

Prace Naukowe Centrum Obliczeniowego
Politechniki Wrocławskiej
Seria: Studia i Materiały

6

3

Mag

Sieci komputerowe Jednolitego Systemu

A4947 I/1

Wrocław 1989



PRACE NAUKOWE POLITECHNIKI WROCŁAWSKIEJ

Scientific Papers of the Computer Centre
of the Technical University of Wrocław

No. 6

No. 6

Studies and Research

No. 3

1989

The SKJS2 Computer Networks

Prace Naukowe Centrum Obliczeniowego
Politechniki Wrocławskiej

6

Seria:
Studia i Materiały

3



Sieci komputerowe Jednolitego Systemu



Wydawnictwo Politechniki Wrocławskiej · Wrocław 1989

Recenzenci
Andrzej BABORSKI
Tadeusz BATYCKI

Redaktor naukowy
Witold KOMOROWSKI

Opracowanie redakcyjne
Maria KOPEC

Korekta
Janina GOWIN

© 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. 7,25. Ark. druk. 6 $\frac{1}{4}$. Papier offset. kl. IV, 70 g, B1.
Oddano do druku w sierpniu 1989 r. Druk ukończono w październiku 1989 r.
Zakład Graficzny Politechniki Wrocławskiej. Zam. nr 1199/89. Cena zł 440,-

PRZEDMOWA

Prace nad sieciami komputerowymi w Polsce, aczkolwiek z wieloletnim opóźnieniem, zostały podjęte w wielu ośrodkach uczelnianych i przemysłowych, między innymi w ramach problemów resortowych Ministerstwa Nauki i Szkolnictwa Wyższego, Ministerstwa Hutnictwa i Przemysłu Maszynowego oraz Ministerstwa łączności.

Prace nad budową rozległych sieci komputerowych, zapoczątkowane w roku 1978 w Politechnice Wrocławskiej, przyniosły liczne konkretne rozwiązania dotyczące sprzętu i programu. Eksperymentalnie eksploatowana Międzyuczelniana Sieć Komputerowa MSK połączyła w 1984 roku trzy ośrodki naukowe. Jest to sieć heterogeniczna, łącząca komputery Odra i Jednolitego Systemu w jedną całość poprzez podsieć komutacji pakietów.

W 1986 roku Politechnika Wrocławska we współpracy z Instytutem Komputerowych Systemów Automatyki i Pomiarów, opracowała bazowe oprogramowanie sieci komputerowej maszyn Jednolitego Systemu SKJS2 wersja 1. Jest to sieć homogeniczna.

Cechą charakterystyczną procesu budowy i późniejszej eksploatacji sieci komputerowej staje się duża złożoność pojawiających się problemów. Zakres rozwiązywanych zagadnień nie ogranicza się wyłącznie do spraw technicznych, lecz obejmuje także problemy organizacyjne, prawne i ekonomiczne.

W oddawanym do rąk Czytelnika niniejszym zeszycie przedstawiono wybrane zagadnienia tworzenia bazowego oprogramowania sieci SKJS2 wersja 1.

Na niniejszy zeszyt składa się 8 artykułów. W pierwszej pracy przedstawiono w zarysie strukturę sieci oraz wskazano zagadnienia omówione w kolejnych pracach. L. Budzianowski, J. Wietrzych i A. Frys omówili warstwową strukturę sieci. Szczegółowo zaprezentowali algorytm wyboru drogi. A. Kaliś i A. Huzar przedstawili problemy tworzenia oprogramowania sieciowego. M. Jacukowicz, A. Janiszewski i J. Kwiatkowski poświęcili swoją pracę dostępowi terminalowemu w sieci. D. Żak i J. Stańko omówili styki występujące w sieci. A. Huzar i A. Kaliś

zaprezentowali model synchronizacji procesów wykorzystywany w sieci, na przykładzie stacji transferu zbiorów.

Uruchomienie sieci i jej testowanie to najważniejsze etapy tworzenia oprogramowania bazowego. Narzędziem je wspomagającym jest aparatura pomiarowo-diagnostyczna, którą przedstawił Z. Fryźlewicz. Kończącą pracę K. Dyrki, E. Rutkowskiego i P. Kremienowskiego poświęcono badaniom i pomiarom wybranych parametrów sieci.

Autorzy prac mają nadzieję, że zaprezentowany materiał przybliży Czytelnikom zagadnienia związane z tworzeniem rozległej sieci komputerowej, której przykładem jest sieć SKJS2 wersja 1.

Andrzej KALIŚ

*Rozległa sieć komputerowa,
struktura, sprzęt,
oprogramowanie*

Andrzej HUZAR*

Andrzej KALIS*

SIEĆ KOMPUTEROWA JEDNOLITEGO SYSTEMU SKJS2 WERSJA 1

Praca zawiera podstawowe informacje o sieci komputerowej maszyn cyfrowych Jednolitego Systemu o nazwie SKJS2 wersja 1. Przedstawiono konfigurację sprzętową i architekturę logiczną sieci. Scharakteryzowano pracę podstawowych aplikacji sieciowych: podsystemów TSO i SKOT oraz stacji transferu zbiorów

1. WSTĘP

Stopień rozpowszechnienia sieci komputerowych na świecie jest tak znaczący, że w krajach wysoko rozwiniętych stały się one nieodłącznym elementem środowiska pracy badawczej i naukowej. Choć sieci są nadal rozwijane i rośnie liczba ich użytkowników, to jednak liczba sieci funkcjonujących w pełni użytkowym trybie jest ciągle dosyć ograniczona [5]. O użytkowaniu sieci rozstrzygają koszty projektowania i budowy, a zwłaszcza eksploatacji. Sieci wciąż nie stanowią jednorodnego, ostatecznie uformowanego obiektu, a raczej prze-

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

ciwnie: ich rozwój ujawnia wciąż nowe problemy i różnorodne propozycje ich rozwiązań. Wynika to przede wszystkim stąd, że gdy pokonano problemy techniczne umożliwiające budowanie sieci o praktycznie nieograniczonej wielkości, wówczas pojawiły się problemy związane z brakiem standardów mówiących o ich architekturze logicznej. Na przykład w Europie stosuje się około 10 standardów, w USA - 3, a w Japonii tylko 2 [9]. Dopełnieniem sytuacji jest fakt, że prace dotyczące określenia standardu prowadzone przez ISO (ang. International Standard Organization) posuwają się znacznie wolniej niż to przewidywano jeszcze 2-3 lata temu.

Świadomość, że sieci komputerowe będą w przyszłości stawać się elementem światowego obiegu i przetwarzania informacji, zrodziła potrzebę nowego spojrzenia na różne systemy łączności. Stąd wyłoniła się koncepcja ISDN - Zintegrowanego Systemu Usług Komunikacyjnych - i oparcie na takich systemach wszystkich nowo projektowanych sieci komputerowych (wielomiliardowe nakłady finansowe w USA).

Zasadnicze usługi, które są wykorzystywane w sieciach to transfer zbiorów danych oraz różne formy poczty elektronicznej i dostępu terminalowego. Przykładowo, Europejska Agencja Przestrzeni Kosmicznej udostępnia przez sieci komputerowe ponad 80 banków danych, wśród nich znane krajowym użytkownikom bazy takie, jak INSPEC, JSMEC, PASCAL. Statystyki wskazują, że przeciętnie w ciągu godziny odwołuje się do nich około 7000 użytkowników, w tym około 35% stanowią zapytania w trybie konwersacyjnym.

W trakcie swego rozwoju wykształciły się wyraźnie trzy typy sieci komputerowych: lokalne (instytucjonalne), krajowe i regionalne (rozległe) oraz międzynarodowe. Z każdym z tych typów wiąże się różne problemy techniczne, standaryzacyjne, organizacyjne i prawne. Pierwsze bazują na prostej podsieci transmisji danych, pozostając całkowicie w gestii użytkownika, natomiast pozostałe na podsieci transmisji danych, stanowiących wydzielone systemy zarządzane przez poczty.

Niniejszy artykuł dotyczy sieci komputerowej Jednolitego Systemu, nazwanej SKJS2 wersja 1. Jest to sieć pracująca wyłącznie na sprzęcie produkowanym lub kompletowanym przez Zakłady Elektroniczne ELWRO we Wrocławiu. Najlepiej nadaje się ona do pracy w rozproszonych terytorialnie przedsiębiorstwach posiadających kilka komputerów Jednolitego Systemu.

2. SIEĆ KOMPUTEROWA SKJS2 WERSJA 1

W roku 1984 Centrum Obliczeniowe Politechniki Wrocławskiej rozpoczęło realizację umowy na dostarczenie oprogramowania bazowego sieci komputerowej maszyn Jednolitego Systemu. Zespoły wykonawcze podjęły się tego złożonego zadania mimo trwania końcowych prac przy uruchamianiu sieci MSK (Międzyuczelniana Sieć Komputerowa). O ile prace nad siecią MSK w znacznym stopniu miały charakter badawczy i eksperymentalny oraz w swym zamierzeniu dopiero przygotowywały kadry specjalistów, o tyle sieć SKJS2 jest produktem handlowym. Sieć SKJS2 była już gotowa w końcu 1986 roku, czyli jej realizacja trwała około 3 lat, wliczając w ten okres koncepcję, projekt, kodowanie i testowanie oprogramowania sieciowego. Oprogramowanie to jest oferowane przez ELWRO-SERVICE, a wyniki prac opisano w raportach [1,2,3,4,7,8] i artykułach zamieszczonych w niniejszym zeszycie [10,11,12,13,14,15] .

2.1. Geneza projektu

Systemy komputerowe Jednolitego Systemu EC1032 oraz EC1034 oferują swoim użytkownikom oprogramowanie TELE JS szeroko rozpowszechnione w kraju i poza jego granicami. Umożliwia ono dostęp z terminali do wybranych podsystemów, takich jak System Kontroli i Obsługi Terminali (SKOT) i dalej do baz danych czy do podsystemu wielodostępnego TSO. Terminale zdalne (podłączone do komputera czołowego EC8371.01) obsługiwane są przez program emulacyjny (EP) umieszczony w tym komputerze. Terminale lokalne, podłączone bezpośrednio do kanału komputera obliczeniowego, obsługiwane są na poziomie fizycznym metodą TCAM lub BTAM (metody telekomunikacyjne). Podstawową wadą oprogramowania TELE JS jest brak standardów jego wykorzystania. Dlatego, chociaż jest możliwość łączenia się z terminala z różnymi aplikacjami, rzadko się z tego korzysta (wygenerowanie takiego programu obsługi terminali nie jest zresztą proste).

Idąc za przykładem firmy IBM, która rozpowszechniła swoją architekturę sieci SNA (niezgodnej z modelem ISO) tworząc własne sieci komputerowe (np. EARN), polski przemysł komputerowy wystąpił z inicjatywą zaprojektowania również własnej sieci komputerowej, zgodnej z architekturą sieci otwartych.

2.2. Główne wymagania i ograniczenia

Przystępując do zaprojektowania sieci SKJS2 wersja 1 zamawiający przedstawił wymagania i założenia, które były następujące:

a) wymagania użytkowe

- zapewnić komunikację między dowolnym terminalem sieci a dowolnie wybraną aplikacją sieciową

- zapewnić łączność między konsolami operatorskimi systemów autonomicznych (komputerów obliczeniowych)

- zapewnić transfer zbiorów sekwencyjnych między dowolnymi komputerami obliczeniowymi sieci,

b) wymagania projektowe

- umożliwić tworzenie dowolnej konfiguracji sieci

- umożliwić rozbudowę sieci o dodatkowe komputery obliczeniowe, węzły i zasoby (aplikacje, terminale)

- zapewnić prostą organizację własnych systemów użytkowych

- zapewnić obsługę w sieci dosyć szerokiego wachlarza terminali,

c) wymagania dotyczące sprzętu

- sprzęt wykorzystany do pracy w sieci musi być produkowany lub kompletowany przez ZE ELWRO

- oprogramowanie dostarczane użytkownikom, pracujące dotychczas pod systemem TELE JS (TSO, SKOT), powinno być dostępne w sieci,

d) wymagania eksploatacyjne

- zapewnić diagnozowanie stanów sieci

- zapewnić łagodzenie skutków awarii sprzętu

- zapewnić raportowanie i pomiary pracy sieci.

Jednocześnie założono, że terminale podłączone do komputera czołowego będą obsługiwane przez program NCP (ang. Network Control Program), co spowodowało konieczność wykorzystania telekomunikacyjnej metody dostępu TCAMS.

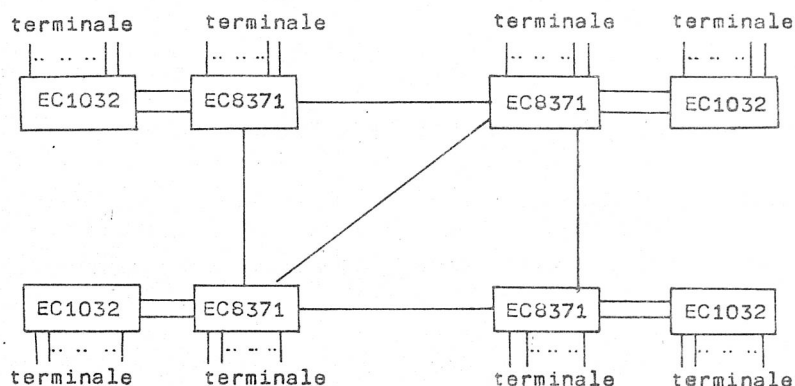
2.3. Struktura sieci

Sieć komputerowa składa się ze środków transmisji danych, komputerów i terminali. Ogólną strukturę sieci przedstawiono na rys. 1. Istotną cechą tej sieci jest połączenie w jednym komputerze EC8371.01 funkcji komputera czołowego i komputera węzła sieci. Każdy komputer EC8371.01 umieszczony jest w niewielkiej odległości od komputera obliczeniowego (identycznie jak w TELE JS). Dwa komputery EC8371.01 lub większa ich liczba połączone są między sobą modemami i liniami telefonicznymi.

W sieci można wykorzystywać następujące komponenty sprzętowe:

a) komputery obliczeniowe JS z systemem OS/JS 5.01 MVT,

b) komputer czołowy i węzła EC8371.01,



Rys.1. Przykładowa struktura sieci SKJS2 wersja 1
Fig. 1. The sample structure of the SKJS2 v.1 computer network

c) terminale - stacja abonencka EC8575M, rodzina monitorów ekranowych EC7900 (7910, 7914, 7915, 7917).

d) środki transmisji - dowolny terminal podłączony jest do EC8371.01 jak dotychczas. EC8371.01 między sobą łączone są za pomocą modemów EC8013 i czterodrotu (lub innych modemów pracujących z szybkością do 9600 b/s).

Cechą charakterystyczną struktury sprzętowej jest to, że nie różni się ona praktycznie od tej, jaką użytkownicy posiadają w konfiguracjach autonomicznych (potrzebne są tylko dodatkowe modemy i linie między węzłami).

2.4. Oprogramowanie sieci

System operacyjny OS/JS 5.01 MVT, telekomunikacyjna metoda dostępu TCAM5 oraz program sterujący NCP stanowią środowisko, w którym umiejscowione jest oprogramowanie sieciowe. Prace w trybie sieciowym zapewniają moduły programowe, do których należą:

a) Komputer obliczeniowy. Głównym modułem oprogramowania jest moduł MŁOT (moduł łącznikowy obsługi transmisji), który steruje terminalami oraz tworzy połączenia sieciowe. Moduł komunikuje się z oprogramowaniem węzła w EC8371.01 przez wydzielony podkanał multiplekserowy i z programem MCP realizującym metodę dostępu TCAM5, która obsługuje między innymi terminale na poziomie fizycznym [11]. Program MCP wygenerowany do użytku sieciowego stanowi medium, za pomocą którego następuje przepływ komunikatów między programem MŁOT a aplikacjami sieciowymi i terminalami. Moduł MŁOT zakodowano w języku

stęp do tych zasobów jest możliwy z każdego terminala sieci. Połączenie z wybraną aplikacją następuje po wydaniu z terminala komendy NETON z odpowiednimi parametrami, rozłączenie po wydaniu komendy NETOFF. Po rozłączeniu można połączyć się ponownie z dowolną aplikacją. Usługi komunikacyjne są realizowane przez oprogramowanie sieciowe i są niewidoczne dla użytkownika.

Operator konsoli operatorskiej, która jednocześnie pełni funkcję konsoli sieciowej, ma możliwość prostego diagnozowania połączeń sieciowych, badania ich liczby i stanu oraz może wymieniać informacje z operatorem innego komputera obliczeniowego sieci.

2.6. Otrzymane wyniki

Całość bazowego oprogramowania sieci została przetestowana na konfiguracji składającej się z trzech zestawów maszynowych [14]. Podczas pracy terminalowej nie odczuwa się czy współpracuje się z aplikacją lokalnego czy zdalnego komputera obliczeniowego (praktycznie nie odczuwa się opóźnień czasowych [16]).

Krótką charakterystyką pracy z aplikacjami sieciowymi:

a) TSO - wprowadzone zmiany umożliwiają pracę dwukierową, co wpływa na przyspieszenie wprowadzania danych z terminala, możliwe jest w każdej chwili wysłanie sygnału przerwania; uzyskano też nowe możliwości pracy ze standardowym edytorem TSO - po wyświetleniu na ekranie tekstu numerowanego można dokonać zmian we wszystkich liniach i przesyłać poprawiony tekst do edytora przyciskając jeden klawisz [6, 12];

b) SKOT - pracuje tak samo jak dotychczas (niedawno w ramach prac własnych uruchomiono w sieci SKJS2 wersja 1 pod systemem SKOT system wyszukiwania danych bibliograficznych ISIS - istnieje zatem możliwość korzystania z baz danych; baza używana w Politechnice zawiera wszystkie skatalogowane przez Bibliotekę Główną pozycje zwarte ;

c) STZ - jest jedyną aplikacją sieciową utworzoną od podstaw z myślą o pracy sieciowej; umożliwia ona przesyłanie dowolnych zbiorów sekwencyjnych (po rozładowaniu można przysyłać zbiory biblioteczne i indeksowo-sekwencyjne)

LITERATURA

- [1] BUDZIANOWSKI I. i in. , Projekt techniczny oprogramowania PTD jako procesora czołowego i węzła w sieci SKJS2 wersja 1, Inst.Cybern. Techn. PWr., Raport SPR 11/84, Wrocław 1984.

- [2] DUBIELEWICZ A. i in., Stacja transferu zbiorów w sieci SKJS2 wersja 1, CO PWr., Raport SPR 18/85, Wrocław 1985.
- [3] DUBIELEWICZ I. i in., Aparatura pomiarowo-diagnostyczna dla sieci SKJS2 wersja 1, CO PWr., Raport SPR 13/86, Wrocław 1986.
- [4] DYRKA K. i in., Zasady tworzenia i udostępniania zasobów w sieci SKJS2 wersja 1, CO PWr., Raport SPR 14/86, Wrocław 1986.
- [5] HARMS J., La situation internationale, SWITCH, no. 1, April 1986, s. 18-22.
- [6] HUZAR A. i in., Terminal w sieci SKJS2 wersja 1, CO PWr., Raport SPR 20/86, Wrocław 1986.
- [7] HUZAR A. i in., Projekt techniczny oprogramowania EC1032 jako komputera obliczeniowego w sieci SKJS2 wersja 1, CO PWr., Raport SPR 24/86, Wrocław 1986.
- [8] KREMIENOWSKI P. i in., Dokumentacja eksploatacyjna zadań kontrolnych i pomiarowych do badań sieci SKJS2 wersja 1, IKSAiP, Wrocław 1986.
- [9] SADLAK J., The future of academic networks in Europe, CRE-INFORMATION No. 26, 4th Quarter, 1986, s. 163-167.
- [10] BUDZIANOWSKI J., WIETRZYCH J., FRYŚ A., Struktura warstwowa a standardy w sieci SKJS2 wersja 1, (w niniejszym zeszycie).
- [11] STAŃKO J., ZAK D., Styki a sterowanie przepływem w sieci SKJS2 wersja 1 (w niniejszym zeszycie).
- [12] JACUKOWICZ M., JANISZEWSKI A., KWIATKOWSKI J., Dostęp terminalowy w sieci SKJS2 wersja 1 (w niniejszym zeszycie).
- [13] HUZAR A., KALIŚ A., Problemy synchronizacji w stacji transferu zbiorów (w niniejszym zeszycie).
- [14] KALIŚ A., HUZAR A., Tworzenie oprogramowania bazowego sieci komputerowej SKJS2 wersja 1 (w niniejszym zeszycie).
- [15] FRYŻLEWICZ Z., Aparatura pomiarowo-diagnostyczna dla sieci komputerowej (w niniejszym zeszycie).
- [16] DYRKA K., KREMIENOWSKI P., RUTKOWSKI E., Charakterystyki czasowe i przepustowość w sieci SKJS2 wersja 1 (w niniejszym zeszycie).

Praca wpłynęła do Redakcji 1988.03.28

THE COMPUTER NETWORK SKJS2 VERSION 1

The basic information on the network of the unified system called SKJS2 version 1, are given. The hardware and software network configuration are described. The work of the basic network applications: subsystems TSO, CICS and file transfer station is characterized.

ВЫЧИСЛИТЕЛЬНАЯ СЕТЬ ЕДИНСТВЕННОЙ СИСТЕМЫ SKJS2 ВЕРСИЯ 1

В статье представлены основные информации на тему вычислительной сети компьютеров единственной системы, называемой SKJS2 версия 1. Обсуждены оборудованная конфигурация и логическая архитектура сети. Изображена работа основных применений сети: подсистем TSO, SKOT и станции трансфера файлов.

*Sieć komputerowa, węzeł
komutacji pakietów, protokoły,
algorytm wyboru drogi*

Lesław BUDZIANOWSKI *

Jerzy WIETRZYCH*

Andrzej FRYS*

STRUKTURA WARSTWOWA A STANDARDY W SIECI SKJS2 WERSJA 1

W pracy przedstawiono strukturę logiczną sieci komputerowej SKJS2 wersja 1. Omówiono poszczególne warstwy i przyjęte protokoły z uwzględnieniem nałożonych ograniczeń. Omówiono przede wszystkim algorytm wyboru drogi oraz sposoby ochrony przed przeciążeniem podsieci SKJS2 wersja 1. Następnie przedstawiono budowę oprogramowania realizującego cztery dolne warstwy według modelu OSI/ISO, które umieszczono w procesorze EC8371.01.

1. WPROWADZENIE

Budowę sieci komputerowej SKJS2 wersja 1 oparto na maszynach cyfrowych EC1032 lub innych z serii EC spełniających funkcję komputerów obliczeniowych KO oraz procesorach teleprzetwarzania danych EC8371.01, które pełnią rolę komputerów czołowych KC i węzłów podsieci komunikacyjnej. W sieci tej jako aplikacje stosuje się oprogramowanie użytkowe współpracujące z metodą dostępu telekomunikacyjnego TCAM5. Ponadto w komputerze obliczeniowym realizowane są usługi transferu zbiorów. Natomiast procesor teleprzetwarzania danych spełnia następujące funkcje:

x

Centrum Obliczeniowe Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-307 Wrocław.

- komputera czołowego do EC1032 i innych,
- koncentratora terminali,
- węzła podsieci komunikacyjnej,
- łącznika do podsieci zgodnej z zaleceniem X.25.

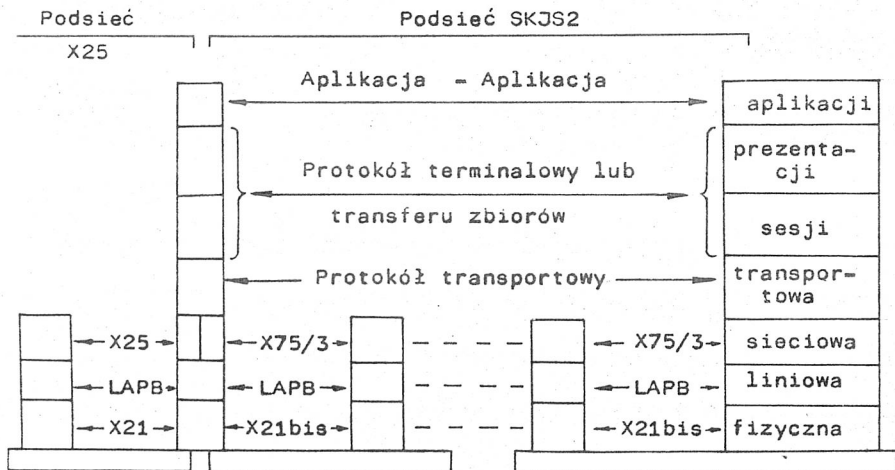
Z podanych założeń wynika, że sieć SKJS2 wersja 1 ma charakter sieci homogenicznej.

2. PROTOKOŁY STOSOWANE W SIECI SKJS2 WERSJA 1

W sieci SKJS2 wersja 1 przyjęto układ protokołów zgodny z modelem odniesienia ISO/OSI (do czwartej warstwy włącznie). Wynika stąd następujący podział na warstwy i zawarte w nich protokoły współpracy:

- warstwa fizyczna (protokół X.21 bis według CCITT)
- warstwa liniowa (protokół LAPB według X.25 CCITT)
- warstwa sieciowa (X.25 według CCITT oraz X.75 do współpracy międzywęzłowej)
- warstwa transportowa (protokół transportowy według ISO)
- warstwa sesji i prezentacji (własny protokół terminalowy umożliwiający negocjację typów terminali oraz protokół transferu zbiorów).

Wyspecyfikowany układ protokołów przedstawiono na rys. 1. Przyjęto dodatkowe ustalenie, w myśl którego jedno połączenie sesyjne jest odwzorowane w jedno połączenie transportowe, a to z kolei w jedno połączenie sieciowe.



Rys.1. Układ protokołów wykorzystywany podczas współpracy z podsiecią
Fig.1. Structure protocols used in the computer network

2.1. Warstwa fizyczna

Warstwa fizyczna ma za zadanie umożliwić korzystanie z rzeczywistych fizycznych środków łączności, które są dostępne w formie usług dla warstwy liniowej. Usługi te polegają na przezroczystej transmisji strumienia bitów między dwoma stacjami warstwy liniowej z zachowaniem ich kolejności. Są one realizowane przez protokół określający własności mechaniczne, elektryczne i funkcjonalne warstwy, niezbędne do aktywacji, utrzymania i deaktywacji połączenia między dwoma stacjami warstwy liniowej. W sieci SKJS2 wersja 1 w warstwie fizycznej zastosowano protokół zgodny z X.21 bis CCITT.

2.2. Warstwa liniowa

Warstwa liniowa dostarcza środków proceduralnych i funkcjonalnych do ustanowienia, utrzymania i zwalniania połączenia między stacjami warstwy sieciowej oraz do przesyłania jednostek danych, zwanych ramkami. Warstwa ta wykrywa i usuwa błędy, które mogą powstać w warstwie fizycznej. Steruje także przepływem danych, dostosowując tempo przesyłania ramek do możliwości odbiorczych stacji warstwy sieciowej. Błędy, nienaprawialne przez warstwę liniową są sygnalizowane warstwie sieciowej. Powyższe funkcje są realizowane w warstwie przez stację protokołu liniowego. W sieci SKJS2 wersja 1 zastosowano protokół LAPB w wersji podstawowej (bez rozszerzonej numeracji i połączeń wielopunktowych).

2.3. Warstwa sieciowa

Warstwa sieciowa dostarcza środków do ustanowienia, utrzymania i likwidowania połączeń sieciowych między systemami końcowymi oraz procedur i środków funkcjonalnych do wymiany sieciowych jednostek danych, zwanych pakietami. Warstwa ta uniezależnia połączenie od charakterystyk transmisji w stacjach liniowych oraz dostarcza usług warstwie transportowej takich, jak: zawiadamianie o błędach, zachowanie kolejności przesyłanych jednostek danych, zerowanie (usuwanie wszystkich jednostek danych z połączenia sieciowego). Realizowane są w niej również funkcje wykrywania i korekty błędów sygnalizowanych z warstwy liniowej. Powyższe funkcje realizuje w warstwie stacja protokołu pakietowego.

W sieci SKJS2 wersja 1 zastosowano dwa protokoły pakietowe. Pierwszy, zgodny z zaleceniem X.25 CCITT [4], wykorzystywany jest tylko do współpracy komputerów obliczeniowych przez podsieć publiczną. Natomiast drugi, zgodny z zaleceniem X.75 CCITT [5], wykorzystywany

jest wewnątrz sieci SKJS2 wersja 1 i służy do realizacji własnej podsieci komunikacyjnej SKJS2. Należy zauważyć, że większość procedur jest wspólna dla obu protokołów pakietowych.

Dla protokołu pakietowego opartego na zaleceniu X.25 przyjęto ustalenia, zgodnie z którymi jedyną wykorzystywaną usługą będą połączenia typu VIRTUAL CALL (VC). Będą one służyły poszczególnym stacjom transportowym do utrzymania połączeń między systemami aplikacyjnymi i terminalami. Dodatkowo założono, że nie dopuszcza się stosowania pakietu REJECT oraz opcji FAST SELECT. Pakiety są numerowane modulo 8, a ich długość nie przekracza 128 oktetów. Cechą charakterystyczną tego protokołu jest też to, że rozłączanie, zerowanie, jak i potwierdzanie przesyłanych danych może mieć działanie o charakterze lokalnym lub zdalnym - wymagającym zgody obu współpracujących stacji końcowych.

Dla protokołu pakietowego międzywęzłowego opartego na X.75, który definiuje reguły wymiany danych na styku STEX/STEY (gdzie STEX i STEY są stacjami tej wymiany) założono, że wszystkie reguły przyjęte w protokole pakietowym typu X.25 obowiązują w protokole międzywęzłowym. Dodatkowo wykorzystywane są usługi typu bezpołączeniowego, w których jednostką danych jest DATAGRAM (DG). Usługi te są stosowane wyłącznie do komunikacji procesów operatorskich oraz do sterowania podsiecią.

Ważną funkcją warstwy sieciowej jest sterowanie przepływem danych. Powinno ono chronić zasoby systemów końcowych przed przeciążeniem. W tym celu kontrolowane są poszczególne kolejki odebranych pakietów DATA w połączeniach sieciowych. W razie przekroczenia dopuszczalnej długości takiej kolejki zostaje wysłany pakiet RNR, nakazujący wstrzymanie wysyłania dalszych pakietów danych. Dodatkowo kontrolowana jest liczba wolnych buforów przeznaczonych do przechowywania pakietów DATA. Gdy liczba ta jest mniejsza od zadanej wartości granicznej, w poszczególnych połączeniach sieciowych wysyłane są pakiety RNR w celu powstrzymania napływających danych. Powrót do stanu przesyłania danych, osiągany przez wysłanie pakietu RR następuje wtedy, gdy długość kolejki otrzymanych pakietów DATA zmaleje do wielkości granicznej minimalnej, a równocześnie liczba wolnych buforów zwiększy się powyżej wartości progowej. W przypadku kontrolowania długości kolejek istotny jest problem właściwego doboru wartości granicznych, których osiągnięcie powoduje wysłanie RNR lub RR, gdyż od tych wartości zależy przepustowość poszczególnych połączeń sieciowych.

Istotną funkcją warstwy sieciowej, a tym samym i protokołu pakietowego międzywęzłowego, jest zestawianie połączenia sieciowego

między systemami końcowymi. W tym celu wykorzystuje się odpowiedni algorytm wyboru drogi.

2.3.1. Wybór drogi w sieci SKJS2 wersja 1

Podstawą działania algorytmu wyboru drogi w sieci SKJS2 wersja 1 jest odpowiednio skonstruowana tablica kierunków. Tablica kierunków jest strukturą danych umieszczoną w każdym EC8372.01 (węźle) sieci, decydującą o kierunkach przekazywania żądań nawiązania połączeń i datagramów. Dotyczy to wszystkich połączeń zarówno inicjowanych z lokalnego komputera obliczeniowego, jak i tranzytowych (rozpoczynających się w innych węzłach sieci). W tablicy kierunków każdego EC8371.01 reprezentowane są adresy i drogi dojścia do każdego komputera obliczeniowego w sieci łącznie z komputerem lokalnym.

Podstawą do utworzenia tablicy kierunków każdego z węzłów jest określenie dróg pierwotnych i zastępczych wiodących z danego węzła do wszystkich pozostałych. Droga pierwotna będzie używana do tworzenia połączenia między węzłami w sytuacji, gdy wszystkie elementy sieci są sprawne. Drogi zastępcze będą wybierane do tworzenia połączenia po wykryciu niesprawności drogi pierwotnej. O kolejności wybierania dróg zastępczych do próby tworzenia połączeń decyduje kolejność ich wyszczególnienia w tablicy kierunków. Określając w tablicy kierunków drogę między węzłami A i B specyfikujemy w każdym z nich tylko jej początek, tzn. numer wyjścia teleprocesora, przez który ma być wysłane żądanie ustanowienia połączenia, tak aby dotarło ono do celu używając drogi pierwotnej lub zastępczej.

Między dowolnymi węzłami A i B musi być określona co najmniej droga pierwotna i ewentualnie drogi zastępcze w miarę potrzeb i możliwości ich rozróżnienia. Wyszczególnienie tych dróg jest umieszczane w tablicy kierunków, która jest listą jednokierunkową, składającą się z elementów określających węzły docelowe w sieci. W każdym z tych elementów znajdują się następujące pola:

- wskaźnik (adres) następnego elementu w liście
- sieciowy adres docelowego EC8371.01
- liczba możliwych kierunków prowadzących do wskazanego adresu docelowego
- pierwszy numer wyjścia
- drugi numer wyjścia
- itd.
- n-ty numer wyjścia.

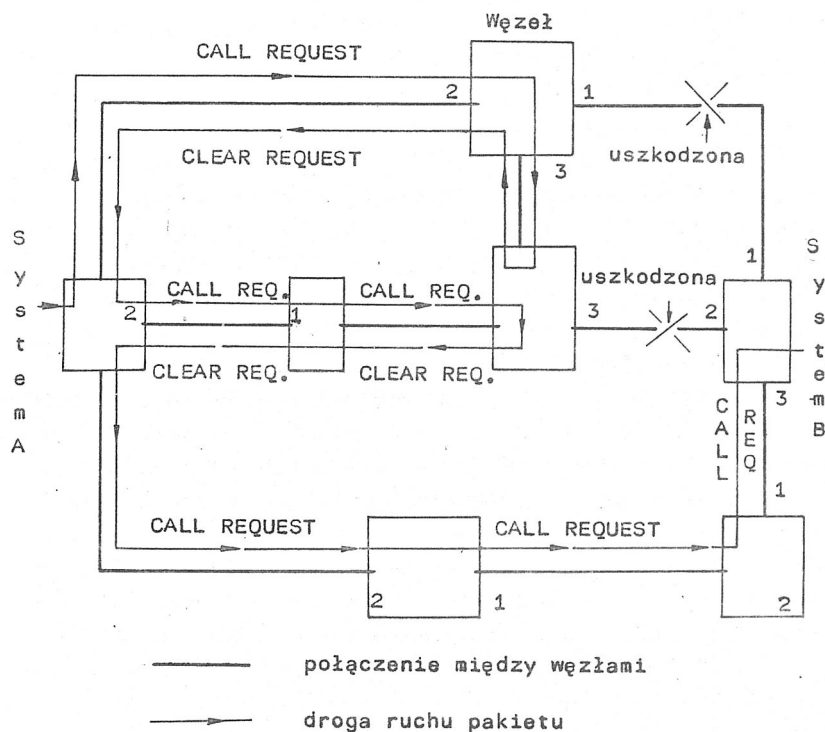
Żądanie ustanowienia połączenia sieciowego niezależnie od pochodzenia (może ono pochodzić od stacji transportowej, gdy jest inicjowane lub od stacji sieciowej, gdy przychodzi z innego węzła) powo-

duje wybranie z tablicy kierunków elementu określającego żądany adres docelowy. Brak takiego elementu powoduje odrzucenie połączenia ze wskazaniem przyczyny - błąd adresacji. Natomiast, gdy istnieje możliwość dojścia do systemu końcowego, określonego przez adres docelowy, wybierane jest odpowiednie wyjście pierwotne lub zastępcze teleprocesora. Następnie, po sprawdzeniu jego aktywności i możliwości podjęcia obsługi nowego kanału, w tym wybranym kierunku wysyła się pakiet CALL REQUEST, który będzie analogicznie rozpatrywany w następnym węźle. Jeżeli dany węzeł okaże się węzłem docelowym, następuje potwierdzenie połączenia sieciowego pakietem CALL ACCEPTED pod warunkiem, że stacja transportowa jest aktywna i zdolna do tworzenia nowych połączeń transportowych. Brak aktywnego wyjścia, przez które można skierować żądanie ustanowienia połączenia w kierunku docelowym powoduje odesłanie żądania rozłączenia, ze wskazaniem przyczyny "brak dojścia" do węzła poprzedzającego.

W celu zabezpieczenia się przed powstaniem pętli każde połączenie sieciowe ma nadany numer identyfikacyjny, unikatowy w całej sieci. Gdy pakiet CALL REQUEST zgłaszający nowo tworzone połączenie zawiera numer identyfikacyjny taki sam jak numer jednego z kanałów logicznych utworzonych wcześniej w danym węźle, wykrywane jest powstanie pętli. Nowe żądanie jest wtedy odrzucane (za pomocą pakietu CLEAR REQUEST) ze wskazaniem przyczyny: "powstanie pętli" i propaguje się je dalej przez sieć, kasując utworzony fragment połączenia. Należy tu wyjaśnić, że informacje o poszczególnych połączeniach sieciowych przechodzących przez dany węzeł są przechowywane w powiązanych ze sobą tablicach odzwierciedlających stany kanałów logicznych przypisanych do odpowiednich wyjść komputera czołowego.

Otrzymane w dowolnym węźle żądanie rozłączenia zgłaszane przez pakiet CLEAR REQUEST z przyczyną inną niż powstanie pętli lub brak dojścia jest przekazywane dalej do żądającego ustanowienia połączenia sieciowego, co ostatecznie kończy próby znalezienia drogi do punktu docelowego. Natomiast w razie powstania pętli lub braku dojścia następuje próba znalezienia innej drogi zastępczej przez przeszukiwanie pozostałych kierunków jeszcze w danym węźle nie sprawdzonych.

Przykładową próbę zestawienia połączenia przedstawiono na rys. 2. Pokazano na nim drogę, jaką musi przebyć pakiet CALL REQUEST od systemu A do B w przypadku, gdy istnieją uszkodzenia niektórych dróg pierwotnych. Najpierw próbuje się wysłać go z systemu A wyjściem numer 1. Po otrzymaniu odpowiedzi CLEAR REQUEST z tego kierunku dokonywana jest próba wysłania pakietu wyjściem numer 2, a następnie



Rys. 2. Przykładowe zestawienie połączenia sieciowego

Fig. 2. Example of network connection establishment

wyjściem numer 3. Dopiero ta trzecia próba okazuje się skuteczna i w jej wyniku może się pojawić odpowiedź CALL ACCEPTED po uzyskaniu zgody pozostałych systemów biorących udział w tworzeniu połączenia.

Reguły wyboru drogi podczas przesyłania datagramów opierają się, podobnie jak w zestawieniu połączeń sieciowych, na zawartości tablicy kierunków. Cechą charakterystyczną, różniącą przesyłanie datagramów od zestawiania połączenia sieciowego, jest brak śladu w danym węźle po wysłaniu datagramu. W związku z tym wskaźniki niezbędne do uzyskania poprawnej pracy algorytmu wyboru drogi umieszczane są wewnątrz datagramu i przesyłane wraz z nim. Dotyczy to wskaźnika kierunków ruchu przyjmującego dwie wartości: ruch "do przodu" lub ruch "wstecz" oraz licznika kroków ruchu "do przodu", wykorzystywanego do ochrony przed powstaniem pętli.

W celu wysłania datagramu sprawdza się tablicę kierunków, aby znaleźć element o adresie docelowym zgodnym z podanym w datagramie. Po znalezieniu takiego elementu następuje wybranie numeru wyjścia, przez które będzie można wysłać datagram. Dzieje się to pod warunkiem, że wybrane wyjście (stacja sieciowa) jest aktywne, a liczba datagramów oczekujących w kolejce na wysłanie nie przekracza zadanego limitu. W takiej sytuacji datagram jest dołączany do kolejki z zaznaczeniem w nim, że jest on wysyłany "do przodu" oraz zwiększeniem zawartego w nim licznika kroków ruchu "do przodu" o 1.

Gdy w tablicy kierunków nie ma takiego elementu bądź nie ma dostępnego wyjścia, datagram jest odsyłany w tym samym kierunku, z którego przyszedł z zaznaczeniem w nim, że jest wysyłany "wstecz". Przyjmuje się, że datagram dotarł do celu w chwili, gdy wykrywa się zgodność adresu docelowego w datagramie z adresem własnym danego węzła.

W razie otrzymania datagramu ze wskaźnikiem informującym o ruchu datagramu "wstecz" następuje wybranie następnej nie sprawdzonej drogi zastępczej. Gdy taka droga istnieje, datagram jest wysyłany z zaznaczeniem kierunku ruchu "do przodu". Natomiast, gdy nie istnieje inna droga zastępcza w tym węźle doprowadzająca datagram do celu, dany datagram wysyłany jest "wstecz" do kolejnego węzła. Osiągnięcie węzła źródłowego podczas ruchu datagramu "wstecz" powoduje skasowanie datagramu z powodu niemożności dotarcia do adresu docelowego. Datagram jest kasowany także, gdy zawarty w nim wskaźnik liczby prób wyboru drogi "do przodu", podający liczbę przejść przez węzły sieci, przekroczy zadany limit. Zabezpiecza to przed krążeniem datagramów w sieci i powstawaniem pętli.

Jak można się przekonać, algorytm zastosowany w sieci SKJS2 wersja 1 spełnia w znacznym stopniu postulaty stawiane w literaturze [6]. Przede wszystkim ma charakter rozproszony, decyzje o wyborze drogi podejmowane są w poszczególnych węzłach na tworzonej drodze. W tablicach kierunków nie jest przechowywana droga z końca do końca, a jedynie kierunek do najbliższego sąsiada, który może pośredniczyć w osiągnięciu punktu docelowego. Algorytm zapewnia wykrycie i likwidację powstałej pętli i jest wolny od oscylacji w sieci. Jest stosowany do ustanawiania logicznych połączeń sieciowych, jak i do przesyłania informacji bez tworzenia połączeń (zgodnie z X.25 i X.75). Zalecany w literaturze warunek adaptacyjności algorytmu wyboru drogi nie jest spełniony. Cecha ta wiąże się z koniecznością użycia dodatkowej pamięci zewnętrznej, której wymaga system zbierania, analizy i rozsyłania informacji o zmianach w sieci. Procesor EC8371.01 charakteryzuje się brakiem własnej pamięci zewnętrznej o bezpośrednim dostępie, jak i ograniczoną pamięcią operacyjną. Pewne elementy adapta-

cyjności zostały uwzględnione przez wprowadzenie mechanizmu aktywacji poszczególnych kierunków. Gdy brak aktywności danego kierunku, nie jest on brany pod uwagę w rozważaniach dotyczących wyboru drogi. Istnieje też możliwość interwencji operatorskiej przez włączenie lub wyłączenie pewnych kierunków w celu dostrojenia parametrów sieci. Sam algorytm może pracować przy adaptacyjnym modelu wyboru drogi. Wiąże się to tylko z dynamiczną zmianą kolejności numerów kierunków w tablicy kierunków. Pewną niedogodnością tego rozwiązania, związaną z ograniczonymi zasobami pamięci, jest konieczność zmiany tablicy kierunków we wszystkich węzłach w razie dołączenia nowego węzła.

Istnieją duże trudności z wykazaniem, że w dowolnym rzeczywistym przypadku dany algorytm jest najlepszy. Niektóre algorytmy górują nad innymi, gdy jest ustabilizowane natężenie ruchu, ale bardzo słabo reagują na sytuacje wyjątkowe. Wybór algorytmu powinien uwzględniać wielkość sieci, niezawodność sprzętu i posiadane zasoby pamięci.

Wydaje się, że zaproponowany algorytm wyboru drogi spełnia żądane wymagania w sieci SKJS2 wersja 1, co też zostało potwierdzone eksperymentalnie.

2.4. Warstwa transportowa

Warstwa transportowa udostępnia warstwie wyższej środków do przesyłania danych w sposób minimalizujący koszty i zasoby, gwarantując jednocześnie wymaganą jakość.

Warstwa transportowa podnosi jakość usług dostarczonych przez warstwę sieciową do poziomu wymaganego przez warstwę sesji. Równocześnie warstwa ta odciąża i uniezależnia wyższe warstwy od wszystkich czynności związanych z przesyłaniem informacji.

Protokół transportowy wybrany dla sieci SKJS2 wersja 1 należy do klasy protokołów zorientowanych połączeniowo i funkcjonalnie odpowiada propozycji ISO dla protokołu transportowego klasy 1 [3]. Protokół ten opiera się na usługach dostarczanych przez kanały logiczne protokołu pakietowego X.25 lub X.75 (tworzących połączenia sieciowe) i przejmuje funkcje dwukierunkowego przesyłania nieinterpretowanych danych użytkownika połączeniem transportowym.

Wykorzystując usługi warstwy sieciowej oraz wzbogacając je i uzupełniając funkcjami własnymi, stacje protokołu transportowego dostarczają związanym stacjom warstwy wyższej (terminalowym lub transferu zbiorów) następujących usług transportowych:

- nawiązywanie połączenia transportowego wraz z negocjacją warunków przesyłania danych

- przesyłanie potwierdzonych danych nie podlegających mechanizmowi sterowania przepływem na poziomie transportowym

mowi sterowania przepływem na poziomie transportowym

- przesyłanie potwierdzonych danych przyspieszonych
- odtwarzanie połączeń sieciowych i synchronizacji ciągów danych po sygnalizowanych do warstwy transportowej niesprawnościach w warstwie sieciowej
- informowanie użytkownika o nienaprawialnych błędach własnych protokołu
- uporządkowanego zamykania połączenia transportowego
- informowania o jakości usług transportowych.

Podstawowym powodem, dla którego zaproponowano protokół transportowy, jest zautomatyzowane odtwarzanie dwukierunkowej transmisji danych użytkownika po sygnalizowanych niesprawnościach z warstwy sieciowej, jako następstwa warunków CLEAR lub RESET, pojawiających się w podsieci komunikacyjnej. W ten sposób naprawiane są wszelkie sygnalizowane do warstwy transportowej niesprawności podsieci, bez informowania o fakcie warstwy wyższej.

Komunikacja pary stacji transportowych polega na wzajemnej wymianie jednostek danych i jednostek sterujących zwanych komunikatami, przy czym zarówno komunikaty danych, jak i komunikaty sterujące, przenoszone są wyłącznie w pakietach DATA protokołu pakietowego. A zatem trzy kolejne fazy funkcjonowania protokołu transportowego - nawiązanie połączenia, transmisja danych i zamykanie połączenia, występują wyłącznie w fazie transmisji danych połączenia sieciowego. Zasady komunikowania się pary stacji transportowych są zgodne z pracą [1,2], przy czym w implementacji protokołu na potrzeby SKJS2 wersja 1 przyjmuje się, że potwierdzenie komunikatów danych następuje w warstwie transportowej oraz wykorzystuje się transportową drogę danych przyspieszonych. Dodatkowo dopuszczono stosowanie klasy 3 bez opcji podziału łącza na podkanały. Wersja ta umożliwia sterowanie przepływem komunikatów danych w warstwie transportowej. W sieci SKJS2 wersja 1 wysyła się potwierdzenia AK tylko wtedy, gdy kolejka z odebranymi komunikatami danych nie przekroczyła wartości granicznej. Właściwość ta jest wykorzystywana do zarządzania zasobami systemu końcowego w celu ochrony przed przepełnieniem systemów końcowych.

2.5. Warstwa sesji i prezentacji

W sieci SKJS2 wersja 1 warstwy te są traktowane łącznie, ponieważ sieć ta jest siecią homogeniczną. Przyjęto, że w warstwie tej obowiązują reguły właściwe dla maszyny cyfrowej EC1032. Dlatego też dla współpracy terminal-aplikacja opracowano własny protokół umożliwiający ustalenie typu terminala, który chce współpracować z procesem

aplikacyjnym. Natomiast do transferu zbiorów zastosowano protokół oparty na zaleceniu NIFTP Blue Book.

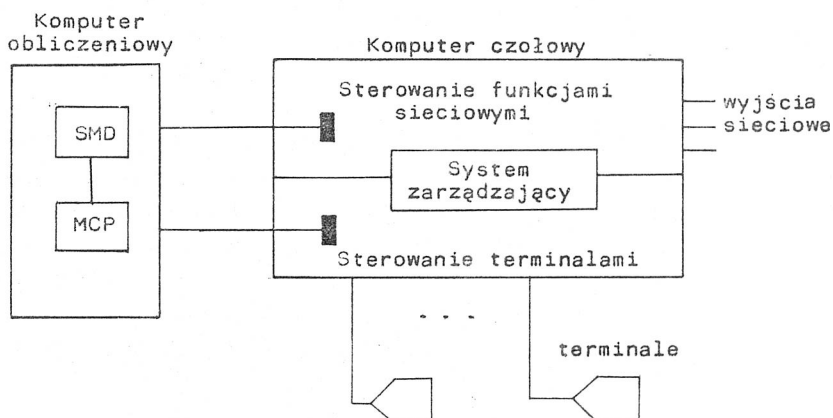
W sieci SKJS2 wersja 1 warstwy 1-4 zostały zaimplementowane w komputerze czołowym i węzła, natomiast warstwy wyższe w komputerze obliczeniowym.

3. OGÓLNA BUDOWA OPROGRAMOWANIA WĘZŁA

Oprogramowanie EC8371.01 składa się z trzech funkcjonalnych części:

- sterującej terminalami
- realizującej funkcje sieciowe
- systemu zarządzającego (supervisora).

Oprogramowanie sterujące terminalami stanowi pośrednik między metodą dostępu telekomunikacyjnego TCAM5 w komputerze obliczeniowym a terminalami podłączonymi do komputera czołowego. Ta część oprogramowania nie komunikuje się z częścią realizującą funkcje sieciowe. Obie części korzystają natomiast ze wspólnego systemu zarządzającego zasobami procesora EC8371.01. Schematycznie przedstawiono to na rys. 3.



Rys. 3. Ogólna struktura oprogramowania komputera czołowego i węzła

Fig. 3. The general structure of the front-end-processor and node software

Współpraca terminali z siecią odbywa się za pośrednictwem komputera obliczeniowego. Dwa z wymienionych tu elementów, tj. sterowa-

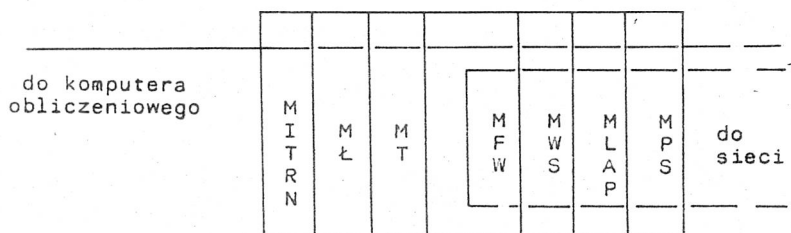
wanie terminalami na poziomie fizycznym oraz system zarządzający, zostały przyjęte ze standardowego oprogramowania o nazwie NCP (Network Control Program). Sterowanie funkcjami sieciowymi jest oryginalnym elementem dobudowanym do programu NCP, co daje nowy system sieciowego oprogramowania procesora EC8371.01, spełniającego rolę węzła sieci i komputera czołowego do współpracy z siecią i terminalami. Oprogramowanie to nie realizuje natomiast funkcji koncentratora terminali do celów współpracy z siecią.

Funkcje sieciowe obejmują tworzenie, utrzymanie i likwidowanie połączeń z komputera obliczeniowego do sieci oraz połączeń tranzytowych między elementami składowymi sieci. W zakresie współpracy komputera obliczeniowego z siecią ta część oprogramowania EC8371.01 jest realizatorem warstw: transportowej, sieciowej, liniowej i fizycznej. Równocześnie oprogramowanie to realizuje funkcje węzła sieci SKJS2 wersja 1, a więc zawiera omówione wcześniej mechanizmy sterujące.

Wyróżnia się następujące moduły programowe realizujące funkcje sieciowe:

- obsługi styku transparentnego (MITRN) do celów komunikacji z procesami w komputerze obliczeniowym
- łącznikowy (MŁ) realizujący usługi warstwy transportowej
- transportowy (MT)
- funkcji węzła (MFW)
- wyjść sieciowych (MWS)
- obsługi łącza LAPB (MLAP)
- obsługi podkanałów skanera (MPS).

Moduły MITRN, MŁ i MT biorą udział w obsłudze połączeń z komputera obliczeniowego do sieci. Pozostałe moduły biorą udział w obsłudze obu typów połączeń z komputera obliczeniowego do sieci i tranzytowych wewnątrz sieci. Przedstawiono to na rys. 4.



Rys.4. Przepływy informacji przez moduły części sieciowej węzła
Fig.4. Dataflow between modules of the node software

Procesy obsługiwane przez moduł MLAP i część modułu MITRN realizowane są na trzecim poziomie programowym EC8371.01, procesy obsługi podkanałów skanera na poziomie drugim i trzecim, a pozostałe na poziomie piątym.

Główne funkcje systemu zarządzającego w systemie obejmują zarządzanie zadaniami, zarządzanie kolejkami i buforami. Zarządzanie zadaniami obejmuje ich uaktywnianie zgodne z priorytetami, zmianę stanów oraz przekazywanie sterowania między programem a podprogramami.

Zarządzanie kolejkami polega na umieszczaniu nowych elementów na końcu oraz wybieraniu elementów do obsługi z początku kolejki. Natomiast zarządzanie buforami polega na: przydzielaniu i zwalnianiu buforów dla danych, ich dodawaniu i usuwaniu z kolejek buforów, umożliwieniu dostępu do określonych pozycji w kolejce oraz odzyskiwaniu buforów w stanach przytłumienia pracy sieci, kiedy to liczba wolnych buforów jest mniejsza od minimalnej liczby gwarantującej poprawną pracę EC8371.01. Wszystkie te elementy sterowania istnieją w standardowym programie NCP i zostały przejęte do realizacji części sieciowej oprogramowania.

Przyjęto też z programu NCP sposób organizacji łańcuchów buforów kolejek danych oraz kolejek zadań. W związku z tym tablice używane przez część sieciową, reprezentujące stany takich obiektów, jak stacje LAPB, sieciowe, transportowe, połączenia transportowe czy kanały logiczne mają budowę przystosowaną do wymagań programu zarządzającego (supervisora). Dotyczy to zwłaszcza umieszczenia w tych tablicach bloków QCB (blok sterowania kolejką) dla kolejek wejściowych, pseudowejściowych i roboczych. Dotyczy to także umieszczenia bloków ECB (blok sterowania zdarzeniem) w przypadku odmierzenia oczekiwania (time-out). Wprowadza się tu jednak zmianę polegającą na tym, że bloki ECB są umieszczane w tablicach opisujących stan obiektu, a nie w kolejkach przechowujących dane (BCU) jak w przypadku NCP.

W związku z wymaganiami części oprogramowania realizującego funkcje sieciowe dokonano kilku zmian w zarządzaniu buforami. Przyjęto, że obie części oprogramowania korzystają z tej samej puli buforów, które pobierane są i zwalniane za pomocą tych samych makroinstrukcji. Ze względu na wymagania oprogramowania sieciowego zmieniono minimalny rozmiar bufora z 44 na 152 bajty. Ponadto wprowadzono mechanizm pytania o stan puli wolnych buforów. Na potrzeby tego mechanizmu zdefiniowano progi obciążenia określające procent zajętych buforów puli. Takie pytanie pozwala na ustalenie czy przekroczono zadany próg. Dzięki temu jest możliwe podejmowanie decyzji o dalszym przebiegu procesu w zależności od liczby wolnych buforów.

LITERATURA

- [1] BUDZIANOWSKI L. i in., Projekt techniczny oprogramowania PTD jako procesora czołowego dla EC1032 i komputera węzła w sieci SKJS2 wersja 1. Założenia funkcjonalne, Inst.Cybern.Techn.PWr., Raport SPR 11/84, Wrocław 1984.
- [2] BUDZIANOWSKI L. i in., Projekt techniczny oprogramowania PTD jako procesora czołowego dla EC1032 i komputera węzła w sieci SKJS2 wersja 1. Opis funkcjonalny. Schematy blokowe. Inst.Cybern.Techn.PWr., Raport SPR 3/85, Wrocław 1985.
- [3] Information processing systems - Open systems interconnection - Connection oriented transport protocol specification, June 27, Copenhagen 1984, ISO/DIS 8073 rev.
- [4] YELLOW BOOK, Data communication networks services and facilities, terminal equipment and interfaces, Recommendation X.25, Vol. VII, Fascicle 2, CCITT, Geneva 1981,
- [5] YELLOW BOOK, Data communication networks transmission, singnaling and switching, network aspects, maintenance, administrative arrangements, Recommendation X.75 Vol. VIII, Fascicle 3, CCITT, Geneva 1981,
- [6] OSSOLA M., Routing principles for the eighties: are experience and theory sufficient to settle for distributed and adaptive routing ?, Performance of data communication system and their applications, Pub. Company, North Holland 1981.

Praca wpłynęła do Redakcji 1988.03.28

LAYER STRUCTURE AND STANDARDS IN COMPUTER NETWORK SKJS2

The subject of the paper is the logical structure of the Computer Network SKJS2 mark 1. Presented are layers, protocols and software solutions of the communication controller EC8371.01. Then the main elements of the routing algorithm and its properties are shown.

ЭТАЛОННАЯ СТРУКТУРА И СТАНДАРТЫ В ВЫЧИСЛИТЕЛЬНОЙ СЕТИ SKJS2

В статье представлена эталонная структура вычислительной сети SKJS2. В основной части статьи речь идет о эталонах, протоколах и программах компьютера узла EC8371.01. Затем представлены главные элементы алгоритма прокладки маршрута и краткая оценка принятых решений.

*Rozległa sieć komputerowa,
oprogramowanie bazowe sieci*

Andrzej KALIŚ*
Andrzej HUZAR*

TWORZENIE OPROGRAMOWANIA BAZOWEGO SIECI KOMPUTEROWEJ SKJS2 WERSJA 1

Przedstawiono poszczególne etapy tworzenia oprogramowania bazowego sieci komputerowej SKJS2 wersja 1. Omówiono kolejne fazy projektowania oprogramowania bazowego, poruszono problem efektywności prac programistycznych, a przede wszystkim skoncentrowano się na zagadnieniu testowania i uruchamiania dużego oprogramowania.

1. WSTĘP

W początkowym okresie rozwoju maszyn cyfrowych szczególną troską objęto część sprzętową. Obecnie znacznie większą uwagę zwraca się na tworzenie oprogramowania, czyli pracę programisty. Jeżeli w końcu lat pięćdziesiątych 90% kosztów maszyny cyfrowej przeznaczano na sprzęt, obecnie 90% przeznaczają się na rozwój oprogramowania. Zatem koszty wytworzenia oraz niezawodność oprogramowania (patrz [6]) to obecnie główne problemy ogólnie rozumianego przetwarzania z wykorzystaniem maszyn cyfrowych. Różnica efektywności programistów może być jak 1:5 lub nawet jak 1:100 [10]. Oznacza to, że programiści o jednakowym przygotowaniu i zbliżonym wynagrodzeniu mogą poświęcić na realizację tego samego projektu (oprogramowania) od jednej do stu jednostek czasu. Choćby z tego względu sensowne oszacowanie kosztów utworzenia oprogra-

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

owania (nie wspominając o planowaniu prac programistycznych) jest znacznie utrudnione. Nie wiem jak należy mierzyć wydajność prac podczas tworzenia oprogramowania czy jego projektowania. Jesteśmy skłonni mówić - oprogramowanie "żmudne", "bardzo żmudne", "duże" lub "ogromne". Oczywiście tego rodzaju sformułowania nie mogą być pomocne. Dla pomiaru i ilościowego określenia wydajności prac programistycznych, rozumianej jako szybkość wykonania programów całkowicie gotowych do wykonania, nie istnieje żadna dobra metoda, ponieważ [6]:

- trudno jest oszacować jak duży jest zakres prac
- nie potrafimy oszacować wydajności zespołu
- zazwyczaj nie jesteśmy w stanie stwierdzić, jaką część prac już wykonaliśmy
- trudno powiedzieć jak daleko się posunęliśmy w tworzeniu oprogramowania.

Można w tym miejscu zapytać za autorem pracy[6] - Czy znajdzie się człowiek, który zajmie się prowadzeniem prac programistycznych przy tylu niewiadomych? Jest to jeden z kluczowych problemów każdego rozpoczynającego się przedsięwzięcia programistycznego. Jak to się jednak dzieje, że pomimo tylu niewiadomych, spora część prac zostaje ukończona w przyjętym terminie.

Problem dotrzymywania terminu jest jednym z najtrudniejszych aspektów zarządzania pracami programistycznymi. Prace programistyczne są zazwyczaj niepowtarzalne. Tworzenie oprogramowania to autentyczna praca artystyczna. Tutaj właśnie nabiera znaczenia podstawowa różnica między programowaniem a jakąkolwiek działalnością konstrukcyjną. Z praktyki wynika jednak, że doświadczone zespoły programistów-projektantów, których częścią doświadczenia jest umiejętność oceny stopnia trudności projektu, dostarczają dosyć złożone oprogramowanie dokładnie na czas (zazwyczaj mnożąc tę ocenę przez współczynnik bezpieczeństwa większy od 1) [11].

Opracowywanie oprogramowania to wieloraka działalność nie tylko związana z pracą na maszynie cyfrowej. Tworzenie dużego oprogramowania jest przedsięwzięciem kolektywnym, czyli istotne są problemy organizacji pracy grupy programistów, warunków ich pracy, administrowania nimi, czy wreszcie wpływu czynników osobowościowych poszczególnych członków grupy na pracę całego zespołu i jego powiązań z innymi zespołami. Kierowanie to twórczość, którą trudno opanować. Zazwyczaj główne źródła trudności są odzwierciedleniem złej struktury organizacyjnej [10].

Tworzenie bazowego oprogramowania sieci komputerowej jest dosyć złożone i kosztowne. Złożoność jest konsekwencją naszkicowanych otwartych problemów przedsięwzięcia programistycznego, co niewątpliwie od-

bija się na kosztach. Nie bez znaczenia są także koszty czasu maszynowego kilku zestawów komputerowych.

W artykule niniejszym przedstawiono poszczególne fazy tworzenia bazowego oprogramowania sieci komputerowej SKJS2 wersja 1. Potrzeba retrospektywnego spojrzenia na proces projektowo-implementacyjny wynika z braku podobnych polskich opracowań. Ponowne spojrzenie na organizacyjną stronę przedsięwzięcia oraz podjęcie próby jego oceny wydaje się pożyteczne.

Prace projektowe nad bazowym oprogramowaniem sieci SKJS2 wersja 1 rozpoczęły się na początku roku 1984. Umowa z Instytutem Komputerowych Systemów Automatyki i Pomiarów (IKSAiP) we Wrocławiu obejmowała kilka etapów - założenia, projekt, implementacje oraz testowanie i uruchomienie na instalacji eksperymentalnej (trzy komputery obliczeniowe - 2 x EC1032 oraz EC1055). Centrum Obliczeniowe Politechniki Wrocławskiej wraz z Instytutem Cybernetyki Technicznej wykonywało bazowe oprogramowanie sieciowe komputera obliczeniowego (TSO, stacja transferu zbiorów) oraz w całości oprogramowanie komputera czołowego i węzła [1], [5]. Zespoły IKSAiP-u realizowały prostą wersję poczty elektronicznej, narzędzia do pomiaru przepustowości sieci oraz przystosowały podsystem SKOT do pracy sieciowej.

Bazowe oprogramowanie sieci komputerowej nie jest oprogramowaniem "zwartym", tzn. wykonanym na jedną autonomicznie pracującą maszynę cyfrową. Jest to kilka części programowych, umiejscowionych w różnych maszynach, które mają ze sobą współpracować. Jeżeli oprogramowanie jednej z maszyn sieci ulega załamaniu (awarie sprzętu lub inne przyczyny), to nie może to spowodować upadku całej sieci czy innych jej komponentów. Dodatkowo w każdej maszynie oprogramowanie jest podzielone. Między poszczególnymi jego częściami istnieją logiczne więzi. Również tu upadek jednej części oprogramowania nie może wpływać na pozostałe. Tym samym jawią się nam tutaj dodatkowe problemy organizacyjne o większej skali złożoności niż przy tworzeniu oprogramowania zwartego. Przede wszystkim dużej staranności wymagała organizacja współpracy między zespołami programistów.

2. PROJEKTOWANIE OPROGRAMOWANIA

2.1. Założenia i wymagania

Początek każdego przedsięwzięcia programistycznego rozpoczyna się od sformułowania założeń techniczno-ekonomicznych. Jest to pierwsza faza uzgadniania warunków między zamawiającym (użytkownikiem),

a wykonawcą. Stopień ważności tej fazy jest stosunkowo duży. Warto przypomnieć za pracą [8], że jeżeli twórca oprogramowania ma nikłe lub nie ma żadnego pojęcia o sensownych oczekiwaniach użytkownika, to całe przedsięwzięcie zakończy się niepowodzeniem. Zdarzają się projektanci systemów zarządzania bazą danych, którzy nigdy nie wdrożyli tego rodzaju systemu. Powoduje to znaczne błędy oprogramowania. Wydaje się, że w naszym przypadku zamawiający nie żywił obaw wynikających z tych spostrzeżeń. Należy nawiązać taką współpracę z użytkownikiem, aby mógł on w sposób istotny wpływać na decyzje podejmowane w fazie określania wymagań i celów oraz w fazie powstawania specyfikacji. Oczywiście nie należy jednak angażować użytkownika do podejmowania decyzji wykraczających poza jego kompetencje [8].

Odrębnym problemem jest udział użytkowników w budowie produktów nie zapowiadanych na rynku informatycznym. Prawdopodobnie do tej grupy należy tworzenie bazowego oprogramowania sieci komputerowej w Polsce. Mimo, że produkt nie jest adresowany do wyraźnie sprecyzowanego użytkownika, to jednak wykonawca (producent) oprogramowania powinien mieć dostateczne wyobrażenie o swych przyszłych klientach.

Przypomnijmy w tym miejscu wymagania postawione wykonawcy przez zamawiającego oprogramowanie bazowe sieci SKJS2 wersja 1:

- sprzęt wchodzący w skład sieci powinien być produkowany bądź kompletowany przez ZE ELWRO (dotyczy to EC1032, EC1034 oraz EC8371.01)
- oprogramowanie systemowe powinno być dostarczone przez ZE ELWRO (zwłaszcza system OS/MVT 5.01 JS)
- należy zachować metodę dostępu telekomunikacyjnego TCAM5 (komputer obliczeniowy) oraz NCP (komputer czołowy i węzła)
- użytkownik powinien mieć możliwość korzystania z podsystemów TSO i SKOT oraz przysyłać zbiory sekwencyjne
- nie należy wyodrębniać osobnej podsieci komunikacyjnej.

Podane wymagania stanowiły podstawowe ograniczenia w realizacji. W znacznym stopniu ograniczyły one pole manewru na etapie projektu całości oprogramowania oraz w znacznym stopniu wpłynęły na architekturę sieci.

2.2. Organizacja zespołów programistów

Dobrze zorganizowany proces projektowy sprowadza się do coraz bardziej utrwalonego przekonania, że programiści powinni pracować w zespołach [6],[11], gdyż wtedy łatwiej jest utrzymać ciągłość prac, jeżeli ktoś nieoczekiwanie opuści zespół. Istnieją poza tym jeszcze trzy inne podstawowe racje przemawiające na rzecz pracy zespołowej [11]:

a) pewne problemy są tak trudne, że do ich rozwiązania jest potrzebne rozwiązanie wielu podproblemów;

b) pracując zespołowo można rozwiązać cały problem w znacznie krótszym czasie dzięki jednoczesnemu rozwiązywaniu tych zagadnień;

c) niektóre problemy wymagają ścisłej współpracy wyspecjalizowanych programistów; opanowanie wszystkich potrzebnych umiejętności może wymagać zbyt wiele czasu [7].

Najczęstszymi wariantami organizacji zespołów programistów są:

- zespół głównego programisty [2,6]
- zespół specjalistów [4]
- zespół demokratyczny [12].

Nie wdając się w omawianie poszczególnych wariantów (patrz literatura), naszkicujemy podział na zespoły pracowników Centrum Obliczeniowego, uczestniczących w pracach nad siecią. Otóż, jak już wspomnieliśmy, w momencie przystąpienia do realizacji umowy pracownicy Centrum byli zaangażowani w końcową fazę prac nad siecią MSK, a więc były już utworzone i sprawdzone zespoły pracowników realizujących wyznaczone cele. Wyróżnić można było zespół zajmujący się tylko oprogramowaniem komputera obliczeniowego, komputera czołowego oraz podsieci komunikacyjnej. Przystępując do prac nad siecią SKOS2 waroja 1, utworzyliśmy tylko dwa zespoły komputera obliczeniowego oraz jeden zespół komputera czołowego i węzła. Podstawę zespołów komputera obliczeniowego stanowili członkowie zespołu realizującego stację transferu zbiorów w sieci MSK. Jeden z tych zespołów zajmował się w dalszym ciągu tylko stacją transferu zbiorów, drugi natomiast tworzeniem oprogramowania umożliwiającego podłączenie aplikacji do sieci. Zespół zajmujący się oprogramowaniem komputera czołowego w sieci MSK został powiększony o zespół realizujący oprogramowanie komputera czołowego dla Odry 1305 sieci MSK. Zespół ten odpowiedzialny był przede wszystkim za oprogramowanie warstwy 4 i 3 modelu odniesienia ISO/OSI.

W każdym zespole dało się wyróżnić wiodącego specjalistę, który skupiał w swoich rękach wszystkie informacje o postępie prac pozostałych specjalistów. Wybierając wiodącego specjalistę przede wszystkim należy zwrócić uwagę na to, by wcześniej wykonywał już taką pracę. Takiego, który kierował pracami od początku do końca [6]. W naszym przypadku były to osoby kierujące pracami nad siecią MSK. Nie oznacza to, że organizacja zespołów była zgodna z drugim podanym wariantem. Raczej było to połączenie wariantu drugiego i trzeciego, ponieważ przydziały prac były wewnętrzną sprawą zespołu i zależały przede wszystkim od zdolności jego członków.

Klamrą spinającą prace obu zespołów były ogólne cotygodniowe posiedzenia sieciowe (nie licząc seminariów w zespołach), na których

podejmowano wiążące ustalenia. Były to spotkania bardziej problemowe niż robocze, chociaż relacjonowano również szczegóły. Spotkania były otwarte dla innych zespołów Centrum Obliczeniowego. Oceniając je z pewnej perspektywy czasu należy uznać, że było bardzo ważny element w całym okresie prac. Nie było to tylko forum, na którym wypracowywano merytoryczną integrację zespołów, lecz spełniało ono także rolę "koleżeńskej inspekcji"[6],[10]. Polegało to na tym, że poszczególni programiści przepytywani byli z tego, jak realizują swoje zadania. Oczywiście nie należy traktować przepytywania jako przesłuchania w negatywnym sensie tego słowa. Robiliśmy to z myślą o koleżeńskej pomocy, wysłuchania propozycji o być może lepszych (sprawdzonych) rozwiązaniach. Przepytywanie ma oprócz wymienionych, kolejne niemniej ważne zalety. Zespół realizujący zadanie (podzadanie) jest dokładnie zaznajamiany z tym co robi kolega i jak to robi (nieocenione w razie odejścia członka zespołu - taka sytuacja miała miejsce). Ważny jest także element poznawczy. Członek zespołu uczy się od innych członków grupy przyjmując jego zdaniem lepsze metody programowania, możliwości nowych języków, poznaje niuanse rozwiązywanego tematu ("programista powinien mieć możliwość i chęć uczenia się"[10]). Decyzje podejmowane na spotkaniach były wiążące dla wszystkich biorących udział w pracach.

Termin tego spotkania wyznaczono w dniu, w którym dokonywana była cotygodniowa konserwacja sprzętu komputerowego ("nie ma dostępu do maszyny - mogę poświęcić czas na dyskusję"). Jest to niezmiernie ważny czynnik warunkujący "swobodną" atmosferę na spotkaniach programistów.

2.3. Planowanie prac

Cykl życia oprogramowania możemy podzielić na trzy części: opracowanie, wykorzystanie oraz przedłużone opracowywanie. Zazwyczaj dwie ostatnie części zachodzą na siebie. W okresie opracowywania musimy określić wymagania, zaprojektować oprogramowanie, napisać programy, skalic je, przetestować i zweryfikować, a następnie zdokumentować. W okresie przedłużonego opracowywania oprogramowanie ulega rozszerzeniu o nowe funkcje i modyfikacje starych, może ulec wymianie sprzęt oraz poprawia się błędy.

W okresie opracowywania istotne są następujące elementy składowe procesu, a więc: czas, kadry, narzędzia oraz pieniądze. W okresie wykorzystania na pierwszy plan wysuwają się realizowane funkcje, efektywność i szybkość, zasoby, łatwość wykorzystania oraz brak błędów. Natomiast w okresie przedłużonego opracowywania istotne są: dokumentacja użytkowa, modyfikowalność, brak błędów, pieniądze, kadry, czas oraz narzędzia.

Bardzo ważnym okresem cyklu życia oprogramowania jest okres przedłużonego opracowywania, niestety, źle widziany i rozumiany przez polskiego użytkownika (zamawiającego), przede wszystkim ze względów finansowych. Powołując się na problem Hume'a [6] należy wyciągnąć następujący wniosek - instytucje produkujące oprogramowanie powinny mieć zapewnione finansowe wsparcie, pozwalające im pielęgnować oraz - co istotniejsze - rozwijać już eksploatowane oprogramowanie systemowe.

Niestety utarta praktyka zawierania umów w dotychczasowym ich kształcie, nie pozwala zespołom programistów na kontynuowanie pracy. System finansowy wyznacza sztywne ramy czasowe przeznaczone na pracę ("od odbioru do odbioru"). Po sprzedaniu pracy zajmujemy się już innym problemem, "zapominając" co zrobiliśmy dotychczas.

Dlatego mimo wszystko, przystępując do realizacji zamówienia na utworzenie dużego systemu należy zawczasu w kalkulować możliwość jego przyszłych modyfikacji i wzbogacania, np. o dodatkowe funkcje i usługi. Okazuje się, że ok. 25-75% prac programistycznych polega na modyfikacji i ulepszaniu istniejącego oprogramowania [10]. Poza tym, prawie zawsze po pewnym czasie eksploatacji oprogramowania zachodzi konieczność wprowadzenia pewnych zmian. Wystarczy, że zamawiający nieprecyzyjnie sformułował swoje wymagania. Oczywiście powinniśmy ustrzec się od tego rodzaju sytuacji, jednak nie ustrzeżemy się od ciągłych zgłoszeń dodatkowych wymagań od zamawiającego. Ponadto źródłem zmian mogą być sugerowane poprawki bądź wykrycie błędu. Dlatego doświadczony programista zdaje sobie z tego sprawę, projektuje zatem oprogramowanie w taki sposób, aby zmiany nie były wstrząsem i nie wpływały na niezawodność produktu. Jednocześnie prostota modyfikacji jest bardzo ważną charakterystyką jakości programu. Szybkość rozwiązania zadania modyfikacji może być wykorzystana jako miara rozumienia programu [10].

Przedstawiony problem możemy rozwiązać, jeżeli będziemy starali się kierować tworzeniem oprogramowania mając na uwadze następujące zalecenia [6]:

1. Należy projektować nasze oprogramowanie tak, aby było ono w maksymalnym stopniu zmodularyzowane.

2. Moduły należy pogrupować tak, aby zminimalizować wzajemne oddziaływania na siebie.

3. Należy wykorzystywać tablicową metodę sterowania programem. Zmieniając wiersz tablicy, zmieniamy algorytm.

4. Koniecznie należy ustanowić pewien "standard" programowania i wymagać jego przestrzegania.

5. Należy zorganizować ścisłą i szczegółową kontrolę.

6. Zawczasu należy zaplanować przesunięcie niektórych członków zespołu do prac nad rozpracowywaniem kolejnych wariantów i wersji oprogramowania.

7. Należy zaplanować przechowanie środowiska testującego.

8. Podane zalecenia należy uwzględnić w kosztach przedsięwzięcia.

W sieci SKJS2 wersja 1 planowanie prac było niezwykle utrudnione. Wynikało to zwłaszcza z tego, że sformułowane wymagania własności sieci mówiły o tym, iż dotychczasowe aplikacje (TSO, SKOT) mają być aplikacjami w sieci. To wymaganie połączone z koniecznością wykorzystania telekomunikacyjnej metody dostępu TCAM5 ograniczało możliwości projektantów struktury oprogramowania. Problemem nie było znalezienie efektywnej modularnej struktury, lecz takiej, której realizacja spełniałaby wymagane funkcje. Wymagało to więc gruntownego zapoznania się z własnościami tych aplikacji pod kątem ich użytkowania w sieci. Ponieważ (jak wykazują badania [14] - patrz tab. 1) koszty usunięcia błędów powstałych na etapie projektu w momencie, gdy już są zaawansowane dalsze prace tworzenia oprogramowania szybko rosną, ten etap został zakończony dopiero, gdy istniała pewność co do jego poprawności.

T a b e l a 1

Relatywny koszt wykrycia i usunięcia błędów

Etap	Koszt usunięcia błędu
Projekt	1
Testowanie modułów	4
Testowanie funkcji systemu	5
Testowanie całego systemu	15
Eksploatacja	30

Końcowy projekt struktury oprogramowania powstał dopiero wtedy, gdy zostały sprawdzone wszystkie te elementy, które uważano za krytyczne (własności środowiska - TCAM i system operacyjny). Cóż bowiem zrobić, gdy na przykład program MCP nie działa zgodnie z opisem, który nie jest precyzyjny? Ze względu na brak źródeł tego oprogramowania, szczupłość zespołów i brak czasu, nie wchodziło w rachubę poprawianie oprogramowania systemowego. Stąd też dopiero po upływie niemal 2/3 czasu realizacji oprogramowania powstał ostateczny projekt jego struktury. Oczywiście dysponowano też bogatą biblioteką pod-

programów potrzebnych do spełnienia wszystkich funkcji realizowanych przez środowisko. Również na etapie projektu zwrócono uwagę na te elementy, które umożliwią testowanie i uruchamianie oprogramowania (zmieniono początkowo niesymetryczny protokół wymiany komunikatów między komputerem obliczeniowym a czołowym na symetryczny, po to, by można było oprogramowanie komputera obliczeniowego uruchamiać i testować bez użycia komputera czołowego - wystarczyło zawracać komunikaty w komputerze obliczeniowym).

3. EFEKTYWNOŚĆ PRAC PROGRAMISTYCZNYCH

Wraz z rozwojem komputerów termin "efektywność prac programistycznych" stał się bardzo niejednoznaczny. Można wyróżnić jednak dwie różne miary efektywności - szybkość rozwiązywania zadania oraz czas oczekiwania na rozwiązanie. Dlatego jeżeli mówimy o efektywności, to należy jasno sobie powiedzieć co będziemy (i jak) mierzyć. Niestety nie wiemy jednak, jak należy mierzyć efektywność prac podczas programowania czy projektowania oprogramowania. Nierozzerwalnie z efektywnością prac wiąże się organizacja prac programistycznych (zespoły programistów). Każdy odpowiedzialny za przedsięwzięcie programistyczne powinien wiedzieć, na jakim etapie jest jego praca. Przy czym wyróżnić można tutaj następujące etapy: euforia, rozczerowanie, szukanie winnych, karanie niewinnych oraz wynagradzanie nieuczestniczących w pracach (?). Nie należy powyższych etapów traktować w sposób tylko humorystyczny! Doświadczenie uczy, że szukanie winnych, w konsekwencji którego zamieniamy pierwszego odpowiedzialnego drugim odpowiedzialnym, zazwyczaj kończy się w identyczny sposób. Odpowiedzialnego zmienić łatwo, jednak nie zawsze decyzja ta prowadzi do dobrych wyników. Przede wszystkim ujemnie wpływa ona na wszystkich pozostałych współpracowników ("stało się to z nimi, może zdarzyć się i mnie"). Wiadomo, że zestresowany programista to zły programista.

Dobry administrator powinien być dostatecznie wymagający, aby zapewnić właściwą intensywność prac, ale i dostatecznie sympatyczny, by nie poróżnić pracowników. Administrator powinien być kompetentny, lecz bardziej konieczną jego cechą jest przenikliwość ("dobry programista nie musi być dobrym kierownikiem").

Nagrodą za dużą efektywność powinny być premie i progresywne wypłaty, przy czym żaden administrator nie może liczyć na więcej niż entuzjazm podległych mu pracowników.

Bardzo ważną rolę z punktu widzenia efektywności odgrywają warunki pracy programisty [9,13] .

W przypadku sieci SKJS2 wersja 1 była realizowana następująca zasada: "Nikt nigdy nie może pozostawać bez zajęcia, każdy ma zawsze coś jeszcze do wykonania". Stosowanie takiej zasady nie jest łatwe, ponieważ członkowie zespołu pracują nierównomiernie. Trudno było (na szczęście!) ocenić, które z prac są bardziej pracochłonne, a które mniej. Ze względu na dużą uniwersalność członków zespołu można było wymagać od nich wykonania coraz to nowych prac. Szczególnie ważne było poczucie współodpowiedzialności każdego z członków zespołów za całość prac.

4. TESTOWANIE

Testowaniem nazwiemy działalność, której celem jest pokazanie, że oprogramowanie realizuje to, co założono. Jednym z charakterystycznych aspektów testowania jest indukcyjne wnioskowanie na podstawie dobranego, małego zbioru zachowań, że program zachowa się tak jak powinien. Indukcyjne, ponieważ związane jest z problemem - żadna liczba przypadków potwierdzających poprawność nie może całkowicie wykluczyć możliwości, iż istnieje nieprzetestowany przypadek podważający tę poprawność [3]. Pojawia się więc zamysł dowiedzenia, iż program zawsze zachowa się tak jak powinien przyjąwszy, że dowód jest przekonującym i trudnym do obalenia argumentem. Jednak dowód matematyczny jest poprawny lub niepoprawny i trudno jest określić, czy poprawny jest dany konkretny dowód. Stąd najczęstszym sposobem obalenia dowodu (twierdzenia) jest znalezienie kontrprzykładu [3].

Najlepszym testowaniem dowolnego oprogramowania jest jego normalna eksploatacja. Szczególnie odnosi się to do systemów czasu rzeczywistego. Jednak zanim to nastąpi, zazwyczaj pragniemy znaleźć błędy na najwcześniejszym etapie realizacji projektu (powiemy o tym w następnym rozdziale).

Przed uświadomieniem sobie na czym polega działalność grupy testującej, należy przypomnieć, że grupa ta naprawdę powinna być zainteresowana wykryciem każdego błędu [8] ("nie powinna usiłować wykazać poprawności oprogramowania"), zgodnie zresztą z pierwszą zasadą testowania, która mówi - testowanie jest to proces wykonywania programu (lub jego części) w celu i z intencją wykrycia błędów.

Na czym polega działalność grupy testującej? Otóż polega ona na [6]:

- rozumieniu funkcji, które powinny być realizowane przez system
- wykryciu "wąskich gardeł" systemu: ograniczeń czasowych oraz niestabilności wynikającej z parametrów wejściowych
- zaprojektowaniu kolejności testów ("ćwiczeń")
- określeniu w testowanym systemie zarówno sprzętowych, jak i programowych składowych
- zbudowaniu systemu testującego
- planowaniu rzeczywistego przebiegu testowania oraz umiejętnym nim kierowanie.

Ponadto dobrze jest uświadomić sobie następujące stwierdzenia urastające do rangi postulatów:

- należy zadbać o to, by testowalność była kluczowym założeniem projektu oprogramowania
- testowanie musi rozpocząć się od ustalenia celów (podobnie jak każda działalność)
- nie można testować własnego programu ("daj program wariatowi, bo w tym względzie wariat jest genialny")
- nigdy nie zmieniaj programu po to, aby ułatwić testowanie.

W literaturze przedmiotu precyzuje się i wprowadza nazewnictwo, uściślając rozumienie poszczególnych faz procesu testowania. Omówimy je krótko. Weryfikacja nazwiemy próbę wykrycia błędów przez wykonanie programu w roboczym środowisku, natomiast konfirmacja analogiczną próbę, ale wykonaną w danym, rzeczywistym środowisku. Natomiast uruchamianie docieka dokładnych przyczyn znanych już błędów i usiłuje je poprawić.

Jak zapewne będzie wynikać z następnego rozdziału, tworząc oprogramowanie bazowe sieci SKJS2 wersja 1, zrealizowaliśmy wszystkie wymienione fazy procesu testowania.

Jedną z uznanych metod testowania, zapewniających rozwiązanie kluczowego problemu w "końcowej" fazie procesu testowania, jest zaangażowanie użytkowników do testowania. W przypadku sieci SKJS2 wersja 1 pierwszym uruchomionym podsystemem było TSO, które poddawaliśmy testowi poligonowemu, używając go po prostu jako narzędzia do budowy programów użytkowych. Należy zatem postępować w myśl zasady - używaj własnego produktu jako narzędzia, zanim przystąpią do tego inni [8].

5. TESTOWANIE OPROGRAMOWANIA BAZOWEGO SIECI SKJS2 WERSJA 1

Wspomnieliśmy już, że oprogramowywaniem sieci SKJS2 wersja 1 zajmowały się zespoły raczej doświadczonych programistów i projektan-

tów. Doświadczenie to dotyczyło przede wszystkim merytorycznej strony zagadnień związanych z sieciowymi zaleceniami standaryzacyjnymi (protokoły, usługi) oraz wycuciem zagrożeń, jakie niosą duże przedsięwzięcia programistyczne. Niestety oprogramowanie systemowe, na którym oparliśmy oprogramowanie bazowe sieci, zawsze sprawiało nam niespodzianki pomimo kilkuletniej z nim współpracy. Z punktu widzenia psychologicznego sytuacja ta bardzo ostro wyczuliła nas na problemy testowania już na najbardziej elementarnym poziomie (np. sprawdzenie czy to co napisano w dokumentacji oprogramowania systemu operacyjnego jest prawdą?!). Jeżeli ktoś nie odszedł z zespołu i wytrwał do końca prac - znaczy to, że sprawdzał wszystko! Kto próbował od razu przejść do fazy confirmacji, ignorując ważność fazy weryfikacji, ten narażony był na duże stresy z własnej winy. Poza tym pracownicy ci nigdy nie mogli udokumentować weryfikowalności założeń realizacyjnych swoich modułów programowych, czyli po scaleniu i uruchomieniu oprogramowania nastroczali oni wielu kłopotów podczas prób usuwania przyczyn znanych już błędów. Omawiana sytuacja występuje wtedy, gdy każdy członek zespołu twierdzi, że błąd nie jest związany z jego modułem. Jeżeli każdy moduł przeszedł fazę weryfikacji, to z praktycznego punktu widzenia pozostaje tylko analiza więzi między-modułowej. Nawiasem mówiąc, jeżeli opisana sytuacja występuje, po prostu oznacza to, że nie przywiązywano szczególnej uwagi do testowania jako kluczowego założenia projektu oprogramowania ("błąd powinien wskazywać winnego").

Za pracą [8] można proces testowania usystematyzować, dzieląc go na następujące fazy:

- testowanie modułu jest weryfikacją pojedynczego modułu programu zazwyczaj w środowisku wyodrębnionym od innych modułów
- testowanie integracyjne jest weryfikacją pośrednictw między częściami systemu
- testowanie funkcji zewnętrznych polega na weryfikacji zewnętrznych funkcji systemu, określonych w specyfikacjach zewnętrznych
- testowanie systemu jest weryfikacją i (lub) confirmacją systemu względem jego celów początkowych
- testowanie akceptacyjne jest confirmacją systemu lub programu względem wymagań użytkownika
- testowanie instalacyjne jest confirmacją każdej konkretnej instalacji systemu w celu wykrycia błędów popełnionych podczas instalowania systemu.

Najpierw zaczęliśmy testować program MCP, a zwłaszcza tę jego opcję, która umożliwia przekazywanie komunikatu z jednego programu

do innego (łączność program - program). Przykładowe i w praktyce najczęściej stosowane opcje dotyczyły łączności program-terminal. Ponadto wykorzystana w sieci metoda dostępu telekomunikacyjnego (realizowana przez program MCP) była z poziomu 5, a więc inna od rozpowszechnionej dotychczas metody TCAM poziomu 4. Wykonano wiele generacji programu MCP, sprawdzając jego opcje na prostych programach testowych komunikujących się ze sobą oraz z terminalami (na początek lokalnymi).

Program MŁOT, realizujący bardzo bogaty zestaw funkcji, został napisany w języku Pascal 360. Postąpiono tak, ponieważ łatwiej można modyfikować i uruchamiać program napisany w języku wysokiego poziomu przewidując, że w trakcie testowania funkcji zewnętrznych oraz systemu wystąpią dodatkowe wymagania na realizowane funkcje (co oczywiście miało miejsce). Wybór języka Pascal 360, oprócz wymienionych powodów, został podyktowany jeszcze tym, że powszechnie był wykorzystywany w innych pracach, a więc jego środowisko było w wystarczającym stopniu rozpoznane oraz jest to bardzo przejrzysty i zrozumiały język (w znacznym stopniu przyjęcie tego języka spełniało postulat łatwej modyfikowalności oprogramowania). Pozostałą część oprogramowania bazowego, już znacznie mniejszą, zakodowano w języku Assembler 360.

Standardowe operacje wejścia/wyjścia dostępne w języku Pascal nie obejmują telekomunikacyjnej metody dostępu (ma ją tylko PL 1 optymalizujący). Należało zatem dobudować do środowiska Pascal operacje wejścia-wyjścia, umożliwiające kontaktowanie się z programem MCP (wysyłanie i odbieranie komunikatów).

Szkieletowe oprogramowanie modułu MŁOT przetestowano w środowisku identycznym, jakie stworzyliśmy do przetestowania opcji MCP (program MŁOT-program, program-MŁOT-terminal). Następnie testowaniu poddano program przesyłający i odbierający dane do/od oprogramowania węzła. Na początku program ten nie kontaktował się bezpośrednio z oprogramowaniem węzła, a jedynie emulował jego obecność ("zawracanie" na styku z EC8371.01). Następnie powtórzono omówiony proces testowania. Łatwo zauważyć, że taka kolejność weryfikacji niejako w naturalny sposób wskazywała źródło błędów - było nim dołączone oprogramowanie bądź realizacja styku między modułami.

Jednocześnie prowadzono prace nad przystosowaniem podsystemu TSO do pracy w sieci. Wystarczyło potraktować podsystem TSO jako jeszcze jeden program współpracujący z programem MCP, by wykorzystując już istniejące środowisko testujące praktycznie "bez problemów", udostępnić TSO do pracy użytkowej (analogicznie postąpiliśmy z podsystemem SKOT, który przystosowywany był poza Politechniką Wrocławską i został włączony bez większych problemów).

Udostępniając podsystem TSO do pracy z terminali lokalnych, zaczęto łączyć się z oprogramowaniem NCP w celu udostępnienia terminali zdalnych, fizycznie podłączonych do EC8371.01. Na tak przygotowanym oprogramowaniu (TSO, MCP, MŁOT oraz NCP) zaczęto oprogramowywać stację transferu zbiorów (STZ), prostą pocztę elektroniczną oraz oprogramowanie węzła.

Stację transferu zbiorów weryfikowano w sztucznym środowisku (testowanie poziomu 0 oraz 1 protokołu NIFTP Blue Book). Następnie przeprowadzono confirmację, ale jeszcze z zawracaniem komunikatów "za" programem MŁOT - czyli nadal testowano program MCP oraz MŁOT ale w warunkach zwiększonych przepływów danych.

Oprogramowanie węzła oraz innych modułów umieszczonych w EC8371.01 powstawało w komputerze obliczeniowym. Weryfikowano je najpierw na programowym emulatorze procesora EC8371.01, który specjalnie w tym celu zaprojektowano i zaimplementowano. Tak zweryfikowane oprogramowanie wprowadzano (ładowano) do komputera EC8371.01 i poddawano confirmacji. Założono, że najpierw będzie testowane oprogramowanie odpowiedzialne za styk fizyczny i logiczny (sieciowa metoda dostępu) między komputerem obliczeniowym a oprogramowaniem węzła w EC8371.01. Zrealizowano to "zawracając" strumień komunikatów z oprogramowaniem sieciowej metody dostępu. Do sprawdzenia tego styku wystarczyło wykorzystać już istniejące środowisko testujące - STZ, TSO, MCP, MŁOT oraz terminale.

W tym momencie spotkała nas bardzo przykra niespodzianka. W środowisku tym, jak łatwo zauważyć, między oprogramowaniem komputera obliczeniowego a oprogramowaniem komputera czołowego i węzła istnieje łączność fizyczna przez dwa wydzielone podkanały multiplekserowe. W takiej konfiguracji istniejące firmowe oprogramowanie nigdy nie było generowane. Okazało się, że podkanał obsługujący łączność z oprogramowaniem węzła ulega trwałemu zawieszeniu (stan trwałej zajętości), co wstrzymywało przepływ danych przez ten podkanał. Okazało się, że produkowana wersja EC8371.01 wykonywana jest niezgodnie z dokumentacją. Po wprowadzeniu niewielkiej zmiany sprzętowej w EC8371.01, podkanał działał poprawnie. Kolejne zawracanie strumienia komunikatów nastąpiło na wyjściach skanerowych (zapętlenie na modemach). Powtórzono dotychczasowe testy. W warunkach prawie "rzeczywistych" intensywnie przetestowano oprogramowanie (sesja STZ, otwarte sesje użytkowników TSO).

Na tym etapie prac można było przenieść oprogramowanie na instalację eksperymentalną sieci, składającą się z trzech zestawów komputerowych (tyle udostępnił zamawiający) i przetestować oprogramowanie węzła w zakresie współpracy z innymi węzłami oraz zasymulować "upadki"

wybranych zestawów komputerów. Można było wreszcie przystąpić do pomiarów sieci i oceny jej parametrów czasowych.

Po przeniesieniu oprogramowania bazowego na instalację eksperymentalną rozpoczęły się próby i badania poczty elektronicznej, którą wykonywały zespoły spoza Politechniki Wrocławskiej. Oprócz niewielkich korekt w rozumieniu styku logicznego między oprogramowaniem poczty a programem MŁOT, oprogramowanie to zostało uruchomione praktycznie od razu.

Krótkie przypomnienie poszczególnych etapów tworzenia oprogramowania bazowego sieci SKJS2 wersja 1 wskazuje, że w zasadzie zgodne one były z ogólnie zalecanymi w metodologii programowania zasadami testowania. Nie próbowano uruchomić całości od razu, a w przemyśłany sposób włączano poszczególne części oprogramowania, osiągając w stosunkowo krótkim czasie wyznaczony cel ("oprogramowanie to nie cel, lecz środek do celu").

6. OCENA WŁASNOŚCI OPROGRAMOWANIA

Ponieważ oprogramowanie sieci SKJS2 wersja 1 należy do oprogramowania systemowego, należy dbać o to, by spełniało ono wiele różnorodnych wymagań [11]. W przypadku tworzenia tego oprogramowania wymagania te były spełniane mniej lub bardziej świadomie. Warto jednak skonfrontować w jakim stopniu oprogramowanie sieci spełnia najważniejsze z nich:

1. Zgodność z założoną specyfikacją (poprawność). Formalnej specyfikacji sieci nie było. Zamawiający określił jedynie funkcje sieci, które zostały spełnione. Nie wydaje się zresztą możliwe podanie formalnej specyfikacji środowiska, w którym działa sieć.
2. Niezawodność obejmująca odporność systemu na błędy własne, środowiska, sprzętu i użytkownika. Oprogramowanie jest odporne na część błędów. Odporne jest zwłaszcza na wszystkie błędy użytkownika, na niektóre awarie sprzętu (możliwość tworzenia drogi zastępczej w połączeniach sieciowych w przypadku fizycznego połączenia między węzłami). Nie jest natomiast odporne na błędy systemowe oraz większość błędów własnych.
3. Wydajność w sensie oszczędnego wykorzystania zasobów takich, jak: czas, pamięć, urządzenia zewnętrzne. Oprogramowanie sieci działa szybciej niż oprogramowanie o takich samych funkcjach systemu dotychczasowego (np. TSO w pracy lokalnej). Wielkość zajmowanej pamięci jest o około jedną trzecią większa niż w dotychczasowych systemach

teleprzetwarzania. Natomiast są zajmowane urządzenia zewnętrzne, które są obsługiwane przez sieć.

4. Rozszerzalność, tzn. możliwość rozszerzania systemu w przyszłości bez konieczności dokonywania zbyt wielu zmian w istniejącej części. Ten postulat jest spełniony bardzo dobrze. Można bez większych problemów zwiększyć liczbę aplikacji sieciowych, typów obsługiwanych terminali, przenieść funkcje węzła z procesora czołowego do procesora pełniącego tylko funkcje węzła. Można też rozdzielić między dwa komputery czołowe funkcje węzła i procesora obsługującego terminale.

5. Przenoszalność, tzn. możliwość łatwej adaptacji systemu do różnych konfiguracji systemowo-sprzętowych.

6. Łatwość pielęgnacji, tzn. podatność systemu na wprowadzanie niewielkich zmian bez nadmiernego ryzyka wprowadzenia błędu. Ten postulat jest spełniony dzięki użyciu języka Pascal do tworzenia oprogramowania komputera obliczeniowego. Podczas uruchamiania czas znalezienia przyczyny zauważonego błędu i jego usunięcie był często krótszy niż 0,5 godziny. Ze względu na brak dobrego języka do programowania komputera czołowego, szukanie i poprawianie błędów było tam bardziej czasochłonne i skomplikowane.

7. Odzyskiwalność, tzn. możliwość użycia modułów systemu w innych systemach lub nowych generacjach tego samego systemu. Wydaje się, że spełnienie tego postulatu zależy przede wszystkim od charakteru zmian w nowym systemie. Twórcy gwarantują działanie oprogramowania tylko w systemie OS/JS 5.01. W innych systemach typu OS/JS oprogramowanie sieciowe będzie przypuszczalnie musiało być częściowo zmodyfikowane lub przynajmniej wygenerowane od początku.

8. Odporność na włamania (odporność przed niepożądanym użyciem). Ze względu na przeznaczenie sieci oraz własności systemu OS/JS ta cecha nie jest spełniona. Zachowano jednak te zabezpieczenia, które mają aplikacje sieciowe (np. TSO wymaga znajomości nazwy użytkownika i hasła). Istnieją jednak możliwości łatwej modyfikacji oprogramowania, by oprogramowanie akceptowało jedynie zarejestrowanych użytkowników.

9. Zgodność ze standardami.

10. Łatwość użycia, tzn. prostota koncepcyjna systemu umożliwiająca szybkie jego opanowanie oraz pomoc udzielana użytkownikom w trakcie pracy. Korzystanie z sieci jest bardzo proste i niemal nie wymaga szkolenia (trzeba natomiast nauczyć się korzystania z aplikacji sieciowych). System udziela prostej pomocy użytkownikom w razie nawiązywania i rozłączania połączeń.

11. Dobra dokumentacja będąca podstawą do modyfikacji systemu przez producenta, a także do napisania odpowiednich podręczników dla różnych użytkowników. Dokumentacja techniczna była robiona tak, by autorzy mogli z niej w przyszłości skorzystać bez problemu (samodokumentujące się programy). Natomiast autorzy sporządzili osobno dokumentację użytkową i eksploatacyjną z myślą o użytkownikach nie znających tajników oprogramowania.

Jak widać z przedstawionego bilansu, oprogramowanie sieci spełnia niemal wszystkie z przytoczonych postulatów. Niestety do dziś (marzec 1988) sieć nie jest eksploatowana w warunkach rzeczywistych i dlatego trudno o praktyczną weryfikację jej możliwości. Z racji rozproszenia oprogramowania należy sądzić, że jej wdrażanie stanie się źródłem kolejnych doświadczeń oraz umożliwi wykrycie i poprawienie dotychczas nie dostrzeżonych błędów, jak również jej własności użytkowych.

LITERATURA

- [1] Aneks do założeń techniczno-ekonomicznych na sieć komputerową SKJS2 wersja 1, CO PWr., Raport SPR 5/84, Wrocław 1984.
- [2] BAKER F.T., Chief programmer team management of production programming, IBM System J., Vol. 11, No. 1, 1972, s. 56-73.
- [3] BRADY J.M., Informatyka teoretyczna w ujęciu programistycznym, Warszawa, WNT, 1983.
- [4] BROOKS F.P., The mythical man-month: essays on software engineering, Reading, Massachusetts, Addison-Wesley, 1975.
- [5] BUDZIANOWSKI L. i in., Projekt techniczny oprogramowania PTD jako procesora czołowego dla EC1032 i komputera węzła w sieci SKJS2 wersja 1: Założenia funkcjonalne, Inst.Cybern.Techn.PWr., Raport SPR 11/84, Wrocław 1984.
- [6] FOX J.M., Software and its development, Prentice Hall Inc., 1982.
- [7] JONES C.B., Konstruowanie oprogramowania metodą systematyczną, Warszawa, WNT, 1984.
- [8] MYERS G.J., Projektowanie niezawodnego oprogramowania, Warszawa, WNT, 1980.
- [9] McCUE G. M., IBM Santa Teresa laboratory - Architectural design for program development, IBM System J., Vol. 17, No. 1, 1978, s. 4-25.
- [10] SHNEIDERMAN B., Software psychology. Human factors in computer and information systems, Winthrop Pub. Inc., Cambridge 1980.
- [11] TURSKI W.M., Metodologia programowania, Warszawa, WNT, 1982.
- [12] WEINBERG G.M., The psychology of computer programming, Van Nostrand Reinhold, New York 1971.
- [13] Van TASSEL D., Praktyka programowania, Warszawa, WNT, 1982.

- [14] HERZOG O., Formal Software Development Methods in Industrial Environments, in Formal Software Development: Combining Specification Methods, Proc. Workshop, Nyborg 1984.

Praca wpłynęła do Redakcji 1988.03.28

THE CREATION OF BASIC SOFTWARE OF THE NETWORK SKJS2 VERSION 1

The paper describes the particular parts of the software creation of the network SKJS2 version 1. One after the other, the phases of the project are given. The problems of effectivity of the programming works are considered. There is an attempt to answer the question: How to test, debug and start the big software?

ОБРАЗОВАНИЕ ОСНОВНОГО МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ SKJS2 ВЕРСИЯ 1

В настоящей статье представлены отдельные этапы образования основного математического обеспечения вычислительной сети SKJS2 версия 1. Обсуждены последовательные фазы проектирования основного математического обеспечения, проблемы эффективности работы программистов, а прежде всего — тестирования и большого математического обеспечения.

*Sieć komputerowa,
transfer zbiorów,
język dostępu do usług*

Maria JACUKOWICZ*
Andrzej JANISZEWSKI*
Jan KWIATKOWSKI*

DOSTĘP TERMINALOWY W SIECI SKJS2 WERSJA 1

Opisano sposób wykorzystania terminali w sieci SKJS 2 wersja 1. Przedstawiono rodzaje terminali, z których można uzyskać dostęp do sieci oraz aplikacji sieciowych. Podano postać komend umożliwiających ten dostęp. Opisano rozwiązania programowe dotyczące sposobu połączenia z aplikacją, formatowania ekranu i zmian umożliwiających wykorzystanie podsystemu TSO w pracy sieciowej.

1. WSTĘP

Jedną z usług oferowanych przez sieć SKJS2 wersja 1 jest możliwość dostępu terminalowego do aplikacji wielodostępnych działających w komputerach obliczeniowych sieci. W założeniach wstępnych [2,3] przyjęto możliwość dostępu do następujących aplikacji:

- TSO (ang. Time-Sharing Options)
- SKOT (System Kontroli i Obsługi Terminali)..

Jednocześnie założono możliwość stosowania innych aplikacji sieciowych, konstruowanych samodzielnie przez użytkownika sieci [2].

Możliwość współpracy z aplikacjami mają zarówno terminale podłączone do danego komputera obliczeniowego (tzw. lokalne), jak rów-

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

niez terminale podłączone (fizycznie) do innych komputerów obliczeniowych w sieci.

Współpracę terminali z aplikacjami systemu, umieszczonymi w dowolnym komputerze obliczeniowym, zapewnia język dostępu do sieci. Programem odpowiedzialnym za interpretację komend języka dostępu sieciowego oraz zestawienie połączeń z odpowiednimi aplikacjami w komputerze obliczeniowym jest moduł o nazwie MŁOT (Moduł Łącznikowy Obsługi Terminali).

2. RODZAJE TERMINALI

W sieci SKJS2 wersja 1 dostęp przez terminale obsługiwany jest przez program PSPC wchodzący w skład oprogramowania węzła, metodę dostępu telekomunikacyjnego TCAM5 oraz program MŁOT. Program PSPC powstał na bazie programu NCP i umieszczony jest w komputerze czołowym (spełniającym jednocześnie funkcje węzła), natomiast pozostałe dwa w komputerze obliczeniowym.

Aplikacje dostępne przez terminale, są aplikacjami umożliwiającymi pracę interakcyjną. Przyjęte rozwiązanie umożliwia zastosowanie następujących urządzeń:

- niezależnych stacji abonenckich EC8575M
- monitorów EC7917 oraz drukarki EC7914 ze sterownikiem grupowym EC7912
- monitorów EC7917 z grupowym sterownikiem zdalnym EC7911
- monitorów niezależnych EC7915.

Obsługa terminali w sieci SKJS2 wersja 1, opierająca się na programach MCP i NCP, pozwala na obsługę także innych typów terminali. W obecnej wersji oprogramowania bazowego sieci obsługiwane są tylko wymienione typy. Ograniczenie to wynika z cech interakcyjności każdej aplikacji oraz z szerokiego rozpowszechnienia wymienionych urządzeń w kraju. Ewentualne rozszerzenie o nowe typy wymaga modyfikacji programu MŁOT oraz dostarczenia aplikacji współpracujących z tymi terminalami.

3. ZESTAWIENIE POŁĄCZEŃ TERMINALOWYCH

Po załadowaniu oprogramowania sieciowego do komputera obliczeniowego oraz komputera czołowego i węzła, terminale sieciowe, zdefiniowane w opisie programu MCP i NCP, stają się terminalami aktywnymi.

Terminalem aktywnym nazywamy terminal, który może nawiązać połączenie z dowolną interakcyjną aplikacją sieciową. Do nawiązania połączenia służą komendy dostępu sieciowego. Obsługę komend dostępu sieciowego wykonuje się w programie MŁOT. Program ten identyfikuje rodzaj terminala przez nagłówek komunikatu wysyłanego przez program MCP. Potrzeba identyfikacji rodzaju terminala wynika z różnic w sterowaniu między terminalami ekranowymi a terminalami wierszowymi (np. typu DZM). Stosowanie programu MŁOT wymaga informacji o rodzaju interakcyjnej aplikacji sieciowej, z którą użytkownik chce nawiązać połączenie. Możemy tutaj wyróżnić dwa rodzaje aplikacji sieciowych - aplikacje, które same sterują terminalami oraz aplikacje, które tej własności nie wykazują. Podsystem SKOT jest aplikacją, która samodzielnie formatuje dane dla terminala sieciowego. Dla aplikacji samodzielnie formatujących dane terminala zakłada się tzw. transparentne połączenie z terminalem użytkownika. Oznacza to, że program MŁOT nie ingeruje w treść komunikatów wymienianych między terminalem a tą aplikacją, stanowiąc jedynie swego rodzaju "medium" transmisyjne dla przepływu komunikatów. Jeżeli aplikacja sieciowa sama nie formatuje danych terminala, funkcję tę przejmuje program MŁOT, tworząc połączenie nietransparentne.

Taką aplikacją w sieci SKJS2 jest podsystem TSO. W firmowym oprogramowaniu podsystemu TSO funkcję tę wypełniało oprogramowanie TIOC, stanowiące część programu MCP. W sieci SKJS2 wersja 1 zrezygnowano z usług oprogramowania TIOC z następujących względów:

- oprogramowanie to zapewniało jednokierunkową obsługę komunikatów, tzn. wysyłając komendę do podsystemu TSO należało oczekiwać na jej realizację i dopiero potem wysłać następną komendę
- koncepcja oprogramowania TIOC nie dopuszczała nawet standardowej obsługi terminali zdalnych pracujących w trybie NCP
- pobieżna analiza oprogramowania TIOC wskazuje na małą jego efektywność, przede wszystkim ujawniającą się w niezadowalającej szybkości obsługi terminali.

Daje się więc zauważyć, że program MCP w sieci SKJS2 wersja 1 wykorzystywany jest jedynie jako mechanizm przenoszący komunikaty między aplikacją sieciową a programem MŁOT oraz programem MŁOT a terminalami (lokalnymi i zdalnymi). Jeżeli program MCP jest wykorzystywany przede wszystkim do przenoszenia komunikatów między programami (i terminalami), to jest on programem efektywnym (szybka obsługa kolejek komunikatów).

W sieci nie istnieje oprogramowanie terminala wirtualnego, ponieważ jest to sieć homogeniczna.

Realizacja przez program MŁOT funkcji sterowania terminalami sieciowymi (dla aplikacji nietransparentnych) pozwoliła uzyskać, w stosunku do firmowego oprogramowania podsystemu TSO, pracę w trybie duplexowym. Oznacza to, że użytkownik ma możliwość wysyłania kolejnych poleceń do podsystemu TSO bez konieczności oczekiwania na ich realizację.

Niewielka zmiana w sposobie formatowania każdego wiersza przez program MŁOT (w stosunku do formatowania ekranu przez TIOC) pozwoliła na uzyskanie cech właściwych edytorom ekranowym (można zredukować dowolnie wybrane fragmenty ekranu). Oczywiście właściwość ta w znacznym stopniu przyspiesza edycję zbiorów tekstowych.

4. KOMENDY DOSTĘPU SIECIOWEGO

Komendy dostępu sieciowego umożliwiają komunikację użytkownika z siecią. Służą one do nawiązania lub rozłączenia połączenia sieciowego z dowolnym komputerem obliczeniowym włączonym do sieci SKJS2 wersja 1.

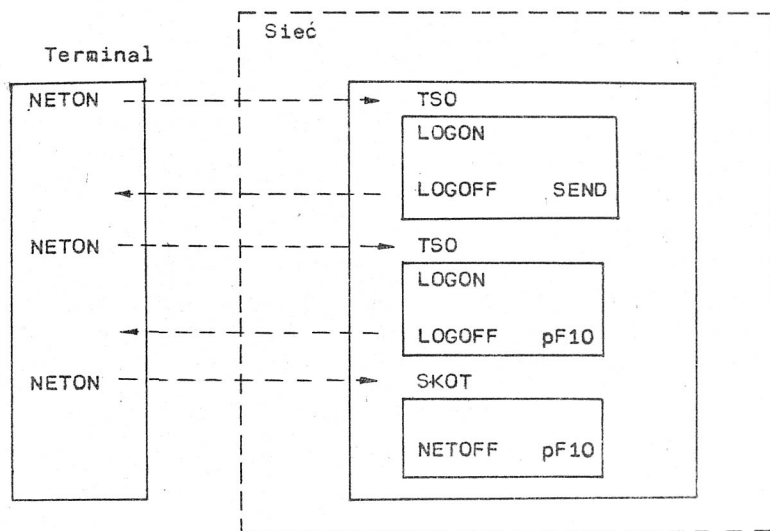
Język dostępu sieciowego składa się tylko z dwóch komend [4]: NETON oraz NETOFF. Komendy te można wprowadzać używając dużych lub małych liter.

Wydanie komendy NETON powoduje nawiązanie połączenia z wybraną aplikacją sieciową. Parametrami tej komendy są numer komputera obliczeniowego, w którym umieszczono wybraną aplikację oraz numer aplikacji w tym komputerze. Numer komputera obliczeniowego jest unikatowy w całej sieci, a numer aplikacji w danym komputerze.

Komenda NETOFF służy do rozłączenia istniejącego połączenia między terminalem a aplikacją. Jest to komenda bezparametrowa. W czasie trwania połączenia z aplikacją użytkownik prowadzi z nią dialog, używając komend specyficznych dla tej aplikacji.

Do wprowadzania komend dostępu sieciowego służy specjalnie zdefiniowany klawisz funkcyjny. Dla komendy NETON jest to klawisz PF10 (wartość domyślna dla terminali ekranowych). Identyczny klawisz służy do akceptacji komendy NETOFF. Jeżeli jednak aplikacja sieciowa wykorzystuje do swoich celów klawisz PF10, to w komendzie NETON mamy możliwość wskazania innego klawisza, który pozwoli zaakceptować komendę NETOFF.

Dla terminali typu DZM rolę klawisza funkcyjnego pełni znak twardy z cyrylicy, poprzedzający komendy sieciowe. Na rysunku zilustrowano użycie komend dostępu sieciowego.



Rys. Ilustracja komend dostępu sieciowego
Fig. Illustration of the network access commands

5. SYSTEM TRANSFERU ZBIORÓW

Jedną z aplikacji sieciowych dostępnych przez terminale w sieci SKJS2 wersja 1 jest usługa transferu zbiorów [1]. Usługa ta umożliwia użytkownikowi przenoszenie między komputerami obliczeniowymi sieci zbioru o strukturze sekwencyjnej (F.FB. V. VB). Dostęp do usługi realizowany jest przez system komend systemu transferu zbiorów umożliwiający:

- a) zdefiniowanie polecenia transferu do/z lokalnego komputera obliczeniowego (komenda TRANSFER)
- b) zdefiniowanie zapytania do systemu transferu zbiorów o stan zaawansowania realizacji uprzednio wydanego polecenia transferu (komenda WHATORDER)
- c) zdefiniowanie polecenia nakazującego skasowanie uprzednio wydanego polecenia transferu (komenda MODIFYORDER)
- d) zdefiniowanie zapytania o sposób zakończenia realizacji uprzednio wydanego polecenia transferu (komenda MAIL).

Powyższe polecenia wydawane są w sposób transakcyjny, tzn. wydane polecenie jest rejestrowane i następnie wykonywane niezależnie od

dalszej działalności użytkownika. Ten sposób obsługi nakłada na użytkownika konieczność zdefiniowania wszystkich szczegółów dotyczących transferu lub innych poleceń w jednej interakcji z systemem, a jednocześnie umożliwia przyjmowanie poleceń do systemu transferu zbiorów niezależnie od czasu eksploatacji systemu.

6. PODSUMOWANIE

W pracy przedstawiono sposób dostępu do aplikacji sieciowych w sieci SKJS2 wersja 1 przez terminale. Podano postać komend umożliwiających dostęp do sieci oraz przedstawiono język dostępu do usługi transferu zbiorów. Zaprezentowano przyjęte rozwiązania sprzętowo-programowe.

LITERATURA

- [1] DUBIELEWICZ A. i in., Stacja transferu zbiorów w sieci SKJS2 wersja 1 - Dokumentacja użytkownika, CO PWr., Raport SPR 24/86, Wrocław 1986.
- [2] DYRKA K., RUTKOWSKI E., Zasady tworzenia i udostępniania zasobów w sieci SKJS2 wersja 1, CO PWr., Raport SPR 14/86, Wrocław 1986.
- [3] HUZAR A., ŻAK D., Protokół interfejsu pomiędzy komputerem czołowym metodą dostępu w komputerze obliczeniowym i aplikacjami w sieci SKJS2 wersja 1, CO PWr., Raport SPR 3/86, Wrocław 1986.
- [4] HUZAR A., JACUKOWICZ M., Terminal w sieci SKJS2 wersja 1 - Podręcznik użytkownika, CO PWr., Raport SPR 20/86, Wrocław 1986.

Praca wpłynęła do Redakcji 1988.03.28

TERMINAL ACCESS TO SKJS2 COMPUTER NETWORK

The paper is concerned with the method of utilizing different terminals in the SKJS2 version 1 computer network. Some chosen problems connected with network software development are described. On this base the new possibilities obtained as result of some changes in TSO system are presented. At the end user interface to network and its application is given.

ДОСТУП К ТЕРМИНАЛАМ В СЕТИ SKJS2 ВЕРСИЯ 1

В статье описан способ использования терминалов в сети SKJS2 версия 1. Представлены типы терминалов, из которых можно получить доступ к сети и сетевым программам пользователя. Описаны команды, которые дают возможность этого доступа. Описаны также способы решения следующих задач: соединение с сетевыми программами пользователя, установление формата экрана терминала и переработки, касающиеся работы подсистемы TSO в сетевом режиме.

*Sieć komputerowa,
sterowanie przepływem*

Józef STAŃKO*

Danuta ŻAK*

STYKI A STEROWANIE PRZEPŁYWEM W SIECI SKJS2 WERSJA 1

Omówiono transparentny styk między komputerem obliczeniowym a węzłem w jednorodnej sieci komputerowej Jednolitego Systemu SKJS2 wersja 1. Styk ten umożliwia użytkownikowi sieci komunikowanie się z aplikacjami zainstalowanymi w komputerach obliczeniowych i rozpoznającymi go. Obowiązuje on również między aplikacją i programem MŁOT w komputerze obliczeniowym, pośredniczącym w komunikacji. Opis styku obejmuje opis procedur, strukturę komend oraz opis implementacji w węźle.

1. WSTĘP

W sieci SKJS2 stosuje się trzy rodzaje połączeń terminali z węzłem EC 8371.01:

- trwałe punkt-punkt
- trwałe wielopunkt
- komutowane punkt-punkt.

Transmisja na tych połączeniach może być synchroniczna lub asynchroniczna. Dwa węzły mogą być połączone dwoma łączami trwałymi lub komutowanymi. Łącza te są dwupłaskowe i transmisja na nich odbywa się zgodnie z protokołem bitowym HDLC [1].

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

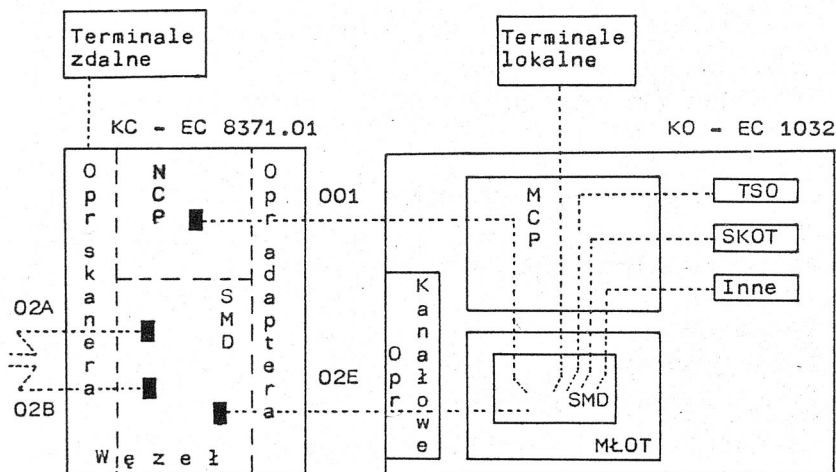
Komputer węzła jest włączony w kanał multiplekserowy komputera obliczeniowego. Transmisja na tym łączu odbywa się zgodnie ze stykiem logicznym TCAM5/NCP2 wykorzystując jeden podkanał multiplekserowy komputera obliczeniowego.

Żądania użytkownika terminala są przekazywane przez program MCP do aplikacji. Zaimplementowana wersja TCAM-u nie współpracuje z TSO. Stąd powstała potrzeba opracowania specjalnego styku w celu komunikowania się z TSO. Dla tego styku został użyty jeden dodatkowy podkanał multiplekserowy oraz opracowany program kanałowy w komputerze obliczeniowym, obsługujący ten podkanał oraz specjalne oprogramowanie adaptera kanałowego w węźle.

Opracowanie oddzielnego styku logicznego i wydzielenie oddzielnego podkanału umożliwiło w stosunkowo prosty sposób włączenie podsystemu TSO do sieci SKJS2.

2. STYKI W SIECI SKJS2 WERSJA 1

Komputer czołowy (KC) jest połączony z komputerem obliczeniowym (KO) łączem kanałowym. Podstawowe styki fizyczne i logiczne przedstawiono na rysunku.



Rys. Styki w sieci SKJS2 wersja 1
Fig. Interfaces in SKJS2 network

Standardowy program NCP (Network Control Program) współpracuje z Metodą Dostępu Telekomunikacyjnego (TCAMD) przez pojedynczy podkanał multiplekserowy (np. 001).

Zaimplementowane wersje TCAM5 i NCP2 nie współpracują z systemem TSO. Aby umożliwić użytkownikom zdalnym sieci SKJS2 (tzn. nawiązującym połączenie z KO przez podsieć) korzystanie z systemu TSO, opracowano dodatkowy styk, tzw. sieciową metodę dostępu (SMD). Komendy tego styku są opisane w rozdz. 3.

Zaprojektowano i zaimplementowano w KO program o nazwie MŁOT, który jest aplikacją współpracującą w standardowy sposób z programem MCP. Program MŁOT współpracuje z aplikacjami zgodnie ze stykiem logicznym SMD. Istniejące aplikacje typu TSO, SKOT zostały zmodyfikowane w celu obsługi tego styku. Do obsługi SMD na poziomie fizycznym wykorzystano po stronie KO oddzielny podkanał (np. 02E) i własny program kanałowy. Po stronie KC zmodyfikowano obsługę adaptera kanałowego oraz opracowano moduły obsługujące linie sieciowe (HDLC) skanera.

Logiczny styk SMD obowiązuje na połączeniu MŁOT-aplikacja w KO oraz na styku MŁOT-węzeł w KC. Pozostałe styki w sieci SKJS2 są zgodne z modelem OSI/ISO warstw 1-4.

3. PROCEDURY STYKU SMD

3.1. Procedury globalne

3.1.1. Procedura restartu

Procedura restartu jest wykorzystywana do inicjowania lub reinicjowania styku MŁOT-aplikacja lub MŁOT-węzeł. W przypadku styku MŁOT-aplikacja restart jest dokonywany na żądanie aplikacji. W przypadku styku MŁOT-węzeł inicjatorem restartu jest program MŁOT. Procedura restartu kasuje wszystkie połączenia sieciowe. Inicjowana jest komenda "restart" (RST), a potwierdzana komendą "potwierdzenie restartu" (RSTC).

3.1.2. Procedura wstrzymania nadawania

Procedura wstrzymania nadawania jest podejmowana wówczas, gdy istnieje przeciążenie KC lub KO i pula wolnej pamięci nie wystarcza na obsłużenie przepływu danych na wszystkich połączeniach. Jest ona inicjowana komendą "żądanie wstrzymania nadawania" (ZWN). Blokada ta trwa do momentu przesłania komendy "kontynuacja nadawania" (KNN).

3.1.3. Procedura zakończenia pracy

Procedura zakończenia pracy jest inicjowana komendą "koniec pracy" (KPR). Wówczas wszystkie sesje zostają zamknięte (przez sesje rozumiemy połączenie użytkownika z aplikacją).

3.2. Procedury sesji

3.2.1. Procedury nawiązania połączenia

Nawiązywanie połączenia transparentnego jest inicjowane komendą "żądanie nawiązania połączenia" (ZNP). Może ona być wysłana przez KO lub KC. Nawiązanie połączenia jest potwierdzone komendą "potwierdzenie nawiązania połączenia" (PNP) lub odmową "odrzućcie nawiązania połączenia" (ONP).

3.2.2. Procedura transferu danych

Dane normalne mogą być przesyłane w obu kierunkach za pomocą komendy "dane normalne" (DNN). Każda komenda DNN przenosi całkowitą jednostkę danych lub jej część. Koniec jednostki danych jest wskazany za pomocą znaku końca EOT.

Otrzymywanie danych normalnych jest potwierdzane komendą "potwierdzenie danych normalnych" (PDN) po otrzymaniu liczby komend równej wielkości okna. Wielkość okna można zmienić - standardowa wartość wynosi 10. Mechanizm okna służy również do sterowania przepływem, które polega na możliwości wstrzymania przepływu danych.

Przesyłanie danych przyspieszonych jest realizowane komendą "dane przyspieszone" (DNP). Komenda ta nie jest potwierdzana.

3.2.3. Procedura rozłączenia połączenia

Zgłoszenie kasowania (rozłączenia) połączenia następuje przez przesłanie przez styk MŁOT-aplikacja lub MŁOT-węzeł komendy "zgłoszenie rozłączenia połączenia" (ZRP). Komenda ta jest potwierdzana komendą "potwierdzenie rozłączenia połączenia" (PRP).

3.3. Struktura komend

Komendy składają się z części stałej i zmiennej. Część stała jest 3-bajtowa, część zmienna ma najwyżej 125 bajtów. W skład części zmiennej wchodzi parametry kluczowe. Część stałą tworzą trzy jedno-bajtowe pola: kod komendy, identyfikator portu oraz licznik zawierający długość części zmiennej.

Kod komendy	Identyfikator portu	Długość części zmiennej	Parametry kluczowe właściwe dla typu komendy
← część stała →			← część zmienna →

Dopuszczalne komendy i ich parametry kluczowe są umieszczone w tab. 1.

T a b e l a 1

Komendy styku KO-KC

Komenda	Skrót	Parametry kluczowe (opcjonalne)
Restart	RST	Identyfikator aplikacji
Potwierdzenie restartu	RSTC	Identyfikator aplikacji
Ządanie wstrzymania nadawania	ZWN	Identyfikator aplikacji
Kontynuacja nadawania	KNN	Identyfikator aplikacji
Koniec pracy	KPR	Identyfikator aplikacji
Ządanie nawiązania połączenia	ZNP	Adres procesu docelowego, adres nadawcy, wielkości okien, dane użytkownika, identyfikator portu nadawcy
Potwierdzenie nawiązania połączenia	PNP	Identyfikator portu nadawcy, dane użytkownika
Odrzucenie nawiązania połączenia	ONP	Przyczyna, dane użytkownika
Dane normalne	DNN	Identyfikator EOT, dane
Dane przyspieszone	DNP	Identyfikator EOT, dane
Potwierdzenie odbioru danych	PDN	Identyfikator EOT, dane
Zgłoszenie rozłączenia połączenia	ZRP	Przyczyna, dane użytkownika
Potwierdzenie rozłączenia	PRP	Przyczyna, dane użytkownika

4. IMPLEMENTACJA STYKU MIĘDZY KO I KC

4.1. Struktura bloków sterujących

Do utworzenia odcinka drogi przepływu danych są wykorzystywane bloki sterujące w postaci tablic. Dla każdego połączenia sieciowego jest wykorzystywana jedna tablica, tzw. tablica portu normalnego. Dla wszystkich połączeń jest wykorzystywana dodatkowo jedna tablica, tzw. tablica portu systemowego. Tablice opisujemy w postaci sekcji pozornych języka asemblera.

*
* DEFINICJA TABLICY PORTU NORMALNEGO

*

XXCXTTPN DSECT

TPN DS OF

TPNQCB EQU * OPIS PROCESU

XQCB PREFIX=TPN,TYPE=INPUT

DS OF

TPNECB EQU *-XXCXTTPN

XECB PREFIX=TPN

TNOQCB EQU *-XXCXTTPN OPIS KOLEJKI

XQCB PREFIX=TNO,TYPE=WORK

* KOMEND Z KO

TINQCB EQU *-XXCXTTPN OPIS KOLEJKI

XQCB PREFIX=TIN,TYPE=WORK

* DANYCH DO KO

TIPQCB EQU *-XXCXTTPN OPIS KOLEJKI DANYCH

XQCB PREFIX=TIP,TYPE=WORK PRZYSPIESZONYCH

TPNTYP DS X TYP TABLICY

TPNIDP DS X IDENTYFIKATOR PORTU

TPNSTP DS X

TPNIDWKO DS X

* ZWIĄZANEGO W KO

TPNATPT DS H

* ADRES TABLICY POŁĄCZENIA

* TRANSPORTOWEGO Z KTÓRĄ JEST

* USTANAWIANY ZWIĄZEK

TPNTYT DS X

* TYP TABLICY, Z KTÓRĄ JEST

* USTANOWIONY ZWIĄZEK

TPNOKZ DS X

TPNLOD DS X

TPNOKD DS X

TPNLND DS X

TPNSYG DS B

TPNSYGO DEFMSK 0

* OTRZYMANO

TPNSYG1 DEFMSK 1

* T-DISCONNECT INDICATION

* OTRZYMANO

TPNSYG2 DEFMSK 2

* T-DISCONNECT INDICATION

* OTRZYMANO

TPNSYG3 DEFMSK 3

* T-CONNECT CONFIRM

* OTRZYMANO

TPNSYG4 DEFMSK 4

* T-WARNING INDICATION

* OTRZYMANO

TPNSYG5 DEFMSK 5

* T-CONTINUE INDICATION

* WYŚLANO SYGNAŁ

TPNSYG6 DEFMSK 6

* NA PRZEPŁNIENIE KOLEJKI

TPNPRZ DS XC

* OCZEKIWANIE NA KOMENDĘ PDN Z KO

TPNQCBL DS X

TPNQCBL DS X

END

*

STAN PORTU NORMALNEGO

IDENTYFIKATOR PORTU

ADRES TABLICY POŁĄCZENIA

TRANSPORTOWEGO Z KTÓRĄ JEST

USTANAWIANY ZWIĄZEK

TYP TABLICY, Z KTÓRĄ JEST

USTANOWIONY ZWIĄZEK

OKNO Z KO

LICZNIK ODEBRANYCH KOMEND DNN

OKNO DO KO

LICZNIK NADANYCH KOMEND DNN

SYGNAŁY

OTRZYMANO

T-DISCONNECT INDICATION

OTRZYMANO

T-DISCONNECT INDICATION

OTRZYMANO

T-CONNECT CONFIRM

OTRZYMANO

T-WARNING INDICATION

OTRZYMANO

T-CONTINUE INDICATION

WYŚLANO SYGNAŁ

NA PRZEPŁNIENIE KOLEJKI

OCZEKIWANIE NA KOMENDĘ PDN Z KO

PRZYCZYNA ROZŁĄCZENIA

POŁĄCZENIA

DŁUGOŚĆ KOLEJKI DO KO

DŁUGOŚĆ KOLEJKI Z KO

* DEFINICJA TABLICY PORTU SYSTEMOWEGO

```

*
XXCXTTPS  DSECT
TPS        DS      OF
TPSQCB     EQU      *          OPIS PROCESU
          XQCB     PREFIX=TPS.TYPE=INPUT
          DS      OF
TPSECB     EQU      *-XXCXTTPS
          XECB     PREFIX=TPS
TPSTYP     DS      X          TYP TABLICY
TPSIDN     DS      X          IDENTYFIKATOR PORTU
TPSLPN     DS      X          LICZBA WYGENEROWANYCH
*          PORTÓW NORMALNYCH
TPSSYG     DS      B          SYGNAŁY
TPSSYG0    DEFMSK 0          OTRZYMANO KOMENDĘ ZWW
TPSSYG1    DEFMSK 1          NALEŻY WYŚLAĆ KOMENDĘ ZWW
TPSSYG2    DEFMSK 2          NALEŻY WYŚLAĆ KOMENDĘ RST
TPSSYG3    DEFMSK 3          UPŁYNAŁ TIME-OUT
TPSSTP     DS      X          STAN PORTU SYSTEMOWEGO
          DS      OF          OPIS KOLEJKI KOMEND Z KO

TSYQCB     EQU      *-XXCXTTPS
          XQCB     PREFIX=TSY.TYPE=WORK
TPSTIME     DS      X          TIME OUT DO WYŚLANIA KNN
TPSLICZ     DS      X          LICZNIK KROTNOŚCI SPRAWDZEŃ
* PULI BUFORÓW W CELU WYŚLANIA KNN
TPSDLUG     EQU      *-XXCXTTPS
          END

```

4.2. Tabele stanów portów w procesorze EC.8371.01

T a b e l a 2

Tablica stanów portu systemowego

Stan portu Komenda	X'00' nieaktywny	X'04' aktywny	X'01' oczekiwanie na RSTC
R-RST	X'04'	X'04'	X'04'
S-RST	-	X'01'	-
R-RSTC	Ignoruj	Ignoruj	X'04'
R-KPR	Abend X'834'	X'00'	X'00'
R-ZWN	Abend X'835'	X'04'	Ignoruj
R-KNN	Abend X'836'	X'04'	Ignoruj
R-ZNP	Abend X'837'	X'04'	Ignoruj
R-DNN	Abend X'83C'	X'04'	Ignoruj

Oznaczenia: R - otrzymano komendę, S - nadano komendę

Tablica przejść portu normalnego (cz. 1)

Komenda \ Stan portu	X'00' nieaktywny	X'01' wolny	X'02' oczekiwanie na po- twierdzenie PNP z podsieci
R=RST	X'01'	X'01'	X'01'
S=RST	X'00'	X'00'	X'00'
R-KPR	X'00'	X'00'	X'00'
ZNP z KO	-	X'02'	-
ZNP z podsieci	-	X'03'	-
PNP z KO	Ignoruj	Abend X'807'	Abend X'808'
ONP z KO	Ignoruj	Abend X'80B'	Abend X'80C'
DNN z KO	Ignoruj	Abend X'80F'	Abend X'810'
PNP z podsieci	AbendX'801'	Abend X'802'	X'04'
DNN z podsieci	AbendX'812'	Abend X'813'	Abend X'814'
DNP z KO	Ignoruj	Abend X'818'	Abend X'819'
DNP z podsieci	AbendX'82F'	Abend X'830'	Abend X'831'
PDN z KO	Ignoruj	Abend X'81C'	Abend X'81D'
ZRP z KO	Ignoruj	Abend X'825'	-
ZRP z podsieci	AbendX'827'	Abend X'828'	X'01'
PRP z KO	Ignoruj	Abend X'82B'	Abend X'82C'

- Uwagi: 1. Otrzymanie komendy ZRP, gdy uprzednio wydano komendę ZRP, jest traktowane jako potwierdzenie połączenia (PRP).
 2. Ządania do/z podsieci nie są w postaci komend styku KO i KC.

Port systemowy pośredniczy w dialogu operatora systemu OSMVT 5.01 z oprogramowaniem sieciowym procesora EC 8371.

Wśród komend wprowadzanych przez operatora wyróżnia się następujące grupy:

- wysyłanie datagramów
- ustawianie kategorii monitorowania

- zmiany stanu kierunku sieciowego
- zmiany stanu połączenia sieciowego
- zmiany stanu połączenia transportowego.

T a b e l a 3

Tablica przejść portu normalnego (cz. 2)

X'03' oczekiwanie na PNP z KO	X'04' transfer danych	X'07' oczekiwanie na PRP z KO
X'01'	X'01'	X'01'
X'00'	X'00'	X'00'
X'00'	X'00'	X'00'
-	-	-
-	-	-
X'04'	Abend X'809'	Ignoruj
X'01'	Abend X'80D'	Abend X'845'
Abend X'811'	X'04'	Ignoruj
Abend X'803'	Abend X'804'	Abend X'805'
Abend X'815'	X'04'	Abend X'816'
Abend X'81A'	X'04'	Ignoruj
Abend X'832'	X'04'	Abend X'833'
Abend X'81D'	X'04'	Ignoruj
Abend X'826'	X'01'	X'01'
Oczekiwanie na odp. z KO	X'07'	Abend X'829'
Abend X'82D'	Abend X'82D'	X'01'

Komunikaty wyprowadzane do operatora należą do jednej z następujących kategorii:

- sygnalizacji błędów wprowadzanych komend
- odpowiedzi na pytanie o stan
- sygnalizacji zmian stanu kierunku sieciowego
- sygnalizacji stanu indywidualnych połączeń sieciowych
- sygnalizacji o przesyłaniu datagramów.

4.3. Przepływ danych

Do sterowania przepływem danych w komputerze 'czołowym' zostały wykorzystane makroinstrukcje nadzorcy (supervisora) programu NCP. Makroinstrukcje NCP spełniają cztery podstawowe funkcje:

- zarządzanie zadaniami
- zarządzanie kolejkami
- zarządzanie buforami
- usługi nadzorcy.

Obsługa portu normalnego, jak również systemowego, jest procesem (task) w sensie NCP. Makroinstrukcja ENQUE [4] dołącza bufor do wskazanej kolejki i budzi proces obsługujący tę kolejkę. Obudzony proces obsługuje najpierw bajt sygnałów ze stacji transportowej, później w kolejności: kolejkę komend z KO, kolejkę danych przyspieszonych i komend sterujących do KO oraz kolejkę danych do KO.

Do pobierania buforów z kolejki służy makroinstrukcja DEQUE[4].

Do wstrzymania napływu danych z KO służy mechanizm okna. Jeśli procesor EC 8371 nie potwierdzi danych, to KO nie może nadsyłać dalszych danych. Podobnie, jeśli KO nie potwierdzi porcji danych, regulowanych wielkością okna, to procesor EC 8371 nie może posyłać dalszych danych.

4.4. Procedura generowania sieciowego programu procesora EC 8371.01

Program sieciowy procesora EC8371 generuje się ze zmodyfikowanych bibliotek generacyjnych Network Control Program. Procedurę generowania można podzielić na trzy etapy:

1. Generowanie konfiguracji sieciowej EC 8371.
2. Generowanie opisu zadań, których wykonanie spowoduje utworzenie programu sieciowego. Generowanie to wykonuje się tak samo jak generowanie standardowego programu NCP z uwzględnieniem dodatkowych parametrów makroinstrukcji generacyjnych. Rozszerzenia makroinstrukcji dokonano podczas projektowania węzła.
3. Wykonanie zadań generacyjnych, z których ostatnie zawiera krok łączenia modułów w sieciowy program ładowalny.

W etapie 1. są tworzone statyczne struktury danych opisujące konfigurację sieciową. Dane te należy umieścić jako człon KONFIG w zbiorze określonym przez parametr QBULIB makroinstrukcji BUILD [4].

W etapie 3. wykonuje się kompilacja wymaganych modułów oraz następuje łączenie modułów w program sieciowy. W czasie łączenia do standardowych modułów programu NCP są dołączone moduły sieciowe.

LITERATURA

- [1] Dokumentacja oprogramowania sieci SKJS2 wersja 1.
- [2] Procesor teleprzetwarzania danych EC8371. Architektura logiczna.
- [3] IBM TCAM System Programmer Guide.
- [4] IBM 3704 and 3705 Communications Controllers. Network Control Program VS.PLM. Version 2. SY 30-3007.
- [5] IBM Network Control Program. Generation and Utilities.
- [6] BERNARDYN R., JANISZEWSKI A., STAŃKO J., WIETRZYCH J., Procedura generowania sieciowego programu procesora czołowego sieci SKJS2 wersja 1. CO PWr., Raport SPR 10/86, Wrocław 1986.

Praca wpłynęła do Redakcji 1988.03.28

THE INTERFACES AND FLOW CONTROL IN SKJS2 NETWORK, RELEASE 1

In the paper, the transparent interface between a computer host and computer node of SKJS2 network is presented. This interface enables the utilization of application programmes in hosts of SKJS2. It is also applied between the MLOT programme (intermediary in communications) and an application programme in the host. The interface description contains descriptions of procedures, structures of commands and implementation in the node.

СОЕДИНЕНИЯ И УПРАВЛЕНИЕ ПОТОКАМИ В СЕТЯХ SKJS2 ВЕРСИЯ 1

В статье описано прозрачное соединение между вычислительным компьютером и узлом в компьютерной сети Единственной Системы SKJS2 версия 1. Это соединение дает пользователю сети возможность общаться с прикладными программами, находящимися в вычислительных компьютерах. Оно обязывает также при сотрудничестве прикладных программ со специальной программой МЛОТ, которая играет роль посредника. В статье представлены: структура команд, описание процедур и программного обеспечения узла.

*Synchronizacja,
sieć komputerowa,
transfer zbiorów*

Andrzej HUZAR*

Andrzej KALIŚ*

PROBLEMY SYNCHRONIZACJI W STACJI TRANSFERU ZBIORÓW SIECI KOMPUTEROWEJ SKJS2 WERSJA 1

Omówiono sposoby synchronizacji procesów w stacji transferu zbiorów sieci komputerowej SKJS2 wersja 1 na tle ogólnych metod synchronizacji procesów oraz uwarunkowań środowiska, w którym stacja transferu zbiorów powstała. Przedstawiono model synchronizacji procesów i jego realizację na potrzeby stacji. Krótko omówiono zalety proponowanego podejścia.

1. PROBLEMY SYNCHRONIZACJI W STACJI TRANSFERU ZBIORÓW

Wraz z powstaniem możliwości i potrzeb tworzenia oprogramowania złożonego z kilku procesów traktowanych jako programy sekwencyjne, pojawiła się konieczność ich synchronizacji. Korzystają one zazwyczaj z pewnych wspólnych zasobów, do których dostęp w danej chwili mogą mieć tylko niektóre z nich (często tylko jeden). W przypadku tworzenia programów działających współbieżnie, ich twórcy znajdują się w następującej sytuacji: z jednej strony dostępny jest im zbiór metod synchronizacji, z drugiej zaś środowisko programowe dostarcza pewnych mechanizmów, za pomocą których można synchronizować procesy.

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

Usługi dostarczane przez system operacyjny (środowisko) nie muszą być gotowymi do wykorzystania mechanizmami synchronizacji. W większości przypadków charakter oprogramowania narzuca wybór pewnych metod synchronizacji.

Wydaje się, że współpracujące procesy można podzielić na dwie grupy różniące się sposobem synchronizacji:

1. Procesy, które zasadniczo pracują niezależnie, tzn. praca innych procesów nie jest potrzebna do ich właściwego działania, korzystają jedynie od czasu do czasu ze wspólnych zasobów środowiska (drukarek, dysków itp.). Nad ich niezależnością czuwa nadzorca systemu, rozwiązując problemy synchronizacji.

2. Procesy, które współpracując ze sobą wymieniają przetwarzaną informację (np. procesy będące implementacją tej samej warstwy sieci komputerowej pracujące w różnych komputerach).

Stacja transferu zbiorów została zaprojektowana jako zbiór współpracujących ze sobą podakcji. W terminologii systemu operacyjnego OS/MVT przez pojęcie podakcji rozumiemy zadanie, które przez system operacyjny traktowane jest jak każde inne zadanie. Oznacza to, że system może spowodować przejście podakcji w stan "niegotowe", jeżeli podakcja musi na coś czekać. Przy tym zasady generalne, ustalone dla całego systemu, są stosowane do podakcji. Z jednej strony oczekiwanym zdarzeniem może być na przykład: zakończenie operacji wejścia-wyjścia, odebranie odpowiedzi od operatora, upłynięcie pewnego odcinka czasu, uzyskanie możliwości modyfikacji zbioru przetwarzanego równocześnie przez wiele zadań. Z drugiej zaś strony, podakcja traktowana jako proces może sama przechodzić w stan zawieszenia, oczekując na realizację pewnego zdarzenia, uzależnionego zazwyczaj od wykonania ściśle ustalonych czynności przez współpracującą z nią podakcją. Innymi słowy, w pewnych punktach podakcji realizującej jeden proces musimy zakładać, że inny proces już coś zdążył zrobić.

W stacji transferu zbiorów procesy współpracują ze sobą różnie, tzn. przez dostęp do wspólnych zasobów i przez wymianę komunikatów. Również synchronizacja w każdej z tych grup jest zróżnicowana i odbywa się częściowo bezpośrednio przez system operacyjny i częściowo przez stosowanie mechanizmów utworzonych tylko na potrzeby stacji. W stacji można wskazać następujące miejsca, w których następuje synchronizacja procesów:

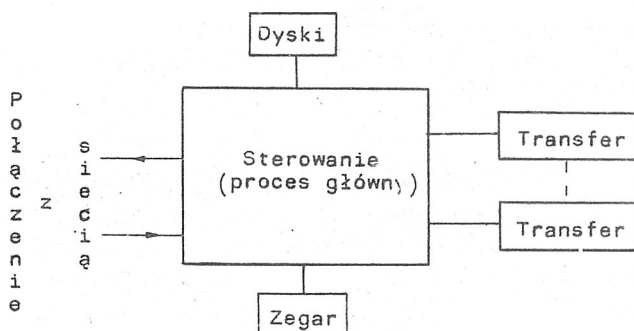
1. Stacja wysyła i odbiera treść przesyłanych zbiorów. Musi być więc rozwiązany problem dostępu do zbiorów przez stację i inne programy. W tym przypadku skorzystano z systemowego mechanizmu synchroniza-

cji (makroinstrukcje supervisora ENQ i DEQ z odpowiednią nazwą regionu krytycznego [4]).

2. Ze stacją współpracują programy uruchamiane przez użytkowników systemu transferu zbiorów, które wpisują zlecenia do kolejki. W tym przypadku również skorzystano z systemowego mechanizmu regionów krytycznych, ustalając własną jego nazwę.

3. Stacja komunikuje się z inną stacją. Tutaj synchronizację zapewnia zastosowany protokół transferu zbiorów [5].

4. Wewnętrzna budowa stacji jest wieloprotocowa (rys.).



Rys. Struktura stacji transferu zbiorów
Fig. File transfer station structur

Proces główny o nazwie STEROWANIE nadzoruje pracę stacji, pobiera zlecenia z kolejki zleceń, pośredniczy w przesyłaniu komunikatów do sieci oraz komunikuje się z innymi procesami stacji, którymi są:

- a) proces ZEGAR informujący o upływie kolejnego kwantu czasu,
- b) procesy o nazwie TRANSFER (praktycznie w dowolnej liczbie), które realizują protokół transferu zbiorów.

Utworzenie takiej struktury wynika przede wszystkim z możliwości podziału funkcji między poszczególnymi modułami (procesami) oraz z tego, że system OS/MVT dostarcza środków do stosunkowo łatwego tworzenia programów wieloprotocowych (współbieżnych). Problemy, które musieli rozwiązać twórcy sami, to:

- a) definicja wspólnych struktur danych, zadanie to okazało się dość łatwe
- b) definicja własnego modelu synchronizacji zgodnego z potrzebami i możliwościami realizacyjnymi.

2. MODEL SYNCHRONIZACJI PROCESÓW

Przyjęto, że procesy będą przysyłać żądania wykonania pewnych akcji między sobą przez dwa kanały logiczne (po jednym dla każdego kierunku). Każdy kanał może się znajdować tylko w jednym z dwóch stanów:

1. Kanał wolny - wtedy proces nadający może zainicjować żądanie wykonania pewnej akcji (zbiór możliwych żądań jest ustalony). Żądanie spowoduje zmianę stanu kanału na zajęty i uaktywnienie odbiorcy (jeśli był zawieszony).
2. Kanał zajęty - wtedy proces odbierający może wykonać żadaną akcję zmieniając stan kanału na wolny (po jej zakończeniu). W tym stanie proces nadający nie może inicjować nadania kolejnych żądań akcji.

W dowolnej chwili każdy z procesów może odczytywać stan kanału. Istotną cechą zastosowanego modelu synchronizacji jest to, że żaden z procesów nie musi czekać ani na dotarcie żądania do odbiorcy, ani na dostarczenie żądania od nadawcy. Ma to istotne znaczenie w przypadku, gdy proces STEROWANIE przesyła do procesu TRANSFER dane mające być zapisane na dysku, natomiast proces TRANSFER nie może tych danych zapisać na dysku z powodu chwilowej awarii.

3. REALIZACJA MODELU SYNCHRONIZACJI PROCESÓW

W celu realizacji przedstawionego modelu synchronizacji wykorzystano następujące mechanizmy i makroinstrukcje supervisoru systemu OS/MVT (ich pełny opis można znaleźć w pracy [4]):

1. ECB - zmienna, z którą związany jest mechanizm zawieszania i uaktywniania procesów (podakcji). Zawartość ECB wskazuje na to czy jakiś proces jest zawieszony, może zostać zawieszony lub nie zostanie zawieszony.

2. POST (ECB) - makroinstrukcja supervisoru, która uaktywnia proces zawieszony lub zapobiega zawieszeniu takiego procesu, gdy ten wykona makroinstrukcję WAIT (ECB). Proces identyfikowany jest przez nazwę ECB.

3. WAIT (ECB) - makroinstrukcja powodująca zawieszenie danego procesu, aż do momentu wykonania przez inny proces makroinstrukcji POST (ECB). Oczywiście proces nie będzie zawieszony, gdy wcześniej została wykonana makroinstrukcja POST (ECB).

Jak można się domyślić, wartość zmiennej ECB ustawiana jest przez system operacyjny.

Po to, by wykorzystać opisane makroinstrukcje w celu realizacji przedstawionego modelu synchronizacji, zdefiniowano wspólną strukturę danych. Poniżej została ona pokazana dla połączenia TRANSFER-STEROWANIE:

type

```
AKCJA = (wolny, akcja1, akcja2, ..., akcjan); {odpowiada
                                                stanowi kanału}
ECB    = integer;
POLE_SYNCHRONIZACJI = record
    ecb_sterowania,
    ecb_transferu      : ECB;
    kanał_transfer_sterowanie : akcja;
end;
```

Na zmiennej x typu POLE SYNCHRONIZACJI proces STEROWANIE działa według następującego schematu:

```
with x do
begin
    ecb_sterowania := 0;
    wykonaj_początek;
    while true do
    begin
        wait (ecb_sterowania);
        ecb_sterowania := 0;
        {jeśli ecb_sterowania=0, to wykonanie wait(ecb_sterowania)
        powoduje zawieszenie procesu}
        case kanał_transfer_sterowanie of
            wolny : ;
            :
            :
            akcjai : if możliwe_wykonanie_akcjii then
                begin
                    wykonaj_akcjęi;
                    kanał_transfer_sterowanie:=wolny;
                    post (ecb_transfer)
                end;
            :
            :
        end; {case}
    end; {while}
end; {with}
```

Natomiast proces TRANSFER może żądać wykonania pewnej akcji od procesu STEROWANIE tylko wtedy, gdy wartość zmiennej kanał-transfer-sterowanie = wolny, czyli kanał jest wolny.

Analogicznie został zdefiniowany kanał dla kierunku przeciwnego. Warto zauważyć, że:

- proces nadający uaktywnia odbiorcę, jeśli ten jest zawieszony oraz uniemożliwia jego zawieszenie dopóki odbiorca nie sprawdzi stanu tego kanału,
- żaden z procesów nie musi aktywnie czekać w pętli na żądanie innych procesów,
- proces nadający zawsze jest odwieszany (jeśli był zawieszony przez proces odbierający po zrealizowaniu żądania (jest w ten sposób informowany o zrealizowaniu akcji).

4. PODSUMOWANIE

Przedstawione sposoby synchronizacji w stacji transferu zbiorów sprawdziły się w praktyce, ponieważ:

- a) umożliwiły niezależne uruchamianie poszczególnych procesów;
- b) nie doprowadziły nigdy do jakichkolwiek problemów z synchronizacją w stacji;
- c) stacja pracując w warunkach pełnego obciążenia wykorzystuje około 10% czasu procesora, co wydaje się wielkością dopuszczalną;
- d) metoda synchronizacji pełni w oprogramowaniu stacji rolę metody strukturalnej; wprowadzić nie następuje syntaktyczne sprawdzenie poprawności metody, ale łatwe jest (co było robione) sprawdzenie "ręczne" tekstu programu;
- e) metoda jednolita dla całej stacji oraz innych komponentów oprogramowania komputera obliczeniowego (zwłaszcza oprogramowanie modułu MŁOT) i nie zależy od liczby typów akcji.

Okazało się też, że mimo braku nowoczesnego języka do programowania współbieżnego (np. Concurrent Pascala), można uzyskać podobny komfort programowania stosując się jedynie do pewnych reguł programowania [1] (moduł STEROWANIE zakodowany został w języku Pascal 360).

LITERATURA

- [1] HUZAR A., Programowanie współbieżne w systemie OS/JS na poziomie języka Pascal, Informatyka nr 6, 1983.
- [2] HUZAR Z., Wstęp do programowania współbieżnego, Wrocław, Wyd.PWr. 1985.

- [3] ISZKOWSKI W., MANIECKI M., Programowanie współbieżne, Warszawa, WNT, 1982.
- [4] OS/JS Makroinstrukcje supervisora, WZE Mera-Elwro, Wrocław 1976.
- [5] A Network Independent File Transfer Protocol, File Transfer Protocol Implementore Group, NIFTP-B (80), 5th Feb. 1981 (Blue Book).

Praca wpłynęła do Redakcji 1988.03.28

SOME SYNCHRONIZATION PROBLEMS IN THE FILE TRANSFER STATION OF THE NETWORK SKJS2 VERSION 1

The paper describes the methods of synchronization of the processes in the file transfer station of the network SKJS2 version 1, one the base of general synchronization methods and the system OS/JS environment conditions. The synchronization model in the file transfer station and its realization is presented. The advantages of this model are shortly described.

ПРОБЛЕМЫ СИНХРОНИЗАЦИИ В СТАНЦИИ ТРАНСФЕРА ФАЙЛОВ

В настоящей статье представлены способы синхронизации процессов в станции трансфера файлов вычислительной сети SKJS2 версия 1 на фоне общих методов синхронизации и условий среды, в которой она возникла. Показана модель синхронизации процессов, ее достоинства и реализация в станции.

*Sieć komputerowa,
pomiar, analizator
protokołów, tester*

Zbigniew FRYŻLEWICZ *

APARATURA POMIAROWO-DIAGNOSTYCZNA DLA SIECI KOMPUTEROWEJ

Skonstruowana aparatura pomiarowo-diagnostyczna spełnia funkcję analizatora/testera protokołów. Urządzenie skonstruowano dobudowując do modułowego systemu mikroprocesorowego (MSM) trzy dodatkowe karty oraz projektując całe oprogramowanie. Jest ono zdekomponowane na procesy i monitor, który tworzy dla nich środowisko do pracy współbieżnej i w czasie rzeczywistym. W opracowaniu podano cechy funkcjonalne urządzenia oraz zasady współpracy procesów w tym środowisku.

1. WSTĘP

Doświadczenia uzyskane w trakcie realizacji sieci MSK wskazywały, że zastosowanie specjalistycznych urządzeń monitorująco-testujących wydatnie ułatwia diagnostykę i lokalizację błędów transmisji wynikających z nieprawidłowej implementacji protokołów. W sieci SKJS/2, w której infrastruktura komunikacyjna była budowana od podstaw, przewidywano podobną skalę problemów podczas diagnozowania błędów w implementacjach protokołów komunikacyjnych stosowanych między różnymi jej komponentami. Do komunikacji miano stosować różne protokoły i to różnych klas. Planowano włączenie do sieci terminali stosujących protokoły asynchroniczny, synchroniczny znakowy BSC i syn-

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

chroniczny bitowy SDLC. Zastosowane w sieci SKJS/2 teleprocesory przetwarzania miały wykorzystywać do wzajemnej komunikacji dwupoziomowy protokół X.25 (protokół pakietowy X.25/3 i liniowy LAPB).

W fazie projektu funkcjonalnego i technicznego uwzględniono te wszystkie wymagania wzbogacając je dodatkowo o takie opcje, które czynić miały budowane urządzenie przydatne do jak najszerszej klasy zastosowań. W czasopiśmiennictwie technicznym są opisywane podobne urządzenia i podzielić je można umownie na dwie klasy. W pierwszej mieszczą się te, które spełniają najogólniej rozumiane funkcje analizatora/testera protokołów. Do klasy drugiej należą te, które dodatkowo wypełniają funkcje sterujące w podsieci komunikacyjnej (zbiieranie danych i wyznaczanie dróg, optymalizacja wykorzystania zasobów podsieci). W początkowej fazie projektowania budowane urządzenie miało należeć do drugiej klasy i tworzyć zintegrowany system pomiarowo-sterujący (SPS). Ze względu na poczynione ograniczenia w koncepcji całej sieci SKJS/2 zrezygnowano z aktywnego sterowania w podsieci komunikacyjnej i zbudowana aparatura o "historycznej" nazwie SPS jest urządzeniem klasy pierwszej. Urządzenia podobnego typu produkuje wiele firm elektronicznych, np. [1], [2], [5], lecz na przełomie lat 1983/1984 (start tematu) były one w Polsce niedostępne.

W punkcie 2 artykułu opisano cechy funkcjonalne SPS ilustrując je pomiarami przykładowymi. Przykłady dobrano tak, aby pokazać zastosowanie SPS do monitorowania /testowania transmisji według różnych protokołów, podać ich parametry i zakresy wartości. W punkcie 3 opisano budowę oprogramowania SPS. Jest ono zdekomponowane na procesy i monitor. W pracy opisano elementy składowe struktury pojedynczego procesu i szczegółowo podano usługi monitora, który tworzy dla procesów środowisko do ich wzajemnej synchronizacji, komunikacji i podziału zasobów. W kolejnym punkcie zestawiono wykorzystane przy budowie SPS standardowe moduły systemu MSM oraz podano podstawowe charakterystyki specjalnie zbudowanych, dodatkowych kart. Całość kończy ocena implementacji wraz z uwagami o możliwościach modyfikacji i rozbudowy SPS.

2. CECHY FUNKCJONALNE

Zbudowana aparatura pomiarowo-diagnostyczna [4] może być stosowana do:

- monitorowania transmisji przechodzącej przez styk S2,
- testowania komponentów sieci komputerowej połączonych z SPS stykiem S2.

W pierwszym przypadku SPS jest połączony trójnikiem z układem pomiarowym i jest obserwatorem oraz rejestratorem dialogu. W drugim przypadku SPS jest aktywną stroną dialogu, tj. generuje do układu sekwencje znaków i/lub bitów oraz rejestruje nadsyłane odpowiedzi. W obu przypadkach parametry mechaniczne, elektryczne, funkcjonalne i proceduralne styku S2 są opisane przez zalecenia V.24/V.28.

W maksymalnej konfiguracji programowo-sprzętowej SPS ma repertuar 24 komend, w tym 6 komend pomiarowych, 9 komend redagujących i 9 komend pomocniczych.

2.1. Monitorowanie

SPS umożliwia monitorowanie na dwóch poziomach szczegółowości. Poziom pierwszy to monitorowanie stanów obwodów styku S2 o numerach ze zbioru (103...115,125). Poziom drugi to monitorowanie transmisji prowadzonej według jednego z następujących protokołów: asynchronicznego, BSC, LAPB, SDLC i X.25/3 (pakietowego).

Komendy i parametry monitorowania są do SPS wprowadzane dialogowo. Koniec dialogu określa moment rozpoczęcia pomiaru, lecz koniec pomiaru wynika już z jego typu. W przypadku monitorowania stanu obwodów styku S2 pomiar kończy się w momencie zapełnienia wewnętrznego bufora o ustalonej generacyjnie pojemności. Na rysunku 1 podano przykład dialogu i uzyskanych wyników dla opisanego typu pomiaru.

Schemat pomiaru na drugim poziomie szczegółowości jest bardziej złożony i wymaga trzech typów informacji:

- charakterystyki monitorowanego protokołu
- ogólnych warunków transmisji i pomiaru
- warunków zakończenia monitorowania, tzw. pułapek.

Pierwszy typ informacji ściśle zależy od rodzaju protokołu i np. dla protokołu asynchronicznego należy podać informacje o:

- długości znaku
- długości elementu STOP
- typie parzystości.

Drugi i trzeci typ informacji mają części jednakowe dla wszystkich rodzajów protokołów oraz części specyficzne dla danego rodzaju protokołu. Na przykład dla protokołu LAPB, definiując warunki pomiaru, podaje się informacje o:

- szybkości transmisji
- konieczności rejestracji (lub nie) czasu pojawiania się jednostek protokołu
- typie kodowania (NRZ vs. NRZI)
- lokalizacji SPS względem DTE.

MI

* INTERFACE MONITOR *

Sample step 0.1 ms x (1... 65535) = 1

Displayed circuits (Y/N)

# 125	= N
# 115	= Y
# 114	= Y
# 113	= N
# 112	= N
# 111	= N
# 110	= N
# 109	= Y
# 108	= N
# 107	= N
# 106	= N
# 105	= N
# 104	= Y
# 103	= Y

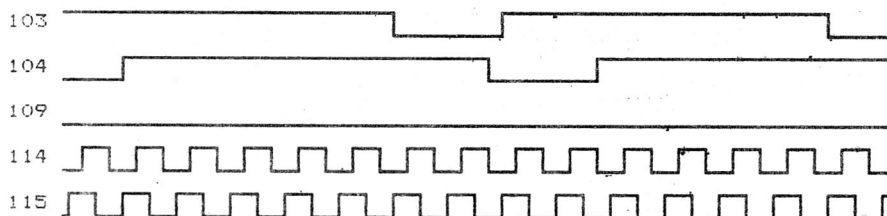
* * *

a)

* INTERFACE MONITOR *

START 14:25:22

SAMPLE STEP 1

CORRECT TRAP
CIRCUITS STATES

b)

Rys. 1. Monitorowanie stanu obwodów styku S2:

a - wprowadzenie komendy, b - kopia wyników

Fig. 1. Interface monitoring: a - command delivery, b - results

Przykładowo, definiując pułapkę kończącą monitorowanie transmisji według protokołu BSC, należy podać czy jest nią jedno z następujących zdarzeń:

- upływ czasu
- zapełnienie bufora pamięci pojemność na 100 jednostek protokołu średniej długości
- przepływ k-tej jednostki protokołu

- n-krotne wystąpienie błędu sumy kontrolnej
- m-krotne wystąpienie sekwencji najwyżej 6 znaków (z operacją maskowania dowolnych bitów).

Wszystkie warunki, z wyjątkiem pierwszego, mogą definiować niezależne lub wspólne pułapki dla strumienia odbiorczego (RX), nadawczego (TX) lub wspólnego (TX + RX).

Przykładową postać dialogu ustalającego warunki monitorowania transmisji według protokołu X.25/3 przedstawiono na rys. 2. Pomiar

*ME

* ,BIT-ORIENTED PROTOCOL MONITOR *

Line speed (500...9600)	= 2400
Protocol (L/S/X)	= X
Your tester local to DTE (Y/N)	= Y
Time mark (Y/N)	= Y
NRZI (Y/N)	= N

- Trap events -

Time limit (Y/N)	= Y
Value (m:s)	= 2
Traps def. for	
(CS/BS/TX/RX/NO)	= BS

TX

FCS (Y/N)	= Y
No (1...65535)	= 10
Abort (Y/N)	= Y
No (1...65535)	= 20
Buffer full (Y/N)	= N
Pattern (Y/N)	= N

RX

FCS (Y/N)	= N
Abort (Y/N)	= N
Buffer full (Y/N)	= Y
Pattern (Y/N)	= Y
Length (1...6)	= 2
# 1	= 11XX00XX
# 2	= 1111XXXX
No (1...65535)	= 10

* * *

Rys. 2. Definiowanie warunków i pułapek monitorowania transmisji według protokołu X.25/3

Fig. 2. X.25/3 transmission monitoring - parameters and traps initialization

rozpoczyna się z chwilą wprowadzenia wymaganych danych, a kończy w momencie zarejestrowania przez SPS zdarzenia realizującego zdefiniowaną pułapkę. W dowolnej chwili monitorowanie można przerwać, zatrzymać lub uzyskać raport bieżący z jego przebiegu. Z chwilą zakoń-

* LAPB PROTOCOL MONITOR *

```

Start          10:27:28   Duration          00:04:46:720
                                Pattern TX Trap
Pattern TX
XXXXXXXX11
Pattern RX
XXXXXXXX01

Results          TX          RX
U                6          6
S                7          6
I                3          3

FCS Errors       0          0
Aborts           0          0
Patterns         11         5

```

Rys. 3. Monitorowanie transmisji według protokołu LAPB - raport końcowy

Fig. 3. Final report of LAPB transmission monitoring

czenia pomiaru SPS generuje raport końcowy zawierający między innymi dane o czasie trwania pomiaru, przyczynie zakończenia, rodzajach i ilościach zarejestrowanych jednostek. Ten rodzaj raportu, z monitorowania transmisji według protokołu LAPB, przedstawiono na rys. 3.

W trakcie trwania pomiaru SPS zbiera wszelkie istotne informacje o wszystkich obserwowanych jednostkach protokołu, a w buforze cyklicznym jest pamiętanych po 100 jednostek ostatnich z każdego strumienia. Po skończonym pomiarze mogą one być przedmiotem dalszej analizy.

Komendy sterujące redagowaniem i wyprowadzaniem są tak dobrane, że pozwalają dokonać analizy jednostek protokołu na różnym poziomie szczegółowości, w tym zilustrować ich strukturę, wartości parametrów oraz dynamikę wymiany. SPS rejestruje wszystkie przepływające jednostki, również nierozpoznane, jeżeli tylko ich struktura, być może szkieletowa, umożliwia synchronizację układu sprzęgającego. Takie nierozpoznane jednostki są w raporcie bieżącym/kończącym specjalnie klasyfikowane, a ich treść należy wyprowadzać szesnastkowo. Jeżeli wszystkie jednostki monitorowanej transmisji należą do ustalonego protokołu, to ich wyprowadzanie i analizę wygodniej jest prowadzić w trybie proceduralnym, tj. zlecając SPS ich dekodowanie zgodnie z właściwym formatem.

Rysunki 4 i 5 zawierają wydruk w trybie szesnastkowym i proceduralnym tego samego fragmentu zarejestrowanej transmisji prowadzonej według protokołu LAPB.

```

TX + RX  STREAM
TX      10:32:06:440  03 3F
RX      10:32:06:540  01 3F
TX      10:32:06:580  01 73
RX      10:32:06:600  03 73
TX      10:32:06:660  03 00 10 00 FB 07 00
RX      10:32:06:720  01 00 10 00 FB 00 00
Last entity of TX
TX      10:32:06:740  01 21
Last entity of RX
RX      10:32:06:800  03 21

↑↑      ↑      ↑      ↑
A B      C      D      D(w ASCII)

```

Rys. 4. Wydruk szesnastkowy zawartości buforów rejestrujących monitorowanie transmisji według LAPB: A - typ strumienia, B - wskaźnik poprawności przesyłania ramki, C - czas, D - wartość

Fig. 4. LAPB transmission monitoring - hexadecimal listing of buffers

```

                X Y Z T W
                ↓ ↓ ↓ ↓ ↓
TX + RX  STREAM
TX      10:32:06:440  03 C P U SABM
RX      10:32:06:540  01 C P U SABM
TX      10:32:06:580  01 R F U UA
RX      10:32:06:600  03 R F U UA
TX      10:32:06:660  03 C I I N(R)=0 N(S)=0
                        10 00 FB 07 00
RX      10:32:06:720  01 C I I N(R)=0 N(S)=0
                        10 00 FB 00 00

```

Rys. 5. Wydruk proceduralny zawartości buforów rejestrujących monitorowanie transmisji według LAPB: X - pole adresu, Y - rozkaz (C)/odpowiedź (R), Z - stan bitu P/F, T - typ ramki, W - nazwa i wskaźniki

Fig. 5. LAPB transmission monitoring - procedural listing of buffers

2.2. Testowanie

SPS umożliwia prowadzenie testowania dwóch rodzajów. Pierwszy rodzaj to tzw. testowanie standardowe. Realizowana procedura - test 511B według CCITT - pozwala zmierzyć elementową i blokową stopę błędów wybranego kanału transmisji danych. Rodzaj drugi to tzw. testowanie programowe wykorzystujące specjalnie zbudowany język do nada-

wania, odbioru i analizy buforów danych między SPS a środowiskiem testowym.

Zainicjowanie testowania standardowego jest możliwe po zestawieniu odpowiedniej konfiguracji i wprowadzeniu do SPS informacji o:

- konfiguracji podlegającej testowaniu,
- warunkach zakończenia testowania (liczba cykli lub czas).

W testowaniu programowym użytkownik SPS ma do dyspozycji specjalny język programowania testów (TPL), którego instrukcje zestawiono w tabeli. W TPL obiektami działań są bufor, liczniki i stopery. Bufory stanowią podstawowe struktury danych. Zawarte w nich informacje są nadawane/odbierane do/z urządzenia testowanego i podlegają odpowiedniej analizie oraz obróbce. Liczniki i stopery są strukturami pomocniczymi i umożliwiają łatwą organizację pętli wraz z warunkowaniem działań upływem czasu. Eksploatacyjna wersja SPS dopuszcza używanie 16 buforów o definiowanej długości, 8 stoperów zliczających z rozdzielczością 20 ms i 8 liczników 8-bitowych.

T a b e l a

Język programowania testów

Kod	Składnia	Opis
00	HALT	Zatrzymanie programu testu
01	EXIT	Koniec definicji programu testu
10	SEND Bi	Nadaj bufor Bi
20	RCEV Bj	Odbieraj dane do bufora Bj
30	CMP Bi, Bj	Porównaj dane w buforach Bi i Bj
31	MASK Bi, Bj	Zeruj bity w Bi zgodnie z wzorcem w Bj
32	TEST Bi, Bj	Zlicz w liczniku C0 różnicę bitów między Bi i Bj
40	JUMP NK	Skocz bezwarunkowo do NK
41	JPEQ NK	Skocz do NK, gdy bufor jest jednakowy
42	JPNQ NK	Skocz do NK, gdy bufor jest niejednakowy
43	JNS NK	Skocz do NK, gdy nadawanie niepoprawne
44	JNR NK	Skocz do NK, gdy odbiór niepoprawny
45	JLEQ NK	Skocz do NK, gdy Ci równy 0
46	JNLQ NK	Skocz do NK, gdy Ci nierówny 0
47	JPTQ NK	Skocz do NK, gdy Ti równy 0
50	WAIT n	Czekaj n * 20 ms, n ≤ 255
51	RUNT n, Ti	Wpisz n do stopera Ti i odblokuj zliczanie
52	SET n, Ci	Wpisz n do licznika Ci
53	STPT Ti	Zatrzymaj stoper Ti
60	CLR Ci	Zeruj licznik Ci
62	INC Ci	Dodaj 1 do Ci
63	DEC Ci	Odejmij 1 od Ci

Na rysunku 6 przedstawiono kod programu testu, w którym SPS, spełniając rolę DTE w protokole X.25/3, przeprowadza w wybranej linii procedurę restartu. Następuje ona po fazie nawiązania połączenia liniowego zgodnie z wymaganiami protokołu LAPB.

*TP

* PROGRAMMABLE TEST *

```

Line speed (500...9500)      = 2400
Modem speed (H/L)             = H
Line mod (H/D)                = D
Protocol class (C/B)          = B
Protocol (H/S)                = S
NRZI (Y/N)                    = N

```

INSERT TESTING PROGRAM

```

1      60      CLR      C0
2      20      RCEV      B0      ; odbierz ramkę
3      50      WAIT      10
4      44      JNR      3
5      62      INC      C0
6      30      CMP      B0,B1      ; odebrano SABM ?
7      42      JPNE      2      ; nie, skocz
8      10      SEND      B2      ; tak, nadaj UA
9      50      WAIT      2
10     43      JNS      9
11     20      RCEV      B3      ; odbierz ramkę
12     50      WAIT      3
13     44      JNR      12
14     30      CMP      B3,B4      ; odebr. RESTART REQUEST ?
15     41      JPEQ      20      ; tak, skocz
16     10      SEND      B5      ; nie, nadaj RR
17     50      WAIT      2
18     43      JNS      17
19     40      JUMP      11
20     10      SEND      B6      ; nadaj RESTART CONFIRMATION
21     50      WAIT      2
22     43      JNS      21
23     0       HALT
24     1       EXIT      ; koniec definicji

```

* * *

Rys. 6. Test programowy - procedura restartu według X.25/3
(w buforach B1, B2, B5 wstawiono wzorce ramek SABM,UA, RR
a w buforach B4, B6 wstawiono wzorce pakietów RESTART REQUEST i RESTART CONFIRMATION)

Fig. 6. The programmed test (restart procedure according X.25/3)

Programując sekwencje testowe użytkownik SPS ma do dyspozycji, oprócz instrukcji języka TPL, również inne komendy systemu. Służą one do podziału przestrzeni dostępnej pamięci na wymaganą liczbę buforów, zapisania i edycji danych w tych buforach, zapisania i edycji definicji testu oraz uzyskania raportu bieżącego i końcowego z realizacji testu programowego.

Należy tutaj podkreślić uniwersalność rozwiązania. SPS nie realizuje stałych, wbudowanych sekwencji testowych, lecz oferuje język i dane do generowania takich sekwencji. Istniejące standardowe sprzęgi SPS, przeznaczone do transmisji synchronicznej znakowej lub bi-

towej, umożliwiając testowanie urządzeń pracujących według różnych protokołów. Najczęściej stosowane sekwencje testowe mogą tworzyć bibliotekę procedur składowaną na urządzeniach zewnętrznych. Jedyne ograniczeniem implementacyjnym testu programowego jest to, że nie może on zawierać więcej niż 127 linii programu.

3. BUDOWA OPROGRAMOWANIA

SPS należy do klasy systemów działających w czasie rzeczywistym. Budowa oprogramowania aplikacyjnego dla takiego systemu opiera się zwykle na systemie operacyjnym gwarantującym współbieżność, synchronizację i komunikację procesów. Rozwiązaniem alternatywnym jest budowa specjalizowanego monitora, co zwykle gwarantuje krótszy czas reakcji systemu na pojedyncze przerwanie, krótszy czas przełączania procesów i mniejszy obszar pamięci przeznaczony na kod systemu (monitora). Te trzy istotne cechy monitora uzyskuje się kosztem rezygnacji z pewnych usług systemu operacyjnego stanowiących o jego uniwersalności, a zatem i przydatności dla wielu aplikacji. Budowę oprogramowania SPS oparto na specjalizowanym monitorze, rezygnując po analizie - z głębokich modyfikacji firmowego systemu DOPS zgodny z CP/M). W fachowej literaturze opisano wiele realizacji monitorów. Implementację monitora dla SPS oparto na pracy [6], adaptując modelowe rozwiązanie do innego sprzętu i rozszerzając listę usług zgodnie z wymaganiami procesów aplikacyjnych.

Oprogramowanie aplikacyjne zdekomponowano na pewną liczbę statycznych procesów, z których każdy realizował się na pewnej maszynie wirtualnej. Dla pojedynczego procesu maszyną taką tworzy procesor z pamięcią wspólną i dzieloną, system przerw (źródło pierwotnych zdarzeń w systemie) oraz monitor, dostarczający między innymi usług synchronizacji i komunikacji międzyprocesowej. Monitor, tworząc środowisko dla procesów kodowanych w języku makroasemblera, narzuca im strukturę. Elementy składowe struktury procesu przedstawiono na rys.7.

Jak to wynika z rys. 7, każdy z procesów ma trzy punkty wejścia. Wyznaczają one kod prologu, epilogu oraz startu "normalnego". Kod prologu każdego z procesów jest wykonywany w momencie (re)startu monitora i działanie to pozwala na ustalenie stanu początkowego procesu. Monitor aktywuje proces od etykiety epilogu, tj. DHi, jeżeli inny proces zażąda usunięcia podanego procesu z systemu. Monitor, uruchamiając proces od etykiety epilogu DHi, usuwa go ze wszystkich kolejek, w których taki proces mógł oczekiwać na spełnienie swoich żądań i kod epilogu musi gwarantować integralność danych usuwanego

	Priorytet
	Obszar pamiętania kontekstu
	Obszar komunikacyjny
	Obszar stosu procesu
	Obszar zmiennych procesu
DFi:	Kod procesu
DII:	Kod procesu - prolog
DHi:	Kod procesu - epilog

Rys. 7. Elementy składowe struktury procesu
Fig. 7. Elements of the process structure

procesu. Po procedurze (re)startu wszystkie procesy mają w swoich strukturach danych ustawiony stan początkowy i są gotowe do "normalnego" startu od etykiady DFi. Podczas wykonywania niektórych usług monitor przełącza procesy. Jeżeli proces zwalnia procesor, to jego ślad jest pamiętany w "obszarze pamiętania kontekstu".

3.1. Usługi monitora

W opisywanym systemie monitor tworzy środowisko i zarządza statyczną liczbą procesów o stałych priorytetach. W systemie jednoprocessorowym tylko jeden proces może w wyróżnionej chwili wykonywać się na procesorze. Pozostałe procesy przebywają w jednej z czterech kolejek, tj. procesora (gotowości), żądań aktywacji, czasowej i czasowo-zdarzeniowej. Monitor przesuwa procesy między kolejkami w wyniku realizacji zleconych mu usług, które z punktu widzenia procesu należą do kilku grup. Przede wszystkim służą one do:

- aktywacji/deaktywacji procesów,
- synchronizacji z innym procesem,
- przesyłania komunikatów między procesami,
- odmierzania czasu.

Każdy proces może zażądać aktywacji/deaktywacji innych procesów lub deaktywacji samego siebie. Wymaga to użycia usług monitora o następujących definicjach (I - identyfikator procesu):

ACTIVATE (I): jeżeli proces I nie jest zawieszony w kolejce czasowej lub czasowo-zdarzeniowej, to jest wstawiany do kolejki procesora wraz z procesem bieżącym; w przeciwnym razie proces I jest wstawiany do kolejki żądań aktywacji, a proces bieżący do kolejki procesora,

DEACTIVATE (I): proces I jest usuwany ze wszystkich kolejek i z etykietą epilogu jest wstawiany do kolejki procesora wraz z procesem bieżącym,

HOLD (n): proces bieżący zwalnia procesor i jest wstawiany do kolejki czasowej, skąd po upływie n tików jest przesuwany do kolejki procesora,

BYE: proces bieżący zwalnia procesor; jeżeli kolejka żądań aktywacji tego procesu jest niepusta, to jest zmniejszana o 1 i proces jest wstawiany do kolejki procesora.

W celu synchronizacji jednego procesu z innym wprowadzono pojęcie zdarzenia. Może być ono sygnalizowane (ustawiane), zerowane i oczekiwane przez pewien proces. Cechą wyróżniającą jest to, że monitor buforuje zdarzenia. Jeżeli przez E oznaczymy zdarzenie, to w tej grupie są dostępne usługi o następujących definicjach:

WAIT (E, n): jeżeli kolejka zdarzenia jest niepusta, to jest zmniejszana o 1, a proces zwalniając procesor jest ponownie wstawiany do kolejki procesora; jeżeli kolejka zdarzenia jest pusta, to proces zwalnia procesor i jest wstawiany do kolejki czasowo-zdarzeniowej na maksymalny okres n jednostek czasu (tików), a zdarzenie jest znakowane jako oczekiwane;

SEND (E): jeżeli zdarzenie jest oczekiwane, to jest zerowane, a procesy bieżący i oczekujący na E są przesuwane do kolejki procesora; jeżeli kolejka E jest pusta lub jest niepusta, a E nie jest oczekiwane, to kolejka E jest zwiększana o 1, a proces bieżący przechodzi do kolejki procesora;

CLEAR (E): kolejka E zostaje wyzerowana, a proces bieżący jest przesuwany do kolejki procesora.

Jeżeli proces jest wznawiany, to w jego obszarze komunikacyjnym jest informacja czy:

- zdarzenie E już oczekiwało na konsumenta,
- zdarzenie E wystąpiło przed czasem przeterminowania,
- czas minął, a zdarzenie E nie wystąpiło.

Są dwa sposoby komunikacji między procesami. Sposób pierwszy to dostęp do wspólnych danych dzielonych. Integralność takich danych nie jest gwarantowana przez monitor i musi być realizowana przez procesy

(co łatwo osiągnąć korzystając z usług WAIT(..) i SEND(..) . Sposób drugi to komunikacja przez przesyłanie buforów z wiadomościami (komunikatami). Bufory dla komunikatów są pobierane z puli buforów systemu. Są ich dwa rodzaje, tj. krótkie i długie o liczbie i rozmiarach ustalanych generacyjnie. Definicje usług monitora w tej grupie są następujące:

GETSHORT_BUF: jeżeli lista krótkich lub długich buforów systemu jest niepusta, to jest zmniejszana o 1 i adres bufora (p) jest przekazywany procesowi,

GETLONG_BUF: jeżeli lista długich buforów systemu jest niepusta, to jest zmniejszana o 1 i adres bufora (p) jest przekazany procesowi,

FREE_BUF(p): bufor o adresie p jest dołączony do listy buforów wolnych,

SEND_COMM(I,p): do procesu I jest przesyłany komunikat uformowany w buforze o adresie (p) bufor jest dopisywany do kolejki wejściowej wskazanego procesu),

CHECK_QUEUE: jeżeli kolejka wejściowa bieżącego procesu jest niepusta, to jest zmniejszana o 1 i adres pierwszego bufora jest przekazywany procesowi przez jego obszar komunikacyjny.

Procesy pobierając bufory z puli systemowej nie mają przy tych usługach jednakowych uprawnień. Uprawnienia wynikają z priorytetu procesu, który jest argumentem funkcji progowej rozstrzygającej czy dla danego procesu monitor ma bufory wolne.

Ostatnią grupę tworzą usługi wykorzystywane przez procesy do odmierzenia czasu. Są to:

STARTTIC_TIM(n,j): aktywacja czasomierza j z nastawą dla odliczenia n tików,

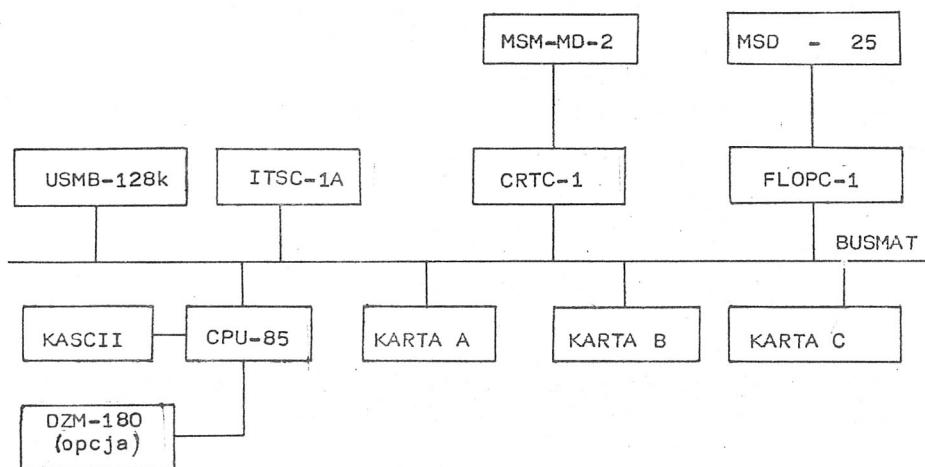
STOPTIC_TIM(j): wyłączenie czasomierza j,

STARTMIN_TIM(n): aktywacja minutnika z nastawą dla odliczenia n minut;

STOPMIN_TIM: wyłączenie minutnika.

4. MODUŁY SPRZĘTOWE

SPS został zbudowany z modułów systemu MSM [3] firmy Impol-1 oraz specjalizowanych kart sprzęgających [4] zaprojektowanych i wykonanych w Politechnice Wrocławskiej. Podstawowa konfiguracja SPS jest przedstawiona na rys. 8. Architektura systemu MSM, jak i funkcje jego standardowych modułów, są szczegółowo zdokumentowane w pracy [3].



Rys. 8. Podstawowa konfiguracja SPS
Fig. 8. Basic configuration of the SPS

Wszystkie moduły są umieszczone w kasecie wraz z zasilaczem impulsowym MSM-SPS-1 i połączone standardowym interfejsem międzypakietowym BUSMAT. Są to:

- moduł jednostki centralnej MSM-CPU-85 z procesorem 8085 pracującym z częstotliwością zegara 3.072 MHz
- moduł pamięci MSM-USMB 128k stanowiący pamięć operacyjną systemu zrealizowaną jako pamięć statyczną RAM o pojemności 64 KB
- moduł obsługi przerwań MSM-ITSC-1 stanowiący programowany sterownik ośmiu przerwań priorytetowych systemu
- moduł sterownika monitora ekranowego MSM-CRTC-1
- moduł sterownika pamięci na dyskach elastycznych MSM-FLOPC-1
- moduł sprzęgający do monitorowania obwodów styku S2 (karta A)
- moduł sprzęgający do monitorowania/sterowania transmisji asynchronicznej i synchronicznej znakowej (karta B)
- moduł sprzęgający do monitorowania/sterowania transmisji synchronicznej bitowej (karta C).

Standardowe urządzenia zewnętrzne systemu SPS stanowią:

- klawiatura alfanumeryczna MSM-KASCII dołączona do modułu CPU-85
- monitor monochromatyczny MSM-MD-2 dołączony do modułu CRTC-1.

Karta A służy do monitorowania obwodów styku S2 w programowanych odstępach czasu. Blok portu wejściowego oraz blok programowanego generatora przerwań zbudowano z wykorzystaniem odpowiednio układu 8255 i układu licznika 8253. Wbudowany blok konwersji poziomów sygnałów pozwa-

la współpracować z modemami typu EC 8013 i EC 8006 lub innymi zgodnymi z nimi mechanicznie, elektrycznie i funkcjonalnie. Kaskadowe połączenie dwóch liczników układu 8253 pozwala uzyskać zakres czasów między próbkowaniami od kilku mikrosekund do 2000 s. Ograniczenia programowe podnoszą jednak minimalny próg do wartości 50 mikrosekund.

Karta B służy do monitorowania lub sterowania transmisją danych przez styk S2 dla protokołów asynchronicznych i synchronicznych znakowych. Blok transmisji zbudowano z użyciem dwóch układów 8251A, które w zależności od trybu pracy są wykorzystywane jako układ nadajnika - odbiornika lub dwa układy odbiorników. Karta ma własny blok zegara (dwa liczniki układu 8253), co umożliwia programowy wybór wewnętrznej lub zewnętrznej (modemowej) podstawy czasu. Systemowy port wejściowy umożliwia odczyt wybranych linii styku S2, a port wyjściowy umożliwia sterowanie obwodem 111, wybór podstawy czasu, maskowanie i otwieranie przerwań z bloku transmisji. Blok konwersji poziomów sygnałów jest analogiczny jak w karcie A.

Karta C służy do monitorowania lub sterowania transmisją danych przez styk S2 dla protokołów synchronicznych bitowych. Blok transmisji zbudowano z użyciem dwóch układów 8273, które w zależności od trybu pracy są wykorzystywane jako układ nadajnika-odbiornika lub dwa układy odbiorników. Zaprojektowana konfiguracja umożliwia obsługę układów 8273 trzema sposobami, lecz w SPS układy te są obsługiwane w trybie programowego DMA. Karta ma własny oscylator kwarcowy o częstotliwości 4 MHz i blok zegara zbudowany z liczników układu 8253. Systemowy port wejściowy wraz z portami wewnętrznymi układów umożliwia odczyt stanu wybranych linii styku S2, co zapewnia łatwą diagnostykę zachowań modemu. Port wyjściowy umożliwia wybór podstawy czasu oraz maskowanie i otwieranie przerwań od układów 8273. Blok konwersji poziomów sygnałów jest analogiczny jak w karcie A.

5. OCENA IMPLEMENTACJI

Do budowy oprogramowania zastosowano dość prymitywne narzędzie jakim jest język makroasemblera. W rezultacie wymagane było napisanie prawie 22 tysięcy linii programu źródłowego. Kod wynikowy wszystkich procedur monitorowania oraz testowania zajął odpowiednio 34 i 26 CB pamięci operacyjnej nie licząc obszarów na bufory. Pięćosobowy zespół projektowy pracował nad tym zagadnieniem przez 1,5 roku.

Podstawową cechą użytkową budowanej aparatury miała być jej zdolność do monitorowania i testowania transmisji dwukierunkowej w całym typie szeregu aż do wartości 9600 b/s włącznie. Szacunkowe obliczenia

dokonane w fazie projektowej nie dyskwalifikowały osiągnięcia tej szybkości. Przeprowadzono cały cykl testowań definiując różną liczbę pułapek monitorowania, a tym samym wpływając na czas procedur obsługi przerw i analizy danych gromadzonych do analizy w czasie rzeczywistym. We wszystkich tych przypadkach, łącznie z szybkością 9600 b/s, SPS nie rejestrował stanu przeciążenia. W trakcie badań sprawdzano również ergonomię obsługi SPS i czytelność generowanych wyników. Gromadzono także uwagi na ten temat osób spoza kręgu realizatorów. Okazało się, że niektóre założenia, a w rezultacie i wykonanie, zostały podporządkowane "łatwości" realizacji a nie wygodzie użytkownika. Zmodernizowana, obecna wersja 1.20 oprogramowania, jest częściowo wolna od tych niedomagań jakkolwiek pewne zagadnienia zostały odłożone do momentu budowy urządzenia nowej generacji, opartego na nowych typach kontrolerów w kartach sprzęgających. Kontrolery te, np. 8274 lub 82530 w połączeniu z mikroprocesorami serii iAPX88/86/186 i układami sprzętowego DMA, umożliwią zbudowanie pojedynczej karty integrującej funkcje trzech kart dotychczasowych, tj. A, B i C i pozwolą na uzyskanie nowych, zwiększonych wartości parametrów użytkowych. Istniejącą wersję SPS można natomiast, ze względu na budowę modułową, łatwo wzbogacać o nowe funkcje. Dotyczy to przede wszystkim monitorowania protokołów kolejnych warstw, tj. transportowego, sesyjnego itd. Rozbudowa oprogramowania realizującego te funkcje jest prowadzona w ramach dyplomowych prac magisterskich.

LITERATURA

- [1] Data Analyzer DA-10, Prospekt firmy Mandel & Golterman, 1981.
- [2] Data Communications Test Sets, Prospekt firmy Electrodata Inc.
- [3] Dokumentacja systemu MSM, PPZ Impol-1, Warszawa.
- [4] DUBIELEWICZ I., FRYZLEWICZ Z., KOLEŚNIK K., KOMOROWSKI W., MISIURA L., Aparatura pomiarowo-diagnostyczna dla sieci SKJS/2. Dokumentacja eksploatacyjna, Raport SPR 12/86, CO Pwr., Wrocław 1986.
- [5] Measurements, Computation, Systems: Katalog firmy Hewlett-Packard 1984.
- [6] THORELLI L.E., A Monitor for Small Computers, Software-Practice and Experience, Vol. B, s. 439-450, 1978.

Praca wpłynęła do Redakcji 1988.03.23

THE MICROCOMPUTER BASED PROTOCOL TESTER AND ANALYZER

Based on the standard MSM hardware the protocol tester and analyzer is designed for applications to a WAN communication protocols. Three additional cards are designed and shortly described as well as all the software. This one is decomposed to the processes and the monitor which enables concurrent and real time processing. The monitor's services are summerized and structural aspects of processes are given.

ИЗМЕРИТЕЛЬНО-ДИАГНОСТИЧЕСКАЯ АППАРАТУРА В ВЫЧИСЛИТЕЛЬНОЙ СЕТИ

Сконструированная аппаратура исполняет функции анализатора/тестера протоколов в сети WAN. Устройство сконструировано путем пристраивания к системе MSM трех дополнительных карт-пакетов и сделан проект программного обеспечения, состоящего из процессов и монитора, который создает для них рабочую среду для синхронной работы в реальном времени. В статье даны функциональные черты устройства и принципы сотрудничества процессов в этой среде.

*Sieć komputerowa,
pomiar, przepustowość,
opóźnienie tranzytowe*

Kazimierz DYRKA*
Piotr KREMIENIOWSKI*
Edward RUTKOWSKI*

CHARAKTERYSTYKI CZASOWE I PRZEPUSTOWOŚĆ W SIECI SKJS/2 WERSJA 1

W artykule przedstawiono wyniki pomiarów niektórych charakterystyk ilościowych sieci komputerowej SKJS/2 wersja 1. Omówiono metody i narzędzia pomiarów ze szczególnym uwzględnieniem pomiarów na poziomie programów aplikacyjnych wykonywanych w komputerach obliczeniowych. Określono warunki realizacji pomiarów.

1. WSTĘP

W sieci komputerowej z komutacją pakietów podczas przesyłania komunikatów zawsze powstają opóźnienia będące sumą czasów transmisji tych komunikatów przez linie i inne elementy pośredniczące.

Jednym z najważniejszych parametrów charakteryzujących sieć komputerową jest opóźnienie przesyłanych przez nią komunikatów. Opóźnienie to zależy od natężenia strumieni przesyłanych komunikatów, przepustowości elementów pośredniczących (linii i węzłów) oraz przepustowości kanałów, sposobu obsługi pakietów w komputerach obliczeniowych, a także od właściwości wykorzystywanych systemów operacyjnych i metod dostępu. Oczywiście opóźnienie zależy także od mocy obliczeniowej i obciążenia użytkowanych komputerów obliczeniowych.

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

W wersji 1 SKJS/2 zrealizowana jest usługa przesyłania zbiorów między komputerami obliczeniowymi. Usługę tę realizuje tak zwana Stacja Transferu Zbiorów wraz ze sprzęgami międzyprogramowymi, zaimplementowanymi w komputerach obliczeniowych. Czasy wykonania usługi przesyłania zbiorów zależą od wielkości zbiorów, a także od wymienionych czynników wpływających na opóźnienia przesyłania komunikatów.

Problemy analizy, modelowania i optymalizacji sieci komputerowych ujęte są w pracy [4], gdzie opisano i przeanalizowano wyniki bezpośrednich pomiarów sieci. Natomiast w artykule [1] omówiono wyniki pomiarów i statystyki sieci MERI7, dokonywane w ciągu dziesięcioletniej pracy tej sieci. Analiza wymienionych prac i innych potwierdziła tezę, że przepustowość elementów sieci oraz całej sieci bardzo silnie zależy od sprzętu i oprogramowania i może być oceniana dopiero w konkretnej realizacji sieci.

W dalszym ciągu tego artykułu podane będą wyniki bezpośrednich pomiarów niektórych charakterystyk czasowych i przepustowość sieci SKJS/2 wersji 1.

2. POZIOMY POMIARÓW SKJS/2

Pomiary mogą być dokonywane z użyciem środków sprzętowych lub programowych na poziomie podsieci komunikacyjnej i komputerów obliczeniowych. W wersji 1 SKJS/2 dokonano pomiarów na poziomie podsieci komunikacyjnej za pomocą specjalnego sprzętu - aparatury pomiarowo-diagnostycznej. Natomiast na poziomie komputerów obliczeniowych pomiary zostały wykonane za pomocą specjalnego oprogramowania.

2.1. Pomiary na poziomie podsieci komunikacyjnej

Pomiary na tym poziomie są dokonywane za pomocą aparatury pomiarowo-diagnostycznej, która jest specjalnie skonstruowana dla sieci SKJS/2 wersja 1 [3]. Miejscem włączenia urządzenia do sieci jest styk S2 (między DTE i DCE). Urządzenie może być włączone szeregowo lub równolegle.

Konstrukcja systemu oparta jest na mikrokomputerze oraz specjalnych modułach umożliwiających sprzęg systemu ze stykiem S2. W skład systemu wchodzi specjalne oprogramowanie.

System SPS umożliwia testowanie:

- w czasie normalnej pracy
- w warunkach symulacji DTE
- sygnałów styku DTE-DCE.

Do badania przepustowości i natężenia może być wykorzystana funkcja pomiarowa SPS Monitorowanie transmisji danych polegające na rejestrowaniu strumieni danych płynących do i z DTE. Wykorzystując tę funkcję uzyskujemy następujące wyniki statystyczne dla każdego strumienia osobno:

- liczbę zarejestrowanych jednostek protokołu
- liczbę zarejestrowanych błędów
- liczbę wyróżnionych typów jednostek protokołu (ramek, pakietów)
- liczbę jednostek dostępnych w buforze.

Przy tych wynikach jest podanych czas rozpoczęcia i zakończenia oraz czas aktualny. Wyniki te są również bardzo pomocne w weryfikacji pomiarów wykonywanych innymi metodami.

W sieci SKJS/2 badano natężenie przepływu między dwoma współpracującymi RTD; wykorzystując program generujący ruch w sieci oraz system SPS. Przy natężeniu w linii około 2000 b/s udział informacji wysyłanych przez program użytkowy wynosił 80%, przy natężeniu 2300 b/s udział informacji wysyłanych przez program użytkowy wynosił 90%.

2.2. Pomiary na poziomie komputerów obliczeniowych

Podstawową usługą sieci jest przesyłanie komunikatów między nadawcą a odbiorcą. Inną istotną usługą jest przesyłanie zbiorów między dwoma komputerami obliczeniowymi. Czasy wykonywania tych usług są najważniejszymi parametrami charakteryzującymi daną sieć.

Dla sieci SKJS/2 wersja 1 dokonano pomiarów tych podstawowych charakterystyk czasowych. W dalszej części tego punktu w rozbiciu na te dwa rodzaje usług podano definicje mierzonych wielkości, metodykę badań, narzędzia służące do pomiarów oraz omówiono wyniki pomiarów.

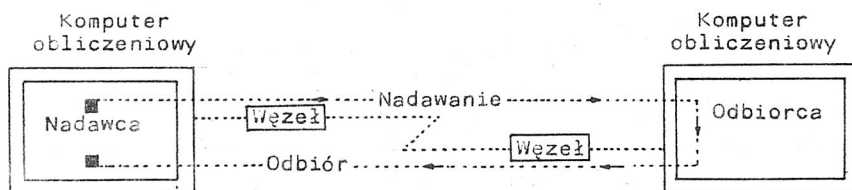
2.2.1. Przesyłanie komunikatów

W zakresie usługi przesyłania komunikatów na poziomie komputerów obliczeniowych dokonano następujących pomiarów: czasu opóźnienia tranzytowego oraz natężenia przepływu.

Opóźnienie tranzytowe jest to parametr techniczny sieci określany jako czas potrzebny na przesłanie komunikatu o określonej długości między nadawcą (użytkownikiem lub aplikacją), zlokalizowanym w danym komputerze obliczeniowym, a odbiorcą (użytkownikiem lub aplikacją), zlokalizowanym w innym komputerze obliczeniowym.

Natężenie przepływu określane jest jako stosunek ilości przesłanych danych (długości komunikatu) do czasu ich przesłania między nadawcą i odbiorcą. Natężenie jest podawane jako liczba przesłanych bajtów między nadawcą a odbiorcą w czasie jednej sekundy.

Pomiar opóźnienia tranzytowego dokonywany był przez dokładne zmierzenie czasu przesyłania komunikatów określonej długości między dwoma programami umiejscowionymi w dwóch komputerach obliczeniowych. Ponieważ nie ma możliwości dokładnej synchronizacji zegarów maszynowych komputerów obliczeniowych pracujących w sieci, pomiar czasu był dokonywany w komputerze wysyłającym komunikat, określanym jako nadawca. Natomiast program odbiorca, znajdujący się w drugim komputerze obliczeniowym, otrzymane komunikaty natychmiast wysyłał z powrotem do nadawcy. Zasadę współdziałania obu programów zilustrowano na rys. 1.



Rys. 1. Przepływ komunikatów między nadawcą a odbiorcą przez elementy pośredniczące

Fig. 1. Flow of messages between sender and receiver through intermediate elements

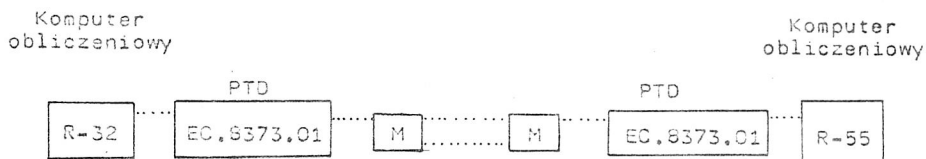
Przebieg pomiaru jest następujący:

- program nadawca wysyła komunikat do programu odbiorcy i rozpoczyna liczenie czasu
- komunikat jest dzielony na pakiety i przesyłany przez podsieć komunikacyjną; po dotarciu do odbiorcy komunikat jest natychmiast wysyłany z powrotem do nadawcy
- po odebraniu komunikatu nadawca kończy liczenie czasu.

Praktycznie komunikat jest zatem przesyłany w obie strony po tych samych łączach. Przyjmuje się, że czas opóźnienia tranzytowego wynosi połowę zmierzonego czasu. Pomiary są dokonywane z dokładnością zegara maszynowego, tj. 26.04166 ms. Aby uniknąć przypadkowości wyników, pomiarów dokonuje się wielokrotnie (n razy), wylicza się wartość średnią opóźnienia oraz podaje się wartości minimalne i maksymalne.

Pomiary zostały wykonane na pilotowej instalacji sieci SKJS/2 wersja 1, której schemat jest przedstawiony na rys. 2.

Komputery obliczeniowe R-32 i R-55 były połączone przez procesory teleprzetwarzania danych (PTD) i modemy pracujące z szybkością 2400 b/s. Modemy były połączone symulowaną linią telefoniczną, co zmniejsza wartość uzyskanych pomiarów. Wyniki z trzech różnych pomiarów pokazano na rys. 3. Pomiary te przeprowadzono z różnymi liczbami



Rys.2. Schemat pilotowej instalacji sieci SKJS/2 wersja 1

Fig.2. Configuration of the prototype instalation of SKJS/2 network version 1

powtórzeń (n). Na wykresie podano wartości średnie. Linia ciągłą połączone są punkty wyników pomiarów opóźnienia tranzytowego dla komunikatów o długościach 100, 200, 300, ..., 2000 bajtów. Pomiary były powtarzane trzykrotnie. Linia przerywaną połączone są punkty wyników pomiarów komunikatów o długościach 100, 300, 500... 1900 bajtów. Natomiast kropkami połączone są punkty wyników pomiarów i komunikatów o długościach 124, 355 (zwiększanych o 211), aż do 2023 bajtów. Z wykresów widać liniową zależność opóźnienia tranzytowego od długości przesyłanego komunikatu. Widoczne są wahania wielkości opóźnienia tranzytowego dla komunikatów o tej samej długości, co jest spowodowane wykonywaniem w tym czasie przez system operacyjny operacji systemowych.

Pomiar natężenia przepływu dokonywany był w warunkach symulowanego ruchu w sieci za pomocą dwóch programów umieszczonych w komputerach R-32 i R-55 instalacji pilotowej (rys. 2). Po zainicjowaniu pracy przez jeden z programów przez wysłanie komunikatu, programy wzajemnie wysyłają ten komunikat natychmiast po otrzymaniu. Zatem komunikat jest "odbijany" z maksymalną prędkością na jaką pozwala łącze komunikacyjne. Zliczana jest liczba odbić. Natężenie przepływu jest wyliczane ze wzoru (w bajtach na sekundę).

$$\text{Nat} = \frac{2nd}{p} ,$$

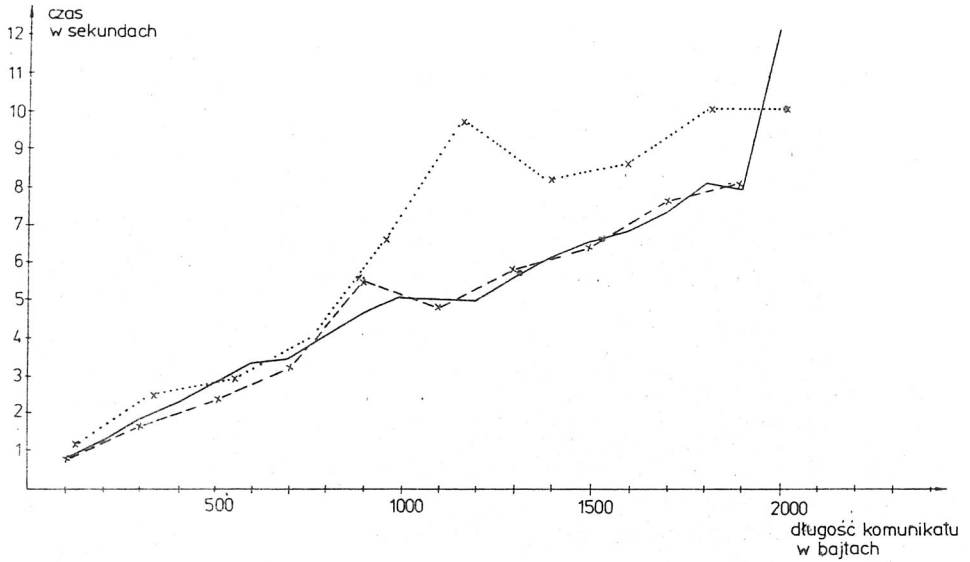
gdzie:

n - liczba odbić,

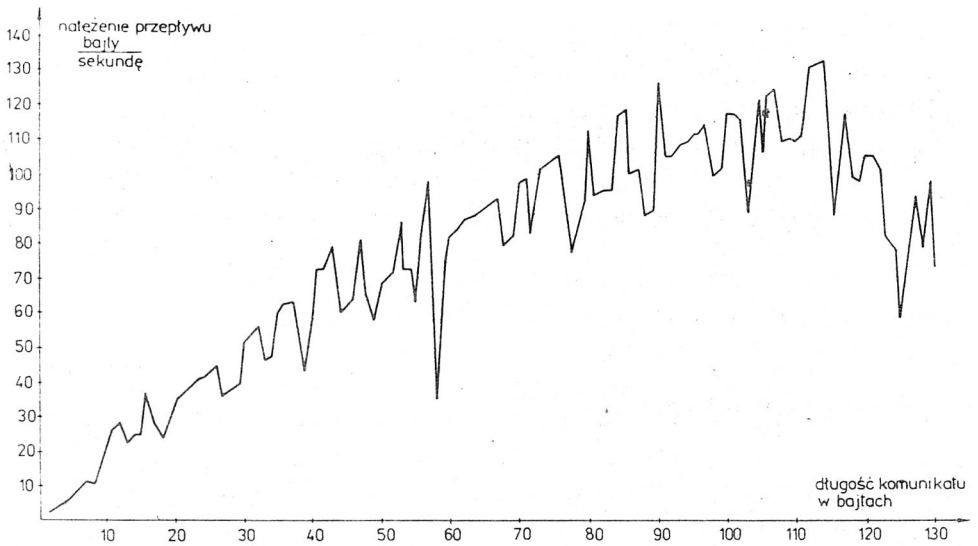
d - długość komunikatu,

p - czas odbijania.

Na rysunku 4 podano natężenie przepływu dla krótkich komunikatów z przedziału 1-130 bajtów. Na wykresie widać charakterystyczne załamanie się wielkości natężenia dla komunikatów, których długość przekroczyła długość pakietu w SKJS/2 wersja 1, wynoszącą 128 bajtów.



Rys.3. Zależność czasu opóźnienia tranzytowego od długości komunikatu
 Fig.3. Dependence of transit delay from message length



Rys.4. Natężenie przepływu w zależności od długości komunikatu
 Fig.4. Flow intensity as function of message length

2.2.2. Przesyłanie zbiorów

Podstawowym narzędziem w sieci SKJS/2 wersja 1 do realizacji usługi przesyłania zbiorów jest System Transferu Zbiorów. Jego własności w dużej mierze stanowią o walorach użytkowych sieci.

Szczegółowy program prób i badań oraz narzędzia i metody przedstawiono w pracy [2].

Należy zwrócić uwagę, że ocena uzyskanych wyników musi uwzględniać cechy funkcjonalne zaimplementowanego systemu. Zwłaszcza dla systemu Transferu Zbiorów w sieci SKJS/2 wersja 1 należy w analizie wyników uwzględnić kompresję danych znakowych i możliwość wznowień przerwanych transferów w wyniku awarii sprzętu.

Przy badaniach ilościowych charakterystyk systemu zwrócono przede wszystkim uwagę na :

a) czas przesyłania zbioru zdefiniowany jako czas przesyłania informacji z pamięci dyskowej jednego komputera do pamięci dyskowej innego komputera obliczeniowego; czas przesyłania zbioru jest liczony od momentu wydania pierwszej komendy (nawiązanie połączenia) do momentu wydania ostatniej komendy (rozwiązanie połączenia),

b) natężenie przepływu danych zdefiniowane jak w p-ktcie 2.1.

Pomiary wykonano w warunkach pracy sieciowej dla zbiorów sekwencyjnych zblokowanych o stałej i zmiennej długości rekordów. Pomiary czasu dokonywane są z dokładnością zegara maszynowego.

Średni czas przesyłania zbioru zmierzono dla zbiorów o wielkościach 2 kB i 1 kB i otrzymano wyniki dla zbioru 2 kB - 50 s, dla zbioru 1 kB - 70 s.

Pomiary natężenia przepływu danych wykonano dla zbiorów o wielkościach od 2 kB do 30 kB. Wartości średnie otrzymanych wyników zawarte są w przedziale od 700 b/s do 1800 b/s, przy czym dla zbiorów krótkich przepustowość maleje ze względu na narzuty czasowe związane z nawiązywaniem połączeń, negocjacją warunków transferu i ustaleniem końcowego statusu zakończenia transferu.

3. UWAGI KOŃCOWE

Uzyskane w czasie badań wyniki stanowią mogą podstawę wstępnej oceny charakterystyk ilościowych. Występujące w konfiguracji eksperymentalnej odstępstwa od konfiguracji rzeczywistej (dwa komputery, mała odległość między poszczególnymi systemami, zastosowanie sprzężenia galwanicznego między PTD z użyciem par przewodów modemowych i zasymu-

lowania zegara modemowego przez zewnętrzny generator spowodowały zmniejszenie błędów transmisji i utrudniły do pewnego stopnia wnioskowanie o rzeczywistej przepustowości połączeń.

Istnieje więc potrzeba weryfikacji wyników i rozszerzenie zakresu badań instalacji użytkowej opartej na istniejącym sprzęcie pomiarowo-diagnostycznym i oprogramowaniu pomiarowym, jak również informacjach dostępnych w kronice pracy sieci.

LITERATURA

- [1] AUPPERIE E.M., Merit's evolution - statistically speaking, IEEE Trans. on Computers. Vol. c-32, No. 10, October 1983.
- [2] DUBIELEWICZ A. i in., Program prób i badań usługi transferu zbiorów dla SKJS2 wersja 1., CO PWr., Raport SPR 8/86.
- [3] DUBIELEWICZ I., FRYŻLEWICZ Z., KOMOROWSKI W., Założenia techniczno-ekonomiczne na aparaturę pomiarowo-diagnostyczną, CO PWr., Raport SPR 13/86, Wrocław 1986.
- [4] KLEINROCK L. Queueing systems. Vol. II: Computer Applications, New York, Wiley, 1976.

TIME AND THROUGHPUT CHARACTERISTICS OF SKJS/2 VERSION 1 COMPUTER NETWORK

This paper is intended to familiarize the reader with the result of direct measuring of computer network SKJS/2 version 1. Levels of measurement, hardware and software tools for measurement are shortly presented and the results of measurement transitive delay, flow intensity and file transfer delay are given.

ЧИСЛОВЫЕ ХАРАКТЕРИСТИКИ И ПРОПУСКНАЯ СПОСОБНОСТЬ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ SKJS2 ВЕРСИЯ 1

В статье представлены результаты числовых измерений некоторых характеристик вычислительной сети SKJS2 версия 1. Описаны методы и условия измерений, в частности измерения на уровне прикладных программ. Представлены условия реализации измерений.

SPIS TREŚCI

Przedmowa	3
A. HUZAR, A. KALIŚ, Sieć komputerowa Jednolitego Systemu SKJS2 wersja 1	5
L. BUDZIANOWSKI, J. WIETRZYCH, A. FRYŚ, Struktura warstwowa a standardy w sieci SKJS2 wersja 1	13
A. KALIŚ, A. HUZAR, Tworzenie oprogramowania bazowego sieci komputerowej SKJS2 wersja 1	27
M. JACUKOWICZ, A. JANISZEWSKI, J. KWIATKOWSKI, Dostęp termi- nalowy w sieci SKJS2 wersja 1	45
J. STAŃKO, D. ŻAK, Styki a sterowanie przepływem w sieci SKJS2 wersja 1	53
A. HUZAR, A. KALIŚ, Problemy synchronizacji w stacji trans- feru zbiorów sieci komputerowej SKJS2 wersja 1.....	65
Z. FRYŻLEWICZ, Aparatura pomiarowo-diagnostyczna dla sieci komputerowej	73
K. DYRKA, P. KREMIENOWSKI, E. RUTKOWSKI, Charakterystyki cza- sowe i przepustowość w sieci SKJS2 wersja 1	91

CONTENTS

Preface	3
A. HUZAR, A. KALIŚ, The computer network SKJS2 version 1	5
L. BUDZIANOWSKI, J. WIETRZYCH, A. FRYŚ, Layer structure and standards in computer network SKJS2	13
A. KALIŚ, A. HUZAR, The creation of basic software of the network SKJS2 version 1	27
M. JACUKOWICZ, A. JANISZEWSKI, J. KWIATKOWSKI, Terminal access to the SKJS2 computer network	45
J. STAŃKO, D. ŻAK, The interfaces and flow control in SKJS2 network	53
A. HUZAR, A. KALIŚ, Some synchronization problems in the file transfer station on the SKJS2 network	65
Z. FRYZLEWICZ, The microcomputer based protocol tester and analyzer	73
K. DYRKA, P. KREMIENOWSKI, E. RUTKOWSKI, Time and throughput characteristics of SKJS2 computer network	91

PRACE NAUKOWE CENTRUM OBLICZENIOWEGO
(wydane w latach 1985-1987)

Nr 3, Konferencje nr 1, <i>Sieci komputerowe — Usługi, protokoły, modele</i> , cz. I, Wrocław 1985	160,—
Nr 4, Konferencje nr 2, <i>Sieci komputerowe — Usługi protokoły, modele</i> , cz. II, Wrocław 1986	170,—
Nr 5, Konferencje nr 3, <i>Sieci komputerowe — Teoria, technika, zastosowa- nia</i> , Wrocław 1987	335,—



Cena zł 440,—

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 3, 50-007 Wrocław

ISSN 0860-1623