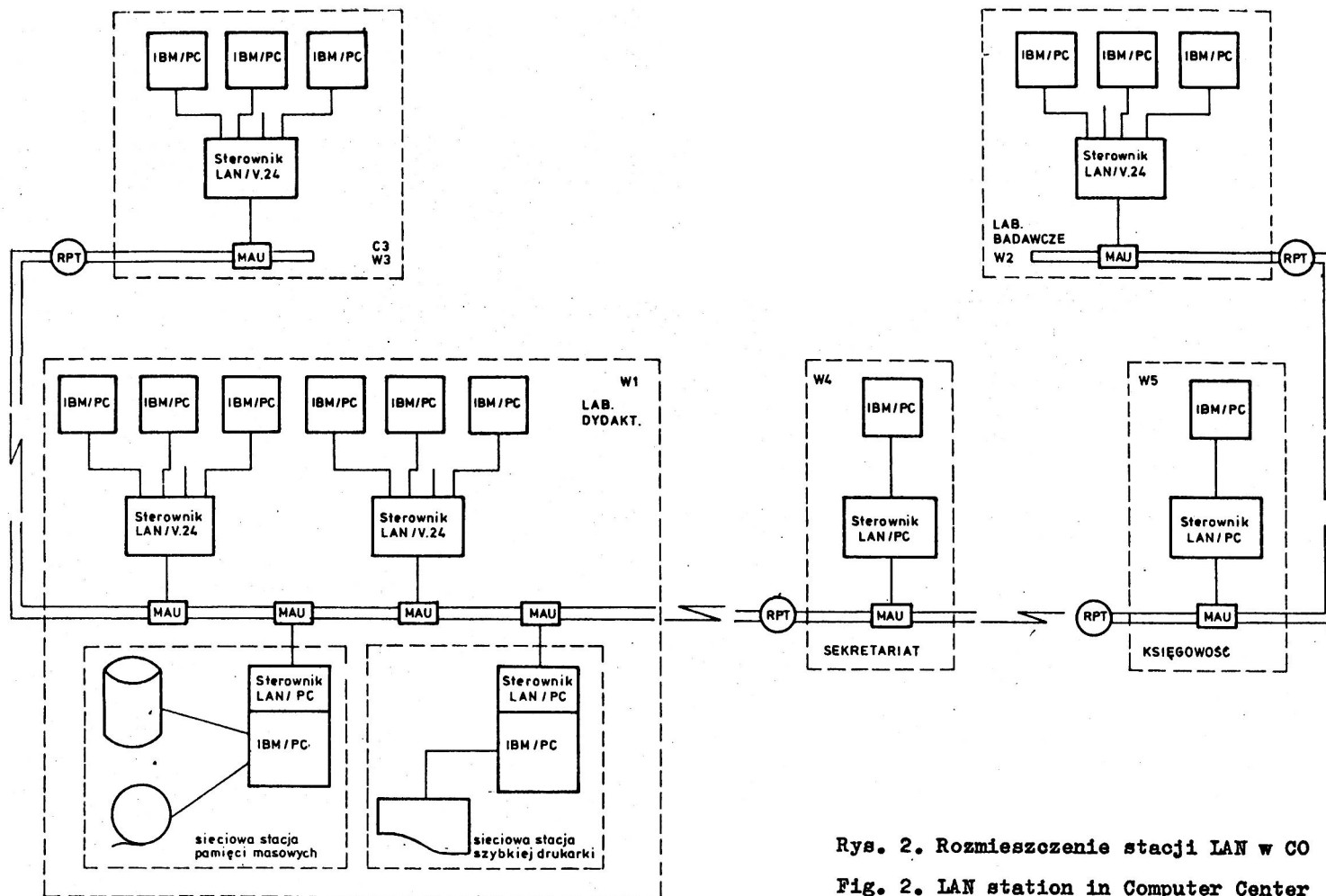
Usługi te powinny być dostępne z każdego terminala. Uwzględniając jednak specyfikację pracy wyróżnionych stacji można zauważyć dominację określonych potrzeb poszczególnych grup użytkowników. Potrzeby te wyrażają się w sposobie komunikacji między stacjami. I tak, analizując graf z rys. 1 pod kątem współpracy użytkowników, można wyodrębnić 4 formy komunikacji pomiędzy użytkownikami:

a) terminal-terminal, realizowana w większości jako transmisja typu punkt-punkt, rzadziej w trybie rozgłoszeniowym, występująca przede wszystkim w kontaktach między stacjami administracyjnymi a pozostałymi (np. $u_5 \rightarrow u_3$, $u_4 \rightarrow u_3$, $u_4 \rightarrow u_2$ itd),

b) terminal-pamięć masowa, umożliwiająca nieautoryzowany dostęp z terminali do zbiorów systemowych, usługowych (np. WORDSTAR) oraz ogólnodostępnych baz danych; ta forma komunikacji będzie stosowana zwłaszcza w kontaktach między $u_3 \rightarrow u_1$, $u_3 \rightarrow u_2$,

c) terminal-pamięć o autoryzowanym dostępie, uwidaczniająca się w dwustronnych kontaktach administracyjnych $u_4 \leftrightarrow u_5$,

d) pamięć-pamięć, realizowana przede wszystkim pomiędzy użytkownikami, dysponującymi dużymi pojemnościami pamięci, co w rozpatrywanym przypadku występuje w komunikacji między $u_1 \leftrightarrow u_2$.



Rys. 2. Rozmieszczenie stacji LAN w CO

Fig. 2. LAN station in Computer Center

Sprecyzowane formy komunikacji pomiędzy użytkownikami sugerują zastosowanie w stacji LAN podstawowych usług sieciowych takich, jak: transfer zbiorów (FT), wirtualny terminal (VT) oraz poczta elektroniczna.

Rozmieszczenie stacji LAN w CO wraz z planowanym wyposażeniem sieciowym, po uwzględnieniu wag przypisanych poszczególnym grupom użytkowników, pokazano na rys. 2, gdzie również uwidoczniono proponowany przydział zasobów do poszczególnych grup użytkowników.

Niezależnie od warunków, w jakich sieć jest projektowana, ostateczna decyzja o jej instalacji musi być poprzedzona rachunkiem ekonomicznym.

Oszacowanie kosztów wprowadzenia LAN do ośrodka nie posiadającego komputerów jest dość proste, ponieważ jest to koszt zakupu i instalacji sieci. Opłacalność inwestycji wynikać będzie nie tylko z szybszego przetwarzania danych, obiegu dokumentów czy przesyłania informacji w ośrodku, ale także z wygody i postępu technicznego.

Oszacowanie efektów ekonomicznych instalacji LAN w ośrodku już skomputeryzowanym jest trudniejsze, ponieważ powinno być wykonane porównawczo w stosunku do stanu istniejącego oraz powinno uwzględniać nakłady na komputeryzację w razie rozwoju pojedynczych stanowisk mikrokomputerowych w miejsce LAN. Niewątpliwie niewymierną korzyścią z instalacji LAN jest możliwość przetwarzania lokalnych zadań w różnych jednostkach systemu [5].

W przypadku instalacji sieci lokalnej w specjalistycznym ośrodku informatycznym, jakim jest Centrum Obliczeniowe, istotnym czynnikiem decydującym o zastosowaniu LAN jest profil dydaktyczny i badawczy ośrodka. Kształcenie studentów wymaga, oprócz przekazywania im wiedzy teoretycznej, właściwego zaplecza laboratoryjnego, opartego na najnowszych rozwiązaniach. Prowadzenie zajęć z zakresu sieci komputerowych, bez możliwości poparcia ich ćwiczeniami praktycznymi, stanowi półśrodek nie spełniający założeń nauczania.

LITERATURA

- [1] ARATO A., SERKODI-NOGY, SULYAN J., TELBISZ F., A Software Architecture of a Local Area Network, Computer Network Usage: recent experiences, North Holland 1986.
- [2] BUDKA M., Modele i usługi sieci lokalnych. Sieci komputerowe. Usługi, protokoły, modele. Część 1, Wyd. PWr., Wrocław 1985.
- [3] KLEINROCK L., Distributed Systems, Communication of the ACM, Vol.28, Nov. 1985.
- [4] PYYKKONEN M., Plan for your LAN, Datamation, October 1987.

- [5] WAH B. W., Jie-Yong Young Resource scheduling for local computer systems with a multi-access network, IEEE Transactions on Comp. Vol. C-64, No. 12, Dec. 1985.
- [6] YAKUBITIS E. A., Architecture of Computer Network for Large Offices Computer Network Usage: recent experiences, North Holland 1986.

DESIGN OF LAN ON AN EXAMPLE OF COMPUTER CENTER

In the paper there are presented two approaches to designing of LAN in an institution. The first one considers the design of LAN in case when the institution does not own any computer equipment, and the second, if the institution has had already some microcomputers. The design steps are illustrated on example of Computer Center of Wrocław Technical University.

ПРОЕКТИРОВАНИЕ СЕТИ LAN НА ОСНОВЕ ПРИМЕРА ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА ВРОЦЛАВСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

В работе представлены вопросы проектирования локальной компьютерной сети (LAN). Рассмотрены два исходных положения:

- институт не располагает вычислительной техникой,
- институт уже оснащен такой техникой.

Этапы проектирования иллюстрированы пользуясь примером Вычислительного Центра Вроцлавского политехнического института.

Praca wpłynęła do Redakcji w marcu 1989 r.
w ostatecznej formie w lipcu 1989 r.

*Sieć komputerowa,
program, projektowanie*

Andrzej STANISZ*

ELEGANCJA CZY EFEKTYWNOŚĆ – CZYLI JAK PROJEKTOWAĆ OPROGRAMOWANIE SIECIOWE?

W artykule rozważa się podstawowe problemy projektowania oprogramowania sieciowego opartego na modelu OSI/ISO: zasady dekompozycji oprogramowania, styki międzywarstwowe, współbieżność realizacji funkcji i wewnętrzną strukturę warstw. Opisuje się niektóre typowe rozwiązania, wskazując ich zalety i wady. Podaje się wiele wskazówek projektowych, prowadzących do rozwiązań racjonalnie godzących wymagania "elegancji" programu i efektywności jego działania.

1. WPROWADZENIE

Projektowanie programowania sieciowego jest problemem, przed którym staje coraz więcej programistów. Rozwiązanie w znacznym stopniu zależy od umiejętności sensownego pogodzenia wielu sprzecznych nieraz wymagań. Niejednokrotnie wybór rozwiązania jest ograniczony architekturą komputera, istniejącym systemem operacyjnym, językiem programowania, zaleceniami wynikającymi z ogólnej metodologii programowania, istniejącymi wzorcami, subiektywnymi ocenami projektanta i innymi czynnikami. Wybór najczęściej jest kompromisem między kryteriami realizacyjnymi, takimi jak poprawność konstrukcji programu, łatwość oddzielnego uruchamiania i testowania poszczególnych modułów, łatwość konserwacji i rozwoju itp., a kryterium efektywności działania programu.

* Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław.

Metodologia programowania nakazuje dekomponować program na moduły, a także zapewnić możliwie wąskie styki międzymodułowe i izolację danych lokalnych modułów, tak by po opracowaniu specyfikacji modułu można go było projektować, kodować i testować niezależnie od pozostałych. Względny efektywności często przemawiają za innymi, niezbyt "eleganckimi" rozwiązaniami, wolnymi od zbędnych narzutów i komplikacji. Pojawia się pytanie: elegancja czy efektywność, co wybrać?

Przez pojęcie elegancja programu rozumie się tu te cechy oprogramowania, które postuluje się w ogólnej metodologii programowania, jak np. struktura modułowa, dobrze określone funkcje modułów, izolowanie danych między modułami, brak skutków ubocznych, konstrukcja kodu oparta na programowaniu strukturalnym itp.

Program, spełniający te postulaty, jest z metodologicznego punktu widzenia poprawny, ale nie zawsze efektywny (w sensie szybkości przetwarzania i zajętości pamięci). I na odwrót, program, który nie spełnia tych postulatów, z metodologicznego punktu widzenia jest niepoprawny, np. z powodu niepoprawnej konstrukcji, ale może być bardziej efektywny.

Dokonując wyboru rozwiązania, programista musi zdecydować się na kompromis między metodologiczną poprawnością programu (elegancją) a jego efektywnością. Oczywiście kompromis ten musi być rozważany w ramach założenia, że program zawsze spełnia założone funkcje użytkowe, tzn. jest poprawny z użytkowego punktu widzenia.

Artykuł jest próbą przedstawienia najważniejszych problemów, które muszą być rozstrzygnięte na etapie projektu i implementacji oprogramowania sieciowego, opartego na warstwowym modelu OSI/ISO [1]. Przedstawia się niektóre typowe rozwiązania, wskazując ich zalety i wady.

Rozważania oparto na pewnym doświadczeniu autora uzyskanym podczas realizacji lub projektu oprogramowania komunikacyjnego dla sieci z komutacją pakietów - w ramach prac nad siecią MSK/KASK. Przedstawione wnioski odnoszą się jednak do pewnej klasy produktów programowych, budowanych według ogólnych zasad, modelu OSI/ISO, niezależnie od typu komputera i stosowanego systemu operacyjnego.

2. DEKOMPOZYCJA OPROGRAMOWANIA

Oprogramowanie komunikacyjne (sieciowe) jest zazwyczaj produktem programowym o znacznej objętości i dużym stopniu złożoności wewnętrznej. Wymaga to udziału i współpracy większej liczby programistów. Rutynowym postępowaniem w takim przypadku jest dekompozycja programu na moduły o dobrze zdefiniowanej funkcji i stosunkowo słabych powiązaniach wzajemnych.

Współczesne oprogramowanie komunikacyjne w większości przypadków oparte jest na modelu, w którym wyróżnia się strukturę warstwową. Model referencyjny OSI/ISO [1, 2, 4] jest uważany za obowiązujący standard, jakkolwiek istnieją inne modele, również oparte na koncepcji warstw. Z tego powodu naturalną jednostką modularyzacji jest warstwa.

Rys. 1. Ogólna struktura oprogramowania sieciowego: OS - system operacyjny, E - wewnętrzne środowisko programowe, A, P, S, T, N, DL, Ph - odpowiednio, warstwy: aplikacji, prezentacji, sesji, transportowa, sieciowa, liniowa i fizyczna

Fig. 1. General structure of network software: OS - operating system, E - internal program environment, A, P, S, T, N, DL, Ph - application, presentation, session, transport, network, data link and physical layer, respectively

OS	E	A
		P
		S
		T
		N
		DL
		Ph

W programie komunikacyjnym, zdekomponowanym na warstwy (rys. 1), powstaje w ten sposób kilka, względnie autonomicznych modułów, odpowiadających poszczególnym warstwom. Tak wyodrębnione moduły są nadal obiektami dość złożonymi, zwłaszcza ze względu na sposób inicjowania ich działania: uaktywnienie modułu następuje pod wpływem pojawienia się komunikatu (prymitywu) z sąsiednich warstw lub ingerencji modułu zarządzania warstwami, lub też wskutek zdarzeń generowanych w samej warstwie (upływ limitów czasowych). Wszystkie tego rodzaju sytuacje podaje się w specyfikacji warstwy.

Dekompozycja modułów realizujących poszczególne warstwy nie jest już taka oczywista i zależy m. in. od środowiska programowego, oferowanego przez system operacyjny (i ewentualnie własnego), przyjętej koncepcji komunikacji warstw, zasad współbieżnej realizacji poszczególnych funkcji programu itp. Zagadnienie to będzie jeszcze omawiane w p. 5.

3. WSPÓLBIEŻNOŚĆ

Funkcje oprogramowania komunikacyjnego muszą być realizowane współbieżnie (ściślej mówiąc, w środowisku jednoprosesorowym, quasi-współbieżnie). Tę współbieżność można osiągnąć w różny sposób:

a) przez opracowanie pojedynczego programu o działaniu sekwencyjnym, w którym poszczególne warstwy uaktywniane są bezpośrednio w przyjętej kolejności lub wskutek wzajemnych wywołań;

b) przez wykorzystanie standardowych środków organizujących pracę wieloprogramową (wielozadaniową), oferowanych przez system operacyjny;

c) przez zorganizowanie w ramach programu sekwencyjnego wewnętrznej współbieżności, opartej na pewnej koncepcji środowiska programowego, zorientowanego na implementację oprogramowania komunikacyjnego.

Rozwiązanie a jest pozornie proste, ale nie sprzyja modularnej budowie programu, a także utrudnia jego testowanie i modyfikację. Ze względu na złożoność programu trudne staje się panowanie nad całością, zwłaszcza w większym zespole programistów.

Rozwiązanie b oferuje atrakcyjny sposób dekompozycji zgodnie z zasadą: każda warstwa jest oddzielnym programem. Zakłada się przy tym, że system operacyjny oferuje pewne udogodnienia dla komunikacji i synchronizacji programów.

Bezkrytyczne stosowanie takiego rozwiązania prowadzi na ogół do znacznego zmniejszenia efektywności, wynikającego z niedopuszczalnie dużych narzutów wprowadzonych przez system operacyjny. Autorowi znany jest przypadek pewnego typu oprogramowania sieciowego, którego twórcy starali się oszacować jego przepustowość i - nie doceniając wspomnianych narzutów - uzyskali wynik przekraczający o rząd (sic!) wartość uzyskaną po bardziej wnikliwej analizie.

Pewną modyfikacją tego typu rozwiązania jest wyodrębnienie grupy warstw jako oddzielnego programu. Przykładem może tu być program realizujący dolne 4 warstwy modelu ISO, realizujący usługi transportowe dla kilku aplikacji, działających jako oddzielne programy w tym samym komputerze. Rozwiązanie tego typu było proponowane w pracy [5]. Problem dekompozycji wewnętrznej tego typu programu pozostaje przy tym nadal aktualny.

Rozwiązanie c pozwala pogodzić wymagania dekompozycji programu i efektywności jego działania. Wyodrębnienie wewnętrznego środowiska programowego jest celowe z kilku powodów:

a) zapewnia udogodnienia programowe związane ze specyfiką oprogramowania komunikacyjnego i potrzebne we wszystkich modułach programowych (np. zarządzanie pamięcią dynamiczną-gospodarka buforami);

b) dostarcza zunifikowane mechanizmy umożliwiające quasi-współbieżną realizację wielu akcji w różnych warstwach oprogramowania, a także standardowe środki komunikacji i synchronizacji tych akcji;

c) stwarza szanse przenośności oprogramowania przez zlokalizowanie bezpośrednich kontaktów z systemem operacyjnym w modułach środowiska; dzięki temu przeważająca część programu nie zależy wprost od zastosowanego systemu operacyjnego, co w połączeniu z odpowiednim językiem programowania czyni ją przenośną na każdy komputer, w którym będzie zaimplementowane identyczne środowisko wewnętrzne.

Wewnętrzne środowisko programowe powinno być wystarczająco uniwersalne i elastyczne, tak by dostarczało programiście istotnych udogodnień do organizacji wewnętrznej programu i sprzyjało poprawnej konstrukcji programu (ułatwienie dekompozycji oprogramowania, modularyzacja, możliwość niezależnego kodowania i testowania modułów). Z drugiej strony środowisko to powinno być stosunkowo proste i łatwe do opanowania przez programistów, a także efektywne, tak by nie wprowadzało nadmiernych narzutów podczas działania programu. Propozycję wewnętrznego środowiska programowego można znaleźć w pracy [5]. Jego rozwiniętą i udoskonaloną wersję przyjęto później w projekcie podobnego produktu programowego [6], [7].

4. STYKI MIĘDZYWARSTWOWE

Funkcjonowanie oprogramowania komunikacyjnego jako całości wiąże się z intensywną komunikacją międzywarstwową. Bardzo często jednostki danych muszą być odpowiednio przetworzone we wszystkich warstwach. Wymaga to sprawnych i efektywnych mechanizmów komunikacji międzywarstwowej. Niewłaściwe rozwiązanie w tym zakresie może doprowadzić do nadmiernych narzutów w postaci zbędnych operacji kopiowania, dodatkowych czynności wynikających z przyjętych zasad komunikacji itp. Dodatkową trudnością jest konieczność pewnego kompromisu między efektywnością komunikacji a konsekwentnym przestrzeganiem zasad modularyzacji.

4.1. Usługi warstwy i prymitywy

Podstawą do projektowania styków międzywarstwowych jest definicja usług warstwy, określona w odpowiednich dokumentach standaryzacyjnych, np. [2]. Intencją tych dokumentów jest określenie ogólnych zasad współpracy warstw w sposób dostatecznie ogólny i ścisły, ale jednocześnie tak, by nie ograniczać możliwości jego implementacji. Takie podejście co prawda pozostawia projektantowi spory margines swobody, ale w wielu kwestiach szczegółowych nie daje konkretnych propozycji rozwiązań.

Usługi warstwy określa się w kategoriach tzw. prymitywów. Prymityw jest jednostkową, niepodzielną interakcją między sąsiednimi warstwami, której na ogół towarzyszy przekazanie pewnych parametrów i danych. Prymityw jest w zasadzie oddziaływaniem jednokierunkowym: warstwa generująca prymityw kieruje go do warstwy sąsiedniej, przekazując pewne dane w tym samym kierunku. Ewentualna odpowiedź warstwy odbierającej prymityw nie może być przekazana bezpośrednio, ale musi pojawić się w oddzielnym prymitywie, przekazywanym w odwrotnym kierunku. Dokumenty standaryzacyjne definiują jedynie prymitywy, mające znaczenie dla komunikacji między odległymi systemami. Projektantom konkretnych systemów pozos-

tawia się możliwość implementacji dowolnych mechanizmów o znaczeniu lokalnym, w tym także możliwość wprowadzenia prymitywów lokalnych.

W niektórych dokumentach (np. [3]) prymityw międzywarstwowy określano się jako wzajemną, dwukierunkową interakcję, złożoną z dwóch faz: akcji i reakcji. Warstwa inicjująca prymityw realizuje akcję, przysyłając pewne dane do warstwy sąsiedniej. Warstwa odbierająca prymityw przyjmuje te dane i generuje reakcję, w której podaje informację o przyjęciu lub odrzuceniu prymitywu, a także być może przekazuje pewne dane. Każdej akcji musi towarzyszyć reakcja. Jakakolwiek dalsza akcja może nastąpić jedynie wtedy, gdy nastąpi odpowiednia reakcja na poprzednią akcję. Ciąg akcja-reakcja jest niepodzielny, jest więc prymitywem.

W tej koncepcji zasadniczym elementem prymitywu jest akcja, gdyż w niej zawarta jest istotna treść związana z komunikacją między odległymi systemami. Reakcja ma jedynie znaczenie lokalne, związane ze współpracą warstw w jednym systemie.

4.2. Implementacja prymitywów

Implementacja styku międzywarstwowego powinna pogodzić dwa sprzeczne wymagania:

- efektywną wymianę danych przekazywanych w prymitywie,
- maksymalną izolację danych poszczególnych warstw.

Dane przenoszone w prymitywach mogą być przekazywane między warstwami w różny sposób:

- a) przez komunikaty międzywarstwowe,
- b) przez obszary wspólne dla obu warstw,
- c) w parametrach podprogramów.

Sposób a, wychodzący z koncepcji komunikacji programów współbieżnych w środowisku rozproszonym, polega na umieszczeniu wszystkich danych związanych z prymitywem w buforze (zwykle wydzielonym w obszarze pamięci dynamicznej) i przekazaniu adresu bufora do drugiej warstwy. Generacja prymitywu i jego przetwarzanie odbywają się niezależnie. Warstwa generująca prymityw nie oczekuje na odpowiedź; ewentualna reakcja lokalna warstwy odbierającej musi być przekazana w przeciwnym kierunku, w oddzielnym prymitywie.

Zalety takiego rozwiązania to: prostota koncepcyjna mechanizmu komunikacji warstw, dobra ochrona danych między warstwami (warstwa odbierająca prymityw operuje na kopii danych w buforze i nie przekazuje żadnych informacji zwrotnych), łatwość testowania (warstwę można w znacznym stopniu przetestować przez sprawdzanie reakcji na różne komunikaty przygotowane w buforach).

To "eleganckie" rozwiązanie ma jednak i wady wynikające przede wszystkim z jednokierunkowej koncepcji prymitywu, m. in.: konieczność

kopiowania części parametrów prymitywu do i z bufora (np. adresów sieciowych - warstwa generująca prymityw kopiuje adres ze swoich struktur danych do bufora, skąd na ogół zostanie on ponownie skopiowany przez warstwę odbierającą prymityw do jej struktur danych), utrudnienia w identyfikacji połączeń jednoznacznej dla obu warstw, bardziej skomplikowana kontrola poprawności formatu i dopuszczalności sekwencji prymitywów, konieczność wprowadzania dodatkowych prymitywów lokalnych w celu koordynacji współpracy warstw itp.

Sposób b umożliwia dwukierunkowe przekazywanie danych (parametry prymitywu wpisywane są do wspólnego obszaru komunikacyjnego, gdzie również umieszczane są ewentualne informacje zwrotne). Wadą tego rozwiązania jest konieczność zapewnienia synchronizacji dostępu obu warstw do bufora i brak możliwości tworzenia kolejek prymitywów.

Cechą charakterystyczną obu tych sposobów było koncepcyjne rozdzielanie fazy generowania prymitywu i fazy jego przetwarzania. Takie spojrzenie wynika na ogół z chęci izolowania działań poszczególnych warstw, mającego na celu ułatwienie ich testowania (w czasie przetwarzania w jednej warstwie należy jedynie zainicjować niezbędne działania w innych; działania te zostaną wykonane później, gdy sterowanie będzie przekazane kolejno tym warstwom, zgodnie z wewnętrznym algorytmem współbieżności).

Sposób c, oparty na naturalnej i narzucającej się implementacji prymitywu jako podprogramu, zmierza w kierunku wspomnianej już koncepcji prymitywu, widzianego jak wzajemna, dwukierunkowa interakcja warstw. Zakłada się, że w czasie tej interakcji na styku warstw występuje (jako niepodzielna całość) następująca sekwencja czynności:

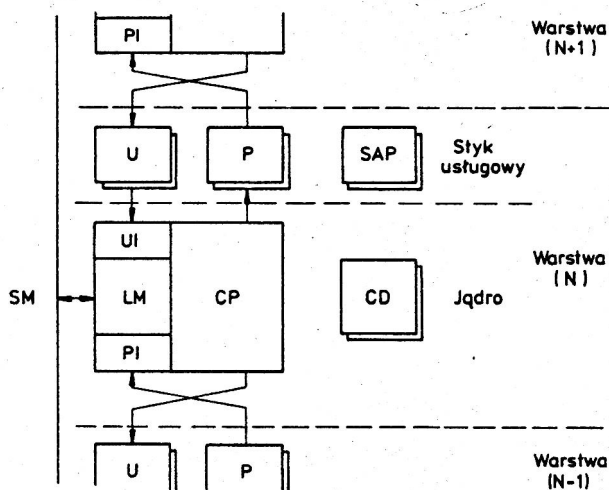
- kontrola dopuszczalności prymitywu (ze względu na kontekst i parametry)
- przekazanie danych z warstwy generującej prymityw do warstwy odbierającej (z ewentualnym skopiowaniem danych do odpowiednich struktur)
- zainicjowanie dalszego przetwarzania w warstwie odbierającej (lub całkowita realizacja przetwarzania)
- przekazanie informacji zwrotnej do warstwy generującej prymityw (np. kodu informującego o przyjęciu lub odrzuceniu prymitywu, identyfikatora zakładanego połączenia, ustalanego przez warstwę dostarczającą usług, informacji sterującej przepływem danych na styku międzywarstwowym itp.).

Najistotniejsze zalety tego rozwiązania to: prostota koncepcyjna (niepodzielność interakcji jest zapewniona automatycznie, odpadają problemy synchronizacji prymitywu i odpowiedzi lokalnej), wyeliminowanie zbędnych narzutów na komunikację i synchronizację warstw, a także pośredniego kopiowania niektórych danych.

Za efektywność płaci się jednak utratą "elegancji": dla dwukierunkowego przepływu danych należy dopuścić wzajemną ingerencję warstw w struktury danych. W podprogramach mogą wystąpić zarówno parametry wejściowe (dla wskazania zasadniczych danych przekazywanych w prymitywie - faza akcji), jak i parametry wyjściowe (dla wskazania miejsca, gdzie mają być umieszczone informacje zwrotne - faza reakcji). Nieeleganckie jest i to, że podczas realizacji prymitywu (podprogramu) wykonują się jednak i pewne funkcje warstwy sąsiedniej. Stanowi to pewne utrudnienie dla testowania: już nie można testować izolowanej warstwy; potrzebne są owe podprogramy realizacji prymitywów lub ich namiastki (symulatory).

5. STRUKTURA WARSTWY

Opierając się na dotychczasowych rozważaniach można dokonać próby określenia ogólnego modelu implementacyjnego warstwy. W celu ułatwienia porównań z modelem OSI/ISO, w poniższym opisie stosowane będą konwencje terminologiczne, zaczerpnięte z pracy [1].



Rys. 2. Uogólniony model implementacyjny warstwy: U, P - podprogramy przekazywania prymitywów, SAP - dane punktu dostępu, UI, PI - obsługa prymitywów, CP - procedury obsługi połączeń, CD - dane połączeń, LM - zarządzanie warstwą, SM - zarządzanie systemem

Fig. 2. Generalized implementation model of a layer: U, P - user/provider primitives subroutines, SAP - service access point data, UI, PI - user/provider primitives interfaces, CP - connection processing, CD - connection data, LM - layer management, SM - system management

W module odpowiadającym warstwie (rys. 2) wyróżnia się dwa submoduły (moduły lokalne):

- styk usługowy, (N)-interface
- jądro warstwy, (N)-protocol-machine.

5.1. Styk usługowy

Warstwa (N) udostępnia usługi warstwie (N+1) wyłącznie przez swój styk usługowy i korzysta z usług warstwy (N-1) przez styk usługowy tejże warstwy. Jądro warstwy (N) jest dostępne dla sąsiednich warstw wyłącznie przez podprogramy, należące do styków usługowych.

Podprogramy styku usługowego, odpowiadające prymitywom, zdefiniowanym dla styku warstw (N) i (N+1), dzielą się na dwie grupy:

a) podprogramy do realizacji prymitywów generowanych w warstwie (N+1), tzn. typu (N)-xxx-request i (N)-xxx-response (rys. 2, submoduł U),

b) podprogramy do realizacji prymitywów generowanych w warstwie (N), tzn. typu (N)-xxx-indication i (N)-xxx-confirm (rys. 2, submoduł P).

Wywołania podprogramów grupy a powodują przekazanie sterowania oraz odpowiednich danych do submodułu UI (rys. 2), wchodzącego w skład jądra warstwy (N). Wywołanie podprogramu grupy b przekazuje sterowanie i odpowiednie dane do submodułu PI warstwy (N+1). Analogiczne zasady stosuje się odpowiednio na styku warstwy (N-1) i (N).

Podprogramy styku usługowego stanowią udogodnienia dla obu warstw, przesłaniające szczegóły ich komunikacji. Szczegóły te muszą jednak być znane w submodułach UI i PI. Rozwiązanie takie umożliwia skupienie obsługi wszystkich prymitywów w jednym punkcie, tj. odpowiednio w modułach UI i PI (jakkolwiek nie jest to konieczne).

W skład styku usługowego wchodzi ponadto struktury danych, odpowiadające punktom dostępu do usług, (N)-service-access-points (rys. 2, submoduł SAP). Struktury te zawierają następujące dane:

- adres punktu dostępu do usług, (N)-address
- wskaźnik przyłączonej jednostki w warstwie dostarczającej usług, (N)-entity
- wskaźnik przyłączonej jednostki w warstwie korzystającej z usług, (N+1)-entity
- maksymalną i aktualną liczbę połączeń w punkcie dostępu, (N)-connections
- tablicę identyfikatorów usługowych połączeń, (N)-service-connection-identifiers do powiązania punktu dostępu do usług ze strukturami w jądrze warstwy, dotyczącymi istniejących w tej warstwie połączeń.

5.2. Jądro warstwy

W skład jądra warstwy wchodzi następujące submoduły (rys. 2):

- zarządzanie warstwą (submoduł LM),
- obsługa prymitywów (submoduły UI i PI),
- procedury obsługi połączeń (submoduł CP).

Ponadto do jądra warstwy należą struktury danych opisujące połączenia (submoduł CD).

Wewnętrzna struktura warstwy w znacznym stopniu zależy od środowiska programowego, w którym ma ona funkcjonować (m. in. od systemu operacyjnego, przyjętej koncepcji współbieżności itp.), a także od pomysłu projektanta. Na ogół jądro warstwy tworzy zbiór procedur lub procesów, powiązanych ze sobą wzajemnymi wywołaniami i operujących na wspólnych danych. Podstawą wyróżnienia procedur (procesów) może być:

a) wyodrębnienie poszczególnych funkcji warstwy niezależnie od tego, jakiego połączenia dotyczy,

b) wyodrębnienie wszystkich akcji, dotyczących pojedynczego połączenia, widzianych jako realizacja procesu.

Dekompozycja warstwy na procedury (procesy) jest ściśle związana z usługami oferowanymi przez środowisko. I na odwrót, środowisko programowe powinno być tak zaprojektowane, by ułatwić dekompozycję warstw. Te decyzje projektanta determinują funkcjonowanie warstwy. Nie wchodząc w szczegóły związane z konkretnym środowiskiem programowym, można jedynie określić ogólne zasady funkcjonowania warstw.

5.3. Funkcjonowanie warstwy

Uaktywnienie warstwy następuje pod wpływem jednego z następujących zdarzeń:

- a) pojawienie się prymitywu z warstwy (N+1),
- b) pojawienie się prymitywu z warstwy (N-1),
- c) upływ limitu czasowego, wyznaczonego we wcześniejszych działaniach warstwy,
- d) akcja ze strony modułu zarządzania systemem (rys. 2, moduł SM).

W zależności od usług oferowanych przez wewnętrzne środowisko programowe możliwe są następujące działania w warstwie:

a) przyjęcie uaktywnienia bez jakiegokolwiek przetwarzania i zainicjowanie dalszego przetwarzania przez zainicjowanie odpowiedniej procedury (lub procesu) w warstwie; procedura ta będzie realizowana zgodnie z zasadami szeregowania, obowiązującymi w danym środowisku programowym;

b) wstępne przetworzenie zdarzenia (np. kontrola poprawności, dopuszczalności w aktualnym stanie warstwy, przeniesienie danych do własnych struktur warstwy itp.) i zainicjowanie procedury (procesu) dalszego przetwarzania przez usługi środowiska;

c) pełne przetworzenie zdarzenia łącznie z wygenerowaniem kolejnych prymitywów, uaktywniających dalsze warstwy.

Sposób a charakterystyczny jest dla jednokierunkowej koncepcji prymitywu i wymiany danych przez komunikaty międzywarstwowe. Sposoby b i c odpowiadają dwukierunkowej koncepcji prymitywów i ich implementacji jako podprogramów, przy czym sposób b zakłada istnienie elementarnych usług kolejowania procedur, a sposób c jest zorientowany na czysto sekwencyjną strukturę programu sieciowego.

6. WNIOSKI

Przedstawione rozważania prowadzą do następujących wniosków:

1. Projekt oprogramowania sieciowego dowolnego typu powinien być oparty na racjonalnym kompromisie między sprzecznymi na ogół wymaganiami elegancji (tj. metodologicznej poprawności konstrukcji programu) i efektywności jego działania (ocenianej szybkością przetwarzania i zajętością pamięci).

2. Program sieciowy powinien być zdekomponowany na moduły odpowiadające warstwom, z wyodrębnionymi submodułami, realizującymi styk międzywarstwowy zgodnie z przyjętą definicją usług warstwy. W oprogramowaniu realizowanym zgodnie z filozofią modelu OSI submoduły te powinny zawierać:

a) struktury danych reprezentujące punkty dostępu do usług warstwy,

b) podprogramy, realizujące przekazywanie prymitywów usługowych między sąsiednimi warstwami, zdefiniowanych dla danej warstwy w odpowiednich dokumentach ISO z rozszerzeniami o funkcje o znaczeniu lokalnym.

3) Implementacja prymitywów usługowych powinna umożliwiać dwukierunkowe przekazywanie danych, tak by warstwa odbierająca prymityw mogła bezpośrednio przekazać informację zwrotną o znaczeniu lokalnym (nie zdefiniowaną w modelu ISO).

4) Sterowanie współbieżną realizacją funkcji poszczególnych warstw powinno bazować na prostym środowisku programowym, zdefiniowanym wewnątrz pojedynczego programu (w zasadzie sekwencyjnego). Ze względu na intensywność interakcji międzywarstwowych, w celu uniknięcia nadmier-nych narzutów na przełączanie i komunikację jednostek programowych, na-

leży unikać rozwiązań opartych na standardowych środkach wieloprogramowości (wielozadaniowości), oferowanych przez systemy operacyjne.

5. Wewnętrzne środowisko programowe powinno dostarczać mechanizmów, ułatwiających wewnętrzną komunikację i synchronizację różnych akcji programu. Powinno ono być zarazem tak pomyślane, by oferowane udogodnienia nie wprowadzały nadmiernych narzutów programowych.

W artykule położono nacisk na styki międzywarstwowe i ogólną strukturę warstw. Nie rozwijano - z braku miejsca - problematyki wewnętrznego środowiska programowego. Większość rozważań odnosi się do oprogramowania komunikacyjnego implementowanego w środowisku jednoprosesorowym. Przedstawione oceny i wnioski powinny być interesujące dla projektantów tego typu oprogramowania, mimo że wiele stwierdzeń zawartych w artykule odzwierciedla subiektywne poglądy autora i ma - być może - charakter dyskusyjny.

LITERATURA

- [1] Information Processing Systems - Open System Interconnection - Basic Reference Model, ISO Standard IS 7498, 1984.
- [2] CCITT Recommendations X.200-X.250, Red Book, Vol. VIII, Fascicle VIII.5, Geneva 1985.
- [3] Network Independent Interface, CEPT, SWG CD/IP, Wien 1985.
- [4] Protokoły w międzyuczelnianej sieci komputerowej (MSK) (pod red. E. Bilskiego), Biblioteka Informatyki Szkół Wyższych, Wyd. PWr., Wrocław 1987.
- [5] JANCZEWSKI K., ŁANOWSKA B., STANISZ A., Oprogramowanie minikomputera SM-4 jako minihosta sieci MSK. Projekt techniczny CO PWr., Raport SPR 24/85, Wrocław 1985.
- [6] Mikrokomputer kompatybilny z IBM PC jako mikrohost w sieci KASK. Projekt techniczny. PGM-15/02/88 (pod red. A. Stanisza), CO PWr., Raport SPR nr 12/88, Wrocław 1985.
- [7] JANCZEWSKI K., STANISZ A., Mikrokomputery klasy IBM PC/XT/AT jako mikrohosty w sieci KASK (w niniejszym zeszycie).

ELEGANCE VS. EFFICIENCY.

HOW TO DESIGN THE NETWORK SOFTWARE?

In the paper the main problems of designing of network software based on OSI/ISO Reference Model are discussed; principles of software decomposition, layer interfaces, concurrent function realization and in-

ternal layer structure. Some typical solutions are described and their advantages and disadvantages are pointed out. A number of design directives is given, which lead to solutions being a rational compromise between requirements for both program "elegance" and efficiency of its operation.

ЭЛЕГАНТНОСТЬ ИЛИ ЭФФЕКТИВНОСТЬ? КАК ПРОЕКТИРОВАТЬ СЕТЕВЫЕ ПРОГРАММЫ?

В статье рассматриваются главные проблемы проектирования сетевых программ, основанных на модели OSI/ISO: принципы декомпозиции программ, межуровневые сопряжения, параллельная реализация функций и внутренняя структура уровней. Описываются некоторые типические решения и указываются их преимущества и недостатки. Формулируется ряд проектных рекомендаций, ведущих к решениям, которые разумно удовлетворяют требованиям эlegantности программы и эффективности ее работы.

Praca wpłynęła do Redakcji w marcu 1988 r.
w ostatecznej formie w lipcu 1989 r.

SPIS TREŚCI

1. Przedmowa	3
2. K. Janczewski, E. Kosmulska-Bochenek, Funkcje mikrokomputerów w sieci KASK	5
3. K. Janczewski, A. Stanisław, Mikrokomputery klasy IBM PC/XT/AT jako mikrohosty w sieci KASK	13
4. I. Dubielewicz, Z. Fryźlewicz, K. Kolesnik, W. Komorowski, L. Misiura, Zastosowanie aparatury pomiarowo-diagnostycznej w sieci KASK	29
5. I. Dubielewicz, K. Kolesnik, System testowania zgodności implementacji protokołów w sieci KASK	39
6. E. Kosmulska-Bochenek, E. Hudyma, Projektowanie sieci LAN na przykładzie Centrum Obliczeniowego	51
7. A. Stanisław, Elegancja czy efektywność - czyli jak projektować oprogramowanie sieciowe?	59

CONTENTS

1. Preface	3
2. K. Janczewski, E. Kosmulska-Bochenek, Functions of microcomputers in computer communication network KASK	5
3. K. Janczewski, A. Stanisław, IBM PC/XT/AT compatible microcomputers as microhosts in the KASK computer network	13
4. I. Dubielewicz, Z. Fryźlewicz, K. Kolesnik, W. Komorowski, L. Misiura, Application of the SPS data analyzer and tester in the KASK network	29
5. I. Dubielewicz, K. Kolesnik, Conformance testing system of protocol implementation in the KASK network	39
6. E. Kosmulska-Bochenek, E. Hudyma, Design of LAN on an example of Computer Center	51
7. A. Stanisław, Elegance vs. efficiency - How to design the network software?	59



PRACE NAUKOWE CENTRUM OBLICZENIOWEGO
(wydane w latach 1985—1989)

Nr 3, Konferencje nr 1, <i>Sieci komputerowe. Usługi, protokoły, modele.</i> Część 1, Wrocław 1985	160,—
Nr 4, Konferencje nr 2, <i>Sieci komputerowe. Usługi, protokoły, modele.</i> Część 2, Wrocław 1986	250,—
Nr 5, Konferencje nr 3, <i>Sieci komputerowe. Teoria, technika, zastosowania,</i> Wrocław 1987	335,—
Nr 6, Studia i Materiały nr 3, <i>Sieci komputerowe Jednolitego Systemu,</i> Wrocław 1989	440,—
Nr 8, Monografie nr 1, Z. Huzar, <i>Programowanie procesów komuniku-</i> <i>jących się w czasie rzeczywistym,</i> Wrocław 1989	300,—
Nr 9, Konferencje nr 4, <i>Sieci komputerowe. Projektowanie, zastosowa-</i> <i>nia, eksploatacja,</i> Wrocław 1989	1000,—

Cena zł 300,—

Subscription should be sent (at any time of the year) to:

„Ars Polona”

Krakowskie Przedmieście 7, 00-068 Warszawa

Bank account number: PBK XIII Oddz. W-wa 370044-1195-139-11

Zamówienia na prenumeratę można składać:

OR PAN, PKiN, 00-901 Warszawa

Nr konta bankowego: PBK IX Oddz. W-wa, 370031-4792

Wydawnictwa Politechniki Wrocławskiej

ma stałe na składzie Księgarnia Wr 49

Wybrzeże Wyspiańskiego 27, 50-370 Wrocław

oraz Wojewódzka Księgarnia Techniczna

ul. Świdnicka 8, 50-067 Wrocław

ISSN 0860-1623