P 4193/02

STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*
Quarterly

Volume 23, Number 1 (47)

Andrzej KWIECIEŃ

ANALIZA PRZEPŁYWU INFORMACJI W KOMPUTEROWYCH SIECIACH PRZEMYSŁOWYCH

Silesian University of Technology Press
Gliwice 2002

P. 4193/02

STUDIA INFORMATICA

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*
Quarterly

Volume 23, Number 1 (47)

Andrzej KWIECIEŃ

ANALIZA PRZEPŁYWU INFORMACJI W KOMPUTEROWYCH SIECIACH PRZEMYSŁOWYCH

Silesian University of Technology Press
Gliwice 2002

STUDIA INFORMATICA **Volume 23, Number 1 (47)**

Formerly: *Zeszyty Naukowe Politechniki Śląskiej, seria INFORMATYKA*

Editor in Chief

Dr. Marcin SKOWRONEK
Silesian University of Technology
Gliwice, Poland

Editorial Board

Dr. Mauro CISLAGHI
Project Automation
Monza, Italy

Prof. Bernard COURTOIS
Lab. TIMA
Grenoble, France

Prof. Tadeusz CZACHÓRSKI
Silesian University of Technology
Gliwice, Poland

Prof. Jean-Michel FOURNEAU
Université de Versailles - St. Quentin
Versailles, France

Prof. Jurij KOROSTIL
IPME NAN Ukraina
Kiev, Ukraine

Dr. George P. KOWALCZYK
Networks Integrators Associates, President
Parkland, USA

Prof. Peter NEUMANN
Otto-von-Guericke Universität
Barleben, Germany

Prof. Olgierd A. PALUSINSKI
University of Arizona
Tucson, USA

Prof. Svetlana V. PROKOPCHINA
Scientific Research Institute BITIS
Sankt-Petersburg, Russia

Prof. Karl REISS
Universität Karlsruhe
Karlsruhe, Germany

Prof. Jean-Marc TOULOTTE
Université des Sciences et Technologies de Lille
Villeneuve d'Ascq, France

Prof. Sarma B. K. VRUDHULA
University of Arizona
Tucson, USA

Prof. Hamid VAKILZADIAN
University of Nebraska-Lincoln
Lincoln, USA

Prof. Adam WOLISZ
Technical University of Berlin
Berlin, Germany

Dr. Lech ZNAMIROWSKI
Silesian University of Technology
Gliwice, Poland

© Copyright by Silesian University of Technology Press, Gliwice 2002
PL ISSN 0208-7286
Printed in Poland

ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ

OPINIODAWCY:
Dr hab. inż. Zbigniew Huzar, Prof. nadzw. Politechniki Wrocławskiej
Dr hab. inż. Henryk Zygmunt, Prof. Akademii Górniczo-Hutniczej

KOLEGIUM REDAKCYJNE
REDAKTOR NACZELNY – Prof. dr hab. Zygmunt Kleszczewski
REDAKTOR DZIAŁU – Dr inż. Marcin Skowronek
SEKRETARZ REDAKCJI – Mgr Elżbieta Leśko

REDAKCJA
Mgr Roma Łoś

REDAKCJA TECHNICZNA
Alicja Nowacka

SPIS TREŚCI

1. Wstęp 9
1.1. Cel pracy 14
2. Sieciowe, przemysłowe systemy rozproszone czasu rzeczywistego – cechy i wymagania 16
2.1. Problemy projektowania rozproszonych systemów czasu rzeczywistego 19
2.2. Przemysłowe rozproszone systemy czasu rzeczywistego 20
2.3. Węzeł systemu 22
2.4. Protokół komunikacyjny 23
2.5. Stacja operatorska 24
2.6. Podsumowanie 25
3. Rozproszony system sterowania 28
3.1. Sterownik swobodnie programowalny jako podstawowy element układu sterowania 35
3.2. Koprocesor sieci 37
3.3. Protokoły dostępu do łącza stosowane w sieciach przemysłowych 41
4. Sieci o dostępie typu TOKEN –BUS 42
4.1. Opis wymiany informacji 43
4.2. Powstawanie „dziur" w sieci 44
4.3. Kontrola czasu nadawania 47
4.4. Cykl sieci a cykl wymiany informacji 49
4.5. Określenie czasu trwania cyklu sieci 50
4.6. Maksymalny czas wymiany żetonu 50
4.7. Przypadek sieci z „dziurami" 53
4.8. Maksymalny czas trwania cyklu sieci z transmisją danych użytkowych 54
4.9. Buforowanie transmisji 60
4.10. Sprawność sieci i jej przepustowość użyteczna 61
4.11. Poprawa parametrów czasowych wymian 63
5. Sieci o dostępie MASTER–SLAVE 67
5.1. Przygotowanie scenariusza wymian i opis wymiany informacji 68
5.2. Powstawanie „dziur" w sieci 70
5.3. Parametryzacja wymian 72
5.4. Zjawiska zachodzące na styku „koprocesor ← jednostka centralna" 74
5.5. Określenie czasu trwania cyklu sieci 78
5.6. Sprawność sieci i jej przepustowość użyteczna 85
5.7. Poprawa parametrów czasowych wymian 87
6. Sieci o dostępie Producent – Dystrybutor – Konsument 89
6.1. Model typu PRODUCENT – DYSTRYBUTOR – KONSUMENT 90
6.2. Zasada działania sieci wykorzystującej protokół PDK 91
6.3. Struktura zmiennych 95
6.4. Transakcja wymiany periodycznej 97

6.5. Żądanie aperiodycznej transakcji zmiennej ... 98
6.6. Żądanie transmisji komunikatów ... 99
6.7. Analiza parametrów czasowych związanych z protokołem sieci FIP ... 101
6.8. Analiza czasowa pracy protokołu ... 103
6.8.1. Analiza statyczna ... 104
6.8.2. Analiza dynamiczna ... 106
6.9. Analiza bitowa transakcji ... 114
6.9.1. Bitowa reprezentacja ramek ... 114
6.9.2. Transakcja periodyczna ... 115
6.9.3. Transakcja aperiodyczna ... 116
6.10. Podsumowanie analizy i kontrola realizowalności wymian ... 116
6.11. Poprawa parametrów czasowych wymian w sieci typu FIP ... 119
6.12. Ocena wydajności i sprawności sieci FIP ... 124
6.13. Podsumowanie ... 129
7. Poprawa parametrów pracy sieci przemysłowych z cyklicznymi transakcjami wymiany informacji ... 131
7.1. Makrocykl wymiany informacji ... 132
7.2. Dobór parametrów makrocyklu ... 136
7.3. Metody budowy i optymalizacji makrocykli ... 138
7.3.1. Metody algorytmiczne ... 139
7.3.2. Metoda oparta na sztucznej inteligencji ... 142
7.4. Realizacja praktyczna algorytmów i wnioski ... 146
8. Metoda klasyfikacji protokołów komunikacyjnych w sieciach przemysłowych ... 148
8.1. Klasyfikacja sieci ... 153
8.2. Kryteria doboru ... 155
8.3. Dobór rozwiązania ... 155
8.4. Podsumowanie ... 157
9. Metody integracji sieci przemysłowych najniższego poziomu ... 158
9.1. Integracja sieci na poziomie I ... 159
9.2. Integracja sieci na poziomie II ... 162
9.3. Integracja sieci na poziomie III ... 164
9.4. Przykład integracji ... 164
9.5. Podsumowanie ... 169
10. Zastosowanie protokołu TCP/IP w sieciach przemysłowych najniższego poziomu ... 171
10.1. Protokół TCP/IP w sieciach przemysłowych ... 172
10.2. Wpływ architektury systemu na parametry sieci przemysłowych najniższego poziomu ... 174
10.3. Podsumowanie ... 181
11. Przegląd rozwiązań sieci przemysłowych ... 184
11.1. Sieć Modbus ... 185
11.2. Sieć Genius/N-80 ... 185
11.3. Sieć LonWorks ... 186
11.4. Sieć CAN (Controller Area Network) ... 186
11.5. Sieć SDS (Smart Distributed System) ... 187
11.6. Sieć DeviceNet ... 187
11.7. Sieć HART ... 188
11.8. Sieć FIP/ WorldFIP ... 188
11.9. Sieć Profibus-PD ... 189

11.10. Sieć Intrerbus-S ... 190
11.11. Sieć Fieldbus Foundation H1, H2 ... 190
11.12. Sieć Fieldbus Foundation HSE – High Speed Ethernet ... 190
11.13. Sieć ControlNet ... 191
11.14. Sieć AS-I ... 191
11.15. Sieć P-Net ... 191
11.16. Prezentacja parametrów sieci przemysłowych ... 191
11.17. Analiza czasów transmisji danych ... 197
11.18. Kierunki rozwoju sieci przemysłowych ... 203
11.19. Stan prac normalizacyjnych ... 204
12. Wnioski końcowe ... 205
13. Skorowidz najczęściej używanych określeń i symboli ... 210
Literatura ... 215
Streszczenia ... 226

CONTENTS

1. Introduction 9
1.1. The work objectives 14
2. The industrial distributed real time systems-features and requirements 16
2.1. The problems of distributed real time systems designing 19
2.2. The industrial distributed real time systems 20
2.3. The system's node 22
2.4. The communication protocol 23
2.5. The operator station 24
2.6. Summary 25
3. The distributed control system 28
3.1. The programmable logic controller as main element of control system 35
3.2. The network coprocessor 37
3.3. The access network protocols applied in industrial networks 41
4. The networks with TOKEN BUS protocol 42
4.1. Description of data exchange 43
4.2. The "gaps" network formation 44
4.3. The time transmission checking 47
4.4. The network cycle and cycle of exchange data 49
4.5. Definition of network cycle time 50
4.6. The maximum of token exchange time 50
4.7. The case of network with "gaps" 53
4.8. The maximum of cycle time network with useful data transmission 54
4.9. The transmission buffering 60
4.10. The network efficiency and useful flow capacity 61
4.11. Correction of exchanges parameters 63
5. The networks with MASTER-SLAVE protocols 67
5.1. Preparing of exchange scenario and exchange data description 68
5.2. The "gaps" network formation 70
5.3. The exchanges parameterisation 72
5.4. The phenomena on "coprocessor-central unit" border 74
5.5. Definition of network cycle time 78
5.6. The network efficiency and useful flow capacity 85
5.7. Correction of exchanges parameters 87
6. The networks with Producer-Distributor-Consumer protocol 89
6.1. The Producer-Distributor-Consumer model 90
6.2. The principle of operation network with PDC protocol 91
6.3. The data structure 95
6.4. The transaction of periodical exchange 97
6.5. The request of aperiodical variable transmission 98
6.6. The request of message transmission 99
6.7. The analysis of time parameters connected with FIP protocol 101

6.8. The time analysis of protocol work 103
6.8.1. The static analysis 104
6.8.2. The dynamic analysis 106
6.9. The bit analysis 114
6.9.1. The bit frame representation 114
6.9.2. The periodic transaction 115
6.9.3. The aperiodic transaction 116
6.10. Analysis summary and realizability checking 116
6.11. Correction of exchanges time parameters in FIP network 119
6.12. Assessment of FIP network network efficiency and flow capacity 124
6.13. Summary 129
7. Correction of industrial networks parameters with cyclical data exchange transaction 131
7.1. The macrocycle of data exchange 132
7.2. The choice of macrocycle parameters 136
7.3. The methods of microcycle building and optimisation 138
7.3.1. The algorithmical methods 139
7.3.2. The method based on artificial intelligence 142
7.4. The practical realisation and conclusion 146
8. The method of communication protocols classification in the industrial networks 148
8.1. The network classification 153
8.2. The criteria of choice 155
8.3. The solution choice 155
8.4. Summary 157
9. The methods of lowest level industrial networks 158
9.1. Integration on I level 159
9.2. Integration on II level 162
9.3. Integration on III level 164
9.4. An example of integration 164
9.5. Summary 169
10. The TCP/IP protocol applying in industrial networks 171
10.1. The TCP/IP protocol in industrial networks 172
10.2. The system architecture influence on lowest level networks parameters 174
10.3. Summary 181
11. The solution review 184
11.1. The Modbus network 185
11.2. The Genius/N-80 network 185
11.3. The LonWorks network 186
11.4. The CAN (Controller Area Network) network 186
11.5. The SDS (Smart Distributed System) network 187
11.6. The DeviceNet network 187
11.7. The HART network 188
11.8. The FIP/ WorldFIP network 188
11.9. The Profibus-PD network 189
11.10. The Intrerbus-S network 190
11.11. The Fieldbus Foundation H1, H2 network 190
11.12. The Fieldbus Foundation HSE - High Speed Ethernet 190
11.13. The ControlNet network 191
11.14. The AS-I network 191

11.15. The P-Net network 191
11.16. The presentation of industrial networks parameters 191
11.17. The analysis of data transmission 197
11.18. The trends of industrial networks progress 203
11.19. The state of standardisation works 204
12. The final conclusions 205
13. The index of most frequently used definitions and symbols 210
Bibliography 215
Summary 226

1. WSTĘP

Szybki rozwój technologii elektronowej sprzyja powstawaniu urządzeń komputerowych o dużym stopniu niezawodności. Zjawisko to spowodowało i powoduje coraz szersze stosowanie środków, narzędzi i metod informatycznych w wielu gałęziach przemysłu. Z jednej strony zachęca to technologów do stosowania techniki komputerowej w procesach regulacji i sterowania, z drugiej zaś wymusza szukanie nowych rozwiązań sprzętowych i programowych, które by, w zadowalający sposób, wykorzystały rozwój technologii. Innymi słowy, coraz większe wymagania, stawiane przez procesy technologiczne, zmuszają do opracowywania nowych rozwiązań lub optymalizacji już istniejących.

Stosowanie techniki komputerowej w realizacji systemów sterowania i regulacji to złożony proces ewolucyjny.

Lata 60. i 70. bieżącego stulecia - to rozwój i stosowanie koncepcji scentralizowanego systemu sterowania i regulacji, w którym główną rolę pełnił pojedynczy komputer (rys. 1). Wyposażony w specjalizowane układy wejścia/wyjścia, binarne i analogowe pobierał dane z całego obiektu i samodzielnie na niego oddziaływał. Dobrym przykładem takiego komputera były na przykład minikomputery serii PDP-11 firmy DEC, a w Polsce maszyna cyfrowa rodziny ODRA 1300 (ODRA1310, ODRA1325). Była ona wyposażona w pełni autonomiczny kanał transmisji danych z obiektu technologicznego. Kanał ten, noszący miano kanału przemysłowego, obsługiwany z poziomu systemu operacyjnego (egzekutora) ekstrakodami typu „PERI" [107], umożliwiał bardzo szybki i niezawodny dostęp do danych pomiarowych i obiektowych stanów binarnych. Wraz z rosnącą liczbą zadań i wymogami czasu rzeczywistego pojawiły się dwa kluczowe problemy:

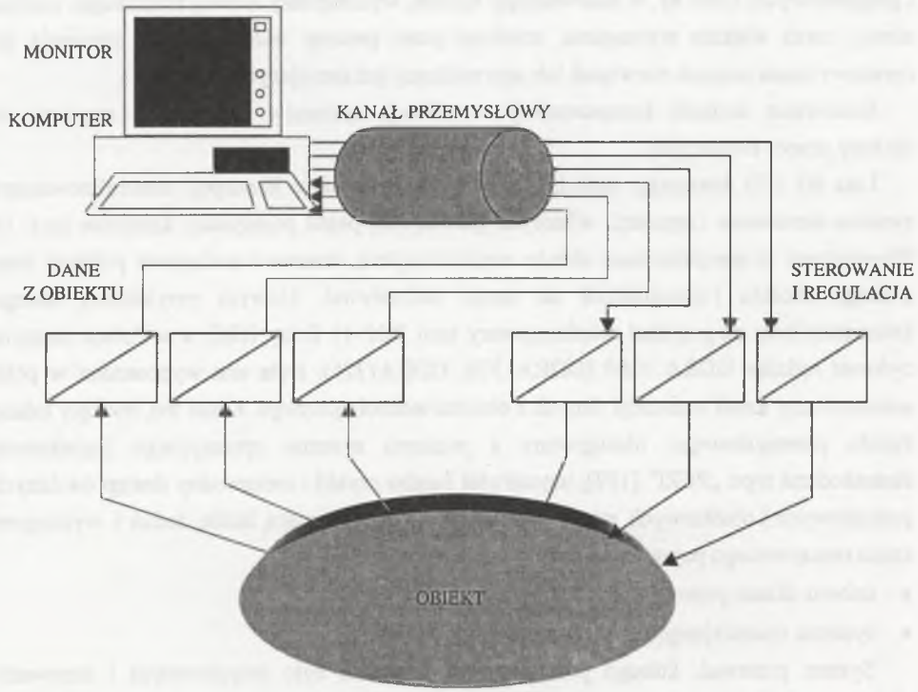
- doboru układu przerwań,
- systemu operacyjnego czasu rzeczywistego.

System przerwań, którego podstawowym zadaniem było przyjmowanie i sterowanie obsługą zdarzeń obiektowych, stał się niewydolny, gdyż pojawienie się na przykład tak zwanej „lawiny zdarzeń" powodowało, że realizacja programów obsługi już odebranych przerwań była „wstrzymywana" przez kolejno nadchodzące i system operacyjny nie był w stanie realizować żadnych innych zadań, ponad przekazywanie sterowania do kolejno wywoływanych programów obsługi przerwań. Zjawisko to nosiło miano „migotania przerwań" [170].

Zatem realizacja cyklu pętli programu systemu operacyjnego, celem zapewnienia stałej jego wartości, musiała zostać podzielona na dwa etapy [17, 142, 143]:

- etap 1 związany z pobraniem pomiarów z obiektu, filtracją, obliczeniami sterowania i aktualizacją wyjść sterujących obiektem (ang. *Foreground*),
- etap 2 związany z realizacją jednego programu interakcyjnego (ang. *Background*).

Przykładem takiego systemu operacyjnego może być choćby system RSX-11 dla komputerów PDP-11 lub też system RT-11 dla komputera MERA-60. [43, 1]. Przy takim podziale czasu pracy systemu operacyjnego powstaje problem, ile czasu przeznaczyć na realizację etapu 1, a ile na realizację etapu 2.



Rys. 1. Scentralizowany system sterowania
Fig. 1. Centralised control system

W systemach scentralizowanych, wykorzystujących pojedyncze komputery, występowały ograniczenia ich stosowalności.

Ograniczenia stosowalności „od dołu" dotyczyły małych obiektów lub prostych technologii, gdzie wysoka cena komputera była barierą jego stosowania.

Ograniczenia stosowalności „od góry" natomiast były związane ze wzrostem liczby obsługiwanych wejść/wyjść i stale rosnącym stopniem złożoności algorytmów sterowania

i regulacji. Powodowało to powstanie wielu negatywnych zjawisk, z których najważniejszymi były:

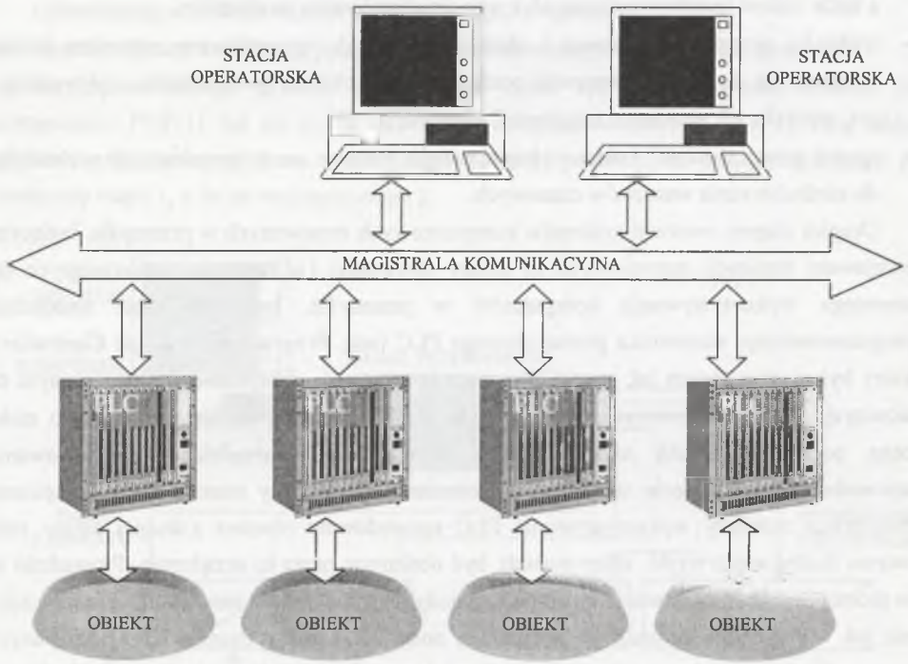
- stale rosnący stopień złożoności oprogramowania powodujący wzrost czasu realizacji, a także wzrost kosztów związanych z jego uruchomieniem na obiekcie,
- trudności w realizacji systemu i obsłudze przerwań, spowodowane wzrostem liczby koniecznych do obsługi przerwań pochodzących z obiektu (powstawanie, opisywanego już, zjawiska tak zwanego „migotania" przerwań),
- spadek przepustowości i niezawodności całego systemu, co w konsekwencji prowadziło do niedochowania warunków czasowych.

Drugim etapem ewolucji systemów komputerowych stosowanych w przemyśle, będącym przejawem tendencji rozwojowych ze strony elektroniki i informatyki, zmierzających do szerszego wykorzystywania komputerów w przemyśle, było powstanie swobodnie programowalnego sterownika przemysłowego PLC (ang. ***Programmable Logic Controller***), który był niczym innym jak specjalizowanym komputerem, fabrycznie przystosowanym do akwizycji sygnałów binarnych i analogowych pochodzących z obiektu. Stosunkowo niska cena połączona z dużą niezawodnością i efektywnymi narzędziami programowania spowodowała przesunięcie się dolnej, wspomnianej już, granicy stosowalności komputera. Pozytywne rezultaty wykorzystywania PLC spowodowały również z drugiej strony stały wzrost liczby wejść/wyjść, które musiały być obsłużone przez to urządzenie. Prowadziło to w prostej linii do powstawania złożonych, pojedynczych układów sterowania, które zaczęły, tak jak w przypadku centralnego komputera, nosić znamiona systemów scentralizowanych z wszystkimi tego ujemnymi konsekwencjami. Stały wzrost liczby wejść/wyjść był efektem przekazywania systemom komputerowym realizacji coraz bardziej skomplikowanych algorytmów technologicznych. Dalszy rozwój w tym kierunku komputerowych systemów sterowania znów stał się niemożliwy, ze względu na malejącą niezawodność całego układu i wzrost złożoności oprogramowania, które wymagało coraz to większych mocy obliczeniowych.

Dodatkowym efektem projektowania tak dużych systemów był znaczny wzrost kosztów realizacji i kolejny spadek niezawodności, związany z okablowaniem obiektu. Sumaryczna długość kabli doprowadzających sygnały z obiektu i wyprowadzających efekty pracy komputera do obiektu zaczęła być liczona w dziesiątkach, a nawet setkach kilometrów.

Podsumujmy zatem, wprowadzenie sterownika swobodnie programowalnego zwiększyło zakres wykorzystywania komputera, co objawiło się przez znaczne przesunięcie dolnej granicy jego stosowalności, ale w niewielkim stopniu przesunęło granicę górną. Ciągle mieliśmy do czynienia z systemem scentralizowanym, a to już mogło zahamować stosowanie informatyki w przemyśle. Przyszła zatem pora na poszukiwanie zmian strukturalnych w przemysłowych, informatycznych systemach sterowania. W sukurs tym poszukiwaniom

przyszedł rozwój sieci komputerowych, a przede wszystkim wzrost ich niezawodności. Jest to kolejny etap ewolucji. Sieci komputerowe spowodowały powstanie i lawinowy rozwój struktur zdecentralizowanych, nazywanych również rozproszonymi (rys. 2).

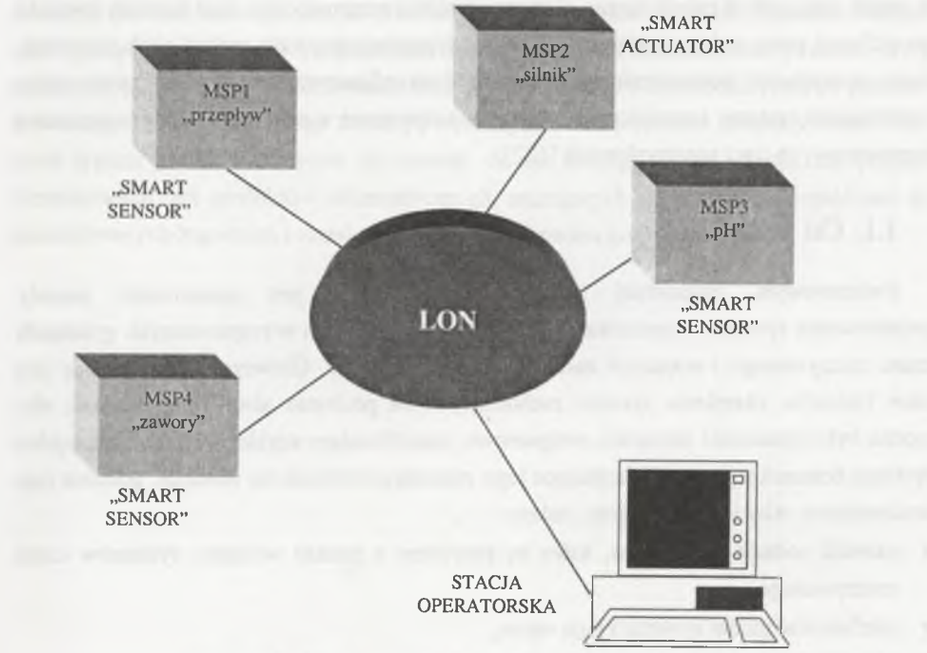


Rys. 2. Zdecentralizowana struktura systemu sterowania
Fig. 2. The structure of distributed control system

W systemach tych zadania zostały podzielone pomiędzy pojedyncze komputery – sterowniki swobodnie programowalne.

Dzięki temu zniknął cały szereg wad systemu scentralizowanego, ale pojawiła się konieczność zastosowania systemu komunikacji, wynikająca choćby ze zdalnego sterowania i monitorowania obiektu.

Równolegle z rozwojem systemów zdecentralizowanych, bazujących na sieciach komputerowych i PLC, rozwijają się systemy typu LON (ang. *Local Operating Network*), bazujące na szybkich sieciach lokalnych, a w miejsce PLC stosujące „inteligentne", wysoce specjalizowane, moduły sprzętowo-programowe (rys. 3) [81, 129, 19, 66, 45].



Rys. 3. System sterowania wykorzystujący „inteligentne" moduły sprzętowo-programowe
Fig. 3. The remote software-hardware modules in control system

Specjalizacja dotyczy realizacji wąskich zadań technologicznych. Rozwiązanie to pozwala na rozbicie procesu sterowania na wiele elementów. Przez to, że każdy z elementów realizuje bardzo niewiele, ściśle określonych, funkcji (brak złożonych programów, wydłużających czas reakcji), można tworzyć bardzo szybkie systemy sterowania. Ale i tu, tak jak w systemach zdecentralizowanych, bazujących na PLC, wymagany jest sprawny i niezawodny system komunikacyjny.

Wszystko to, o czym była do tej pory mowa, w połączeniu z rozwojem sieci komputerowych, wzrostem ich niezawodności, spowodowało lawinowy rozwój tak zwanych systemów rozproszonych [6, 157]. Ich stosowanie rozwiązało wiele problemów, ale i tu napotyka się na szereg barier, które zmuszają do szukania rozwiązań strukturalnych i odpowiedniego sposobu projektowania informatycznych systemów sterowania. Podstawowym ich parametrem jest czas reakcji na zmiany zachodzące na obiekcie. Czas ten jest wypadkową czasów reakcji pojedynczych jego elementów, na który ma wpływ sprawność komunikacji i szybkość wymiany informacji. Szybkość wymiany informacji, mierzona w wartościach bezwzględnych, nie tylko powinna być jak największa, ale powinno się zagwarantować jej minimalną, nieprzekraczalną wartość. Gwarantowany, zdeterminowany

w czasie czas reakcji pojedynczego elementu systemu rozproszonego daje bowiem pewność prawidłowej pracy całego systemu. Tak więc zdeterminowany czas reakcji obok niezawodności, umiejętności podnoszenia się z upadków i rekonfigurowalność to podstawowe cechy wyróżniające systemy komunikacyjne (sieciowe) stosowane w przemyśle, które noszą nazwę komputerowych sieci przemysłowych [122].

1.1. Cel pracy

Podstawowym, najbardziej ogólnym, celem pracy jest opracowanie metody projektowania systemów komunikacyjnych, wykorzystywanych w rozproszonych systemach czasu rzeczywistego i mających zastosowanie w przemyśle. Głównym celem metody jest takie ilościowe określenie zjawisk zachodzących na poziomie sieci komputerowej, aby można było opracować narzędzie programowe, umożliwiające szybką weryfikację projektu systemu komunikacyjnego i ułatwiające jego rekonfigurowalność na obiekcie, podczas jego uruchamiania. Aby ten cel osiągnąć, należy:

- określić rodzaje protokołów, które są przydatne z punktu widzenia systemów czasu rzeczywistego,
- zdefiniować model systemu i jego węzła,
- określić ilościowo i jakościowo zjawiska zachodzące w węzłach systemu,
- zdefiniować grupę wielkości, nie stosując żadnych uproszczeń i przybliżeń, i określić ich wpływ na wydolność systemu komunikacyjnego, przy czym
  liczba określonych parametrów powinna pozwolić na jak największe uogólnienie. Dzięki temu będzie można analizować liczną grupę rozwiązań firmowych, różniących się między sobą w szczegółach, ale zgodnych z podstawowymi cechami protokołów sieci przemysłowych branych pod uwagę,
- przeprowadzić analizę czasową przepływu informacji, a w tym określić ilościowo czas trwania cyklu sieci, sprawność i przepustowość użyteczną,
- zaproponować sposób klasyfikacji protokołów komunikacyjnych ułatwiający ich właściwy dobór,
- określić inne źródła opóźnień i podać metody ich eliminacji.

Oryginalność metody polega na tym, że umożliwia ona nie tylko właściwie konfigurować sieci, gwarantując spełnienie wymagań systemu czasu rzeczywistego, ale również pozwala na dużą elastyczność w projektowaniu. Opisując bowiem ilościowo zjawiska zachodzące w węźle sieci, pozwala projektantowi na parametryzację odpowiadającą wymaganiom technologii i procesu przemysłowego. Ponadto, umożliwia opracowanie narzędzia informatycznego, dzięki któremu można przyspieszyć proces projektowania i poprawiać parametry pracy sieci. Ma to szczególnie znaczenie wtedy, gdy istnieje konieczność zmiany istniejącego projektu

w trakcie uruchamiania systemu na obiekcie, i nie ma już czasu na kolejne, żmudne analizowanie przepływu informacji. Proponowana metoda i narzędzia informatyczne, powstałe na jej podstawie, pozwalają na konstruowanie sieci przemysłowych o deterministycznych protokołach dostępu do medium. Zatem zakres jej wykorzystywania jest ściśle związany ze stosowanymi typami protokołów. Autor nie znalazł, co nie znaczy, że nie istnieją, ani pozycji literaturowych, ani produktów informatycznych, cechujących się podobnym podejściem do prezentowanych zagadnień i rozwiązujących sygnalizowaną grupę problemów.

2. SIECIOWE, PRZEMYSŁOWE SYSTEMY ROZPROSZONE CZASU RZECZYWISTEGO – CECHY I WYMAGANIA

Stale wzrastająca niezawodność i moc obliczeniowa systemów komputerowych skłania do szerszego ich stosowania w przemyśle. Systemy komputerowe realizują bardzo odpowiedzialne zadania, a współczesna technologia stawia coraz poważniejsze wymagania, związane przede wszystkim z gwarantowanym i nieprzekraczalnym czasem realizacji pojedynczego cyklu sterowania bądź regulacji. To właśnie zmusza do przeprowadzania bardzo szczegółowej analizy możliwości systemu informatycznego. Tym bardziej że owe systemy, z punktu widzenia informatyki, są klasycznymi systemami rozproszonymi czasu rzeczywistego, stosującymi sieć komputerową jako podstawowe medium wszelkiego rodzaju wymian informacji.

Pojęcie „system rozproszony czasu rzeczywistego" zawiera w sobie dwa elementy [156]:

- system rozproszony,
- system czasu rzeczywistego.

System rozproszony charakteryzuje się tym, że wiele procesów jest realizowanych na wielu procesorach. Może temu towarzyszyć, aczkolwiek nie musi, rozproszenie terytorialne, które jest tak typowe dla obiektów przemysłowych, zarządzanych i sterowanych za pomocą komputerów. Procesy rozproszone są koordynowane przez grupę wewnętrznych dla węzła procesów komunikacyjnych i synchronizacyjnych. Korzyści wynikające ze stosowania systemów rozproszonych są następujące:

- zwiększona moc obliczeniowa wynikająca ze stosowania przetwarzania równoległego,
- zwiększona niezawodność, istnieje bowiem możliwość takiego zaprojektowania systemu, aby w chwili awarii funkcje nie działającego elementu (węzła) systemu zostały przejęte przez inne - działające,
- zwiększona adaptacyjność (rekonfigurowalność) dzięki możliwościom modyfikacji programów i łatwemu podziałowi zasobów.

W pełni rozproszonym systemem nazywamy taki, w którym sterowanie, sprzęt (*ang. hardware*) i dane są rozproszone. Istnieją bowiem systemy, które nie w pełni są rozproszone. Do nich zaliczyć należy [156]:

- Systemy typu SIMP – w których sprzęt jest zmultiplikowany, ale występuje scentralizowane sterowanie,
- Systemy typu MIMD – w których sprzęt jest scentralizowany, a rozproszone jest sterowanie.

Systemy rozproszone można podzielić również na:

- heterogeniczne lub homogeniczne,

a te zaś na:

- scentralizowane (np. MASTER-SLAVE, KLIENT-SERVER),
- zdecentralizowane, w których węzły systemu są autonomiczne.

Można zatem dokonywać coraz to innych klasyfikacji, ale wspólną cechą lub tendencją obowiązującą przy projektowaniu tych systemów jest dążność do stosowania sieci komputerowej w celu tworzenia grup stacji roboczych i komunikacji międzyprocesowej.

System czasu rzeczywistego charakteryzuje się silnymi ograniczeniami czasowymi (ang. *„timing constraints"*). Czas jako parametr staje się w tych systemach wielkością krytyczną. Typowe systemy czasu rzeczywistego zawierają podsystemy kontrolująco-sterujące, do których zalicza się na przykład specjalizowane kontrolery komputerowe, a także podsystemy kontrolowane i sterowane (na przykład warstwa fizyczna obiektu).

Obie grupy podsystemów podlegają silnej interakcji, którą można opisać trzema operacjami [2, 6, 24, 27]:

- zbieranie danych z warstwy fizycznej obiektu (ang. *„sampling"*),
- natychmiastowe przetwarzanie zebranych danych i ewentualne uruchamianie procesów obliczeniowych (algorytmów) (ang. *„processing"*),
- odpowiedź systemu skierowana do obiektu warstwy fizycznej, będąca wynikiem działania warstwy obliczeniowej (ang. *„responding"*).

Wymienione trzy operacje, co również charakteryzuje system czasu rzeczywistego, muszą zakończyć się w określonym, wyspecyfikowanym czasie.

Systemy czasu rzeczywistego można podzielić na dwie grupy:

- systemy, w których toleruje się przekraczanie ograniczeń czasowych (ang. *„soft real-time systems"*),
- systemy, w których tolerowanie przekraczania ograniczeń czasowych albo nie jest dopuszczalne, albo dopuszcza się je, ale w wyjątkowych przypadkach, lub ograniczonym zakresie (ang. *„hard real-time systems"*) [22].

W systemach, w których dopuszcza się przekraczanie ograniczeń czasowych, nie pociąga to za sobą znacznych konsekwencji. System pracuje poprawnie nawet przy przekroczeniu ograniczeń. Przekroczenie ograniczeń nie może jednak wystąpić w sposób drastyczny.

Typowe przykłady takich systemów to:

- system zdalnego dostępu do danych,
- system rezerwacji biletów,
- system bankomatów.

W systemach bez tolerancji łamania ograniczeń ich przekroczenie grozi znacznymi konsekwencjami. Typowymi takimi systemami są:

- systemy, od poprawności działania których zależy na przykład ludzkie życie,
- systemy kontroli ruchu lotniczego,
- systemy sterowania i regulacji większością procesów przemysłowych,
- systemy monitorowania pacjentów,
- systemy z krytycznym czasem reakcji,
- systemy sterowania pracą robotów,
- systemy telefonii.

Jak już wspomniano, w systemach czasu rzeczywistego pojęcie czasu jest niezwykle ważne. Występuje on w czterech aspektach:

- dostępu do zegara,
- opóźnienia przetwarzania,
- detekcji przekroczeń, przeterminowania (ang. *„time-out"*),
- specyfikacji ograniczeń czasowych i określenia czasu reakcji.

Dostęp do zegara może być absolutny przez dostęp do zegara czasu rzeczywistego albo pośredni przez pomiar, lub detekcję czasu wykonywania pętli. Dostęp może być również związany z synchronizacją zegarów w węzłach systemu. Jest to niezwykle trudny i typowy problem przemysłowych systemów rozproszonych czasu rzeczywistego, gdzie musi być brany pod uwagę tak zwany „najgorszy przypadek" [88] i wszelkie operacje muszą być odniesione do zegara „najwolniejszego".

Zdolność opóźniania przetwarzania jest związana z zagadnieniem kolejkowania. Metoda kolejkowania eliminuje bowiem tak zwane stany „zbędnego oczekiwania"(ang. *„busy wait"*). Proces może skierować zadanie do kolejki, które będzie wykonane później już jako zdarzenie, zamiast generować sztuczne opóźnienie w postaci stanów *„busy wait"*.

Detekcja tak zwanych *„time-out"*-ów dotyczy zdolności procesu do rejestracji braku pojawienia się zdarzenia i w konsekwencji do „znieczulania" się na pojawienie się określonego zdarzenia.

Specyfikacja ograniczeń czasowych jest związana z respektowaniem ograniczeń czasowych, narzucanych przez warstwę fizyczną obiektu. Jest to związane z minimalnym i maksymalnym czasem realizacji pętli programu (ang. *„deadline specification"*).

2.1. Problemy projektowania rozproszonych systemów czasu rzeczywistego

Podstawowe problemy projektowania i późniejszej realizacji systemów czasu rzeczywistego są związane z ich podstawowymi właściwościami, do których możemy zaliczyć:

- działanie w sposób ciągły w bardzo trudnych warunkach,
- ostre ograniczenia czasowe,
- interakcję pomiędzy procesami asynchronicznymi,
- przewidywalność opóźnień i warunki tak zwanego „wyścigu",
- nieokreśloność i brak powtarzalności (cykliczności) zachowań,
- globalność zegara i stanu obiektu,
- wielowątkowość interakcji procesów.

Większość rozproszonych systemów czasu rzeczywistego, a w tym również systemów sterowania i monitorowania procesów przemysłowych [5], musi pracować ciągle i zachowywać interakcję ze sterowanym obiektem. Mówiąc o trudnych warunkach, należy mieć na myśli przede wszystkim szereg wzajemnych zależności czasowych pomiędzy procesami wewnętrznymi. Zależności te objawiają się w sposób wyjątkowo niebezpieczny i „dokuczliwy" w chwilach stanów nieustalonych obiektu, co powoduje szereg wywołań zadań, które są ze sobą powiązane kolejnością i czasem ich realizacji. Ma to na przykład miejsce w chwilach pojawiania się tak zwanej „obiektowej lawiny zdarzeń". Trudne warunki pracy dotyczą również, co ma miejsce w zastosowaniach przemysłowych, warunków środowiskowych i klimatycznych, w których sprzęt informatyczny ma pracować.

Ostre ograniczenia czasowe wpływają na „poprawność" wykonywania zadań przez rozproszone systemy czasu rzeczywistego. Poprawność pracy jest określona nie tylko przez liczbę i szybkość procesorów sterujących podwarstwami całego systemu, ale także przez ograniczenia czasowe narzucone przez czas obsługi obiektu (otoczenia).

Rozproszone systemy czasu rzeczywistego są projektowane do interakcji z warstwą fizyczną, a procesory tych systemów są rozproszone terytorialnie. Procesy asynchroniczne, na przykład te, które są zlokalizowane w węzłach systemu, a są odpowiedzialne za „dajmy na to" realizację algorytmów sterowania, zawierają wewnętrzne systemy rozproszone czasu rzeczywistego. One i zewnętrzne procesy fizyczne komunikują się pomiędzy sobą poprzez wymianę komunikatów. Sekwencje zdarzeń w procesach asynchronicznych są trudne do przewidzenia i często przyjęte założenia na etapie projektowania są naruszane w czasie działania.

Z powodu trudnego do przewidzenia natężenia ruchu w sieci komunikacyjnej i dystansu pomiędzy dwoma węzłami komunikacyjnymi systemu opóźnienia wnoszone przez komunikację międzyprocesorową są nieprzewidywalne, ale nie można ich pominąć. Stąd konieczność stosowania do analizy „metody najgorszego przypadku" [88]. Dodatkowo, może wystą-

pić zjawisko „wyścigu", gdy dwa procesy dzielą się tymi samymi zasobami. Tak więc sekwencje zdarzeń specyfikowane na etapie projektowania, a służące synchronizacji podprocesów mogą się zmieniać podczas normalnej pracy systemu.

Z powodu trudności przewidywania opóźnień komunikacyjnych i warunków „wyścigu" pomiędzy procesami [14] i procesorami, zachowanie systemów rozproszonych czasu rzeczywistego może być nieokreślone. Może być bowiem tak, że ponowne wykonanie tego samego programu, z tymi samymi stanami wejść, niekoniecznie musi produkować te same rezultaty.

Każdy procesor systemu ma własny zegar działający niezależnie od zegarów innych procesorów. Jest niezwykle trudno określić precyzyjnie czas globalny, który jest niezbędny dla określenia właściwych stanów globalnych i w procesie diagnozowania i monitorowania pracy całego systemu.

W przeciwieństwie do programów sekwencyjnych, mających pojedynczy mechanizm sterowania, systemy rozproszone czasu rzeczywistego cechują wielokrotnie przeplatające się sekwencje (przeploty) sterowania. Także owe sekwencje (przeploty) sterowania są swobodnie lub rygorystycznie powiązane z protokołami komunikacyjnymi. Odbywa się to albo za pośrednictwem pamięci, albo za pomocą komunikatów. Dodatkowo, istnienie wielowątkowości jest utrudnieniem przy analizie i monitorowaniu pracy systemu.

Podsumowując, systemy rozproszone czasu rzeczywistego odróżniają się zasadniczo od innych systemów. Podstawowymi wyróżnikami są ograniczenia czasowe. Najbardziej istotne zagadnienie, niezwykle ważne przy analizie i śledzeniu pracy systemu, to znalezienie miejsc jego pracy, gdzie nie są respektowane ograniczenia czasowe. Komunikacja międzyprocesorowa (pomiędzy procesorami węzłów systemu) i międzyprocesowa (między procesami węzła sieci lub między dwoma procesami różnych węzłów) to najważniejsze źródła nierespektowania zależności czasowych.

2.2. Przemysłowe rozproszone systemy czasu rzeczywistego

Jak powszechnie wiadomo, podstawowym elementem przemysłowego systemu sterowania jest komputer przemysłowy lub jego odmiana, jaką jest sterownik swobodnie programowalny (PLC), albo specjalizowane urządzenie mikroprocesorowe wyposażone w koprocesor komunikacyjny. Rozproszenie systemu uzyskuje się w sposób naturalny, bo o tym decyduje obiekt przemysłowy, w którym da się wyodrębnić terytorialnie oddzielone procesy technologiczne, które będą sterowane niezależnie i w dużym stopniu autonomicznie przez węzły systemu komputerowego. Węzłem systemu komputerowego jest zazwyczaj pojedynczy sterownik (PLC) lub ich grupa. Dlatego cały system staje się natychmiast rozproszonym systemem w sensie informatycznym, gdyż następuje tutaj rozproszenie zarówno procesorów, jak i zasobów i sterowania. Oprócz komputerów przemysłowych czy

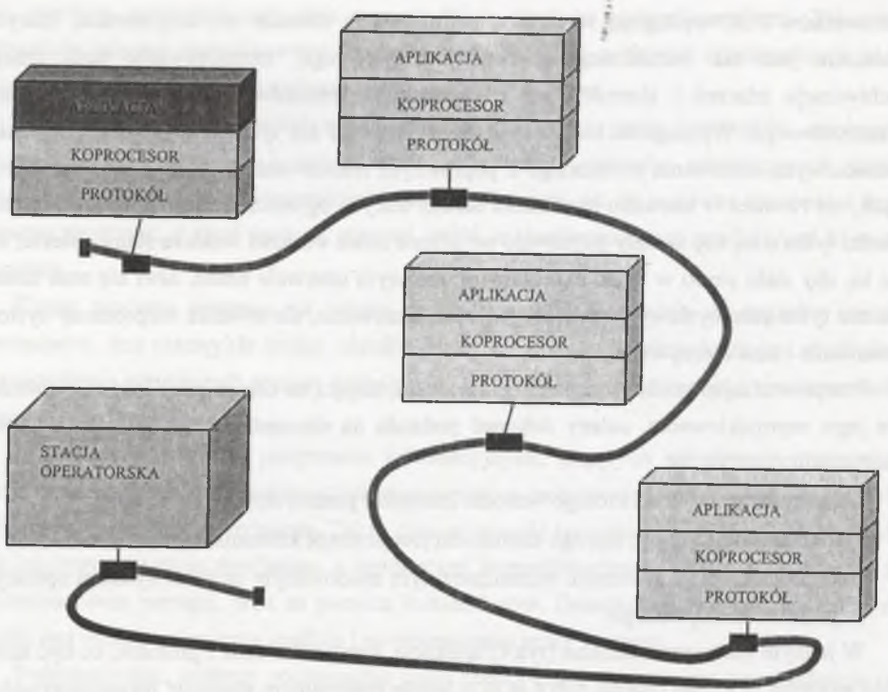
sterowników PLC występują w takim systemie stacje robocze lub inżynierskie, których zadaniem jest tak wizualizacja procesu przemysłowego, monitorowanie jego pracy, archiwizacja zdarzeń i alarmów, jak i generowanie rozkazów kierowanych do obiektu przemysłowego. Wymagania technologii idą w kierunku nie tylko precyzyjnego, w sensie jakościowym, sterowania związanego z poprawnym realizowaniem algorytmów numerycznych, ale również w kierunku narzucania bardzo ostrych ograniczeń czasowych. A więc nie chodzi tylko o to, aby zostały poprawnie wyliczone nowe wartości wektora stanu obiektu, ale i o to, aby stało się to w nieprzekraczalnym, zadanym interwale czasu. Jawi się nam zatem już nie tylko przemysłowy rozproszony system sterowania, ale również rozproszony system sterowania czasu rzeczywistego.

Przeprowadzając analizę pracy takiego systemu, mającą na celu między innymi prawidłowe jego zaprojektowanie, należy dokonać podziału na elementarne składniki, do których można zaliczyć (rys. 4):

- węzeł systemu, w skład którego wchodzi komputer przemysłowy (lub PLC),
- system komunikacyjny, którego elementem jest protokół komunikacyjny,
- stację operatorską z systemem wizualizacyjnym zbudowanym na bazie systemu operacyjnego czasu rzeczywistego.

W jednym bloku umieszczono (rys.4) aplikację, koprocesor sieci i protokół, co być może robi wrażenie pewnego chaosu, gdyż są to w sensie materialnym elementy nieporównywalne, ale zabieg ten jest celowy, ponieważ każdy z owych elementów ma pewne cechy, które w bardzo istotny sposób na siebie wpływają, a wspólną płaszczyzną jest czas realizacji typowych dla każdego z nich zadań.

Każdy z wymienionych elementów zachowuje znaczne możliwości pracy autonomicznej, ale stanowią one składniki systemu rozproszonego i muszą spełniać te same wymagania, o których była mowa w poprzednich rozdziałach. Projektując przemysłowy, rozproszony system czasu rzeczywistego, należy każdy z wymienionych elementów poddać niezależnej analizie, zmierzającej do określenia globalnego parametru, którym niech będzie tak zwany graniczny czas T_G reakcji całego systemu na dowolne zdarzenie, które może wystąpić na obiekcie. Określenie wartości T_G posłuży tym samym do określenia granic stosowalności całego systemu.



Rys. 4. Model rozproszonego systemu przemysłowego
Fig. 4. The model of distributed industry system

2.3. Węzeł systemu

Jak już wspomniano, węzłem systemu jest sterownik przemysłowy (PLC) lub grupa sterowników. Rolą sterownika jest realizacja programu rezydującego w jego pamięci operacyjnej. Program jest zakodowanym, numerycznym algorytmem sterowania. Programowanie sterownika jest niczym innym jak tworzeniem małego systemu operacyjnego czasu rzeczywistego, którego podstawową cechą jest cykliczność realizacji. Przy tworzeniu oprogramowania należy kierować się tymi samymi kryteriami co przy tworzeniu dużych systemów operacyjnych czasu rzeczywistego. Szczególną uwagę należy zwracać na czas realizacji podstawowej pętli programu [122, 3]. Jest to czas, który upływa od chwili pobrania wektora stanu obiektu do chwili „wystawienia" jego nowej wartości na wyjściu węzła (przesłanie nowej jego wartości do sterowanego obiektu). Czas trwania pętli zależy wprost proporcjonalnie od czasu realizacji aplikacji (patrz rys.10). To znów jest zależne od złożoności algorytmów, które mają być aplikowane i od umiejętności programisty, którego rolę trudno przecenić. Doskonała znajomość parametrów sterownika i jego architektury

pozwala tworzyć oprogramowanie w sposób optymalny, to znaczy minimalizować czas realizacji programu przy jednoczesnej minimalizacji zajętości pamięci. Czas cyklu realizacji programu ma decydujący wpływ na sprawną komunikację. Sterownik, a tym samym węzeł systemu komunikuje się z innymi węzłami za pośrednictwem koprocesora sieci. Koprocesor natomiast ma dostęp do zasobów macierzystej jednostki centralnej tylko raz na jeden cykl realizacji pętli programu. Zatem, im dłuższy jest cykl pojedynczej realizacji programu systemowego (aplikacja jest systemem operacyjnym czasu rzeczywistego), tym rzadziej koprocesor może kontaktować się z zasobami i tym później jest w stanie wyemitować niezbędne dane.

Na granicy koprocesor-jednostka centralna węzła systemu zachodzi wiele zjawisk, których poznanie jest niezbędne do przeprowadzenia prawidłowej analizy pracy systemu. Zjawiska te, jak choćby przetwarzanie ramki czy generacja sumy kontrolnej, nie tylko należy znać, ale również należy precyzyjnie określić ich czas trwania. Wpływa to, jak łatwo się domyślić, w sposób znaczący na sprawność systemu komunikacyjnego. Będzie o tym mowa w następnych rozdziałach pracy.

2.4. Protokół komunikacyjny

Protokół komunikacyjny jest kolejnym elementem rozproszonego systemu czasu rzeczywistego. Aby spełnić warunki systemu czasu rzeczywistego, musi to być protokół o zdeterminowanym w czasie dostępie. Należy przez to rozumieć, że protokół gwarantuje każdemu uczestnikowi procesu wymiany informacji możliwość emisji i odbioru danych w nieprzekraczalnym, gwarantowanym i znanym czasie. Nie oznacza to bynajmniej, że parametr ten jest cechą protokołu. To dopiero analiza przepływu danych w systemie da odpowiedź, ile ten czas wynosi, a protokół zapewni jedynie, że nie zostanie on przekroczony. To, że sieć jest w stanie przesłać określoną ilość informacji w pewnym czasie, którego wielkość zależy choćby od parametrów transmisji, takich jak prędkość czy długość bloku danych, nie determinuje sprawności komunikacji. Nie należy również rozpatrywać protokołu komunikacyjnego w oderwaniu od koprocesora sieci. Tu, podobnie jak w węźle systemu, zachodzi wiele zjawisk, które będą wpływać na czas wymiany informacji. Ważne jest, aby znać budowę koprocesora i jego wszystkie parametry czasowe oraz pewne cechy konstrukcyjne, jak choćby liczbę buforów do transmisji i odbioru.

Współcześnie istnieje kilka powszechnie znanych i akceptowanych protokołów o zdeterminowanym w czasie dostępie do medium. Do nich należy zaliczyć protokoły z „krążącym żetonem" typu „Master-Slave" czy „Producent-Dystrybutor-Konsument". Na marginesie tej informacji należy pamiętać, że na wybór protokołu będzie miał wpływ nie tylko determinizm czasowy, ale również rodzaj aplikacji. Będą bowiem rozwiązania

„szybsze" jak i „wolniejsze", do których należy dobrać odpowiednią sieć, będącą najtańszym rozwiązaniem, ale spełniającym wszelkie wymagania czasowe.

2.5. Stacja operatorska

Stację operatorską należy rozpatrywać w dwóch kontekstach. Pierwszym z nich jest pojęcie stacji jako węzła systemu. Przy takim podejściu pozostają w mocy wszystkie uwagi zamieszczone w poprzednich dwóch podrozdziałach. Wyodrębniając w niej moduł komunikacji, należy dokonać analizy czasowej związanej z procesem odbioru jak i nadawania. Taki podproces trzeba potraktować, z logicznego punktu widzenia, jako koprocesor sieci i wniknąć w jego budowę, aby znaleźć odpowiedź na mniej więcej takie same pytania, jakie powstają przy analizowaniu klasycznego koprocesora. A więc należy obliczyć następujące czasy:

- czas detekcji ramki,
- czas analizy odebranej ramki danych,
- czas przetwarzania ramki przez odpowiednią warstwę oprogramowania,
- czas dostępu do warstwy aplikacji,
- czas obliczania cyklicznej sumy kontrolnej,
- czas przygotowania ramki do transmisji, uwzględniający czas oczekiwania na dostęp do pamięci macierzystej jednostki centralnej,
- czas zapełnienia buforów nadawczych.

Drugie zagadnienie, pod kątem którego należy rozpatrywać stację operatorską, to stacja jako jednoprocesorowy system czasu rzeczywistego, będący aplikacją zrealizowaną na bazie systemu operacyjnego czasu rzeczywistego. Przy tworzeniu takiej aplikacji napotykamy na wiele tych samych problemów związanych z poprawnym projektowaniem systemów czasu rzeczywistego, o których była już mowa. Stacja operatorska charakteryzuje się tym, że składa się z szeregu procesów, które wzajemnie muszą współgrać, mając dostęp do tych samych zasobów, komunikować się między sobą za pomocą pamięci, wykluczać się i blokować w przypadkach konfliktów, przekazywać sterowanie pomiędzy sobą, a wszystko to musi spełniać wymagania systemu czasu rzeczywistego. Nie bez znaczenia jest również współpraca z platformą systemu operacyjnego [36, 26].

Niezwykle ważnym problemem jest tworzenie konkretnej aplikacji użytkowej stacji wizualizacyjnej (operatorskiej) za pomocą systemu wizualizacyjnego, o którym była mowa powyżej, a który, jak pamiętamy, jest systemem czasu rzeczywistego. Pojawiają się zatem problemy zsynchronizowania aplikacji z resztą systemu rozproszonego, właściwego obciążania systemu wizualizacyjnego, właściwej koincydencji wywoływania specjalizowanych procedur (ang. *„users programs"*), dopuszczalnej liczby obiektów graficznych i ich ewentualnego odświeżania i animowania. Wszystko to razem powoduje, że zaprojektowanie aplikacji wizualizacyjnej nie jest banalne i wymaga wnikliwej analizy i znacznej wiedzy informatycznej.

2.6. Podsumowanie

Reasumując, w świetle tego co już zostało powiedziane, współczesne systemy komputerowe stosowane w przemyśle, projektowane z myślą o sterowaniu i monitorowaniu procesów przemysłowych są klasycznymi, z punktu widzenia informatyki, rozproszonymi systemami czasu rzeczywistego (*ang. DRTS – **Distribiuted Real Time System***) [156]. Mamy tu bowiem do czynienia zarówno z rozproszeniem mocy obliczeniowej i zasobów pamięci, jak i z rozproszeniem terytorialnym. Ponadto czas jest parametrem krytycznym, a komunikacja między węzłami (podprocesami) odbywa się za pośrednictwem sieci komputerowej. Natomiast tworzenie oprogramowania aplikacyjnego, rezydującego w węzłach systemu jest niczym innym jak budową wielu lokalnych systemów czasu rzeczywistego. Podstawowe pytanie, jakie rodzi się w chwili tworzenia takiego systemu, to, czy system ów spełni wymagania dotyczące gwarantowanego czasu realizacji wszystkich zadań postawionych przez proces technologiczny, przy jednoczesnym zapewnieniu jego niezawodnej pracy.

W niniejszej pracy zaproponowano, oryginalną zdaniem autora, metodę analizy przepływu danych w przemysłowych systemach rozproszonych czasu rzeczywistego, wykorzystujących przemysłowe sieci komputerowe. Oparta jest ona na zaproponowanym modelu, który precyczyjnie określa wszystkie zjawiska na płaszczyźnie czasowej, zachodzące na styku „sieć-koprocesor-aplikacja w węźle". Jest to owszem podejście analityczne, ale bez stosowania żadnych uproszczeń. Wydaje się również, że jest ona dość uniwersalna, gdyż nie ogranicza się do konkretnych rozwiązań firmowych, a opiera się na ogólnym opisie protokołów deterministycznych. Daje więc możliwości potencjalnemu projektantowi zastosowania jej do szerokiej gamy rozwiązań komercyjnych dzięki temu, że opisuje ilościowo wszystkie zjawiska zachodzące w czasie.

Metoda ta nie ogranicza się jedynie do analizy przepływu danych w sieciach, ale wskazuje inne źródła opóźnień, traktując węzeł sieci jako zbiór trzech, ściśle ze sobą powiązanych, elementów:

- oprogramowanie będące systemem czasu rzeczywistego rezydujące w jednostce centralnej sterownika swobodnie programowalnego (PLC),
- koprocesor komunikacyjny i procesy w nim zachodzące,
- protokół komunikacyjny.

Wynikiem tej analizy jest nie tylko ilościowe określenie opóźnień i wyliczenie maksymalnych czasów wymiany informacji w systemie (metoda „najgorszego przypadku"- [88]), ale wskazanie sposobów polepszenia parametrów czasowych całego systemu. W

literaturze spotyka się opisy wielu metod analizy przepływu informacji w sieciach komputerowych. Są to metody bądź analityczne, oparte na pewnych uproszczonych modelach [38, 39, 74, 49, 8, 11, 23], bądź metody symulacyjne [151, 49]. Niestety, wymagania systemu czasu rzeczywistego, w których czas jest parametrem krytycznym, nie pozwalają posługiwać się metodami przybliżonymi, w określaniu parametrów czasowych przemysłowego systemu rozproszonego. Owszem, na wstępnym etapie projektowania systemu, szczególnie wtedy gdy jest on bardzo rozbudowany pod względem zarówno liczby węzłów, jak i ilości przesyłanej i liczonej w bitach informacji, można je wykorzystać, ale jedynie w celu określenia pewnych warunków brzegowych, dotyczących czasów realizacji transferu danych. W końcowej fazie projektowania i tak trzeba posłużyć się metodami bardziej dokładnymi, bowiem obie wspomniane grupy metod przybliżonych nie odpowiadają precyzyjnie na pytanie, czy sieć, a także cały system sterowania, są w stanie obsłużyć zjawiska związane choćby z tak zwaną „lawiną zdarzeń" obiektowych. I nie chodzi tu o odpowiedź obarczoną pewnym błędem, przy pewnym poziomie ufności, lecz o absolutną gwarancję realizacji obsługi i archiwizacji zdarzeń w nieprzekraczalnym, zadanym przedziale czasu. W trakcie realizacji niniejszej pracy nie spotkano (co nie znaczy, że nie istnieją) pozycji literaturowych traktujących problem determinizmu czasowego i projektowania przemysłowych systemów rozproszonych czasu rzeczywistego w prezentowanym ujęciu. Projektant współczesnych rozproszonych, komputerowych systemów sterowania musi bowiem znaleźć odpowiedź na pytanie, jak pogodzić wymagania procesu technologicznego, aktualne możliwości sprzętu i oprogramowania oraz ograniczone środki finansowe, aby stworzyć niezawodną, efektywną i rekonfigurowalną, komputerową sieć przemysłową. Niniejsza praca poświęcona jest zagadnieniom związanym z analizą przepływu danych w przemysłowych systemach rozproszonych, wykorzystujących przemysłowe sieci komputerowe i powinna, choć w części, ułatwić odpowiedź na to pytanie. Jest to tym istotniejsze, że niestety zastosowanie systemów informatycznych w przemyśle do sterowania i regulacji nie pozwala popaść projektantom w rutynę, gdyż to przede wszystkim rodzaj procesu technologicznego narzuca sposób tworzenia oprogramowania rezydującego w węzłach systemu. Słowo „niestety" oznacza mniej więcej tyle, że nie ma dwóch absolutnie identycznych procesów technologicznych i nie da się stworzyć ani identycznego oprogramowania sterującego procesem, ani identycznego oprogramowania stacji operatorskiej, ani wreszcie identycznego oprogramowania komunikacyjnego sterującego przepływem informacji. Za każdym razem proces tworzenia systemu musi rozpocząć się jego analizą czasową. Nie można zatem stworzyć sztywnych metod projektowania, które miałyby zastosowania w każdym przypadku. Można jedynie zbudować ogólny model węzła systemu, przedstawić i starać się opisać w sposób formalny wszystkie zjawiska w nim zachodzące, a szczególnie te na styku „aplikacja-koprocesor-protokół komunikacyjny", tak aby proces projektowania

sformalizować i przygotować do należytego testowania [156, 157]. Testowanie systemu nie może natomiast mieć miejsca na obiekcie przemysłowym, gdyż, z praktycznego punktu widzenia, jest to nierealne. Tym bardziej należy przywiązywać duże znaczenie do analizy przepływu informacji w systemie na etapie jego projektowania. Należy podkreślić ponadto, co było już poruszane w podrozdziale 1.1., że proponowana metoda pozwala stworzyć narzędzie informatyczne, wspomagające zarówno proces projektowania, jak i również uruchamiania systemu na obiekcie przemysłowym.

3. ROZPROSZONY SYSTEM STEROWANIA

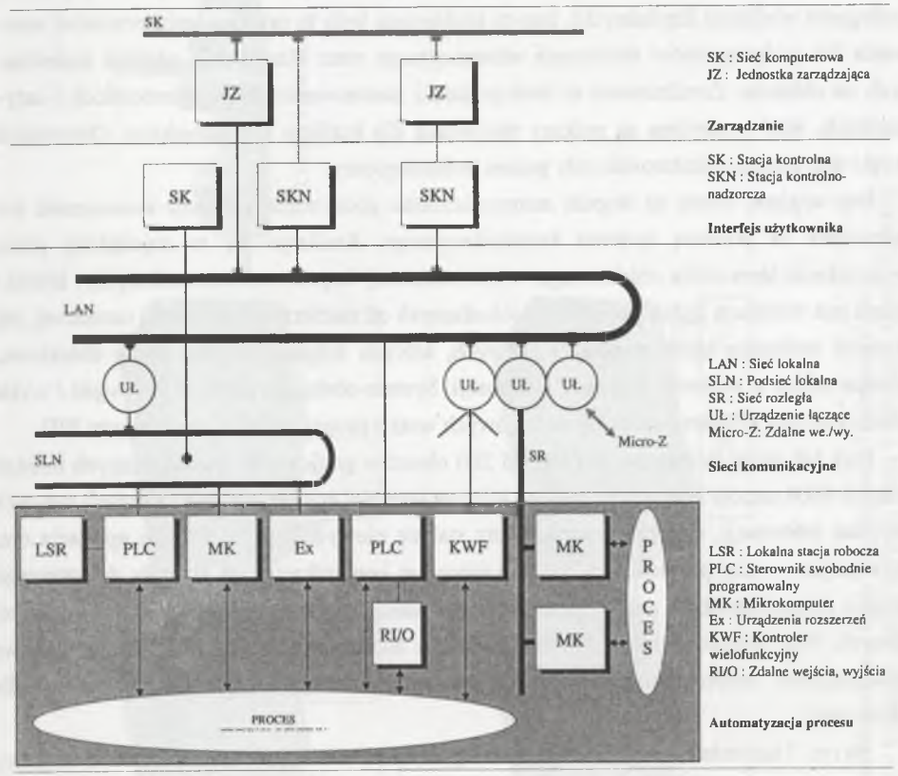
Jak wspomniano na wstępie, zaistniała konieczność stosowania systemów rozproszonych. W rozdziale 2 określono pojęcie „system rozproszony". Przypomnijmy jedynie, że gdyby podzielić, z punktu widzenia technologii, cały proces, który ma być sterowany oraz monitorowany, na podprocesy, które można traktować w dużym zakresie autonomicznie, i przydzielić dla każdego z nich autonomiczny sterownik swobodnie programowalny oraz gdyby ów podział logiczny pokrywał się z podziałem terytorialnym obiektu, to uzyskalibyśmy klasyczny system rozproszony.

Na rys. 5 przedstawiono hierarchiczną strukturę dużego, współczesnego systemu informatycznego, zarządzającego procesem technologicznym. Wyróżniono na nim kilka warstw. Od warstwy automatyzacji procesu przez sieci komunikacyjne i interfejs użytkownika po warstwę zarządzania produkcją.

Warstwa automatyzacji procesu skupia w sobie różnego rodzaju sprzęt informatyczny, którego celem jest zbieranie informacji o procesie i bezpośrednie nim sterowanie. Stosowany w tej warstwie sprzęt to [3]:

- mikrokomputery,
- sterowniki swobodnie programowalne,
- urządzenia zdalnej transmisji (w tym również typu "smart-actuators" i „smart-sensors"),
- wielofunkcyjne kontrolery oraz lokalne stacje robocze.

Charakter procesu technologicznego wymusza nie tylko typ sprzętu, który będzie wykorzystany, ale również rodzaje sieci komunikacyjnych, co również ilustruje rys. 5. Tu bowiem skupia się zdecydowana większość problemów związanych z zapewnieniem gwarantowanego czasu dostępu do medium transmisyjnego dla każdego uczestnika procesu komunikacyjnego. Ta właśnie warstwa, z punktu widzenia tematyki niniejszej pracy, jest najbardziej interesująca.



Rys. 5. Hierarchiczna struktura systemu zdecentralizowanego
Fig. 5. The structure of hierarchical distributed system

Na rys. 6 przedstawiono istniejącą i zrealizowaną przy współudziale autora (w zakresie analizy i oprogramowania komunikacji pomiędzy abonentami sieci) instalację sterowania i wizualizacji pracy stacji przygotowania i demineralizacji wody na potrzeby Elektrowni „TREBOVICE" w Czechach [145].

Jest to system rozległy, w skład którego wchodzi pięć stacji obiektowych (siedem sterowników przemysłowych) oraz dwie stacje wizualizacyjne (stacja inżynierska i stacja operatorska z systemami wizualizacyjnymi P1200). Wszystkie stacje abonenckie zostały połączone siecią N10 (protokół TOKEN-BUS), której długość przekroczyła 1 200 m.

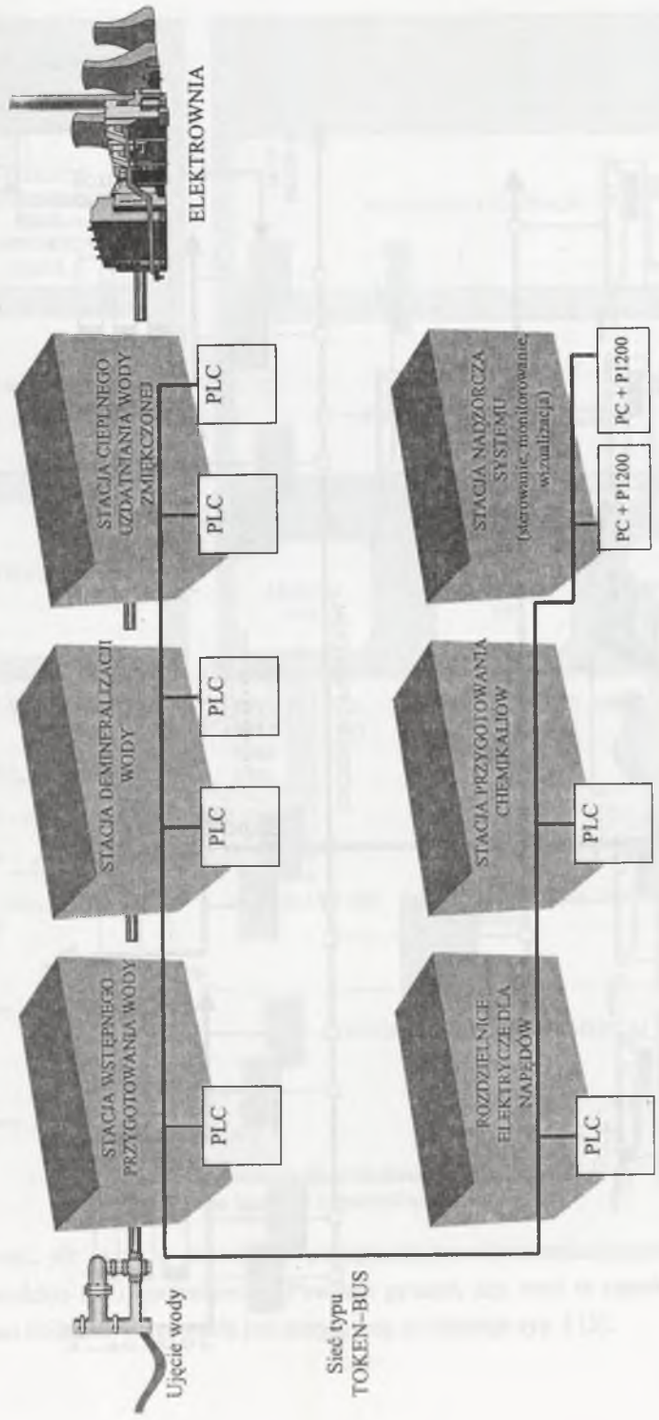
Każdy z procesów ma tu własny podsystem sterowania, zrealizowany z zastosowaniem swobodnie programowalnego sterownika lub sterowników, który jest jednocześnie abonentem sieci komputerowej. W zakresie podsystemu realizowany jest algorytm sterowania. Podsystem taki jest daleko samowystarczalny, jeśli chodzi o dane niezbędne do realizacji

algorytmu, ale żąda również pewnych danych, które powstają poza nim (na przykład pomiary analogowe wielkości fizykalnych). Innym problemem było tu uruchamianie procesów sterowania lub podprogramów sterowania sekwencyjnego oraz wizualizacja zdarzeń zachodzących na obiekcie. Zrealizowano te funkcje dzięki zastosowaniu stacji operatorskich i inżynierskich, skąd przesyłane są rozkazy sterowania dla każdego z podobiektów. Oczywiście dzięki nim można monitorować cały proces technologiczny.

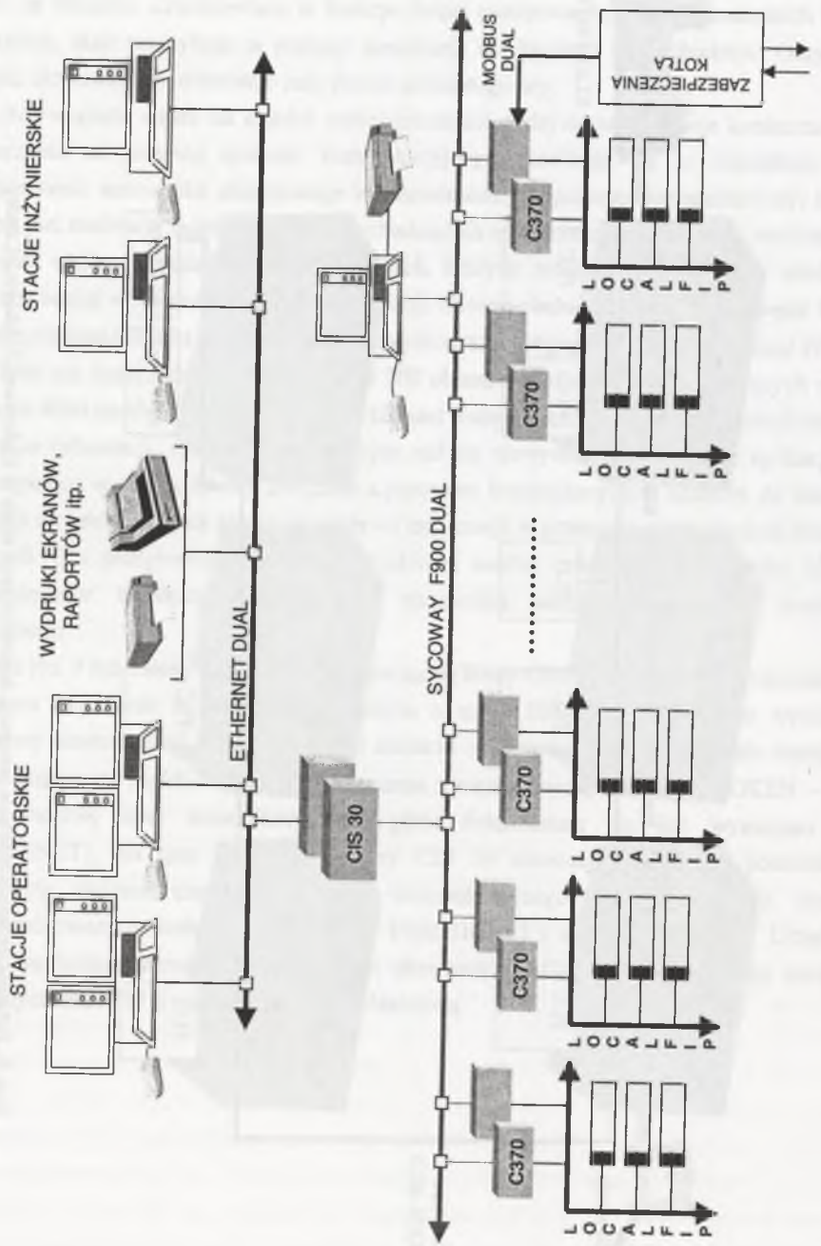
Bez względu zatem na stopień autonomiczności podsystemów istnieje konieczność ich połączenia za pomocą systemu komunikacyjnego. Realizuje się to najczęściej przez wyposażenie sterownika obiektowego w autonomiczny koprocesor komunikacyjny, którego celem jest realizacja żądań transmisji, pochodzących od macierzystej jednostki centralnej, jak również realizacja żądań transmisji zdalnych, których źródłem są inne stacje obiektowe, biorące udział w procesie wymiany informacji. System obsługuje ponad 4 000 wejść / wyjść binarnych oraz kilkaset pomiarów analogowych wraz z programowymi regulatorami PID.

Przy tak dużej liczbie danych (ponad 200 obrazów graficznych wizualizujących między innymi 4000 stanów binarnych obiektu i kilkaset wartości analogowych) i żądaniach częstych wymian informacji, system komunikacyjny stał się niewydolny. I ta właśnie aplikacja oraz występujące w niej problemy związane z procesem komunikacyjnym skłoniły do szerszego zajęcia się problematyką analizy przepływu informacji w przemysłowych sieciach komputerowych. Bez przeprowadzenia bardzo wnikliwej analizy pracy sieci nie byłoby możliwe uruchomienie automatycznego procesu sterowania stacją przygotowania wody dla elektrowni.

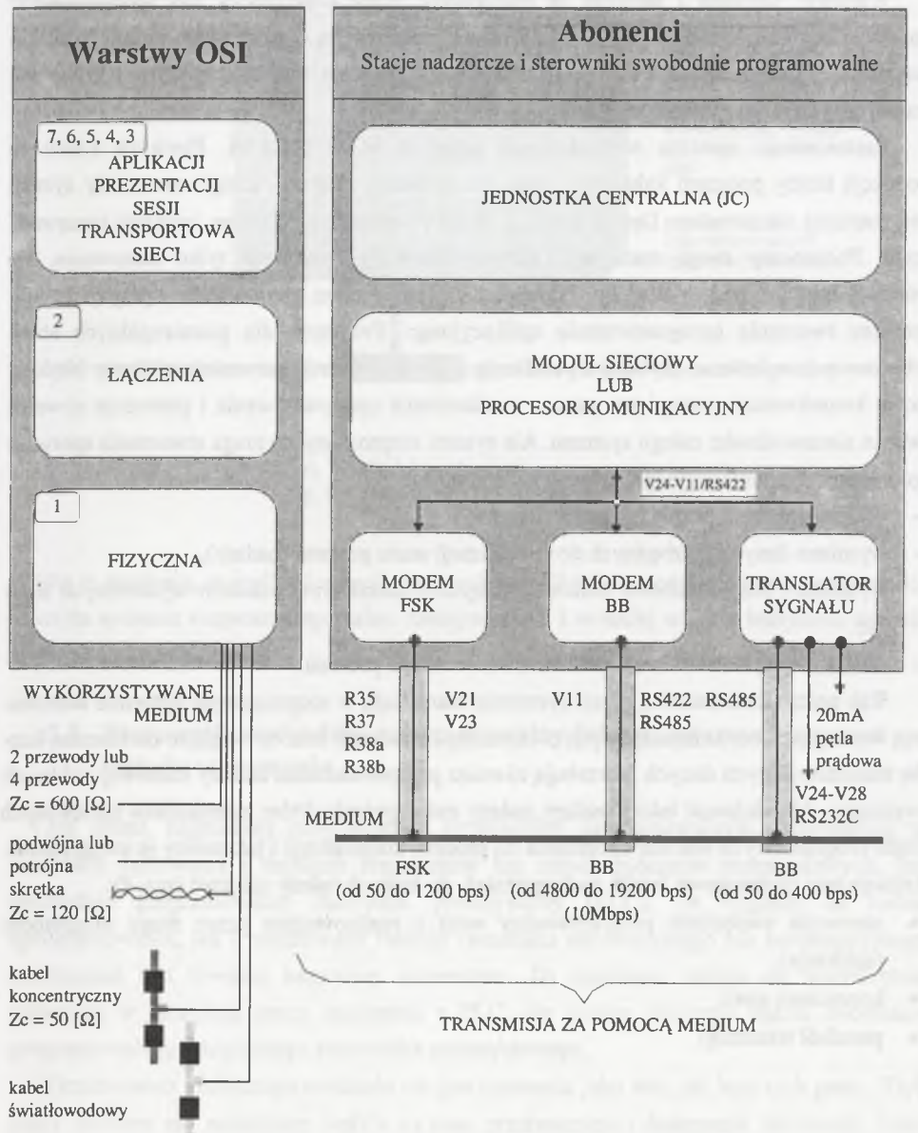
Na rys. 7 natomiast, dla porównania i za zgodą firmy CEGELEC, przedstawiono strukturę systemu sterowania blokiem energetycznym o mocy 200 MW. I tu można wyodrębnić warstwę automatyzacji z zastosowanymi sieciami o zdeterminowanym w czasie dostępie do łącza (segmenty szybkich sieci FIP połączone z lokalną siecią o dostępie TOKEN – BUS) oraz warstwę sieci komunikacyjnych, gdzie determinizm nie jest wymagany (sieć ETHERNET). Na tym rysunku elementy CIS 30 oznaczają urządzenia pośredniczące pomiędzy dwiema częściami systemu komunikacyjnego (integracja sieci): siecią o zdeterminowanym dostępie (SYCOWAY F900 DUAL) i siecią ETHERNET. Urządzenia C370 są koncentratorami danych (są to sterowniki PLC), które umożliwiają integrację lokalnych sieci FIP z wewnętrzną siecią obiektową.



Rys. 6. Schemat układu sterowania instalacją demineralizacji wody w Elektrowni Trzebovice Czechy
Fig. 6. The structure of control system for water treatment station in Power Plant Trzebovice - Czech Republic



Rys. 7. Rozproszony system sterowania blokiem energetycznym 2 x 200MW
Fig. 7. The distributed control system of power units 2 x 200MW



Rys. 8. Warstwowa budowa sieci przemysłowych
Fig. 8. The layers of industrial networks

Tak więc, jak widać, w sterowaniu i zarządzaniu procesami technologicznymi stosuje się różnego rodzaju sieci komputerowe. Powstaje pytanie, czy sieci te respektują warstwowy model sieci OSI/ISO. Odpowiedź jest pozytywna, co ilustruje rys. 8 [3].

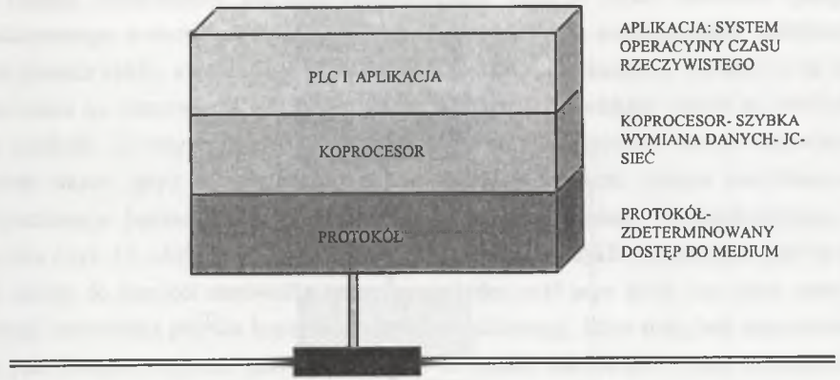
Warstwy: fizyczna i łączenia są realizowane przez koprocesory lub specjalizowane moduły komunikacyjne. Natomiast pozostałe warstwy są realizowane przez jednostki centralne, współpracujące z koprocesorami sieci. Realizacja wyższych warstw odbywa się zazwyczaj na drodze programowej.

Zastosowanie systemu rozproszonego przynosi wiele korzyści. Pierwsza z nich to redukcja liczby połączeń kablowych oraz ich globalnej długości. Dzięki temu cały system jest bardziej niezawodny. Decentralizacja zadań również czyni system bardziej niezawodnym. Podprocesy mogą realizować autonomiczne algorytmy nie tylko sterowania, ale również zabezpieczeń, blokad itp. Istotną zaletą jest również uproszczenie i przyspieszenie procesu tworzenia oprogramowania aplikacyjnego. Programy dla poszczególnych stacji obiektowych są krótsze, szybciej się realizują, a przez to są źródłem mniejszej liczby błędów, co w konsekwencji przyspiesza proces uruchamiania oprogramowania i powoduje również wzrost niezawodności całego systemu. Ale system rozproszony wymaga stworzenia sprawnego systemu komunikacyjnego, którego zadaniem jest:

- wymiana danych pomiędzy podprocesami,
- wymiana danych niezbędnych do wizualizacji stanu procesu (nadzór),
- wymiana i bezpieczeństwo transmisji danych o charakterze rozkazów wydawanych zdalnie,
- archiwizacja, alarmowanie i monitorowanie całego procesu.

Tak postawione zadania, przed systemem transmisji, w rozproszonym systemie sterowania wymagają sieci komputerowych o określonych cechach oraz ze względu na znaczną liczbę transmitowanych danych potrzebują również przeprowadzenia analizy czasowej cyklu ich wymiany. Aby dokonać takiej analizy, należy zadać pytanie, które z elementów sprzętowych bądź programowych wnoszą opóźnienia do procesu komunikacji i jak należy je uwzględniać. Istnieją trzy podstawowe źródła tych opóźnień, do których należy zaliczyć (rys. 9):

- sterownik swobodnie programowalny wraz z realizowanym przez niego programem (aplikacja),
- koprocesor sieci,
- protokół transmisji.



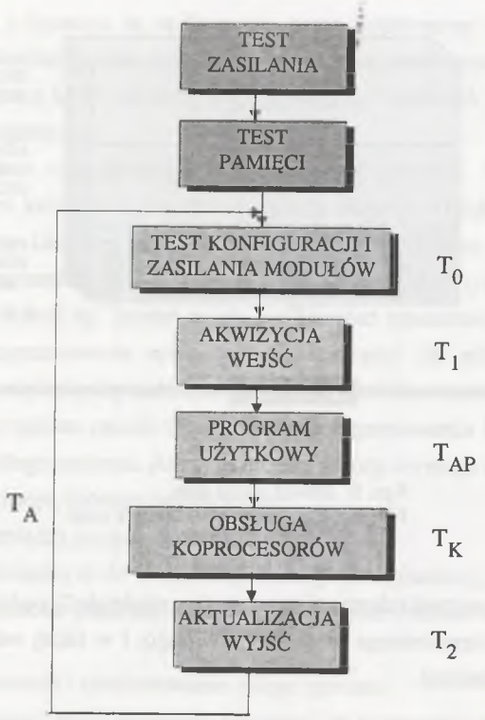
Rys. 9. Model węzła sieci
Fig. 9. The model of network's node

Źródła te stanowią, w myśl informacji zawartych w rozdziale 2, podstawowe elementy modelu węzła systemu rozproszonego czasu rzeczywistego. I w takiej właśnie kolejności zostaną poddane analizie czasowej.

3.1. Sterownik swobodnie programowalny jako podstawowy element układu sterowania

Jak dotąd, najbardziej powszechnym urządzeniem programowalnym, stosowanym w procesach sterowania i regulacji fragmentów lub całych procesów przemysłowych, jest swobodnie programowalny sterownik przemysłowy (PLC). Ze względu na rodzaj oprogramowania, jak i realizowane funkcje (automatu sekwencyjnego lub kombinacyjnego sterowania) jest również nazywany automatem. To określenie będzie się wielokrotnie pojawiało w niniejszej pracy zamiennie z PLC, ale zawsze dotyczyć będzie swobodnie programowalnego, logicznego sterownika przemysłowego.

Przedmiotem niniejszego rozdziału nie jest sterownik jako taki, ale jego tryb pracy. Tryb pracy bowiem ma zasadniczy wpływ na czas przetwarzania i dostępność informacji, które mają być przedmiotem transferu do systemu rozproszonego. Realizacja programu w czasie rzeczywistym wymaga, aby ów program był wykonywany cyklicznie. Czas trwania cyklu realizacji programu jest ważnym czynnikiem, który należy omówić. Cykliczną pracę sterownika przedstawia rys. 10.



Rys. 10. Podstawowy cykl pracy sterownika jednoprocesorowego
Fig. 10. The work cycle of single processor PLC

Na rys. 10 przedstawiono podstawowe czynności realizowane w jednym cyklu pracy. Każda z nich wnosi swój udział czasowy w całkowity czas trwania cyklu.

Wielkości:

T_0 – czas testu konfiguracji i napięć zasilających,

T_1 – czas akwizycji wejść – przepisanie fizycznych stanów wejść do pamięci,

T_2 – czas aktywizacji wyjść – przepisanie zawartości pamięci sterownika na fizyczne wyjścia,

są stałe dla danej konfiguracji i łatwo mierzalne. Natomiast czas T_{AP} dotyczy realizacji programu użytkownika (aplikacja) i jest zmienny, a jego precyzyjne zmierzenie nie jest łatwe. Nieco inaczej wygląda problem obsługi koprocesorów. Zależy on przede wszystkim od ich liczby i rodzaju. Inaczej na cykl automatu wpływa obsługa koprocesora sieci, a inaczej na przykład obsługa koprocesora do obliczeń numerycznych, jeśli takowy został zainstalowany. Tak czy inaczej czas pojedynczego cyklu automatu jest sumą pięciu wielkości:

$$T_A = T_0 + T_1 + T_{AP} + T_K + T_2$$

i nie jest stały.

Istnieją mechanizmy pozwalające na pomiar każdego cyklu automatu (programu realizowanego w sterowniku). Można ponadto, w niektórych rozwiązaniach, zadeklarować czas trwania cyklu, a każde jego przekroczenie będzie sygnalizowane. Pozwoli to na etapie testowania na oszacowanie maksymalnej jego wielkości. Największy wpływ na wielkość T_A ma wielkość T_P dotycząca czasu realizacji programu aplikacyjnego. Jest to oczywiste, ale bardzo ważne, gdyż jak się później okaże, będzie to element, którego modyfikacja lub optymalizacja będzie źródłem zwiększenia sprawności systemu komunikacyjnego. Jak wynika z rys. 10, obsługa koprocesora sieci jest elementem cyklu. Koprocesor sieci bowiem ma dostęp do pamięci sterownika tylko raz na jeden cykl jego pracy (na okres czasu T_K). Pamięć sterownika jest dla koprocesora źródłem informacji, które mają być transmitowane, ale jest również miejscem przeznaczenia dla informacji odebranych z sieci komputerowej. **Tak więc długość cyklu automatu (sterownika) wpływa bezpośrednio na sprawność wymiany informacji pomiędzy sterownikiem a koprocesorem sieci, a w konsekwencji wpływa na czas wymiany informacji w całej sieci.**

3.2. Koprocesor sieci

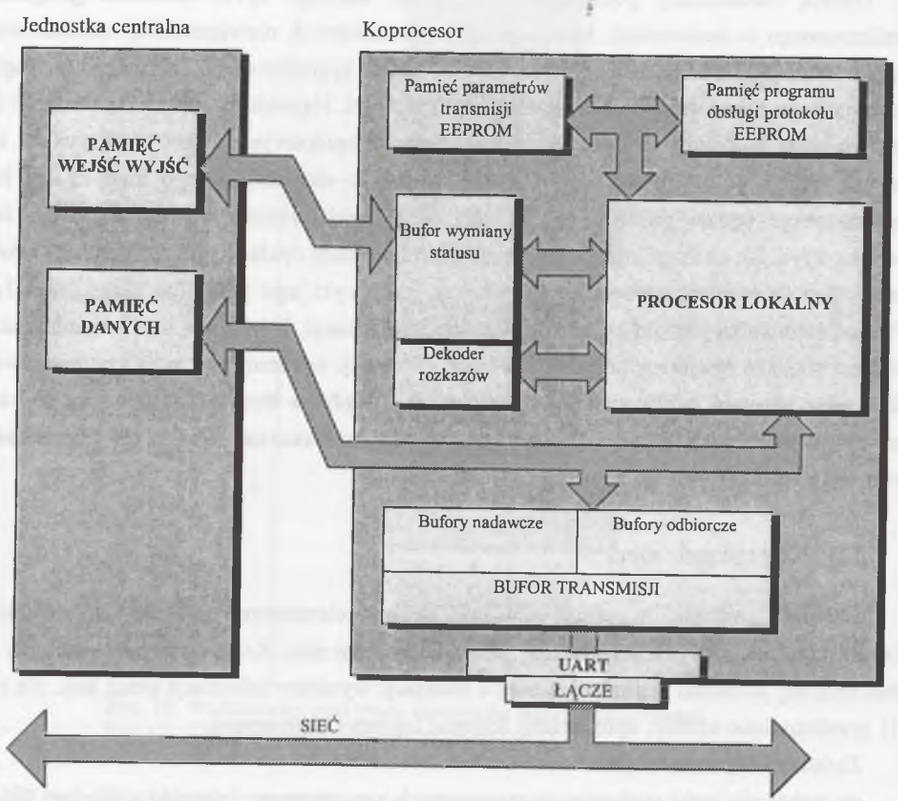
Sterownik swobodnie programowalny będący elementem systemu rozproszonego (abonentem sieci) zwykle wyposażony jest w koprocesor sieci. Koprocesor jest niezależny od macierzystej jednostki centralnej w sensie realizacji wymiany informacji przez sieć. Na rys. 11 przedstawiono ogólny, uproszczony schemat logiczny koprocesora.

Zadaniem koprocesora jest:

- na podstawie treści rozkazów otrzymywanych z macierzystej jednostki centralnej (Bufor wymiany statusu – Dekoder adresu) skopiować żądane obszary pamięci (Bufory nadawcze), utworzyć dane w postaci tak zwanych ramek transmisyjnych i w odpowiednim momencie je wytransmitować za pomocą nadajnika (UART),
- przyjąć informacje docierające siecią, zdekodować ich przeznaczenie, ewentualnie przepisać do Bufora odbiorczego i w odpowiednim momencie przepisać do pamięci macierzystej jednostki centralnej.

Obie grupy czynności, które ma wykonać koprocesor, są zazwyczaj realizowane niezależnie. Oznacza to, że macierzysta jednostka centralna nie wysyła rozkazów transmisji w sposób zsynchronizowany z pracą sieci. To samo dotyczy procesu odbioru.

Niezależność pracy koprocesora jest źródłem powstawania znacznych obciążeń sieci. Spróbujmy prześledzić zjawiska zachodzące na styku „koprocesor – jednostka centralna".



Rys. 11. Schemat logiczny koprocesora sieci
Fig. 11. The logical layout of network coprocessor

Nadawanie

Jednostka centralna może użyć, najogólniej rzecz biorąc, następujących rozkazów transmisji:

- żądanie transmisji periodycznej,
- żądanie transmisji wyzwalanej,
- zmiana trybu transmisji.

Transmisja periodyczna polega na cyklicznym, z zadanym interwałem czasowym, wysyłaniu określonego obszaru pamięci. Rozkaz taki jest wysyłany do koprocesora tylko raz, a ten po jego interpretacji transmituje do sieci żądane dane. Proces transmisji odbywa się zgodnie z określonymi zasadami, które są zdefiniowane przez protokół dostępu. Jeśli zatem

czas cyklu pracy sieci (czas pomiędzy realizacją tych samych wymian) jest dłuższy od zadanego interwału czasowego wymian periodycznych, to część z nich może być pominięta. Następuje zatem zjawisko rozsynchronizowania się żądań macierzystej jednostki centralnej z procesem transmisji w sieci. Efektem tego zjawiska jest uzyskiwanie przez odbiorcę danych, o których mówi się, że są „nieświeże". Znając graniczną wartość cyklu pracy sieci, można określić minimalny okres wymian periodycznych i już na etapie projektowania sieci pozbyć się złudzeń co do wymagań stawianych przez technologię.

Transmisja wyzwalana polega na wysłaniu przez macierzystą jednostkę centralną, rozkazu pojedynczej transakcji wymiany, określonego obszaru własnej pamięci. I tu znów może dojść do opóźnienia, gdyż realizacja rozkazu może odbyć się tylko wtedy, jeśli pozwoli na to protokół dostępu.

W obydwu przypadkach istotną rolę odgrywa czas trwania cyklu sterownika T_A. Bez względu na to, z jaką wymianą mamy do czynienia, dostęp do pamięci może nastąpić po jego zakończeniu. Toteż znajomość wartości cyklu sterownika T_A jest ważna i należy ją uwzględnić przy obliczaniu cyklu sieci.

Żądanie transmisji wyzwalanej może pojawić się wraz z realizacją wymiany cyklicznej. Zazwyczaj wyższy priorytet ma realizacja wymiany wyzwalanej. Autor oprogramowania jednostki centralnej musi mieć świadomość, że wysyłanie w sposób ciągły żądań transmisji wyzwalanej „zatka" koprocesor. W granicznym przypadku nie będzie on w stanie zrealizować żadnych wymian cyklicznych. Istnieją mechanizmy pozwalające określić realizowalność wymiany przez analizę słowa statusu wymiany. Uruchomienie kolejnej wymiany wyzwalanej musi być poprzedzone testowaniem słowa statusu, które informuje, czy poprzednia wymiana wyzwalana została zrealizowana. Obciążenie zbyt wielką liczbą wymian wyzwalanych musi doprowadzić do skomplikowania oprogramowania jednostki centralnej. Należy bowiem sterować kolejkowaniem, obsługą statusu itp., co w znacznym stopniu wydłuży cykl wymiany informacji w sieci oraz wpłynie na szybkość procesu sterowania. Opisane zjawisko również należy brać pod uwagę na etapie projektowania systemu.

Generalnie rzecz biorąc, należy ograniczyć liczbę wymian wyzwalanych, gdyż one wnoszą do analizy czasowej najwięcej niewiadomych. Oczywiste jest, że z wymian tych nie należy rezygnować. Trudno bowiem wprowadzać na listę wymian cyklicznych żądań transmisji tych zmiennych, których wartości ulegają zmianie bardzo rzadko (np. raz na godzinę). Należy jednak wykorzystywać wymiany wyzwalane do transmisji alarmów, zmiennych synchronizujących procesy (z dokładnością do cyklu sieci) itp.

Odbiór

Koprocesor generalnie może odbierać następujące polecenia:

- żądania transmisji periodycznych,
- żądania transmisji pojedynczych (wyzwalanych przez nadawcę).

Zjawiska tu zachodzące dotyczą możliwości przepełnienia buforów odbiorczych. „Bombardowanie" koprocesora przez abonentów sieci coraz to nowymi żądaniami transmisji doprowadzi do zapełnienia buforów odbiorczych i spowoduje, że koprocesor będzie negatywnie odpowiadał nadawcy lub wręcz zawiesi transmisję (*time–out dla nadawcy*). Rezultatem tego zjawiska, podobnie jak w przypadku nadawania, będzie zwiększenie stopnia złożoności oprogramowania jednostki centralnej nadawcy o procedury obsługi kolejkowania i przepływu danych, co doprowadzi, jak poprzednio, do wydłużenia cyklu sterownika z wszystkimi tego konsekwencjami dla procesu sterowania i transmisji danych.

Żądania transmisji periodycznych, przy źle dobranym interwale czasowym, doprowadzą również do rozsynchronizowania się transmisji z cyklem sterownika i jak poprzednio spowodują to, że odbiorca będzie otrzymywał nieaktualne, w sensie żądanej częstotliwości próbkowania, dane.

Zwiększenie liczby buforów odbiorczych nie rozwiązuje problemu, przesuwa go jedynie w czasie. Zawsze będą występowały zjawiska przepełnienia, a ponadto jeszcze bardziej się skomplikuje oprogramowanie sterujące koprocesora, co wydłuży jego własny czas pracy i w konsekwencji wpłynie na czas trwania cyklu sieci.

Koprocesor w każdym przypadku, czy mówimy o nadawaniu czy odbiorze, wnosi własny narzut czasowy związany z detekcją ramki i jej dekodowaniem oraz przygotowaniem ramki do transmisji (np. serializacja). Dodatkowo wymagany jest czas na transfer typu „pamięć – pamięć" pomiędzy koprocesorem a jednostką centralną. Jest on wprawdzie pomijalnie mały w porównaniu na przykład z czasem detekcji i dekodowaniem, ale należy o nim wspomnieć.

Parametrem, cechującym koprocesor, jest tak zwany czas przetwarzania ramki *Tp*, w którym są zawarte wszystkie czasy cząstkowe związane z omawianymi zjawiskami. Czas ten jest w przybliżeniu stały i nie bez kłopotów można go odszukać w dokumentacji technicznej producenta. Biorąc pod uwagę najgorszy przypadek, można określić czas niezbędny do całkowitego zakończenia transakcji wymiany pomiędzy siecią a macierzystą jednostką centralną:

$$T_K = T_P + T_A,$$

gdzie: T_K – oznacza czas transakcji „koprocesor – jednostka centralna" pojedynczej ramki danych,

T_P – oznacza czas przetwarzania pojedynczej ramki,

T_A – oznacza czas trwania cyklu automatu (czas trwania cyklu sterownika).

W przypadku gdy wszystkie bufory nadawcze i odbiorcze będą zajęte, całkowity czas transakcji (najgorszy przypadek) wymiany będzie wynosił:

$$T_K = LBN\,(T_P + T_A) + LBO\,(T_P + T_A),$$

gdzie: *LBN* – oznacza liczbę buforów nadawczych,

LBO – oznacza liczbę buforów odbiorczych.

W dalszej części pracy będzie brany pod uwagę czas T_K w odniesieniu do transakcji pojedynczej ramki, gdyż upraszcza to zdecydowanie analizę czasową przepływu informacji w całej sieci.

3.3. Protokoły dostępu do łącza stosowane w sieciach przemysłowych

Jak już wspomniano, podstawową cechą sieci przemysłowych jest gwarantowany, w określonym, nieprzekraczalnym przedziale czasu, dostęp abonenta do łącza, co wyraża się w tak zwanym gwarantowanym czasie dostępu T_{GD}.

Jedynie trzy współcześnie stosowane protokoły spełniają powyższe żądania gwarancji deterministycznego czasowo dostępu do medium dla abonenta sieci. Są to:

- protokół TOKEN – RING (TOKEN – BUS),
- protokół MASTER–SLAVE,
- protokół PRODUCENT – DYSTRYBUTOR-KONSUMENT (PDK).

Dwa pierwsze z nich doczekały się bardzo wielu implementacji i to zarówno w postaci sprzętowych koprocesorów współpracujących ze sterownikami przemysłowymi, produkowanymi przez różne firmy, jak i w postaci systemowych aplikacji obiektowych. Są więc bardzo dobrze znane i dlatego w niniejszej pracy zostaną opisane jedynie w takim zakresie, który jest niezbędny dla przeprowadzenia analizy czasowej. Trzeci z nich jest protokołem nowym i do niedawna ciągle modyfikowanym. Co prawda, jest już wielu producentów aparatury, w której stosuje się sieć FIP*(ang. Factory Instrumentation Protocol)*, będącą praktycznym przykładem implementacji protokołu PDK, to jednak temu protokołowi trzeba poświęcić dużo więcej miejsca, wyjaśniając wszystkie szczegóły.

4. SIECI O DOSTĘPIE TYPU TOKEN –BUS

Protokół TOKEN –RING (TOKEN-BUS) umożliwia budowanie sieci, które cechują się:

- gwarantowanym czasem dostępu pomiędzy abonentami tego samego segmentu,
- zdecentralizowaną architekturą,
- łatwą rozbudową w systemy rozległe,
- możliwością tworzenia struktury drzewiastej, co oznacza, że wszyscy abonenci danego segmentu sieci są na tym samym poziomie hierarchii i mogą inicjować dialog,
- możliwością nawiązywania dialogu pomiędzy segmentami sieci,
- możliwością automatycznej rekonfiguracji,
- ograniczoną długością ramki,
- możliwością tworzenia redundancji na poziomie medium i na poziomie koprocesora sieci,
- możliwością pracy sieci nawet w przypadku awarii któregoś z abonentów,
- „przezroczystością", dla jednostek centralnych abonentów sieci, wymian informacji.

Z punktu widzenia analizy czasowej są bez znaczenia funkcje (rozkazy) realizowane przez poszczególnych abonentów. Należy mieć tu na uwadze funkcje cyklicznego lub „na żądanie" inicjowania transakcji wymiany. Była już o tym mowa w rozdziale dotyczącym koprocesora sieci.

Przyjmijmy jedynie pewne, ogólne założenia dotyczące budowy ramek informacji, które są zgodne z wieloma firmowymi rozwiązaniami sieci o tym protokole.

Niech zatem :

- ramka serwisowa (w tym żeton) jest dwubajtowa,
- ramka danych jest zawsze poprzedzona 7-bajtowym nagłówkiem,
- ramki danych mają zmienną długość liczoną w bajtach, która jednak nie przekracza pewnej liczby (zwykle 256 bajtów),
- każda ramka jest zakończona dwubajtowym słowem kontrolnym (CRC16).

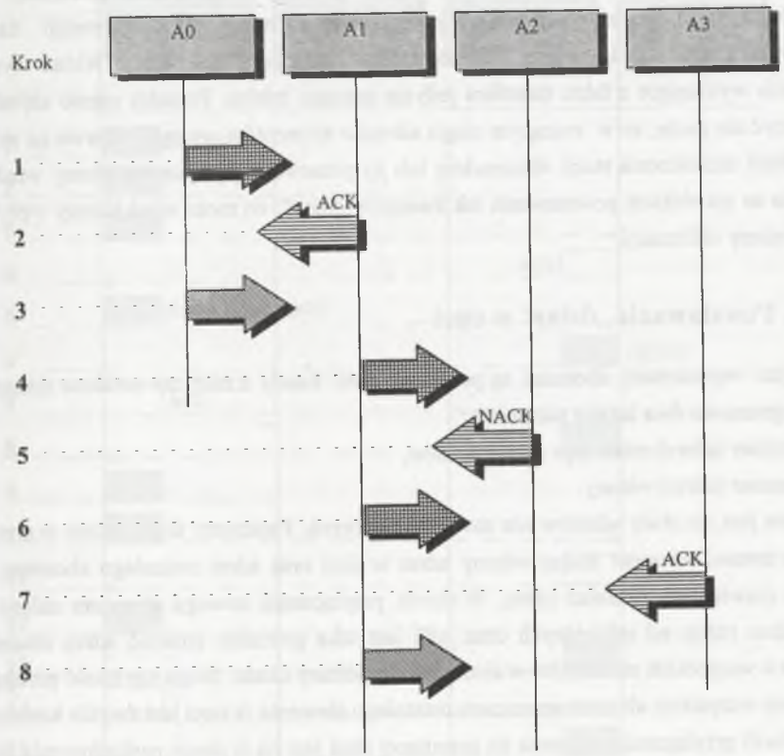
Przyjmijmy ponadto, że abonenci są ponumerowani (adres abonenta) i wartości ich adresów stanowią rosnący ciąg liczb naturalnych.

4.1. Opis wymiany informacji

Przyjmijmy, że w protokole dostępu wyróżnia się dwa typy ramek:

- wielobajtowe ramki danych,
- dwubajtowe ramki serwisowe, do których zalicza się również tak zwany żeton.

Celem zapewnienia determinizmu czasowego w sieci krąży żeton, a jego posiadacz ma prawo do transmisji ramek z danymi. Prześledźmy proces transmisji analizując rys. 12.



Rys. 12. Algorytm przekazywania żetonu
Fig. 12. The pass token algorithm

W kroku 1 abonent A0 przesyła żeton do abonenta A1 i czeka do upłynięcia czasu oczekiwania (T_{OD} – *„time-out"*) na ramkę akceptacji przyjęcia żetonu. Ma to miejsce w kroku 2. W kroku 3 abonent A0 wysyła ramkę zezwolenia na przejęcie żetonu przez abonenta A1. To kończy proces przekazywania żetonu. Powyższy proces ma miejsce, gdy sieć nie jest przeciążona lub wszystkie jej elementy pracują poprawnie. Nie zawsze przebiega on bez zakłóceń. Ilustruje to krok 4, w którym abonent A1 wysyła ramkę serwisową (żeton) do abonenta A2 i oczekuje na ramkę akceptacji (ACK). Niestety, abonent A2 odpowiada

ramką negatywnego potwierdzenia (NACK), co oznacza, że nie jest w stanie przyjąć żetonu. Może się tak zdarzyć, gdy abonent A2 ma przepełniony bufor odbiorczy. Zjawisko transmisji negatywnego potwierdzenia może wystąpić nie tylko w procesie transmisji żetonu. Może również mieć miejsce przy transmisji danych użytkowych. W naszym przykładzie abonent A1 wysyła żeton do abonenta A3 (krok 6) i uzyskuje pozytywne potwierdzenie (krok 7). Proces przekazywania „żetonu" kończy się w kroku 8. Cykl przekazywania żetonu kończy się w chwili przekazania go przez ostatnią w ciągu logicznym stację do stacji pierwszej.

Omówiony tu został mechanizm przekazywania żetonu bez transmisji danych użytkowych i w obecności wszystkich abonentów skonfigurowanej sieci. Widać wyraźne opóźnienia wynikające z faktu transferu jedynie samego żetonu. Ponadto często się zdarza lub zdarzyć się może, że w rosnącym ciągu adresów abonentów wystąpi przerwa na skutek, na przykład uszkodzenia stacji abonenckiej lub jej planowego wyłączenia. Mamy wtedy do czynienia ze zjawiskiem powstawania tak zwanych „dziur", co może mieć istotny wpływ na czas wymiany informacji.

4.2. Powstawanie „dziur" w sieci

Jak już wspomniano, abonenci są ponumerowani. Każdy z nich ma ustalone sprzętowo bądź programowo dwa istotne parametry:

- numer (adres) ostatniego abonenta sieci,
- numer (adres) własny.

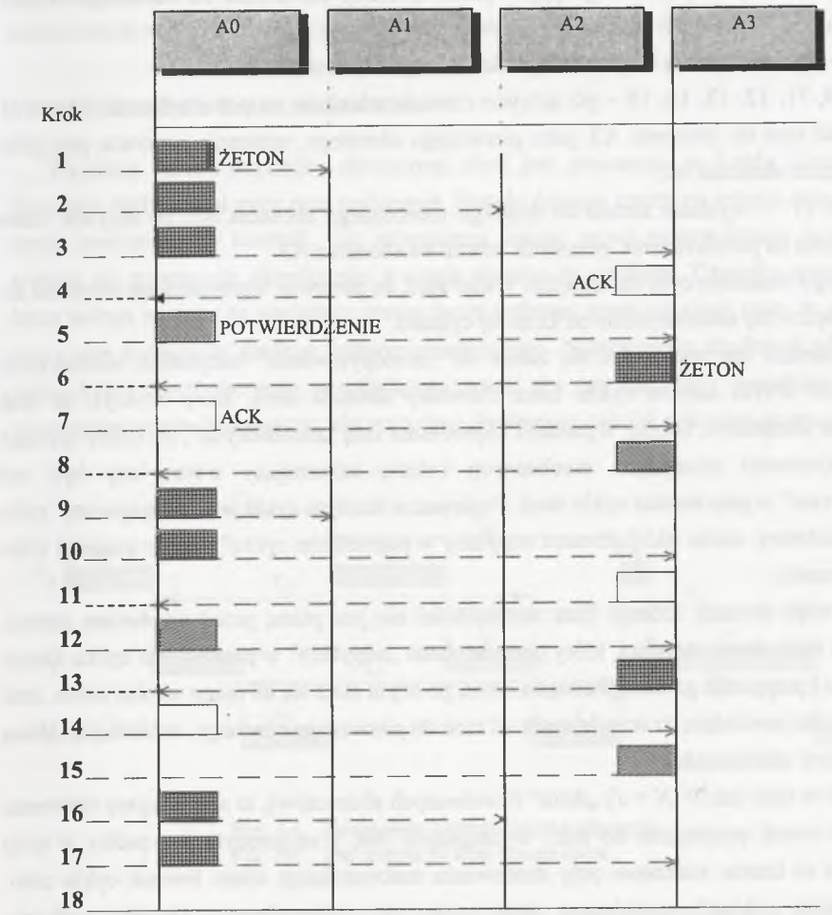
Oczywiste jest, że zbiór adresów nie zawiera powtórzeń. Parametry te są ważne w procesie przesyłu żetonu. Abonent znając własny adres w sieci oraz adres ostatniego abonenta, jest w stanie prawidłowo przesłać żeton. W chwili przyłączenia nowego abonenta należy mu nadać adres różny od istniejących oraz jeśli jest taka potrzeba, zmienić adres ostatniego abonenta u wszystkich abonentów w sieci. Jak się później okaże, druga czynność polegająca na zmianie wszystkim abonentom numeru ostatniego abonenta w sieci jest zwykle konieczna.

W chwili przyłączenia abonenta do pracującej sieci jest on w stanie nasłuchiwania. Jeżeli nie odbierze żetonu w czasie równym maksymalnemu cyklowi sieci, abonent ten sam generuje żeton zgodnie z procedurą TOKEN – PASS. Abonent posiadający żeton, znając numer własny i kolejny numer następnego abonenta, przesyła do niego żeton. Jeśli nie uzyska potwierdzenia jego przyjęcia, zapamiętuje numer nieobecnego abonenta i przesyła żeton do następnego. Czynność ta powtarza się aż do uzyskania potwierdzenia przyjęcia żetonu. Abonent, który przesłał potwierdzenie przyjęcia żetonu, jest uważany za „następnego" w sieci, a numery wszystkich poprzednich nieobecnych, którzy takiego potwierdzenia nie przesłali, są zapamiętane. Każdy nieobecny abonent tworzy w strukturze sieci „dziurę".

Jest istotna różnica pomiędzy wprowadzeniem nowego abonenta do sieci a powtórnym przyłączeniu abonenta, który już w niej egzystował. Wiąże się to zarówno z numerami

abonentów, jak i programową strukturą sieci, którą rozpoznają i zapamiętują wszyscy abonenci.

Powstanie „dziur" może mieć miejsce na skutek wadliwej konfiguracji sieci (adresy kolejnych abonentów nie rozpoczynają się od zera i nie są kolejnymi liczbami naturalnymi) lub, jak już wspomniano, wskutek awarii. Powstawanie „dziur" w sieci oraz sposób ich zapamiętywania przedstawia rys. 13.



Rys. 13. Powstanie „dziur" w sieci
Fig. 13. The „gaps" in the network

Konsekwencje stanu rzeczy przedstawionego na rys. 13 (sieć składa się z czterech abonentów) są następujące:

Krok 1 – wysłanie żetonu do abonenta A1, po upływie czasu oczekiwania na potwierdzenie, zapamiętanie „dziury" i numeru nieobecnego abonenta A1,

Krok 2 – wysłanie żetonu do abonenta A2, po upływie czasu oczekiwania na potwierdzenie, zapamiętanie „dziury" i numeru nieobecnego abonenta A2,

Krok 3 – wysłanie żetonu do abonenta A3, po uzyskaniu potwierdzenia, zapamiętanie numeru abonenta A3 jako następnego, obecnego w sieci,

Kroki 4, 5, 6, 7,8 – ponowne przyjęcie przez abonenta A0 żetonu od ostatniego w sieci abonenta A3,

Krok 9 – wysłanie żetonu do pierwszego nieobecnego abonenta A1,

Kroki 10, 11, 12, 13, 14, 15 – po upływie czasu oczekiwania na potwierdzenie, przesłanie żetonu od razu do abonenta A3, jako pierwszego obecnego, następnie ponowne przyjęcie żetonu przez abonenta A0,

Kroki 16, 17 – wysłanie żetonu do drugiego nieobecnego abonenta A2, po upływie czasu oczekiwania na potwierdzenie, przesłanie żetonu do abonenta A3.

Od tego momentu cykl się zamyka. Widać stąd, że ponowne wprowadzenie abonenta do sieci odbędzie się automatycznie po kilku jej cyklach.

Mechanizm ten sprowadza się zatem do „nieodpytywania" wszystkich nieobecnych abonentów w tym samym cyklu sieci. Dowolny abonent sieci, który wykrył, że brak kolejnych abonentów, tworzy w pamięci koprocesora listę „nieobecnych", na której zapisuje adresy (numery) następnych nieobecnych i status informujący o tym, czy byli oni „odpytywani" w poprzednim cyklu sieci. Ponieważ w każdym cyklu jest „odpytywany" tylko jeden nieobecny, status taki („abonent odpytany w poprzednim cyklu") będzie posiadał tylko jeden abonent.

Tak więc abonent, którego lista nieobecności nie jest pusta, przed wysłaniem żetonu, szuka na niej adresu abonenta, który posiada status „odpytany" w poprzednim cyklu, kasuje ów status i przypisuje go następnemu na liście, po czym stara się do niego wysłać żeton. Jeśli próba się nie powiedzie, to wysyła żeton od razu do pierwszego obecnego, zaniechując dalszą analizę listy nieobecności.

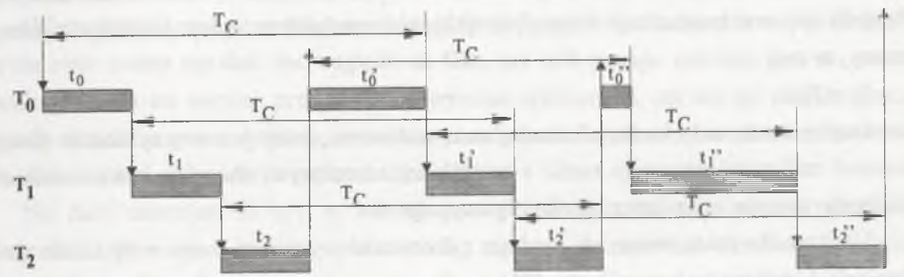
Jeżeli w sieci jest N ($N > 1$) „dziur" (nieobecnych abonentów), to aby wszyscy nieobecni abonenci zostali przyłączeni do sieci, wymaganych jest, w najgorszym przypadku, N cykli sieci. Ma to istotne znaczenie przy szacowaniu maksymalnego czasu trwania cyklu sieci. Powstawanie „dziur" w strukturze sieci może być niebezpieczne z punktu widzenia gwarantowanego czasu dostarczenia danych do abonentów sieci. Jeśli podczas konfigurowania sieci nie wzięto pod uwagę tego zjawiska, to może się okazać, że poprawnie działająca sieć w obecności wszystkich abonentów będzie pracować wadliwie (wydłużenie cyklu pracy), gdy któregoś zabraknie.

Przyjęcie takiego mechanizmu wprowadzania abonenta po jego chwilowej nieobecności pozwala na automatyczne przyłączenie go do pracy w sieci bez konieczności jej restartowania. Chyba że nieobecność abonenta jest planowana na dłuższy okres czasu, to być może warto wtedy całą sieć przekonfigurować. Nie jest to proste, szczególnie wtedy gdy w sieci pracują stacje wizualizacyjne, które pobierają od tego abonenta dane. Wymaga to rekonfiguracji stacji nadrzędnej, zmiany elementów obrazu itp.

Istnieje również inny mechanizm reagowania na powstawanie „dziur" w sieci, ale o nim będzie mowa w rozdziale dotyczącym sieci o dostępie „MASTER–SLAVE".

4.3. Kontrola czasu nadawania

Wymiana żetonu pomiędzy abonentami sieci jest gwarancją, że każdy z uczestników transmisji będzie miał swój czas nadawania. Reguła dostępu oparta na żetonie, bez dodatkowych mechanizmów kontroli, nie zabezpiecza jednak przed monopolizacją łącza, a tym samym nie gwarantuje określonego w czasie dostępu do medium. Zjawisko monopolizacji łącza polega na tym, że posiadacz żetonu może nadawać przez tak długi czas, że gwarantowany czas dostępu do medium zostanie przekroczony. Należy zatem zbudować taki mechanizm, który zmusi abonenta posiadającego żeton, aby ten zakończył transakcję wymiany w żądanym czasie. Mechanizm taki może być zbudowany, tak jak pokazano na rys. 14.



Rys. 14. Mechanizm kontroli czasu nadawania
Fig. 14. The control of time transmission

Stacja abonencka A_0 z chwilą uzyskania żetonu rozpoczyna nadawanie i jednocześnie w niej rozpoczyna się odliczanie czasu T_C, który jest czasem trwania cyklu sięci. Transmisja trwa przez czas t_0, a następnie żeton jest przekazany do stacji A_1. Tu proces się powtarza. Rozpoczyna się transmisja, która trwa czas t_1 i równocześnie rozpoczyna się odliczanie czasu T_C. W stacji abonenckiej A_2 proces jest identyczny. Gdy znów stacja A_0 otrzyma żeton, może transmitować jedynie przez czas, który jej pozostał do upłynięcia czasu trwania cyklu T_C.

Rozpoczyna ponadto odliczanie kolejnego interwału czasowego równego T_C. Omawiany mechanizm zapewnia dostęp do medium każdemu abonentowi po czasie nie dłuższym niż cykl sieci. Ale z drugiej strony nakłada na projektanta obowiązek oszacowania czasu T_C. Złe obliczenie czy oszacowanie czasu T_C spowodować może rozsynchronizowanie pracy sieci na wiele okresów, podczas których krążyć będzie jedynie żeton. Zjawisko to spowoduje wydłużenie cyklu wymiany informacji w całej sieci. Jest to znaczący problem, tym bardziej że nie jest możliwe wyliczenie czasu, o który ten proces się wydłuży. Trudności w obliczeniu opóźnienia mogą być spowodowane brakiem możliwości określenia liczby i rozmiaru „bloków" transmisji w systemach, w których następuje duża liczba wymian, ze szczególnym uwzględnieniem wymian wyzwalanych (na żądanie). Ma to miejsce zwłaszcza w sytuacjach awaryjnych procesu technologicznego, który jest obsługiwany przez sieć. Pojawienie się lawiny zdarzeń, które mają być obsłużone przez sieć, spowodować może niekontrolowany wzrost żądań obsługi sieci, a co za tym idzie zmniejszenie przepustowości sieci czy wręcz ustanie procesu transmisji danych użytkowych. Dlatego tak ważne jest określenie czasu trwania cyklu sieci, do czego potrzebna jest precyzyjna, jak tylko się da, analiza wszystkich wymian danych w konfigurowanej sieci. Kontrola czasu nadawania zapewnia, że czas transmisji poszczególnego abonenta nie przekroczy czasu trwania cyklu sieci T_C. Jest to warunek graniczny i nie należy tak konfigurować sieci, aby czas nadawania był równy czasowi T_C, ale zdecydowanie mniejszy. Można wprowadzić dodatkowy mechanizm, który indywidualnie narzuci wartość przyznanego każdemu abonentowi czasu nadawania T_N. Spotyka się w rozwiązaniach firmowych [33] taki mechanizm, który powoduje z jednej strony, że czas

$$T_N < T_C,$$

a z drugiej strony zaś określa minimalny czas nadawania, który jest w przybliżeniu równy czasowi całkowitej transmisji ramki o największej z możliwych długości. Jest to wielkość parametryzowana i określana przez konfigurującego sieć.

Limitowanie czasu transmisji każdego z abonentów wymuszać może w tej klasie sieci istnienie dwóch typów komunikatów [33]:

- komunikaty o zwykłej pilności,
- komunikaty o najwyższym priorytecie pilności.

Dlatego projektant sieci powinien pamiętać o kilku istotnych regułach nadawania, po uzyskaniu przez abonenta żetonu:

- najpierw powinien być transmitowany komunikat najpilniejszy z listy komunikatów superpilnych (wybór, który z komunikatów jest najpilniejszy, narzuca zwykle proces technologiczny – jest to jednak zawsze zadanie programisty),
- w następnej kolejności wymagana jest emisja komunikatów pilnych, pod warunkiem że nie upłynął czas nadawania,
- w ostatniej kolejności emitowane mogą być komunikaty o zwykłym stopniu pilności, pod warunkiem że nie upłynął czas nadawania.

Wszystkie komunikaty, których nadawanie rozpoczęło się w chwili upłynięcia czasu nadawania, będą wyemitowane w całości w trakcie kolejnych cykli sieci. Kontrola czasu nadawania jest jednym z najistotniejszych mechanizmów zapewniających duży poziom niezawodności, ale prawidłowa jego praca jest uwarunkowana prawidłową konfiguracją sieci, na którą składają się przede wszystkim:

- prawidłowe dobranie cyklu pracy sieci,
- prawidłowa sprzętowa konfiguracja sieci pozwalająca zminimalizować wpływ powstawania „dziur" w sieci, co omawiano w poprzednich rozdziałach.

Wszystko to wpływa w efekcie na sprawne działanie systemu komunikacyjnego.

4.4. Cykl sieci a cykl wymiany informacji

W kolejnych rozdziałach bardzo często będzie się pojawiało określenie „cykl sieci" i „cykl wymiany informacji". Nie są to, w rozumieniu autora, pojęcia tożsame. Spróbujmy zatem na potrzeby niniejszej pracy określić znaczenie obu pojęć.

Przez „cykl sieci" T_C, czy też „cykl pracy sieci" należy rozumieć czas, który upływa od momentu rozpoczęcia przez abonenta „N" transmisji do momentu uzyskania przez niego kolejnego żetonu.

Należy dążyć i zazwyczaj tak jest, aby cykl sieci T_C był stały dla danej instalacji. Stałość cyklu sieci można uzyskać bez względu na fakt, czy sieć pracuje stabilnie, jest w „stanie ustalonym", co ma miejsce przy realizacji wymian cyklicznych, czy też jej stabilna praca została naruszona przez realizację wymian wyzwalanych. Zapewnia to mechanizm kontroli czasu nadawania omówiony w poprzednim rozdziale.

Nie musi natomiast tak być, że w czasie trwania jednego cyklu sieci zostaną przesłane wszystkie informacje. Ustalając cykl sieci T_C, staramy się zapewnić określony w czasie dostęp do medium każdego z abonentów. Są to wymagania procesu technologicznego. Z ustalenia czasu T_C wynika jakiś podział czasu nadawania przypadający na każdego uczestnika procesu wymiany informacji. Może się okazać, że dla danego abonenta jemu przydzielony czas nie jest wystarczający do wyemitowania wszystkich danych. Trzeba wtedy kilku czy kilkunastu cykli sieci, aby wszystkie dane zostały przesłane.

Całkowity czas potrzebny do wyemitowania wszystkich danych przy założeniu „stabilnej" pracy sieci można nazwać cyklem wymiany informacji T_{CW}. Najlepiej by było, aby cykl pracy sieci był równy cyklowi wymiany informacji ($T_C = T_{CW}$), ale w wielu przypadkach stosowania sieci przemysłowych nastąpiłoby znaczne wydłużenie cyklu sieci i ograniczyłoby to zakres jej stosowania. Z drugiej zaś strony, w systemach sterowania ma miejsce szereg

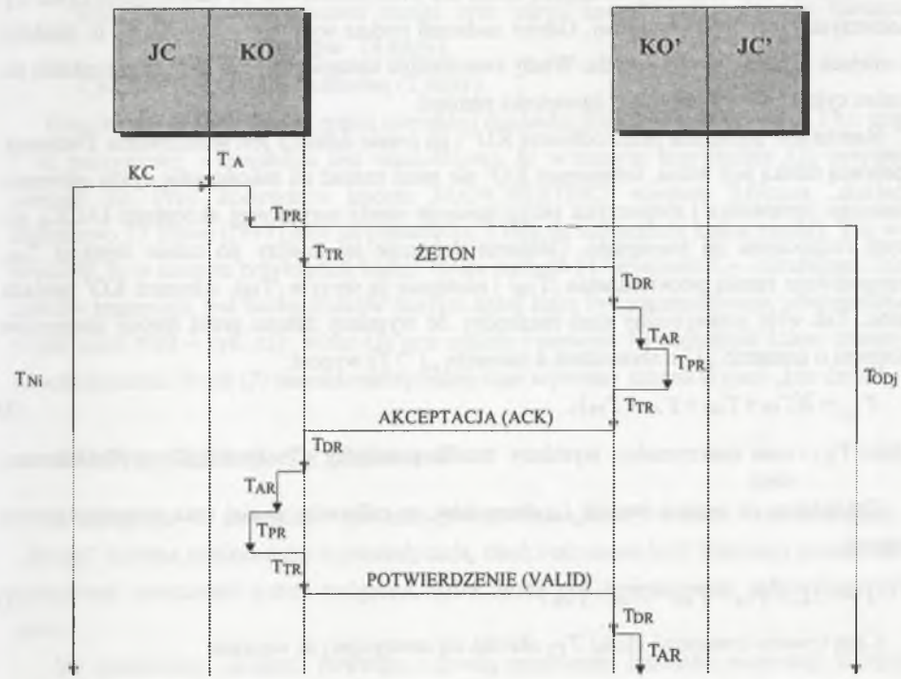
transmisji danych, których pilność dostarczenia nie musi być krytyczna. Można wtedy pozwolić na ich transmisję partiami, w kolejnych cyklach sieci *Tc*.

4.5. Określenie czasu trwania cyklu sieci

Z przeprowadzonych dotychczas rozważań wypływa kilka wniosków. Po pierwsze, rzeczą najistotniejszą przy konfigurowaniu sieci jest obliczenie maksymalnego czasu trwania cyklu sieci, gdyż parametr ten ma decydujące znaczenie dla określenia czasów nadawania przez poszczególnych abonentów. Po drugie, maksymalny czas trwania cyklu rzutuje na podjęcie decyzji o przydatności określonej sieci dla realizacji zadania. Może się bowiem okazać, że przy przyjętych założeniach dotyczących liczby abonentów, liczby żądanych wymian, ilości informacji wymaganych do transmisji, liczby wymian o najwyższym priorytecie pilności, sieć nie jest w stanie sprostać zadaniom. Wtedy należy podjąć decyzję dotyczącą albo zmiany typu sieci, albo przekonfigurowania istniejącej. Problem dotyczący zmiany konfiguracji i optymalizacji sieci będzie poruszany w innym rozdziale niniejszej pracy. Umiejętność odpowiedzi na pytanie o przydatności sieci do realizacji, na przykład funkcji sterowania obiektem, jest niezmiernie istotna. Może ustrzec przed wieloma problemami, które mogą wystąpić po uruchomieniu systemu na obiekcie. Analiza zjawisk zachodzących w pracującym systemie nie jest łatwa, a czasami mogą one powodować błędną pracę systemu, co z kolei może doprowadzić do błędnej pracy obiektu z wszystkimi tego konsekwencjami wynikającymi z procesu technologicznego.

4.6. Maksymalny czas wymiany żetonu

Przed przystąpieniem do obliczenia maksymalnego czasu wymiany żetonu przyjrzyjmy się raz jeszcze zjawiskom zachodzącym na styku „sieć – koprocesor" i „koprocesor – jednostka centralna". Ilustruje to rys. 15.



Rys. 15. Analiza czasowa transmisji żetonu
Fig. 15. The timing of token transmission

Na rys. 15 poszczególne symbole oznaczają:

JC – jednostka centralna nadawcy,
KO – koprocesor sieciowy nadawcy,
JC' – jednostka centralna odbiorcy,
KO' – koprocesor sieciowy odbiorcy,
KC –„koniec cyklu automatu" – sygnał przerwania JC do koprocesora,
T_{ODj} – maksymalny, definiowany czas odpowiedzi (*ang. time – out*),
T_{Ni} – definiowany czas nadawania,
T_A – czas trwania cyklu automatu (najgorszy przypadek),
T_{PR} – czas programowego przygotowania ramki (tu: żeton),
T_{DR} – czas detekcji ramki (jest to na przykład w systemie kodowania MANCHESTER2, czas transmisji 3.5 znaku przy przyjętej prędkości transmisji),
T_{AR} – czas analizy ramki,
T_{TR} – czas transmisji ramki (tu: czas transmisji ramki serwisowej).

Jednostka koprocesora posiadająca żeton nie musi oczekiwać na zakończenie cyklu T_A macierzystej jednostki centralnej. Gdyby nadszedł rozkaz wymiany wyzwalanej, to miałoby to miejsce w czasie trwania cyklu. Wtedy ewentualnie nastąpiłaby konieczność zaczekania na koniec cyklu celem przepisania zawartości pamięci.

Ramka jest odbierana przez odbiorcę KO' i po czasie detekcji jest analizowana. Ponieważ odebraną ramką jest żeton, koprocesor KO' nie musi czekać na zakończenie cyklu automatu własnego sterownika i rozpoczyna przygotowanie ramki serwisowej akceptacji (ACK), po czym rozpoczyna jej transmisję. Odbiorca dokonuje jej analizy po czasie detekcji T_{DR}, przygotowuje ramkę potwierdzenia (T_{PR}) i następnie ją wysyła (T_{TR}). Abonent KO' posiada żeton. Tak więc maksymalny czas niezbędny do wymiany żetonu przez dwóch abonentów (abonent o numerze „*i*" z abonentem o numerze „*j*") $T_{\dot{Z}}$ wynosi:

$$T_{\dot{Z}i,j} = 3(T_{TR} + T_{DR} + T_{AR} + T_{PR}), \tag{1}$$

gdzie: $T_{\dot{Z}i,j}$ – czas maksymalny wymiany żetonu pomiędzy „*i*" – tym a „*j*" – tym abonentem sieci.

Zakładając, że w sieci istnieje L_A abonentów, to całkowity w niej czas wymiany żetonu wynosi:

$$T_{\dot{Z}} = 3L_A(T_{TR} + T_{DR} + T_{AR} + T_{PR}) \tag{2}$$

Czas trwania transmisji ramki T_{TR} określa się następującym wzorem:

$$T_{TR} = \frac{L_{ZT} L_{BZ} + B_S}{V}, \tag{3}$$

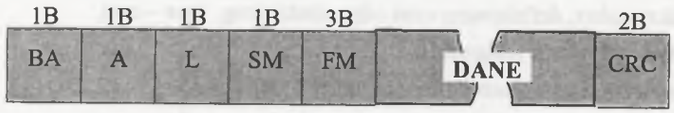
gdzie: V – jest prędkością transmisji wyrażoną w bitach na sekundę (b/s),

L_{ZT} – jest liczbą znaków transmisji,

L_{BZ} – jest liczbą bitów przypadających na jeden znak transmisji,

B_S – jest liczbą bitów serwisowych związanych ze sposobem kodowania (warstwa liniowa).

Dla przykładu, w firmowej sieci N10 [33] postać ramki jest następująca (rys. 16):



Rys. 16. Postać ramki w sieci N10
Fig. 16. The frame of N10 network

gdzie: BA – oznacza adres nadawcy (1 bajt),

A – oznacza adres przeznaczenia (1 bajt),

L – oznacza długość ramki (1 bajt),

FM+SM – oznacza numer ramki oraz adres komunikatów w trybie transmisji komunikatów (4 bajty),

CRC – oznacza sumę kontrolną (2 bajty).

Przyjmując, że do każdego znaku transmisji dochodzi dodatkowo na przykład 1 bit stopu, 1 bit parzystości, a transmisja jest ośmiobitowa, to w naszym przykładzie L_{BZ} przyjmuje wartość 10. Przy kodowaniu kodem MANCHESTER2 warstwa fizyczna „dokłada" dodatkowo 11 bitów ((5+3) bity na preambułę, 3 bity na oznaczenie końca ramki). Tak więc wielkość B_S w naszym przykładzie będzie miała wartość 11. Wielkość L_{ZT}, określająca liczbę znaków transmisji, jest liczbą znaków danych, które mają być transmitowane, powiększoną o 9 (dla sieci N10 – rys. 15). Wzór (*3*) jest ogólny i pozwala na obliczenie czasu transmisji dowolnej ramki. Wzór (*2*) określa maksymalny czas wymiany żetonu w sieci „bez dziur".

4.7. Przypadek sieci z „dziurami"

W rozdziale 4.2 omawiano zjawisko powstawania „dziur" w sieci. Trudno przyjąć, że „dziury" zostaną zaplanowane z premedytacją, choć i tak może być! Niemniej jednak należy oszacować opóźnienie z nimi związane, które może być wniesione do całkowitego cyklu sieci.

Jak pamiętamy, „dziura" powstaje z chwilą przerwania łańcucha numeracji kolejnymi liczbami naturalnymi abonentów sieci. Powoduje ona opóźnienie wynoszące T_{0Di}. Jest to czas odpowiedzi „*i*" – tego abonenta. Krótko mówiąc, brak „*i*" tego abonenta powoduje oczekiwanie wynoszące T_{0Di}. Czas ten jest wielkością parametryzowaną. Wobec tego opóźnienie wnoszone przez „dziury" do cyklu sieci wynosi:

$$T_D = \sum_{i=1}^{L_D} T_{ODi} + L_D(T_{TR} + T_{PR}), \tag{4}$$

gdzie: L_D oznacza liczbę „dziur" powstałych lub zakładanych w sieci.

Jest to czas bezpowrotnie stracony.

Wzór (*4*) jest prawdziwy dla sieci bez redundancji medium transmisyjnego. Czas nim wyrażony ulega zdwojeniu, gdy ma miejsce redundancja. Jest to związane z tym, że abonent, który chcące wysłać żeton, a nie uzyska odpowiedzi, próbuje powtórnie go przesłać linią zapasową, według tych samych, jak poprzednio, reguł. Wobec tego, dla sieci z redundancją czas T_D (teraz oznaczany jako T_{DR}) wyniesie:

$$T_{DR} = 2\sum_{i=1}^{L_D} T_{ODi} + 2(L_D T_{TR} + T_{PR}) \tag{5}$$

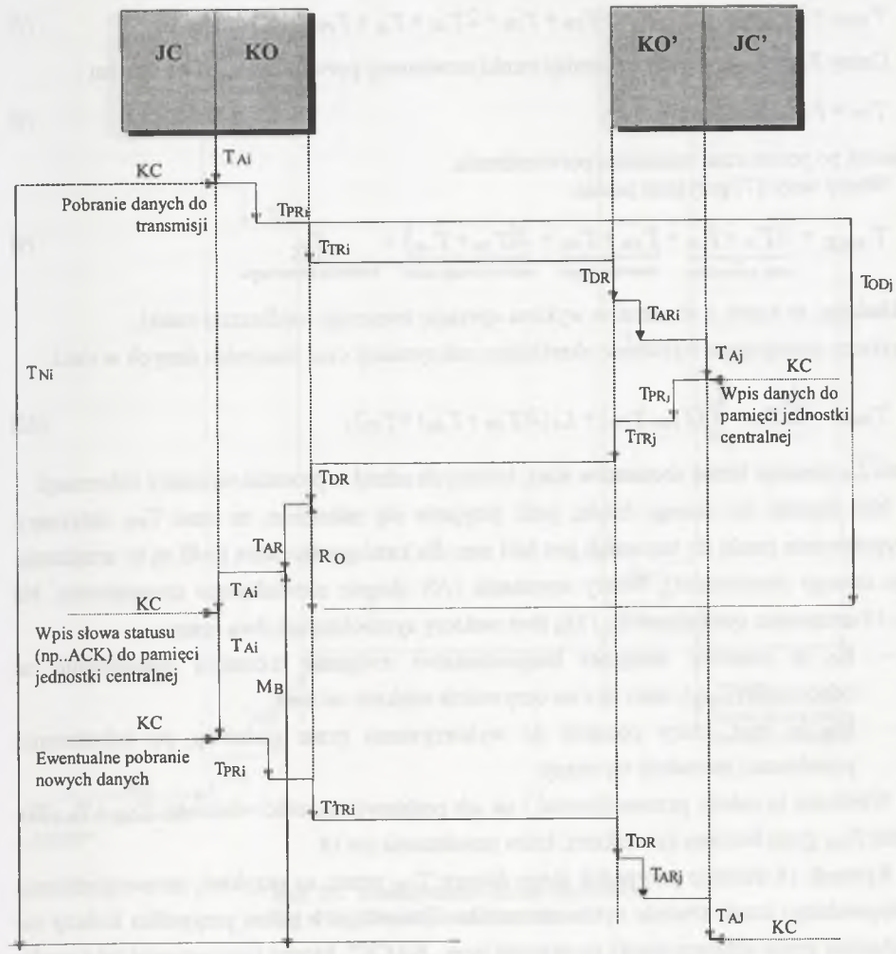
Całkowity, maksymalny czas wymiany żetonu w całej sieci można ująć następująco:

$$T_{MZ} = T_Z + T_D = 3L_A + 3L_A(T_{TR} + T_{DR} + T_{AR} + T_{PR}) + \\ + \sum_{i=1}^{L_D} T_{ODi} + L_D(T_{TR} + T_{PR}) \qquad (6)$$

4.8. Maksymalny czas trwania cyklu sieci z transmisją danych użytkowych

Przystępując do określenia czasu wymiany informacji w sieci z transmisją danych użytkowych prześledźmy, jak poprzednio, zjawiska zachodzące na styku „jednostka centralna – koprocesor" i „koprocesor – sieć". Zjawiska te przedstawiono symbolicznie na rys. 17. W tym przypadku mamy do czynienia tylko z transmisją żetonu. Na tym samym rysunku przedstawiono zależności czasowe dotyczące transmisji danych użytkowych, mające miejsce bądź podczas wymian cyklicznych, bądź wyzwalanych (na „żądanie"). Proces ów jest realizowany po otrzymaniu, detekcji i analizie ramki serwisowej kończącej proces pozyskiwania żetonu. Krótko mówiąc, stacja „i"-ta otrzymała żeton i rozpoczyna transmisję.

Nadawca musi zaczekać na zakończenie trwania cyklu sterownika T_{Ai}. Do analizy będzie się przyjmować maksymalną wartość tego czasu. Na rysunku 17 wektory nazwane T_{Ai} są różnej długości (podobnie zresztą jak wektory nazwane T_{Aj}). Jest to zrobione specjalnie, aby zobrazować fakt, że oczekiwanie na koniec cyklu sterownika może być różne, ale, co należy podkreślić, do analizy będzie brany maksymalny czas trwania cyklu. W chwili detekcji końca cyklu przez koprocesor nadawczy (sygnał KC – koniec cyklu) następuje przepisanie z pamięci jednostki centralnej danych, które mają być przetransmitowane. Po przygotowaniu ramki, jej transmisji oraz detekcji i analizie u odbiorcy musi on zaczekać na koniec cyklu swojego sterownika, co oznaczono wektorami T_{Aj}. W chwili otrzymania sygnału KC koprocesor odbiorcy przepisuje dane do pamięci jednostki centralnej, po czym przygotowuje ramkę odpowiedzi z odpowiednią zawartością dotyczącą statusu operacji. Po otrzymaniu detekcji i analizie ramki odpowiedzi nadawca danych znów musi czekać na zakończenie cyklu własnej jednostki centralnej (sygnał KC), by wpisać do słowa statusu wymiany jego wartość. Jeśli nie upłynął czas nadawania T_{Ni}, nadawca może spróbować rozpocząć transmisję kolejnej porcji danych.



Rys. 17. Analiza czasowa transmisji danych użytkownika
Fig. 17. The timing of useful data transmission

Jak widać na rys. 17, jest to możliwe, ale w praktyce zazwyczaj jedna transmisja danych przypada na jeden cykl automatu. Upraszcza to zdecydowanie analizę i program obsługi sterowania transmisją. Przyjmując do analizy założenie, że blok transmisji może dotyczyć ramki maksymalnej długości, uzyskamy następującą zależność na maksymalny czas trwania transmisji ramki danych użytkowych pomiędzy „i" – tym a „j" – tym abonentem.

$$T_{MAXi} = 2T_{Ai} + T_{PRi} + 2T_{DR} + T_{TRi} + T_{TRj} + 2T_{AR} + T_{Aj} + T_{PRj} \quad (7)$$

Czasy T_{TRj} i T_{PRj} dotyczą transmisji ramki serwisowej potwierdzenia. Niech zatem

$$T_{TP} = T_{TRj} + T_{PRj} \quad (8)$$

oznacza po prostu czas transmisji potwierdzenia.

Wtedy wzór (7) przyjmie postać:

$$T_{MAX_i} = \underbrace{2T_{Ai} + T_{Aj}}_{\text{cykle sterowników}} + \underbrace{T_{PRi} + T_{TRi}}_{\text{transmisja danych}} + \underbrace{2(T_{DR} + T_{AR})}_{\text{analiza i detekcja ramek}} + \underbrace{T_{TP}}_{\text{transmisja potwerdzenia}} \quad (9)$$

Zakładając, że każdy z abonentów wykona operację transmisji najdłuższej ramki, uzyskamy następujące wyrażenie określające maksymalny czas transmisji danych w sieci.

$$T_{MAX} = 3\sum_{i=1}^{L_A} T_{Ai} + \sum_{i=1}^{L_A}(T_{PRi} + T_{TRi}) + L_A(2(T_{DR} + T_{AR}) + T_{TP}), \quad (10)$$

gdzie L_A oznacza liczbę abonentów sieci, biorących udział w procesie wymiany informacji.

Nie popełni się dużego błędu, jeśli przyjmie się założenie, że czas T_{PRi} dotyczący przygotowania ramki do transmisji jest taki sam dla każdego abonenta (jeśli są to urządzenia tego samego producenta!). Wtedy wyrażenie (*10*) ulegnie niewielkiemu uproszczeniu. Na rys. 17 oznaczono symbolami R_O i M_B dwa wektory symbolizujące dwa czasy:

- R_O to czasowy margines bezpieczeństwa związany z czasem oczekiwania na odpowiedź (T_{ODj}), musi być on oczywiście większy od zera,
- M_B to czas, który pozostał do wykorzystania przez nadawcę, po zakończeniu pojedynczej transakcji wymiany.

Wielkości te należy przeanalizować i na ich podstawie określić wielkości T_{ODj} i T_{Ni}. Zły dobór T_{ODj} grozi bowiem zjawiskiem, które przedstawia rys 18.

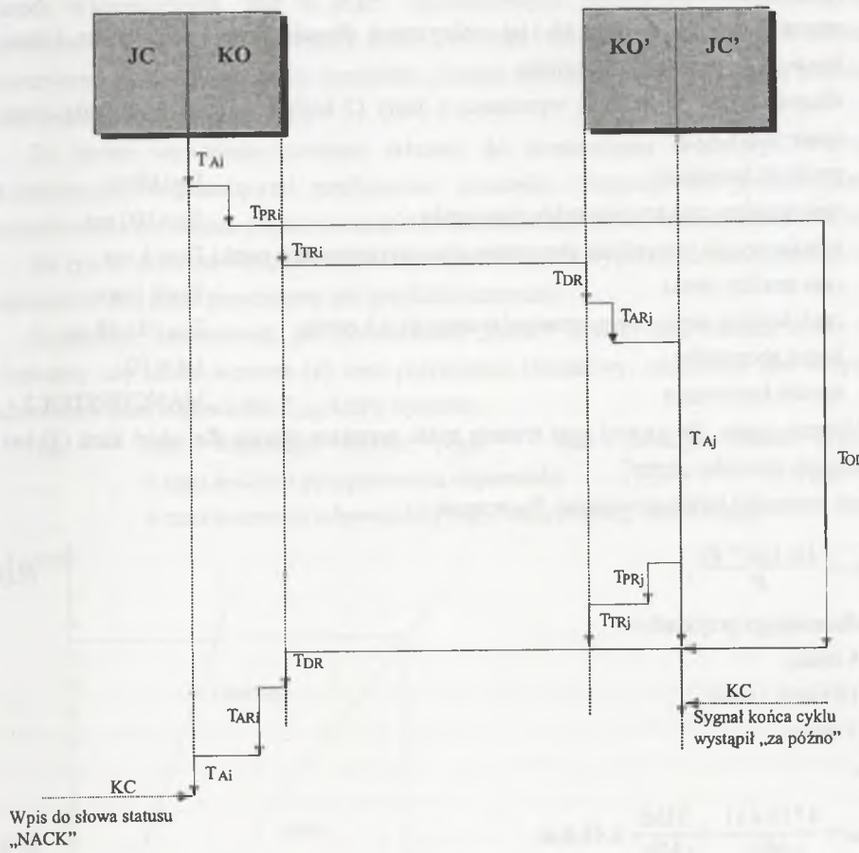
Rysunek 18 ilustruje przypadek złego doboru T_{ODj} przez, na przykład, nieuwzględnienie maksymalnego czasu trwania cyklu sterownika. Transakcja w takim przypadku kończy się wysłaniem przez odbiorcę ramki serwisowej typu „NACK". Można interpretować taką ramkę jako zajętość jednostki centralnej odbiorcy. Powróćmy jednak do określenia końcowego maksymalnego, spodziewanego cyklu sieci.

Biorąc pod uwagę trzy procesy:

- transmisję i cyrkulację żetonu,
- możliwość powstawania „dziur",
- transmisję danych użytkowych,

można zauważyć, że są one od siebie niezależne. Zatem spodziewany, maksymalny cykl sieci wynosi:

$$T_C = T_Z + T_D + T_{MAX} = T_{MZ} + T_{MAX} \quad (11)$$



Rys. 18. Przekroczenie czasu odpowiedzi
Fig. 18. The response time exceed

Podstawiając do wzoru (*11*) zależności (*6*) i (*10*) otrzymamy:

$$T_C = 3L_A(T_{TR} + T_{DR} + T_{AR} + T_{PR}) + \sum_{i=1}^{L_D} T_{ODi} + L_D(T_{PR} + T_{TR}) + \\ + 3\sum_{i=1}^{L_A} T_{Ai} + \sum_{i=1}^{L_A}(T_{PRi} + T_{TRi}) + L_A(2(T_{DR} + T_{AR}) + T_{TP}) \quad (12)$$

Na czas trwania cyklu składają się zatem dwa podstawowe, zawsze występujące składniki. Spróbujmy, przyjmując konkretne wartości, zobrazować, które czynniki i jak wpływają na czas trwania cyklu.

Przyjmijmy zatem:

- postać ramki jak na rys. 16 i jej maksymalną długość wraz z nagłówkiem i sumą kontrolną wynoszącą 256 bajtów,
- długość ramki serwisowej wynoszącą 4 bajty (2 bajty informacji + 2 bajty sumy kontrolnej CRC),
- prędkość transmisji V=1Mb/s,
- maksymalny czas trwania cyklu sterownika T_{AI}= 100 ms,
- jednakowy dla wszystkich abonentów czas przygotowania ramki T_{PR}= 1 ms,
- czas analizy ramki T_{AR} = 1 ms,
- czas detekcji ramki równy czasowi transmisji 2.5 znaku T_{DR} =33.38 μs,
- liczbę abonentów LA = 10,
- sposób kodowania MANCHESTER 2.

Obliczmy zatem, ile wynosi czas trwania cyklu wymiany żetonu dla takiej sieci (*2*) bez powstawania zjawiska „dziur".

Czas transmisji ramki serwisowej T_{TR} wynosi:

$$T_{TR} = \frac{L_{ZT} L_{BZ} + B_S}{V}, \qquad (13)$$

gdzie dla naszego przykładu:

L_{ZT} = 4 znaki,

L_{BZ} = 10 bitów / znak,

B_S = 11 bitów.

Wtedy:

$$T_{TR} = \frac{4*10+11}{1\,Mb/s} = \frac{51bit}{1\,Mb/s} \cong 48.6\mu s \qquad (14)$$

a

$$T_Z = 10*3\times(48.6\mu s + 33.38\mu s + 1ms + 1ms) = \\ = 30\times(81.98\mu s + 2ms) = 62.46ms \qquad (15)$$

Uzyskany wynik określa minimalną, teoretyczną wartość czasu trwania cyklu sieci, rzecz jasna nieosiągalną. To proste wyliczenie już na wstępie projektowania sieci odpowie na pytanie, czy będzie ona wystarczająco szybka, aby spełnić wymagania obiektu. Innymi słowy, jeśli konieczny będzie dostęp do jakiejś wielkości w czasie krótszym lub równym niż ta wyliczona, to z całą pewnością badana sieć będzie zbyt wolna. Kolejnym istotnym elementem są czasy T_{AR} i T_{PR} związane z szybkością przetwarzania koprocesorów. Widać wyraźnie, że to te dwa składniki mają największy wpływ na czas transmisji. Warto o tym pamiętać w chwili podejmowania decyzji o wyborze producenta elementów sieci. Nawet najbardziej imponująca prędkość transmisji nie zniweluje strat związanych z przetwarzaniem

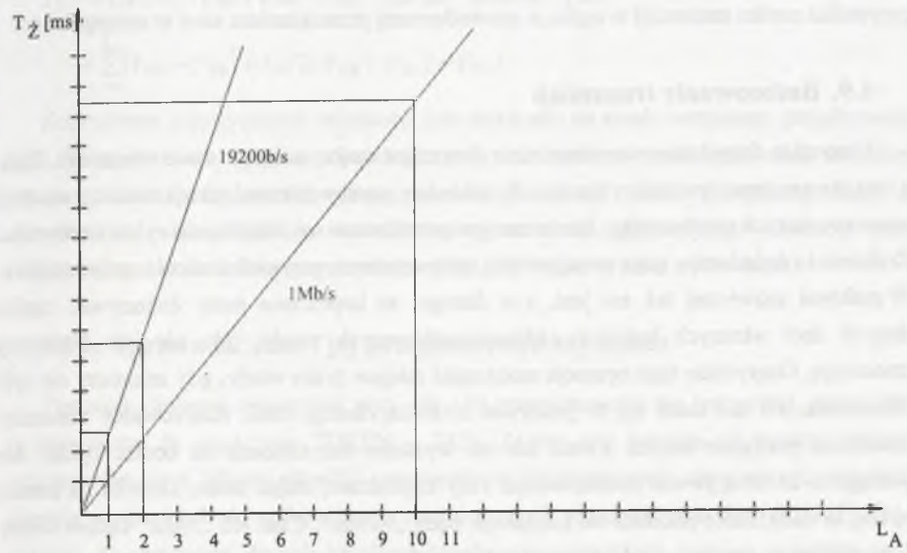
ramek w koprocesorze. Jest to jeden z podstawowych parametrów koprocesora, który powinien być znany projektantowi sieci. Cytowana wielkość 2 ms nie jest wielkością teoretyczną podaną na potrzeby przykładu. Została ona zaczerpnięta z katalogu producenta sieci przemysłowych [32].

To proste wyliczenie powinno skłaniać do niepopadania w zachwyt, w związku z istniejącymi, imponującymi prędkościami transmisji. Sugeruje ono ponadto kierunki poszukiwania rozwiązań, mających na celu zwiększenie szybkości wymiany danych w sieci.

Na rys. 19 przedstawiono zależność czasu trwania cyklu wymiany żetonu w funkcji liczby abonentów *LA*, gdzie parametrem jest prędkość transmisji.

Spróbujmy zobrazować, jak powstawanie „dziur" opóźni czas trwania cyklu sieci. Posłużmy się zatem wzorem (*4*) oraz przyjmijmy jednakowy, minimalny dla wszystkich abonentów czas odpowiedzi T_{ODi}, który wyniesie:

T_{OD} = czas transmisji żetonu (T_{TR}) + czas detekcji ramki (T_{DR}) + + czas analizy i przygotowania odpowiedzi (T_{AR} + T_{DR}) + + czas transmisji odpowiedzi (T_{TR}) + czas detekcji ramki (T_{DR}).



Rys. 19. Czas trwania cyklu wymiany żetonu
Fig. 19. The time of token exchange

Przyjmując, jak poprzednio, dane do obliczeń, uzyskamy:

$$T_{OD} \cong 2.164ms \qquad (16)$$

Zakładając, że w sieci powstaną trzy „dziury" i posługując się wzorem (*4*), uzyskamy:

$$T_D = 3*T_{OD} + 3\times(2ms + 48.6\mu s) \cong 6.5ms + 6.15ms = 13.65ms \qquad (17)$$

Gdybyśmy mieli do czynienia z redundancją sieci, czas ten byłby dwukrotnie większy i wyniósłby około 27. 3 ms, a to stanowi już ponad 43% czasu trwania cyklu wymiany żetonu dla sieci z 10 abonentami z poprzedniego przykładu.

Do ilustracji pozostał zatem ostatni element wpływający na czas trwania cyklu a mianowicie czas związany z transmisją danych. Oszacowanie czasu wymiany danych użytkowych należy do projektanta, co jest procesem żmudnym i pracochłonnym. Wyprowadzone zależności mogą posłużyć jako podstawa wyliczenia cyklu sieci. Należy jednak brać pod uwagę specyfikę różnych sieci. Aby przeprowadzić dokładną analizę czasową sieci z transmisją danych, zajęłoby to znaczną część niniejszego opracowania. W tym miejscu wydaje się, że wystarczy stwierdzić, iż jest to możliwe. Opierając się na wyprowadzonych zależnościach, dokonano w pracy pełnej analizy czasowej systemu sterowania instalacją Stacji Przygotowania Wody dla Elektrowni „TRZEBOVICE" (Republika Czeska), na podstawie której sieć, składająca się z siedmiu abonentów i transmitująca stany 4000 wejść/wyjść binarnych i kilkudziesięciu pomiarów analogowych, została prawidłowo skonfigurowana i pracuje nieprzerwanie od 1993 roku bez żadnego przypadku zaniku transmisji w ogóle, a spowodowanej przeciążeniem sieci w szczególności.

4.9. Buforowanie transmisji

Wszystkie dotychczasowe rozważania dotyczące maksymalnego czasu transmisji T_{MAX}, a tym samym czasu trwania cyklu sieci T_C zakładały, że aby dokonać jakiejkolwiek transakcji wymiany danych użytkownika, konieczne jest oczekiwanie na zakończenie cyklu sterownika. Zrobiono to świadomie, gdyż w najbardziej niekorzystnym przypadku może to mieć miejsce. W praktyce zazwyczaj tak nie jest, a to dlatego, że koprocesor może dokonywać zapisu danych do/z własnych buforów nadawczo-odbiorczych wtedy, gdy nie jest obciążony transmisją. Oczywiście taka operacja może mieć miejsce tylko wtedy, gdy zakończy się cykl sterownika, ale nie musi się to pokrywać z chwilą obsługi sieci. Buforowanie transmisji umożliwia przyjęcie danych z sieci lub ich wysłanie bez czekania na koniec cyklu. Ale pociąga to za sobą pewne konsekwencje. Gdy koprocesor, mając żeton, zaczeka na koniec cyklu, to dane, które pobierze do transmisji, będą „świeże". Czas ich „życia" będzie równy w przybliżeniu czasowi cyklu automatu. Jeśli pobierze je w „wolnej chwili" poza obsługą sieci, to dane te mogą być „starsze" o czas trwania jednego cyklu sieci. Dotyczy to wymian wyzwalanych. Nie ma to natomiast znaczenia przy odbiorze danych. Mogą być to dane tak czy inaczej „starsze" o jeden cykl sterownika. Przy odbiorze natomiast jest istotne uzyskanie potwierdzenia od macierzystej jednostki centralnej poprawności bądź nieprzeprowadzenia transakcji zapisu do pamięci. Koprocesor odbiorcy może odebrać ramkę i wysłać potwierdzenie jej przyjęcia, ale nie czekając na koniec cyklu sterownika, nadawca nie uzyska pewności, że dane te zostały poprawnie zapisane w pamięci odbiorcy. Generalnie rzecz

biorąc, buforowanie może skrócić czas T_{MAX}, ale zwiększanie liczby buforów nadawczo – odbiorczych w ostatecznym rezultacie nie musi poprawić pracy sieci. Zwiększenie bowiem liczby buforów komplikuje oprogramowanie koprocesora i wydłuża parametry T_{AR} i T_{PR}, które, jak wykazano, decydują znacznie o czasie wymiany. Wzrost liczby buforów komplikuje również proces sterowania przepływem danych na styku „koprocesor – jednostka centralna". Ponadto zła parametryzacja sieci lub „wyśrubowanie" jej parametrów i tak może doprowadzić do zapełnienia wszystkich buforów, co skończy się wysłaniem do nadawcy sygnału NACK, a do macierzystej jednostki centralnej sygnału nasycenia buforów nadawczych. Jednak można przy analizie pracy sieci oprócz wzoru (*10*) posłużyć się jego uproszczoną postacią

$$T_{MAX} = \sum_{i=1}^{L_A} (T_{PRi} + T_{TRi}) + L_A(2(T_{DR} + T_{AR}) + T_{TP}), \tag{18}$$

co w konsekwencji uprości wzór (*12*) do następującej postaci:

$$T_C = 3L_A(T_{TR} + T_{DR} + T_{AR} + T_{PR}) + \sum_{i=1}^{L_D} T_{ODi} + L_D(T_{PR} + T_{TR}) + \\ + \sum_{i=1}^{L_A} (T_{PRi} + T_{TRi}) + L_A(2(T_{DR} + T_{AR}) + T_{TP}) \tag{19}$$

Korzystanie z powyższych zależności jest wskazane na etapie wstępnego projektowania sieci. Da to bowiem wstępną odpowiedź na pytanie, jakiego rzędu będą czasy transmisji i może pomóc w szukaniu poprawy parametrów pracy sieci, polegającej bądź na rekonfiguracji sieci, bądź na zmianie oprogramowania aplikacyjnego abonenta sieci, zmierzającego do skrócenia czasu trwania cyklu, lub wręcz wymusi wybór innej.

4.10. Sprawność sieci i jej przepustowość użyteczna

Definicja zarówno sprawności sieci, jak i jej przepustowości nie jest prosta, szczególnie w odniesieniu do sieci typu TOKEN – BUS. Zależy ona bowiem od punktu widzenia użytkownika sieci. Można określić sprawność sieci jako stosunek czasu trwania transmisji danych użytkowych do całkowitego czasu transakcji wymiany tych danych [112]. Jeżeli przyjąć, że transmisja danych użytkownika dotyczy wszystkich wymian w jednym cyklu sieci, to całkowity czas ich wymiany określa wzór (*12*) lub w prostszej postaci wzór (*19*). Tak obliczona sprawność będzie dużo niższa niż ta liczona według [112], choć wydaje się być najbardziej obiektywna. Bo przecież dla użytkownika najważniejsza jest informacja, jak szybko realizowany jest cykl wymiany wszystkich informacji w sieci, a nie czas pojedynczej wymiany czy nawet czas trwania pojedynczego cyklu sieci. Ponieważ w sieciach TOKEN – BUS problem cyklu sieci jest dość złożony, co wynika i z liczby wymian i ze zmiennej

długości ramek oraz aby znaleźć platformę porównania z innymi typami sieci, proponuje się przyjęcie następujących definicji sprawności i przepustowości użytecznej sieci.

Niech sprawność wynosi

$$\eta = \frac{\text{czas transmisji danych użytkowych w pojedynczej transakcji wymiany}}{\text{całkowity czas pojedynczej transakcji wymiany}} *100\ [\%]$$

a przepustowość użyteczna

$$P = \frac{\text{liczba danych użytkowych w pojedynczej transakcji wymiany}}{\text{całkowity czas pojedynczej transakcji wymiany}}\ [kb/s].$$

Określmy zatem czas transmisji danych użytkowych i całkowity czas pojedynczej transakcji danych. Zakładając, że w ramce danych transmitowanych jest „n" bajtów, a całkowity czas transakcji jest sumą transmisji danych i transmisji żetonu, otrzymamy:

$$\eta = \frac{\frac{8n}{V}}{T_{\dot{z}i,j} + T_{MAXi}} *100\ [\%], \qquad (20)$$

$$P = \frac{8n}{T_{\dot{z}i,j} + T_{MAXi}}\ [kb/s], \qquad (21)$$

gdzie: V – jest prędkością transmisji wyrażoną w b/s,

$T_{\dot{z}ij}$ – jest czasem transmisji żetonu pomiędzy „i" – tym a „j" – tym abonentem,

T_{MAXi} – jest maksymalnym czasem transmisji ramki o największej długości przy założeniu buforowania bez konieczności na oczekiwanie końca cyklu automatu.

Zatem:

$$\eta = \frac{\frac{8n}{V}}{\left(3\left(T_{TR} + T_{DR} + T_{AR} + T_{PR}\right)\right) + \left(T_{PRi} + T_{TRi} + 2\left(T_{DR} + T_{AR}\right) + T_{TP}\right)} *100\ [\%] \qquad (22)$$

$$P = \frac{8n}{\left(3\left(T_{TR} + T_{DR} + T_{AR} + T_{PR}\right)\right) + \left(T_{PRi} + T_{TRi} + 2\left(T_{DR} + T_{AR}\right) + T_{TP}\right)}\ [kb/s] \qquad (23)$$

Wielkości występujące w obu wzorach (*22*, *23*) zostały określone w rozdziale 4.6. Przyjmując dane takie jak w przykładzie z rozdziału 4.8, obliczona wartość sprawności osiąga 11%, a przepustowości około 111 k b/s, przy długości ramki równej 256 bajtów.

W obu wzorach uwzględniono pełny czas transakcji, a więc zarówno transmisję żetonu, jak i transmisję potwierdzenia po odbiorze ramki danych. Spotyka się również definicje, które nie uwzględniają transmisji żetonu i potwierdzenia, ograniczając się jedynie do czystego czasu transmisji ramki danych, nie uwzględniając również czasu detekcji i analizy ramki. Tak wyliczona sprawność jest jedynie miarą narzutu czasu wnoszonego przez warstwę liniową sieci. Porównajmy zatem wartości sprawności w obu przypadkach. Nie uwzględniając, jak już wspomniano, ani transmisji żetonu $T_{\dot{z}}$, ani ramki serwisowej potwierdzenia (T_{TP}) i pomijając czasy detekcji i analizy ramki uzyskamy (patrz. rozdział 4.6, 4.8, wzór (*3*)):

$$\eta = \frac{\frac{8n}{V}}{\frac{L_{\dot{Z}T} L_{BZ} + B_S}{V}} = \frac{8n}{L_{\dot{Z}T} L_{BZ} + B_S} *100\ [\%] \qquad (24)$$

$$P = \frac{8n}{\frac{L_{\dot{Z}T} L_{BZ} + B_S}{V}}\ [kb/s]. \qquad (25)$$

Zakładając, jak poprzednio, postać ramki danych (rys. 16) uzyskamy:

$$\eta = \frac{8n}{(n+7)10+11} *100\ [\%], \qquad (26)$$

$$P = \frac{8nV}{(n+7)10+11}\ [kb/s]. \qquad (27)$$

Dla tak przyjętych zależności sprawność waha się od 8.8% do 77.5% (w zależności od przyjętej definicji sprawności), a przepustowość dla ramki o długości 256 bajtów użytkowych osiąga wartość 775 kb/s.

Opisane dwa sposoby obliczania sprawności i przepustowości ilustrują, jak wielkie mogą wystąpić różnice, zależnie od podejścia do problemu. Zwracają również uwagę na to, jakimi informacjami producenta należy dysponować, aby oba parametry wyliczyć. Ponadto nakazują dużą ostrożność w interpretowaniu parametrów sieci, które to parametry można spotkać w różnego rodzaju literaturze bądź materiałach reklamowych.

4.11. Poprawa parametrów czasowych wymian

Z rozważań do tej pory przeprowadzonych wynika, że najistotniejszym problemem przy konfigurowaniu sieci jest określenie jej maksymalnego cyklu. Parametr ten ma decydujące znaczenie dla określenia czasów nadawania każdego z abonentów. Dzięki obliczeniu cyklu sieci precyzyjnej można określić czasy odpowiedzi T_{OD} poszczególnych abonentów. W niektórych rozwiązaniach jest to wielkość parametryzowana. Ponadto, jak już wspomniano, maksymalny czas trwania cyklu sieci decyduje o podjęciu decyzji co do zakresu jej stosowalności z punktu widzenia parametrów czasowych procesu technologicznego. Może wtedy zapaść decyzja albo o zmianie typu sieci, albo podjęta zostanie próba poprawy parametrów czasowych wymian. Polepszenie parametrów można uzyskać stosując wiele zabiegów, które mogą polegać na:

– zminimalizowaniu liczby wymian oraz wielkości porcji danych transmitowanych,

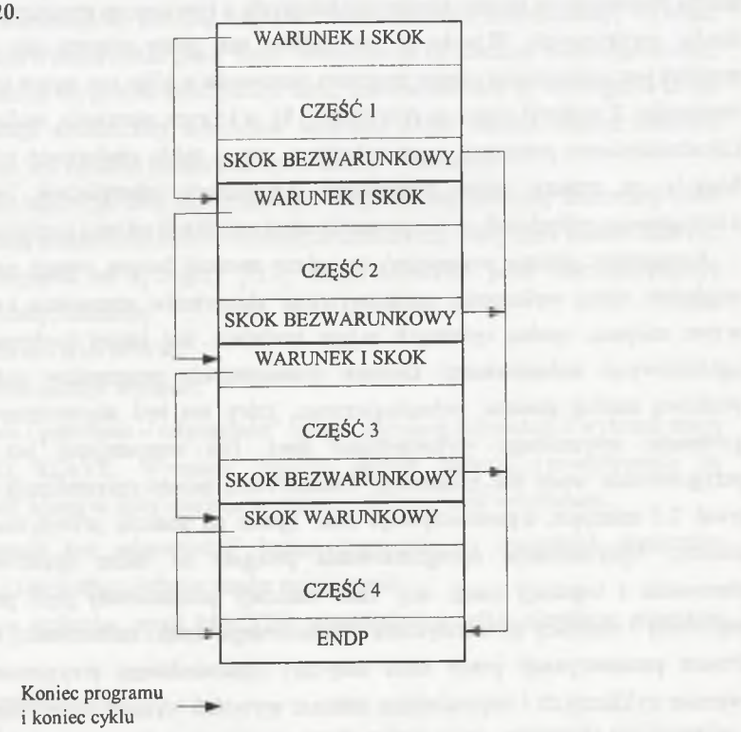
- zmniejszeniu liczby abonentów występujących w jednym segmencie sieci przez utworzenie kilku segmentów ze sobą połączonych, o tym samym priorytecie,
- zmianie priorytetów transmitowanych komunikatów, co związane jest z przestrzeganiem kilku podstawowych reguł:
  - najpierw powinien być transmitowany komunikat najpilniejszy z listy komunikatów superpilnych (o tym, który z komunikatów jest pilny, decydują wymagania technologiczne),
  - w następnej kolejności należy emitować komunikaty pilne, a jako ostatnie transmitowane winne być tak zwane komunikaty zwykłe.

Poza wymienionymi czynnikami, które wpływają na optymalizację wymian, istnieje jeszcze cały szereg innych, które mają wpływ nie tylko na czas trwania cyklu sieci, ale również na cykl wymiany całej informacji, która siecią jest przesyłana. Poprawne określenie cyklu sieci daje gwarancję, że każdy abonent otrzyma, w żądanym czasie, prawo do nadawania, ale nie oznacza to bynajmniej, że co cykl sieci dowolny abonent wykona w całości zadanie związane z przesyłem danych. Zależy to właściwie od kilku trudno przewidywalnych zjawisk, co wcale nie oznacza, że należy je pominąć. Pierwszym z takich zjawisk jest możliwość lawinowego pojawiania się różnych sygnałów pochodzących z obiektu sterowania, co zazwyczaj wywołuje lawinę wymian wyzwalanych. Zjawisko to powoduje gwałtowne obciążenie sieci, gdyż rośnie liczba wymian i liczba informacji przeznaczonych do transmisji. Była już o tym mowa, ale powtórzmy, że koniecznie należy dokonać hierarchizacji ważności transmitowanych danych. Zwykle, przy stabilnej pracy obiektu, a tym samym stabilnej pracy sieci, przeważają wymiany cykliczne. Liczba danych transmitowanych jest łatwa do oszacowania, a tym samym wyliczenie cyklu sieci nie jest kłopotliwe. Dopiero lawinowe pojawienie się zdarzeń i w konsekwencji zachowanie się sieci spowoduje weryfikację poprawności jej konfiguracji. Zatem projektant sieci, poza innymi informacjami, musi znać maksymalną liczbę wymian wyzwalanych. Informacje te musi przekazać autor oprogramowania sterownika.

Dochodzimy do kolejnego, bardzo ważnego elementu cyklu wymiany, jakim jest cykl pracy sterownika, realizującego algorytmy sterowania, regulacji i monitorowania, rezydujące w postaci programu, w jednostce centralnej stacji abonenckiej. Skrócenie czasu trwania cyklu programu sterownika jest, poza hierarchizacją komunikatów ze względu na ich ważność i doborem odpowiednich odstępów czasu pomiędzy kolejnymi wymianami sieciowymi, rzeczą najważniejszą. Skrócenie cyklu automatu w prostej linii oznacza skrócenie czasu wykonywania programu. Jego długość, a zatem czas realizacji pętli, wpływa na częstość dostępu do pamięci jednostki centralnej przez koprocesor, a tym samym wpływa na cykl wymiany całej informacji.

Innym istotnym elementem programu mogą być występujące w nim uwarunkowania, co może powodować, że program główny musi czekać na spełnienie określonego warunku, aby dane do transmisji były ważne. Ma to szczególne znaczenie, gdy wykorzystywane są wieloprocesorowe jednostki centralne, pracujące w trybie pracy równoległej. Dzięki odpowiedniej, właściwej synchronizacji zadań współbieżnych nie wystąpi zjawisko zbędnego oczekiwania na zakończenie pojedynczego cyklu. Zahaczamy w tym momencie o problematykę równoległej pracy jednostek wieloprocesorowych, co jest problemem samym w sobie, wykraczającym poza zakres niniejszej pracy.

Skrócenie czasu trwania cyklu automatu można nieraz osiągnąć przez podzielenie programu na części tak, aby koprocesor co kilkanaście milisekund miał dostęp do pamięci jednostki centralnej. Sposób ten polega na realizacji całego programu w kilku cyklach. Ilustruje to rys. 20.



Rys. 20. Podział programu sterownika na części
Fig. 20. The partition of PLC program

W każdym cyklu pracy sterownika (automatu) będzie realizowana tylko jedna część programu. Na początku każdej z nich umieszczony jest skok warunkowy decydujący o tym, czy dana część programu ma być realizowana. Jeżeli dana część była już realizowana

w danym makrocyklu, to wykona się rozkaz skoku do początku następnej części programu. W przeciwnym wypadku skoku się nie wykona i dana część programu będzie realizowana. Na końcu każdej części programu umieszczony jest skok do końca programu Na rys. 20 widać, że w tym przykładzie koprocesor sieci w czasie realizacji całego programu w sterowniku będzie miał czterokrotny dostęp do pamięci jednostki centralnej. Rozwiązanie to ma pewne wady. Bardzo rozważnie należy bowiem podzielić program sterowania procesem na etapy. W tak podzielnym programie utrudnione jest wykrywanie szybkich zmian stanów wejściowych, a także tak zwane różniczkowanie sygnałów, polegające na wykrywaniu zboczy sygnałów wejściowych. Dodatkową wadą jest to, że realizowane układy czasowe mogą działać niedokładnie, co w wielu przypadkach jest niedopuszczalne. Trzeba również pamiętać o tym, że taki sposób realizacji programu spowoduje zwolnienie reakcji układu sterowania na zmiany stanów wejściowych, a tym samym zmniejszy dynamikę zmian stanów wyjściowych. W praktyce, szczególnie tam gdzie procesy nie są. zbyt szybkie, możliwe jest realizowanie całego programu sterowania w kilku czy nawet kilkunastu cyklach sterownika. Z praktyki znane są przykłady [18], w których sterownik, realizujący sterowanie kilkudziesięcioma pompami, musi w każdym swym cyklu obsługiwać tylko kilka z nich. Reakcja na zmiany wejść związanych z sygnałami zabezpieczeń będzie wolniejsza o kilkadziesiąt milisekund, co nie powoduje dyskwalifikacji takiego rozwiązania.

Reasumując, można powiedzieć, że należy zwracać baczną uwagę na optymalne, pod względem czasu wykonania, opracowywanie algorytmów sterowania i regulacji. Trudno w tym miejscu, oprócz opisanych w tym rozdziale, dać jakieś konkretne recepty, poza ogólnikowymi wskazówkami. Dopiero doświadczenie programisty połączone z bardzo wnikliwą analizą procesu technologicznego, który ma być algorytmizowany, może dać gwarancję optymalnego wykorzystania sieci. Dla wspomnianej już instalacji stacji przygotowania wody dla Elektrowni TREBOVICE proces optymalizacji oprogramowania trwał 2.5 miesiąca, a parametryzacja sieci oparta na analizie przepływu danych kolejny miesiąc. Optymalizacja oprogramowania polegała na takim opracowaniu algorytmu sterowania i regulacji stacji, aby czas realizacji podstawowej pętli programu był jak najkrótszy i mniejszy od teoretycznie wyznaczonego, dzięki zastosowanej metodzie analizy. Proces parametryzacji pracy sieci dotyczył odpowiedniego przygotowania scenariusza wymian cyklicznych i odpowiednim doborze wywołań wymian wyzwalanych. Cały proces uruchamiania oprogramowania został zakończony dwumiesięcznym testowaniem w warunkach laboratoryjnych.

5. SIECI O DOSTĘPIE MASTER–SLAVE

Innym sposobem zapewnienia w sieciach przemysłowych zdeterminowanego w czasie dostępu do medium transmisyjnego (łącza) dowolnemu abonentowi jest metoda MASTER – SLAVE. Protokół ten jest znacznie prostszy od protokołu TOKEN – BUS, co przejawia się między innymi tym, że programista nie ma tak wiele swobody w doborze różnych sposobów transmisji. Reguły transmisji są bardziej sztywne i uniemożliwiają optymalizację wymian, celem skrócenia czasu trwania cyklu pracy sieci. W zamian za to, kosztem wielu ograniczeń, zdecydowanie upraszcza się proces konfiguracji sieci, minimalizowane są wymagania co do oprogramowania stacji abonenckiej z poziomu aplikacji, a tym samym analiza czasowa przepływu informacji jest bardziej przejrzysta i prostsza do wykonania.

Zasadniczą cechą tego typu sieci jest możliwość bezpośredniej wymiany informacji tylko pomiędzy wyróżnioną stacją MASTER a dowolnym abonentem, który nosi miano SLAVE, choć znane są odstępstwa od tej reguły [119]. Stacja MASTER pełni rolę dystrybutora wymian i inicjuje każdą transmisję.

Innymi wyróżnikami tych sieci są:

- dwa podstawowe rodzaje wymian:
  - **„zapytanie / polecenie – odpowiedź"** będąca wymianą informacji z wybraną stacją abonencką SLAVE. Wymiana zawiera zawsze żądanie i towarzyszącą jej odpowiedź. Mogą tu mieć miejsce wymiany cykliczne bądź wyzwalane,
  - **„rozgłoszenie bez odpowiedzi"** będąca transmisją do wszystkich abonentów SLAVE, i zawierającą jedynie ramkę rozgłoszenia,
- stały repertuar rozkazów stacji MASTER umożliwiający tylko określone transakcje wymiany,
- brak ramek serwisowych,
- niezmienne, bez zatrzymania i ponownego uruchomienia sieci, sekwencje cyklicznych wymian danych.

Uproszczenie protokołu transmisji ma jeszcze inną ważną zaletę. Przyspiesza znacznie proces uruchamiania systemu komunikacyjnego, co nie jest bez znaczenia.

W odróżnieniu od sieci o dostępie „TOKEN – BUS" w sieci MASTER–SLAVE programuje się jedynie koprocesor stacji MASTER. Cały ciężar konfiguracji sieci przeniesiony jest na projekt tak zwanego scenariusza wymian, który umieszczony jest w pa-

mięci koprocesora stacji MASTER. Kapitalnie upraszcza to analizę pracy sieci. Koprocesory stacji SLAVE nie są zwykle programowane, a realizują jedynie przesłane ze stacji MASTER **polecenia – żądania**, dotyczące wymian na styku „koprocesor – jednostka centralna". Jest to realizowane na poziomie sprzętowym [119].

5.1. Przygotowanie scenariusza wymian i opis wymiany informacji

Przyjmijmy, że w omawianym protokole wyróżnia się dwa podstawowe typy ramek:
- „zapytanie / polecenie" – transmisja MASTER–SLAVE,
- „odpowiedź" – transmisja SLAVE–MASTER.

Mamy tu do czynienia również z ramką **„rozgłoszenia"**, ale jej budowa jest identyczna jak pozostałych (występują te same pola informacyjne).

Przed przystąpieniem do opisu tworzenia scenariusza wymian należy opisać postać ramek i zdefiniować funkcje realizowane przez stacje SLAVE.

Podstawową strukturę ramki przedstawia rys. 21.

Rys. 21. Budowa ramki
Fig. 21. The frame form

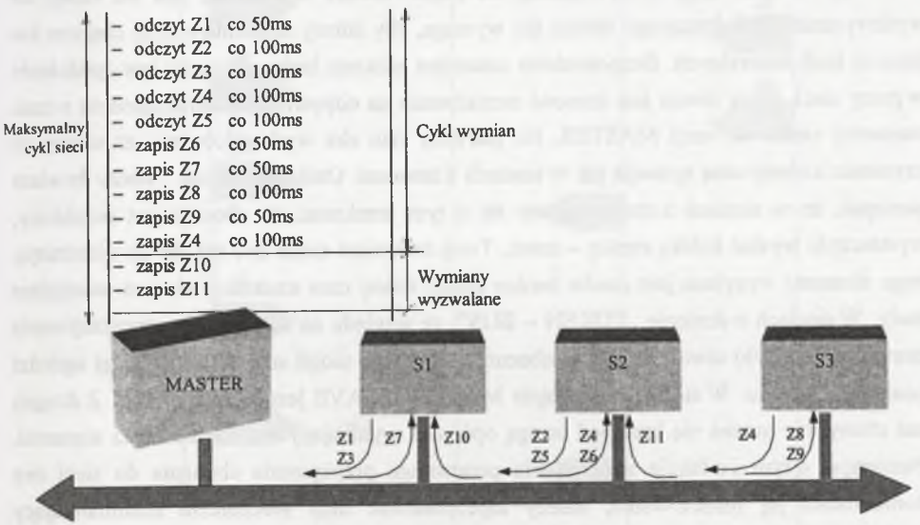
W polu „DANE" mogą być, w zależności od kodu realizowanej funkcji, zawarte dodatkowe informacje, takie jak:
- kod funkcji dodatkowej (są to dodatkowe funkcje, takie jak na przykład: zatrzymanie pracy jednostki centralnej abonenta, przesłanie programu do jednostki centralnej abonenta, zdalny restart),
- adres danych do odczytu lub zapisu,
- długość łańcucha danych.

Postać poszczególnych pól zależeć będzie oczywiście od konkretnego rozwiązania firmowego, co w żadnym stopniu nie ogranicza, jak sądzę, dalszych rozważań. Przedstawiona na rys. 21 ramka jest typowa zarówno dla transferu MASTER–SLAVE, jak i SLAVE – MASTER. Dwa pola w ramce odpowiedzi stacji SLAVE są takie same jak w ramce „żądania". Są to: „Adres Abonenta" oraz „Kod Funkcji".

Dzięki temu stacja MASTER może dokonać wstępnej weryfikacji poprawności transmisji. Pole danych ma zmienną długość i w zależności od rozwiązań firmowych może mieć od kilkudziesięciu do kilkuset bajtów. Na potrzeby niniejszej pracy określmy funkcje, które mogą być realizowane przez stację SLAVE. Niech tymi funkcjami będą:
- żądanie odczytu ze stacji SLAVE „n" bitów,

- żądanie odczytu ze stacji SLAVE „n" słów,
- żądanie zapisu do stacji SLAVE jednego bitu,
- żądanie zapisu do stacji SLAVE jednego słowa,
- żądanie „szybkiego" odczytu ze stacji SLAVE, niezsynchronizowanego z cyklem jednostki centralnej stacji SLAVE, wartości jednego bajtu,
- żądanie zapisu „n" bitów do stacji SLAVE,
- żądanie zapisu „n" słów do stacji SLAVE,
- odczyt ze stacji SLAVE zawartości licznika wymian,
- odczyt / zerowanie w stacji SLAVE rejestrów diagnostycznych.

Mając zdefiniowane funkcje, można przystąpić do konstrukcji scenariusza wymian. Pojęcie „scenariusz" zostało wprowadzone przez autora niniejszej pracy, natomiast sposób jego tworzenia jest dowolny i w dużej mierze zależy od indywidualnych upodobań konfigurujących sieć. Scenariusz wymian to nic innego jak lista zawierająca nazwy zmiennych, których wartości mają być transmitowane. Zawiera ona, oprócz nazw zmiennych, również kierunek transmisji i co jest bardzo ważne proponowany zazwyczaj przez technologię czas ich „odświeżania". Umieszcza się na niej również planowane wymiany wyzwalane. Stanowi to pierwszy krok do konfiguracji sieci i poprzedza proces analizy czasowej, której rezultaty zazwyczaj spowodują modyfikację scenariusza wymian. Ilustruje to rys. 22. Stacja MASTER żąda transmisji wartości zmiennych Z1, Z2, Z3, Z4, Z5, a sama wysyła wartości zmiennych Z6, Z7, Z8, Z9. Dodatkowo stacja SLAVE **2** żąda transmisji wartości zmiennej Z4. Są to żądania transmisji cyklicznej.

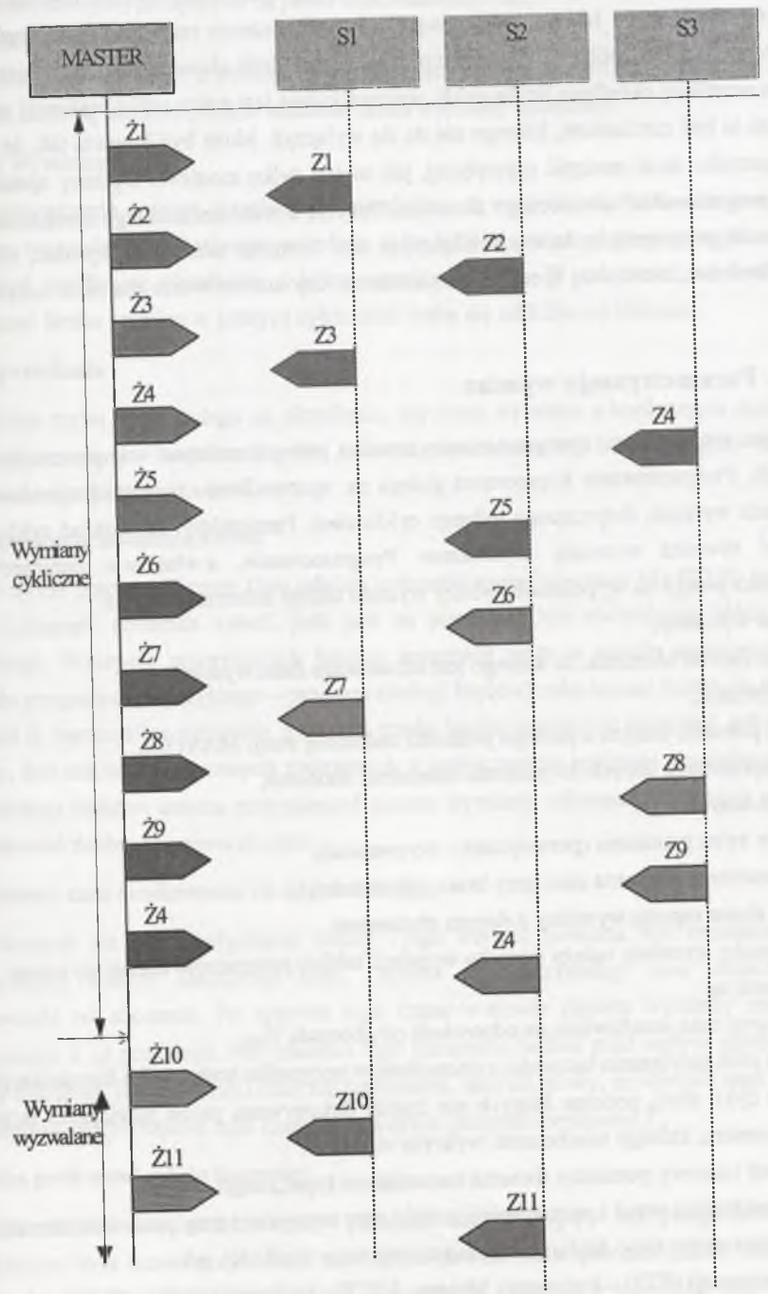


Rys. 22. Scenariusz wymian
Fig. 22. The exchanges scenario

Ponadto wprowadzono do scenariusza wymian dwie wymiany wyzwalane Z10 i Z11, których realizacja będzie inicjowana przez program aplikacyjny jednostki centralnej stacji MASTER. Widać wyraźnie znaczne „usztywnienie" możliwości transmisji w porównaniu z siecią o dostępie „TOKEN – BUS", ale, jak już wspomniano, ułatwia to znacznie analizę czasową. W sieci o dostępie „TOKEN – BUS" bowiem nie planuje się na poziomie koprocesora wymian wyzwalanych. Jakkolwiek w obu typach sieci są one inicjowane przez jednostki centralne, to tylko w sieci MASTER-SLAVE konieczne jest ich zaprogramowanie (wprowadzenie do scenariusza wymian) w koprocesorze. Wszystkie bowiem żądania transmisji muszą być zaplanowane, a jedynie ich liczba musi być ograniczona ze względu na maksymalny czas wymiany danych, a tym samym ze względu na gwarantowany czas dostępu do medium. Innymi słowy, w sieci MASTER-SLAVE nie można zrealizować wymiany nie zaplanowanej. Na rys. 23 przedstawiono sposób wymiany informacji dla przyjętego scenariusza wymian.

5.2. Powstawanie „dziur" w sieci

Podobnie jak w omawianej już sieci o dostępie „TOKEN – BUS" tu również może nastąpić zjawisko powstawania „dziur". Przyczyną tego zjawiska jest tylko faktyczny brak abonenta lub jego uszkodzenie. Nie powstaną „dziury" wskutek błędnej adresacji, jak to miało miejsce poprzednio. Każda bowiem wymiana umieszczona w scenariuszu jest adresowana do konkretnego abonenta. Oczywiście można zrobić błąd planując wymianę do nieistniejącego abonenta, ale analizując słowo raportu, na etapie choćby testowania, jest on łatwy do wychwycenia. Brak krążącego żetonu nie wymaga, aby adresy abonentów były ciągiem kolejnych liczb naturalnych. Bezpośrednim natomiast efektem braku abonenta jest opóźnienie w pracy sieci, które równe jest czasowi oczekiwania na odpowiedź, powiększonemu o czas transmisji ramki od stacji MASTER. Na pierwszy rzut oka wydawałoby się, że mamy do czynienia z identyczną sytuacją jak w sieciach z żetonem. Otóż tak nie jest. Należy bowiem pamiętać, że w sieciach z żetonem, aby się o tym przekonać, że abonent jest nieobecny, wystarczyło wysłać krótką ramkę – żeton. Tutaj natomiast może być tak, że do nieistniejącego abonenta wysyłana jest ramka bardzo długa, której czas transmisji nie jest pomijalnie mały. W sieciach o dostępie „TOKEN – BUS", ze względu na sekwencyjne przekazywanie żetonu, można było utworzyć listę nieobecności, na której mogli znaleźć się najbliżsi sąsiedzi posiadacza żetonu. W sieciach o dostępie MASTER–SLAVE jest to niemożliwe. Z drugiej zaś strony, nie można nie brać pod uwagę opóźnień wynikających z nieobecności abonenta. Pamiętając o tym i o fakcie zapewnienia ponownego przyłączenia abonenta do sieci bez konieczności jej restartowania, należy zaproponować inny mechanizm minimalizujący opóźnienia będące rezultatem powstawania „dziur". Jedną z propozycji, które spotyka się



Rys. 23. Realizacja scenariusza wymian
Fig. 23. The execution of exchanges scenario

w rozwiązaniach firmowych [119], jest określenie liczby cykli sieci, w czasie trwania których nieobecny abonent nie będzie „odpytywany". Po zakończeniu realizacji określonej liczby cykli ponawiana jest próba nawiązania z nim łączności. Jeśli abonent jest ciągle nieobecny, znów na wcześniej określoną liczbę cykli, wstrzymywana jest z nim próba realizacji wymian. Nie może to być mechanizm, którego nie da się wyłączyć. Może być bowiem tak, że nawiązanie łączności musi nastąpić najszybciej, jak to jest tylko możliwe. Opisany sposób nosi nazwę „autousuwania" nieobecnego abonenta. Decyzję o uruchamianiu tego mechanizmu pozostawia się programiście, który, konfigurując sieć i tworząc scenariusz wymian, decyduje przez określenie, zazwyczaj specjalnego parametru, czy autousuwanie ma mieć miejsce czy nie.

5.3. Parametryzacja wymian

Jak już wspomniano, oprogramowanie wymian jest umieszczone w koprocesorze stacji MASTER. Programowanie koprocesora polega na wprowadzeniu wcześniej opracowanego scenariusza wymian, dotyczącego jednego cyklu sieci. Pamiętajmy, że w skład cyklu mogą wchodzić również wymiany wyzwalane. Programowanie, a właściwie parametryzacja koprocesora polega na wypełnieniu tablicy wymian takimi informacjami, jak:

- numer wymiany,
- numer (adres) abonenta, do którego jest adresowana dana wymiana,
- kod operacji,
- adres pobrania danych z pamięci jednostki centralnej stacji MASTER,
- adres przesłania danych do jednostki centralnej abonenta,
- liczba danych,
- wybór trybu przesłania (periodyczna – wyzwalana),
- autousuwanie abonenta sieci przy braku odpowiedzi,
- adres słowa raportu wymiany z danym abonentem.

Dla każdej wymiany należy ponadto wypełnić tablicę parametrów każdej wymiany. Tymi parametrami są:

- graniczny czas oczekiwania na odpowiedź od abonenta T_{OD},
- liczba prób nawiązania łączności z abonentem w przypadku braku z nim komunikacji,
- liczba cykli sieci, podczas których nie będzie dokonywana próba nawiązania łączności z abonentem, którego nieobecność wykryto w sieci,
- interwał czasowy pomiędzy dwiema transmisjami typu „rozgłoszenie",
- czas opóźnienia przed i po transmisji ramki i czas autoryzacji przy pracy z modemami.

Globalnie oczywiście deklaruje się dodatkowo takie wielkości, jak:

- tryb transmisji (RTU – kodowanie binarne, ASCII – kodowanie znakowe),
- liczba bitów stopu i parzystość,

- liczba bitów przypadających na jeden znak transmisji L_{BZ},
- prędkość transmisji V.

Znaczenie większości z wymienionych parametrów jest oczywiste. Omówione zostaną tylko te, które są istotne z punktu widzenia czasu wymiany informacji.

Numer wymiany

Wprowadzenie numeru wymiany jest niezbędne ze względu na możliwość generowania wymian wyzwalanych. Ponieważ wymiany te są inicjowane przez program aplikacyjny, to musi być możliwość określenia, o którą wymianę chodzi. W zależności od konkretnych rozwiązań liczba wymian w jednym cyklu sieci waha się od kilku do kilkuset.

Tryb przesłania

Wybór trybu pracy polega na określeniu, czy dana wymiana o konkretnym numerze ma być realizowana bez inicjalizacji przez jednostkę centralną stacji MASTER, czy też ma się odbywać w każdym cyklu sieci.

Autousuwanie abonenta z sieci

W trybie pracy cyklicznej (bez udziału jednostki centralnej stacji MASTER) koprocesor może „usunąć" abonenta z sieci, jeśli jest on nieobecny lub stwierdzono błędy podczas transmisji. Wykrycie notorycznych błędów transmisji może w sposób automatyczny (bez udziału programu aplikacyjnego – program obsługi błędów) zablokować dostęp do abonenta.

Jest to bardzo silne narzędzie, z którego trzeba bardzo umiejętnie korzystać, gdyż z jednej strony, bez nakładów czasowych związanych z koniecznością realizacji specjalnego programu obsługi błędów, można przyspieszyć proces wymiany informacji, z drugiej zaś można zablokować dostęp abonenta do sieci.

Graniczny czas oczekiwania na odpowiedź T_{OD}

Parametr ten jest niesłychanie ważny i jego wartość powinna być rezultatem bardzo precyzyjnej analizy obciążenia sieci. Określa on maksymalny czas oczekiwania na odpowiedź od abonenta. Po upływie tego czasu w słowie raportu wymiany znajdują się informacje o jej przebiegu. Na ustalenie tego parametru będzie miał wpływ rezultat analizy pracy abonenta, jego zadania i czas ich wykonania. Innymi słowy, na wartość tego parametru zasadniczy wpływ będzie miał czas trwania cyklu jednostki centralnej T_A.

Liczba prób nawiązania łączności

Jest to również bardzo istotny parametr mający wpływ na przepustowość sieci, a o którym była mowa przy okazji omawiania powstawania zjawiska „dziur" w sieci. Jeśli stacja MASTER wykryje nieobecność abonenta, to, zgodnie z wartością tego parametru, może przez określoną liczbę razy próbować nawiązać z nim łączność. Parametr ten będzie

miał małą wartość lub wręcz przyjmie wartość zero dla wymian mało istotnych i dużą dla wymian o podstawowym znaczeniu dla pracy całego systemu. Za pomocą tego parametru można również niejako przydzielić dodatkowy czas dla stacji SLAVE, jeśli z góry wiadomo, że w określonych okolicznościach nie jest ona w stanie przygotować na czas odpowiedzi. Dzięki temu parametrowi można optymalizować czas wymiany. Czyniąc to, nie należy zapominać, przy analizie czasowej, o braniu pod uwagę najgorszego przypadku. Najgorszy przypadek dotyczy okoliczności, w których stacja MASTER będzie, zgodnie z tym parametrem, starać się nawiązać łączność maksymalną liczbę razy.

Maksymalna liczba cykli sieci przy braku odpowiedzi

O tym parametrze również wspomniano opisując zjawisko „dziur". Jeżeli stacja MASTER wykryje brak abonenta, to przez określoną tym parametrem liczbę cykli sieci nie będzie próbowała z nim nawiązać komunikacji. Ma to znaczny wpływ na przepustowość sieci. Za pomocą tego parametru podobnie jak i poprzedniego można przydzielić „dodatkowy" czas dla określonego abonenta, jeśli wiadomo, że nie będzie on w stanie, w pewnych ekstremalnych przypadkach, przygotować odpowiedzi w zadanym czasie T_{OD}.

5.4. Zjawiska zachodzące na styku „koprocesor – jednostka centralna"

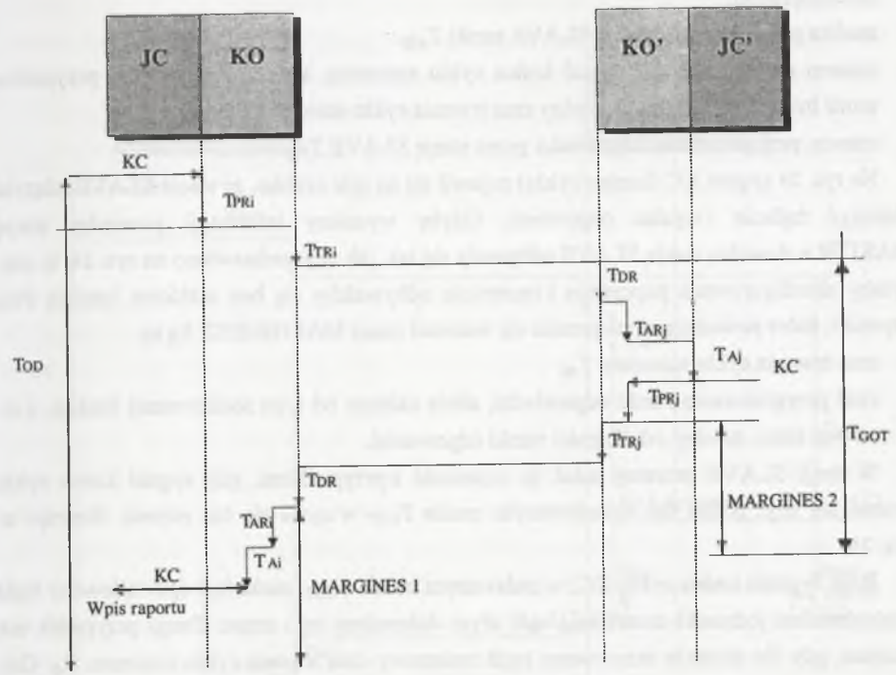
Analiza zjawisk zachodzących na styku „koprocesor – jednostka centralna" jest bardzo istotna dla obliczenia cyklu wymiany informacji, a także dla prawidłowego skonfigurowania całej sieci. Z informacji dotyczących parametryzacji wymian wynika konieczność bardzo dokładnego określenia dwóch wielkości, które są związane z:

- czasem oczekiwania przez stację MASTER na odpowiedź ze stacji SLAVE, (T_{OD}),
- czasem oczekiwania na gotowość jednostki centralnej stacji SLAVE (T_{GOT}).

Jeżeli oba te parametry zostaną źle dobrane, to sieć będzie pracować wadliwie. Będzie się to objawiało częstymi komunikatami nadchodzącymi ze stacji SLAVE o braku gotowości jednostki centralnej i niezrealizowaniu wymiany i / lub sygnalizacji przez stację MASTER o uszkodzeniu abonenta, choć to nie musi być prawdą.

Na rys. 24 przedstawiono zależności obu czasów T_{OD} i T_{GOT} od takich czynników, jak: czas analizy i przygotowania ramki, czas transmisji ramki czy czas trwania cyklu sterownika. Na rys. 24 kolejne symbole oznaczają:

- KO i JC odpowiednio koprocesor i jednostka centralna stacji MASTER,
- KO' i JC' odpowiednio koprocesor i jednostka centralna stacji SLAVE,
- T_{OD} czas odpowiedzi stacji SLAVE dla pojedynczej transakcji wymiany, będący parametrem,



Rys. 24. Zależności czasowe dla pojedynczej wymiany typu „żądanie – odpowiedź"
Fig. 24. The timing of single exchange „request-answer"

- T_{GOT} czas gotowości jednostki centralnej stacji SLAVE, będący parametrem,
- T_{PRi} czas przygotowania ramki żądania,
- T_{TRi} czas transmisji ramki żądania,
- T_{DR} czas detekcji ramki,
- T_{ARj} czas analizy ramki żądania w stacji SLAVE,
- T_{Aj} czas trwania cyklu automatu jednostki centralnej stacji SLAVE,
- T_{PRj} czas przygotowania ramki odpowiedzi,
- T_{TRj} czas transmisji ramki odpowiedzi,
- T_{ARi} czas analizy ramki odpowiedzi,
- T_{Ai} czas trwania cyklu automatu jednostki centralnej stacji MASTER,
- KC sygnał końca cyklu automatu.

Na rys. 24 przedstawiono przypadek takiego doboru czasów T_{OD} i T_{GOT}, że powstały dwa marginesy czasowe MARGINES1 i MARGINES2. MARGINES1 jest różnicą pomiędzy zadanym czasem T_{OD} a czasem, który upłynął od chwili emisji danych od stacji MASTER do chwili zakończenia detekcji ramki odpowiedzi. MARGINES2 jest natomiast różnicą pomiędzy zadanym dla abonenta czasem gotowości T_{GOT} a sumą czasów związanych z:

- detekcją ramki,
- analizą przyjętej przez stację SLAVE ramki T_{ARj},
- czasem oczekiwania na sygnał końca cyklu automatu, który w granicznym przypadku może być równy T_{Aj} (maksymalny czas trwania cyklu automatu),
- czasem przygotowania odpowiedzi przez stację SLAVE T_{PRj}.

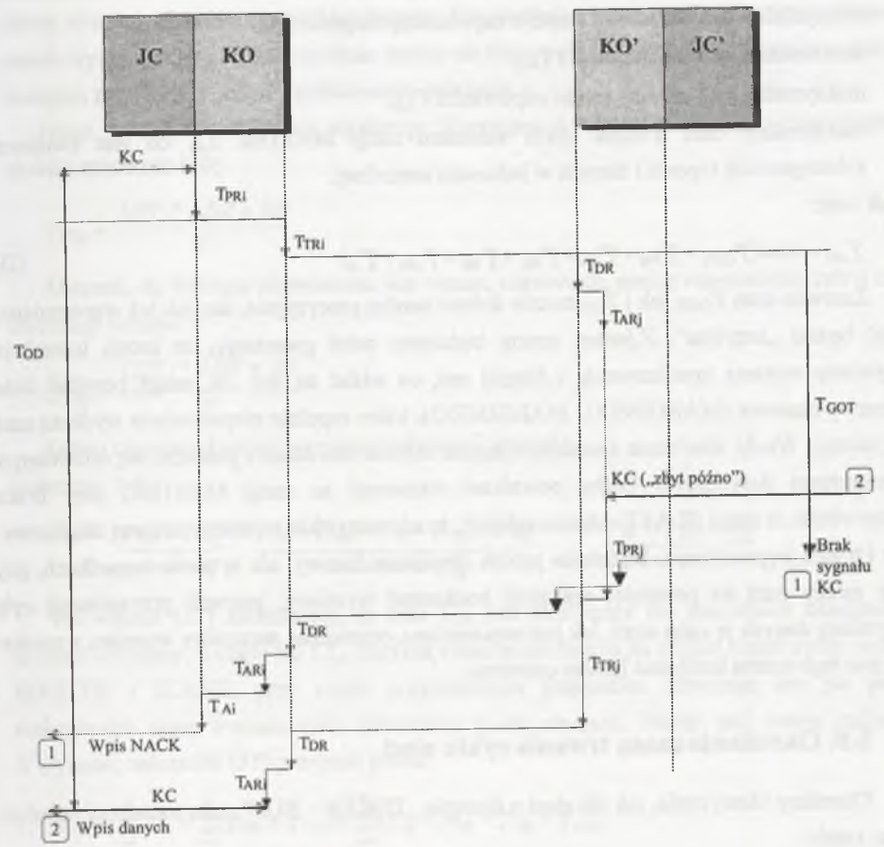
Na rys. 24 sygnał KC (koniec cyklu) pojawił się na tyle szybko, że stacja SLAVE zdążyła obsłużyć żądanie i wysłać odpowiedź. Gdyby wymiany informacji pomiędzy stacją MASTER a dowolną stacją SLAVE odbywały się tak, jak to przedstawiono na rys. 24, to sieć byłaby skonfigurowana poprawnie i transmisja odbywałaby się bez zakłóceń. Istnieją dwa czynniki, które powodują zmniejszanie się wartości czasu MARGINES2. Są to:

- czas trwania cyklu automatu T_{Aj},
- czas przygotowania ramki odpowiedzi, silnie zależny od typu realizowanej funkcji, a co za tym idzie, zależny od długości ramki odpowiedzi.

W stacji SLAVE możemy mieć do czynienia z przypadkiem, gdy sygnał końca cyklu pojawi się zbyt późno lub w zadawanym czasie T_{GOT} w ogóle się nie pojawi. Ilustruje to rys. 25.

Brak sygnału końca cyklu KC, w zadawanym czasie T_{GOT}, może być spowodowany bądź uszkodzeniem jednostki centralnej, bądź złym dobraniem tego czasu. Drugi przypadek ma miejsce, gdy źle zostanie oszacowany bądź zmierzony czas trwania cyklu automatu T_{Aj}. Gdy w zadawanym czasie T_{GOT} nie pojawi się sygnał KC, to stacja SLAVE przygotuje ramkę odpowiedzi NACK i ją wyśle. Gorzej będzie, gdy sygnał KC pojawi się zbyt późno, a ramka odpowiedzi będzie złożona. Dotyczy to na przykład transmisji wielu słów lub transmisji zawartości pamięci programu stacji SLAVE. Wydłuży się wtedy zarówno czas przygotowania ramki odpowiedzi, jak i czas jej transmisji. Może to już mieć znaczny wpływ na jakość transmisji.

Na rys. 25 ramka odpowiedzi ze stacji SLAVE, przy zbyt późnym pojawieniu się w niej sygnału KC, została zdekodowana i zrealizowana dokładnie w chwili pojawienia się sygnału KC w stacji MASTER, co pokryło się z zakończeniem okresu czasu T_{OD}. Z rys. 25 wynika ponadto, że stacja MASTER może otrzymać szybciej ramkę odpowiedzi w przypadku upłynięcia czasu T_{GOT} w stacji SLAVE (np. klasyczny *time–out*) niż w przypadku zbyt późnego, ale w granicach czasu T_{GOT}, pojawienia się sygnału końca cyklu automatu. Nie można zatem konfigurować takiej sieci jedynie biorąc pod uwagę klasyczny czas odpowiedzi (*time–out*).



Rys. 25. Zależności czasowe przy braku lub zbyt późnym pojawianiu się sygnału KC
Fig. 25. The timing of exchanges when KC interrupt is absent or is too late.

W systemach sieciowych, w których zastosowano sieć o dostępie MASTER–SLAVE, czas T_{GOT} może być indywidualnym parametrem każdej ze stacji SLAVE, ale może być również jej parametrem globalnym. Bez względu na fakt globalności czy lokalności tego parametru jego wartość dla każdej ze stacji powinna być dobrana następująco:

T_{GOT} > CZAS DETEKCJI RAMKI + CZAS ANALIZY RAMKI + MAKSYMALNY CZAS TRWANIA CYKLU AUTOMATU STACJI SLAVE + MAKSYMALNY CZAS PRZYGOTOWANIA RAMKI ODPOWIEDZI

$$T_{GOTj} = T_{DR} + \max(T_{ARj}) + \max(T_{Aj}) + \max(T_{PRj}) \quad (28)$$

Przy doborze czasu T_{OD} dla stacji MASTER należy pamiętać o tym, że musi być on większy od największej wartości T_{GOTj} i powiększony o:

- maksymalny czas transmisji ramki żądania T_{TRi},
- maksymalny czas transmisji ramki o największej długości T_{TRj},
- czas detekcji ramki odpowiedzi T_{DR},
- maksymalny czas analizy ramki odpowiedzi T_{ARi},
- maksymalny czas trwania cyklu automatu stacji MASTER T_{Ai}, co jest związane z dostępnością raportu i danych w jednostce centralnej.

Tak więc:

$$T_{OD} = \max(T_{GOTj} + T_{PRi} + T_{TRi} + T_{TRj} + T_{DR} + T_{ARi} + T_{Ai}) \tag{29}$$

Zarówno czas T_{GOT}, jak i T_{OD} można dobrać bardzo precyzyjnie, ale, jak już wspomniano, sieć będzie „sztywna". Z jednej strony będziemy mieć gwarancję, że każda transakcja wymiany zostanie zrealizowana, z drugiej zaś, co widać na rys. 24, mogą powstać duże rezerwy czasowe (MARGINES1, MARGINES2), które zupełnie niepotrzebnie wydłużą czas wymiany. Wtedy absolutnie świadomie można skrócić oba czasy i posłużyć się omawianym parametrem dotyczącym liczby powtórzeń transmisji ze stacji MASTER, przy braku odpowiedzi ze stacji SLAVE. Można założyć, że nie wszystkie wymiany zostaną zrealizowane i trzeba je powtórzyć. Będzie to proces absolutnie losowy, ale w wielu wypadkach, gdy nie zależy nam na pewności realizacji konkretnej wymiany, pozwoli przyspieszyć cykl wymiany danych w całej sieci. Jak już wspomniano, omawiając parametry wymian, z mechanizmu tego trzeba korzystać bardzo ostrożnie.

5.5. Określenie czasu trwania cyklu sieci

Określmy identycznie, jak dla sieci o dostępie „TOKEN – BUS", czas transmisji pojedynczej ramki

$$T_{TR} = \frac{LZT*LBZ + BS}{V}, \tag{30}$$

gdzie kolejne symbole oznaczają jak poprzednio:

LZT – liczba znaków transmisji,

LBZ – liczba bitów przypadająca na jeden znak transmisji,

BS – liczba bitów sterujących związanych z kodowaniem (warstwa liniowa),

V – prędkość transmisji wyrażona w bitach na sekundę.

Rozpatrzmy najpierw przypadek sieci, w której nie dopuszcza się możliwości powtórzeń transmisji do stacji, która nie odpowiada w czasie T_{OD} lub odpowiada ramką NACK. Innymi słowy, będzie rozpatrywana sieć bez „dziur".

Jak wiadomo, każdej ramce transmitowanej od stacji MASTER towarzyszy ramka odpowiedzi od stacji SLAVE, do której wymiana była adresowana. Po opracowaniu

scenariusza wymian wiadomo precyzyjnie, ile i jakie wymiany będą realizowane. Znane są zatem również długości wszystkich ramek. Nie wnikając w szczegóły dotyczące długości ramek wyrażonych w bitach, co silnie zależy od firmowej implementacji sieci o omawianym dostępie, przyjmijmy jedynie podstawowe założenia.

Niech w analizowanej sieci występuje N wymian, a czas transmisji „i" – tego żądania określa zależność (*30*).

$$T_{TRi} = \frac{LZT_i * LBZ + BS}{V} \tag{31}$$

Abonent, do którego adresowana jest ramka, odpowiada ramką odpowiedzi, której czas transmisji wynosi:

$$T_{TRj} = \frac{LZT_j * LBZ + BS}{V} \tag{32}$$

Zatem czas pojedynczej wymiany informacji wynosi:

(patrz rys. 24)

$$T_{Wi} = \underbrace{T_{PRi} + T_{TRi}}_{\text{emisja żądania } T_{\dot{z}i}} + \underbrace{T_{DR} + T_{ARj} + T_{Aj} + T_{PRj} + T_{TRj}}_{\text{emisja odpowiedzi } T_{Oj}} + \underbrace{T_{DR} + T_{ARi} + T_{Ai}}_{\text{analiza i wpis rezultatu wymiany } T_{RAi}} \tag{33}$$

We wzorze (*33*) zakłada się, że czas T_{DR} jest identyczny dla wszystkich uczestników procesu wymiany, a czasy T_{Ai} i T_{Aj} dotyczą czasu oczekiwania na sygnał końca cyklu w stacji MASTER i SLAVE, przy czym w granicznym przypadku oznaczają one po prostu maksymalne czasy trwania cykli automatów w obu stacjach. Biorąc pod uwagę realizację N wymian, zależność (*33*) przyjmie postać:

$$T_{CW} = \sum_{i=1}^{N} T_{Wi} = \sum_{i=1}^{N} (T_{PRi} + T_{TRi}) + \sum_{j=1}^{N} (T_{Aj} + T_{PRj} + T_{TRj} + T_{ARj}) + \sum_{i=1}^{N} (T_{ARi} + T_{Ai}) + 2N T_{DR} \tag{34}$$

Zależność (*34*) pozwala określić „bezpieczne" wartości czasów T_{GOT} i T_{OD}, po przygotowaniu oczywiście listy wszystkich wartości T_{Wi} i wybraniu wartości maksymalnych, które występują w zależnościach (*28*), (*29*).

Cykl wymiany informacji wydłuży się, jeżeli dopuszczone zostaną repetycje, co odpowiada pracy sieci z „dziurami". Parametryzując sieć, określa się, która wymiana wyzwalana ma być powtarzana i jaką liczbę razy.

Załóżmy, że przewiduje się liczbę N_R repetycji dla całego scenariusza wymian. Każdej repetycji towarzyszy transmisja ramki żądania i brak odpowiedzi lub pojawienie się ramki NACK. W najgorszym przypadku, a taki trzeba brać pod uwagę, stacja MASTER będzie czekać przez czas T_{ODi} na odpowiedź. Nakład czasowy na realizację repetycji może, w najgorszym przypadku, być równy:

$$T_{REP} = \sum_{i=1}^{N} N_{Ri} T_{TRi} + N_R T_{ODi}, \qquad (35)$$

gdzie: T_{TRi} – oznacza czas transmisji żądania dla „i" – tej repetycji,

N_{Ri} – oznacza liczbę repetycji do „i" –tego abonenta,

N – oznacza liczbę abonentów.

Zatem maksymalny czas wymiany informacji w sieci wyniesie:

$$T_{WMAX} = T_{CW} + T_{REP} \qquad (36)$$

Zależność (*36*) określa maksymalny czas wymiany informacji w sieci, ale tylko dla realizacji wymian cyklicznych. Jeśli dopuścić również wymiany wyzwalane, realizowane na rozkaz jednostki centralnej stacji MASTER, to czas T_{WMAX} się wydłuży. Aby odpowiedzieć na pytanie o ile, trzeba znać dokładnie firmową implementację sieci. Jeżeli konkretna implementacja zakłada, że w trakcie trwania cyklu automatu może być realizowana tylko jedna wymiana wyzwalana (jest to najczęściej spotykane rozwiązanie), to pojedynczy cykl wymiany wydłuży się o czas

$$T_{RW} = T_{Wi},$$

który należy obliczyć wiedząc, jaki rodzaj wymiany będzie realizowany (funkcja i jej realizacja).

Jeżeli w pojedynczym cyklu pracy aplikacji węzła zostanie wyzwolonych N_W wymian, to ich realizacja będzie wymagała dodatkowo N_W cykli tejże aplikacji. Wtedy całkowity czas wymiany wszystkich informacji w sieci wzrośnie o

$$T_{WW} = N_w T_{Aj} + \sum_{i=1}^{N_w} T_{Wi}, \qquad (37)$$

gdzie: T_{Wi} jest czasem realizacji „i" – tej wymiany wyzwalanej.

Realizacja wymian wyzwalanych powoduje zwiększenie nakładu programowego w jednostce centralnej stacji MASTER na ich obsługę. Realizując przykładowy (rys. 26) algorytm komunikacji pomiędzy jednostką centralną a koprocesorem, program aplikacyjny musi rozbudować się o elementy nie tylko związane z analizą raportów wymian, ale kolejkowania wymian wyzwalanych, gdyż nie można rozpocząć realizacji następnej wymiany, gdy poprzednia jeszcze się nie zakończyła. Realizacja pojedynczej wymiany wyzwalanej polega bowiem na wykonaniu w jednostce centralnej określonego rozkazu lub sekwencji rozkazów. Rozkazy te są oczywiście elementem składowym programu aplikacyjnego, który wykonuje się cyklicznie. Jeżeli zatem wykonamy szereg po sobie występujących rozkazów inicjujących wymiany wyzwalane, to polecenia te zostaną przekazane do koprocesora. Jeśli ten otrzymuje dostęp do pamięci macierzystej jednostki tylko raz w czasie jednego cyklu, to reszta wymian nie zostanie w tym cyklu wykonana.

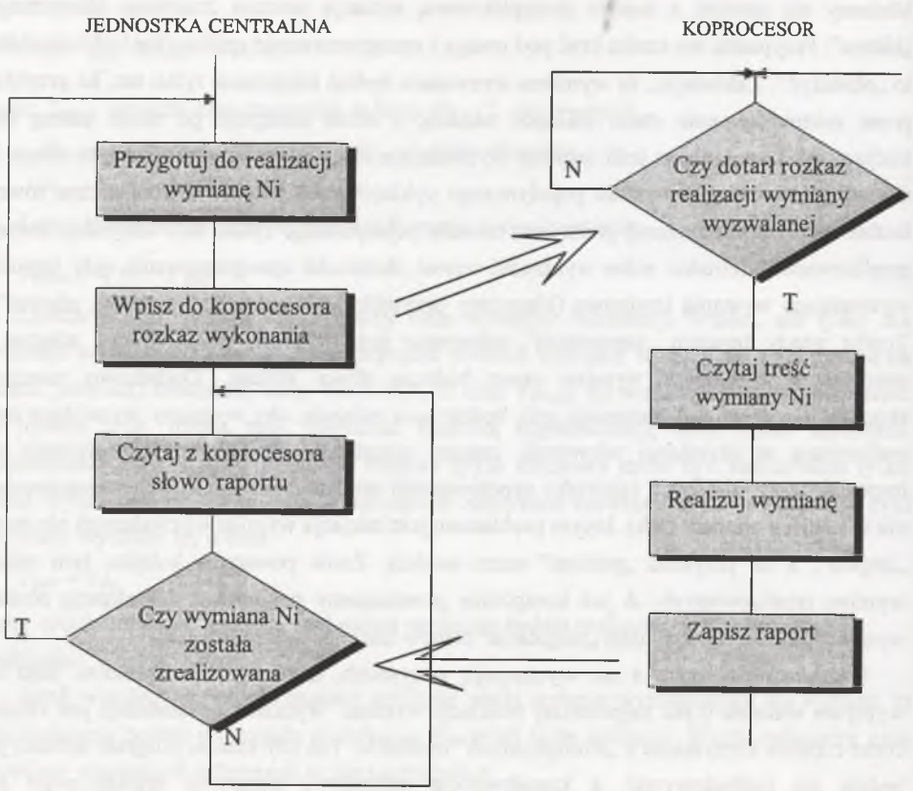
Możemy się spotkać z bardzo skomplikowaną sytuacją noszącą znamiona klasycznego „zatoru". Przypadek ten trzeba brać pod uwagę i oprogramowanie aplikacyjne musi zjawisko to „obsłużyć". Zakładając, że wymiana wyzwalana będzie inicjowana tylko raz, na przykład przez zróżniczkowanie stanu jakiegoś wejścia, i niech następuje po sobie szereg tak inicjowanych wymian, to jeśli impulsy wyzwalające nie występują w tym samym czasie (z dokładnością do czasu trwania pojedynczego cyklu), trzeba jedynie odczekać czas równy liczbie wymian pomnożonej przez czas trwania pojedynczego cyklu, aby wszystkie zostały zrealizowane. Nietrudno sobie wyobrazić wzrost złożoności oprogramowania, gdy impulsy wyzwalające wystąpią lawinowo (klasyczny przypadek tak zwanego „monitora zdarzeń"). Trzeba wtedy bowiem „zapamiętać" sekwencję pojawiających się lawinowo zdarzeń i sterować wyzwalaniem wymian przez badanie słowa statusu. Dodatkowo znacznie skomplikuje się proces transmisji, gdy będzie nam zależało, aby wymiany wyzwalane były realizowane w określonej sekwencji, czasem niezależnie od kolejności pojawiania się impulsów wyzwalających (zjawisko synchronizacji uruchamiania i realizacji oprogramowania w stacjach abonenckich). Innym problemem jest inicjacja wymian wyzwalanych nie przez „impuls", a na przykład „poziom" stanu wejścia. Znów powstanie kolejka, tym razem wymian zrealizowanych. A już kompletnie przestaniemy panować nad realizacją obsługi wymian, gdy nastąpi zjawisko „migotania" stanów inicjujących transmisję.

Omawiane tu zjawiska nie wyczerpują wszystkich, możliwych przypadków. Stąd też wypływa wniosek o jak najprostszej realizacji wymian. Wyrazem tej tendencji jest choćby coraz częstsze korzystanie z „inteligentnych" modułów. Tak czy inaczej program aplikacyjny będzie się rozbudowywał, a konsekwencją rozbudowy programu aplikacyjnego jest wydłużenie się czasu trwania cyklu automatu, a to prowadzi, zgodnie z opisanymi już zjawiskami, do wydłużenia cyklu pracy sieci. Przy tej okazji warto jeszcze raz podkreślić istotną dla czasu realizacji wymian sieciowych konieczność optymalizacji programów sterowania, regulacji i wizualizacji, które są realizowane przez jednostkę centralną stacji abonenckiej. Jest bardzo wskazane, aby na etapie uruchamiania oprogramowania zmierzyć czas trwania cyklu sterownika lub przynajmniej oszacować z pewnym bezpiecznym marginesem jego wartość.

Rozpatrzmy następujący przykład.

Obliczyć maksymalny czas trwania cyklu sieci MASTER–SLAVE dla trzech abonentów przy następujących założeniach:

- stacja MASTER realizuje 3 wymiany:
  - wymiana nr 1 z abonentem nr 1 dotyczy odczytu 1 słowa ze stacji SLAVE,
  - wymiana nr 2 z abonentem nr 2 dotyczy zapisu 5 słów do stacji SLAVE,
  - wymiana nr 3 z abonentem nr 3 dotyczy zapisu pojedynczego bitu do stacji SLAVE,



Rys. 26. Przykłady algorytmu realizacji wymiany wyzwalanej
Fig. 26. An example of trigger exchange algorithm

- maksymalny czas trwania cyklu sterownika stacji MASTER T_{Ai} wynosi120 ms,
- wymiany odbywają się w trybie pracy cyklicznej (wymiany cykliczne),
- dla każdej ze stacji SLAVE przyjęto następujące, maksymalne czasy cykli sterowników:
  - $TA_1 = 100$ ms,
  - $TA_2 = 80$ ms,
  - $TA_3 = 150$ ms,

oraz liczby powtórzeń przy braku odpowiedzi lub otrzymania odpowiedzi negatywnej (NACK):

- dla stacji 1 $Nr_1 = 2$,
- dla stacji 2 $Nr_2 = 5$,
- dla stacji 3 $Nr_3 = 10$,

- liczba bitów przypadająca na 1 znak transmisji *LBZ* wynosi 10, a liczba bitów sterujących *BS* równa jest 11,
- prędkość transmisji równa jest 19 200 b/s, stąd T_{DR} (czas transmisji 2.5 znaku- wzór *(13)*) wynosi 1.875 ms.

Niech ponadto (dane konkretnego producenta) maksymalne czasy T_{ARi} i T_{PRi} dla wszystkich koprocesorów wynoszą odpowiednio:

- $T_{ARi} = 2.0$ ms,
- $T_{PRi} = 3.6$ ms.

Przyjmijmy, że dla kolejnych wymian liczby znaków transmisji wynoszą (sieć MODBUS RTU):

- wymiana 1 – odczyt 1 słowa 16-bitowego:
  - liczba znaków ramki żądania *LZT* wynosi 8,
  - liczba znaków ramki odpowiedzi *LZT* wynosi 7,
- wymiana 2 – zapis 5 słów 16-bitowych:
  - liczba znaków ramki żądania *LZT* wynosi 19,
  - liczba znaków ramki odpowiedzi *LZT* wynosi 8,
- wymiana 3 – zapis pojedynczego bitu:
  - liczba znaków w ramce żądania *LZT* wynosi 10,
  - liczba znaków w ramce odpowiedzi *LZT* wynosi 8.

W tabeli 1 zamieszczono między innymi wyliczone czasy T_{TRi} i T_{TRj} dla każdej wymiany (rys. 24 i rys. 25).

Tabela 1

Maksymalne wartości czasów transmisji, obsługi i wymian

Nr wymiany	T_{TRi} ms	T_{TRj} ms	T_{2i} ms	T_{Oj} ms	T_{RAi} ms	T_{Wi} ms	T_{GOTj} ms	T_{ODi} ms
1	4.74	4.22	8.34	111.64	123.82	243.80	107.47	**300.05**
2	10.47	4.74	14.07	92.16	123.82	230.05	87.42	
3	5.78	4.74	9.38	162.16	123.82	295.36	157.42	

Zawiera ona również wartości czasów T_{GOTj} i T_{ODi} dobranych według zależności (wzór *(28)* i wzór *(29)*. Ponadto także wartości czasów T_{Wi} dla każdej wymiany.

Wyniki w tabeli 1 świadczą, jak silnie czasy wymiany T_{Wi} oraz czasy T_{ODi} i T_{GOTj} zależą od długości cykli automatów i czasu przetwarzania koprocesora. Maksymalny czas transmisji ramki żądania i odpowiedzi wynosi bowiem 15.21 ms (dla wymiany 2), co stanowi około 5.1% żądanego czasu odpowiedzi T_{ODi}.

Zaprogramowanie dla hipotetycznej instalacji sieciowej podanych czasów T_{ODi} dla stacji MASTER (jeśli oczywiście jest to parametr globalny) i indywidualnych czasów gotowości

T_{GOTj} dla każdej stacji SLAVE gwarantuje, że każda żądana wymiana informacji zostanie zrealizowana bez dodatkowych powtórzeń, o ile oczywiście wszystkie stacje abonenckie będą sprawne techniczne.

W omawianym przykładzie założono dodatkowo, że będą występowały powtórzenia transmisji do stacji SLAVE. Przy tak dobranych parametrach T_{ODi} i T_{GOTj} założenie dotyczy jedynie przypadków awaryjnych związanych z niesprawnością koprocesorów. Obliczmy zatem, ile wynosi cykl wymiany informacji i o ile wzrośnie jego wartość, jeśli będą powtórzenia.

Korzystając z zależności (wzór *(35)*) dodatkowy czas związany z repetycją ramek wyniesie:

$$T_{REP} = \left(N_{R1}T_{TR1} + N_{R2}T_{TR2} + N_{R3}T_{TR3}\right) + \left(N_{R1} + N_{R2} + N_{R3}\right) \times T_{ODi} \qquad (38)$$

Podstawiając do wzoru wyliczone wartości z tabeli 1 oraz założone liczby powtórzeń, uzyskamy wartość czasu T_{REP}, która wyniesie:

$$T_{REP} = 119.63ms + 5100.85ms = 5.22s \qquad (39)$$

A suma czasów T_{Wi} wyniesie:

$$T_{CW} = 769.21\,m \qquad (40)$$

Uzyskane wielkości są zaskakujące. Na rys. 22 przedstawiono przykładowy scenariusz wymian, gdzie zaproponowano czasy repetycji rzędu od 50 ms do 100 ms. Gdyby chcieć zrealizować przedstawiony na rys. 22 scenariusz wymian, to w świetle uzyskanych wyników widać, że jest to nie do osiągnięcia przy przyjętych założeniach. Zakładając brak powtórzeń (przy sprawnej technicznie pracy sieci), uzyskaliśmy czas trwania cyklu ponad 7.6 raza większy od maksymalnego, założonego cyklu repetycji wymiany (100 ms). Powtórzmy raz jeszcze, że sieć została skonfigurowana dość „sztywno", ale dzięki temu mamy gwarancję, że każda wymiana zostanie zrealizowana bez powtórzeń. Powstaje pytanie, czy możliwe jest inne jej skonfigurowanie, mniej rygorystyczne. Odpowiedź jest trudna, gdyż zależy to od zastosowania sieci. Jeśli proces obsługiwany przez sieć wymaga pełnej gwarancji transmisji danych w określonym czasie (np. transmisja zabezpieczeń), to nie ma wyjścia i należy ją skonfigurować, tak jak to pokazano. Jeśli natomiast dopuszczamy ryzyko, że nie każda wymiana musi się udać, to wtedy mamy do dyspozycji parametr dotyczący liczby repetycji, który pozwala uzyskać dużo lepsze wyniki. Pierwsza wskazówka dotyczy maksymalnego czasu trwania cyklu automatu. Jeśli założymy o połowę krótszy czas trwania cyklu, to zmniejszą nam się wartości czasów T_{GOTj}, a tym samym również czas T_{ODi}. Wtedy należy dla każdej ważnej wymiany przewidzieć odpowiednią liczbę powtórzeń, tak aby zwiększyć prawdopodobieństwo jej realizacji. Trzeba przewidzieć, że przy takim podejściu do konfiguracji końcową jej weryfikację uzyskamy dopiero podczas prób na obiekcie.

5.6. Sprawność sieci i jej przepustowość użyteczna

Niech sprawność sieci wyraża się, jak poprzednio, zależnością:

$$\eta = \frac{\text{czas transmisji danych użytkowych w pojedynczej transakcji wymiany}}{\text{całkowity czas pojedynczej transakcji danych}}[\%], \qquad (41)$$

a przepustowość użyteczna zależnością:

$$P = \frac{\text{liczba danych użytkowych w pojedynczej transakcji wymiany}}{\text{całkowity czas pojedynczej transakcji danych}}[kb/s] \qquad (42)$$

Zakładając, że w pojedynczej wymianie transmitowanych jest „n" bajtów danych użytkowych i pamiętając, że transmisji ramki danych zawsze towarzyszy ramka „żądania", sprawność i przepustowość użyteczną sieci można wyrazić następująco:

$$\eta = \frac{\frac{n*8}{V}}{T_{Wi}}[\%] \qquad (43)$$

$$P = \frac{n*8}{T_{Wi}}[kb/s] \qquad (44)$$

Przyjmując raz wartość maksymalną raz minimalną wielkości T_{Wi}, co ma miejsce przy założeniu minimalnego czasu trwania cyklu automatu, otrzymamy maksymalne i minimalne wartości sprawności i przepustowości. Dla ilustracji, biorąc pod uwagę omawiany przykład, uzyskamy:

$$\eta[\%] = \frac{\frac{n*8}{V}}{T_{Wi}} = \frac{\frac{10*8}{9600}}{230.05} = \frac{80}{9600*230.05} = 3.5[\%] \qquad (45)$$

$$P = \frac{n*8}{T_{Wi}} = \frac{80}{230.05} = 0.36[kb/s] \qquad (46)$$

Otrzymane rezultaty, na pierwszy rzut oka, są zastanawiające, ale biorąc pod uwagę najgorszy przypadek, są prawdziwe. Żaden z producentów, w materiałach reklamowych, nie pochwali się takimi parametrami. Jak już wspomniano, uzyskane rezultaty uwzględniają czasy związane z pełną transakcją wymiany porcji danych. Podobnie, jak w omawianej już sieci o dostępie TOKEN – BUS, spotyka się i tutaj definicje obu omawianych parametrów, w których nie uwzględnia się ani cykli automatów, ani czasów detekcji i przetwarzania ramki, ani czasu transmisji ramki „żądania [53]. Jest to znów raczej miara narzutu czasowego wnoszonego przez warstwę liniową sieci. Obliczmy zatem sprawność i przepustowość w odniesieniu jedynie do transmisji ramki danych. Uzyskamy wtedy następujące zależności:

$$\eta = \frac{\frac{n*8}{V}}{\frac{LZT*LBZ+BS}{V}} = \frac{n*8}{LZT*LBZ+BS}[\%] \quad (47)$$

$$P = \frac{n*8}{LZT*LBZ+BS}V[kb/s] \quad (48)$$

gdzie przypomnijmy:

V – oznacza prędkość transmisji,

LZT – oznacza liczbę znaków w ramce zawierającej dane,

LBZ – oznacza liczbę bitów przypadających na jeden znak transmisji,

BS – oznacza liczbę bitów sterujących, które „dokłada" warstwa liniowa.

Obliczmy zatem, tak jak poprzednio, dla wymiany 2 z analizowanego przykładu zarówno sprawność, jak i przepustowość użyteczną.

W przykładzie tym:

$n = 10$,

$LZT = 10$ (8 bitów / znak + 1 bit stopu + 1 bit parzystości),

$BS = 11$ (3 + 5 (preambuła) + 3(postambuła)),

$V = 9\,600$ b/s.

Wtedy:

$$\eta[\%] = \frac{n*8}{LZT*LBZ+BS} = \frac{80}{19*10+11} = 39.8[\%] \quad (49)$$

$$P = \frac{n*8}{LZT*LBZ+BS}V = 3.82kb/s \quad (50)$$

Oba sposoby obliczania sprawności i przepustowości sieci ilustrują, jak wielkie są różnice w wynikach w zależności od podejścia do zagadnienia.

Oczywiście, uzyskane wyniki należy odnieść do analizowanego przykładu. Przy dłuższych ramkach zarówno sprawność, jak i przepustowość będą dużo większe. Celem przykładu było zwrócenie uwagi na fakt, iż operowanie w materiałach firmowych wartościami obu parametrów nie odzwierciedla faktycznych wielkości w odniesieniu do konkretnej aplikacji. Ponadto z zależności (*47*) i (*48*) wynika, że nie zależą one od czasu transmisji ramki danych. Wystarczy zatem zwiększyć długość ramki, aby oba parametry poprawić. Ale wtedy przecież rośnie czas transmisji i przede wszystkim czas przetwarzania ramki, co w znacznym stopniu wydłuży cykl sieci.

5.7. Poprawa parametrów czasowych wymian

Wyniki uzyskane w poprzednim rozdziale wskazują na podstawowy wpływ takich parametrów, jak czasy cykli sterowników (bądź abonentów nie będących sterownikami swobodnie programowalnymi) oraz czasy przetwarzania koprocesorów, na całkowity czas wymiany informacji w sieci i nakazują znów ostrożność w dążeniu do zwiększenia prędkości transmisji, celem poprawienia przepustowości użytecznej i sprawności sieci. Rezerw czasowych należy szukać raczej w procesie polepszenia parametrów czasowych wymian, a gdy to zawiedzie, to pozostaje jedynie wybór szybszych sterowników i koprocesorów. Trudno jest podać uniwersalną zależność, która definiowałaby proces poprawy parametrów czasowych wymian. Można raczej podać kilka zasad, których programista, konfigurujący sieć, powinien przestrzegać. Proces zmierzający do poprawy parametrów czasowych dotyczy przede wszystkim stacji MASTER, gdyż ona realizuje wszystkie wymiany. Tak więc poprawna parametryzacja modułu koprocesora MASTER jednostki centralnej sterownika (abonenta) głównego jest najistotniejsza. Jeśli chodzi o stacje SLAVE, to należy zadbać o to, aby cykle ich jednostek centralnych były jak najkrótsze. Jest to oczywiste i nie wymaga komentarza. Podajmy zatem zasady, których przestrzeganie pomoże w tym, aby realizacja wymian sieciowych odbywała się najsprawniej.

Należy pamiętać o tym, że są do dyspozycji dwa rodzaje wymian: wyzwalana i cykliczna. Pomimo że wymiany wyzwalane są realizowane wolniej niż wymiany cykliczne, to należy zmierzać do wyodrębnienia jak największej ilości informacji, które mogą być realizowane w trybie wymian wyzwalanych. Zazwyczaj są to polecenia lub dane wysyłane stosunkowo rzadko. Można do nich zaliczyć na przykład: przesył parametrów regulacji, nastaw oraz odczyt wartości monitorowanych do wizualizacji. Pozwoli to znacznie odciążyć sieć i sprawniej przesyłać dane superpilne, pilne i szybkozmienne.

Czas trwania cyklu stacji MASTER powinien być jak najkrótszy (stacji SLAVE również - była już o tym mowa). Podobnie jak w omawianej poprzednio sieci o dostępie TOKEN – BUS, często warto podzielić program jednostki centralnej na fragmenty (etapy), aby umożliwić częstszą komunikację jednostki centralnej z koprocesorem sieciowym. Można zaoszczędzić czas, rezygnując z raportu wymiany. Dotyczy to tylko przypadków, gdy mamy do czynienia z kilkoma kolejnymi wymianami periodycznymi z tym samym abonentem. Jeśli stacja MASTER odwołuje się do tej samej stacji SLAVE, to może być zbędne żądanie wpisu raportu z każdej ze zrealizowanych wymian.

Dane dotyczące tego samego abonenta powinny być transmitowane w minimalnej liczbie ramek. Oznacza to, że dane przeznaczone dla tego samego abonenta należy grupować w pamięci jednostki centralnej i przesyłać w dłuższej ramce. Istotna jest także kolejność wymian cyklicznych w scenariuszu wymian. Jeżeli stacja MASTER ma dokonać kilku wymian z tym samym abonentem (na przykład zapis, a następnie odczyt słów), to wymiany

te nie powinny następować jedna po drugiej, gdyż po każdej z nich abonent potrzebuje czasu na interpretację ramki i zawartego w niej polecenia, musi czekać aż uzyska dostęp do pamięci jednostki centralnej i przygotować ramkę odpowiedzi. Jeżeli istnieje konieczność wysłania do tego samego abonenta rozkazu zapisu i odczytu danych, to najpierw należy realizować odczyt, a potem zapis, gdyż w tej kolejności operacje te są szybsze. Jest to szczególnie ważne w tych implementacjach sieci, w których występuje buforowanie transmisji.

Jak wspomniano na wstępie rozdziału 5, sieci o dostępie MASTER–SLAVE są prostsze w konfigurowaniu i łatwiejsza jest analiza przepływu danych. Można precyzyjnie określić wszystkie przebiegi czasowe i wyliczyć, w każdym przypadku, czas trwania cyklu sieci. Zwracano już uwagę na fakt ogromnego wpływu czasu przetwarzania koprocesora i czas trwania cyklu automatu, na czas wymiany informacji. Nie we wszystkich aplikacjach abonentem sieci jest sterownik swobodnie programowalny, to też nie we wszystkich konkretnych zastosowaniach sieci ostatni z wymienionych parametrów będzie istotny. Nie mniej jednak należy zwrócić uwagę przy podejmowaniu decyzji o zakupie konkretnego sprzętu na pewne jego parametry, które jest bardzo trudno znaleźć w materiałach reklamowych. Dotyczy to szczególnie czasów przetwarzania koprocesorów.

W oparciu o przedstawioną analizę przepływu informacji w sieci o dostępie MASTER–SLAVE dokonano jej praktycznej weryfikacji. Przedmiotem analizy była rozległa sieć (długość medium wynosiła 3,5 km), w której występowało 24 abonentów. Abonentami sieci były zdalne moduły wejść analogowych (zdalne pomiary temperatury), a stacją MASTER sterownik swobodnie programowalny ALSPA C50, którego zadaniem było nie tylko zarządzanie transmisją, ale również sterował on zespołem sprężarek układu chłodniczego oraz systemem zabezpieczeń technologicznych. Dodatkowym zadaniem sterownika była obsługa dwukierunkowej transmisji, przy użyciu niezależnego łącza, z komputerem, będącym stacją wizualizacyjną. Komputer przetwarzał również algorytmy technologiczne i wysyłał polecenia do sterownika, który je realizował. Wyniki analizy zostały praktycznie zweryfikowane, a fakt, że system pracuje niezawodnie od 1992 roku, może świadczyć o jej prawidłowym przeprowadzeniu. Wyniki przeprowadzonej analizy były elementem projektu technicznego, którego właścicielem jest Zakład Przetwórstwa Mięsnego w Tarnowie. Toteż nie jestem w stanie odesłać Czytelnika do materiałów literaturowych.

6. SIECI O DOSTĘPIE PRODUCENT – DYSTRYBUTOR – KONSUMENT

Większość współczesnych systemów komunikacyjnych, bez względu na to, czy są objęte standardem czy nie, faworyzuje metodę komunikacji „punkt – punkt" pomiędzy rozproszonymi aplikacjami. Jeśli zatem więcej niż dwie aplikacje powinny być w interakcji, to użytkownik sieci musi zapewnić organizację interaktywnej wymiany danych. Dlatego też model „Klient – Serwer" jest często używany do opisu interakcji pomiędzy dwoma procesami. Proces („Klient") posiada inicjatywę w wymianie danych, a inny proces („Serwer") wykonuje żądania „Klienta". Omawiany model „Klient – Serwer" został zaadoptowany w komunikacji pionowej, mającej miejsce na przykład w stacji abonenckiej, gdzie ruch danych odbywa się poprzez wszystkie warstwy oprogramowania od warstwy fizycznej do warstwy aplikacji, a żądanej przez te same funkcje, pomiędzy różnymi warstwami oprogramowania. Żądania obsługi mogą pochodzić bądź z warstw wyższych, lub z warstw niższych oprogramowania. Model „Klient – Serwer" stawia więc, w uprzywilejowanej pozycji, punkt widzenia akcji opisanej przez funkcję, zaniedbując jednocześnie fakt, że te same dane mogą być wymieniane i używane przez różne funkcje. Dla przykładu, w tym modelu taka funkcja jak MMS (*ang. "Message Management System"* -transakcja komunikatów) pozwala dowolnej aplikacji uzyskać dostęp do zmiennej, której właścicielem jest inna aplikacja. Ale jeśli dwie aplikacje używają tej usługi, żądając dostępu do tej samej zmiennej, to wtedy nie ma gwarancji na to, że uzyskane wartości zmiennych będą te same. Jest to znany problem ochrony i synchronizacji dostępu do danych. Wychodząc naprzeciw żądaniom dotyczącym komunikacji poziomej, odbywającej się na przykład pomiędzy tymi samymi warstwami oprogramowania (funkcjami), ale w różnych stacjach abonenckich, w podstawowym okresie ważności i spójności danych, konieczne jest zdefiniowanie innego modelu. Tym modelem może być model „PRODUCENT – DYSTRYBUTOR – KONSUMENT" (PDK) [105, 106]. Protokół ten przypomina strukturę informatyczną, sterowaną przepływem argumentów. W literaturze można znaleźć publikacje, w których zagadnienia przepływu sterowania za pomocą strumienia argumentów są prezentowane, a których analiza jest przydatna przy analizie modelu PDK [162, 163, 164].

6.1. Model typu PRODUCENT – DYSTRYBUTOR – KONSUMENT

Podstawowymi cechami tego modelowego protokołu są:

- czasowa zgodność produkcji danych,
- czasowa spójność transmisji,
- przestrzenna zgodność wartości zmiennych.

Czasowa zgodność produkcji danych oznacza, że dane zostały wyprodukowane w tym samym przedziale czasowym. Cecha ta pozwala zapewnić próbkowanie wszystkich danych w tym samym momencie. Może się to okazać konieczne dla zapewnienia spójności algorytmu w aplikacji użytkownika.

Czasowa spójność transmisji oznacza, że dane docierają do wszystkich „konsumentów" w tym samym przedziale czasowym i w ten sposób są dostępne, kiedy zostaje uruchomiony algorytm aplikacyjny.

Przestrzenna zgodność wartości zmiennych dotyczy używania przez różne aplikacje, pracujące na różnych stacjach, list identycznych zmiennych. Przy czym użytkownicy tej samej listy zmiennych mają gwarancję, że kopie wartości zmiennych są identyczne dla nich wszystkich.

„**Producent**" jednostki danych jest odpowiedzialny na poziomie aplikacji za „produkcję" danych, która może być periodyczna lub nie, synchroniczna lub nie z innymi aplikacjami lub „produkcjami" danych.

„**Dystrybutor**" danych jest natomiast odpowiedzialny za transfer danych od „Producenta" do wszystkich „Konsumentów".

„**Konsumenci**" danych są aplikacjami, które będąc wykonywane, żądają danych. I znów, użycie danych może być periodyczne lub nie i synchroniczne bądź nie z innymi aplikacjami. Dystrybucja danych powinna gwarantować odpowiednią, czasową charakterystykę „produkcji" danych, uwzględniającą konsumpcję i ich dostępność, zgodnie z wymaganiami „Konsumentów". Dystrybucja danych również może być periodyczna lub nie, synchroniczna lub nie z innymi aplikacjami. Omawiany model PDK stawia na pozycji uprzywilejowanej „Konsumentów". Poza tym funkcja dystrybucji, statycznie zaprojektowana i skonfigurowana, może być modyfikowana, spełniając zmieniające się w czasie wymagania konsumentów. Model PDK zatem doskonale nadaje się do zaadoptowania w poziomych wymianach i również może być zadowalający dla określenia niektórych wymian pionowych.

6.2. Zasada działania sieci wykorzystującej protokół PDK

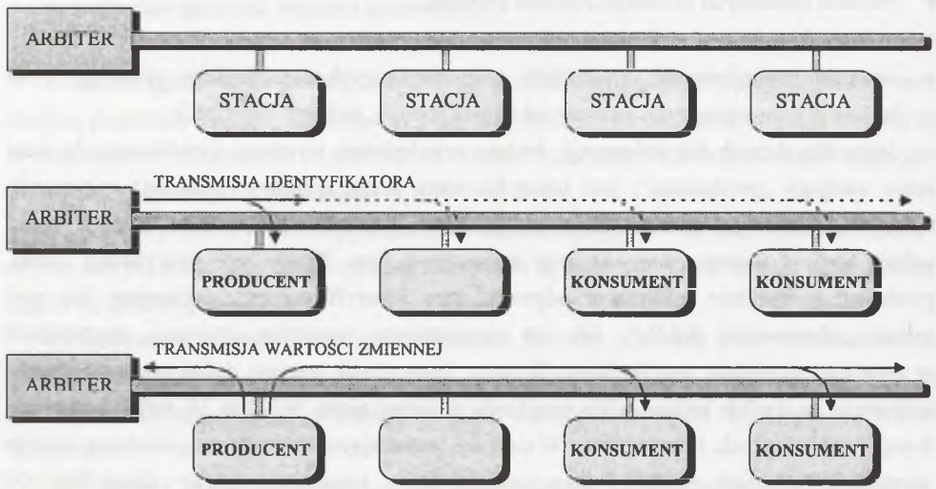
Siecią krążą wartości zmiennych, będące elementami listy wszystkich zmiennych systemowych (patrząc od strony funkcji programów sterujących) i posiadają swoje identyfikatory. Tymi zmiennymi mogą być:

- wartości zmiennych określających stan procesu,
- rozkazy do urządzeń wykonawczych,
- wartości przesyłane do swobodnie programowalnych sterowników przemysłowych będące ich wewnętrznymi zmiennymi (słowa statusu, wektory stanu itp.).

Jednostka danych lub informacji, będąca przedmiotem wymiany i emitowana do sieci przez swojego „producenta", jest identyfikowana przez unikalną (globalną) w systemie nazwę. Producent jest informowany o żądaniu emisji jednostki danych, które posiada, przez odbiór kodu – nazwy odpowiadającej żądanym danym. Wtedy, po identyfikacji ramki, producent transmituje informacje odpowiadające identyfikatorom. Mechanizm ten nosi miano „adresowania źródła". Jak już wspomniano, wszystkie zmienne, występujące w systemie, posiadają swoje identyfikatory, którymi są nazwy definiowane globalnie, kodowane na dwóch bajtach. (Dla przykładu zmienną może być stan 16 wejść binarnych, 8 wejść analogowych, lista, tablica). W sieci tej, posiadającej strukturę magistralową, istnieje wyróżniony abonent noszący miano arbitra, który transmituje nazwy identyfikatorów umieszczone na jego liście, a więc jest dystrybutorem danych. Identyfikatory zmiennych są transmitowane periodycznie i / lub aperiodycznie. Lista wszystkich zmiennych, istniejących w systemie, jak wspomniano, umieszczona jest w pamięci arbitra, zaś abonenci sieci posiadają lokalne listy zmiennych, których abonent jest bądź „producentem", bądź „konsumentem". Listy zmiennych w pamięci arbitra i listy zmiennych w pamięci abonentów sieci są tworzone na etapie jej konfiguracji i aplikacji. Tak więc w sieci krążą transmitowane przez arbitra identyfikatory zmiennych, a w odpowiedzi na nie transmitowane są przez producentów wartości owych zmiennych. Transmisje w sieci są typu „rozgłoszenie" (ang. „*broadcast*"), tak więc trafiają jednocześnie do wszystkich abonentów. Dotyczy to zarówno transmisji identyfikatora, jak i wartości zmiennej jej odpowiadającej.

Generalnie transmisje są tak zwanymi „transmisjami bez potwierdzenia", ale istnieją funkcje serwisowe żądające usługi transakcji danych z potwierdzeniem. „Konsumenci" czuwają jedynie nad detekcją błędów transmisji ramek, bazując na obserwacji czasu transmisji podczas procesu wymiany. Ten sposób detekcji podyktowany jest tym, że dane, we wszystkich przypadkach, mają ograniczony czas „życia" (ważności). Uproszczony sposób wymiany informacji przedstawia rys. 27. Dzięki tak przyjętemu sposobowi wymiany danych sieć zapewnia każdemu abonentowi periodyczne ich „odświeżanie". Dane te umieszczone są na liście arbitra sieci. W tym sensie sieć, z zastosowanym protokołem PDK, kreuje dla

rozproszonej bazy danych system jej aktualizacji w czasie rzeczywistym. Inne wymagania prowadzą do określenia dostępu do sieci dla aktualizacji danych w czasie „martwym" pomiędzy transmisjami periodycznymi. Wtedy mamy do czynienia z transmisją aperiodyczną. Poza tym istnieją jeszcze wymiany typu transakcja komunikatów („MESSAGE"), o których będzie dalej mowa.



Rys. 27. Sposób wymiany informacji
Fig. 27. The method of data exchange

Najbardziej powszechną aplikacją, która uzyskała standard europejski **EN 50170** [46], sieci wykorzystującej protokół PDK jest sieć FIP (*ang.Factory Instrumentation Protocol*). I właśnie dla tej sieci zastanie dokonana analiza czasowa przepływu informacji. Zanim to nastąpi, należy podać postać podstawowej ramki oraz omówić główne rodzaje wymian. Nawiasem mówiąc, protokół ten znalazł swoje mało istotne rozszerzenie, firmowane przez międzynarodową organizację WorldFIP, co czasami prowadzi do pewnych nieporozumień, gdyż zasadnicza różnica polega na tym, iż w protokole WorldFIP warstwa liniowa „dokłada" trzy dodatkowe bity.

Sieć FIP realizuje obsługę komunikacji niższych warstw procesów sterowania i produkcji „dyskretnej". W obszarze realizacji tych warstw rozróżnia się różne rodzaje wymian informacji:

- informacje (dane) przychodzące z procesu technologicznego tworzone przez układy wejścia (inicjatory),
- rozkazy tworzone przez system i zwracane do procesu i realizowane przez układy wyjścia (układy wykonawcze),

- informacje wymieniane przez różnych abonentów sieci i przez różne systemy sterowania w kontekście zastosowanej aparatury, takiej jak: sterowniki swobodnie programowane, klasyczne lub „inteligentne" inicjatory i układy wykonawcze,
- informacje niezbędne dla monitorowania, sterowania i pomiarów.

Pierwsze trzy typy wymian dotyczą komunikacji poziomej w sieci, gdzie następuje podział informacji od inicjatorów do układów wykonawczych czy sterowników przemysłowych. Te rodzaje wymiany mają miejsce w trakcie aktualizacji rozproszonej bazy danych. Natomiast ostatni rodzaj wymian można określić jako pionowy ruch danych służący monitorowaniu i zdalnemu sterowaniu procesem technologicznym. Te wymiany są mniej krytyczne z punktu widzenia wymagań czasowych. Sieć FIP stara się jednocześnie zaspokoić wymagania wynikające z „pionowej" wymiany danych, ze szczególnym uwzględnieniem periodycznej wymiany danych, obwarowanej czasem krytycznym i aperiodycznej komunikacji z wyższymi warstwami oprogramowania.

Najogólniej rzecz biorąc, w sieci występują dwa typy ramek:

- ramki transmitowane przez arbitra,
- ramki transmitowane od „producenta" wartości zmiennej.

Postacie tych ramek przedstawia rys. 28.

Ramki transmitowane przez arbitra.

Preambuła	Kod	IDENTYFIKATOR	Kontrola	Postambuła

Ramki transmitowane przez „producenta".

Preambuła	Kod	BUFOR	Kontrola	Postambuła

Rys. 28. Podstawowe typy ramek
Fig. 28. The basic types of frames

W tym miejscu należy przypomnieć, że takie nazwy, jak:

- warstwa fizyczna,
- warstwa łączenia,
- warstwa aplikacji,

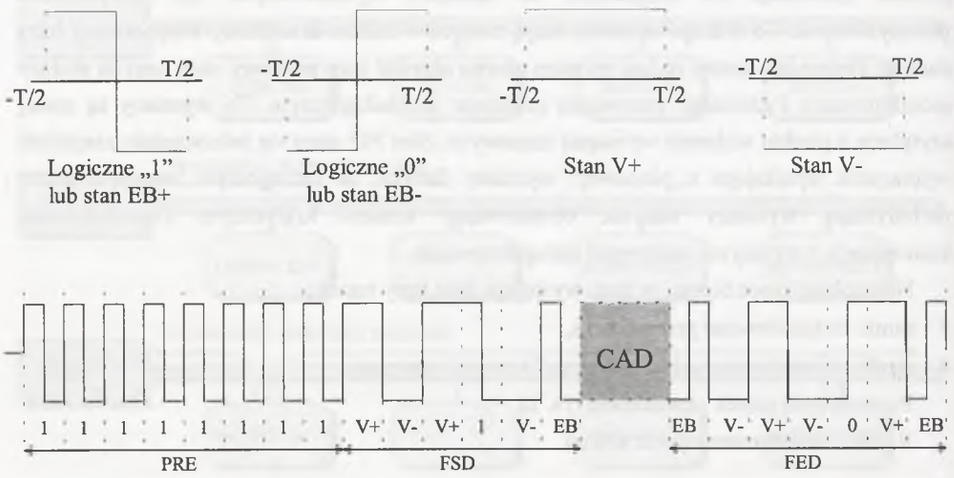
są tożsame znaczeniowo z identycznymi nazwami używanymi w siedmiowarstwowym modelu sieci OSI/ISO. Była o tym mowa w rozdziale 3.

Kodowanie ramki dokonywane jest z użyciem kodu Manchester 2.

Preambuła składa się z dwóch części [53]:

- serii 8 „jedynek" służących do synchronizacji odbiornika z zegarem nadajnika (PRE),

- sekwencji stanów V+, V-, V+, 1, V- zakończonych stanem EB+, informującej warstwę łączenia o początku sekwencji danych (FSD),
- postambuła natomiast jest sekwencją stanów EB-, V-, V+, V-, 0, V+, EB- (FED), służącą warstwie łączenia do identyfikacji końca pola danych. Tak więc warstwa fizyczna „dokłada" 21 bitów do ramki transmisji (rys. 29) (w standardzie WorldFIP 24 bity).



Rys. 29. Sekwencje kontrolne ramek
Fig. 29. The sequences of control frames

Pola oznaczone na rysunku 28 jako kod i identyfikator w ramce transmitowanej przez arbitra oraz pola oznaczone jako bufor i kod w ramce transmitowanej przez abonenta są traktowane jako blok danych. Warstwa łączenia uzupełnia ramkę o kod funkcji (1 bajt) i sumę kontrolną CRC (2 bajty), a warstwa aplikacji dołącza do ramki identyfikator (nazwę zmiennej) kodowany na dwóch bajtach.

Biorąc pod uwagę dwa podstawowe typy ramek oraz tryby transmisji (periodyczna, aperiodyczna, typu „MESSAGE"), można szczegółowiej podzielić ramki na:

- ramki identyfikatora,
- ramki odpowiedzi z danymi,
- ramki żądań,
- ramki komunikatów,
- ramki potwierdzeń.

Ich szczegółowa budowa będzie opisana w rozdziale 6.7.

W sieci wykorzystującej protokół PDK (w tym w sieci FIP) występują dwa rodzaje usług oferowanych przez warstwę łączenia i aplikacji.

Są to:

- usługi periodyczne (aperiodyczne),
- usługi transmisji komunikatów (MESSAGE).

Usługi periodyczne (aperiodyczne) dla warstwy aplikacji noszą miano MPS, zaś dla warstwy łączenia (ang. *"link layer"*) AP. Usługi transmisji komunikatów noszą odpowiednio miano SUB – MS i MSG. Periodyczne i aperiodyczne usługi zostały specjalnie dobrane do wymagań dystrybucji danych w czasie rzeczywistym.

Służą one do:

- periodycznej aktualizacji rozproszonej pomiędzy jednostkami sprzętowymi systemu sterowania, bazy danych,
- aktualizacji na żądanie, wartości elementów bazy danych (dodatkowa, na żądanie, aktualizacja wartości zmiennych),
- obliczania statusu czasowej spójności, związanego z bazą danych (status ten określa „świeżość" danych),
- obliczania statusu zgodności związanego z zapamiętywaniem danych do bazy danych.

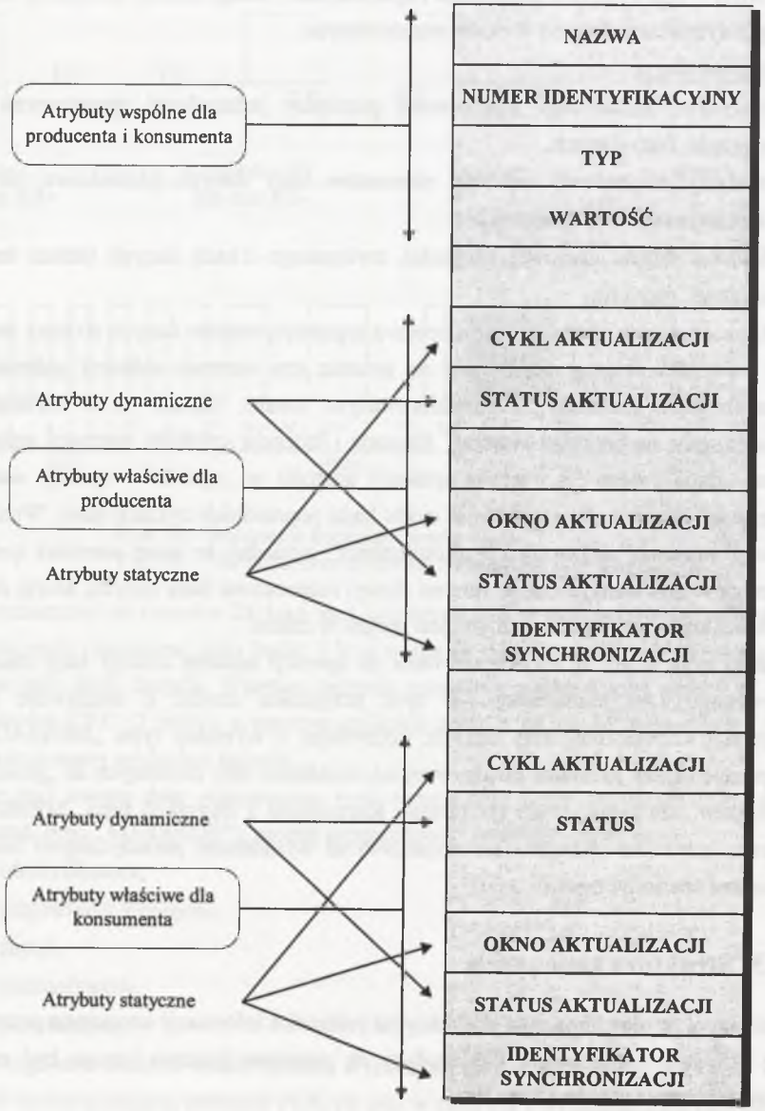
Status zgodności dotyczy odpowiedzi na pytanie, czy warstwa aplikacji pobrała wartości zmiennych bądź zmiennej w zadeklarowanym czasie. Chodzi tu o wyeliminowanie przypadku, gdy na przykład warstwy fizyczna i łączenia uzyskały wartości zmiennych ze statusem „dane świeże", a warstwa aplikacji wartości te „zgubiła" i operuje wartościami, które zostały odebrane w poprzednim cyklu bądź poprzednich cyklach sieci. Wtedy nie jest spełniony warunek, aby wszyscy „Konsumenci" posiadali te same wartości tych samych zmiennych w tym samym czasie. Innymi słowy, rozproszona baza danych, której elementami są wartości krążących zmiennych, nie jest spójna w czasie.

Usługi typu MMS są zwykle używane do operacji żądania obsługi bazy danych czasu rzeczywistego i jej konfiguracji. W tym przypadku chodzi o możliwość przesłania konfiguracji rozproszonej bazy danych, korzystając z wymiany typu „MESSAGE". Przez konfigurację należy rozumieć między innymi, określenie listy zmiennych do „produkcji" i do „konsumpcji" dla każdej stacji (abonenta). Korzystanie z transakcji typu „MESSAGE" jest korzystne, gdyż nie obciąża sieci dodatkowymi wymianami periodycznymi lub wieloma wymianami aperiodycznymi.

6.3. Struktura zmiennych

Zmienna jest określona jako abstrakcyjna jednostka informacji oznaczona przez unikalną nazwę w systemie sterowania. Jej atrybuty są związane z nazwą i mogą być następujące (rys. 30):

- atrybuty typu podstawowego, takie jak numer identyfikacyjny lub typ zmiennej, przy czym typ zmiennej może być prosty (liczby całkowite, zmiennoprzecinkowe, zmienne boolowskie) lub złożony (tablice, struktury),
- atrybuty statyczne, definiowane podczas konfiguracji sieci i określające lokalne mechanizmy związane ze zmienną (synchronizacja, „producent", „konsument"),



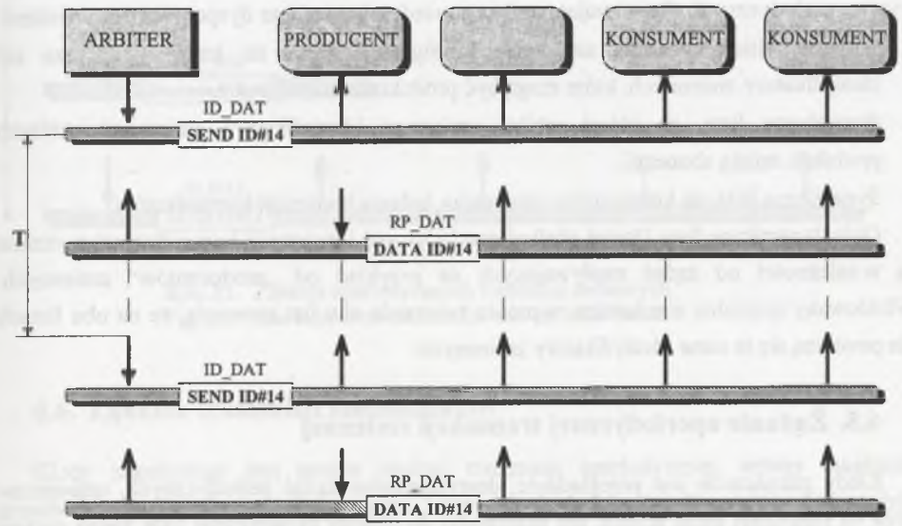
Rys. 30. Struktura danych transmitowanych w sieci
Fig. 30. The structure of transmitting data

- atrybuty dynamiczne, powracające od abonenta wraz z wartością zmiennej i określające jej dynamiczny stan (np. status aktualizacji-zmienna „ważna" - "zmienna nieważna").

Adresacja zmiennych jest realizowana przez symboliczną nazwę globalną, co pozwala na unikalną identyfikację lokalnej kopii zmiennej. Adresacja jest globalna dla całego systemu wewnątrz obszaru roboczego mechanizmu dystrybucyjnego. Baza danych jest rozproszona na wszystkie stacje abonenckie (lub urządzenia) bezpośrednio podłączone do stacji, a ta sama nazwa zmiennej jest unikalna w całym systemie.

6.4. Transakcja wymiany periodycznej

Na rys. 31 przedstawiono realizację wymiany periodycznej.



T – okres wymiany periodycznej

Rys. 31. Transakcja periodyczna
Fig. 31. The periodic transaction

Po transmisji ramki identyfikującej zmienną ID_DAT (identyfikator # 14) arbiter ustawia czas oczekiwania i czeka na ramkę odpowiedzi od tego abonenta, który jest „producentem" wartości zmiennej #14.

Jeżeli odebrana przez arbitra ramka jest ramką odpowiedzi RP_DAT, to arbiter przechodzi do transmisji kolejnego z listy identyfikatora zmiennej. Innymi słowy, rozpoczyna poszukiwanie „producenta" kolejnej zmiennej z listy.

Jeżeli natomiast odebrana ramka jest odpowiedzią skompletowaną, przez tryb żądania transferu aperiodycznego (ramka typu RP_DAT_RQi), arbiter zapamiętuje wyprodukowany identyfikator w kolejce żądań transakcji aperiodycznych z priorytetem żądania RQi i przechodzi do obsługi kolejnych identyfikatorów. Indeks „*i*" w polu RQi oznacza tryb pilności obsługi. I tak, jeśli:

i = 1, oznacza to żądanie obsługi pilnej,

i = 2, oznacza to żądanie obsługi standardowej.

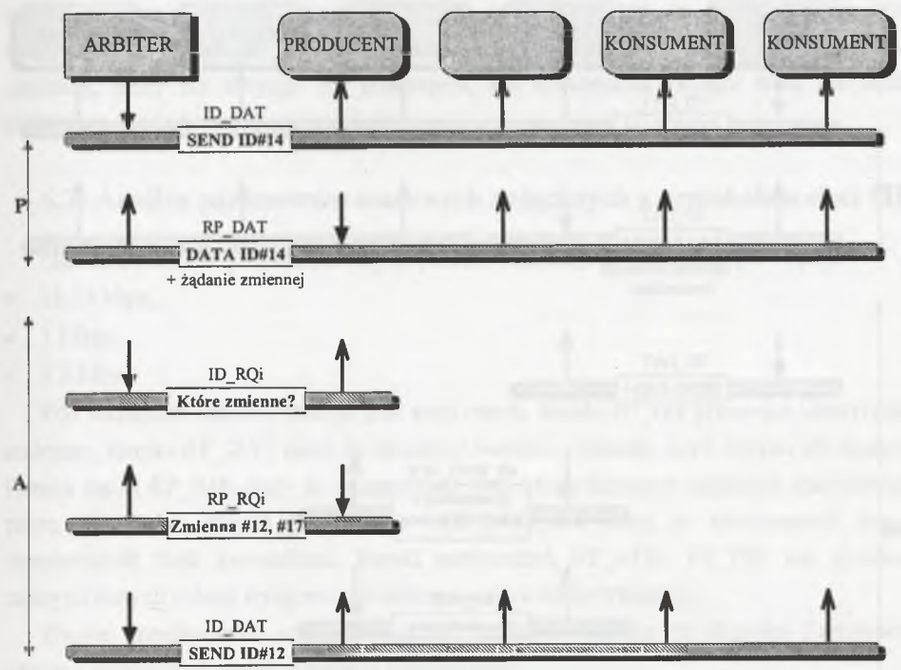
Jeżeli wreszcie odebrana przez arbitra ramka odpowiedzi została skompletowana przez żądanie transferu aperiodycznego (przez umieszczenie w ramce odpowiedzi kodu RP_DAT_RQi_MSG, co oznacza, że abonent żąda dostępu do sieci celem emisji komunikatu), to arbiter zapamiętuje identyfikator tego żądania w odpowiednim zbiorze i przechodzi, jak poprzednio, do obsługi kolejnych identyfikatorów umieszczonych na liście wymian periodycznych. Reasumując, można powiedzieć, że arbiter dysponuje trzema listami:

- statyczną listą, tworzoną na etapie konfiguracji sieci, na której umieszcza się identyfikatory zmiennych, które mogą być produkowane cyklicznie,
- dynamiczną listą, na której arbiter umieszcza identyfikatory zmiennych, których produkcji żądają abonenci,
- dynamiczną listą, na której arbiter umieszcza żądania transmisji komunikatów.

Obie dynamiczne listy (żądań obsługi aperiodycznej i transmisji komunikatów) tworzone są w zależności od żądań napływających na przykład od „producentów" zmiennych. Wbudowany specjalny mechanizm w proces tworzenia obu list zapewnia, że na obu listach nie powtórzą się te same identyfikatory zmiennych.

6.5. Żądanie aperiodycznej transakcji zmiennej

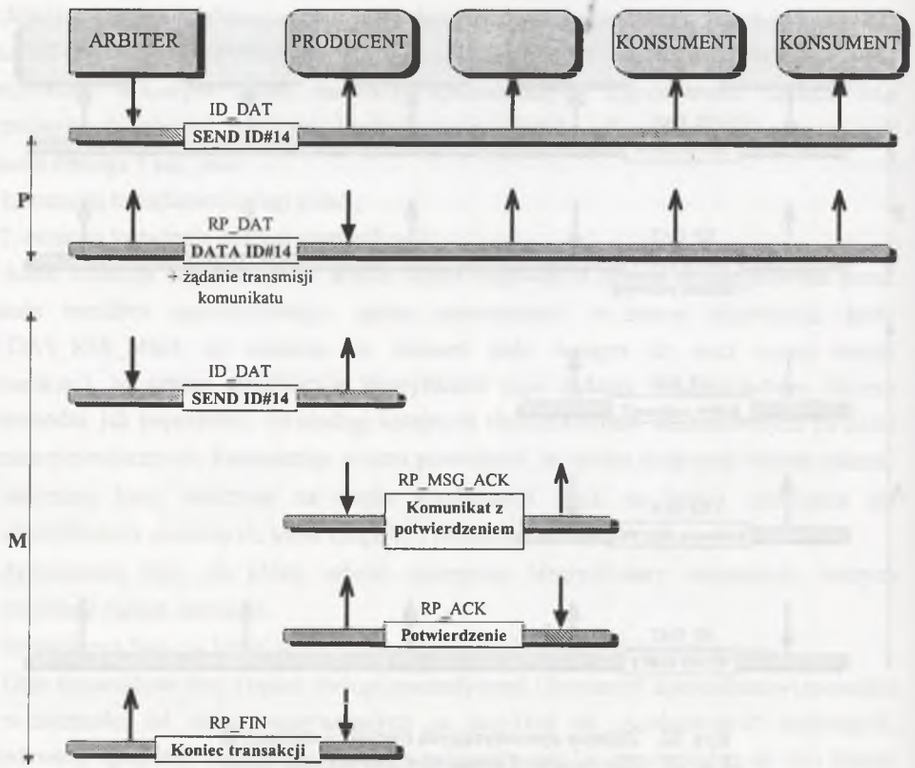
Kiedy zakończone jest przeglądanie identyfikatorów żądań periodycznych, umieszczonych na statycznej liście arbitra, ten przechodzi do obsługi dynamicznej listy żądań aperiodycznych, którą stworzył podczas realizacji wymian periodycznych. Transmituje pierwszą ramkę identyfikatora żądań ID_RQ1 z kolejki pilnych transferów aperiodycznych (rys. 32). Jeżeli kolejka jest pusta, arbiter ustawia czas oczekiwania i oczekuje na ramkę odpowiedzi żądania. „Producent", który rozpoznał swój identyfikator, transmituje w ramce typu RP_RQi wartości zmiennych, których żądanie dotyczyło.



Rys. 32. Żądanie aperiodycznych transakcji zmiennych
Fig. 32. The request of transaction non-periodic values

6.6. Żądanie transmisji komunikatów

Kiedy zakończony jest proces obsługi transmisji aperiodycznej, arbiter magistrali przechodzi do interpretacji listy żądań transmisji komunikatów (rys. 33). Transmituje pierwszy identyfikator komunikatu (ID_MSG) z kolejki komunikatów i jeśli nie jest ona pusta, ustawia czas oczekiwania i oczekuje na ramkę typu „end of transaction". Abonent, który rozpoznał swój identyfikator, transmituje komunikat z listy w formie ramki RP_MSG_ACK. Jeżeli uzyska ramkę potwierdzenia RP_ACK lub jeśli nie uzyska żadnej odpowiedzi po próbie kilku retransmisji, wysyła do arbitra magistrali ramkę typu RP_FIN.



Rys. 33. Żądanie transmisji komunikatów
Fig. 33. The request of messages transmission

W omawianych trzech typach transmisji (rozdz. 6.4, 6.5, 6.6) mogą zaistnieć również inne przypadki:

- jeżeli odebrana przez arbitra ramka jest innego typu niż omawiane, to arbiter wykrywa błąd i przechodzi do obsługi kolejnego identyfikatora,
- jeżeli odebrana przez arbitra ramka zawiera niepoprawne pole kontrolne (FCS- na przykład błędny kod funkcji), arbiter wykrywa błąd transmisji i przechodzi do obsługi kolejnego identyfikatora,
- jeżeli czas oczekiwania na odpowiedź został przekroczony, arbiter sygnalizuje „zagubienie" ramki odpowiedzi i przechodzi do obsługi kolejnego identyfikatora.

Kolejki mają wbudowany mechanizm uniemożliwiający duplikowanie żądań, po których realizacji nie otrzymano odpowiedzi (potwierdzenia). Podczas trwania „okna" synchronizacji

arbiter magistrali określa czas trwania pętli oczekiwania aż do końca trwania cyklu elementarnego. Pętla ta jest realizowana przez powtarzanie transmisji identyfikatora zmiennej, który nie dotyczy ani producenta, ani konsumenta i dzięki temu nie będzie transmitowana żadna ramka odpowiedzi.

6.7. Analiza parametrów czasowych związanych z protokołem sieci FIP

Standard sieci FIP [53] określa trzy prędkości transmisji na magistrali (*Ptr [bps]*:

- 31.25 kbps,
- 1 Mbps,
- 2.5 Mbps.

Pod względem budowy istnieje pięć grup ramek. Ramki ID_xxx przenoszą identyfikator zmiennej. Ramka RP_DAT służy do transmisji wartości zmiennej, czyli danych użytecznych. Ramka żądań RP_RQi służy do przenoszenia listy identyfikatorów żądanych aperiodycznie przez abonentów. Ramki komunikatów RP_MSG_xxx służą do przenoszenia danych stanowiących treść komunikatu. Ramki potwierdzeń RP_ACK, RP_FIN nie przenoszą żadnych danych i służą wyłącznie do informowania o stanie transakcji.

Poniżej przedstawiono schematy budowy ramek i wyliczenia ich długości. Zastosowane nazwy i symbole pól ramek oznaczają odpowiednio:

DTR:	preambuła,
CP:	pole kodu funkcji,
ID:	pole identyfikatora zmiennej,
FCS:	pole sumy kontrolnej,
PDU:	pole typu PDU,
Długość PDU:	pole liczby bajtów danych,
Typ PDU:	pole typu danych,
Dane:	pole danych stanowiących wartość zmiennej,
Status	pole statusu produkcji,
Komunikat:	pole danych stanowiących komunikat,
Przeznaczenie, Źródło:	pole adresowe abonentów,
FTR:	postambuła,

natomiast litera *d* z indeksem oznacza wielkość danego pola w bitach.

ID_DAT; ID_MSG; ID_xxx_RQ1; ID_xxx_RQ2

DTR	CP	ID	FCS	FTR
$d_{dtr}=14$	$d_{cp}=8$	$d_{id}=16$	$d_{fcs}=16$	$d_{ftr}=7$

Długość ramek **ID_xxx** oraz **ID_xxx_RQi** wynosi $d_{ID} = 61$ bitów i jest stała.

RP_DAT

DTR	CP	PDU	Dane	FCS	FTR
$d_{dtr}=14$	$d_{cp}=8$	$d_{PDU}=16$	$d_d=(1..126)*8$	$d_{fcs}=16$	$d_{ftr}=7$

Długość ramki RP_DAT wynosi $d_{RP_DAT} = 61 + n*8$ bajtów, gdzie $1 \le n \le 126$, i zależy od wielkości zmiennej. Pole PDU zawiera dwa bajty, na których zawarty jest typ danych oraz długość zmiennej. Długość zmiennej odnosi się do rzeczywistej wielkości zmiennej podanej w bajtach, wliczając ewentualny bajt statusu „produkcji" wartości zmiennej.

RP_DAT – pola PDU i danych

Typ PDU	Długość PDU	Dane	Status
$d_{PDU_Typ}=8$	$d_{PDU_Len}=8$	$d_d=(1..125)*8$	$d_{REF}=8$

Status „produkcji" wartości zmiennej jest przesyłany na ostatnim bajcie danych kosztem ilości tych danych. Oczywiście wystąpienie bajtu statusu jest opcjonalne.

RP_RQ1; RP_RQ2

DTR	CP	Dane	FCS	FTR
$d_{dtr}=14$	$d_{cp}=8$	$d_d=(1..64)*16$	$D_{fcs}=16$	$d_{ftr}=7$

Długość ramek RP_RQi wynosi $d_{RP_RQi} = 45 + n*16$ bajtów, gdzie $1 \le n \le 64$, i zależy od liczby żądanych zmiennych.

RP_MSG_ACK; RP_MSG_NOACK

DTR	CP	Przeznaczenie	Zródlo	Komunikat	FCS	FTR
$d_{dtr} = 14$	$d_{cp} = 8$	$d_{adr} = 24$	$d_{adr} = 24$	$D_d=(1..256)*8$	$d_{fcs}=16$	$D_{ftr}=7$

Długość ramek **RP_MSG_xxx** wynosi $d_{RP_MSG} = 93 + n*8$ bajtów, gdzie $1 \le n \le 256$, i jest zależna od długości transmitowanego komunikatu. Pola adresowe „Przeznaczenie" i „Źródło" identyfikują nadawcę i odbiorcę komunikatu. Identyfikacja odbywa się przez podanie numeru segmentu sieci (1 bajt) oraz identyfikatora zmiennej produkowanej przez danego abonenta i wyróżnionej do obsługi komunikatów (2 bajty).

RP_ACK; RP_FIP

DTR	CP	FCS	FTR
$d_{dtr}=14$	$d_{cp}=8$	$d_{fcs}=16$	$d_{ftr}=7$

Długość ramek potwierdzeń wynosi $d_{RP_ACK} = 45$ *bitów*, i jest stała.

Istotne parametry czasowe sieci to:

- Czas oczekiwania (ang. *link time-out*) $T_{tout} = const$
- Czas milczenia sieci (ang. *turnaround time-out*) $T_{tr} = const$

Dla sieci FIP:

$$10\,Tmac \le Ttr \le 70\,Tmac, \tag{51}$$

gdzie *Tmac* jest czasem potrzebnym na transmisję jednego bitu.

Sieć FIP pracuje zgodnie z tzw. makrocyklami arbitra magistrali. Na ***makrocykl*** pracy składa się jeden bądź wiele cykli elementarnych ***(mikrocykli)***. Te z kolei składają się z czterech okien czasowych, w których realizowane są wszystkie usługi warstwy łączenia. Czas potrzebny na zrealizowanie pełnego cyklu elementarnego wyniesie:

$$T_{cycle} = \sum_{i=1}^{4} T_i, \tag{52}$$

gdzie T_i jest czasem trwania poszczególnych okien.

6.8. Analiza czasowa pracy protokołu

Po zebraniu kluczowych informacji, niezbędnych dla dalszych rozważań, zajmiemy się czasową analizą pracy protokołu dla wszystkich rodzajów transakcji oraz przedstawimy matematyczny zapis czasu realizacji transakcji.

Poniższe rozważania dotyczą mikrocykli zsynchronizowanych, czyli takich, w których występuje okno synchronizacji. Poczyniono takie założenie, gdyż dla mikrocykli niezsynchronizowanych periodyczność wymian można określić tylko dla przedziału czasu jednostronnie domkniętego, czyli dla przypadków pesymistycznych. Rzeczywisty okres odświeżania zmiennych nie będzie wtedy znany.

W niniejszym rozdziale używane będą symbole i skróty, których znaczenie objaśnione jest poniżej.

L_{p1}, L_{p2}, L_{p3} — długość listy zmiennych i komunikatów periodycznych,

La1, La2 — *długość listy zmiennych aperiodycznych,*

Lm1, Lm2 — *długość listy komunikatów (ACK / NOACK),*

TID_DAT — *czas transmisji ramki identyfikatora,*

TID= TID_prod +TID_DAT — *czas produkcji oraz transmisji ramki identyfikatora,*

TRP_DAT = Txxx_min ..Txxx_max — *czas transmisji ramki RP_DAT,*

TRP_RQi = Txxx_min ..Txxx_max — *czas transmisji ramki RP_DAT,*

TRP_MSG_NOACK = Txxx_min ..Txxx_max — *czas produkcji ramki RP_MSG_NOACK,*

TRP_MSG_ACK = Txxx_min ..Txxx_max — *czas produkcji ramki RP_MSG_ACK,*

TRP_ACK — *czas produkcji ramki RP_ACK,*

TRP_FIN — *czas produkcji ramki RP_FIN,*

Tresp — *czas oczekiwania na odpowiedź abonenta,*

Ttout = Tresp_max — *maksymalny czas oczekiwania (timeout) na odpowiedź abonenta,*

Tresp_Ab — *czas oczekiwania na potwierdzenie komunikatu,*

Ttout_Ab= Tresp_Ab_max — *czas oczekiwania (timeout) na potwierdzenie komunikatu,*
Qrep = 0..3 — *liczba powtórzeń komunikatów,*
QRP_RQi — *jest liczbą żądanych zmiennych przez danego abonenta (wartość dynamiczna),*
TtoutAb — *czas oczekiwania na potwierdzenie komunikatu,*
Tread — *czas odczytu bufora komunikacyjnego (link layer),*
Tproc — *czas przetwarzania informacji (link & application layer),*
Twrite — *czas zapisu bufora komunikacyjnego (link layer),*
TArb_proc — *jest czasem obsługi ramki RP_RQi przez arbitra (ang. "link & application layer"),*
Tas = Tread + Tproc + Twrite — *czas obsługi ramki przez abonenta,*
Txxx — *ramka danego typu, którego tyczy się opis,*
Tmac — *czas transmisji jednego bitu,*
Ttr — *czas milczenia sieci.*

6.8.1. Analiza statyczna

Analiza statyczna odnosi się do sytuacji, kiedy w sieci nie ma dialogu, czyli abonenci nie odpowiadają. Jest to sytuacja, gdy pracuje sam arbiter, a sieć nie jest obciążana transmisją danych. Pracę samego arbitra należy potraktować jako założenie, służące do określenia czasu trwania cyklu sieci, z wykorzystaniem zasady superpozycji (analiza pracy statycznej odbywa się niezależnie od analizy dynamicznej). Arbiter może bowiem pracować bez jakiegokolwiek ruchu w sieci, ale wtedy na pewno nie będą miały miejsca żądania transakcji aperiodycznych i transakcji komunikatów.

Użyte poniżej oznaczenia należy interpretować zgodnie z ogólnym opisem:

- T_{xxx} – oznacza czas,
- L_{xxx} – oznacza liczbę zmiennych obsługiwanych w danej transakcji.

Rozpatrzmy czynności i czasy z nimi związane w przypadku realizacji wymian periodycznych:

arbiter rozsyła ramki identyfikatorów z listy periodycznej zmiennych

$$T_{p1} = (T_{ID} + T_{tr} + T_{tout})L_p,$$

arbiter rozsyła ramki identyfikatorów z listy periodycznej komunikatów z potwierdzeniem

$$T_{p2} = (T_{ID} + T_{tr} + T_{tout})L_{mp1},$$

arbiter rozsyła ramki identyfikatorów z listy periodycznej komunikatów bez potwierdzenia

$$T_{p3} = (T_{ID} + T_{tr} + T_{toutp})L_{mp2},$$

Ogólnie dla transakcji periodycznych otrzymamy:

$$T_p = T_{p1} + T_{p2} + T_{p3}$$

Postąpmy tak samo dla transakcji wymian aperiodycznych wartości zmiennych:

arbiter rozsyła ramki identyfikatorów z listy aperiodycznej pilnej

$$T_{a1} = (T_{ID} + T_{tr} + T_{tout})L_{a1},$$

arbiter rozsyła ramki identyfikatorów z listy aperiodycznej normalnej

$$T_{a2} = (T_{ID} + T_{tr} + T_{tout})L_{a2}.$$

Ogólnie zatem dla transakcji aperiodycznych przesyłów zmiennych otrzymamy:

$$T_a = T_{a1} + T_{a2}.$$

Dla transakcji transmisji identyfikatorów z listy komunikatów:

– arbiter rozsyła ramki identyfikatorów z listy komunikatów bez potwierdzenia

$$T_{m1} = (T_{ID} + T_{tr} + T_{tout})L_{m1},$$

arbiter rozsyła ramki identyfikatorów z listy komunikatów z potwierdzeniem

$$T_{m2} = (T_{ID} + T_{tr} + T_{tout})L_{m2}.$$

Ogólnie dla aperiodycznych transakcji komunikatów otrzymamy:

$$T_m = T_{m1} + T_{m2}.$$

Pozostały transmisje ramek synchronizacyjnych:

arbiter rozsyła ramki synchronizacyjne

$$T_s = nT_{ID},$$

Zatem rzeczywisty czas realizacji cyklu elementarnego wynosi:

$$T_{cycle} = T_p + T_a + T_m + T_s \tag{53}$$

czyli

$$\begin{aligned} T_{cycle} &= (T_{ID} + T_{tr} + T_{tout})L_p + (T_{ID} + T_{tr} + T_{tout})L_{mp1} + (T_{ID} + T_{tr} + T_{toutp})L_{mp2} + \\ &+ (T_{ID} + T_{tr} + T_{tout})L_{a1} + (T_{ID} + T_{tr} + T_{tout})L_{a2} + (T_{ID} + T_{tr} + T_{tout})L_{m1} + \\ &+ (T_{ID} + T_{tr} + T_{tout})L_{m2} + nT_{ID} \end{aligned} \tag{54}$$

Ponieważ okno synchronizacji zapewnia nam jedynie stały czas trwania cyklu elementarnego, dlatego jego uwzględnianie nie ma sensu z punktu widzenia obliczeń czasu trwania cyklu sieci podczas transmisji danych użytkowych. W zależności bowiem od istnienia wymian aperiodycznych będzie się zmieniał czas transakcji „martwych". Tak więc po przekształceniu wzoru (*53*) otrzymamy:.

$$T_{cycle}' = T_p + T_a + T_m, \tag{55}$$

$$\begin{aligned} T_{cycle}' &= T_{ID}(L_p + L_{mp1} + L_{mp2} + L_{a1} + L_{a2} + L_{m1} + L_{m2}) + \\ &+ T_{tr}(L_p + L_{mp1} + L_{mp2} + L_{a1} + L_{a2} + L_{m1} + L_{m2}) + \end{aligned}$$

$$+ T_{tout}(L_p + L_{mp1} + L_{mp2} + L_{a1} + L_{a2} + L_{m1} + L_{m2}).$$

Parametry L_p, L_{mp1}, L_{mp2} oznaczające liczbę zmiennych obsługiwanych w transakcji periodycznej są stałe dla danej sieci. Pamiętając o przyjętej zasadzie superpozycji, pozostałe parametry typu L_{xxx} dla sytuacji milczenia sieci można przyjąć jako zerowe, gdyż skoro abonenci milczą, to nie generują żądań. Otrzymamy zatem:

$$T_{cycle}' = T_{ID}(L_p + L_{mp1} + L_{mp2}) + T_{tr}(L_p + L_{mp1} + L_{mp2}) + \\ + T_{tout}(L_p + L_{mp1} + L_{mp2}) \quad (56)$$

Na podstawie powyższej zależności można wyliczyć, ile czasu trwa rozesłanie informacji arbitrażu w każdym cyklu elementarnym. Można również określić wartość parametru n dla okna synchronizacji przy założeniu wartości czasu trwania cyklu elementarnego.

Przykładowo, dla dziesięciu zmiennych obsługiwanych periodycznie, czasie trwania mikrocyklu równym 5 [ms], prędkości transmisji 1 [Mb/s] i T_{tout}=3150 [μs], czas transmisji informacji nadzorczych (narzut) wynosi:

$$T_{cycle} = 5000 = T_{ID}(10) + T_{tr}(10) + T_{tout}(10) + T_s$$

$$T_{cycle} = 5000 = 610 + 400 + 3150 + T_s$$

$$T_{cycle}' = 4160\ [\mu s]$$

Załóżmy hipotetyczną sytuację, że arbiter nie czeka na odpowiedź, czyli parametr T_{tout}=0[ms]. Otrzymujemy wówczas rzeczywisty czas transmisji informacji nadzorczych, pochodzących od arbitra. Wynosi on:

$$T_{cycle}'' = 1010\ [\mu s],$$

co stanowi około 20% czasu makrocyklu. Należy zauważyć, iż ze wzrostem czasu trwania mikrocyklu czas ten pozostaje bez zmian i dla mikrocyklu 100 [ms] będzie stanowił około 1% czasu trwania mikrocyklu.

6.8.2. Analiza dynamiczna

Analiza dynamiczna odnosi się do stanu rzeczywistej pracy sieci, czyli abonenci odpowiadają. Jest to sytuacja, gdy sieć jest obciążana transmisją danych. Należy teraz zauważyć, iż rzeczywisty czas odpowiedzi abonenta może przyjmować dowolne wartości z przedziału $(0,T_{tout}\rangle$ i jest uzależniony od wielu nieprzewidywalnych czynników, takich jak choćby opóźnienia w wymianie informacji pomiędzy warstwą łączenia a warstwą aplikacji. Prześledźmy teraz zależności czasowe towarzyszące kolejnym typom wymian, mającym miejsce, od wymian periodycznych począwszy, a na transmisji okna synchronizacji skończywszy. Konwencja oznaczeń jest taka sama jak poprzednio.

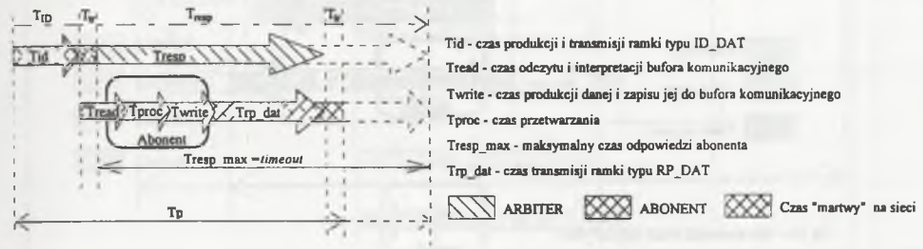
Transakcja periodyczna.

Arbiter rozsyła ramki identyfikatorów z listy periodycznej zmiennych

Czas trwania pojedynczej transakcji wynosi:

$$T_{pl} = (T_{ID} + 2T_{tr} + T_{resp})L_{pl} \quad (57)$$

Na rys. 34 przedstawiono symbolicznie składowe elementarne czasu trwania transakcji periodycznej zmiennej dla jednego identyfikatora.



Rys. 34. Schemat transakcji periodycznej
Fig. 34. The schema of periodic transaction

Wartość maksymalna czasu pojedynczej transakcji T_{pl}, przy danej liczbie identyfikatorów wynosi:

$$T_{pl_max} = (T_{ID} + 2T_{tr} + T_{tout})L_{pl}. \quad (58)$$

Zatem, aby transakcja była możliwa, musi być spełniony następujący warunek:

$$T_{pl} < T_{pl_max}$$

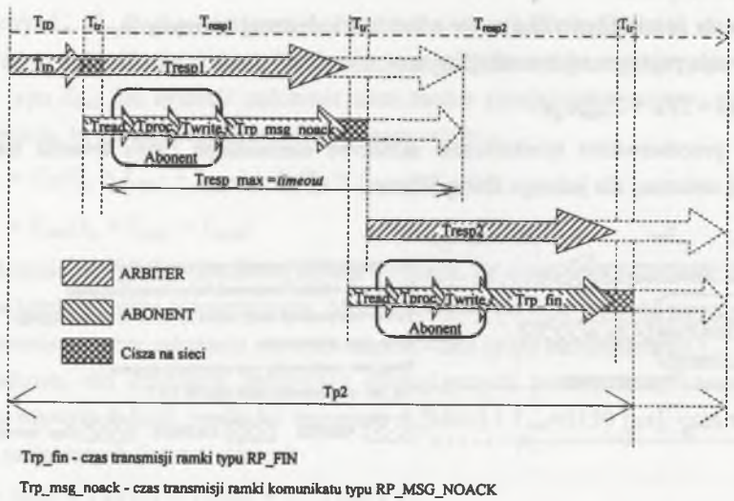
$$T_{pl_max} > (T_{ID} + 2T_{tr} + T_{tout})L_{pl}$$

Arbiter rozsyła ramki identyfikatorów z listy periodycznej komunikatów bez potwierdzenia

Czas trwania tej pojedynczej transakcji wynosi:

$$T_{p2} = (T_{ID} + 3T_{tr} + T_{resp1} + T_{resp2})L_{p2}. \quad (59)$$

Elementy składowe czasu trwania tej transakcji przedstawia rys. 35.



Rys. 35. Schemat transakcji komunikatów bez potwierdzenia
Fig. 35. The schema of message transaction without acknowledgement

Wartość maksymalna czasu transakcji wymiany komunikatów z listy periodycznej T_{p2}, przy danej liczbie komunikatów wynosi:

$$T_{p2_max} = (T_{ID} + 3T_{tr} + 2T_{tout})L_{p2}. \quad (60)$$

Zatem, aby transakcja była możliwa, musi być spełniony warunek:

$$T_{p2} < T_{p2_max}$$

$$T_{p2_max} > (T_{ID} + 3T_{tr} + 2T_{tout})L_{p2}$$

Arbiter rozsyła ramki identyfikatorów z listy periodycznej komunikatów z potwierdzeniem

Czas trwania tej pojedynczej transakcji wynosi:

$$T_{p3} = (T_{ID} + 2T_{tr} + T_{resp1} + Q_{rep}(T_{resp2} + T_{tr}) + T_{resp3} + T_{resp4} + 2T_{tr})L_{p3} \quad (61)$$

Składowe czasu trwania transakcji przedstawia rys. 36.

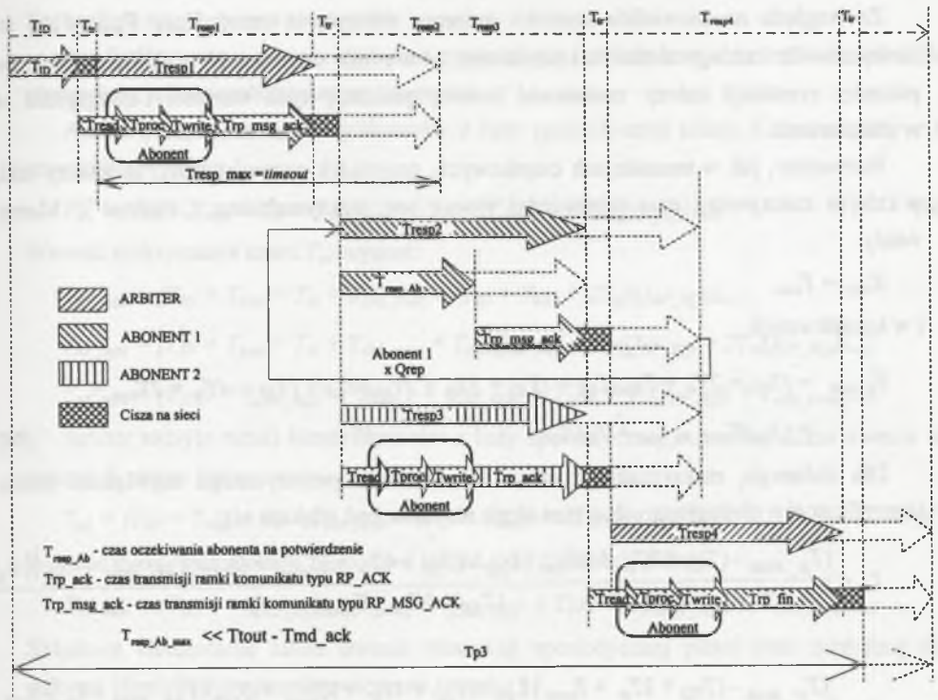
Wartość maksymalna czasu T_{p3} przy danej liczbie komunikatów wynosi:

$$T_{p3_max} = (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))L_{p3}. \quad (62)$$

Zatem, aby transakcja była możliwa, musi być spełniony warunek:

$$T_{p3} < T_{p3_max}$$

$$T_{p3_max} > (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))L_{p3}$$



Rys. 36. Schemat transakcji komunikatów z potwierdzeniem
Fig. 36. The schema of message transaction with acknowledgement

Podsumujmy w tym miejscu czas trwania transakcji periodycznej.

Czas trwania pojedynczej transakcji periodycznej wynosi:

$$T_p = T_{p1} + T_{p2} + T_{p3} \quad (63)$$

$$T_p = (T_{ID} + 2T_{tr} + T_{resp})L_{p1} + (T_{ID} + 3T_{tr} + T_{resp1} + T_{resp2})L_{p2} +$$

$$+ (T_{ID} + 2T_{tr} + T_{resp1}' + Q_{rep}(T_{resp2}' + T_{tr}) + T_{resp3} + T_{resp4} + 2T_{tr})L_{p2}$$

Przy czym czasy T_{respx} odpowiadają czasom odpowiedzi dla każdej wymiany, odpowiednio:

$$T_{resp} = T_{read} + T_{proc} + T_{write} + T_{RP_DAT}$$

$$T_{resp1} = T_{read} + T_{proc} + T_{write} + T_{RP_MSG_NOACK}$$

$$T_{resp2} = T_{read} + T_{proc} + T_{write} + T_{RP_FIN}$$

$$T_{resp1}' = T_{read} + T_{proc} + T_{write} + T_{RP_MSG_ACK}$$

$$T_{resp2}' = T_{resp_Ab} + T_{RP_MSG_ACK}$$

$$T_{resp3} = T_{read} + T_{proc} + T_{write} + T_{RP_ACK}$$

$$T_{resp4} = T_{resp2}$$

Ze względu na niewielkie wartości możemy założyć, iż czasy T_{read}, T_{proc}, T_{write} są identyczne dla każdego abonenta i ramek oraz że w sumie wynoszą 10 [μs]. Oczywiście na potrzeby symulacji należy zastosować losową generację tych wartości i uwzględnić je w obliczeniach.

Rozważmy, jak w transakcjach cząstkowych, przypadek pesymistyczny, to znaczy taki, w którym rzeczywisty czas odpowiedzi równy jest maksymalnemu *(„timeout")*. Mamy wtedy

$$T_{resp} = T_{tout}$$

i w konsekwencji

$$T_{p_max} = (T_{ID} + 2T_{tr} + T_{tout})L_{p1} + (T_{ID} + 3T_{tr} + 2T_{tout})L_{p2} + (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))L_{p3} \quad (64)$$

Dla zadanego, maksymalnego czasu trwania okna periodycznego największa liczba identyfikatorów obsługiwanych w tym oknie nie może być większa niż:

$$L_{p1} < \frac{(T_{p_max} - (T_{ID} + 3T_{tr} + 2T_{tout})L_{p2} - (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{Ab_max} + T_{tr}))L_{p3}}{(T_{ID} + 2T_{tr} + T_{tout})}$$

$$L_{p2} < \frac{(T_{p_max} - (T_{ID} + 2T_{tr} + T_{tout})L_{p1} - (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{Ab_max} + T_{tr}))L_{p3}}{(T_{ID} + 3T_{tr} + 2T_{tout})}$$

$$L_{p3} < \frac{(T_{p_max} - (T_{ID} + 3T_{tr} + 2T_{tout})L_{p2} - (T_{ID} + 2T_{tr} + T_{tout})L_{p1}}{(T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))} \quad (65)$$

Przykład:

Dla zdefiniowanego maksymalnego czasu trwania okna periodycznego $T_{p_max} = 100$ *[ms]*, prędkości transmisji $P_{tr} = 1$ *[Mbps]*, czasu oczekiwania $T_{tout} = 315$ *[μs]*, i czasu $T_{tr} = 42$ *[μs]* oraz liczbie periodycznych komunikatów równej zero, liczba identyfikatorów nie może przekraczać:

$$L_p < 100*10^{-3}/(61*10^{-6} + 84*10^{-6} + 315*10^{-6}),$$

$$L_p < 0{,}1/460*10^{-6},$$

$$L_p < 217{,}4,$$

$$L_p \leq 217.$$

Transakcja aperiodyczna zmiennych

Prześledźmy teraz procesy związane z transakcją aperiodyczną zmiennych. I znów, jak poprzednio, arbiter realizuje szereg transmisji.

Arbiter rozsyła ramki identyfikatorów z listy aperiodycznej pilnej. Czas związany z tą czynnością wynosi:

$$T_{a1} = (T_{ID} + T_{resp} + T_{tr} + T_{Arb_proc} + (T_{ID} + T_{resp1} + 2T_{tr})Q_{RP_RQi})L_{a1} \quad (66)$$

Wartość maksymalna czasu T_{a1} wynosi:

$$T_{a1_max} = (T_{ID} + T_{tout} + T_{tr} + T_{Arb_proc} + (T_{ID} + T_{tout} + 2T_{tr})Q_{RP_RQi})L_{a1},$$

$$T_{a1_max} = (T_{ID} + T_{tout} + T_{tr} + T_{Arb_proc} + T_{tout}Q_{RP_RQi} + T_{ID}Q_{RP_RQi} + 2T_{tr}Q_{RP_RQi})L_{a1},$$

$$T_{a1_max} = (T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + T_{tr}Q_{RP_RQi}) + T_{Arb_proc})L_{a1}.$$

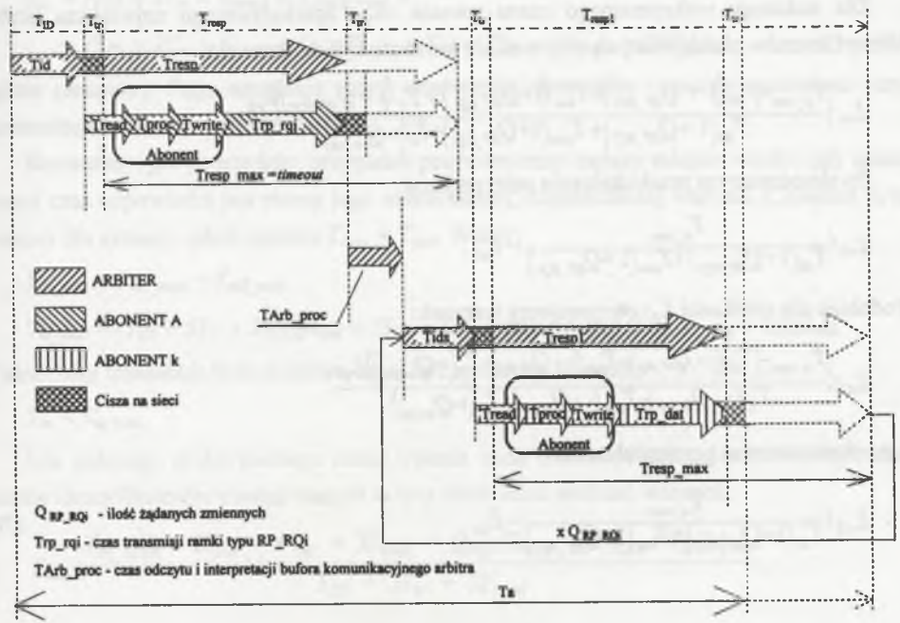
Arbiter rozsyła ramki identyfikatorów z listy aperiodycznej normalnej. Czas trwania tej transakcji równy jest:

$$T_{a2} = (T_{ID} + T_{resp} + T_{tr} + T_{Arb_proc} + (T_{ID} + T_{resp1} + 2T_{tr})Q_{RP_RQi})L_{a2}. \quad (67)$$

Wartość maksymalna czasu T_{a2} wynosi tak jak przy transakcji pilnej:

$$T_{a2_max} = (T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + T_{tr}Q_{RP_RQi}) + T_{Arb_proc})L_{a2}$$

Składowe elementarne czasu trwania transakcji aperiodycznej pilnej oraz normalnej dla jednego identyfikatora przedstawiono na rysunku 37.



Rys. 37. Schemat transakcji aperiodycznej
Fig. 37. The schema of non-periodic transaction

Podsumujmy w tym miejscu składowe czasu trwania transakcji aperiodycznej zmiennych:

$$T_a = T_{a1} + T_{a2},$$

$$T_a = (T_{ID} + T_{resp} + T_{tr} + T_{read1} + (T_{ID} + T_{resp1} + 2T_{tr})Q_{RP_RQi})L_{a1} +$$

$$+ (T_{ID} + T_{resp} + T_{tr} + T_{read1} + (T_{ID} + T_{resp1} + 2T_{tr})Q_{RP_RQi})L_{a2}$$

$$T_a = (T_{ID} + T_{resp} + T_{tr} + T_{read1} + (T_{ID} + T_{resp1} + 2T_{tr})Q_{RP_RQi})(L_{a1} + L_{a2}) \quad (68)$$

przy czym

$$T_{resp} = T_{read} + T_{proc} + T_{write} + T_{RP_Rqi},$$

$$T_{resp1} = T_{read} + T_{proc} + T_{write} + T_{RP_DAT}.$$

Rozważmy, jak poprzednio, przypadek pesymistyczny, to znaczy dla sytuacji, gdy ogólnie $T_{resp} = T_{tout}$. Mamy wtedy:

$$T_{a_max} = T_{a1_max} + T_{a2_max},$$

$$T_{a_max} = (T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + T_{tr}Q_{RP_RQi}))L_{a1}$$

$$+ (T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + T_{tr}Q_{RP_RQi}))L_{a2} \quad (69)$$

Zatem, aby transakcja była możliwa, musi być spełniony warunek:

$$T_a < T_{a_max}.$$

Dla zadanego maksymalnego czasu trwania okna aperiodycznego największa liczba identyfikatorów obsługiwanych w tym oknie nie może być większa niż

$$L_{a1} < \frac{T_{a_max} - (T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + Q_{RP_RQi}))L_{a2}}{T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + Q_{RP_RQi})}.$$

Po elementarnym przekształceniu otrzymujemy:

$$L_{a1} < \frac{T_{a_max}}{T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi})} - L_{a2}.$$

Podobnie dla wielkości L_{a2} otrzymujemy warunek:

$$L_{a2} < \frac{T_{a_max} - (T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + Q_{RP_RQi}))L_{a1}}{T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi}) + T_{tr}(1 + Q_{RP_RQi})},$$

a po elementarnym przekształceniu:

$$L_{a2} < \frac{T_{a_max}}{T_{ID}(1 + Q_{RP_RQi}) + T_{tout}(1 + Q_{RP_RQi})} - L_{a1}. \quad (70)$$

Transakcja aperiodycznego przesyłu komunikatów

Graficzne przedstawienie elementarnych czasów trwania transakcji komunikatów dla jednego identyfikatora oraz ich wyliczenia zaprezentowane są przy omawianiu transakcji periodycznej. I znów, jak poprzednio, prześledźmy czynności wykonywane przez arbitra w trakcie tego typu transakcji.

Arbiter rozsyła ramki identyfikatorów z listy komunikatów bez potwierdzenia. Czas związany z tą czynnością wynosi:

$$T_{m1} = (T_{ID} + 3T_{tr} + T_{resp1} + T_{resp2})L_{m2} \quad (71)$$

Wartość maksymalna czasu T_{m1} wynosi:

$$T_{m1_max} = (T_{ID} + 3T_{tr} + 2T_{tout})L_{m1}.$$

Arbiter rozsyła ramki identyfikatorów z listy komunikatów z potwierdzeniem. Czas realizacji jest równy:

$$T_{m2} = (T_{ID} + 2T_{tr} + T_{resp1} + Q_{rep}(T_{resp2} + T_{tr}) + T_{resp3} + T_{resp4} + 2T_{tr})L_{m2}. \quad (72)$$

Wartość maksymalna czasu T_{m2} wynosi:

$$T_{m2_max} = (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))L_{m2}.$$

Podsumujmy czasy cząstkowe transakcji transmisji komunikatów:

$$T_m = T_{m1} + T_{m2},$$

$$T_m = (T_{ID} + 3T_{tr} + T_{resp1} + T_{resp2})L_{m2} +$$

$$+ (T_{ID} + 2T_{tr} + T_{resp1} + Q_{rep}(T_{resp2} + T_{tr}) + T_{resp3} + T_{resp4} + 2T_{tr})L_{m2}, \quad (73)$$

gdzie parametry T_{respx} oznaczają czasy odpowiedzi abonentów i zostały omówione przy prezentacji transakcji periodycznej.

Rozważmy, jak poprzednio, przypadek pesymistyczny mający miejsce wtedy, gdy szacowany czas odpowiedzi jest równy jego maksymalnej, dopuszczalnej wartości *(„timeout")*, to znaczy dla sytuacji, gdzie ogólnie $T_{resp} = T_{tout}$. Wtedy:

$$T_{m_max} = T_{m1_max} + T_{m2_max}$$

$$T_{m_max} = (T_{ID} + 3T_{tr} + 2T_{tout})L_{m1} + (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))L_{m2}.$$

Zatem, aby transakcja była możliwa, musi być spełniony warunek:

$$T_m < T_{m_max}.$$

Dla zadanego maksymalnego czasu trwania okna transmisji komunikatów maksymalna liczba identyfikatorów obsługiwanych w tym oknie musi spełniać warunek:

$$L_{m1} < \frac{T_{m_max} - (T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))L_{m2}}{T_{ID} + 3T_{tr} + 2T_{tout}},$$

$$L_{m2} < \frac{T_{m_max} - (T_{ID} + 3T_{tr} + 2T_{tout})L_{m1}}{(T_{ID} + 4T_{tr} + 3T_{tout} + Q_{rep}(T_{resp_Ab_max} + T_{tr}))} \quad (74)$$

Arbiter pracuje w oknie synchronizacji

Gdy arbiter pracuje w oknie synchronizacji, w sieć są wysyłane ramki ID_DAT z nieobsługiwanym identyfikatorem. Czas związany z tym procesem wynosi:

$$T_s = n_S T_{ID}$$

Czas trwania okna synchronizacji zależy od czasu trwania okien pozostałych. Parametr n_S jest dobierany dynamicznie.

W tym miejscu należy podsumować wyniki analizy czasu trwania cyklu wymiany informacji w sieci. Przeanalizowano do tej pory wymiany wszystkich rodzajów i podano zależności czasowe pozwalające na oszacowanie nie tylko wartości czasu trwania poszczególnych wymian, ale również umożliwiające podanie bezpiecznych wartości istotnych elementów, takich jak choćby wielkości okien czasowych, czy maksymalnej liczby obsługiwanych wymian aperiodycznych, których liczba może się zmieniać dynamicznie w trakcie realizacji transakcji wymian w sieci. Umiejętność określenia maksymalnej liczby wymian aperiodycznych jest niezwykle ważna, bo dzięki temu można określić granice stosowalności sieci i przed przystąpieniem do jej aplikowania i konfiguracji można odpowiedzieć na pytanie, czy proponowana sieć spełni wymagania stawiane przez proces technologiczny. Ponadto, dzięki proponowanej analizie można pokusić się o stworzenie programowych narzędzi symulujących pracę sieci i ułatwiających jej stosowanie. Przed przystąpieniem do analizy pracy sieci z innego punktu widzenia, podsumujmy dotychczasowe rozważania, za pomocą ogólnych zależności, które określają czasy trwania cyklu sieci, a będące jednocześnie syntezą owych rozważań. Tak więc rzeczywisty czas trwania cyklu wynosi:

$$T_c = T_p + T_a + T_m + T_s, \quad (75)$$

a jego wartość maksymalna:

$$T_{c_max} = T_{p_max} + T_{a_max} + T_{m_max} + T_s \quad (76)$$

6.9. Analiza bitowa transakcji

Analizę czasów trwania transakcji można przeprowadzić z innego punktu widzenia, a mianowicie rozbijając wszystkie możliwe parametry na wielokrotności czasu transmisji pojedynczego bitu. Analizę taką przedstawiono poniżej.

6.9.1. Bitowa reprezentacja ramek

Na podstawie budowy ramek, przedstawionej wcześniej, można ich czasy transmisji zapisać jako iloczyn liczby bitów i czasu trwania transmisji jednego bitu.

ID_xxx	—	$61\ T_{mac}$	
RP_DAT	—	$61\ T_{mac} + 8\ n\ T_{mac}$	$n = 1..126$
RP_RQi	—	$45\ T_{mac} + 16\ m\ T_{mac}$	$m = 1..64$
RP_MSG_xxx	—	$93\ T_{mac} + 8\ k\ T_{mac}$	$k = 1..256$
RP_ACK	—	$45\ T_{mac}$	

6.9.2. Transakcja periodyczna

Prześledźmy kolejno, patrząc z punktu widzenia analizy bitowej, czasy trwania różnych transakcji wymiany.

Transakcja periodyczna zmiennej

Dla pojedynczej transakcji czas realizacji transakcji wynosi:

$$T_{p1}' = 61\ T_{mac} + 2\ T_{tr} + T_{as} + 61\ T_{mac} + 8\ n\ T_{mac},$$

$$T_{p1}' = T_{mac}(122 + 8\ n) + 2\ T_{tr} + T_{as},$$

Natomiast dla całego okna:

$$T_{p1} = 8T_{mac} \sum_{i=1}^{L_{p1}} n_i + (2T_{tr} + T_{as} + 122T_{mac})L_{p1}. \quad (77)$$

Dla przypadku pesymistycznego wartość tego czasu wyniesie:

$$T_{p1_max} = (61T_{mac} + 2T_{tr} + T_{tout})L_{p1} \quad (78)$$

Transakcja periodyczna komunikatu bez potwierdzenia

Dla pojedynczej transakcji czas realizacji transakcji wynosi:

$$T_{p2}' = 61\ T_{mac} + 3\ T_{tr} + 2\ T_{as} + 93\ T_{mac} + 8\ k\ T_{mac} + 45\ T_{mac},$$

$$T_{p2}' = 199\ T_{mac} + 8\ k\ T_{mac} + 3\ T_{tr} + 2\ T_{as},$$

Natomiast dla całego okna:

$$T_{p2} = 8T_{mac} \sum_{i=1}^{L_{p2}} k_i + (3T_{tr} + 2T_{as} + 199T_{mac})L_{p2}. \quad (79)$$

Dla przypadku pesymistycznego:

$$T_{p2_max} = (61T_{mac} + 3T_{tr} + 2T_{tout})L_{p2} \quad (80)$$

Transakcja periodyczna komunikatu z potwierdzeniem

Dla pojedynczej transakcji czas realizacji transakcji wynosi:

$$T_{p3}' = 61\ T_{mac} + 2\ T_{tr} + T_{as} + 93\ T_{mac} + 8\ k\ T_{mac} + T_{as} + 45\ T_{mac} + T_{as} + 45\ T_{mac} + 2\ T_{tr} + + Q_{rep}(T_{resp_Ab} + 93\ T_{mac} + 8\ k\ T_{mac} + T_{tr})$$

$$T_{p3}' = 244\ T_{mac} + 4\ T_{tr} + 3\ T_{as} + 8\ k\ T_{mac} + Q_{rep}T_{resp_Ab} + 93\ Q_{rep}\ T_{mac} + 8\ k\ T_{mac}\ Q_{rep} + + Q_{rep}T_{tr}$$

$$T_{p3}' = 8\,k\,T_{mac}(1 + Q_{rep}) + T_{mac}(244 + 93\,Q_{rep}) + T_{tr}(4 + Q_{rep}) + 3\,T_{as} + Q_{rep}\,T_{resp_Ab}$$

Natomiast dla całego okna:

$$T_{p3} = 8T_{mac}\left(1+Q_{rep}\right)\sum_{i=1}^{L_{p3}} k_i + \left[\left(4+Q_{rep}\right)T_{tr} + 3T_{as} + Q_{rep}T_{resp_Ab} + \left(244+93Q_{rep}\right)T_{mac}\right]L_{p3} \quad (81)$$

Dla wartości pesymistycznych:

$$T_{p3_max} = \left(61T_{mac} + 4T_{tr} + 3T_{tout} + Q_{rep}\left(T_{resp_Ab_max} + T_{tr}\right)\right)L_{p3} \quad (82)$$

6.9.3. Transakcja aperiodyczna

Dla pojedynczej transakcji pilnej lub normalnej czas realizacji transakcji wynosi:

$$T_a' = 61\,T_{mac} + T_{as} + 45T_{mac} + 16\,m\,T_{mac} + T_{tr} + T_{Arb_proc} + QRP_RQi\left(61T_{mac} + T_{as} + 61T_{mac} + 8nT_{mac} + 2T_{tr}\right)$$

Ponieważ długość ramki typu RP_RQi dla każdej transmisji może być różna, stąd poprawnie należałoby zapisać:

$$T_a' = 106T_{mac} + T_{as} + T_{tr} + T_{Arb_proc} + 16mT_{mac} + \sum_{j=1}^{Q_{RP_RQi}}\left(122T_{mac} + T_{as} + 2T_{tr} + 8T_{mac}\,n_j\right) \quad (83)$$

Stąd dla całego okna:

$$T_a = \left[106T_{mac} + T_{as} + T_{tr} + T_{Arb_proc} + \left(122T_{mac} + T_{as} + 2T_{tr}\right)Q_{RP_RQi} + 8T_{mac}\sum_{j=1}^{Q_{RP_RQi}} n_j\right]L_a + 16T_{mac}\sum_{i=1}^{La} m_i \quad (84)$$

gdzie: j – *oznacza indeks długości ramek odpowiedzi w danej transakcji,*

i – oznacza indeks długości ramki żądań w danej transakcji.

Dla wartości pesymistycznych dla pilnej lub normalnej transakcji aperiodycznej czas realizacji transakcji wynosi:

$$T_{a_max} = \left(61T_{mac}\left(1+Q_{RP_RQi}\right) + T_{tr}\left(1+T_{tr}Q_{RP_RQi}\right) + T_{tout}\left(1+Q_{RP_RQi}\right)\right)L_a \quad (85)$$

6.10. Podsumowanie analizy i kontrola realizowalności wymian

W rozdziale 6 przeanalizowano pracę sieciowego protokołu FIP, w wyniku której uzyskano poniższe zależności (wzory (86), (87)):

$$\begin{aligned}
T_{cycle} = {} & 8T_{mac}\sum_{i=1}^{L_{p1}} n_i + \left(2T_{tr} + T_{as} + 122T_{mac}\right)L_{p1} + 8T_{mac}\sum_{i=1}^{L_{p2}} k_i + \left(3T_{tr} + 2T_{as} + 199T_{mac}\right)L_{p2} + \\
& + 8T_{mac}\left(1+Q_{rep}\right)\sum_{i=1}^{L_{p3}} k_i + \left[\left(4+Q_{rep}\right)T_{tr} + 3T_{as} + Q_{rep}T_{resp_Ab} + \left(244+93Q_{rep}\right)T_{mac}\right]L_{p3} + \\
& + \left[106T_{mac} + T_{as} + T_{tr} + T_{Arb_proc} + \left(122T_{mac} + T_{as} + 2T_{tr}\right)Q_{RP_RQi} + 8T_{mac}\sum_{j=1}^{Q_{RP_RQi}} n_j\right]L_{a1} + \\
& + 16T_{mac}\sum_{i=1}^{La1} m_i + \\
& + \left[106T_{mac} + T_{as} + T_{tr} + T_{Arb_proc} + \left(122T_{mac} + T_{as} + 2T_{tr}\right)Q_{RP_RQi} + 8T_{mac}\sum_{j=1}^{Q_{RP_RQi}} n_j\right]L_{a2} + \\
& + 16T_{mac}\sum_{i=1}^{La2} m_i + 8T_{mac}\sum_{i=1}^{La1} k_i + \left(3T_{tr} + 2T_{as} + 199T_{mac}\right)L_{m1} + \\
& + 8T_{mac}\left(1+Q_{rep}\right)\sum_{i=1}^{L_{m2}} k_i + \left[\left(4+Q_{rep}\right)T_{tr} + 3T_{as} + Q_{rep}T_{resp_Ab} + \left(244+93Q_{rep}\right)T_{mac}\right]L_{m2}
\end{aligned} \quad (86)$$

$$\begin{aligned}
T_{cycle_max} = {} & \left(61T_{mac} + 2T_{tr} + T_{tout}\right)L_{p1} + \left(61T_{mac} + 3T_{tr} + 2T_{tout}\right)L_{p2} + \\
& + \left(61T_{mac} + 4T_{tr} + 3T_{tout} + Q_{rep}\left(T_{resp_Ab_max} + T_{tr}\right)\right)L_{p3} + \\
& + \left(61T_{mac}\left(1+Q_{RP_RQi}\right) + T_{tr}\left(1+T_{tr}Q_{RP_RQi}\right) + T_{tout}\left(1+Q_{RP_RQi}\right)\right)L_{a1} + \\
& + \left(61T_{mac}\left(1+Q_{RP_RQi}\right) + T_{tr}\left(1+T_{tr}Q_{RP_RQi}\right) + T_{tout}\left(1+Q_{RP_RQi}\right)\right)L_{a2} + \\
& + \left(61T_{mac} + 3T_{tr} + 2T_{tout}\right)L_{m1} + \\
& + \left(61T_{mac} + 4T_{tr} + 3T_{tout} + Q_{rep}\left(T_{resp_Ab_max} + T_{tr}\right)\right)L_{m2}
\end{aligned} \quad (87)$$

określające dokładny czas trwania cyklu elementarnego oraz wartość czasu trwania cyklu elementarnego dla pesymistycznego przypadku. Uzyskane zależności są pomocne w tworzeniu narzędzi służących do poprawnej konfiguracji sieci, jak również pomocne na etapie projektowania systemu sieciowego. Należy mocno podkreślić konieczność przeprowadzania analizy przepływu danych, gdyż od jej rezultatów zależy poprawna praca sieci. Problem ten staje się bardziej wyrazisty z chwilą stosowania sieci w bardzo odpowiedzialnych instalacjach, a więc w takich, gdzie awaria sieci, a co gorsze niedostarczenie ważnych dla procesu przemysłowego danych w określonym czasie, może spowodować znaczne straty finansowe albo nawet stanowić zagrożenie bezpieczeństwa ludzi. Ponieważ protokół FIP pozwala stosować sieci przemysłowe do obsługi procesów szybkozmiennych, to należy się spodziewać jego wykorzystania właśnie tam. Przykładem tego może być zastosowanie sieci FIP do sterowania i kontroli obwodów parametrów lotu w samolotach lub szybkiej kolei (na przykład: najszybsza kolej świata TGV, samoloty typu AIRBUS, system kontroli ruchu

pociągów w tunelu pod kanałem La Manche). Innym odpowiedzialnym przykładem zastosowania tej sieci może być sterowanie ruchem podziemnej kolei miejskiej, jaką jest metro.

Takie zastosowanie sieci FIP wymusza konieczność przeprowadzania analizy przepływu danych. Zatem zależności *(86)* i *(87)* stanowią podstawę do opracowania metodologii i komputerowych narzędzi symulacyjnych pomagających w doborze elementów i konfiguracji sieci przemysłowej.

Parametry sieci, mające wpływ na jej działanie po procesie konfigurowania (czyli zadawania parametrów sieci), można podzielić na stałe i zmienne.

Tabela 2

Wielkości stałe i zmienne transmitowane w sieci FIP

Stałe	Ptr	–	prędkość transmisji.
	T_{ID_xxx}	–	czas transmisji ramek zapytań.
	T_{RP_xxx}	–	czas transmisji ramek odpowiedzi.
	Q_{rep}	–	ilość powtórzeń.
	L_{p1}	–	długość listy periodycznej zmiennych.
	L_{p2}, L_{p3}	–	długość listy periodycznej komunikatów.
	T_{tout}	–	czas oczekiwania dla danej zmiennej.
	$T_{resp_Ab_max}$	–	max. czas oczekiwania na potwierdzenie komunikatu.
	T_{tr}	–	czas milczenia sieci.
	T_{mac}	–	czas transmisji jednego bitu.
	T_{as}	–	czas obsługi ramek przez abonenta.
Zmienne	T_d	–	czas transmisji ramki danych.
	T_{resp}	–	czas odpowiedzi abonenta.
	T_{resp_Ab}	–	czas oczekiwania na potwierdzenie komunikatu.
	L_{a1}, L_{a2}	–	długość listy aperiodycznej (p/n).
	L_{m1}, L_{m2}	–	długość aperiodycznej listy komunikatów.
	Q_{RP_RQi}	–	ilość żądań aperiodycznych.
	T_{Arb_proc}	–	czas obsługi ramki RP_RQi przez arbitra.

Czas trwania pełnego cyklu zależy w sposób statyczny od części stałej oraz w sposób dynamiczny od części zmiennej.

Kontrola wykonalności transakcji wymiany w sieci, a zatem poprawnego cyklu elementarnego, musi uwzględniać przypadek pesymistyczny wartości parametrów zmiennych przy zadanych parametrach stałych, i sprowadza się do kontroli nierówności:

$$Tdef \geq Tc_max, \tag{88}$$

gdzie *Tdef* jest zdefiniowanym czasem trwania cyklu pracy arbitra, natomiast *Tc_max* jest pesymistycznym czasem trwania cyklu transakcji we wszystkich czterech oknach czasowych.

Oczywiście, w przypadku makrocyklu złożonego z kilku cykli elementarnych analizę należy przeprowadzić dla każdego cyklu elementarnego oddzielnie.

Na podstawie rozważań przeprowadzonych w rozdziale 6 można dodatkowo opracować algorytm automatycznej kontroli parametrów potrzebnych do konfiguracji sieci.

Użytkownik powinien zdefiniować następujące parametry:

- prędkość transmisji,
- liczbę abonentów,
- czas oczekiwania,
- czas milczenia,
- liczbę powtórzeń komunikatów przy transmisji komunikatów z potwierdzeniem,
- liczbę zmiennych i komunikatów obsługiwanych przez sieć periodycznie,
- maksymalną liczbę generowanych żądań (domyślnie *liczba_abonentów*64*):
  - przesyłów aperiodycznych,
  - przesyłów komunikatów,
- cykle realizacji:
  - transmisji cyklicznych ramek **ID_xxx**,
  - występowania okien aperiodycznych,
  - czasu trwania cyklu elementarnego.

Na podstawie tych wielkości można opracować program lub moduł programowy systemu sieciowego, który dokona weryfikacji poprawności konfiguracji i określi realizowalność wymian. W tym miejscu warto podkreślić, że taki program istnieje i jest wynikiem pracy dyplomowej [65]. Ponieważ wyniki badań nad siecią FIP znalazły praktyczną realizację nie tylko w postaci pewnych, gotowych wyrobów (karta sieciowa do komputera klasy IBM PC, moduły zdalnych wejść/wyjść analogowo-cyfrowych), ale również w postaci stanowiska laboratoryjnego, wykorzystywany jest on także jako element stanowiska badawczego i studenckiego laboratorium dydaktycznego.

6.11. Poprawa parametrów czasowych wymian w sieci typu FIP

Poprawa parametrów czasowych wymian w sieci FIP dotyczy, w pierwszym rzędzie, sposobu określenia mikro- i makrocyklu. Pamiętając o tym, że w modelu **D**ystrybutor – **P**roducent – **K**onsument istnieją trzy podstawowe kategorie transakcji wymian:

- wymiany periodyczne,
- wymiany aperiodyczne,
- wymiany komunikatów

należy zwrócić szczególną uwagę na odpowiednią konfigurację wymian periodycznych.

Konfiguracja wymian periodycznych polega na:

- ustaleniu liczby i typu zmiennych, których wartości będą transmitowane w danym cyklu,
- obliczeniu dla każdej zmiennej czasu niezbędnego do realizacji transakcji wymiany (czas potrzebny do emisji zmiennej),
- określeniu mikrocyklu i jego wartości w jednostkach czasu,
- określeniu makrocyklu i jego wartości w jednostkach czasu,
- optymalizacji sekwencji wymian periodycznych,
- określeniu bezpiecznej wartości trwania makrocyklu, tak aby czas, który pozostanie po realizacji wymian periodycznych, był wystarczający do obsługi wymian aperiodycznych i przesyłu komunikatów.

Proces konfiguracji i optymalizacji najlepiej prześledzić na przykładzie.

Załóżmy, że w sieci FIP będą realizowane wymiany cykliczne następujących zmiennych:
zmienna **A** typu – liczba całkowita 1-bajtowa,
zmienna **B** typu – liczba całkowita 2-bajtowa,
zmienna **C** typu – łańcuch 32 znaków,
zmienna **D** typu – status- słowo 32 bity,
zmienna **E** typu – liczba 32-bitowa bez znaku,
zmienna **F** typu – łańcuch 16 znaków.

Niech czas niezbędny do wysłania przez arbitra sieci żądania produkcji wartości dowolnej zmiennej wynosi:

$$T_Q = L_Q T_{MAC},$$

gdzie: T_{MAC} oznacza czas transmisji jednego bitu,
L_Q oznacza liczbę bitów serwisowych w ramce żądania (i w ramce odpowiedzi),
a czas niezbędny do wytransmitowania wyprodukowanej wartości zmiennej wynosi:

$$T_T = L_Q T_{MAC} + n8T_{MAC},$$

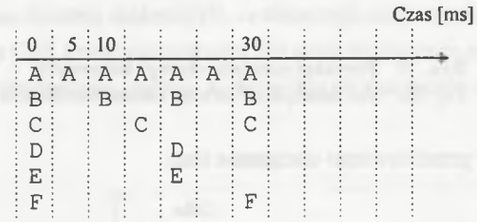
gdzie: n oznacza liczbę bajtów informacji użytkowej (dane).
Całkowity czas realizacji transakcji wymiany (żądanie-odpowiedź) wyniesie zatem:

$$T_{CT} = (L_Q T_{MAC} + T_{tr} + L_Q T_{MAC} + n8T_{MAC} + T_{tr}),$$

gdzie: T_{tr} oznacza czas odpowiedzi, w tym również czas dekodowania ramki (dekodowanie ramki nadanej przez arbitra i dekodowanie ramki nadanej przez producenta).
Przyjmując dla sieci FIP:
L_Q = 61 (warstwa fizyczna „dokłada" do ramki 61 bitów),
T_{MAC} = 1μs (przy prędkości transmisji 1 Mb/s),
T_{tr} = 20 μs,
otrzymujemy następujące czasy transmisji poszczególnych zmiennych wyrażone w mikrosekundach:

A - 170
B - 178
C - 418
D - 194
E - 194
F - 290
Razem: 1444

Tak więc minimalny czas niezbędny do „odpytania" wszystkich zmiennych wynosi 1.444 milisekundy. Określa to dolną, nieprzekraczalną granicę wyznaczania okna czasowego realizacji mikrocykli, przy założeniu że w danym mikrocyklu wszystkie zmienne będą odświeżane. Załóżmy następujące cykle „odpytywania" poszczególnych zmiennych:
A co 5 ms,
B co 10 ms,
C co 15 ms,
D co 20 ms,
E co 20 ms,
F co 30 ms,
i przystąpmy do definicji poszczególnych mikrocykli.



Rys. 38. Przykład niepełnego makrocyklu
Fig. 38. An example of incomplete microcycle

Na rys. 38 [52, 53] przedstawiono realizację poszczególnych mikrocykli. Zmienna „A" występuje w każdym z nich, bowiem cykl odświeżania wynosi 5 ms. Tak więc w zerowym mikrocyklu będą realizowane transmisje wszystkich zmiennych (co, jak zostało wyliczone, zajmie 1.444 ms).

Określmy zatem makrocykl. Na makrocykl składa się pewna liczba mikrocykli. Jak łatwo zauważyć, liczba mikrocykli składających się na makrocykl wynosi:

$$L_M = \frac{NWW}{NWD}. \qquad (88a)$$

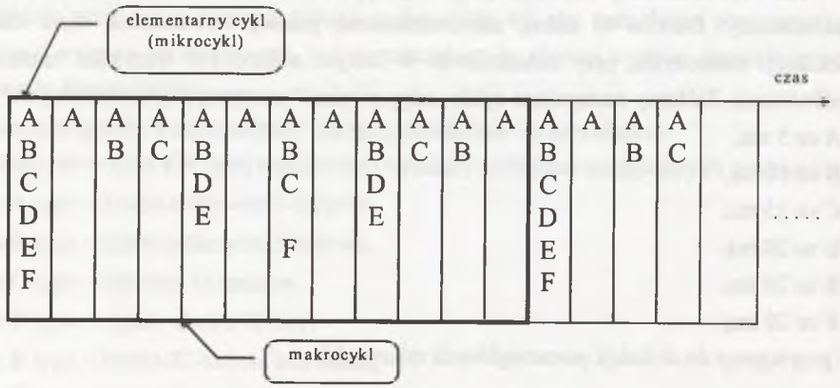
gdzie: *NWW* – oznacza najmniejszą wspólną wielokrotność wszystkich periodyczności zmiennych (w naszym przypadku jest to liczba 60),

NWD – oznacza największy wspólny podzielnik wśród wszystkich periodyczności zmiennych (w naszym przypadku jest to liczba 5).

Tak więc liczba mikrocykli, składająca się na cały makrocykl, wyniesie:

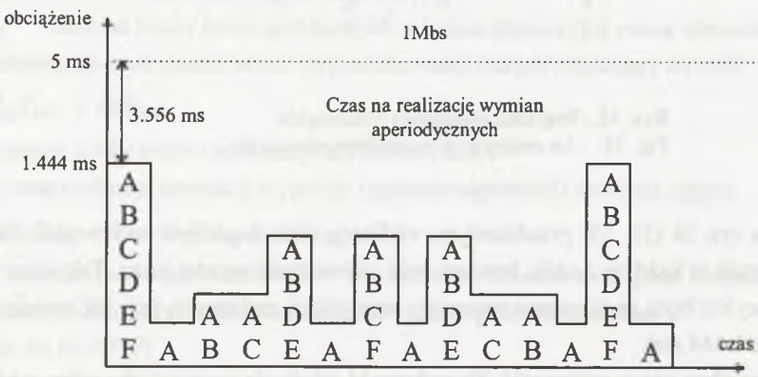
$L_M = 60/5 = 12$

Konstrukcję makrocyklu przedstawia poniższy rys. 39 [52, 53].



Rys. 39. Przykład rozkładu obsługi zmiennych
Fig. 39. An example of service values distribution

Na rys. 40 przedstawiono obciążenie sieci.



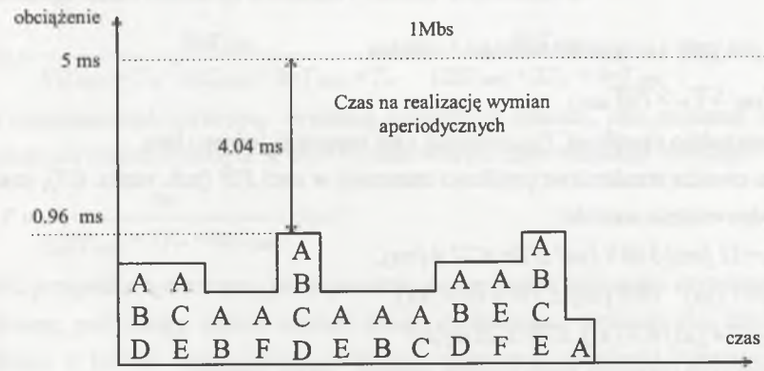
Rys. 40. Przykład obciążenia sieci
Fig. 40. An example of network load

Na rys. 40 widać, że obciążenie sieci jest nieoptymalne. Istnieją bowiem mikrocykle, po zakończeniu których pozostaje 3.556 ms na obsługę wymian aperiodycznych (mikrocykl pierwszy na rys. 40), a są takie, gdzie ten czas wynosi 4.83 ms (mikrocykl drugi na rys. 40).

„Histogram" nie jest zrównoważony. Należy doprowadzić do jego wyrównania tak, aby w każdym mikrocyklu pozostawało mniej więcej tyle samo czasu na obsługę wymian aperiodycznych lub na częstsze (o ile zajdzie taka potrzeba) „odpytywanie" którejś ze zmiennych lub dołączenie dodatkowej zmiennej do listy wymian periodycznych bez pogarszania parametrów czasowych całej sieci. Do optymalizacji makrocykli, szczególnie gdy ich liczba jest znaczna (dziesiątki, setki), należy wykorzystać program komputerowy lub moduł programowy, będący integralną częścią bądź specjalizowanego, programowego narzędzia, służącego do konfiguracji sieci FIP, bądź elementem na przykład systemu wizualizacji z wbudowanym modułem obsługi tej sieci. Wśród metod optymalizacji można zastosować metody algorytmiczne lub metody oparte na przykład na sztucznej inteligencji. Te pierwsze, w tym heurystyczne, dają najlepsze rezultaty, ale wymagają dużej mocy obliczeniowej i znacznego czasu realizacji algorytmu. Polegają one na takim przesuwaniu w ramach makrocyklu żądań obsługi zmiennej, aby otrzymać najkrótsze z możliwych poszczególne mikrocykle, albo są na zasadzie wyszukiwania mikrocykli najmniej obciążonych. Drugie - to wykorzystujące algorytmy ewolucyjne, w których rezygnuje się z ich uniwersalności (algorytm ma bowiem rozwiązywać zadania jednej klasy), na rzecz lepszej realizacji zadania optymalizacji makrocykli. Zagadnienia te w odniesieniu do wszystkich sieci z wymianami cyklicznymi są omówione w następnym rozdziale.

Optymalizacji czasu trwania makrocykli, w obszernej swej części, jest poświęcona również praca dyplomowa [76], której promotorem był autor niniejszego opracowania.

Na rys. 41 przedstawiono budowę naszego makrocyklu po dokonaniu optymalizacji wymian.



Rys. 41. Przykład makrocyklu zoptymalizowanego
Fig. 41. An example of optimised macrocycle

Jak to dobitnie ilustruje rys. 41, niezwykle ważne jest dokonywanie optymalizacji wymian w sieci przemysłowej FIP. Dzięki tej operacji można zdecydowanie poprawić jej przepustowość przez zmniejszenie okna mikrocyklu lub umożliwić realizację transakcji dodatkowych zmiennych bez pogarszania parametrów czasowych sieci.

6.12. Ocena wydajności i sprawności sieci FIP

Z informacji zawartych w poprzednich rozdziałach wynika, że całkowity czas transakcji wymiany zawiera elementarne czasy niezbędne dla:

- transmisji ramki żądania,
- transmisji ramki odpowiedzi,

i jest powiększony o dwukrotną wartość czasu T_{tr}, który zdefiniowaliśmy jako czas ciszy mierzony od momentu zakończenia odbioru (dekodowania) jednej ramki do rozpoczęcia transmisji następnej. Ilustruje to rys. 42.

Rys. 42. Definicja czasu T_{tr}
Fig. 42. The definition of T_{tr} time

Czas T_{tr} jest stały i przyjmuje wartości z zakresu

$$10T_{MAC} \le T_{tr} \le 70T_{MAC},$$

gdzie, jak poprzednio określono, T_{MAC} oznacza czas transmisji jednego bitu.

Mając na uwadze standardowe prędkości transmisji w sieci FIP (zob. rozdz. 6.7), czas T_{tr} przyjmuje odpowiednio wartości:

dla $T_{MAC}=32$ [ms] 320,0 [μs] $\le Ttr \le 22,4$ [ms],

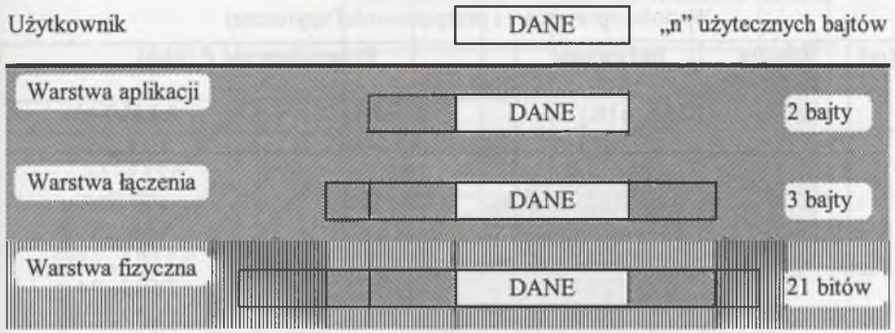
dla $T_{MAC}=1$ [μs] 10,0 [μs] $\le Ttr \le 70,0$ [μs],

dla $T_{MAC}=0,4$ [μs] 4,0 [μs] $\le Ttr \le 28,0$ [μs].

Z dotychczasowych rozważań łatwo określić ponadto „narzut" warstw:

- aplikacji,
- łączenia,
- fizycznej,

w postaci dodatkowych danych, które nie są danymi użytkowymi, a muszą być wyemitowane. Jest to zilustrowane na rys. 43.



Rys. 43. Dołączenie dodatkowych informacji do ramki danych
Fig. 43. Adding of additional information to data frame

Przyjmijmy ponadto, że:

- czas transmisji ramki żądania wynosi $61T_{MAC}$ i jest stały,
- czas transmisji ramki odpowiedzi wynosi $61T_{MAC}+8nT_{MAC}$ (n–liczba bajtów danych użytkownika),
- czas odpowiedzi wynosi $2T_{tr}$ i jest stały dla danej aplikacji.

Zdefiniujmy, podobnie jak to miało miejsce w odniesieniu do sieci „TOKEN-BUS" i „MASTER-SLAVE", sprawność sieci jako stosunek czasu transmisji danych użytkownika do całkowitego czasu transakcji wymiany wyrażany w procentach.

$$\eta = \frac{8nT_{MAC}}{61T_{MAC} + T_{tr} + 61T_{MAC} + 8nT_{MAC} + T_{tr}} = \frac{8nT_{MAC}}{122T_{MAC} + 2T_{tr} + 8nT_{MAC}} \ [\%], \qquad (89)$$

oraz przepustowość użyteczną, wyrażoną w bitach na sekundę, jako stosunek liczby bitów użytecznych transmitowanych w sieci do całkowitego czasu transakcji wymiany.

$$P = \frac{8n}{122T_{MAC} + 2T_{tr} + 8nT_{MAC}} \ [b/s]. \qquad (90)$$

W obu przypadkach n oznacza, jak poprzednio, liczbę bajtów informacji użytecznej.

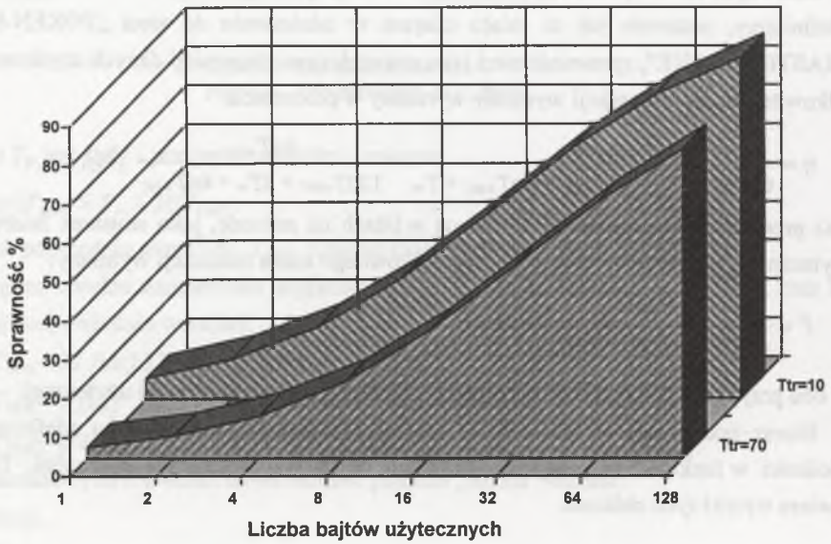
Biorąc pod uwagę wyniki analizy czasowej, dokonano obliczeń obu zdefiniowanych wielkości w funkcji liczby bajtów informacji użytecznej i prędkości transmisji. Tabela 3 zawiera wyniki tych obliczeń.

Tabela 3

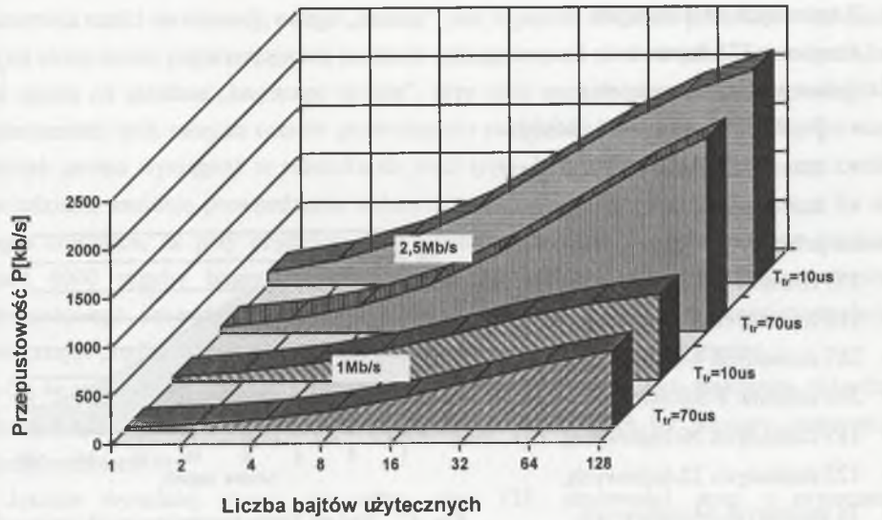
Wartości sprawności i przepustowości użytecznej

T_{tr} [μs]	Liczba bajtów N	Sprawność η [%]	Przepustowość P [kb/s]	
			1 [Mb/s]	*2,5 [Mb/s]*
10	1	5,33	53,33	133,33
	2	10,13	101,26	253,16
	4	18,39	183,90	459,77
	8	31,07	310,67	766,70
	16	47,41	474,07	1185,18
	32	64,32	643,22	1608,04
	64	78,29	782,87	1957,19
	128	87,82	878,22	2195,54
70	1	2,96	29,63	74,07
	2	5,76	57,55	143,88
	3	10,88	108,84	272,11
	8	19,63	196,32	490,80
	16	32,82	328,21	820,51
	32	49,42	494,21	1235,52
	64	66,15	661,50	1653,75
	128	79,63	796,27	1990,67

Wyniki znajdujące się w tabeli zamieszczono również na wykresach (zob. rys. 44, 45).



Rys. 44. Wykres sprawności sieci
Fig. 44. The diagram of network efficiency



Rys. 45. Wykres przepustowości sieci
Fig. 45. The diagram of network capacity flow

Dokonano również symulacji, która miała odpowiedzieć na pytanie, ile i jakiego rodzaju dane można przesłać przy założeniu określonej wartości:

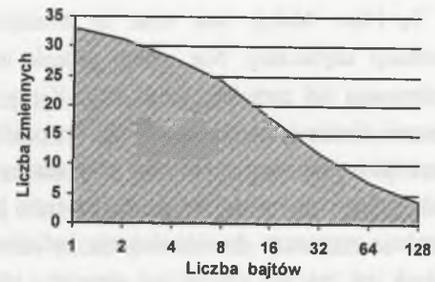
- czasu trwania cyklu elementarnego,
- prędkości transmisji,
- czasu T_{tr}.

Przyjmując:

- prędkość transmisji *1 [Mb/s]*,
- czas trwania mikrocyklu *5 [ms]*,
- czas odpowiedzi T_{tr}*=10 [ms]*,

w sieci FIP można przesłać rozłącznie:

- 33 zmienne 1-bajtowe,
- 31 zmiennych 2-bajtowych,
- 28 zmiennych 4-bajtowych,
- 24 zmienne 8-bajtowe,
- 18 zmiennych 16-bajtowych,



Rys. 46. Wykres liczby transmitowanych zmiennych
Fig. 46. The diagram of number transmiting variables

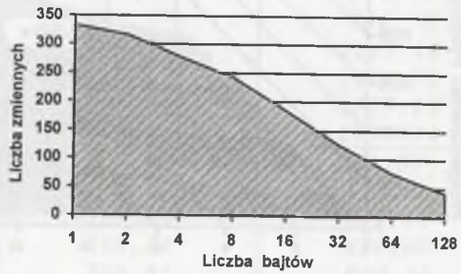
- 12 zmiennych 32-bajtowych,
- 7 zmiennych 64-bajtowych,
- 4 zmienne 128-bajtowe.

Natomiast przyjmując założenia:

- prędkość transmisji 2,5 *[Mb/s],*
- czas trwania mikrocyklu 20 *[ms],*
- czas odpowiedzi T_{tr}=*10 [ms],*

można przesłać:

- 333 zmienne 1-bajtowe,
- 316 zmiennych 2-bajtowych,
- 287 zmiennych 4-bajtowych,
- 242 zmienne 8-bajtowe,
- 185 zmiennych 16-bajtowych,
- 125 zmiennych 32-bajtowych,
- 76 zmiennych 64-bajtowych,
- 42 zmienne 128-bajtowe.



Rys. 47. Wykres liczby transmitowanych zmiennych
Fig. 47. The diagram of number transmiting variables

Na rys. 46 oraz na rys. 47 pokazano graficzną reprezentację opisanych zależności pomiędzy liczbą transmitowanych bajtów w zmiennej a liczbą samych zmiennych.

Analizując wyniki zawarte w tabeli 3 i porównując je z danymi przedstawionymi powyżej, nasuwają się pewne refleksje.

Po pierwsze, dotyczą one sprawności sieci. Z wyprowadzonych zależności wynika wniosek, że przy transmisjach zawierających niewielką liczbę danych (1–8 bajtów), sprawność jest bardzo niewielka. Krańcowo dla n=1 sprawność jest dwa razy mniejsza dla T_{tr}=70μs niż dla T_{tr}=10μs. Maleje ona wraz ze zmniejszającą się liczbą transmitowanych bajtów informacji użytecznej. Nie należy jednak interpretować wyników obliczeń sprawności w oderwaniu od protokołu sieci FIP. Kiedy będziemy dokonywać porównań sieci FIP z innymi sieciami, należy pamiętać, że produkowana informacja dociera do wszystkich uczestników wymiany informacji przyłączonych do magistrali w tym samym momencie. Kiedy informacja jest przeznaczona dla dużej liczby konsumentów (abonentów), pojedyncza transakcja wystarcza do odświeżenia buforów odbiorczych u nich wszystkich. W innych sieciach jest inaczej, a może być również i tak, że będzie wymagana taka liczba pojedynczych transakcji, ilu jest konsumentów w sieci. Wbudowany w sieci FIP mechanizm, którego celem jest zapewnienie transmitowania danych wiarygodnych, w połączeniu z gwarancją cykliczności wymian z dokładnością do deklarowanego okna czasowego, co jest realizowane przez arbitra sieci, eliminuje konieczność stosowania zasady „krążącego żetonu" jako tej,

która jest gwarantem determinizmu czasowego, niezbędnego w sieciach przemysłowych. Transmisja ramki serwisowej, owego „żetonu", jest, z punktu widzenia przepustowości sieci, czystą stratą czasu, pogarszającą ten parametr oraz sprawność sieci również. Tak więc żadna sieć oparta na zasadzie „krążącego żetonu", przy tych samych parametrach transmisji i z zachowaniem tych samych czasów przetwarzania ramek, nie dorówna sieci FIP. Ten sam wniosek można wyciągnąć w stosunku do sieci typu „MASTER–SLAVE". To, co zostało powiedziane, znajduje potwierdzenie w danych zawartych na rys. 46, 47. Wydawać by się mogło szokujące, że przy prędkości 1Mb/s jesteśmy w stanie odczytać co 5ms wartości ponad 4000 stanów binarnych docierających do systemu informatycznego z obiektu przemysłowego, i rozesłać te dane do wszystkich abonentów, którzy tych danych potrzebują. Zwiększając prędkość transmisji 2,5 raza, osiągamy rezultaty 10–krotnie lepsze!

Są to wyśmienite rezultaty, powodujące, że sieć FIP może być stosowana w bardzo szerokim zakresie w obsłudze bardzo szybkich procesów, takich jak pomiary, sterowania zabezpieczeniami itp.

Jeszcze wyraźniej rysują się zalety sieci FIP, stosowanej wraz z systemami wizualizacyjnymi czy ze stacjami nadzorczymi. Nie bez znaczenia jest bowiem fakt, że zwiększenie liczby konsumentów w sieci nie powoduje żadnych zmian w analizie czasowej. Pojawianie się coraz to nowych konsumentów w żaden sposób nie obciąża sieci, a stacje wizualizacyjne są przecież w 90% konsumentami danych. Są też takie, jak na przykład stacje monitorujące, które w 100% są konsumentami danych, Jest to bardzo istotne w procesie wizualizacji, gdyż bez żadnych dodatkowych konsekwencji, związanych z czasem przepływu danych (mogą występować ograniczenia, ale tylko związane, na przykład z długością segmentu medium), można instalować stacje wizualizacyjne pełniące funkcje monitorowania, alarmowania i archiwizacji.

6.13. Podsumowanie

Zamieszczone w rozdziale 0 wyniki analizy przepływu danych w sieci FIP są rezultatem prowadzenia przez kilka lat prac badawczych nad zakresem przemysłowych zastosowań sieci o tym protokole. Wyniki badań znalazły zastosowanie w budowie programowych symulatorów sieci FIP [65], a także stały się podstawą dla prac konstrukcyjnych nad kartami sieciowymi. Prace konstrukcyjne oparte na wynikach badań zaowocowały powstaniem własnej konstrukcji karty sieciowej realizującej pełny zakres protokołu FIP [86, 95, 85, 84, 91]. W ramach projektu celowego PC/RAu2/95 powstała pełna dokumentacja produkcyjna karty sieciowej do komputera klasy IBM PC oraz jej model i prototyp. Owa dokumentacja stanowiła podstawę do wdrożenia do seryjnej produkcji karty sieciowej o nazwie ZEG–121 w Zakładach Elektroniki Górniczej w Tychach. Karta ZEG–121 została przebadana przez firmę CEGELEC, głównego pomysłodawcę protokołu FIP, i uzyskała certyfikat pełnej zgodności

ze wzorcem (FULLFIP2). Kolejnym etapem prac konstrukcyjnych jest powstanie obiektowej karty sieci FIP, która może być wykorzystywana jako interfejs pomiędzy magistralą sieciową a urządzeniami obiektowymi, takimi jak pompa, wentylator, silnik, układ zdalnych wejść / wyjść itp. Obiektowa karta sieci FIP realizuje okrojony protokół sieci FIP (tzw. mikroFIP), gdyż nie potrafi realizować funkcji arbitra. Funkcja ta, w odniesieniu do armatury obiektowej, jest bez znaczenia, ale w zamian za to maleje zdecydowanie koszt produkcji karty bez rezygnacji z zalet sieci FIP do szybkiej wymiany informacji pomiędzy abonentami.

Równolegle z pracami konstrukcyjnymi były prowadzone prace nad oprogramowaniem. Rezultatem tych prac jest powstanie następujących narzędzi:

- symulatorów pracy sieci FIP, pozwalających poprawnie ją konfigurować i określać zakres jej stosowalności [65],
- programów testujących kartę sieciową, umożliwiających uruchamianie prototypów i ich testowanie,
- programów narzędziowych do konfiguracji stacji abonenckiej i arbitra magistrali na komputerze klasy IBM PC,
- pakietów edukacyjnych o sieci FIP (programy i systemy pomocy),
- modułów komunikacyjnych DLL sieci FIP dla systemu Windows 3.1,
- sterowników urządzeń wirtualnych sieci FIP dla systemu Windows 95 i WindowsNT.

Wyniki prac programistycznych i konstrukcyjnych pozwalają obecnie na praktyczną implementację sieci FIP w dowolnej gałęzi przemysłu. Dotyczy to nie tylko stacji obiektowych, ale również systemów zawierających stacje wizualizacyjne.

Warto również podkreślić, że rezultaty prac badawczych znalazły swe praktyczne zastosowanie. Dokonanie pełnej analizy przepływu informacji w sieci FIP pozwoliło zrealizować wiele sprawnie działających instalacji, do których należy zaliczyć:

- rozproszony system sterowania i regulacji, wraz z wizualizacją, instalacją odpylania spalin w Elektrociepłowni „Miechowice",
- rozproszony system sterowania pompami wody zasilającej i „zimnego zmieszania", wraz z wizualizacją w Ciepłowni „Stargard Szczeciński",
- rozproszony system sterowania i regulacji, wraz z wizualizacją, pompami wody zasilającej w Elektrociepłowni „Tychy",
- rozproszony system sterowania i regulacji, wraz z wizualizacją, przepływem wody grzewczej w magistrali cieplnej miasta Poznania - Zespół Elektrociepłowni Poznańskich- Elektrociepłownia „Garbary".

We wszystkich wymienionych aplikacjach sieć FIP pełni, między innymi, odpowiedzialną funkcję realizacji szybkich zabezpieczeń silnoprądowych obwodów elektrycznych.

7. POPRAWA PARAMETRÓW PRACY SIECI PRZEMYSŁOWYCH Z CYKLICZNYMI TRANSAKCJAMI WYMIANY INFORMACJI

Niniejszy rozdział dotyczy problematyki doboru długości mikrocyklu wymiany informacji w przemysłowych sieciach komputerowych, w których tryb wymian cyklicznych jest udostępniony lub wręcz jest podstawowym trybem ich pracy. Przedstawia w zarysie metody optymalizacji długości trwania mikrocyklu, które w rezultacie mają dać optymalną lub zbliżoną do optymalnej postać makrocyklu. W literaturze można znaleźć szereg pozycji traktujących o procesach cyklicznych w systemach czasu rzeczywistego [117, 118, 40, 116, 139]. Tamże można znaleźć rozwiązania lub propozycje rozwiązań kolejkowania i wyznaczania optymalnych cykli podstawowych. Najlepsze rezultaty uzyskuje się z zastosowaniem algorytmów heurystycznych, ale mają one pewne wady, o czym była krótko mowa w rozdz. 6.11. Szczególnie dotyczy to praktycznego ich zastosowania. Bardzo często bowiem, w trakcie uruchamiania systemu na obiekcie, następują liczne pojedyncze zmiany scenariuszy wymian. Za każdym razem wymagane jest utworzenie nowego makrocyklu. Bez komputera o dużej mocy obliczeniowej, przy zastosowaniu metod heurystycznych, zadanie to jest niewykonalne ze względu na brak czasu. Dlatego w niniejszym rozdziale przedstawiono również metody (w tym oparte na algorytmach genetycznych), które nie dają rozwiązań optymalnych, ale wystarczające z punktu widzenia wymagań instalacji przemysłowej.

Mówiąc o sieciach przemysłowych z cyklicznymi transakcjami wymiany informacji, należy mieć na uwadze te sieci o zdeterminowanym w czasie dostępie do medium, w których istnieje możliwość budowania w wyróżnionej stacji abonenckiej scenariusza wymian cyklicznych.

Przykładem takich sieci mogą być:

- Sieć FIP.
- Sieć o dostępie MASTER–SLAVE (na przykład sieć MODBUS).

W obydwóch sieciach podstawowymi typami wymian informacji są wymiany cykliczne. Protokoły tych sieci dopuszczają również realizację wymian, które mogą być wymuszane albo przez stację MASTER (sieci o dostępie typu MASTER-SLAVE), albo przez dowolnego abonenta, jak to ma miejsce w sieci FIP. Jeśli protokół komunikacyjny ma zapewniać determinizm czasowy, a tym samym być przydatny do budowy rozproszonych systemów

czasu rzeczywistego, to musi istnieć mechanizm zapewniający realizację wszystkich wymian cyklicznych w nieprzekraczalnym danym czasie T_C.

Chcąc zbudować sprawny system komunikacyjny, należy określić ów czas T_C. Jego wartość jest funkcją kilku czynników, do których należy zaliczyć:

- liczbę planowanych do realizacji wymian cyklicznych,
- czas realizacji wymiany cyklicznej,
- liczbę wymian wyzwalanych (aperiodycznych), które zakłada się, że będą realizowane,
- liczbę transakcji komunikatów, które mogą być inicjowane przez dowolnego abonenta (sieci FIP).

Na podstawie wymienionych danych można określić czas T_C. Jego wartość staje się parametrem systemu rozproszonego czasu rzeczywistego, a gdy ten realizuje funkcje sterowania, regulacji, wizualizacji i monitorowania procesu technologicznego, jest również parametrem instalacji technologicznej. Metody określania czasu T_C zostały przedstawione w rozdziałach 5 i 6.

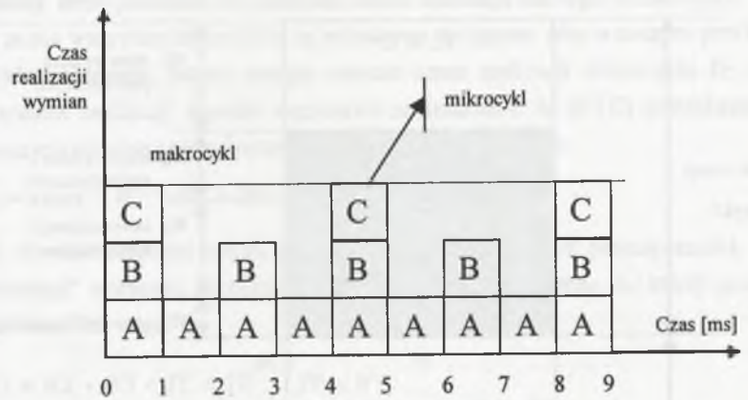
Jeśli wymagania dotyczące czasu T_C, stawiane przez proces technologiczny są spełnione, można bez większych obaw przystąpić do jego projektowania i realizacji. Jeśli natomiast wymagania te nie są spełnione lub należy usprawnić proces komunikacyjny, to wtedy nie pozostaje nic innego, jak szukać „oszczędności". Jedynym ze źródeł wspomnianych „oszczędności" jest prawidłowe określenie cyklu wymian.

7.1. Makrocykl wymiany informacji

Załóżmy, że w sieci mają być realizowane następujące, cykliczne wymiany trzech zmiennych:
zmiennej A, której okres odświeżania wynosi 1 ms,
zmiennej B, której okres odświeżania wynosi 2 ms,
zmiennej C, której okres odświeżania wynosi 4 ms.
Abstrahując na razie, od faktu, że realizacja wymiany każdej ze zmiennych wymaga nakładu czasu, stwórzmy scenariusz wymian. Jest on przedstawiony na rys.48.
Praca sieci polega na realizacji ciągu wymian:

$$W = \{ A, B, C\}, \{A\}, \{A, B,\}, \{A\}, \{A, B, C\}.....$$

Widać, że począwszy od 4 ms, proces realizacji zaczyna się powtarzać. Powstał zatem makrocykl, w którym realizowane są wymiany { A, B, C}, {A}, {A, B,}, {A}, i który będzie realizowany cyklicznie.



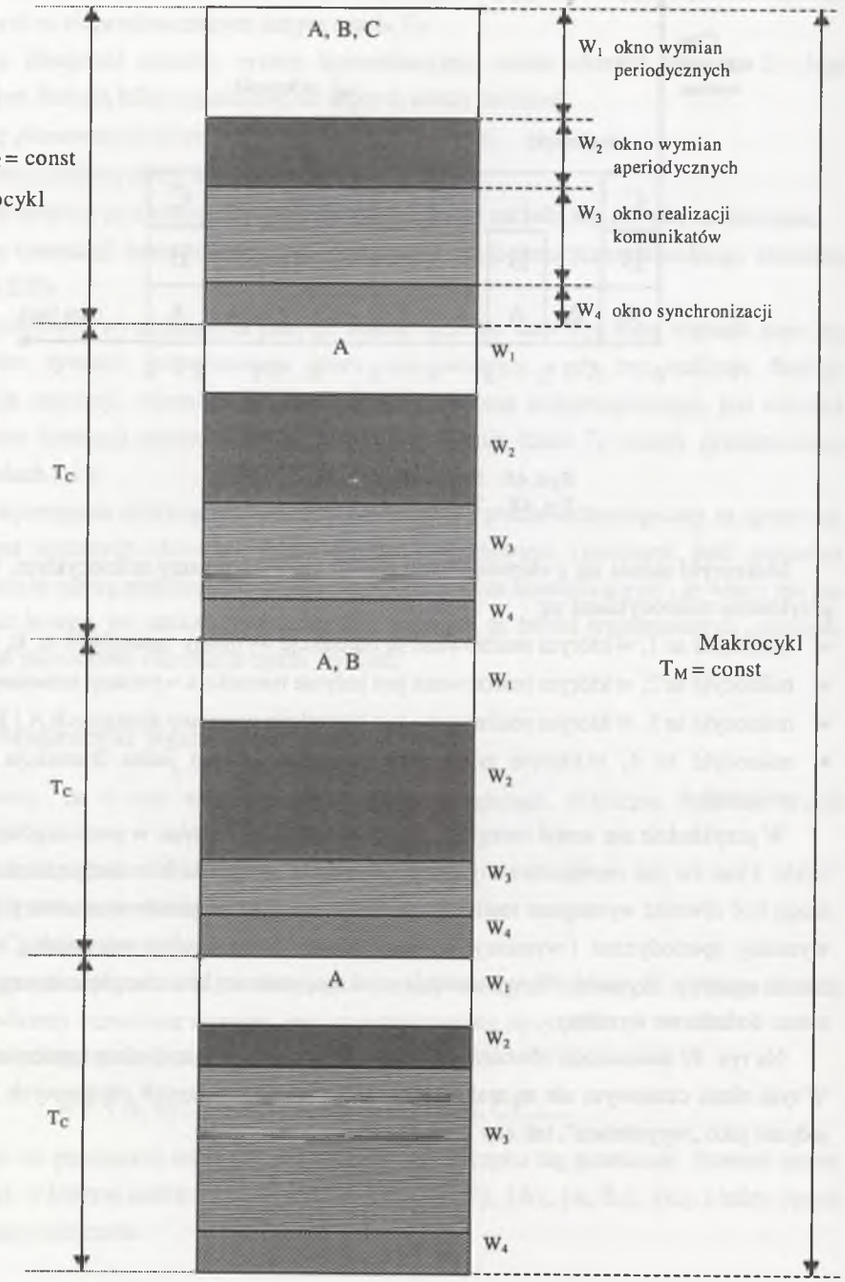
Rys. 48. Scenariusz wymian
Fig. 48. The exchanges scenario

Makrocykl składa się z elementarnych części, które nazywamy mikrocyklem. W naszym przykładzie mikrocyklami są:

- mikrocykl nr 1, w którym realizowane są transakcje wymiany zmiennych A, B, C,
- mikrocykl nr 2, w którym realizowana jest jedynie transakcja wymiany zmiennej A,
- mikrocykl nr 3, w którym realizowana jest transakcja wymiany zmiennych A i B,
- mikrocykl nr 4, w którym znów jest realizowana tylko jedna transakcja wymiany zmiennej A.

W przykładzie nie został uwzględniony czas realizacji wymian w poszczególnym mikrocyklu. Czas ów jest reprezentowany przez „wysokość" bargrafu. Nie uwzględniono faktu, że mogą być również wymagane realizacje pewnych wymian w sposób wymuszony (tak zwane wymiany aperiodyczne i wymiany komunikatów), które również wprowadzą dodatkowy narzut czasowy. Rysunek 49 reprezentuje strukturę makrocyklu, uwzględniającego wspomniane, dodatkowe wymiany.

Na rys. 49 zaznaczono również specjalne okno, noszące nazwę okna synchronizującego. W tym oknie czasowym nie są realizowane żadne wymiany danych użytkowych. Służy ono jedynie jako „wypełniacz", tak aby $T_C = const$.



Rys. 49. Struktura kompletnego makrocyklu
Fig. 49. The structure of complete macrocycle

Należy teraz powrócić do problemu czasu realizacji każdego z mikrocykli. Na rys.48 widać, że nie wszystkie mikrocykle są jednakowo obciążone. Aby w naszym przykładzie sieć pracowała poprawnie, należy przyjąć wartość czasu realizacji mikrocyklu T_C większą od sumy czasów realizacji wymian wszystkich zmiennych (A, B i C) powiększoną o czasy planowanych wymian aperiodycznych i transakcji komunikatów.

$$T_C = const > W1 + W2 + W3 \qquad (91)$$

Należy również pamiętać o tym, aby czas T_C był mniejszy od periodyczności, najczęściej „odświeżanej" zmiennej. W naszym przykładzie jest to zmienna A, której periodyczność wynosi 1 ms. Tak więc:

$$W1 + W2 + W3 < TC < TP \quad i \quad TP < W1, \qquad (92)$$

gdzie: T_P - jest czasem odświeżania zmiennej A (periodyczność zmiennej A).

Łatwo podać przykład, że zależność (92) nie będzie prawdziwa. Niech w naszym przykładzie czasy realizacji wymiany poszczególnych zmiennych hipotetycznie wynoszą:

- Zmienna A – czas realizacji 200 μs,
- Zmienna B – czas realizacji 500 μs,
- Zmienna C – czas realizacji 400 μs.

Tak więc czas realizacji okna W1 będzie wynosić:

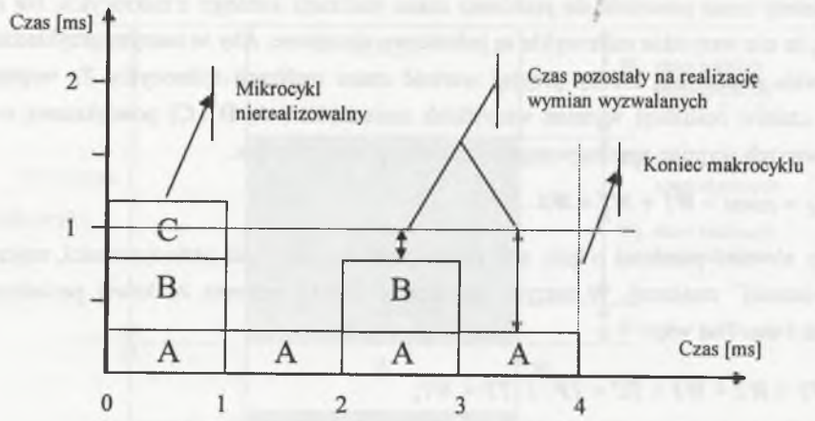
$$W1 = 200\mu s + 500\mu s + 400\mu s = 1.1\ ms,$$

co przy periodyczności zmiennej A wynoszącej 1 ms czyni pracę sieci niemożliwą, nie mówiąc już o tym, że w pierwszym mikrocyklu nie będzie można zrealizować żadnej wymiany wyzwalanej (rys.50).

Stajemy zatem wobec dwóch problemów:

- Czy w sytuacji, gdy suma czasów realizacji transakcji wymiany zmiennych w pojedynczym mikrocyklu jest większa od najmniejszej periodyczności, można w jakikolwiek sposób przekonfigurować scenariusz wymian tak, aby $T_C < T_P$?
- Czy przekonfigurowanie scenariusza może być na tyle skuteczne, aby można było, oprócz realizacji wymian periodycznych, realizować wymiany wyzwalane *(W2 > 0 i W3 > 0)*?

Odpowiedź na oba pytania jest niezwykle istotna z praktycznego punktu widzenia. Gdy bowiem technologia narzuci ostre warunki na częstotliwość odświeżania zmiennych, już na etapie wstępnym projektowania systemu, można odpowiedzieć na pytanie, czy system będzie wydolny. Będzie można określić, czy i ile wymian wyzwalanych da się zrealizować w poszczególnych mikrocyklach i jak to wpłynie na sprawność całego systemu. Wreszcie, powstanie płaszczyzna do dyskusji z technologami, czy i które zmienne mogą być odświeżane rzadziej.



Rys. 50. Realizowalność mikrocykli
Fig. 50. The microcycle realizability

Podjęcie próby poprawy pracy sieci przez odpowiedni dobór makrocyklu czy wręcz optymalizacji wymian cyklicznych umożliwi:

- skrócenie czasu realizacji mikrocyklu,
- skrócenie czasu oczekiwania na wykonanie transakcji wymian wyzwalanych (aperiodycznych) i transakcji komunikatów,
- zwiększenie liczby, możliwych do realizacji, wymian cyklicznych przy zachowaniu tych samych parametrów czasowych sieci,
- ograniczenie możliwości powstawania kolejek wymian aperiodycznych i transakcji komunikatów.

Optymalizacja wymian cyklicznych powoduje większą przepustowość użyteczną systemu komunikacyjnego dzięki bardziej równomiernemu obciążeniu sieci, co w konsekwencji powoduje, że działa ona pewniej i jest lepiej wykorzystana.

7.2. Dobór parametrów makrocyklu

Pierwszym podstawowym parametrem makrocyklu jest jego długość liczona w mikrocyklach. Liczba mikrocykli jest dobierana na podstawie wartości periodyczności wszystkich zmiennych. Można się posłużyć następującą zależnością *(88a)*:

$$L_M = \frac{NWW}{NWD}, \tag{93}$$

gdzie: L_M – oznacza liczbę mikrocykli w makrocyklu,

NWW – oznacza najmniejszą wspólną wielokrotność periodyczności wszystkich zmiennych,

NWD – oznacza największy wspólny podzielnik pośród wszystkich periodyczności zmiennych; jest to jednocześnie maksymalny dopuszczalny czas trwania mikrocyklu T_C ($T_C = NWD$).

Gdyby spróbować przedstawić zależność

$$\frac{T_{WP}}{T_{RM}} = f(L_M), \tag{94}$$

gdzie: *TWP – oznacza czas realizacji wszystkich wymian periodycznych w ramach jednego makrocyklu,*

TRM – oznacza czas realizacji całego makrocyklu,

będącą miarą wypełnienia makrocyklu transakcjami cyklicznymi, to okaże się, że osiąga ona minimum dla wartości L_M wyznaczonej z zależności *(93)* i jej wielokrotności.

Drugim parametrem makrocyklu jest minimalna liczba wystąpień zmiennej w makrocyklu. Można ją wyznaczyć z prostej zależności.:

$$L_{WZ_i} = \frac{L_M T_C}{T_{P_i}}, \tag{95}$$

gdzie: *TPi jest okresem odświeżania (periodyczność) zmiennej „i".*

Trzecim parametrem makrocyklu jest czas jego realizacji T_{PER}, który można wyznaczyć jako sumę czasów realizacji transakcji wszystkich zmiennych:

$$T_{PER} = \sum_{i=1}^{N} t_i L_{WZ_i}, \tag{96}$$

gdzie: ti to czas realizacji transakcji wymiany zmiennej „i",

LWZi to, jak poprzednio, liczba wystąpień zmiennej „i" w makrocyklu.

Zależność (*96*) określa minimalny czas realizacji transakcji zmiennych periodycznych. W czasie krótszym niż T_{PER} nie da się zrealizować założonych transakcji wymiany. Nie pozostaje nic innego, jak tylko odpowiednio umiejscowić w makrocyklu (w kolejnych makrocyklach) transakcje wymiany zmiennych, z zachowaniem ograniczeń czasowych.

Metoda rozmieszczania zmiennych wpływa na globalną liczbę możliwych transakcji. Nie będzie zatem łatwa odpowiedź na pytanie, ile można przesłać zmiennych w ramach jednego makrocyklu i jakie mogą owe zmienne mieć parametry (na przykład długość zmiennej liczona w bajtach). Zadanie sprowadza się do wielokrotnego rozwiązania problemu optymalizacji makrocyklu dla każdego zestawu danych (lista zmiennych). Realizowalność transakcji wymiany dla określonego zestawu zmiennych można dopiero określić, gdy zostanie zbudowany makrocykl. Dotyczy to tych przypadków, gdy suma czasu trwania

pojedynczych transakcji zmiennych przekracza długość wyznaczonego mikrocyklu. W którą zatem stronę powinien iść proces optymalizacji?

Powiązanie czasów trwania mikrocykli z ich liczbą oraz okresem powtarzania zmiennych i ich długości utrudnia budowanie makrocykli tak, aby:

- makrocykl był zrównoważony (podobny czas realizacji każdego mikrocyklu wchodzącego w skład makrocyklu),
- długość makrocyklu nie była zbyt duża (ograniczenie pojemności pamięci lokalnej bazy danych, gdzie postać makrocyklu jest przechowywana),
- współczynnik wypełnienia był jak najmniejszy (*94*).

7.3. Metody budowy i optymalizacji makrocykli

W literaturze niewiele można znaleźć metod rozwiązania problemu budowy makrocykli o najlepszych parametrach w odniesieniu do sieci przemysłowych. Zaprezentowany na rys.48 makrocykl nie jest najlepszy. Można go zbudować lepiej. Wystarczy bowiem przesunąć realizację wymiany zmiennej C z mikrocyklu 1 do mikrocyklu 2. Tak powstały nowy makrocykl będzie równomiernie obciążony. Jest to jednak makrocykl niezwykle prosty, w którym realizuje się wymiany tylko 3 zmiennych. W rozwiązaniach praktycznych liczba zmiennych transmitowanych w systemie może wynosić kilkaset, co czyni proces tworzenia makrocyklu niezwykle trudnym lub wręcz, w ten sposób, niewykonalnym. Przystępując do próby znalezienia metod budowania makrocykli o najlepszych parametrach dokonano oceny przydatności metod stosowanych w optymalizacji parametrycznej [149, 62, 63, 7].

Jest bardzo wiele klas zadań, dla których nie istnieją wystarczająco dobre i skuteczne algorytmy ich rozwiązania. W wielu przypadkach zadania optymalizacji dotyczą konkretnych zastosowań. Można wtedy opracować algorytm, ale zwykle nie daje on optymalnego rozwiązania, a tylko do niego zbliżone. Można posłużyć się również algorytmami probabilistycznymi, ale te z kolei nie gwarantują rozwiązania optymalnego, choć przez losowy wybór odpowiednio licznej grupy reprezentantów rozwiązania, można zmniejszyć prawdopodobieństwo błędu.

Zadanie optymalizacji wymian cyklicznych w sieciach przemysłowych nie jest łatwe, gdyż należy powiązać ze sobą wiele zależności zachodzących podczas transmisji danych przy rygorystycznym zachowaniu determinizmu czasowego. W [76] przeprowadzono dyskusję stosowalności różnych metod optymalizacji dla optymalnego doboru makrocyklu w sieci FIP. Godne uwagi, ze względu na możliwość wykorzystania dla niemalże każdej sieci przemysłowej, są metody algorytmiczna i metoda oparta na sztucznej inteligencji.

7.3.1. Metody algorytmiczne

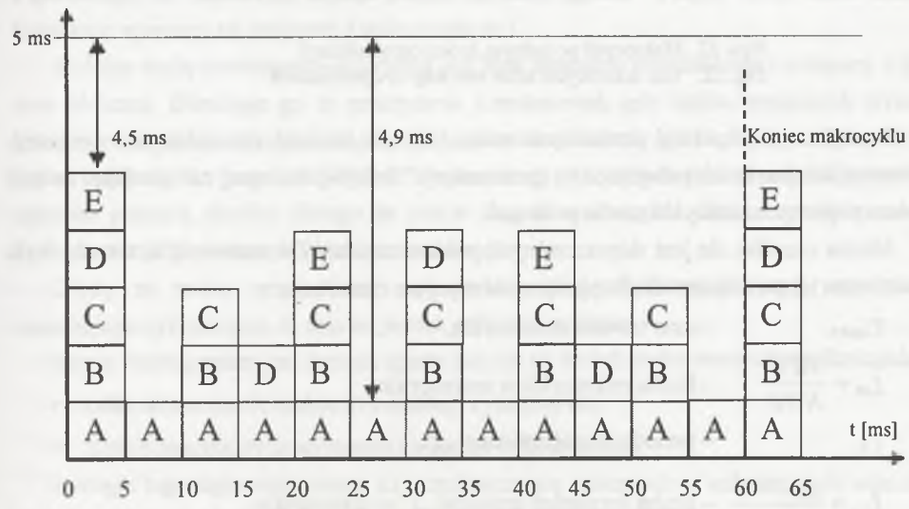
W [106] i [87] przedstawiono algorytm, którego strategia polega na takim „przesuwaniu" zmiennych w ramach makrocyklu z zachowaniem ich periodyczności, aby uzyskać makrocykl o najlepszych parametrach. Celem wyjaśnienia jego strategii posłużmy się przykładem. W sieci są realizowane transakcje wymiany czterech zmiennych o następujących periodycznościach:

A co 5 ms
B co 10 ms
C co 10 ms
D co 15 ms
E co 20 ms

Dla prostoty załóżmy, że czas realizacji wszystkich wymian jest taki sam i wynosi 100 μs. Korzystając z zależności (*93*), określmy liczbę mikrocykli w makrocyklu:

$$L_M = \frac{NWW}{NWD} = \frac{60}{5} = 12$$

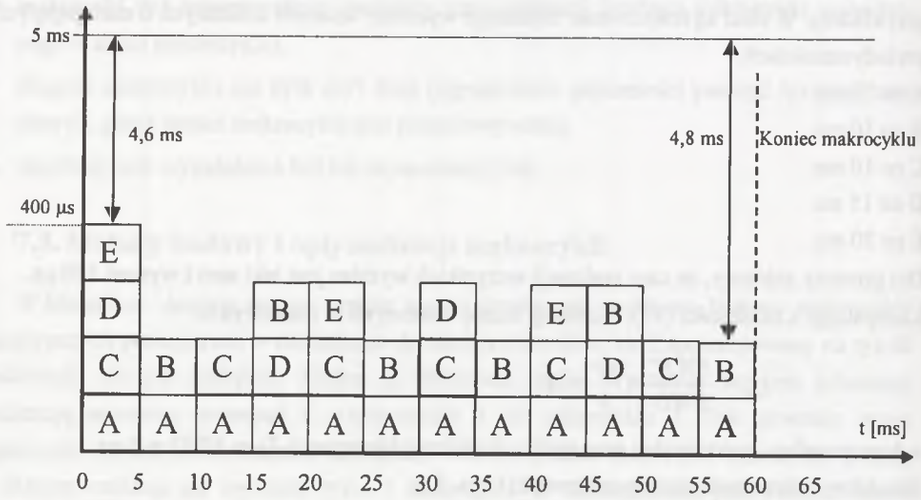
a dopuszczalny, maksymalny czas realizacji mikrocyklu wynosi $T_C = NWD = 5\ ms$.
Zbudujmy zatem podstawowy makrocykl (rys. 51).



Rys. 51. Makrocykl podstawowy
Fig. 51. The basic macrocycle

Widać, że makrocykl nie jest równomiernie obciążony, ale jest realizowalny, gdyż suma czasów realizacji wszystkich wymian wynosi 500 μs, a maksymalny czas realizacji mikrocyklu wynosi 5 ms. Mamy zatem w najgorszym przypadku 4.5 ms na realizację wymian

wyzwalanych (wymiany zmiennych aperiodycznych i transakcje komunikatów). Można jednak pokusić się o jego lepsze zaprojektowanie. „Przesuńmy" w tym celu zmienną B o jedną pozycję w prawo (rys.52) w każdym mikrocyklu, w którym ona występuje. Przesunięcie może występować tylko w obszarze makrocyklu.



Rys. 52. Makrocykl po jednym kroku optymalizacji
Fig. 52. The macrocycle after one step of optimisation

Po wykonaniu operacji przesunięcia makrocykl jest bardziej równomiernie obciążony. Niestety, kolejne kroki, polegające na „przesunięciu" kolejnej zmiennej nic nie dają i na tym proces poprawy makrocyklu trzeba przerwać.

Można określić, ile jest dopuszczalnych położeń zmiennej w makrocyklu, tak aby była zachowana jej periodyczność. Przyjmijmy następujące oznaczenia:

T_{MAKR} – czas trwania makrocyklu,

$L_M = \frac{NWW}{NWD}$ – liczba mikrocykli w makrocyklu,

T_{Pi} – periodyczność zmiennej „i",

$L_{Wi} = \frac{L_M NWD}{T_{Pi}}$ – liczba wystąpień zmiennej „i" w makrocyklu ,

L_{MP_i} – liczba maksymalnych, możliwych położeń zmiennej „i" w obszarze makrocyklu, z zachowaniem jej periodyczności .

Wtedy liczbę maksymalnych możliwych położeń zmiennej „i" można wyrazić zależnością:

$$L_{MP_i} = \frac{T_{MAKR} - (L_{Wi} - 1)T_{P_i}}{NWD} \qquad (97)$$

Aby przeszukać wszystkie możliwe rozwiązania, należy skonstruować C_R różnych postaci makrocykli

$$C_R = L_{MP_1} L_{MP_2} \cdots L_{MP_k} \cdots L_{MP_N} \qquad (98)$$

gdzie: N - jest liczbą występujących zmiennych cyklicznych w makrocyklu.

W każdym ze skonstruowanych makrocykli należy znaleźć mikrocykl o maksymalnym czasie trwania T_{Mi} i stworzyć ich zbiór $M = max \{ T_{Mi} \}$. Kolejnym krokiem jest znalezienie minimum w zbiorze M i zrekonstruowanie makrocyklu, który temu minimum odpowiada.

Algorytm ma kilka wad. Przy istniejącym ograniczeniu dotyczącym pozostawiania zmiennych w równych odległościach, co nie dopuszcza do większej liczby ich przesyłów, niż to jest konieczne, łatwo jest znaleźć takie dane początkowe, dla których wyniki nie będą optymalne, jeśli za miarę optymalizacji wziąć minimalny czas realizacji mikrocyklu. Przykładem niech tu będzie rys.52. Wystarczy bowiem dodać dodatkowe transakcje wymiany zmiennej E w mikrocyklach 2 i 12, co spełni wymagania periodyczności odświeżania, a spowoduje, że makrocykl będzie jeszcze bardziej „płaski". Należy oczywiście usunąć transakcję wymiany tej zmiennej z mikrocyklu nr 1.

Kolejną wadą omawianego algorytmu jest jego złożoność obliczeniowa i związany z tym czas obliczeń. Eliminuje go to praktycznie z zastosowań, gdy liczba zmiennych wynosi kilkaset, co w praktycznych przemysłowych zastosowaniach nie jest liczbą zbyt wielką. Duża liczba zmiennych, przy „złośliwych" parametrach ich odświeżania, powoduje wzrost zajętości pamięci, choćby dlatego że rośnie długość makrocyklu. No i oczywiście rośnie liczba konstrukcji makrocykli i ich przeszukiwań.

Gdyby za miarę optymalizacji makrocyklu wziąć współczynnik jego wypełnienia transakcjami cyklicznymi, to daje on jednak najlepsze rezultaty.

Innym rozwiązaniem jest metoda oparta jedynie na dwóch podstawowych wielkościach:

- całkowitym czasie realizacji transakcji cyklicznych,
- liczbie minimalnych wystąpień zmiennej w makrocyklu.

Strategia tego algorytmu polega na rozmieszczaniu zmiennych w mikrocyklach najmniej obciążonych spośród mikrocykli w obrębie okresu zmiennej.

Kolejne wystąpienie zmiennej musi mieć miejsce w zasięgu okresu w mikrocyklu o lokalnie najmniejszym obciążeniu. Proces ten należy kontynuować aż do momentu uzyskania cykliczności zmiennej w całym makrocyklu. Jeśli w kolejnych cyklach rośnie obciążenie, to aktualnie rozmieszczana zmienna zostanie wstawiona do wszystkich mikrocykli. Aby

wyeliminować tę wadę, można wprowadzić funkcję usuwania nadmiarowych przesyłów. Winna ona usunąć te przesyły zmiennej, których usunięcie nie spowoduje zwiększenia okresu zmiennej ponad dozwolony. W omawianej metodzie istotny jest dobór kolejności rozmieszczania zmiennych. Metoda ta została zaproponowana i przebadana w [62].

Dobre wyniki osiąga się rozmieszczając zmienne począwszy od tej, która ma najdłuższy cykl powtarzania (periodyczność) lub czas jej realizacji jest największy.

Oba omawiane algorytmy nie gwarantują optymalnego rozwiązania. Algorytm oparty na metodzie wyszukiwania najmniej obciążonych mikrocykli jest bardziej uniwersalny. Ponadto jest szybki i przy tym daje powtarzalność wyników.

7.3.2. Metoda oparta na sztucznej inteligencji

Wykorzystanie tej metody jest pomysłem zmierzającym do poprawy parametrów makrocykli. Możliwość zastosowania jej jest oczywista, a pomysł zrodził się z chwilą, gdy okazało się, iż należy wygenerować nowy makrocykl w czasie od kilku do kilkudziesięciu minut, w warunkach rozruchu systemu, na obiekcie przemysłowym.

Ogólnie strategie rozwiązywania zadań wykorzystujące „sztuczną inteligencję" można podzielić na strategie „mocne" i „słabe" [62, 63, 110]. Strategie „słabe" wykorzystują mało założeń dotyczących zastosowań, dzięki czemu są uniwersalne, ale bywają mało skuteczne. Strategie „mocne" opierają się na mocnych założeniach dotyczących zastosowań, gwarantując ich lepsze działanie, ale kosztem ograniczeń zastosowań do pojedynczej klasy zadań. Najpowszechniej do rozwiązywania problemów optymalizacji parametrycznej stosuje się algorytmy genetyczne lub ewolucyjne.

Klasyczne algorytmy genetyczne używają do opisu osobników łańcuchów binarnych o skończonej długości. Są wykorzystywane dwa operatory: binarnego krzyżowania i binarnej mutacji. Oba algorytmy zarówno genetyczny, jak i ewolucyjny mają tę samą budowę. Różnią się tylko operatorami i sposobem reprezentacji osobników. Algorytmy ewolucyjne nie wymagają, aby chromosomy były reprezentowane przez łańcuchy binarne, natomiast operatory muszą być dostosowane do struktury reprezentującej i typu zadania. Algorytmy genetyczne dzięki swej niezależności od zastosowań są uniwersalne, ale mogą nie dawać zadowalających rezultatów przy rozwiązywaniu zadań praktycznych. Stąd też preferuje się bardziej elastyczne algorytmy ewolucyjne, które ściślej są powiązane z rozwiązywanym problemem i mogą dawać szybciej i bardziej optymalne rozwiązania. Dodatkowo klasyczne algorytmy genetyczne wymagają modyfikacji pierwotnego zadania do nadającej się dla nich postaci.

Natomiast w strategii algorytmów ewolucyjnych są one dostosowane do rozwiązywanego problemu przez odpowiedni dobór reprezentacji chromosomowej, potencjalnych rozwiązań i operatorów zmiany.

Algorytmy ewolucyjne nadają się do realizacji zadania optymalizacji makrocykli w przemysłowych sieciach komputerowych, z cykliczną wymianą informacji. Wynik końcowy rozwiązywania zadania zależy od sposobu reprezentacji, przyjętych ograniczeń i stopnia ich uwzględnienia. Stosując ten rodzaj algorytmu do optymalizacji makrocyklu, zawężamy rodzaj zadania do jednej klasy, a tym samym możemy zrezygnować z jego uniwersalności na rzecz lepszej realizacji optymalizacji makrocyklu. Stosowany algorytm ewolucyjny powinien zaliczać się do grupy „mocnych" i powinien stosować możliwie dużo założeń dotyczących rozwiązywanego zadania, dzięki czemu wyniki będą zbliżone do optymalnych i będą otrzymywane szybko.

Proponowany algorytm optymalizacji makrocykli oparty jest na schemacie działania algorytmów ewolucyjnych. Program realizujący algorytm przechowuje populację osobników, która jest po prostu zbiorem makrocykli. Liczba pamiętanych rozwiązań jest stała, a program wykonuje określoną, zadaną liczbę iteracji. Każda iteracja zawiera w sobie selekcję rozwiązań do następnego kroku. Dokonywana jest również modyfikacja niektórych uzyskanych rozwiązań w celu generacji nowych. Kryterium selekcji oparte jest na dopasowaniu rozwiązania. Mówiąc o dopasowaniu w kontekście makrocyklu, należy mieć na uwadze dwa elementy:

- sumę czasów realizacji pojedynczych wymian cyklicznych w makrocyklu, co ma wpływ na współczynnik wypełnienia *(94)*,
- sumę czasów realizacji pojedynczych wymian cyklicznych w najbardziej obciążonym mikrocyklu, co ma wpływ na czas pozostający na realizację wymian asynchronicznych i transakcję wymiany komunikatów w każdym mikrocyklu.

Dopasowanie rozwiązania jest funkcją tych dwóch elementów. Najlepiej „dopasowany" makrocykl jest wybierany do następnego pokolenia. Pozostała część reprezentantów rozwiązań jest dobierana na podstawie względnego dopasowania. Proces selekcji kończy się procesem mutacji, który jest wykonywany z pewnym prawdopodobieństwem. Proces mutacji polega na zmianie jednego z elementów istniejącego osobnika, dzięki czemu otrzymuje się nowe rozwiązanie. Tak uzyskane rozwiązanie jest weryfikowane pod kątem naruszenia narzucanych ograniczeń. Weryfikacja, w zależności od stopnia naruszenia ograniczeń, kończy się ukaraniem osobnika przez wyznaczenie funkcji kary. Osobnik taki może być również poddany procesowi naprawczemu. Proces karania i naprawy ma miejsce w chwili obliczania dopasowania.

Ocena dopasowania rozwiązania musi być wyznaczona dla każdego rozwiązania cząstkowego.

Określmy dopasowanie jako:

$$D = K_1 \frac{T_M L_M}{T_{RM}} + K_2 \frac{T_M T_{MAX}}{T_M} - F, \qquad (99)$$

gdzie: T_M – oznacza czas trwania mikrocyklu,

L_M – oznacza liczbę mikrocykli w makrocyklu,

TRM – oznacza sumę czasów realizacji transakcji wymian w makrocyklu,

$TMAX$ – oznacza czas realizacji transakcji wymian w najbardziej obciążonym makrocyklu,

F – oznacza wartość kary za naruszenie ograniczeń,

K_1, K_2 – oznaczają współczynniki proporcjonalności, które dobiera się doświadczalnie.

Dobór funkcji dopasowującej ma duże znaczenie dla uzyskanego, końcowego wyniku. Manipulując współczynnikami K_1 i K_2 można wpłynąć na to, czy przywiązujemy większą wagę do minimalizacji liczby transakcji wymian, czy bardziej zależy nam na równomierności obciążenia makrocyklu.

Kolejnym zagadnieniem jest określenie ograniczeń i sposobów naprawy osobników zdyskwalifikowanych. Każdy makrocykl musi spełniać następujące warunki:

- transakcje wymian muszą zachowywać periodyczność,
- transakcja wymiany określonej zmiennej może wystąpić w mikrocyklu tylko raz,
- transakcja wymiany każdej zmiennej musi wystąpić co najmniej raz w makrocyklu.

Dla tych makrocykli, w których został przekroczony czas trwania mikrocyklu, jest wyznaczona kara, obniżająca wartość dopasowania. Karę wymierza się również każdej transakcji wymiany, która przekracza wyznaczony czas trwania mikrocyklu.

Sposoby naprawy makrocyklu polegają na:

- wstawieniu transakcji do pierwszego mikrocyklu, jeśli ta w ogóle nie wystąpiła w makrocyklu,
- usunięciu transakcji wymiany z mikrocyklu, jeśli już w nim wystąpiła,
- wstawieniu dodatkowej transakcji wymiany zmiennej, jeśli została naruszona jej periodyczność.

Następnym procesem jest selekcja rozwiązań. Selekcja rozwiązań polega na ocenie chromosomu. Jeśli ocena chromosomu jest większa od średniej, to taki chromosom ma większą szansę zostać wybranym do następnego pokolenia lub procesu reprodukcji. Zadanie selekcji rozpoczyna się od oceny wszystkich osobników, po czym obliczane jest dopasowanie całej populacji (D_p). Odbywa się to przez dodanie do siebie dopasowania poszczególnych makrocykli. Podobieństwo wyboru P_i dla każdego chromosomu wynosi:

$$P_i = \frac{D_i}{D_p}, \qquad (100)$$

gdzie: D_i – jest dopasowaniem makrocyklu „i",

D_p – jest dopasowaniem całej populacji.

Po obliczeniu dystrybuanty dla każdego chromosomu,

$$q_i = \sum_{K=1}^{i} P_K, \quad q_0 = 0, \qquad (101)$$

gdzie: „i" jest indeksem osobnika,

należy wyznaczyć losowy ciąg liczb „n" z zakresu [0,1].

Liczebność ciągu wyznacza rozmiar populacji. Wylosowane wartości będą służyły do wyboru osobników według następującej reguły:

Jeśli

$$q_{i-1} < n \le q_i,$$

to osobnik o indeksie „i" jest wybierany do następnej populacji. Jest to sposób na pozyskanie zaplanowanej liczby osobników do nowej populacji.

W omawianym algorytmie zaproponowano dwa rodzaje mutacji: mutację genów i mutację makrocyklu. Mutacja genów może polegać na (z określonym prawdopodobieństwem dla każdego z nich):

- usunięciu genu,
- wstawieniu identycznego genu w losowo wybranym miejscu makrocyklu,
- wstawieniu przed analizowanym genem genu wybranego losowo.

Mutacja mikrocyklu polega na odnalezieniu mikrocyklu z najdłuższym czasem realizacji wymian periodycznych i dla niego przeprowadza się jedną z następujących operacji:

- albo usuwa się losowo wybrany gen,
- albo przesuwa się wszystkie wystąpienia genu w ramach makrocyklu o losowo wybraną liczbę mikrocykli,
- albo przesuwa się losowo wybrany gen o losowo wybraną liczbę mikrocykli „do tyłu" (w kierunku malejącym osi czasu). Iloczyn liczby przesunąć czasu trwania mikrocyklu musi być mniejszy od okresu (periodyczności) zmiennej.

Ostatnim etapem jest proces uzyskiwania pierwszego pokolenia oraz dobór współczynników i prawdopodobieństw mutacji.

Do stworzenia początkowego pokolenia należy użyć algorytmu, który w sposób losowy rozmieści transakcje wymiany w całym makrocyklu. Spowoduje to znaczne zróżnicowanie populacji na początku wykonywania programu. Strategia rozmieszczenia polega na:

- wylosowaniu mikrocyklu, do którego zostaje wstawiona transakcja wymiany,
- wstawianiu transakcji od tego mikrocyklu począwszy, do „późniejszych" mikrocykli w taki sposób, aby realizacja wymiany miała miejsce nie rzadziej niż okres (periodyczność) zmiennej.

Dzięki takiemu postępowaniu uzyskuje się szybciej rozwiązania, które są zbliżone do optymalnych.

Wartości współczynników proporcjonalności oraz wielkości kar można dobrać empirycznie. Podobnie poszczególne prawdopodobieństwa wystąpienia mutacji. Czyni się to tak, aby uzyskać najlepszą z możliwych zbieżność algorytmu.

7.4. Realizacja praktyczna algorytmów i wnioski

W oparciu o proponowane metody tworzenia optymalnych makrocykli powstała aplikacja, która została dołączona jako moduł narzędziowy do systemu wizualizacyjnego „KRONOS". Aplikacja ta została szczegółowo opisana w [62]. Jest ona regularnie wykorzystywana na etapie projektowania systemów opartych na sieciach przemysłowych typu FIP. Program poszukuje optymalnego rozwiązania, wykorzystując metodą ewolucyjną lub wyszukiwania najmniej obciążonych mikrocykli. Badania nad praktycznymi implementacjami obu algorytmów prowadzą do kilku spostrzeżeń.

Po pierwsze, choć podprogram optymalizacji genetycznej działa wolniej i wymaga więcej pamięci, to daje jednak lepsze rezultaty od pozostałych.

Po drugie, wprowadzenie choćby jednego makrocyklu, wygenerowanego algorytmem wyszukiwania najmniej obciążonych mikrocykli, do pokolenia początkowego algorytmu ewolucyjnego, pogarsza jakość uzyskiwanych wyników. Następuje, między innymi, wydłużenie czasu oczekiwania na zbudowanie optymalnego makrocyklu, a nierzadko wynik końcowy okazywał się gorszy.

Po trzecie, metoda optymalizacji ewolucyjnej okazała się czasochłonna i wymagająca znacznych zasobów komputera. Dotyczy to przede wszystkim zajętości pamięci, szczególnie wtedy gdy dokonuje się selekcji do następnego pokolenia. W najgorszym przypadku może się tak zdarzyć, że wymagania dotyczące pojemności pamięci wzrosną aż dwukrotnie. Ma to miejsce wtedy, gdy tylko jeden z osobników zostanie wybrany do następnego pokolenia. Wtedy bowiem w pamięci trzeba przechowywać liczbę osobników równą:

L_{OS}= *2 x wielkość_populacji-1.*

Po czwarte, zaimplementowane programowo metody nie są idealne. Można zastanawiać się nad innymi strategiami rozmieszczenia transakcji w makrocyklu. Można również metodę optymalizacyjną przeprowadzić na wiele innych sposobów. Można wreszcie przeprowadzić eksperymenty z operatorem krzyżowania, dzięki któremu powstają nowe rozwiązania wskutek złożenia fragmentów kodu genetycznego dwóch osobników.

Bez wątpienia powstała aplikacja jest bazą do badań nad zastosowaniem metod ewolucyjnych do procesów optymalizacji cykli wymiany informacji w sieciach.

Na koniec kilka uwag wynikających z praktycznego wykorzystania modułu programowego do celów projektowania rozproszonych systemów przemysłowych. Przed projektantem

systemu, na wstępnym etapie projektowania, stoi odpowiedź na pytanie, czy przy założonym minimalnym czasie odświeżania wartości zmiennych wystarczy czasu na realizację wszystkich wymian.

Wykorzystanie wspomnianego modułu programowanego doskonale się nadaje do tworzenia makrocykli o bardzo dobrych parametrach, tym bardziej że proces optymalizacji można przerwać w dowolnym momencie, to znaczy wtedy, gdy projektant uzna, że uzyskane rezultaty całkowicie go satysfakcjonują. Przyspiesza to znacząco jego pracę, gdyż nie musi czekać, aż program skończy swoją pracę. Poszukiwanie absolutnie najlepszych rozwiązań występuje incydentalnie, ale to są właśnie najbardziej interesujące i najważniejsze przypadki z punktu widzenia bezpieczeństwa pracy instalacji. Ma to szczególne miejsce wtedy, gdy instalacja już pracuje, a oczekuje się lepszych parametrów systemu komputerowego.

Zaimplementowane metody, jak już wspomniano, są przydatne w budowaniu systemów komputerowych opartych na sieci FIP. Przez zmianę funkcji dopasowania w algorytmie ewolucyjnym można wymuszać lepsze rozwiązania. Projektant może zdecydować, zależnie od swoich celów, czy zależy mu bardziej na zrównoważeniu makrocyklu, przy jednoczesnej minimalizacji liczby transakcji, bo wtedy pozostaje więcej czasu na realizację wymian aperiodycznych i transakcję komunikatów, czy bardziej mu zależy na zrównoważeniu makrocyklu, dzięki czemu uzyska możliwie stały przedział czasu na transakcje aperiodyczne w poszczególnych mikrocyklach. Jasne jest, że w tym drugim przypadku chodzi również o minimalizację liczby transakcji wymiany, ale przy jednoczesnym założeniu likwidacji maksimów obciążenia transakcjami periodycznymi makrocyklu.

8. METODA KLASYFIKACJI PROTOKOŁÓW KOMUNIKACYJNYCH W SIECIACH PRZEMYSŁOWYCH

Obecnie na rynku występuje duża różnorodność komercyjnych rozwiązań problemu transmisji danych w najniższych warstwach obiegu informacji (sieci najniższego poziomu), dedykowanych dla zastosowań bezpośredniej kontroli procesu produkcji. Spotyka się rozwiązania zarówno kosztowne, jak i tanie, umożliwiające szybki transfer informacji, a także mniej wydajne. Każda firma podpisująca się pod danym rozwiązaniem twierdzi, iż ten protokół, a nie żaden inny, jest najlepszy i stanowi właściwy kierunek na przyszłość lub udowadnia swoją wyższość, przedstawiając rachunek ekonomiczny poparty rachunkiem prawdopodobieństwa. Analizując literaturę [52, 93, 136, 160] i inne oraz dane techniczne dotyczące różnych sieci komputerowych stosowanych w przemyśle, trudno jest definitywnie wybrać jedno lub dwa rozwiązania, które przynajmniej częściowo cechowałby uniwersalizm zastosowań lub genialność koncepcji. Niniejszy rozdział stanowi próbę innego, od spotykanych, spojrzenia na klasyfikację sieci i ma na celu zwrócenie uwagi na pewne problemy, które mogą mieć wpływ na to, aby wybrany protokół komunikacyjny był zawsze najbardziej odpowiedni względem potrzeb [92].

Aby ustrzec się nieporozumień odnośnie do dziedziny zastosowań uwzględnianych protokołów komunikacyjnych, poniżej zdefiniowano strukturę typowego sieciowego obiegu informacji w systemach informatycznych stosowanych w przemyśle. Zaproponujmy podział owej struktury na cztery poziomy [92]:

Poziom 1

- niech to będzie poziom wymiany danych związanych z bezpośrednim stanem urządzeń wykonawczych, zadajników sygnałów, pomiarów oraz bezpośrednim sterowaniem, transmisją rozkazów, potwierdzeń, alarmów itp. Obieg informacji na tym poziomie dotyczy tylko konkretnego procesu technologicznego i obiektów z nim związanych. Ruch na tym poziomie w niewielkim stopniu może zależeć od użytkownika. Poziom ten ma bezpośredni wpływ na realizację procesu technologicznego.

Poziom 2

- stanowi sieć nadzorczą obsługującą wymianę informacji pomiędzy stacjami użytkowników – nadzorców procesu. Obieg informacji dotyczy grupy procesów i umożliwia wymianę danych między nimi. Zwiększenie lub stabilizacja ruchu w dalszym

ciągu nie zależy w dużej mierze od użytkownika. Poziom ten ma pośredni wpływ na realizację procesu technologicznego.

Poziom 3

- stanowi sieć nadrzędną. Zwykle funkcję tę pełni zakładowa sieć lokalna. Sieć ta obsługuje wymianę informacji pomiędzy stacjami roboczymi wszelakich użytkowników w zakładzie. W szczególności zachodzi wymiana danych pomiędzy stacjami nadrzędnymi a serwerami i aplikacjami pracującymi na stacjach planowania i przygotowania produkcji, magazynów, kadr, kierownictwa itp. Intensywność ruchu w sieci w dużym stopniu zależy od jej użytkowników. Przebieg procesu technologicznego jest już tylko monitorowany.

Poziom 4

- stanowi sieci zewnętrzne. Wszelkie sieci pozazakładowe dołączone do lokalnych struktur sieciowych zakładu nie mają wpływu na przebieg procesu. Ruch zależy od bardzo wielu czynników.

Sieci, które są przedmiotem niniejszych rozważań (sieci najniższego poziomu), znajdują zastosowanie tylko na Poziomie 1 tak skonstruowanego systemu obiegu informacji. Najczęściej sieci takie spełniają funkcje dystrybucji danych dla obiektów uczestniczących w procesach sterowania, regulacji, monitorowania i wizualizacji.

Dokonajmy podziału abonentów pracujących na tym poziomie na następujące kategorie:

- prosty abonent obiektowy (AP) –urządzenie funkcjonalnie dedykowane (np. pomiarowe, wykonawcze) wraz z interfejsem sieci,
- złożony abonent obiektowy (AZ) –urządzenie funkcjonalnie uniwersalne (np. swobodnie programowalny sterownik),
- stacja nadrzędna (SN) –urządzenie stanowiące interfejs pomiędzy obiektem a użytkownikiem.

Określmy tak zwane drogi procesu wymiany informacji:

- abonent obiektowy → abonent obiektowy (AP→AZ, AZ→AP). Na tej drodze odbywa się wymiana informacji związanej ze sterowaniem, wymianą danych, synchronizacją procesów itp.,
- abonent obiektowy → stacja nadrzędna (AP→SN, AZ→SN). Ta droga jest związana ze śledzeniem obiektu, wizualizacją, raportowaniem i synchronizacją procesów,
- stacja nadrzędna → abonent obiektowy (SN→AP, SN→AZ). Jest to droga wydawania poleceń i rozkazów , a także synchronizacji procesów.

Aby określić kryteria doboru protokołu, niezbędne jest wykonanie:

- klasyfikacji sieci względem specyfiki ruchu,
- określenia wymogów czasowych transmisji,
- określenia sprawności i przepustowości protokołu.

Każdy protokół komunikacyjny służy do przesyłania informacji. Przesyłanie to może odbywać się na wiele sposobów, różnicując między sobą protokoły. Ale każdy z nich generuje w sieci ruch w postaci różnego typu transakcji obsługujących zarówno informacje użyteczne, jak i informacje stanowiące narzut samego protokołu. Przyglądając się ogólnie ruchowi w sieci, można spróbować określić jego specyfikę i intensywność. Specyfika ruchu ma związek z rodzajem i drogami wymian, a także z rodzajem abonentów. W celu określenia intensywności natomiast niezbędne jest określenie punktu odniesienia.

Ilość transmitowanych przez protokół informacji na jednostkę czasu może być stała, a zatem ruch ma wówczas charakter stabilny, czyli posiada stan ustalony. Występuje to dla wymian cyklicznych przy braku generacji wymian aperiodycznych. Sytuacja taka przeważnie zachodzi dla dróg: AP→AZ, AZ→AP, AZ→AZ, AP→SN, AZ→SN.

Może się zdarzyć, że ilość transmitowanych informacji na jednostkę czasu jest stała, lecz dopuszcza się chwilowe zwiększenie ruchu wywołanego wymianami aperiodycznymi. Sytuacja taka może zajść właściwie dla wszystkich dróg: AP→AZ, AP→SN, AZ→AP, AZ→AZ, AZ→SN, SN→AZ, SN→AP.

Kiedy ilość transmitowanych informacji na jednostkę czasu jest stała, lecz dopuszcza się zaistnienie chwilowego zwiększenia ruchu, wymuszonego zmianą określonego stanu procesu oraz zjawisko to może być powielane od wielu obiektów w tym samym czasie, mamy do czynienia z lawiną zdarzeń. Lawinę taką generują wymiany aperiodyczne, których liczba zależy od zewnętrznego wektora stanu obiektu lub wewnętrznego wektora stanu aplikacji abonenta żądającego wymiany. Sytuacja taka przeważnie zachodzi dla dróg AP→AZ, AP→SN, AZ→AZ, AZ→SN.

Otrzymaliśmy zatem dwie charakterystyki ruchu:

- stan ustalony,
- stan ustalony z możliwością wystąpienia lawiny zdarzeń.

Dla rozważań o intensywności ruchu należy przyjąć założenie, iż mówimy o intensywności w stanie ustalonym. Intensywność ruchu w stanie nieustalonym jest zmienna i nie sposób jej klasyfikować. Punktem odniesienia do określenia intensywności ruchu może być maksymalna przepustowość sieci. Korzystając ze wzoru (*42*) w odniesieniu do pojedynczej transakcji danego protokołu, proponuje się zdefiniować maksymalną przepustowość użyteczną sieci jako:

$$P_{MAX} = \max\left(\frac{l}{T_S + T_U + T_O}\right)_{l=l_{min}}^{l_{max}} \quad [b/s], \qquad (102)$$

gdzie: l – jest liczbą transmitowanych bitów użytecznych,

l_{max} – jest maksymalną liczbą bitów użytecznych, która może podlegać transmisji z użyciem danej transakcji,

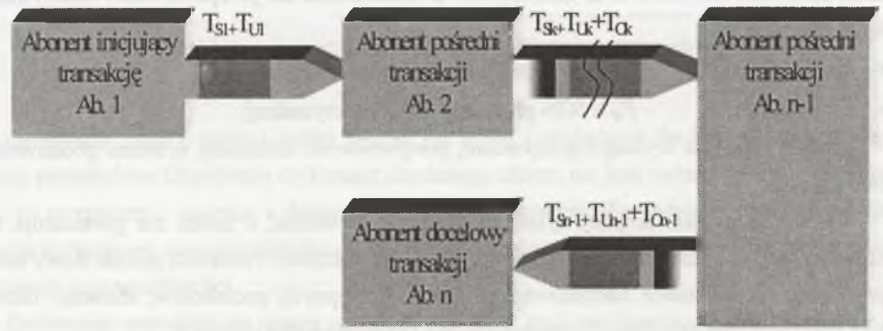
l_{min} – jest minimalną liczbą bitów użytecznych, podlegającą transmisji,

T_S – jest czasem transmisji bitów systemowych, stanowiących narzut warstw sieciowych, a niezbędnych dla zrealizowania transakcji,

T_U – jest czasem transmisji l bitów użytecznych,

T_O – jest czasem oczekiwania na odpowiedź i dla przypadku pesymistycznego jest maksymalnym czasem oczekiwania.

Powyższe czasy stanowią sumę wszystkich czasów składowych każdej wymiany występującej w danej transakcji. Na rysunku 53 przedstawiono uogólniony schemat transakcji wraz z czasami składowymi.



Rys.53. Uogólniony schemat transakcji
Fig. 53. The general diagram of transaction

Oznaczenia na rysunku reprezentują odpowiednio:

$T_{O2}..T_{On}$ – czasy opóźnień dla poszczególnych wymian,

$T_{UI}..T_{Un}$ – czasy transmisji bitów użytecznych dla poszczególnych wymian,

$T_{SI}..T_{Sn}$ – czasy transmisji bitów systemowych dla poszczególnych wymian.

Dla przypadku pesymistycznego $T_{OI}..T_{On} = const = (n\text{-}2)T_{Omax}$, gdzie T_{Omax} jest maksymalnym czasem oczekiwania na reakcję abonenta. Zatem czas oczekiwania dla transakcji wynosi:

$$T_O = \sum_{i=0}^{n-2} T_{Oi} \qquad (103)$$

Czas transmisji bitów użytecznych dla transakcji wynosi:

$$T_U = \sum_{i=0}^{n-1} T_{Ui} \qquad (104)$$

Czas transmisji bitów systemowych dla transakcji wynosi:

$$T_S = \sum_{i=0}^{n-1} T_{Si} \qquad (105)$$

Obliczanie maksimum jest zbędne dla protokołów, gdzie czas transmisji bitów systemowych jest stały. Wówczas należy brać pod uwagę wartość l_{max} w liczniku ilorazu (*102*).

Otrzymana maksymalna przepustowość użyteczna, ze względu na swój wymiar, może być traktowana jako prędkość transmisji bitów użytecznych, czyli użyteczna szybkość transmisji. Wielkość ta informuje, ile maksymalnie bitów użytecznych dany protokół jest w stanie przetransmitować pomiędzy abonentami w czasie jednej sekundy. Dzięki określeniu maksymalnej użytecznej przepustowości uzyskuje się możliwość oszacowania intensywności ruchu względem danego typu protokołu. Sama intensywność ruchu może być wyrażona przez wymaganą przepustowość transmisji. Możemy zatem umownie przyjąć podział intensywności ruchu na niewielki oraz znaczny, co w odniesieniu do przepustowości maksymalnej będzie oznaczało:

- ruch niewielki – $P_W < 50\%$ przepustowości maksymalnej,
- ruch znaczny – $P_W \geq 50\%$ przepustowości maksymalnej,

gdzie P_W określa wymaganą użyteczną przepustowość transmisji systemu poddawanego klasyfikacji.

Rozwiązanie takie dotyczy tylko konkretnego protokołu, a zatem nie gwarantuje nam uniwersalnego punktu odniesienia dla wszystkich protokołów. Ponieważ jednak dobór należy prowadzić na podstawie zdefiniowanego zbioru dostępnych protokołów, możemy określić taki zbiór, a zatem przedział dostępnego pasma przepustowości zbioru:

$$\min\left(P_{MIN\,i}\right)_{i=1}^{n} \leq P_W \leq \max\left(P_{MAX\,i}\right)_{i=1}^{n}, \qquad (106)$$

gdzie: P_{MINi} – określa przepustowość minimalną, czyli minimalną liczbę bitów, jaką dany protokół może transmitować w jednostce czasu. W skrajnym przypadku wynosi ona zero,

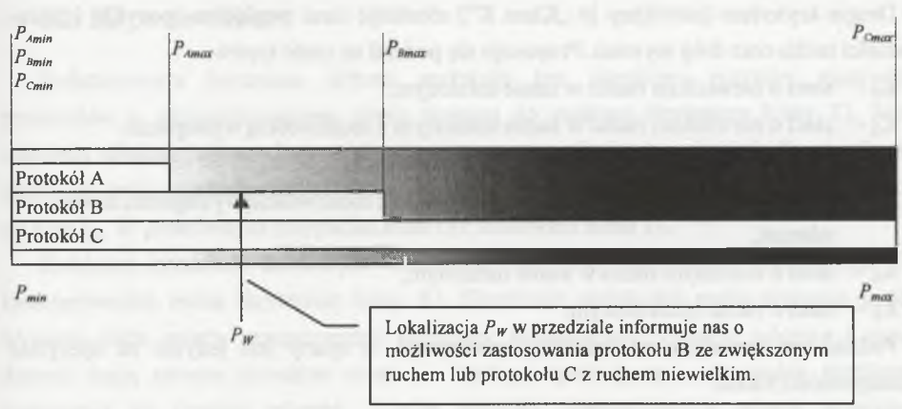
n – określa liczbę protokołów w zbiorze.

Analogicznie do podziału wcześniejszego możemy określić klasyfikowany ruch na podstawie wymaganej użytecznej przepustowości i zlokalizować go w otrzymanym skrośnym zbiorze przepustowości, po czym przydzielić go do jednej z dwóch kategorii:

- ruch niewielki:

  przepustowość minimalna zbioru $< P_W <$ 50% przepustowości maksymalnej zbioru,
- ruch znaczny:

  50% przepustowości maksymalnej zbioru $\leq P_W \leq$ przepustowość maksymalna zbioru.



Rys.54. Przykładowy przedział przepustowości dla trzech protokołów
Fig. 54. An example of flow capacity range for three protocols

Określenie intensywności ruchu odnosi się tylko i wyłącznie do branego pod uwagę zbioru protokołów. Określenie wykonane dla danego zbioru nie jest ważne dla zbioru innego. Jest to oczywiste i wynika z faktu, iż dana przepustowość dla jednego protokołu może stanowić fragment jego możliwości, a dla drugiego znaleźć się na granicy lub nawet poza zakresem jego możliwości.

Dobierając protokoły do zbioru protokołów, należy brać pod uwagę dowolne protokoły o zróżnicowanych przepustowościach maksymalnych i w jak największej liczbie.

8.1. Klasyfikacja sieci

Na podstawie powyższych założeń można stworzyć dwie klasyfikacje sieci przemysłowych. Pierwsze kryterium (nazwijmy je „Klasa T") różnicuje sieci względem niezwykle istotnego podziału protokołów, jakim jest podział wymuszony wymaganym czasowym charakterem wymian.

T_1 – protokół zdeterminowany czasowo,

T_2 – protokół niezdeterminowany czasowo.

Sieci klasy T_1 muszą być stosowane wszędzie tam, gdzie czas dostępu do medium, a zatem czas doręczenia informacji, ma istotne znaczenie. Klasa T_2 jest dopuszczalna dla zastosowań, gdzie czas transmisji nie musi być ściśle określony, a opóźnienia mogą zmieniać się dynamicznie. Można przyjąć, że dla klasy T_2 mamy zawsze do czynienia z wymianami aperiodycznymi. Wynika to z faktu, iż nawet jeśli wymiana jest generowana z określonym cyklem, to protokół nie gwarantując realizacji w określonym czasie, wprowadza dynamiczne zachwiania cyklu, a zatem wymiana zyskuje charakter aperiodyczny.

Drugie kryterium (nazwijmy je „Klasa K") różnicuje sieci względem specyfiki i intensywności ruchu oraz dróg wymian. Proponuje się podział na sześć typów:

K_1 – sieci o niewielkim ruchu w stanie ustalonym,

K_2 – sieci o niewielkim ruchu w stanie ustalonym z możliwością wystąpienia zwiększonego ruchu w specyficznych sytuacjach,

K_3 – sieci o niewielkim ruchu w stanie ustalonym z możliwością wystąpienia lawiny zdarzeń,

K_4 – sieci o znacznym ruchu w stanie ustalonym,

K_5 – sieci o ruchu nieustalonym.

Podział ten abstrahuje od dziedziny zastosowań, a oparty jest jedynie na specyfice i intensywności ruchu.

Klasa K_1 opisuje sieci, w których wykorzystywana przepustowość jest dużo mniejsza od możliwości, jakie daje protokół. Klasa zakłada ruch ustalony, a zatem nie ma możliwości pojawienia się wymian aperiodycznych. Oznacza to, iż sieć jest przewymiarowana. Zastosowanie sieci klasy K_1 ma jedynie uzasadnienie z punktu widzenia rozbudowy systemu i późniejszego zwiększenia ruchu lub zmniejszenia prawdopodobieństwa kolizji w protokołach rywalizacyjnych.

Klasa K_2 opisuje sieci, w których przewiduje się możliwość obsługi aperiodycznej. Przy konfiguracji scenariusza wymian należy brać pod uwagę czas potrzebny do realizacji wymian aperiodycznych. Czas ten można wyznaczyć z różnicy przepustowości maksymalnej i przepustowości stanu ustalonego. Powinien on być tak dobrany, aby możliwa była realizacja wymian aperiodycznych.

Klasa K_3 opisuje sieci, w których przewiduje się możliwość wystąpienia lawiny wymian aperiodycznych. Czas obsługi aperiodycznej powinien być tak dobrany, aby możliwa była realizacja wymian aperiodycznych dla przypadków pesymistycznych, czyli przypadków lawinowej generacji żądań.

Klasa K_4 opisuje sieci, w których przewiduje się zwiększony ruch na poziomie ustalonym. Przypadek taki występuje dla prawidłowo dobranych sieci, w których nie przewiduje się obsługi wymian aperiodycznych oraz rozbudowy.

Klasa K_5 opisuje sieci realizujące tylko wymiany aperiodyczne lub o protokołach rywalizacyjnych. Tego typu sieci powinny być stosowane tylko w przypadkach, gdy dostarczenie danych nie jest obarczone rygorami czasowymi, na przykład w procesach wolnozmiennych.

Analizując projektowany system wymiany informacji i wyodrębniając parametry, w postaci specyfiki i czasowego charakteru wymian oraz wymaganej przepustowości użytecznej, można przydzielić poszukiwane rozwiązanie do powyższych klas, a następnie dobrać konkretne rozwiązanie.

8.2. Kryteria doboru

Podstawowym kryterium doboru protokołu jest określenie potrzeby stosowania protokołów o zdeterminowanym czasie dostępu do medium (kryterium klasy T). Jeżeli charakter obsługi informacji w sieci nie wymaga przesyłania danych w ściśle określonym czasie lub ściśle zdefiniowanym przedziale czasu, to możemy przydzielić nasze rozwiązanie do klas T_2. W przeciwnym przypadku musi być stosowana klasa T_1.

Kolejnym kryterium doboru jest określenie klasyfikacji rozwiązania według specyfiki i intensywności ruchu (kryterium klasy K). Określenie stabilności ruchu wymaga analizy wymian, jakie należy przeprowadzić pomiędzy abonentami. Cykliczne odczyty i zapisy danych mają zawsze charakter ustalony. Obsługa aperiodyczna wprowadza możliwość pojawienia się lawiny zdarzeń. Liczbę wymian aperiodycznych można oszacować w przedziale i na jego podstawie ocenić możliwość wystąpienia zwiększonego ruchu lub lawiny. Przedział szacuje się zwykle pomiędzy zerem – brak żądań realizacji wymian aperiodycznych – a liczbą maksymalną wynikającą z liczby abonentów oraz maksymalnej liczby generacji żądań dla każdego z nich.

Intensywność ruchu w stanie ustalonym można określić na podstawie wymogów, jakie stawia system odnośnie do ilości wymienianej informacji użytecznej w jednostce czasu pomiędzy abonentami dla przypadku pesymistycznego – maksymalnego. Jeżeli określimy sumaryczną liczbę bitów reprezentujących całą informację, jaką należy przesłać przez warstwę sieciową oraz czas całego cyklu wymiany tej informacji (czas realizacji scenariusza wymian), możemy obliczyć średnią przepustowość użyteczną transmisji. Wykorzystanie obliczonej przepustowości do szacowania intensywności ruchu powinno być zadowalające. Można również uwzględnić wymagany czas transakcji dla każdej przesyłanej informacji, obliczyć wymaganą przepustowość i wyszukać wartość maksymalną. Wartość ta może posłużyć do określania klasyfikacji sieci.

Oczywiście, niezbędne jest posiadanie wiedzy dotyczącej maksymalnych i minimalnych prędkości użytecznych dla wszystkich protokołów poddawanych analizie w celu określenia przedziału.

8.3. Dobór rozwiązania

Reguły prowadzenia doboru rozwiązania są proste. Należy zdefiniować zbiór modeli sieci, na bazie których prowadzony będzie dobór rozwiązania. Definicja taka jest niezbędna w celu wyłonienia przybliżonej grupy protokołów spełniających wymogi analizowanego systemu. Poniżej zamieszczono zbiór przykładowy.

1) Modele zdeterminowane czasowo – klasa T_1
   A) Model Master Slave
      a) Protokół MODBUS
      b) Protokół SNP / SNP-X
      c) Protokół LIN
   B) Model Token
      a) Protokół N10
      b) Protokół Genius - N80
   C) Model PDC
      a) Protokół FIP
      b) Protokół WORLDFIP
      c) Protokół FIPWAY
   D) Modele hybrydowe
      a) Protokół P-Net
      b) Protokół Profibus-PD
      c) Protokół H1, H2
2) Modele niezdeterminowane czasowo – klasa T_2
   A) Modele rywalizacyjne
      a) Protokół NetBEUI + Ethernet
      b) Protokół IPX/SPX/NETBios + Ethernet
      c) Protokół HTTP + TCP/IP + Ethernet
   B) Modele rywalizacyjne z obsługą kolizji
      a) Protokół LonWorks
      b) Protokół CAN

Na wstępie doboru należy dla wszystkich wybranych protokołów określić P_{MIN} oraz P_{MAX}, po czym utworzyć przedział przepustowości zbioru (wzór (*106*)).

Klasę modelu sieci wybieramy na podstawie określenia wymagań czasowych, czyli zakwalifikowania wymagań do klasy T.

Dla dalszych rozważań niezbędne jest określenie lokalizacji przepustowości P_W w przedziale analizowanego zbioru i wyodrębnienie protokołów spełniających wymagania czasowe tej przepustowości (rys. 54) oraz należące do wybranej wcześniej klasy T. Określenie przynależności naszego systemu komunikacyjnego do klasy K oraz określenie przynależności protokołów uzyskanych ze zbioru i skorelowanie tych informacji umożliwi wybór najlepszego protokołu w celu realizacji systemu komunikacyjnego rozważanego procesu.

Jest oczywiste, że aby podać obie wielkości P_{MIN} i P_{MAX}, należy wykonać analizę wszystkich protokołów wchodzących w skład klas T_1 i T_2, pod kątem ich przepustowości

użytecznej. W tym celu dla każdego protokołu należy określić zjawiska zachodzące na styku Aplikacja↔Koprocesor↔Protokół (patrz rozdziały 3, 4, 5, 6), aby następnie wskazać ilościowe narzuty czasowe wnoszone przez każdy z protokołów, co z kolei pozwoli wyliczyć największą i najmniejszą przepustowość użyteczną (P_{MIN}, P_{MAX}).

8.4. Podsumowanie

Zaprezentowany w niniejszym rozdziale sposób podejścia do klasyfikacji sieci przemysłowych i ruchu danych jest częścią rezultatów prac zespołu zajmującego się problematyką sieci przemysłowych w Instytucie Informatyki Politechniki Śląskiej w Gliwicach. Prezentowany materiał stał się bazą do powstania modułu programowego, dołączonego do systemu wizualizacyjnego „KRONOS", którego autorem jest mgr inż. Piotr Gaj. Celem owego modułu jest szacowanie jakości doboru protokołu komunikacyjnego do procesu przemysłowego. Jest to dopiero początek drogi, która powinna doprowadzić do ułatwienia i przyspieszenia etapu projektowania i konfiguracji przemysłowych systemów komunikacyjnych, będących podstawą rozproszonych przemysłowych systemów czasu rzeczywistego. W systemach o silnych wymaganiach czasowych proces projektowania i konfiguracji zajmuje ciągle zbyt wiele czasu i jest bardzo kosztowny. Cały szereg narzędzi, które już powstały, jest niezwykle pomocny, ale niestety nie wystarczający. Jak na razie dzięki nim stworzone systemy pracują niezawodnie, ale proces projektowania uległ niewielkiemu przyspieszeniu w stosunku do wymagań.

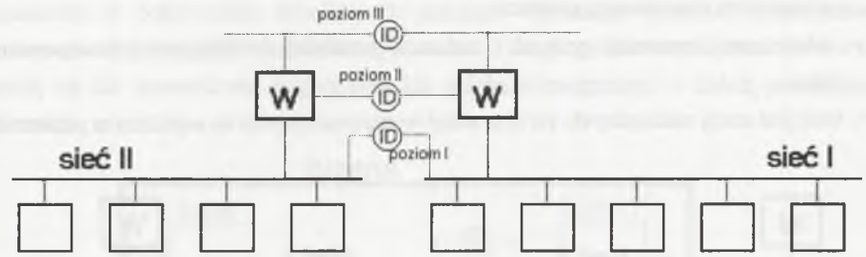
9. METODY INTEGRACJI SIECI PRZEMYSŁOWYCH NAJNIŻSZEGO POZIOMU

Przez integrację sieci najniższego poziomu należy rozumieć takie działanie, które umożliwia wymianę informacji pomiędzy istniejącymi lub dobudowywanymi (rozszerzenia) segmentami sieci, które mogą być względem siebie heterogeniczne lub homogeniczne. Opisywane działanie dotyczy rozszerzenia ruchu informacji zarówno w poziomie, a więc pomiędzy węzłami sieci tego samego poziomu, jak i w pionie pomiędzy węzłami różnego poziomu.

Zarówno sposób integracji, jak i jej zasadność będą zależały od wielu czynników. Głównym bowiem celem integracji powinno być spełnienie wymagań systemu komunikacyjnego, a szczególną uwagę należy zwrócić na determinizm czasowy, jeśli takowy jest wymagany [58]. Podejście projektantów przemysłowych systemów informatycznych do zagadnień integracji sieci, w odniesieniu do sieci najniższego poziomu, uległo w ostatnich latach zmianie. Do niedawna, wskutek fragmentarycznego projektowania instalacji informatycznych, prawie zawsze przy rozbudowie takiego systemu występował problem zintegrowania istniejących fragmentów w jedną całość. Obecnie integracja fragmentów systemów odbywa się najczęściej z dala od poziomu procesu przemysłowego, aczkolwiek dość często występuje konieczność integracji na poziomie najniższym. I znów pojawia się problem determinizmu czasowego, który w odniesieniu do integracji sieci jest związany z następującymi parametrami:

- czasem cyklu sieci, który jest odstępem czasowym pomiędzy transmisjami tego samego abonenta,
- czasem wymiany informacji, który jest minimalnym czasem potrzebnym do wymiany całej informacji pomiędzy wszystkimi abonentami w sieci, i jest wielokrotnością czasu cyklu sieci,
- czasem dostępu do łącza, który jest maksymalnym gwarantowanym czasem, po którym abonent na pewno będzie miał możliwość wysłania przynajmniej części danych,
- przepustowością sieci definiowaną jako iloraz liczby danych użytkowych w pojedynczej transakcji wymiany do całkowitego czasu pojedynczej transakcji wymiany (wzór (*42*)),

- sprawnością sieci definiowaną jako iloraz czasu transmisji danych użytkowych w pojedynczej transakcji wymiany do całkowitego czasu pojedynczej transakcji wymiany (wzór (*41*)).



Rys.55. Trzy możliwe poziomy integracji sieci przemysłowych
Fig. 55. The three levels of industry network integration

Na rys. 55 przedstawiono trzy metody realizacji integracji sieci przemysłowych. Podział ten polega na wprowadzeniu poziomów integracji (poziom I, poziom II, poziom III), związanych z miejscem zastosowania urządzeń i programów łączących sieci przemysłowe. Wybór którejś z nich będzie uwarunkowany zarówno wymaganiami obiektu, jak i celem samej integracji.

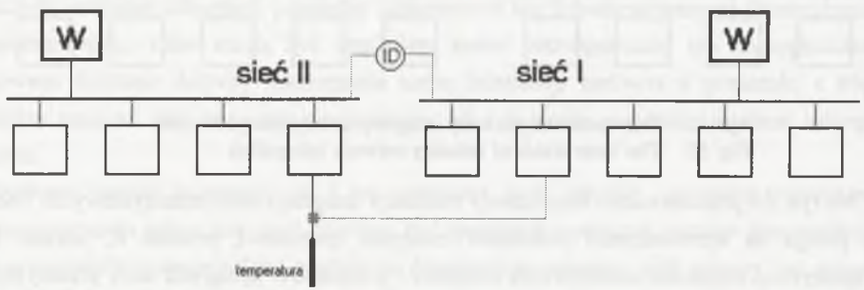
9.1. Integracja sieci na poziomie I

Przypadkami, w których mamy do czynienia z niezbędnością integracji sieci na tym poziomie, są:

- konieczność rozbudowy istniejącego systemu informatycznego o dodatkowy moduł infrastruktury komunikacyjnej wymagający zastosowania na przykład nowego segmentu sieci tego samego poziomu. Ma to miejsce wtedy, gdy nie jest możliwe podłączenie abonenta lub abonentów do istniejącego już segmentu sieci, ponieważ:
  - przyłączenie do istniejącego segmentu sieci nowego abonenta powoduje takie zmiany w scenariuszu wymian, że sieć przestaje spełniać narzucone wymagania czasowe,
  - w istniejącym segmencie sieci przemysłowej występują abonenci, którymi są na przykład sterowniki swobodnie programowalne takich typów i o takich parametrach, których, potocznie mówiąc, albo nie da się już kupić, albo też adaptacja powszechnie stosowanych sterowników do istniejącego systemu komunikacyjnego pociągnęłaby za sobą znaczny wzrost kosztów, lub nieakceptowane pogorszenie parametrów czasowych sieci,

– dla istniejącego segmentu sieci łączna długość medium transmisyjnego może przekroczyć dopuszczalną wartość bądź może być przekroczona maksymalna liczba abonentów w segmencie sieci,

- konieczność udostępnienia kilku dodatkowych sygnałów z obiektu innym istniejącym segmentom tego samego poziomu,
- konieczność transmisji poleceń i rozkazów pomiędzy istniejącymi już segmentami sieci,
- brak jest stacji nadrzędnych, (w tym stacji wizualizacyjnych) na najniższym poziomie.

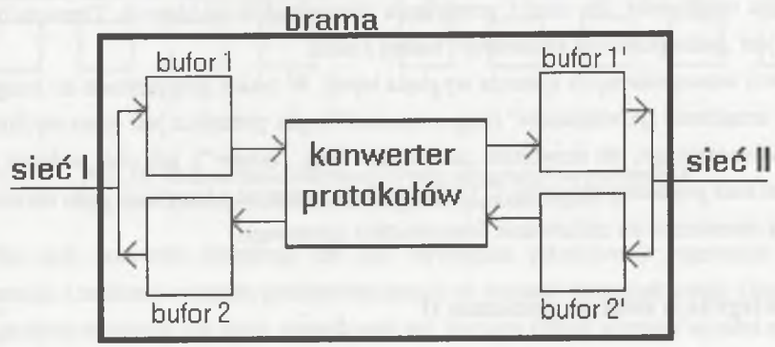


Rys.56. Integracja sieci na najniższym poziomie
Fig. 56. The lowest level network integration

Najbardziej istotnym czynnikiem wpływającym na sposób integracji jest wystąpienie gwarantowanego czasu dostępu do systemu komunikacyjnego.

Na rys. 56 przedstawiono integrację na poziomie I dwóch istniejących segmentów sieci z zaznaczonym dodatkowym sygnałem pomiarowym (tu: temperatura), który ma zostać udostępniony innemu segmentowi sieci.. Zazwyczaj sieć I, jak i sieć II stanowią, a przynajmniej powinny stanowić, jeżeli projekt był poprawnie wykonany, dwa całkowicie autonomiczne podsystemy komunikacyjne, pomiędzy którymi zazwyczaj nie jest wymagana wymiana informacji sterującej. W przypadku pojawienia się jednak nowych pomiarów lub stanów obiektu i konieczności przesłania ich do drugiego segmentu sieci można rozważyć albo integrację sieci, albo podwoić liczbę wejść. Pierwszym kryterium, które należy wziąć pod uwagę, jest zachwianie parametrów czasowych. Niezbędna jest analiza możliwości integracji sieci pod względem zachowania determinizmu czasowego. Jeśli wyniki analizy wykluczają integrację, pozostaje tylko dublowanie stanów wejść, jeśli nie, to decyduje kryterium opłacalności, bo być może integracja okaże się tańszym rozwiązaniem. Ma to miejsce wtedy, gdy mamy do czynienia z trudnymi warunkami technicznymi na obiekcie (znaczne odległości zadajników sygnałów, trudny dostęp).

Innym przypadkiem, w którym integracja na tym poziomie sieci przemysłowych może okazać się konieczna, jest sytuacja, gdy na obiekcie brak jest stacji nadrzędnych i występują sieci o różnych właściwościach (sieci heterogeniczne). Ze względu na brak globalnego standardu sieci przemysłowej nie wszystkie urządzenia (pomiarowe i wykonawcze) są wyposażone w odpowiednie interfejsy do jednego, wybranego typu sieci przemysłowej. Konieczne staje się zastosowanie kilku typów sieci i ich programowe połączenie, ale w taki sposób, by nie spowodować pogorszenia ich działania związanego z ilością przesyłanych informacji.



Rys. 57. Struktura urządzenia pośredniczącego
Fig. 57. The structure of interconnection device

Realizacja programowa integracji nie może być uniwersalna. Główną cechą sieci przemysłowych jest gwarantowany czas wymiany informacji. Patrząc na schemat budowy urządzenia pośredniczącego (bramy), widać, że czas potrzebny na obsługę programową integracji zależy od czasu realizacji funkcji konwersji protokołów, sposobu obsługi i ilości buforów z danymi i wielokrotności czasów działania każdej z obsługiwanych sieci.

$$T_{bramy} = f\left(T_{konw}, T_{buf_we}, T_{buf_wy}, n1 * T_{siecI}, n2 * T_{siecII}\right), \qquad (107)$$

gdzie: T_{bramy} – jest opóźnieniem globalnym związanym z siecią,

T_{konw} – jest czasem potrzebnym na konwersję protokołów,

T_{buf_we} *i* T_{buf_wy} – jest czasem związanym z obsługą buforowania informacji,

T_{siecI} *i* T_{siecII} – są to odpowiednio czasy cyklu sieci I i II,

n1 i n2 – są liczbami cykli pracy sieci potrzebnych do zakończenia realizacji wybranej wymiany.

Czas obsługi bramy jest tym większy, im więcej różnych typów sieci przemysłowych umie ona obsługiwać, im pełniejszy (dokładniejszy) jest proces konwersji protokołów. Można powiedzieć, że im większe różnice w łączonych sieciach, tym czas obsługi ich łączenia

jest większy. Różne sposoby adresowania, różne typy ramek i danych, a nawet różnice prędkości pracy powodują rozbudowę (a także opóźnienie) programu integrującego.

Należy zastanowić się nad sposobami zmniejszenia opóźnień. W przypadku integracji sieci heterogenicznych niewiele można zrobić. Pewnym sposobem może być zaproponowanie własnego typu kodowania ramek, adresowania i przeniesienie warstwy aplikacji (konwertera protokołów) z urządzenia integrującego do wszystkich abonentów. Uzyskamy poprawę szybkości działania bramy kosztem zwolnienia wykonywania programu u pozostałych abonentów. W praktyce integracja taka jest sensowna tylko przy znacznym ograniczeniu możliwości obu sieci i przesyłaniu niewielkiej ilości danych. Oznacza to, że brama nie jest „pełnoprawnym abonentem" żadnej z sieci.

Dla sieci monogenicznych sytuacja wygląda lepiej. W takich przypadkach do integracji wystarczy urządzenie „powtarzające" (ang. *„repeater"*), gdy potrzebne jest tylko wydłużenie medium transmisyjnego, lub urządzenie „adresujące" (ang. *„router"*), gdy obie podsieci mają na przykład inne prędkości transmisji [21]. Opóźnienia związane z integracją są tu niewielkie i wpływają nieznacznie na zachowanie determinizmu czasowego.

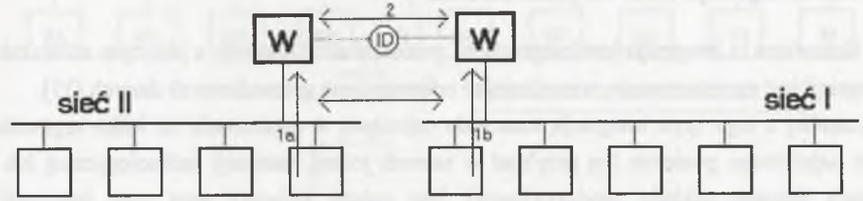
9.2. Integracja sieci na poziomie II

Na poziomie II wymiana informacji następuje bezpośrednio pomiędzy stacjami nadrzędnymi (wizualizacyjnymi) poszczególnych istniejących segmentów. Konieczność zachowania determinizmu zależy nie tylko od zastosowanej sieci, ale również od charakteru danych nią transmitowanych. Na przykład, na poziomie najniższym wymagany jest determinizm czasowy, bo niezbędny jest przepływ rozkazów pomiędzy sieciami przemysłowymi. Jeżeli taka sytuacja nie zachodzi, to integracja nie wymaga gwarantowanych czasów przesyłu informacji.

Integracją na poziomie II będziemy nazywać integrację na poziomie stacji nadrzędnych (np. SCADA). I tutaj sposób realizacji zależy od wymaganego lub nie, gwarantowanego czasu dostępu do łącza. W zależności od przeznaczenia i wymagań czasowych integracja może być wykonana przez połączenie stacji nadrzędnych (komputerów) sieciami zapewniającymi lub nie, określony czas dostępu do łącza (np. za pomocą RPC, OLE, protokołów TCP/IP - MODBUS OVER TCP/IP, Master-Slave, Token-Bus, PDK).

W tym przypadku integracja polega nie tylko na możliwości przesyłania informacji dla procesów sterowania, ale także na gromadzeniu danych celem ich wizualizacji i przetwarzania. Powstają tu problemy związane z zachowaniem spójności danych, synchronizacją, rejestracją zdarzeń (czasy pojawienia się informacji, czasy zmian procesów technologicznych na obiekcie, działania operatorów), występowaniem determinizmu. Jeżeli na obiekcie występują dwie stacje nadrzędne, to gromadzona przez nie informacja powinna być jednakowa pod względem wartości, czasu pozyskania (ang. *"sampling"*) oraz zmian w

czasie (*ang. „time stamping"* [2]). Jeżeli dodatkowo występuje konieczność integracji z uwzględnieniem determinizmu do celów sterowania, stacje nadzorcze powinny przechowywać historię działań związanych ze sterowaniem [102].



Rys. 58. Gromadzenie informacji w celu integracji - na poziomie II
Fig. 58. The data collection in integration process – level II

Na tym poziomie integracji nie jest wymagana jakakolwiek ingerencja w proces transmisji i realizacji wymian przeprowadzanych w sieciach przemysłowych. Oznacza to, że ten poziom integracji nie musi powodować ani wzrostu liczby wymian w sieci najniższego poziomu, ani jakiejkolwiek zmiany charakteru transmisji. Informacje, jak widać, nie są przesyłane bezpośrednio pomiędzy sieciami przemysłowymi.

Każda ze stacji kontrolnych posiada dane potrzebne do prawidłowej pracy fragmentu zarządzanego obiektu. Celem integracji będzie uzyskanie pełnego obrazu stanu obiektu w obu stacjach kontrolnych. Na rysunku 58 wymiany danych oznaczone jako (1a) i (1b) zachodzą podczas normalnej pracy sieci, a za wymiany pomiędzy stacjami kontrolnymi, oznaczone (2), odpowiada ich oprogramowanie. Czas tych wymian nie wpływa na czas pracy sieci przemysłowych. W przypadku zapewnienia determinizmu czasowego dla protokołu łączącego stacje nadrzędne sieci takie można stosować do sterowania, ale po przeanalizowaniu opóźnień w przekazywaniu informacji niezbędnych do sterowania.

Opóźnienie można wyliczyć przyjmując (rys. 58), że do przesłania informacji pomiędzy sieciami wystarczy co najwyżej jeden cykl pracy sieci I, II i sieci pomiędzy stacjami nadrzędnymi. Należy uwzględnić także czasy wykonywania aplikacji stacji nadrzędnych.

$$T_{opóźpóźnie} = f\left(T_{aplikI}, T_{aplikII}, T_{siećWW}, T_{sieć\,I}, T_{sieć\,II}\right), \qquad (108)$$

gdzie: $T_{opóźnienie}$ – jest opóźnieniem związanym z integracją sieci,

T_{aplikI} i $T_{aplikII}$ – są czasami potrzebnymi na obsługę programową aplikacji w stacjach nadrzędnych,

$T_{siećWW}$ – są czasami związanymi z obsługą połączenia pomiędzy stacjami wizualizacyjnymi,

$T_{siećI}$ i $T_{siećII}$ – są to odpowiednio czasy cyklu sieci I i II.

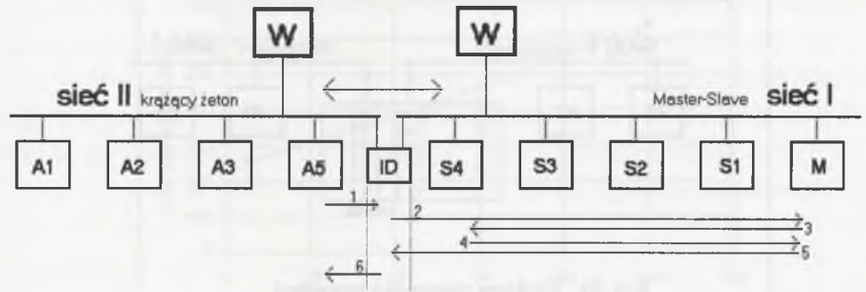
9.3. Integracja sieci na poziomie III

Omawiana tu integracja jest integracją na poziomie sieci lokalnej, a jedynym celem takiej integracji jest monitorowanie, wizualizacja i udostępnianie gromadzonych danych [35]. Najczęściej z tego typu integracją mamy do czynienia w odniesieniu do kilku segmentów sieci najniższego poziomu (na przykład w ramach jednej instalacji technologicznej lub w ramach jednego zakładu produkcyjnego). Nie należy kojarzyć tego typu integracji z możliwościami udostępnienia czy to pełnej wiedzy o obiekcie, czy też dostarczania ewentualnych możliwości sterowania, bądź parametryzacji instalacji na poziomy sieci otwartych (Internet, Intranet). Jest to zagadnienie innej natury, bardzo ważne i niezwykle aktualne, a omówione w rozdziale 10.

Biorąc pod uwagę dostępne protokoły komunikacyjne dla sieci lokalnych, determinizm czasowy nie będzie możliwy do osiągnięcia. Głównym jej zadaniem jest udostępnienie szczegółowej, zgromadzonej i przetworzonej informacji na temat stanu obiektu i jego historii na inne, wyższe poziomy zarządzania produkcją, tworzenia zakładowych baz danych. W celu prawidłowej realizacji tych zadań stacje nadrzędne powinny posiadać odpowiednie oprogramowanie zapewniające zachowanie spójności danych, synchronizację, rejestrację zdarzeń, umożliwienie wielu użytkownikom pobieranie informacji o obiekcie, a także ochronę przed niepowołanym dostępem i mechanizmy dające gwarancję, niezależnie od obciążenia sieci lokalnej, na wymagane czasy do poprawnej pracy programu aplikacyjnego i sieci przemysłowej. Integracja o podobnym charakterze (poziom III), ale z zastosowaniem protokołu TCP/IP jest opisana, jak już wspomniano, w rozdziale 10.

9.4. Przykład integracji

Rozpatrzmy przykład integracji sieci opartych na protokołach Master-Slave i Token-Bus. Integracja polega na wprowadzeniu urządzenia pośredniczącego, umożliwiającego programowe połączenie obu sieci w jeden heterogeniczny system. Program powinien umieć adresować każdego abonenta w zintegrowanej sieci i wysyłać wymagane rodzaje wymian [59].

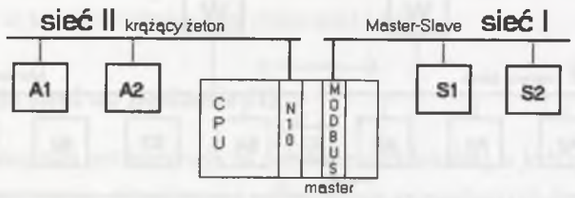


Rys.59. Przesył informacji dla integracji na poziomie sieci przemysłowej
Fig. 59. The integration data transfer on industry network level

W sieciach przemysłowych znaczna część informacji przesyłana jest w trybie zapytanie-odpowiedź. Na rys. 59 informacja (na przykład rozkaz odczytu danych) przesyłana jest od abonenta A5 do urządzenia integrującego ID (ang. *„interconnection device"*). Urządzenie ID musi odczekać cały cykl pracy sieci I, aż stacja Master wyśle do niego ramkę (2) i będzie miało możliwość przekazania informacji z sieci II do stacji Master. Następnie stacja Master komunikuje się ze stacją S4 (3,4), odsyła dane do stacji ID (5), skąd przy następnym dostępie do sieci abonent A5 pobierze dane (6).

Widać, że nawet przy odpowiednim przygotowaniu kolejności wymian w obu sieciach wymagany jest minimum 1 pełny cykl pracy każdej z nich. W praktyce, przy większej ilości wysyłanych danych, 1 cykl na pewno nie wystarczy. Nie można też zapominać, że zwiększy się czas obsługi programowej w urządzeniu ID. W przypadku łączenia sieci o różnych czasach wymiany informacji, w sieci zintegrowanej, czas pełnej wymiany informacji będzie większy od sumy cykli wymiany informacji obu sieci. Oznacza to, że gdy na przykład dla sieci typu Master-Slave (MODBUS) czas cyklu wynosi około 200 ms, a dla sieci typu Token-Bus (N10) około 50 ms, to łączenie tych sieci w sposób pokazany na rys. 58 nie ma sensu, gdyż spowoduje to kilkakrotne opóźnienie szybszej sieci, a tym samym najprawdopodobniej uniemożliwi prawidłową pracę systemu. Ważną sprawą jest też buforowanie transmisji. W przypadku dużych różnic szybkości działania integrowanych sieci informacja z sieci szybszej musi być gromadzona i pamiętana na potrzeby sieci wolniejszej.

W ramach badań przygotowano stanowisko testowe składające się z sieci typu Master-Slave i Token-Bus, tak jak pokazuje to rys. 60.



Rys. 60. Struktura stanowiska testowego
Fig. 60. The structure of the test stand

Obie sieci zostały zintegrowane za pomocą sterownika swobodnie programowalnego (PLC), będącego abonentem każdej z nich, dzięki zastosowaniu w nim specjalizowanych modułów komunikacyjnych i obliczeniowych. Można zatem go traktować jako klasyczną „bramę". Celem testu było zbadanie wpływu czasu realizacji programu obsługującego „bramę" (program egzystujący w jednostce centralnej stacji Master) na czas wymiany informacji w sieci. Przyjęto, dla uproszczenia eksperymentu, założenie zapewniające przepływ danych tylko w jedną stronę. Sieć Master-Slave gromadziła informacje tworząc w pamięci abonenta Master, który jednocześnie był urządzeniem integrującym, bufor zawierający wartości danych użytkowych wszystkich transakcji wymiany obsługiwanych przez sieć. Założenie to spowodowało następujące konsekwencje:

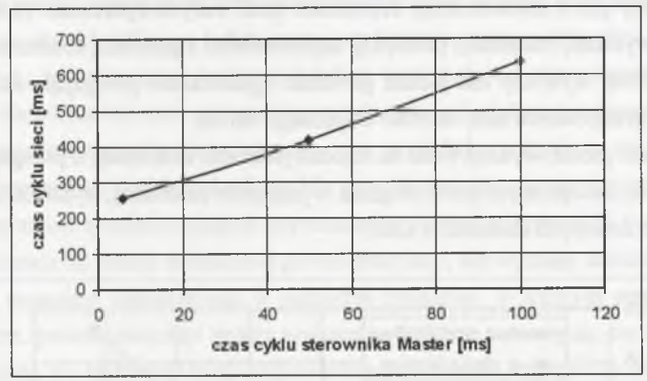
- uproszczenie i redukcję liczby wymian stacji Master, bowiem istotne stają się tylko wymiany przychodzące od strony sieci Token-Bus do sieci Master-Slave,
- uproszczone adresowanie w stacji Master, gdyż stacja Master „nie widzi" sieci Token-Bus,
- brak żądań transmisji od stacji Slave pozwala na uproszczenie programu w stacji Master. Abonent będący stacją Master nie wymaga danych i nie rozpoczyna żadnych transmisji. Odbiera jedynie ramki z sieci Token-Bus, dekoduje je i wysyła ramki odpowiedzi.

Najpierw dokonano badania wpływu czasu wykonywania programu w stacji Master na czas cyklu sieci Master-Slave. Wydłużając realizację programu stacji Master, badano czas transmisji danych z sieci Master-Slave do sieci Token-Bus.

Parametry konfiguracyjne badanej sieci były następujące:

- tryb pracy RTU,
- prędkość transmisji równa 19 200b/s,
- liczba abonentów równa 3.

Wymiany w sieci polegały na odczycie 16 słów 16-bitowych od każdego abonenta.

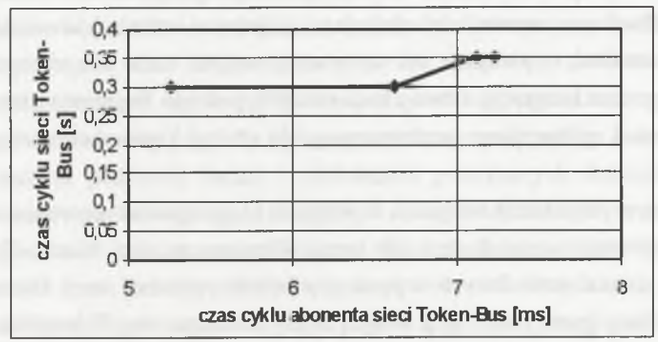


Rys. 61. Zależność pomiędzy czasem cyklu stacji Master a czasem trwania cyklu sieci
Fig. 61. The dependence between cycle time Master station and network cycle time

Dla sieci MODBUS widać wyraźnie (rys. 61) wpływ czasu programu na cykl sieci, co nie jest zaskoczeniem (patrz rozdz.5). Związane jest to głównie z koniecznością pobierania danych przez koprocesor ze wspólnej pamięci, do której jest limitowany dostęp jednokrotnie na cykl pracy podstawowej pętli programu aplikacyjnego.

Kolejnym etapem było sprawdzenie wpływu czasu cyklu abonenta sieci Token-Bus na czas cyklu tej sieci. Dokonano pomiarów dla dwóch typów wymian:

- wymiana transparentna (rys. 62), wykonywana autonomicznie przez koprocesor sieciowy, polegająca na cyklicznym, co 100 ms, rozsyłaniu 1 słowa do innych abonentów sieci,

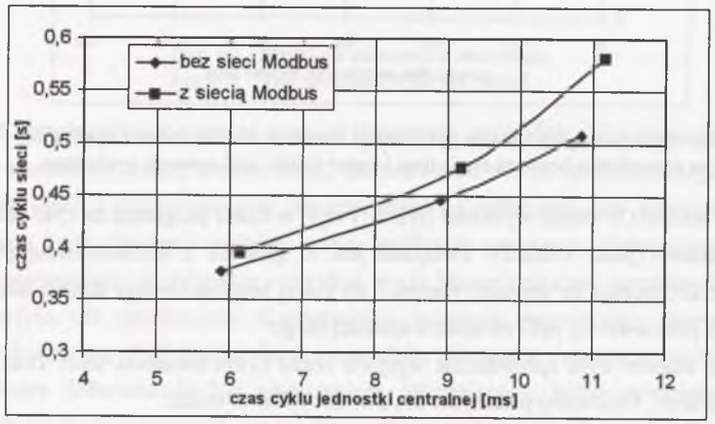


Rys. 62. Wpływ czasu cyklu sterownika na czas cyklu sieci dla wymiany transparentnej
Fig. 62. The influence of PLC cycle on network cycle in transparent exchange

Dla wymiany transparentnej nie widać prostej zależności pomiędzy czasem wykonywania programu a czasem trwania cyklu sieci. Spowodowane jest to realizacją

wymiany przez autonomiczny koprocesor sieci. Jedyne opóźnienie powodowane jest przez wymianę informacji pomiędzy koprocesorem a pamięcią abonenta – „buforem". Ten rodzaj wymiany ma jednak poważne ograniczenie polegające na zmniejszeniu liczby przesyłanych danych tylko do jednego słowa;

- wymiana prosta, wykonywana na żądanie jednostki centralnej, a polegająca na jednokrotnym, zarządzanym przez program w jednostce centralnej, wysłaniu 16 słów 16-bitowych do innych abonentów sieci.



Rys. 63. Wpływ czasu cyklu sterownika na czas cyklu realizacji wymiany prostej
Fig. 63. The influence of PLC cycle on the single exchange cycle

Na rys. 63 widać wpływ działania innej sieci, w tym przypadku sieci Master-Slave na czas wymian informacji w sieci Token-Bus. Integracja, polegająca na stosowaniu specjalizowanych modułów koprocesorów i ich obsłudze w jednostce centralnej abonenta, powoduje wydłużenie czasu sieci, co związane jest raz ze zwiększeniem czasu potrzebnego do obsługi programowej procesu integracji, a dwa z koniecznością podziału fragmentu czasu podstawowej pętli programu aplikacyjnego przeznaczonego do obsługi koprocesorów na obsługę obu sieci.

W omawianym przykładzie integracji, ze względu na poczynione uproszczenia, łatwe jest określenie granicznego czasu dostępu do łącza. Wymiany w sieci Master-Slave dotyczą gromadzenia i uaktualniania danych w pamięci jednostki centralnej stacji Master. Czas ten jest ściśle określony (patrz rozdz. 5). Z drugiej strony, abonenci sieci Token-Bus mogą tylko pobierać dane od abonenta Master, co powoduje, że nie zachodzą wymiany pomiędzy segmentami sieci. Biorąc to pod uwagę, można określić gwarantowany czas cyklu sieci. Dla takiego przypadku integracji możemy powiedzieć, że maksymalny czas cyklu zintegrowanej sieci jest równy sumie maksymalnych czasów cykli w jednej i w drugiej sieci.

Widać, że nawet w przypadku znacznego uproszczenia procesu integracji, polegającego na przesyłaniu danych tylko w jedną stronę, opóźnienie czasu cyklu sieci jest znaczące. Można przyjąć, że najbardziej istotnym źródłem opóźnień jest oprogramowanie w stacji integrującej. Na podstawie rozważań teoretycznych, jak i z badań można zauważyć, że każda rozbudowa oprogramowania integrującego, na przykład polegająca na umożliwieniu wymiany informacji w obu kierunkach, będzie powodować znaczące opóźnienie czasu pracy sieci. Wykorzystanie specjalizowanych koprocesorów do obsługi transakcji wymiany danych, co prawda, ułatwia od strony technicznej proces integracji, ale wymaga nakładów czasowych na realizację programu rezydującego w jednostce centralnej. W każdym przypadku, co nie jest odkrywcze, proces integracji będzie wymagał nakładów czasowych, ale bez wykonania analizy czasowej nie będziemy mogli odpowiedzieć na pytanie, czy proces integracji jest do zrealizowania bez naruszania istotnych parametrów systemu komunikacyjnego.

9.5. Podsumowanie

Proces przekazywania informacji jest zwykle realizowany programowo. Jest więc rzeczą oczywistą, że integracja sieci musi być związana z pogorszeniem parametrów dotyczących czasu wymiany informacji w systemie. Na wydłużenie czasu wymiany informacji w systemie wpływa nie tylko jej ilość, ale również potrzeba programowego zarządzania protokołami transmisji. Jakkolwiek każdy typ sieci ma opracowany protokół transmisji, to z chwilą zintegrowania różnych sieci tego samego producenta konieczne jest opracowanie programu użytkowego, realizującego zarządzanie przepływem informacji pomiędzy dowolnymi abonentami różnych sieci. Zazwyczaj program użytkowy dotyczy procesu buforowania danych podczas transmisji i zarządzania wyszukiwaniem adresata i nadawcy. Wraz ze wzrostem liczby abonentów i rozległością systemu sieciowego czas realizacji programu użytkowego zaczyna mieć istotne znaczenie dla sprawnego realizowania procesów sterowania odbywających się na obiekcie, a więc należy poszukiwać granicy zakresu stosowalności łączenia sieci, tak aby był on prawidłowo realizowany. Z treści niniejszego rozdziału oraz na podstawie badań i wieloletnich praktycznych doświadczeń autora i współpracowników można wyciągnąć następujące wnioski dotyczące procesu integracji:

- integracja sieci najniższego poziomu spowodowana lokalnym rozwojem instalacji, w obecnym stanie rozwoju przemysłowych systemów informatycznych, jest zjawiskiem rzadko spotykanym i ogranicza się jedynie do sporadycznych opisanych w niniejszym rozdziale przypadków,
- istnieje konieczność integracji sieci w związku z wymianą informacji dotyczących procesów wizualizacji, monitorowania i parametryzacji,

– integracja może polegać na łączeniu sieci przemysłowych w celu umożliwienia przepływu danych pomiędzy sieciami lub też tylko na gromadzeniu danych z różnych sieci celem uzyskania informacji do prawidłowego sterowania i monitorowania procesów przemysłowych. Uwaga ta dotyczy systemów lokalnych (jeden proces, terytorium pojedynczego zakładu pracy). Każdy z powyższych sposobów ma inne ograniczenia, dotyczące możliwości jego zastosowania – sterowanie lub tylko wizualizacja.

Problem zakresu stosowalności integracji sieci istnieje bez względu na to, czy będą stosowane specjalizowane urządzenia pośredniczące, czy funkcje integracyjne należeć będą do programów użytkowych. Dotyczy on gwarantowanego czasu wymiany informacji ze względu na bezpieczeństwo sterowania procesami.

Bardzo ważne jest też utrzymanie dotychczasowych parametrów sieci, takich jak szybkość działania, możliwość zmiany parametrów, łatwa rozbudowa, otwartość, niezawodność, redundancja, mechanizmy kontroli komunikacji i gwarantowany czas wymiany informacji. Widać, że jest to dość trudne, biorąc pod uwagę różnorodność sieci przemysłowych, aby w przypadku ich połączenia można było wykorzystać wszystkie cechy integrowanych sieci. Konieczne byłoby odpowiednie oprogramowanie, które potrafi dane z każdej z sieci odpowiednio przesłać do innej. Pojawią się tu problemy związane z typem danych (bit, bajt, słowo, grupa słów), typem wymiany (odczyt słowa, bitu, grupy słów, zapis słowa, bitu, grupy słów), liczbą wymian i skomplikowanym sposobem adresowania poszczególnych sterowników dla przesyłów międzysieciowych. Dla niektórych sieci przemysłowych stosuje się adresowanie abonenta. Na przykład, w sieci Master-Slave każdy abonent typu Slave ma unikalny numer, występują wymiany typu zapytanie-odpowiedź. Dla innych adresuje się informacje. Na przykład, w sieciach typu Producent-Dystrybutor-Konsument wymiany są traktowane jako kolejne rozgłoszenia adresowanych informacji, a procesem tym kieruje jeden z abonentów, tak zwany „dystrybutor" [58].

Wiadomo, że wraz ze wzrostem liczby abonentów i rozległością zintegrowanego systemu sieciowego czas wymiany informacji będzie coraz większy. Składa się na niego czas realizacji programu użytkowego realizowanego przez abonentów (PLC), czas potrzebny na obsługę sieci przez moduł komunikacyjny i zależności wynikające z protokołu transmisji (patrz rozdz. 5, 6, 7). Opóźnienia wynikające z integracji mogą spowodować niemożność sprawnego realizowania procesów sterowania odbywających się na obiekcie, a więc należy poszukiwać granicy zakresu stosowalności łączenia sieci, tak aby był prawidłowo realizowany proces jego sterowania.

10. ZASTOSOWANIE PROTOKOŁU TCP/IP W SIECIACH PRZEMYSŁOWYCH NAJNIŻSZEGO POZIOMU

Bardzo szybki wzrost popularności protokołu TCP/IP jest związany z rozrastającą się w szybkim tempie globalną siecią Internet. Ma to również swoje odbicie w zastosowaniu informatyki w przemyśle [37]. Z jednej strony, przemysłowe systemy informatyczne mają, jak to już wielokrotnie podkreślano w niniejszej pracy, swoją specyfikę i szczególne wymagania dotyczące determinizmu czasowego, z drugiej zaś, wzrost zainteresowania możliwościami protokołu TCP/IP nie ominął także dziedziny informatyki przemysłowej. W tym jednym z bardziej konserwatywnych obszarów zastosowań informatyki obserwuje się coraz częstsze próby wykorzystywania protokołu TCP/IP. Dzieje się tak przede wszystkim dlatego, że zarówno oprogramowanie, jak i sprzęt związany z wykorzystaniem owego protokołu jest bardzo rozpowszechniony, a zatem bardzo tani, żeby nie powiedzieć, że w wielu przypadkach pewne jego elementy (choćby oprogramowanie) są wręcz darmowe. W konfrontacji ze stale rosnącymi cenami szanowanych producentów i ze znacznymi możliwościami integrującymi, protokół ten w połączeniu ze znanymi i sprawdzonymi standardami, jak na przykład: protokołami FIP, PROFIBUS, MAP, czy MODBUS zaczyna być coraz chętniej wykorzystywany w rozwiązaniach sieci przemysłowych. Kolejnym atutem tego protokołu jest dobry stan standaryzacji dzięki normom międzynarodowym oraz czynniki ekonomiczne, do jakich należy zaliczyć brak opłat licencyjnych związanych z jego wykorzystywaniem oraz dostępność rozwiązań sprzętowo-programowych .

Wskutek rosnącej globalizacji również instalacje automatyki przemysłowej zaczynają tracić swoją pierwotną hermetyczność. Początkowo systemy te zamykały się w obrębie jednego bądź kilku segmentów sieci przemysłowych. Na tym poziomie konstruowane były stacje kontrolno-nadzorcze klasy SCADA (ang. *„Supervisory Control and Data Acquisition"*) [102, 22]. W części zastosowań stacje te współpracowały z równie hermetycznymi systemami typu MES (ang. *"Manufacturing Execution System"*) czy systemami ERP (ang. *" Enterprise Resources Planning"*). Pobudki ekonomiczne i rozwijająca się również e-gospodarka powodują, że od przemysłowych systemów sterowania wymaga się w mniejszym bądź szerszym zakresie współpracy z sieciami rozległymi, w tym także coraz częściej z konstruowaną na bazie protokołu TCP/IP siecią Internet.

Wymienione powyżej niewątpliwe zalety protokołu TCP/IP nie przesądzają jednak o możliwości jego bezkrytycznego stosowania jako uniwersalnego środka, zdolnego zaspokoić wymagania każdej instalacji automatyki przemysłowej. Wykorzystywanie w sieciach przemysłowych najniższego poziomu protokołu TCP/IP, jak również integracja tych systemów z sieciami rozległymi i siecią Internet pociąga za sobą wiele nowych, nie występujących w stosowanych dotychczas rozwiązaniach, zjawisk. Niech niniejszy rozdział będzie próbą określenia, tak metod wykorzystania, jak i granic stosowalności protokołu TCP/IP w systemach automatyki przemysłowej, dokonaną z punktu widzenia sieci przemysłowej najniższego poziomu.

10.1. Protokół TCP/IP w sieciach przemysłowych

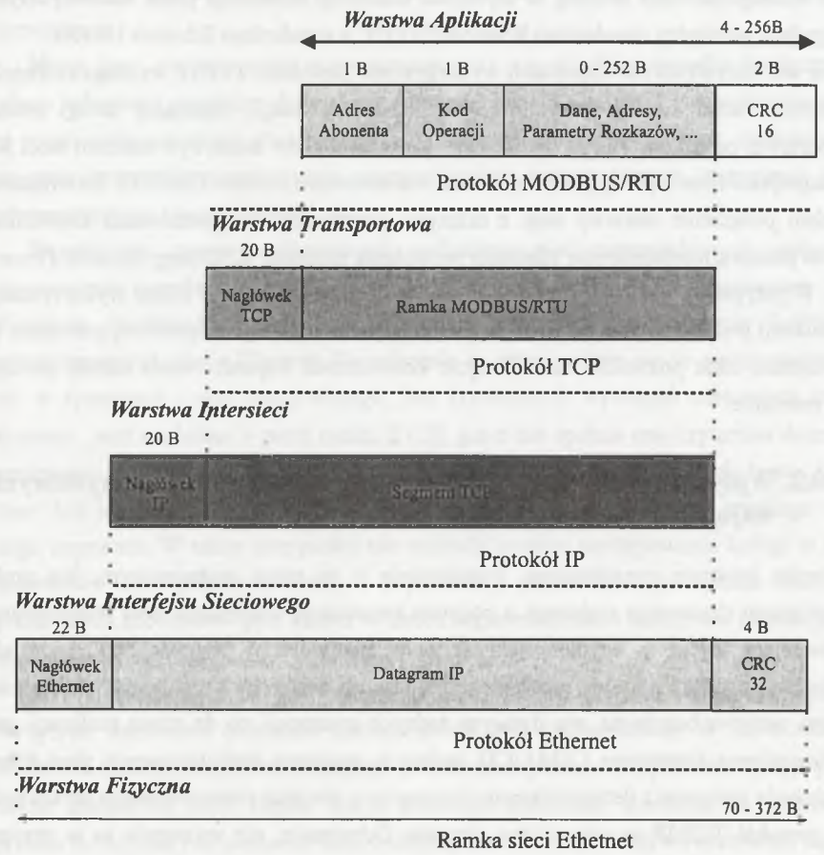
Rosnąca popularność protokołu TCP/IP sprawia, iż znacząca część producentów układów automatyki przemysłowej decyduje się na implementację tego właśnie standardu w swoich urządzeniach. Zazwyczaj rozwiązania wykorzystujące ten protokół łączą sprawdzone w sieciach przemysłowych mechanizmy komunikacyjne z mechanizmami protokołu TCP/IP. Dominuje tu technika tak zwanego kapsułkowania ramek sieci przemysłowej w segmentach TCP. Zagadnienie kapsułkowania ramek transmitowanych w sieciach przemysłowych poprzez protokół TCP zostanie zilustrowane na przykładzie standardu MODBUS/TCP firmy Modicon [122] (rozdz. 5).

Proponowane przez firmę Modicon rozwiązanie MODBUS/TCP [109] zakłada kapsułkowanie ramek zgodnych ze standardem MODBUS/RTU w segmencie TCP. Schemat, według którego realizowane jest wielostopniowe kapsułkowanie, został przedstawiony na rysunku 64.

Na poziomie warstwy aplikacji oprogramowanie odpowiedzialne za obsługę protokołu MODBUS generuje ramki zgodne ze standardem MODBUS/RTU. Po usunięciu z ramki sumy kontrolnej CRC 16 pozostała część kapsułkowana jest w segmencie TCP. Segmenty TCP, zgodnie z warstwową budową stosu protokołów TCP/IP, kapsułkowane są na poziomie Intersieci w datagramy IP. Datagramy IP na poziomie interfejsu sieciowego kapsułkowane są w ramki sieci Ethernet. Warstwa sprzętu zapewnia dostarczenie ramki do adresata. Oprogramowanie komunikacyjne po stronie odbiorcy realizuje odwrotną procedurę. Ramka sieci Ethernet poprzez datagram IP i segment TCP doprowadzana jest do postaci zgodnej ze standardem MODBUS/RTU.

Opisane rozwiązanie techniczne podyktowane zostało względami natury ekonomicznej, bo można wykorzystać istniejące oprogramowanie oraz sprzęt (PLC) dość powszechnie wyposażany w interfejs MODBUS/RTU. Uzupełnieniu podlegają jedynie elementy sprzętowe i moduły programowe zajmujące się kapsułkowaniem ramek sieci MODBUS i transmisją w sieci Ethernet. Oparcie się zatem na dwóch dobrze sprawdzonych i powszechnie

znanych protokołach pozwoliło uniknąć czasochłonnych i kosztownych badań nad nowym protokołem sieci przemysłowej.



Rys. 64. Mechanizm kapsułkowania stosowany w sieci MODBUS/TCP
Fig. 64. The capsulation mechanism in MODBUS/TCP network

Opisane rozwiązanie ma wady, z których najpoważniejsza to narzut czasowy związany z konwersją ramki sieci MODBUS. Ramka ta przenoszona jest „w górę" na poziom warstwy aplikacji i stamtąd poprzez warstwą transportową i warstwę intersieci przekazywana jest do warstwy interfejsu sieciowego. Narzut czasowy związany z operacją kapsułkowania staje się widoczny jeszcze bardziej w przypadku analizy długości danych wymienianych pomiędzy poszczególnymi warstwami. O ile długość ramki definiowanej przez standard MODBUS/RTU wynosi w zależności od przesyłanej treści od 4 (np. odczyt pojedynczego bajtu) do 256 bajtów, to wartości te po przejściu poprzez kolejne poziomy kapsułkowania wynosić będą odpowiednio 22 – 274 bajtów dla segmentu TCP, 42 – 294 bajtów dla datagramu IP i 70 – 372 bajów dla ramki Ethernet.

Długość tak powstałych ramek, wskutek ogromnych narzutów wnoszonych przez kolejne etapy kapsulacji, może wzrosnąć w stosunku do ich pierwotnej długości ponad 1000%. Wada ta jest rekompensowana różnicą, w szybkości transmisji informacji przez warstwę fizyczną, występującą pomiędzy standardem RS485/422/232C a standardem Ethernet 10/100.

Nie we wszystkich rozwiązaniach wykorzystanie protokołu TCP/IP wymaga cofnięcia się z poziomu ramki sieci przemysłowej na poziom aplikacji, zlecającej usługi warstwie transportowej protokołu TCP. Przykładem innej architektury może być standard sieci MAP. Sieć zaprojektowano opierając się na siedmiowarstwowym modelu OSI/ISO. Rozwiązanie to zakładało połączenie warstwy sesji z niższymi warstwami oprogramowania komunikacyjnego za pomocą mechanizmów zdalnego wywołania procedur RPC (*ang. **Remote Procedure Call***). W przypadku realizacji sieci MAP/TCP [61] mechanizm RPC został wykorzystany do komunikacji pomiędzy warstwą sesji protokołu MAP a warstwą transportową protokołu TCP. Rozwiązanie takie pozwoliło na uniknięcie konieczności kapsułkowania ramek na najniższym poziomie.

10.2. Wpływ architektury systemu na parametry sieci przemysłowych najniższego poziomu

Bardzo istotnym zagadnieniem, wielokrotnie w tej pracy podnoszonym, jest problem determinizmu czasowego stojącego u podstaw konstrukcji wszystkich sieci przemysłowych. Zapewniający usługę przesyłania niezawodnymi strumieniami protokół TCP opiera się na pracującym, zgodnie z zasadą przenoszenia w ramach dostępnych możliwości, protokole IP. Kolejna warstwa bazuje na, nie dającym żadnych gwarancji co do czasu realizacji usługi, rywalizacyjnym algorytmie CSMA/CD, będącym podstawą funkcjonowania sieci Ethernet. Zagadnienia związane z determinizmem czasowym w sieciach przemysłowych wykorzystujących protokół TCP/IP są więc dosyć złożone. Odmiennie, niż występuje to w przypadku klasycznych rozwiązań sieci przemysłowych, w rozwiązaniach opartych na protokole TCP/IP bezpieczeństwo sieci zależy od wyższych warstw oprogramowania komunikacyjnego. Patrząc na tak przedstawioną powyżej charakterystykę, nie możemy podać żadnych kryteriów określających w sposób deterministyczny parametry sieci. Opisana w sposób powyższy sieć nie nadaje się więc do wielu zastosowań typu „hard real-time system" (patrz rozdz. 2 i 3).

Sieci przemysłowe wykorzystujące protokół TCP/IP nie mogą być jednak rozpatrywane zarówno w oderwaniu od warstwy zapewniającej realizację usług sieci przemysłowej, jak również w oderwaniu od architektury rozpatrywanego rozwiązania. Rozważając opisane w poprzednim rozdziale rozwiązanie sieci przemysłowej MODBUS/TCP i analizując pracę wydzielonego w sposób fizyczny segmentu tej sieci przemysłowej, można zastosować kryteria pozwalające na określenie w sposób deterministyczny parametrów danej sieci. Zastosowanie

protokołu MODBUS opartego na modelu Master-Slave gwarantuje dla wydzielonego segmentu sieci brak występowania kolizji w warstwie fizycznej sieci Ethernet, a stosując mechanizmy analizy czasowej sieci przemysłowych, można pokusić się o pełną analizę takiego rozwiązania.

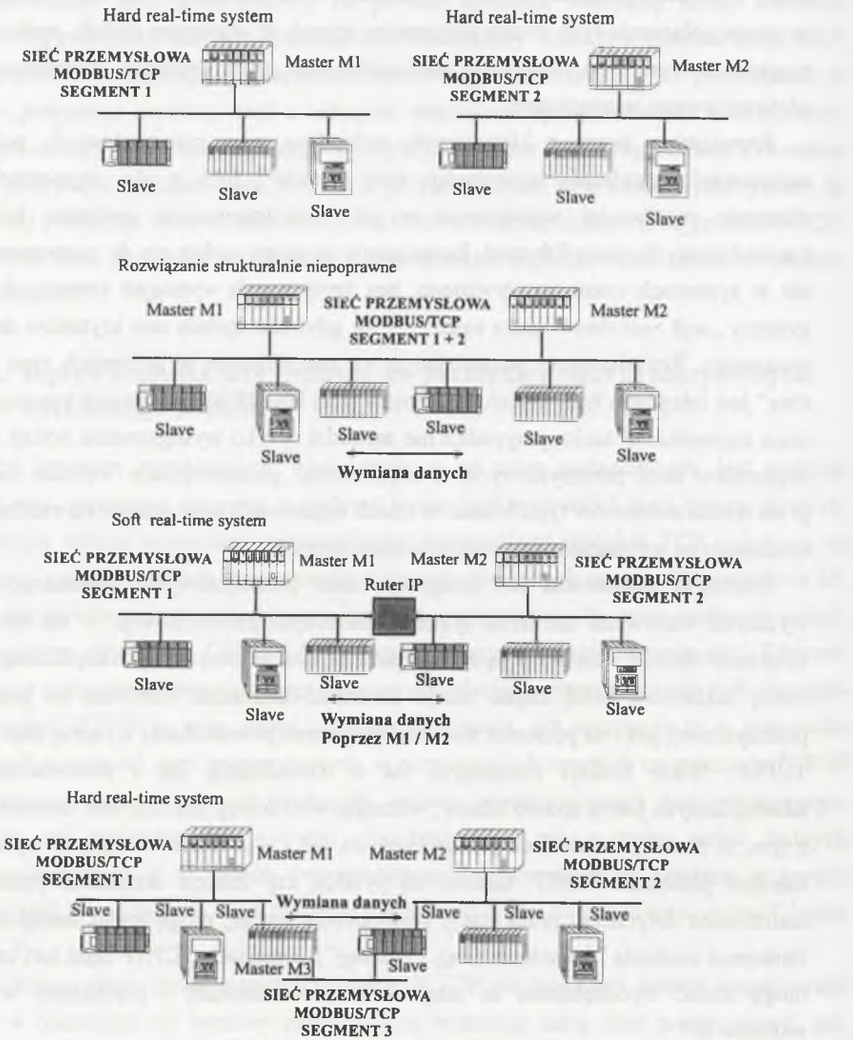
Nieco inny wariant został zaprezentowany na rys. 65. W przypadku konieczności wymiany informacji pomiędzy dwoma segmentami sieci MODBUS/TCP rozwiązanie zakładające proste połączenie tych dwóch segmentów sprawi, iż w miejsce dwóch, spełniających wymagania determinizmu czasowego, segmentów sieci przemysłowych, otrzymamy jeden niezdeterminowany segment sieci.

Rozwiązanie znane z klasycznych architektur sieci przemysłowych, polegające na zastosowaniu urządzenia separującego (ang. *„router"*) ruch w obu segmentach sieci, nie eliminuje możliwości wystąpienia w tak skonstruowanym systemie kolizji ramek transmitowanych siecią Ethernet. Rozwiązanie to zatem nadaje się do zastosowania wyłącznie w systemach czasu rzeczywistego, bez krytycznych wymagań czasowych (tak zwane systemy *„soft real-time"*- patrz rozdz. 2 i 3), gdyż nie spełnia ono kryteriów determinizmu czasowego. Rozwiązaniem pozwalającym na zastosowanie w systemach typu *„hard real-time"* jest integracja tych dwóch segmentów sieci MODBUS za pomocą trzeciego wydzielonego segmentu. W takim przypadku nie zachodzi ryzyko występowania kolizji w żadnym z segmentów sieci przemysłowych, a odpowiednia parametryzacja wymian realizowanych przez trzech abonentów typu Master w trzech segmentach sieci umożliwia realizację systemu spełniającego wymagania determinizmu czasowego.

Osobnym problemem jest integracja sieci przemysłowych najniższego poziomu z wyższymi warstwami hierarchii systemu automatyki przemysłowej. W tak skonstruowanej strukturze funkcję urządzenia łączącego poziom sieci przemysłowych najniższego poziomu z siecią zakładową pełni często stacja kontrolno-nadzorcza. Zarówno na poziomie sieci przemysłowej, jak i na poziomie sieci rozległej coraz powszechniej wykorzystuje się protokół TCP/IP. Wiele funkcji związanych tak z wizualizacją, jak i sterowaniem obiektem udostępnianych jest w sposób zdalny, wliczając w to dostęp poprzez sieć Internet. W związku z tym, iż zarówno w sieciach przemysłowych, jak i w sieciach rozległych wykorzystuje się standard protokołu TCP/IP, nasuwa się pytanie, czy funkcje urządzenia pośredniczącego, realizowane dotychczas przez stację kontrolno-nadzorczą, mogą zostać zastąpione prostymi funkcjami szukania i kojarzenia (*ang. „routing"*) w sieciach TCP/IP bądź też, czy funkcje te mogą zostać wyodrębnione ze stacji kontrolno-nadzorczej i przekazane w całości do routerów IP?

Wydawać by się mogło, iż implementacja w sieciach rozległych i w sieciach przemysłowych tych samych protokołów komunikacyjnych, ba, nawet tych samych rozwiązań sprzętowych, umożliwi bezpośrednie połączenie urządzeń automatyki z poziomem sieci rozległej.

W takim przypadku poszczególne urządzenia stawałyby się wprost abonentami Internetu, pozwalając tym samym na dostęp do danych z dowolnego punktu na świecie. Chociaż wizja ta wydaje się być bardzo kusząca, a opisane rozwiązanie jest z technicznego punktu widzenia dosyć proste do zrealizowania, to jednak rozwiązanie powyższe w zdecydowanej większości przypadków nie nadaje się do zastosowania.



Rys. 65. Integracja segmentów sieci przemysłowej MODBUS/TCP
Fig. 65. The integration of industry network MODBUS/TCP segments

Podstawową przyczyną dyskredytującą rozwiązanie z rys. 65, jest brak mechanizmów zapewniających bezpieczeństwo na poziomie właściwym dla urządzeń automatyki przemysłowej. Stan taki powodowany jest nie tylko niedoskonałością algorytmów warstwy aplikacyjnej stosowanych w konstruowanych urządzeniach, ale wynika on wprost z problemów związanych z konwersją z poziomu protokołu sieci przemysłowych do protokołu TCP/IP. Zazwyczaj jedynym oferowanym na tym poziomie mechanizmem bezpieczeństwa jest konieczność znajomości punktu końcowego połączenia TCP, składającego się z adresu IP i numeru portu komunikacyjnego. Tak zdefiniowany poziom bezpieczeństwa nie stanowi wystarczającej bariery nie tylko dla celowego ataku, ale nawet nie chroni przed przypadkowym zaburzeniem w pracy sieci (np. podczas rozgłaszania danych).

Inny przypadek zachodzi dla konfiguracji, w której wymagana jest dodatkowa wymiana danych pomiędzy poszczególnymi abonentami obsługującymi dany proces przemysłowy. W takim przypadku połączenie współpracujących ze sobą urządzeń za pomocą Intersieci nie daje żadnej gwarancji co do czasu wymiany informacji pomiędzy nimi. Szczególnym przypadkiem może się okazać całkowity paraliż tak skonstruowanego systemu.

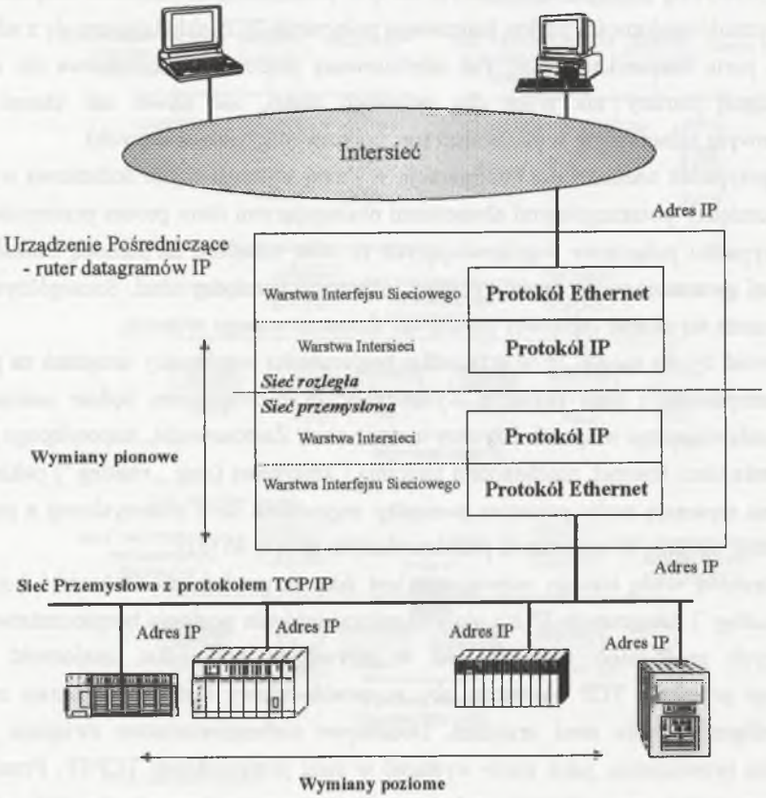
Wydawać by się mogło, iż w przypadku konieczności współpracy urządzeń za pomocą sieci przemysłowej i sieci rozległej wystarczającym rozwiązaniem będzie zastosowanie routera rozdzielającego w sposób fizyczny te dwie sieci. Zastosowanie, stanowiącego podstawę działania sieci Internet, mechanizmu szukania i kojarzenia (*ang. „routing"*) pakietów IP pozwoli na separację ruchu pakietów pomiędzy segmentem sieci przemysłowej a pozostałą częścią sieci, zgodnie ze schematem przedstawionym na rys. 66 [31].

Podstawową wadą takiego rozwiązania jest fakt, iż mechanizm szukania i kojarzenia (*ang. „routing"*) datagramów IP nie wpływa na podniesienie poziomu bezpieczeństwa usług oferowanych przez sieć. Podobnie jak w pierwszym przypadku, znajomość punktu końcowego protokołu TCP wystarcza, aby w sposób celowy bądź przypadkowy zaburzyć pracę podłączonych do sieci urządzeń. Dodatkowe niebezpieczeństwo związane jest ze zjawiskiem przeciążenia, jakie może wystąpić w sieci przemysłowej TCP/IP. Przeciążenie sieci może zostać spowodowane nadmierną liczbą wprowadzonych do niej pakietów. Pomimo najlepiej skonstruowanego oprogramowania aplikacyjnego, tylko architektura protokołów TCP/IP, pozwalająca na kolejkowanie, powielanie i transport różnymi trasami datagramów IP, nie pozwala na wykluczenie sytuacji (co więcej, jest ona w przypadku przeciążenia sieci rozległej wysoce prawdopodobna), w której router łączący tę sieć z siecią przemysłową, stanie przed zadaniem wprowadzenia w krótkim czasie znacznej liczby datagramów IP do sieci przemysłowej, zwiększając tym samym liczbę kolizji w sieci Ethernet. Sytuacja ta spowoduje wzrost obciążenia sieci przemysłowej. Wzrost ten, zgodnie z teorią kolejkowania, wymusi zmianę wariacji czasu podróży pakietu [31]

$q=1/(1-L)$,

gdzie: L – oznacza współczynnik aktualnego obciążenia sieci ($0 \leq L \leq 1$).

Wystąpi tym samym brak możliwości wyznaczenia maksymalnego czasu dostępu do łącza dla abonentów sieci przemysłowej. Stan taki leży w oczywistej sprzeczności z zasadą determinizmu czasowego, stawianą u podstaw konstrukcji sieci przemysłowych.



Rys. 66. Separacja sieci przemysłowej poprzez szukanie i kojarzenie (ang. *„routing"*) pakietów IP

Fig. 66. The separation industry network with using IP routing process

Alternatywą dla rozwiązania szukania i kojarzenia (ang. *„routing"*) datagramów IP może być wykorzystanie separacji sieci przemysłowej od sieci rozległej na poziomie warstwy aplikacji. Przedstawiona na rysunku 67 konfiguracja zakłada całkowitą izolację tych dwóch rodzajów sieci na poziomie aplikacji urządzenia pośredniczącego. Jakikolwiek ruch przychodzący z sieci rozległej zostaje zatrzymany na poziomie urządzenia pośredniczącego. Urządzenie to pełni funkcję serwera usług, pozwalając na dostęp do zasobów znajdujących się po stronie sieci przemysłowej. Oprogramowanie umieszczone w zdalnych stanowiskach dostępu występuje w roli klienta zlecającego odpowiednie usługi do oprogramowania ser-

wera i otrzymującego wyniki ich realizacji za pomocą urządzenia pośredniczącego. Funkcje urządzenia pośredniczącego mogą być realizowane poprzez wydzielony komputer, będący jednocześnie abonentem sieci przemysłowej i abonentem sieci rozległej bądź też funkcje te mogą być realizowane poprzez jeden z modułów stacji kontrolno-nadzorczej.

To rozwiązanie ma wiele istotnych zalet, do których należą:

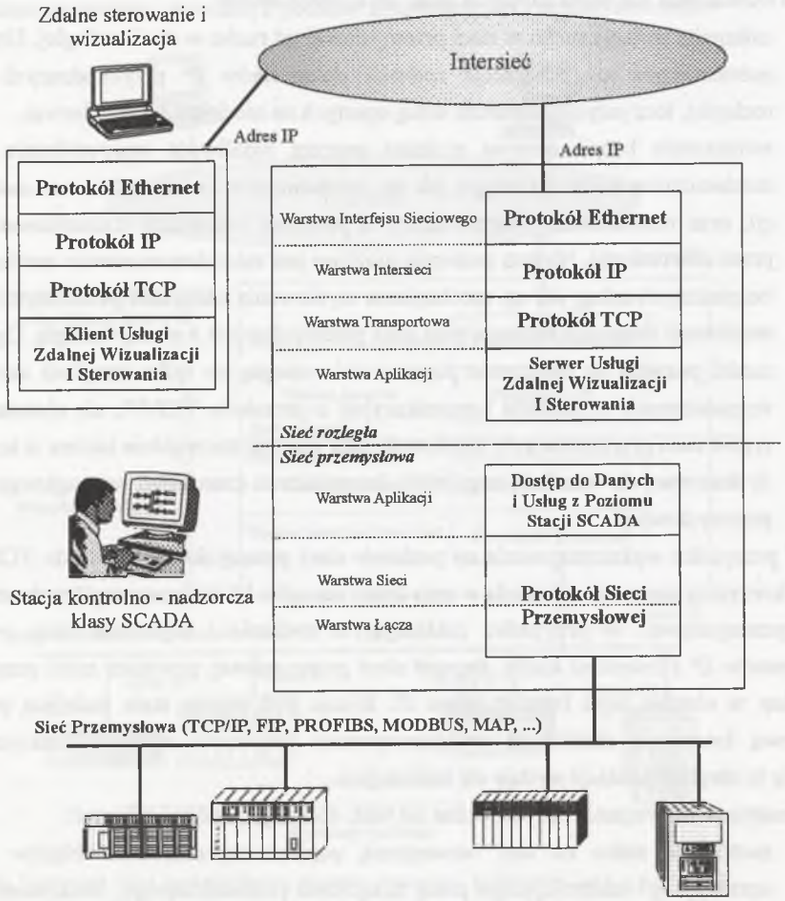
- całkowita izolacja ruchu w sieci przemysłowej od ruchu w sieci rozległej. Urządzenie pośredniczące nie przekazuje *żadnych datagramów IP* przychodzących z sieci rozległej, lecz jedynie dostarcza usług opartych na modelu Klient – Serwer,
- zwiększenie bezpieczeństwa systemu poprzez możliwość wprowadzenia szeregu mechanizmów kontroli dostępu, jak np. mechanizmów *uwierzytelniania i autentyfikacji*, oraz mechanizmów zapewniających *poufność informacji* transmitowanych poprzez sieć rozległą. Na tym poziomie możliwe jest zaimplementowanie mechanizmów bezpiecznych usług, jak np. mechanizmu szyfrowania z kluczem publicznym,
- możliwość integracji różnego typu sieci przemysłowych z siecią rozległą. Opisywany model pozwala na połączenie poprzez sieć rozległą nie tylko urządzeń automatyki, wyposażonych w interfejs komunikacyjny o protokole TCP/IP, ale również *innych* typów *sieci przemysłowych*. Możliwość taka staje się szczególnie istotna w kontekście dyskutowanych wcześniej zagadnień determinizmu czasowego wymaganego od sieci przemysłowych.

W przypadku wykorzystywania na poziomie sieci przemysłowej standardu TCP/IP znaczącą korzyścią może być swoboda w przydziale adresów IP dla poszczególnych abonentów sieci przemysłowej. W przypadku zakładającym szukanie i kojarzenie (ang. *„routing"*) datagramów IP z Intersieci każdy abonent sieci przemysłowej powinien mieć przydzielony unikalny w obrębie sieci Internet adres IP. Biorąc pod uwagę stale malejącą przestrzeń adresową Intersieci, możliwość wykorzystywania lokalnych adresów funkcjonujących jedynie w obrębie instalacji wydaje się interesująca.

Omawiane rozwiązanie nie jest wolne od wad, do których można zaliczyć:

- możliwość ataku na sieć wewnętrzną poprzez wykorzystanie błędów systemu operacyjnego kontrolującego pracę urządzenia pośredniczącego. Możliwość ta staje się szczególnie groźna w przypadku konfiguracji przedstawionej na rysunku 64, w której funkcję urządzenia pośredniczącego pełni jeden z modułów stacji kontrolno-nadzorczej. W takim przypadku atak na system operacyjny może spowodować nie tylko zaburzenie funkcjonowania stacji kontrolno-nadzorczej, lecz także zakończyć się kradzieżą lub zniszczeniem danych posiadanych przez tę stację, w tym również informacji opisujących pracę instalacji oraz informacji odpowiadających za bezpieczeństwo systemu. Przypadek ten obejmuje także zaburzenie funkcjonowania lub wręcz paraliż na poziomie sieci przemysłowej. Prawdopodobieństwo ataku poprzez

system operacyjny urządzenia pośredniczącego nie jest tak duże, jak w przypadku bezpośredniego ataku na sieć i zależy ono od wybranych rozwiązań i zastosowanych mechanizmów zabezpieczeń systemowych, tym niemniej należy o nim pamiętać podczas konstruowania systemów o opisanej powyżej architekturze;



Rys. 67. Separacja sieci przemysłowej na poziomie stacji SCADA
Fig. 67. The separation of industry network on SCADA station level

- ograniczanie dostępu wyłącznie dla autoryzowanych użytkowników. Sytuacja taka sprawia, iż staje się niemożliwe wykorzystanie szeregu usług związanych z publicznym dostępem do danych, jak np. prezentacja wybranych informacji poprzez strony WWW,
- wiele punktów dostępu do sieci utrudnia jej skuteczną ochronę, a także wymaga budowy skomplikowanych algorytmów decydujących o prawach dostępu poprzez poszczególne urządzenia pośredniczące. Wobec tego to rozwiązanie niezbyt dobrze nadaje się do systemów, w których występuje większa liczba urządzeń pośredniczących, posiadających połączenie z siecią rozległą. Dla takiej konfiguracji znacznie lepszym rozwiązaniem wydaje się być architektura z zastosowaniem komputera - bastionu.

Architektura z zastosowaniem komputera-bastionu znana jest z klasycznych systemów sieci komputerowych wykorzystujących ściany ogniowe (ang. „*firewall*") [31]. Rozwiązanie to zakłada całkowitą izolację nie tylko poziomu sieci przemysłowej, ale także poziomu sieci lokalnej łączącej poszczególne stacje kontrolno-nadzorcze. Dostęp do tej struktury z poziomu Intersieci możliwy jest jedynie poprzez komputer-bastion posiadający rozbudowany system zabezpieczeń. Komputer ten stanowi bezpieczny kanał komunikacyjny, blokujący bezpośredni dostęp do poszczególnych stanowisk operatorskich. Zdalne stanowiska wizualizacji i sterowania komunikują się wyłącznie ze znajdującym się poza chronioną strefą bastionem, który separuje instalację automatyki oraz system sterowania i wizualizacji od sieci rozległej. Komunikacja poprzez komputer-bastion może polegać bądź to na udostępnianiu przygotowanych na nim do prezentacji danych, na przykład poprzez strony WWW, bądź też poprzez realizację usług opartych na modelu Klient-Serwer, podobnie jak to występowało w przypadku architektury wykorzystującej urządzenie pośredniczące.

10.3. Podsumowanie

Dynamiczny rozwój sieci opartych na protokole TCP/IP stymulowany wykładniczym tempem rozrastania się Internetu znalazł odzwierciedlenie także w rozwiązaniach wykorzystywanych w sieciach przemysłowych. Patrząc na hierarchiczną strukturę instalacji automatyki przemysłowej, należy stwierdzić, iż zastosowanie protokołu TCP/IP w sieciach najniższego poziomu niesie za sobą zarówno korzyści, jak i nowe, nie występujące w konstruowanych dotychczas systemach automatyki przemysłowej, zagrożenia. Od właściwego zaprojektowania architektury całego systemu komunikacyjnego zależy zarówno bezpieczeństwo całej instalacji automatyki, jak i jej elastyczność, przejawiająca się w łatwości implementacji dostępnych dziś i pojawiających się w przyszłości nowych rozwiązań.

Trwające obecnie prace nad szóstą wersją protokołu IP budzą nadzieje również w dziedzinie sieci przemysłowych. Wprowadzane przez protokół IP w wersji szóstej wsparcie dla rezerwacji zasobów może być kluczem do tworzenia bezpiecznych kanałów komunikacyjnych, zapewniających wymaganą w systemach przemysłowych gwarantowaną przepustowość i stałe opóźnienie.

Bez względu na siłę wprowadzanych nowych rozwiązań sprzętowych i możliwości wnoszonych przez nowe protokoły komunikacyjne bezpośrednia odpowiedzialność za bezpieczeństwo obsługiwanych instalacji spoczywać będzie w dalszym ciągu na właściwie

przygotowanej architekturze systemu, popartej analizą jego pracy pod kątem wymagań instalacji. Systemy automatyki przemysłowej typu „*hard real-time*" wymagać będą architektur gwarantujących determinizm czasowy. Dla realizacji tego postulatu należy poddać analizie nie tylko poszczególne segmenty sieci przemysłowych, wchodzące w skład tych systemów, ale należy zwrócić uwagę na zjawiska związane z ich współpracą. Konstruując system automatyki, zazwyczaj można wyodrębnić sporą grupę zastosowań, dla których wystarczające będzie zapewnienie usług komunikacyjnych typu „*soft real-time*". Sposób realizacji tych usług powinien być dobrany bądź to na drodze rozwiązań sprzętowych, bądź też poprzez rozwiązania programowe, w taki sposób aby nie wpływał on na realizację usług komunikacyjnych typu „*hard real-time*". Zastosowanie protokołu TCP/IP w sieciach przemysłowych najniższego poziomu sprawiło, iż jednym z głównych wyzwań stawianych przed systemami czasu rzeczywistego stała się potrzeba wypracowania mechanizmów, pozwalających na połączenie różnych typów usług. Ze względu na wzrastające w lawinowym tempie wymagania dotyczące stworzenia możliwości integracji systemów automatyki przemysłowej, i to zarówno na poziomie przedsiębiorstwa, jak i na poziomie globalnym, wydaje się, iż badania nad problemami związanymi z metodyką konstruowania rozproszonych systemów automatyki przemysłowej, wykorzystujących protokół TCP/IP, wyznaczać będą kierunki rozwoju dla szerszego spektrum zagadnień stawianych przed informatyką przemysłową.

Rozwój technologiczny, jaki obserwujemy w ostatnich latach, przyczynił się tak dalece do redukcji kosztów związanych z implementacją algorytmów pracy sieci przemysłowych, że mamy do czynienia z bardzo dużą gamą rozwiązań w tej dziedzinie [122, 115, 18, 59, 121].

Doświadczenia kilku ostatnich lat pokazują, że systemy sterowania i wizualizacji gwałtownie zwiększają zapotrzebowanie na ilość informacji, którą należy przesłać siecią przemysłową. Wzrost ilości informacji transmitowanych siecią jest spowodowany zarówno wzrostem liczby danych bezpośrednio związanych ze sterowaniem procesem technologicznym, jak i ze zwiększeniem ilości informacji diagnostycznych i serwisowych, potrzebnych do kontroli poprawności pracy urządzeń wykonawczych [58]. Uzasadnione jest więc, przed zaprojektowaniem systemu komunikacyjnego, przeprowadzenie kalkulacji ilości i charakteru informacji, jaką chcemy przesłać z użyciem sieci w rozwiązaniu docelowym, gdyż bardzo często okazuje się, że przy rozbudowie systemu sterowania czy wizualizacji często dochodzimy do granic możliwości zastosowanej sieci przemysłowej. Analiza przepływu informacji w sieci (rozdziały 3, 4, 5, 6, 7) uchroni projektanta przed ponoszeniem niepotrzebnych kosztów, co jest związane jednakowo zarówno z „niedowymiarowaniem", jak i „przewymiarowaniem" parametrów sieci. Niewłaściwa ocena wymagań czasowych stawianych przed systemem komunikacyjnym albo spowoduje konieczność rozbudowy istniejącej infrastruktury sieciowej o dodatkowe kosztowne urządzenia („szybsze" koprocesory,

jednostki centralne czy wręcz wymiana medium komunikacyjnego), czego zazwyczaj da się uniknąć, albo wymusi zastosowanie urządzeń o najwyższych parametrach czasowych, a więc stosunkowo drogich, gdy takowe w ogóle nie są wymagane.

Koszty poszczególnych interfejsów sieciowych nie różnią się zasadniczo między sobą. Dodatkowe nakłady finansowe, jakie użytkownik ponosi na okablowanie, są porównywalne dla poszczególnych rozwiązań. Dlatego podstawowymi kryteriami wyboru sieci przemysłowej powinny być jej szybkość i przepustowość oraz niezawodność, a przede wszystkim model, w oparciu o który pracuje. Ważne są również możliwości rozbudowy i rozwoju zarówno samej sieci, jak i konkretnej aplikacji, w której dana sieć ma pracować.

Niniejszy rozdział stanowi z jednej strony przegląd istniejących sieci przemysłowych i ma charakter poglądowy, a z drugiej jest wynikiem szeregu eksperymentów i testów przeprowadzonych z dostępnymi autorowi sieciami i porównań z materiałami firmowymi producentów. Zamysłem niniejszego rozdziału było zaprezentowanie istniejącej gamy rozwiązań sieciowych w taki sposób, aby ułatwić ewentualnym konstruktorom dobór odpowiedniego typu sieci do stawianych przed systemem zadań.

11. PRZEGLĄD ROZWIĄZAŃ SIECI PRZEMYSŁOWYCH

Na rynku funkcjonuje bardzo wiele firmowych rozwiązań sieci przemysłowych. Jedne z nich, z powodzeniem stosowane od wielu lat, nie doczekały się i pewnie nie doczekają się żadnych aktów normatywnych (należy tu mieć na myśli przede wszystkim normy IEC), drugie natomiast to sieci pretendujące do miana sieci polowych (norma IEC, *ang. „Multi Protocol Standard"*), i te są lub będą objęte procesem normalizacyjnym. Podstawowym zadaniem stojącym przed projektantem systemu przemysłowego jest umiejętny dobór sieci komunikacyjnych na potrzeby systemu. Należy zatem przede wszystkim określić kryteria, według których protokoły będą porównywane, a następnie dobierane. Takimi kryteriami mogą być, między innymi:

- model protokołu, z którego można wnioskować o deterministycznym w czasie dostępie do medium,
- długość segmentu sieci i maksymalna liczba abonentów, co ułatwia dobór sieci ze względu na rozległość instalacji i liczbę stacji obiektowych,
- prędkość transmisji i rodzaj medium transmisyjnego, dzięki czemu można łatwo ocenić koszt instalacji samej magistrali,
- maksymalna długość bloków komunikatów, co ułatwia ocenę przepustowości sieci,
- objęcie standardem międzynarodowym.

Zanim zostaną zaprezentowane parametry sieci, wymieńmy te, które będą porównywane i krótko je scharakteryzujmy.

Porównywane będą następujące sieci przemysłowe:

- *Modbus*
- *N-80, Genius*
- *LonWorks*
- *CAN*
- *SDS*
- *DeviceNet*
- *HART*
- *AS-I*
- **FIP / WorldFip**
- **Profibus-PD**
- **Intrerbus-S**

- **Fieldbus Foundation H1, H2**
- **Fieldbus Foundation High Speed Ethernet – HSE**
- **ControlNet**
- **P-Net.**

Nazwy sieci zapisane wytłuszczoną czcionką dotyczą sieci polowych, natomiast zapisane czcionką pochyłą dotyczą sieci stosowanych, ale nie objętych ujednoliconym międzynarodowym standardem.

11.1. Sieć Modbus

Sieć Modbus to rozwiązanie firmy Modicon- Gould wprowadzone na rynek w 1980 r. Prostota tego protokołu pozwala na łatwą implementację w dowolnym urządzeniu posiadającym mikrokontroler, co w znacznym stopniu wpłynęło na niskie koszty i popularność.

Modbus jest siecią typu Master–Slave. Wydzielona stacja Master, posługująca się listą wymian cyklicznych i wyzwalanych, odpytuje kolejno poszczególnych abonentów sieci.

Sieć ta pracuje z niewielkimi szybkościami transmisji danych (typowe: 9.6 Kb/s, 19.2 Kb/s) na ograniczonym dystansie wynikającym z typu zastosowanego łącza komunikacyjnego (RS-232, RS-422, RS-485, Modem). Dystrybucja uprawnień do nadawania realizowana przez stację Master gwarantuje zdeterminowany dostęp do medium.

Zastosowanie

Szerokie zastosowanie w aplikacjach przemysłowych o niskich wymaganiach dotyczących szybkości i częstości transmisji danych, w szczególności w systemach z wydzielonym centrum, do którego przesyłane są dane z urządzeń peryferyjnych.

11.2. Sieć N-80, Genius

Sieć Genius to rozwiązanie firmy GE FANUC wprowadzone na rynek w 1985 r., znane również pod nazwą N-80 - firmy ALSTOM (CEGELEC).

Sieć ta jest siecią z przekazywaniem żetonu (*Token-Passing*). Żeton przekazywany jest kolejno od urządzenia o adresie 0 do urządzenia o adresie 31. Wszystkie wymiany realizowane są jako rozgłoszenie (ang. *„broadcast"*). Każde urządzenie w sieci odbiera zawsze dane transmitowane przez stację posiadającą żeton, natomiast może nadawać tylko, gdy go otrzyma. Z uwagi na zdefiniowany czas nadawania przez stację posiadającą sieć Genius / N-80 jest siecią o zdeterminowanym w czasie dostępie do medium transmisyjnego.

Sieć pracuje z następującymi prędkościami transmisji danych:
153.6 Kb/s @ 1060 m (standardowa),
38.4 Kb/s @ 2275 m.

Zastosowanie

Sieć Genius/N-80 może być stosowana w aplikacjach wymagających szybkich transferów danych oraz bloków wejść/wyjść pomiędzy rozproszonymi abonentami.

11.3. Sieć LonWorks

Sieć LonWorks jest rozwiązaniem firmy Echelon [**Błąd! Nie można odnaleźć źródła odsyłacza.**] wprowadzonym na rynek w 1991 r. Posiada bazę elementową w postaci koprocesorów sieciowych zrealizowanych w oparciu o grupę mikrokontrolerów o nazwie *Neuron Chip*. Kontrolery zawierają implementa-cję protokołu sieciowego o nazwie LonTalk, opartego na modelu OSI. Jest to protokół pozwalający na łączenie tysięcy węzłów we wspólną sieć, przy czym topologia nie jest z góry narzucona. Podstawową jednostką w sieci jest węzeł, inteligentne urządzenie komunikujące się z innymi węzłami. Węzłem może być czujnik, kontroler lub komputer. Sieć nie posiada wyróżnionych węzłów kontrolujących transmisję czy sterujących dostępem do medium transmisyjnego, gdyż węzły są równouprawnione. W koprocesorze sieciowym *Neuron Chip* zaimplementowano mechanizmy odpowiadające za rozwiązywanie konfliktów dostępu do wspólnej magistrali, adresowanie i dostarczanie przesyłek sieciowych, zabezpieczenie transmisji, formatowanie danych przesyłanych przez sieć, autoryzację węzłów oraz optymalizację przepustowości łącza.

Sieć LonWorks pracuje z następującymi prędkościami transmisji danych:
1.25 Mb/s @ 500 m,
78.0 Kb/s @ 100 m.

Zastosowanie

Sieć ta najbardziej rozpowszechniona jest w systemach „inteligentnych budynków", gdzie jest stosowana do sterowania oświetleniem, ogrzewaniem, alarmem. Została uznana przez Organizację ANSI za standard w zakresie automatyzacji budynków. Może być z powodzeniem stosowana w aplikacjach do monitorowania i rejestrowania danych pomiarowych (szybki przesył danych na limitowanym dystansie).

11.4. Sieć CAN (Controller Area Network)

Jest to rozwiązanie firmy Bosch wprowadzone na rynek w 1994 r. CAN jest siecią, pracującą w oparciu o model producent-konsument (P-K). Istnieją dwie wersje tej sieci tzw. *Basic CAN* oraz *Full CAN* różniące się sposobem adresacji (*11- lub 29-bitowym*). Dodatkowo sieć ta występuje w różnych odmianach :

- DevicNet - oparta na specyfikacji CAN 2.0A.
- SDS –oparta na specyfikacji CAN.

- CAN Open.

Sieć CAN posiada normy: ISO11898 oraz ISO11519.

Zastosowanie

Sieć CAN stosowana jest do kontroli procesów przemysłowych, budynków, pojazdów oraz do szybkiego przesyłania danych na limitowanym dystansie.

11.5. Sieć SDS (Smart Distributed System)

Jest to rozwiązanie firmy Honeywell (*Microswitch Division*) wprowadzone na rynek w 1994 r. SDS jest siecią opartą na protokole komunikacyjnym sieci CAN, w której Master posiada pełną kontrolę nad zdalnymi modułami pomiarowymi, i w której nie jest możliwa komunikacja między modułami bez stacji Master. Jest to sieć bardzo efektywna przy komunikacji „point to point" pomiędzy stacją Master a zdalnymi modułami wejść/wyjść.

Pracuje z następującymi prędkościami transmisji danych:
500 Kb/s@ 100 m,
125 Kb/s@ 500 m.

Zastosowanie

SDS znalazła zastosowanie w aplikacjach przemysłowych do szybkiego przesyłania danych na limitowanym dystansie. Najefektywniej przesyła dane podczas połączenia zdalnych modułów wejść/wyjść ze sterownikiem swobodnie programowalnym.

11.6. Sieć DeviceNet

Jest to rozwiązanie firmy Rockwell i Allen-Bradley wprowadzone na rynek w 1994 r. DeviceNet jest siecią opartą na protokole komunikacyjnym sieci CAN. Jest systemem otwartym, w którym wszystkie stacje mają te same prawa dostępu do medium komunikacyjnego. Projektant stacji może zaimplementować przekazywanie kontroli do predefiniowanej stacji Master-Slave, ale brak jest mechanizmów zmuszających stację nadającą do takiego podporządkowania się. Wymiany w sieci realizowane są jako rozgłoszenie (*broadcast*). Oznacza to, że wszystkie stacje odbierają wszystkie transmitowane siecią dane, a stosując lokalną filtrację reagują tylko na dane wybierane, którymi są zainteresowane.

Pracuje z następującymi prędkościami transmisji danych:
Mb/s @ 30 m,
125 Kb/s @ 490 m.

Zastosowanie

Sieć DeviceNet znalazła zastosowanie w aplikacjach przemysłowych do szybkiego przesyłu danych na limitowanym dystansie.

11.7. Sieć HART

Jest to rozwiązanie opracowane przez HART Communication Foundation. Protokół komunikacyjny HART oparty jest na telefonicznym standardzie komunikacyjnym Bell 202 oraz na FSK (ang. *„frequency shift keying"*), gdzie bity 1 i 0 kodowane są za pomocą sygnału o częstotliwości 1,200 Hz i 2,200 Hz nałożonego na analogowy sygnał prądowy 4-20 mA. Zastosowana metoda kodowania nie wpływa na pogorszenie sygnału analogowego, gdyż średnia wartość sygnału FSK jest zawsze równa zero.

Urządzenia w sieci HART mogą pracować w konfiguracji point-to-point[1] lub multidrop[2].

Zastosowanie

Sieć HART stosowana jest do komunikacji z inteligentnymi czujnikami i oprzyrządowaniem pracującym na pętli prądowej 4-20 mA.

11.8. Sieć FIP/ WorldFIP

Sieć FIP jest rozwiązaniem firmy CEGELEC wprowadzonym na rynek w 1988 r., natomiast organizacją nadzorującą jest *WorldFIP*. Posiada kompletną specyfikację i europejski standard sieci polowych *EN50170* oraz kompletną listę koprocesorów i specjalizowanych układów scalonych, a także niezbędne narzędzia programistyczne potrzebne do implementacji tej sieci w dowolnym rozwiązaniu sprzętowym (od pojedynczych czujników pomiarowych, poprzez urządzenia wykonawcze, sterowniki aż po karty do komputerów, pełniących rolę stacji SCADA (ang. *„Supervisory Control And Data Acquisition*).

Sieć FIP pracuje w oparciu o model PDC[3]. Praca sieci nadzorowana jest przez arbitra magistrali, który w sposób periodyczny lub aperiodyczny wysyła na magistralę identyfikator zmiennej, żądając udostępnienia jej przez producenta. Producent po rozpoznaniu identyfikatora zmiennej, którą generuje, wysyła jej wartość na magistralę i udostępnia ją w ten sposób wszystkim odbiorcom.

Sieć FIP z uwagi na model, w oparciu o który pracuje, zapewnia producentom zmiennych zdeterminowany czas dostępu do medium (za sprawą arbitra magistrali) oraz zgodność przestrzenną danych docierających do rozproszonych konsumentów (tryb rozgłoszeniowy transmisji zmiennych).

Standardowo koprocesory mają zaimplementowane mechanizmy pozwalające obsługiwać dwie magistrale komunikacyjne, zapewniając w ten sposób redundancję medium transmisyjnego. Transmisje danych dla trybu szybkiego (ang. *„fast mode"*) odbywają się z szybkościami:

[1] point-to-toint - punkt - punkt
[2] multidrop - punkt - wielopunkt
[3] PDC **P**roducer-**D**istributor-**C**onsumer

- 31.25 Kb/s @ 32 km,
- Mb/s @ 4km z 3 repeterami, 1000 m / segment,
- 2.5 Mb/s @ 2 km z 3 repeterami, 500 m / segment,
- 5.0 Mb/s @ 1 km z 3 repeterami, 250 m / segment.

Dodatkowo długość medium transmisyjnego można zwiększyć dwukrotnie w przypadku uruchomienia magistrali w trybie wolnym [4].

FIP z uwagi na stosowanie specjalizowanej bazy elementowej jest siecią bardzo szybką, a przyjmując za kryterium długość medium transmisyjnego, jest obecnie najszybszą dostępną siecią przemysłową. Dodatkowo koszt niezbędnej aparatury sieciowej jest porównywalny lub niższy w stosunku do innych szybkich sieci przemysłowych.

Zastosowanie

Sieć FIP znalazła zastosowanie w aplikacjach przemysłowych do szybkiego przesyłania zarówno małych, jak i dużych bloków danych na znaczne odległości. Może z powodzeniem być stosowania w aplikacjach łączących pojedyncze czujniki pomiarowe z dużymi urządzeniami lub sterownikami oraz stacją SCADA.

11.9. Sieć Profibus-PD

Profibus-PD jest rozwiązaniem firmy SIEMENS wprowadzonym na rynek w 1994 r. (organizacja nadzorująca PNO / PTO). Posiada europejski standard sieci polowych *EN50170*.

Sieć ta charakteryzuje się hybrydowym dostępem do medium transmisyjnego:

- Master-Slave dla wymian pomiędzy aktywną stacją Master a dowolną stacją,
- Token–Passing dla wymian uprawnień pomiędzy stacjami Master (przesyłanie żetonu).

Umożliwia podłączenie maksymalnie do 128 stacji pracujących z szybkościami transmisji danych:

12 Mb/s @ 100 m / 500 m [5],
1,5 Mb/s @ 200 m,
9,6 Kb/s @ 1200 m / 9600 m [6].

Zastosowanie

Sieć ta znalazła zastosowanie w aplikacjach przemysłowych do szybkiego przesyłania małych i dużych bloków danych.

[4] slow mode
[5] 12 Mb/s @ 500 m z zastosowaniem specjalizowanych repeterów.
[6] 31.2Kb/s @ 9600 m z zastosowaniem specjalizowanych repeterów.

11.10. Sieć Intrerbus-S

Intrerbus-S to rozwiązanie firmy Phoenix Contact wprowadzone na rynek w 1984 r. Idea przesyłania danych oparta jest na rejestrze przesuwnym. Pracuje z szybkością transmisji danych 500 Kb/s @ 400 m.

Zastosowanie

Sieć Intrerbus-S znalazła zastosowanie w aplikacjach przemysłowych do szybkiego przesyłania danych na limitowanym dystansie.

11.11. Sieć Fieldbus Foundation H1, H2

Jest to rozwiązanie firmy Siemens. Sieć ta charakteryzuje się hybrydowym dostępem do medium transmisyjnego:

- Model Master-Slave obowiązuje dla wymian pomiędzy aktywnym masterem a dowolną stacją, natomiast model Token–Passing dla wymian uprawnień między masterami (przesyłanie żetonu).
- FF H1 pracuje z szybkością transmisji danych 31.25 Kb/s.
- FF H2 pracuje z szybkością transmisji danych 1.0 Mb/s.

Zastosowanie

Stosowana w aplikacjach przemysłowych do szybkiego przesyłania bloków danych.

11.12. Sieć Fieldbus Foundation HSE - High Speed Ethernet

Ethernet jest rozwiązaniem firmy Xerox wprowadzonym na rynek w 1979 r. (promowany przez Digital Equipment i Intel). W 1985 r. uzyskał standard ISO/IEC 8802-3.

Ethernet jest siecią *multidrop* z dostępem do medium typu CSMA/CD[7], pracującą z prędkościami transmisji 10 Mb/s, 100 Mb/s oraz 1.0 Gb/s. Wersja przemysłowa tej sieci, oprócz nadajnika i kontrolera zapewniającego dostęp do medium, zawiera dodatkowo szybki mikrokontroler, RAM, ROM (EPROM/Flash) oraz inne elementy potrzebne do pracy warstwy aplikacji i implementacji protokołu TCP/IP.

Zastosowanie

Sieć FF HSE może być stosowana do szybkiego przesyłania danych oraz kontroli procesów przemysłowych z wyłączeniem aplikacji, gdzie wymaga się zachowania wysokiego poziomu powtarzalności transmisji danych z zachowaniem determinizmu czasowego.

[7] CSMA/CD – Carrier Sense Multiple Access with Collision Detect

11.13. Sieć ControlNet

ControlNet to rozwiązanie firmy Rockwell wprowadzone na rynek w 1997 r. Sieć ControlNet pracuje w oparciu o model P-K (Producent-Konsument) z prędkością transmisji równą 5 Mb/s.

Zastosowanie

Sieć ControlNet stosowana jest do kontroli procesów przemysłowych oraz do szybkiego przesyłania danych na limitowanym dystansie.

11.14. Sieć AS-I

Jest to rozwiązanie firmy AS-I Consorcium wprowadzone na rynek w 1993 r. AS-I jest siecią typu Master–Slave z zaimplementowanymi mechanizmami zapewniającymi determinizm czasowy.

Zastosowanie

Sieć AS-I stosowana jest do kontroli procesów przemysłowych, budynków, pojazdów oraz do szybkiego przesyłania danych na limitowanym dystansie.

11.15. Sieć P-Net

Jest to rozwiązanie firmy PROCES-DATA wprowadzone na rynek w 1984 r. P-Net jest siecią typu Multi-master z warstwą fizyczną opartą na standardzie RS-485. Z uwagi na małe wymagania protokołu może być implementowana w 8-bitowych mikrokontrolerach jednoukładowych. Posiada europejską normę EN50170-1.

Zastosowanie

Sieć P-Net stosowana jest do kontroli procesów przemysłowych oraz do szybkiego przesyłania danych na limitowanym dystansie.

11.16. Prezentacja parametrów sieci przemysłowych

W niniejszym rozdziale zaprezentowano w postaci tabelarycznej parametry sieci przemysłowych, które podzielono na dwie części:

- Sieci stosowane w przemyśle od wielu lat (ale nie brane pod uwagę w procesie normalizacyjnym organizacji IEC i CENELEC).
- Sieci spełniające kryteria sieci polowych i co najważniejsze brane pod uwagę w procesie normalizacyjnym organizacji **IEC**, jako przyszły standard sieci polowych na nadchodzącą dekadę (ang. *„Multi protocol standard"*).

Informacje w nich zawarte pozwalają na wstępny wybór typu sieci w funkcji jej przeznaczenia.

Tabela 4

Charakterystyka sieci przemysłowych - cz. A1 (na podstawie danych GE FANUC, literatury fachowej, stron WEB oraz badań własnych)

Nazwa sieci / magistrali	Szybkość transmisji @ dystans			Rodzaj medium transmisyjnego	Max liczba stacji / stacji na segment	Max rozmiar bloku danych/komunikatów [bajty]
	Szybkość transmisji	Max długość segmentu [m]	Max długość medium tr. [m]			
Modbus	9.6 Kb/s	1200 [8]	1200	Różne	256 / 32	256
Genius, N-80	153.6 Kb/s 38.4 Kb/s	1000 2250	1000 2250	Skrętka, światłowód	32	128
LonWorks	1.25 Mb/s	500	500	Różne	32÷32k [9]	54 / 31
SDS	1.0 Mb/s	100	500	Skrętka [10]	32 [11]	8
DeviceNet	500 Kb/s	100	500	Skrętka [10]	64	8
CAN	Mb/s 50 Kb/s	40 1000	40 [12] 1000	Skrętka, światłowód	2^{11}, 2^{29}	8
HART	2-3 razy /s [13]	1500 / 3000 [14]	1500 / 3000	Skrętka	15	1
AS-I	167 Kb/s	100	300	Skrętka	31	31

[8] Ograniczenie długości medium transmisyjnego dla łącza szeregowego RS-485
[9] 32÷32385 stacji LonWorks– zależne od typu nadajników / odbiorników oraz topologii (max 32385 stacji, 255 segmentów po 127 stacji)
[10] Skrętka jest jednocześnie medium transmisyjnym i linią zasilającą
[11] 32 stacje SDS dla 1.0Mb/s pozostałe szybkości transmisji 64 stacje
[12] Trudności ze stosowaniem repeterów
[13] Aktualizacja danych 2-3 razy/s, opcjonalnie w trybie burst 3-4 razy /s
[14] 3000 m dla połączenia peer-to-peer, 1500 dla połączenia multidrop

Tabela 5

Charakterystyka sieci przemysłowych - cz. B1 / IEC (na podstawie danych GE FANUC, literatury fachowej, stron WEB oraz badań własnych)

Nazwa sieci / magistrali	Szybkość transmisji @ dystans			Rodzaj medium transmisyjnego	Max liczba stacji / stacji na segment	Max rozmiar bloku danych/komunikatów [bajty]
	Szybkość transmisji	Max długość segmentu [m]	Max długość medium tr. [m]			
FIP, WorldFIP	1.0 Mb/s	1000	4000	Skrętka, światłowód, łącza bezprzewodowe	256 [15] / 32	128 / 250
	2.5 Mb/s	500	2000			
	5.0 Mb/s	250	1000	światłowód		
Profibus-DP	12.0 Mb/s	100	500	Skrętka, światłowód	128 [16] / 32	244
Interbus-S	500 Kb/s	400	400	Skrętka, światłowód	256	64
FF H1	31.25 Kb/s	1900	1900	Skrętka	240 [17]	244
FF HSE	10/100 Mb/s	100	2500	Skrętka, światłowód	100	-
ControlNet	5.0 Mb/s	250	1000	Skrętka, światłowód	99	128 [18]
P-Net	76.8 Kb/s	1200	1200	Skrętka	125 [19]	1
SwiftNet	Bd [20]	Bd	Bd	Bd	Bd	Bd

[15] 256 stacji FIP na całej długości medium transmisyjnego - z zastosowaniem specjalizowanych repeterów
[16] 128 stacji Profibus-DP na całej długości medium transmisyjnego - z zastosowaniem specjalizowanych repeterów
[17] 240 stacji na segment 65,000 segmentów
[18] Wymiana danych z modułem I/O 1÷64 słów, wymiany pomiędzy innymi procesorami ControlNet 1-240 słów
[19] 125 stacji na segment
[20] Bd -Brak danych

Tabela 6

Charakterystyka sieci przemysłowych - cz. A2 (na podstawie danych GE FANUC, literatury fachowej, stron WEB oraz badań własnych)

Nazwa sieci magistrali	Model komunikacji	Typ dostępu do medium	Determinizm czasowy [21]	Wbudowane mechanizmy bezpieczeństwa	ASIC Chip	Typowe aplikacje
Modbus	M-S [22]	BRAK	TAK	NIE	NIE	Kontrola procesów, budynków, automatyzacja fabryk
Genius, N-80	P-C	BRAK	TAK	TAK	NIE	Kontrola procesów, automatyzacja fabryk
LonWorks	M-S, P2P [23]	PM [24]	NIE	TAK	TAK	Kontrola procesów, budynków, automatyzacja fabryk
SDS	M-S,P2P,M-M [25]	CSMA/NBA [26]	NIE	NIE	TAK	Kontrola procesów, budynków, pojazdów, automatyzacja fabryk
DeviceNet	M-S,M-M, inne	CSMA/NBA	NIE	NIE	TAK	Kontrola procesów, budynków, pojazdów, automatyzacja fabryk
CAN	P-C [27]	CSMA/CD + AMP [28]	NIE	NIE	TAK	Kontrola procesów, budynków, pojazdów, automatyzacja fabryk
HART	M-S	BRAK	TAK	TAK	TAK	Inteligentne oprzyrządowanie
AS-I	M-S	CP [29]	TAK	NIE	TAK	Kontrola procesów, budynków, automatyzacja fabryk

[21] Możliwość wykonania zadania w precyzyjnie określonym czasie (przesłania/otrzymania danych w z góry określonym i nieprzekraczalnym czasie).
[22] M-S Master-Slave
[23] P2P Peer to peer
[24] PM Predictive Media / CSMA/CA-P
[25] M-M Multi-Master
[26] CSMA/NBA Carrier Sense Multiple Access with Non-destructive Bitwise Arbitration
[27] P-C Producer-Consumer
[28] CSMA/CD + AMP Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority
[29] CP Cyclic Polling

Tabela 7

Charakterystyka sieci przemysłowych - cz. B2 / IEC (na podstawie danych GE FANUC, literatury fachowej, stron WEB oraz badań własnych)

Nazwa sieci magistrali	Model komunikacji	Typ dostępu do medium	Determinizm czasowy [30]	Wbudowane mechanizmy bezpieczeństwa	ASIC	Typowe aplikacje
FIP, WorldFIP	P-D-C [31]	BA [32]	TAK	TAK	TAK	Kontrola procesów, budynków, pojazdów, automatyzacja fabryk
Profibus-DP,FMS	M-S, P2P	TP	TAK	NIE	TAK	Kontrola procesów, automatyzacja fabryk
Profibus-PA	M-S, P2P	TP	TAK	TAK	TAK	
Interbus-S	M-S	BRAK	TAK	NIE	TAK	Kontrola procesów, automatyzacja fabryk
FF H1	C-S-P-S [33]	TP	TAK	TAK	TAK	Kontrola procesów, automatyzacja fabryk

[30] Możliwość wykonania zadania w precyzyjnie określonym czasie (przesłania/otrzymania danych w z góry określonym i nieprzekraczalnym czasie).
[31] P-D-C Producer-Distributor-Consumer
[32] BA Bus Arbiter
[33] C-S-P-S Client-Serwer-Publisher-Subscriber

Tabela 7 cd.

Nazwa sieci magistrali	Model komunikacji	Typ dostępu do medium	Determinizm czasowy [34]	Wbudowane mechanizmy bezpieczeństwa	ASIC	Typowe aplikacje
FF HSE	M-S, P2P	CSMA/CD [35]	TAK	NIE	TAK	Kontrola procesów, przesył danych
ControlNet	P-C	CTDMA [36]	TAK	NIE	TAK	Kontrola procesów, automatyzacja fabryk
P-Net	M-M [37]	TP	TAK	NIE	NIE	Kontrola procesów, automatyzacja fabryk
SwiftNet	Bd [38]	Bd	Bd	Bd	Bd	Bd

[34] Możliwość wykonania zadania w precyzyjnie określonym czasie (przesłania/otrzymania danych w z góry określonym i nieprzekraczalnym czasie).
[35] CSMA/CD Carrier Sense Multiple Access with Collision Detect
[36] CTDMA Concurrent Time Domain Multiple Access
[37] M-M Multi-Master
[38] Bd Brak danych

11.17. Analiza czasów transmisji danych

Korzystając z zasad i zależności czasowych zamieszczonych w poprzednich rozdziałach niniejszej pracy i w [93] oraz opierając się na materiałach firmowych producentów sieci, przystąpiono do analizy czasów transmisji wszystkich typów sieci, o których była mowa w niniejszym rozdziale. Przedmiotem teoretycznego testu były następujące transakcje wymiany:

- cykliczny przesył 2 bajtów danych z/do 16 stacji (2 bajty x 16),
- cykliczny przesył 16 bajtów danych z/do 16 stacji (2 bajty x 16),
- transfer bloku danych 128 bajtów z/do jednej stacji (128 bajtów x 1).

Pierwszy test odpowiada na pytanie, jak długo trzeba czekać na przesłanie powyższych danych, drugi, ile takich przesyłów możemy zrealizować w ciągu 1000 ms. Test drugi jest również miarą przepustowości użytecznej każdej z sieci.

Wyniki teoretycznych obliczeń czasów trwania transakcji poszczególnych wymian zamieszczono w tabeli 8 i graficznie zaprezentowano na rys. 69, a przepustowości użyteczne odpowiednio w tabeli 9 i na rys. 70.

Dla weryfikacji obliczeń teoretycznych zbudowano stanowisko laboratoryjne przedstawione na rys. 68.

Rys. 68. Konfiguracja stanowiska laboratoryjnego
Fig. 68. The configuration of test stand

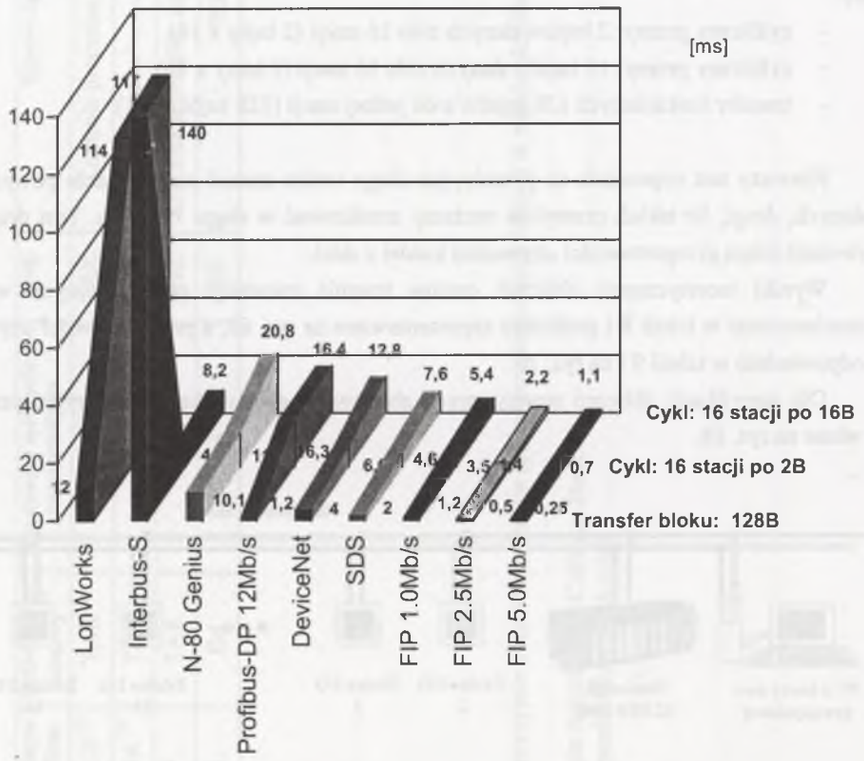
Pomiary, a więc weryfikację eksperymentalną, ze względu na ograniczenia sprzętowe przeprowadzono jedynie dla sieci FIP, N-80 oraz Modbus. Rezultaty eksperymentu zamieszczono w tabeli 8.

Stanowisko składało się z 16 stacji będących zdalnymi modułami wejść/wyjść, sterownika (PLC) pełniącego rolę arbitra dla sieci FIP oraz stacji Master dla sieci Modbus oraz komputera wyposażonego w kartę sieci przemysłowej.

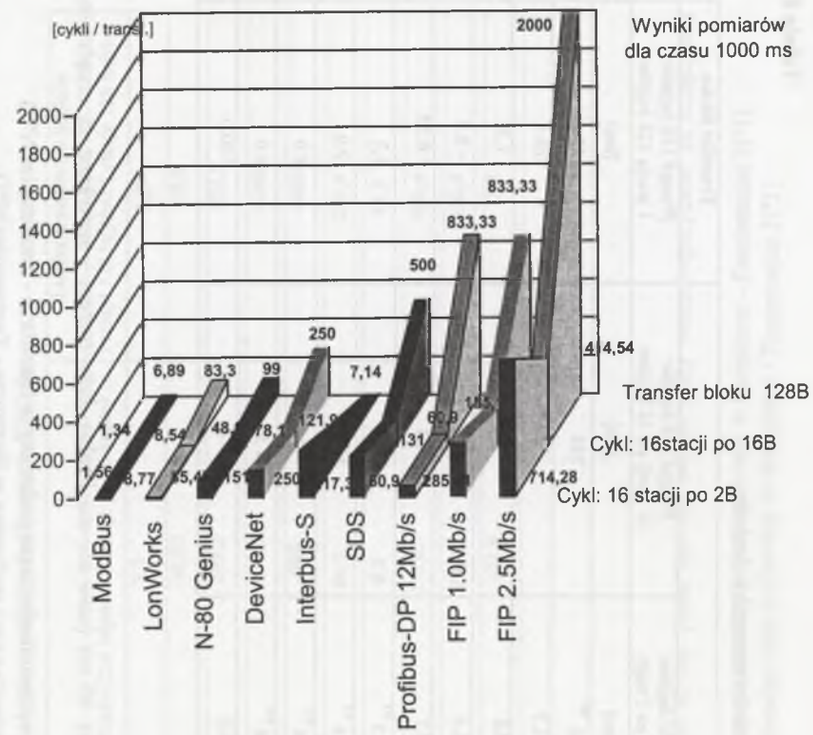
Przeprowadzono dwa eksperymenty polegające na przesyłaniu następujących danych:

- cykliczny przesył 2 bajów danych z/do 16 stacji (2B x16) oraz 16 bajtów danych z/do 16 stacji (16 bajtów x16),
- transfer bloku danych 128 bajtów z/do jednej stacji (128 bajtów x 1).

Wyniki praktycznych eksperymentów, jak widać, dobrze pokrywają się z wartościami uzyskanymi na drodze teoretycznej.



Rys. 69. Zestawienie czasów transmisji cyklicznych danych i transferu bloku
Fig. 69. The setting-up of cyclical transmission time and data blocks



Rys. 70. Zestawienie przepustowości poszczególnych sieci dla transmisji cyklicznych danych i transferu bloku danych (*dla czasu 1000 ms*)
Fig. 70. The setting-up of networks flow capacity for cyclical data transmission and data block *(for time 1000 ms)*

Wyniki obydwu testów pokazują, że sieć FIP zdecydowanie wyróżnia się spośród pozostałych sieci przemysłowych. Również sieci oparte na protokole CAN oraz sieci Profibus charakteryzują się dobrą przepustowością. Biorąc jednak pod uwagę parametry poszczególnych sieci zawarte w tabelach 4 i 6, widzimy, że sieć FIP zdecydowanie wyprzedza pod względem możliwości pozostałe. Jeszcze jedną bardzo ważną cechą sieci FIP jest to, że dodanie dodatkowych stacji monitorujących[39] nie wpływa na wydłużenie cyklu wymiany danych.

Biorąc pod uwagę wszystkie wyżej wymienione zalety oraz fakt, iż sieć FIP znajduje się na czele w procesie standaryzacyjnym organizacji IEC, można śmiało przypuszczać, że będzie to sieć dominująca na rynku w nadchodzącej dekadzie.

[39] Stacje, które nie transmitują danych (nie są producentami żadnej zmiennej).

Tabela 8

Zestawienie czasów transmisji danych

	Czas cyklu		Transfer bloku
	Przesył 32 bajtów: 16 stacji po 2 bajty	*Przesył 256 bajtów:* 16 stacji po 16 bajtów	*Przesył 128 bajtów:* 1 stacja 128 bajtów
	[ms]	[ms]	[ms]
Modbus	638 [40]	743	145
Genius, N-80	11.7	20.8	10.1
FIP 1.0 Mb/s	3,5	5,4	1.2
FIP 2.5 Mb/s	1,4	2,2	< 0.5
FIP 5.0 Mb/s	0,7	1,1	< 0.25
Profibus-DP 12Mb/s	16.3 [41]	16.4	1,2
LonWorks	114 [42]	117	12.0
SDS	4.6 [43]	7.6	2.0
DeviceNet	6.6 [44]	12.8	4.0
Interbus-S	4.0	8.2	140.0

[40] Przyjęto czas odpowiedzi stacji Slave na żądanie Mastera 30 ms (czas ten ma bardzo duży rozrzut - spotykane są urządzenia odpowiadające nawet po 1000 ms -. Simatic S-5)

[41] Przyjęto czas reakcji jednej stacji Profibus-DP równy 1.024 ms (minimalny czas podawany w literaturze - J'automatise N°1)

[42] Przyjęto czas reakcji dla stacji LonWorks równy 7.0 ms (minimalny czas podawany w literaturze - J'automatise N°1)

[43] Przyjęto czas reakcji dla stacji SDS równy 160 µs

[44] Przyjęto czas reakcji dla stacji DeviceNet równy 160 µs

Tabela 9

Zestawienie przepustowości sieci przemysłowych dla czasu 1000 ms

	Liczba cykli		Liczba transferów bloków
	Przesył 32 bajtów: 16 stacji po 2 bajty	Przesył 256 bajtów: 16 stacji po 16 bajtów	Przesył 128 bajtów: 1 stacja 128 bajtów
Modbus	1,6 [45]	1,3	6,9
Genius, N-80	85,5	48,07	99,0
FIP 1.0 Mb/s	285,7	185,2	833,3
FIP 2.5 Mb/s	714,3	454,5	>2000
FIP 5.0 Mb/s	1428	909	>4000
Profibus-DP 12Mb/s	60,9 [46]	60,9	833,3
LonWorks	8,8 [47]	8,5	83,3
SDS	217,4 [48]	131,6	500,0
DeviceNet	151,0 [49]	78,0	250,0
Interbus-S	250,0	121,9	7,2

[45] Przyjęto czas odpowiedzi stacji Slave na żądanie Mastera 30 ms (czas ten ma bardzo duży rozrzut-spotykane są urządzenia odpowiadające nawet po 1000 ms Simatic S-5)

[46] Przyjęto czas reakcji jednej stacji Profibus-DP równy 1.024 ms (minimalny czas podawany w literaturze - J'automatise N°1)

[47] Przyjęto czas reakcji dla stacji LonWorks równy 7.0 ms (minimalny czas podawany w literaturze - J'automatise N°1)

[48] Przyjęto czas reakcji dla stacji SDS równy 160 µs

[49] Przyjęto czas reakcji dla stacji DeviceNet równy 160 µs

Tabela 10

Zestawienie czasów transmisji danych - wyniki przeprowadzonych badań

	Czas cyklu		Transfer bloku
	Przesył 32 bajtów: 16 stacji po 2 bajty	*Przesył 256 bajtów:* 16 stacji po 16 bajtów	*Przesył 128 bajtów:* 1 stacja 128 bajtów
	[ms]	[ms]	[ms]
Modbus	660,0[w]	766,0[w]	150,0
Genius, N-80	12,1	22.0	10,4
FIP 1.0 Mb/s	3,6	5,6	1,2
FIP 2.5 Mb/s	1,4	2,3	0,5
FIP 5.0 Mb/s	0,7	1,1[e]	0,3[e]
Profibus-DP 12Mb/s	17,0[e]	17,0[e]	1,2
LonWorks	118,0	120,0	12,5
SDS	5.1	7,8	2,1
DeviceNet	7,2	13,2	4,1
Interbus-S	4,3	8,4	144,0

11.18. Kierunki rozwoju sieci przemysłowych

Prace, jakie prowadzone są przez organizacje i firmy zajmujące się rozwojem poszczególnych sieci przemysłowych, skupiają się przede wszystkim na następujących zagadnieniach:

- integracji z siecią IP,
- implementacji protokołu TCP/IP – Internet, Intranet, serwer WEB,
- zmniejszania czasochłonności i kosztów związanych z:
  - procesem konfiguracji,
  - diagnostyką,
- serwisowania,
- polepszania parametrów sieci, takich jak:
  - długości medium transmisyjnego,
  - szybkości transmisji, aby możliwe było przesyłanie danych multimedialnych,
- stosowania łączy bezprzewodowych – radiowych.

Siecią, na przykładzie której można obserwować ogólnoświatowe trendy, może być sieć FIP, dla której zdecydowano się na opracowanie nowej rodziny elementów elektronicznych (*FIPHSF, HSF Driver, HSF TR*), zwiększających szybkość transmisji, a przez to umożliwiających przesył danych multimedialnych. Uzyskano również możliwość uruchomienia serwera WEB poprzez zaimplementowanie protokołu TCP/IP (tabela 11).

Tabela 11

Rozszerzenie możliwości sieci FIP/WorldFIP

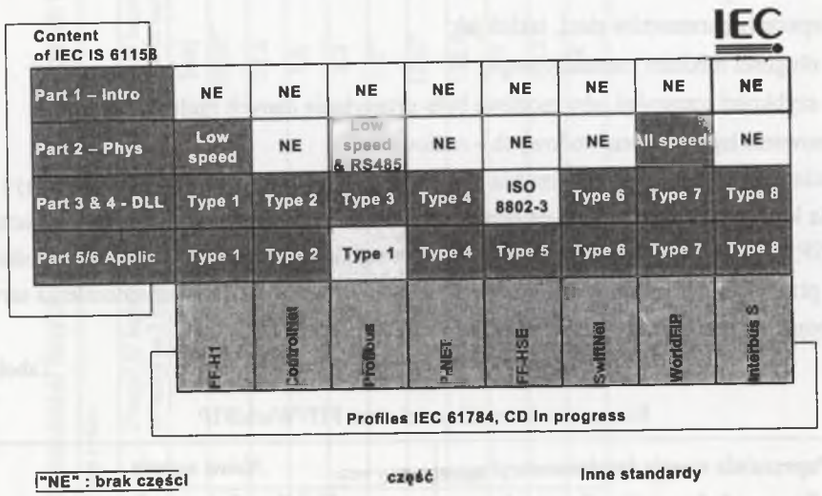
Poprzednia wersja implementacji **Factory Information Protocol**	***Nowa wersja*** **Fieldbus Internet Protocol**
31.25 Kb/s–2.5 Mb/s (5 Mb/s), Redundancja medium transmisyjnego Determinizm czasowy Wymiany aperiodyczne Arbiter magistrali	31.25 Kb/s - 25 Mb/s Redundancja medium transmisyjnego Determinizm czasowy Wymiany aperiodyczne Arbiter magistrali Komunikaty IP TCP / UDP Serwer WEB Przesył danych multimedialnych

11.19. Stan prac normalizacyjnych

Stan prac normalizacyjnych organizacji IEC dla sieci spełniających kryteria sieci polowych przedstawiony jest na rys. 71.

Jak widzimy, tylko sieć FIP/WorldFIP posiada gotowe normy dla warstw: aplikacji, łącza i fizycznej. Pozostałe: FF-H1 i Profibus posiadają niekompletne normy dla warstwy fizycznej, reszta posiada tylko normy dla warstwy aplikacji i łącza (DLL).

Z uwagi na tendencje międzynarodowych organizacji standaryzacyjnych (IEC) rozwijane będą jedynie te sieci, które spełniają surowe kryteria sieci polowych, i tylko one otrzymają status międzynarodowego standardu (ang. *„Multi Protocol Standard"*).



Rys. 71. Stan prac normalizacyjnych organizacji IEC dla normy sieci polowych – „Multi Protocol Standard"

Fig. 71. The state of normalisation works of IEC Organisation for field-bus standard – „Multi Protocol Standard"

12. WNIOSKI KOŃCOWE

W niniejszej pracy przedstawiono analizę przepływu informacji w sieciach przemysłowych o zdeterminowanym w czasie dostępie do łącza. Poddano analizie zjawiska, mające podstawowy wpływ na całkowity czas wymiany informacji pomiędzy abonentami, które można generalnie podzielić na trzy grupy:

- zjawiska związane z budową i zasadą działania sterownika swobodnie programowalnego oraz realizacją programu użytkowego w nim umieszczonego,
- zjawiska zachodzące na styku „jednostka centralna sterownika swobodnie programowalnego-koprocesor sieci",
- zjawiska mające miejsce w koprocesorze sieci oraz na styku „koprocesor-medium transmisyjne" związane z procesem transmisji i przyjętym jej protokołem.

Wśród wymienionych zjawisk są takie, których występowanie jest niezależne od przyjętego protokołu transmisji. Dotyczy to zarówno struktury cyklu pracy sterownika swobodnie programowalnego, realizacji programu aplikacyjnego, jak i ogólnie dostępnych, typowych funkcji wykonywanych przez koprocesor sieci, do których można zaliczyć na przykład realizację wymian cyklicznych czy też wyzwalanych, które to wymiany są dostępne w oderwaniu od protokołu transmisji.

Pozostałe zjawiska mają już ścisły związek z przyjętym protokołem wymiany informacji w sieci i te właśnie były głównym celem analizy i badań. Wśród protokołów, poddanych analizie czasowej, a stosowanych w rozproszonych systemach przemysłowych, znalazły się tylko te, które cechuje zdeterminowany czas dostępu, a więc:

- TOKEN BUS (TOKEN – RING),
- MASTER–SLAVE,
- PRODUCENT – DYSTRYBUTOR – KONSUMENT stanowiący podstawę sieci FIP.

Dla każdej z wymienionych sieci podano nie tylko opis wymiany informacji, ale również przeanalizowano, obok zjawisk zawsze występujących na styku „koprocesor-jednostka centralna", takie jak choćby powstawanie „dziur", co może mieć kolosalne znaczenie dla prawidłowej pracy sieci, szczególnie wtedy gdy parametry czasowe są krytyczne. W każdym przypadku podano maksymalne czasy wymiany informacji danych użytkowych oraz określono sprawności i przepustowości, a w przypadku sieci FIP dokonano dodatkowo analizy statycznej i dynamicznej wymiany informacji. Zwrócono uwagę na problem buforowania

informacji i parametryzacji wymian. Podano wiele wskazówek mających na celu optymalizację pracy sieci, a dotyczących nie tylko opracowywania scenariuszy wymian, ale również programów aplikacyjnych realizowanych przez abonentów sieci. Poddając analizie czasowej procesy transmisji, określono lub zdefiniowano cały szereg parametrów, których ustalenie wartości, na etapie projektowania sieci, ma decydujące znaczenie dla jej prawidłowej pracy. Przedstawiono również cechy i wymagania stawiane przed przemysłowymi rozproszonymi systemami czasu rzeczywistego. Przeanalizowano metody integracji sieci najniższego poziomu, a także przedyskutowano niezwykle aktualne zagadnienia, związane z zastosowaniem protokołu TCP/IP w systemach informatyki przemysłowej. Zamieszczono również skrótowy przegląd najpopularniejszych, współcześnie stosowanych sieci przemysłowych, a także teoretyczne wyniki badań nad ich sprawnością i przepustowością, a w odniesieniu do dwóch z nich przedstawiono również wyniki eksperymentów praktycznych, które dobrze potwierdzają wyniki analizy przepływu danych, zaproponowanej przez autora w niniejszej pracy. Zasygnalizowano zarówno spodziewane kierunki rozwoju sieci przemysłowych, i również stan prac normalizacyjnych.

Zamysłem autora było zwrócenie uwagi na zjawiska zachodzące w procesie transmisji, co w konsekwencji winno zmusić do analizowania w czasie przepływu danych w sieciach przemysłowych. Powinno również wymusić zwracanie baczniejszej uwagi na parametry sprzętu elektronicznego (sterowniki, koprocesory, „inteligentne" moduły zdalnych wejść / wyjść), które to parametry nieczęsto są podawane przez producentów, a mogą mieć decydujący wpływ na poprawną pracę całego systemu. Zawarto także pewne wskazówki praktyczne dotyczące optymalizacji procesu transmisji. Podstawowym kryterium owej optymalizacji jest czas, będący krytycznym parametrem systemu rozproszonego czasu rzeczywistego. Optymalizacja będzie polegać na tym, aby stosując pewne mechanizmy, skrócić czas realizacji pojedynczej wymiany, a w konsekwencji skrócić czas cyklu wymiany informacji w całym systemie. Jest to związane nie tylko z optymalizacją wymian w sieciach, ale również z optymalizacją, dokonywaną pod kątem szybkości realizacji, oprogramowania zawartego w jednostkach centralnych sterowników swobodnie programowalnych. Zwrócono uwagę na wiele czynników będących parametrami wpływającymi tak na szybkość wymiany danych, jak i na poprawność transmisji wymian.

Fundamentalnym, jak się wydaje autorowi, wnioskiem, wypływającym z rozważań przedstawionych w pracy, jest absolutna konieczność oszacowania liczby i mierzonej w bitach długości wymian informacji w sieci, co w połączeniu z proponowanymi metodami analizy pozwala na określenie bądź oszacowanie czasu trwania cyklu wymiany informacji w sieci, a tym samym podanie parametrów krytycznych jej poprawnej pracy.

Z treści pracy wynika również, zdaniem autora, pewien zarys metodologii projektowania sieci. Padło słowo „zarys", gdyż trudno jest sformułować absolutnie jednoznaczną i obiektywną, obowiązującą zawsze regułę postępowania. Podstawowym elementem owej metodologii jest tak zwany scenariusz wymian, który, w zależności od trybu pracy zespołu ludzkiego, programującego pracę systemu, powstaje albo niezależnie od pracy całej sieci i jest ściśle związany z pracą pojedynczego abonenta, a następnie jest scalany przez osobę odpowiedzialną za pracę sieci, albo też jest tworzony przez programistę odpowiedzialnego za komunikację. I to on, znając podstawowe parametry technologiczne, narzuca abonentom tryb i sposób transmisji. Z praktycznych doświadczeń autora wynika, że najczęściej stosuje się pierwszy z opisanych sposobów. Każdy z programistów, tworzących programy aplikacyjne dla poszczególnych stacji abonenckich, wypełnia formularz, przekazując tym samym integratorowi sieci swoje podstawowe wymagania dotyczące:

- liczby wymian cyklicznych i ilości informacji do transmisji związanych z każdą taką wymianą,
- czasu repetycji każdej wymiany cyklicznej,
- liczby wymian wyzwalanych i ilości informacji do transmisji związanych z każdą wymianą wyzwalaną,
- pilności realizacji wymian,
- krytycznego czasu realizacji pojedynczej wymiany i całkowitego czasu realizacji wszystkich wymian (cykl wymiany informacji dla określonego abonenta).

Dodatkowymi informacjami, którymi musi dysponować osoba integrująca to przede wszystkim minimalny i maksymalny czas trwania cyklu aplikacji realizowanej w stacji abonenckiej (czas trwania cyklu automatu). Parametr ten, jak dowodzi niniejsza praca, ma poważny wpływ na całkowity czas trwania cyklu wymiany informacji w sieci. Na podstawie wspomnianych danych powstaje pełny scenariusz wymian, a po analizie (zależnej od przyjętego protokołu) integrator sieci otrzymuje odpowiedź na pytanie, czy stawiane przez abonentów wymagania są do spełnienia. Jeśli odpowiedź jest pozytywna, to niestety, w większości przypadków zaobserwowanych w praktyce przez autora, nie następuje optymalizacja wymian, której wynikiem powinno być nie tylko polepszenie parametrów pracy już skonfigurowanej sieci, ale również określenie parametrów krytycznych dotyczących na przykład liczby wymian o określonej pojemności, które mogą być w przyszłości zrealizowane bez żadnych perturbacji w systemie, pozwalając mu pracować poprawnie. Konsekwencją takiego postępowania jest zazwyczaj ogromne zdziwienie twórców systemu, że jakakolwiek nawet najmniejsza jego modyfikacja, polegająca na wprowadzeniu choćby kilku nowych wejść binarnych, powodująca wzrost natężenia ruchu w sieci, prowadzi do jego załamania. Zatem lepiej jest, gdy po analizie czasowej, odpowiedź na postawione już pytanie jest negatywna. Wtedy na pewno nastąpi proces optymalizacji, który będzie związany z ponowną analizą wymagań technologii. Znane są bowiem autorowi przypadki, gdy zgłaszano żądania transmisji co 100 ms wielkości poziomu wody w zbior-

niku o kontrolowanej pojemności Δ=40 000 litrów, z którego ciecz mogła być odprowadzana bądź doprowadzana z szybkością około 100 litrów na minutę. Następnym krokiem optymalizacji jest zmiana oprogramowania aplikacyjnego i/lub weryfikacja scenariusza wymian, mogąca polegać na przykład na zmianie kolejności realizacji transakcji wymiany danych, dołożeniu dodatkowej wymiany lub eliminacja którejś. Modyfikacja może również polegać na zmianie trybu realizacji z wymiany cyklicznej na wyzwalaną.

Pozostaje jeszcze manipulacja takimi parametrami, jak prędkość transmisji czy liczba bitów przypadająca na jeden znak transmisji. Z wyników analizy zawartych w niniejszej pracy wynika, że jest to bardzo wąski margines, po którym można się poruszać. Będzie to miało znaczenie tylko wtedy, gdy już zbliży się do warunków krytycznych pracy sieci.

Gdy wszystkie wspomniane metody zawiodą, pozostaje jedynie bądź zmiana protokołu dostępu, bądź jeszcze większe rozproszenie systemu (zwiększenie liczby segmentów sieci nie zawsze dające pożądane rezultaty), bądź też nawet zmiany w samym obiekcie polegające na zamianie abonentów obsługujących pewne dane (zmiana „okablowania"). I znów z doświadczeń autora wynika, że jeśli ma miejsce optymalizacja, to zawsze powstaje dokument stwierdzający, jak dalece zaprojektowaną, istniejącą sieć można jeszcze w przyszłości „dociążyć" dodatkowymi transmisjami

Pragnę na koniec zwrócić uwagę, że sposoby analizy pracy sieci, jak i przede wszystkim jej rezultaty, w każdym z trzech przypadków pracy sieci, znalazły praktyczną weryfikację przez aplikację na dużych, bardzo odpowiedzialnych instalacjach przemysłowych, do których zaliczyć należy:

- System automatycznego sterowania i wizualizacji procesu przygotowania wody w Stacji Przygotowania Wody Elektrowni „Trebovice"- Republika Czeska (sieć „TOKEN – BUS"- około 4000 wejść/wyjść),
- System automatycznego sterowania i wizualizacji instalacji chłodniczej w Zakładach Przetwórstwa Mięsnego w Tarnowie (sieć „MASTER-SLAVE" – około 800 wejść/wyjść),
- System automatycznego sterowania i wizualizacji instalacji odpylania, transportu popiołu, regeneracji filtrów wraz z automatyczną regulacją obrotów wentylatorów dużej mocy z zastosowaniem przemienników częstotliwości w Elektrociepłowni „Miechowice" (sieć „FIP" – funkcje szybkich zabezpieczeń oraz sieć „MASTER-SLAVE" – około 1500 wejść/wyjść).

Wydaje się, że właśnie dzięki proponowanym w opracowaniu sposobom analizy pracy sieci, wspomniane w praktyce sieciowe systemy sterowania, wizualizacji i monitorowania, pracują niezawodnie od wielu lat.

Na zakończenie pragnę odesłać Czytelnika do pracy [88], gdzie poddano praktycznej, laboratoryjnej weryfikacji część rezultatów przedstawioną w niniejszym opracowaniu. Zamieszczono w nim wyniki badań sieci i opisano zjawiska, a także przypadki niespełnienia wymagań czasowych rozproszonego systemu czasu rzeczywistego wskutek niebrania pod uwagę pewnych, na pozór błahych, parametrów czasowych koprocesorów sieci.

13. SKOROWIDZ NAJCZĘŚCIEJ UŻYWANYCH OKREŚLEŃ I SYMBOLI

Cykl automatu T_A	Czas realizacji (najgorszy przypadek) pętli programu aplikacyjnego rezydującego w PLC będącym węzłem systemu.
Cykl pracy sieci T_C	Czas, który upływa pomiędzy dwoma kolejnymi dostępami do medium przez dowolnego abonenta.
Cykl wymiany informacji T_{CW}	Czas niezbędny do wyemitowania wszystkich niezbędnych w danej chwili danych przez wszystkich abonentów sieci.
FIP *ang. „Factory Instrumentation Protocol"*	Nazwa firmowa protokołu o dostępie Producent-Dystrybutor- Konsument.
Graniczny czas reakcji systemu T_G	Graniczny czas reakcji całego systemu na dowolne zdarzenie, które może wystąpić na obiekcie. Służy do określenia granic stosowalności całego systemu informatycznego.
Koprocesor	Urządzenie specjalizowane (lokalny komputer z własnym procesorem), współpracujące z PLC i korzystające z jego magistrali. Sterownik PLC może być wyposażony w różne typy koprocesorów, np. numeryczny, sieciowy, języka BASIC lub C++. W niniejszej pracy słowo koprocesor odnosi się do urządzenia sterującego dostępem do sieci przemysłowej i zarządzającym przepływem danych do i ze sterownika PLC.
„Migotanie" przerwań	Zjawisko mające miejsce wtedy, gdy zdarzenia obiektowe (zachodzące na obiekcie przemysłowym) pojawiają się lawinowo, a są obsługiwane przez układ przerwań. Wtedy następuje „zagwożdżenie" systemu, gdyż nie robi on nic innego, jak tylko obsługuje przerwania o różnych priorytetach.

Ramka	Jednostka transmisji danych w sieci komputerowej. Ramka zawiera, oprócz danych użytkowych, również szereg informacji sterujących i kontrolnych.
Scenariusz wymian	Lista sekwencyjnie zaplanowanych transakcji wymian informacji, realizowanych przez stacje uprzywilejowane (np. stacja arbitra, stacja MASTER, stacja posiadająca żeton). Postać listy jest elementem projektowania systemu komunikacyjnego. Zawiera ona cały szereg parametrów przypisanych do każdej wymiany, takich jak np. czas repetycji (periodyczność), identyfikator (np. nazwa lub numer), funkcję (np. zapis lub odczyt).
Sterownik swobodnie programowalny, automat *ang. PLC-„Programmable Logic Controller"*	Specjalizowany komputer do zastosowań przemysłowych, przeznaczony do realizacji funkcji automatu kombinacyjnego i/lub automatu sekwencyjnego.
Strumień sterowania (sekwencja sterowania)	Ciąg rozkazów lub postać pamięci PLC stanowiąca wektor sterowania.
System rozproszony czasu rzeczywistego *ang. DRTS – „Distribiuted Real Time System"*	System komputerowy, w którym mamy do czynienia z rozproszeniem sprzętu i/lub sterowania, a reakcja systemu musi nastąpić w określonym interwale czasowym.
Systemy klasy MIMD *ang. „Multiple Instructions Multiple Data"*	Systemy rozproszone, w których sprzęt jest scentralizowany, a rozproszone jest sterowanie.
Systemy klasy SIMP *ang. „ Single Instructions Multiple Data"*	Systemy rozproszone, w których sprzęt jest zdecentralizowany, ale występuje scentralizowane sterowanie.
Systemy o ostrych ograniczeniach („twarde") *ang. „hard real time systems"*	Systemy czasu rzeczywistego, w których niedopuszczalne jest przekroczenie parametrów czasowych.
Systemy o słabych ograniczeniach („miękkie) *ang. „soft real time systems"*	Systemy czasu rzeczywistego, w których dopuszczalne jest przekroczenie parametrów czasowych.
Systemy rozproszone o zdeterminowanym czasie dostępu do systemu komunikacyjnego, sieci o zdeterminowanym czasie dostępu do medium	Systemy (sieci komputerowe), w których jest gwarantowany czas dostępu do funkcji nadawania i odczytu.

Systemy typu LON *ang. „Local Operating Network"*		Systemy rozproszone, stosujące jako węzły specjalizowane, realizujące bardzo proste funkcje, urządzenia typu „zadajnik sygnału-urządzenie wykonawcze".
Transmisje periodyczne		Transmisje realizowane sekwencyjnie, zgodnie ze scenariuszem wymian.
Transmisje wyzwalane		Transmisje realizowane na żądanie macierzystej jednostki centralnej węzła sieci.
UART *ang. „Unit of Asynchronous Remote Transmission"*		Urządzenie (układ scalony) do realizacji szeregowej transmisji asynchronicznej.
T_K	–	czas transakcji wymiany „koprocesor–jednostka centralna" pojedynczej ramki danych,
T_P	–	czas przetwarzania pojedynczej ramki,
T_A	–	czas trwania cyklu automatu (czas trwania cyklu sterownika),
LBN	–	liczba buforów nadawczych,
LBO	–	liczba buforów odbiorczych,
T_N	–	czas nadawania,
JC	–	jednostka centralna nadawcy,
KO	–	koprocesor sieciowy nadawcy,
JC'	–	jednostka centralna odbiorcy,
KO'	–	koprocesor sieciowy odbiorcy,
KC	–	„koniec cyklu automatu" – sygnał przerwania JC do koprocesora,
T_{ODj}	–	maksymalny,definiowany czas odpowiedzi *(ang. „time-out)*,
T_{Ni}	–	definiowany czas nadawania,
T_A	–	czas trwania cyklu automatu (najgorszy przypadek),
T_{PR}	–	czas programowego przygotowania ramki,
T_{DR}	–	czas detekcji ramki (jest to na przykład w metodzie kodowania MANCHESTER2, czas transmisji 3.5 znaku przy przyjętej prędkości transmisji),
T_{AR}	–	czas analizy ramki,
T_{TR}	–	czas transmisji ramki
T_{OD}	–	czas odpowiedzi stacji SLAVE dla pojedynczej transakcji wymiany, będący parametrem,
T_{GOT}	–	czas gotowości jednostki centralnej stacji SLAVE, będący parametrem,
T_{Pri}	–	czas przygotowania ramki żądania,

T_{TRi}	–	czas transmisji ramki żądania,
T_{DR}	–	czas detekcji ramki,
T_{ARj}	–	czas analizy ramki żądania w stacji SLAVE,
T_{Aj}	–	czas trwania cyklu automatu jednostki centralnej stacji SLAVE,
T_{PRj}	–	czas przygotowania ramki odpowiedzi,
T_{TRj}	–	czas transmisji ramki odpowiedzi,
T_{ARi}	–	czas analizy ramki odpowiedzi,
T_{Ai}	–	czas trwania cyklu automatu jednostki centralnej stacji MASTER,
V	–	prędkość transmisji,
LZT	–	liczba znaków w ramce zawierającej dane,
LBZ	–	liczba bitów przypadających na jeden znak transmisji,
BS	–	liczba bitów sterujących, które „dokłada" warstwa liniowa,
DTR	–	preambuła,
CP	–	pole kodu funkcji,
ID	–	pole identyfikatora zmiennej,
FCS	–	pole sumy kontrolnej,
PDU	–	pole typu PDU,
Długość PDU	–	pole liczby bajtów danych,
Typ PDU	–	pole typu danych,
Dane	–	pole danych stanowiących wartość zmiennej,
Status	–	pole statusu produkcji,
Komunikat	–	pole danych stanowiących komunikat,
Przeznaczenie, Źródło	–	pole adresowe abonentów,
FTR	–	postambuła,
L_{p1}, L_{p2}, L_{p3}	–	długość listy zmiennych i komunikatów periodycznych,
L_{a1}, L_{a2}	–	długość listy zmiennych aperiodycznych,
L_{m1}, L_{m2}	–	długość listy komunikatów (ACK/ NOACK),
T_{ID_DAT}	–	czas transmisji ramki identyfikatora,
$T_{ID} = T_{ID_prod} + T_{ID_DAT}$	–	czas produkcji oraz transmisji ramki identyfikatora,
$T_{RP_DAT} = T_{xxx_min} .. T_{xxx_max}$	–	czas transmisji ramki RP_DAT,
$T_{RP_RQi} = T_{xxx_min} .. T_{xxx_max}$	–	czas transmisji ramki RP_RQi,
$T_{RP_MSG_NOACK} = T_{xxx_min} .. T_{xxx_max}$	–	czas produkcji ramki, RP_MSG_NOACK,
$T_{RP_MSG_ACK} = T_{xxx_min} .. T_{xxx_max}$	–	czas produkcji ramki RP_MSG_ACK,
T_{RP_ACK}	–	czas produkcji ramki RP_ACK,
T_{RP_FIN}	–	czas produkcji ramki RP_FIN,
T_{resp}	–	czas oczekiwania na odpowiedź abonenta,

$T_{tout} = T_{resp_max}$ – maksymalny czas oczekiwania (*ang. timeout*) na odpowiedź abonenta,
T_{resp_Ab} – czas oczekiwania na potwierdzenie komunikatu,
$T_{tout_Ab} = T_{resp_Ab_max}$ – czas oczekiwania (*ang. timeout*) na potwierdzenie komunikatu,
$Q_{rep} = 0..3$ – liczba powtórzeń komunikatów,
Q_{RP_RQi} – jest liczbą żądanych zmiennych przez danego abonenta (wartość dynamiczna),
T_{toutAb} – maksymalny czas oczekiwania *(ang.time-out)* na potwierdzenie komunikatu,
T_{read} – czas odczytu bufora komunikacyjnego (*ang. link layer*),
T_{proc} – czas przetwarzania informacji (*ang.link & application layer*),
T_{write} – czas zapisu bufora komunikacyjnego (*ang.link layer*),
T_{Arb_proc} – jest czasem obsługi ramki RP_RQi przez arbitra (*ang. "link & application layer"*),
$T_{as} = T_{read} + T_{proc} + T_{write}$ – czas obsługi ramki przez abonenta,
T_{xxx} – ramka danego typu, którego tyczy się opis,
T_{mac} – czas transmisji jednego bitu,
T_{tr} – czas „milczenia" sieci,
Ptr – prędkość transmisji,
T_{ID_xxx} – czas transmisji ramek zapytań,
T_{RP_xxx} – czas transmisji ramek odpowiedzi,
Q_{rep} – liczba powtórzeń,
L_{p1} – długość listy periodycznej zmiennych,
L_{p2}, L_{p3} – długość listy periodycznej komunikatów,
T_{tout} – czas oczekiwania dla danej zmiennej,
$T_{resp_Ab_max}$ – max czas oczekiwania na potwierdzenie komunikatu,
T_{tr} – czas milczenia sieci,
T_{mac} – czas transmisji jednego bitu,
T_{as} – czas obsługi ramek przez abonenta,
T_d – czas transmisji ramki danych,
T_{resp} – czas odpowiedzi abonenta,
T_{resp_Ab} – czas oczekiwania na potwierdzenie komunikatu,
L_{a1}, L_{a2} – długość listy aperiodycznej (p/n),
L_{m1}, L_{m2} – długość aperiodycznej listy komunikatów,
Q_{RP_RQi} – liczba żądań aperiodycznych,
T_{Arb_proc} – czas obsługi ramki RP_RQi przez arbitra.

LITERATURA

1. Accetta M., Baron R.,Golub D.: A new kernel foundation for UNIX development. In Accetta Proc. of Summer 1986 USENIX Conf., 1986.
2. Ahuja M., Carlson T.,Gahlot A.,Shands D.: Timestamping events for interferring affects relation and potential casualty. In Proc of IEEE 15 Int.'l Computer Software and Application Conf., Tokyo, September 1991.
3. ALSPA Communication Networks. Materiały firmowe firmy Cegelec. Paryż 1994.
4. Anand T.S., Gupta R.: A tool for evaluating compiler-based parallelization strategies. Mathematics and Computers in Simulation vol. 31, 1989.
5. Anderson T., Lemos de R., Saeed A.: Analysis of Safety Requirements for Process Control System, in: Predictably Dependable Computing System, Eds: B. Randell, J. C. Laprie, B. Littlewood, H. Kopetz, pp. 27-40, Springer, Berlin 1993.
6. Arvind K., Ramamritham K., Stankovic J. A.: A local area network architecture for communication in distributed real-time systems, Real –Time Systems Journal, Vol. 3, No. 2, May 1991.
7. Azencott R.: Simulated anneling and parallelization techniques. John Wiley&Sons, New York 1992.
8. Balaji S. et al. S-Nets.:A Petri net based model for performance evaluation of real-time scheduling algorithms. J.Parallel Distrib. Comput. 1992.
9. Baxter M. J. and all: Development framework approach to heterogeneous system design for control systems, Control Eng. Practice, Vol. 4, No. 2, pp. 229-238, 1996.
10. Baxter M.J. and all: Development framework approach to heterogeneous system design for control systems. Control Eng. Practice, Vol. 4, No 2, 1996.
11. Bayne J.S.: A distributed programming model for real-time industrial control. Control Eng. Practice, Vol. 8, No. 2, 1995.
12. Beale G. O.: Adaptive Integration Operators for Real-Time Digital Simulation, IEEE Trans. on Industrial Electronics, Vol. 34, No. 1, pp. 65-69, 1987.
13. Beale G. O.: Real-time simulation of dynamic systems on a pyramid architecture, IEEE Transaction on Industrial Electronics, Vol. 37, No. 3, pp. 212-220, 1990.
14. Belschner R.: Simulation of distributed real-time systems, Proc. European Simulation Symposium, Istambul, 1994.

15. Bertsekas D. P., Tsitsiklis J. N.: Some aspects of parallel and distributed iterative algorithms – A survey, Automatica, Vol. 27. No. 1, pp. 3-21, 1991.
16. Bertsekas D.P., Tsitsiklis J.N.: Some aspects of parallel and distributed interactive algorithms – A survey, Automatica Vol. 27. No. 1, 1991.
17. Bi. Y.D. and Tsai. J. J.-P.: Inegrating data collection, debugging, and vizualization. In Proc. of 7th Int.'l Conf. on Advanced Science and Technology, Argonne, IL, pp. 267-279, March 1991.
18. Bigewski Z.: Optymalizacja pracy sieci przemysłowych. ZN Pol. Śl. s. Informatyka z. 28, Gliwice 1995.
19. Boroń W.: Zastosowanie Internetu do nadzoru systemu automatyzacji budynku. Studia Informatica Vol.21 No.31, Gliwice 2000.
20. Burns A. and Wellings A.: Real-Time Systems and Thier Programming Languages. Addison-Wesley, Reading, MA, 1990.
21. Caban D., Fojcik M., Małysiak H., Zieliński B.: System transmisji radiowej dla sieci przemysłowej. Zeszyty Naukowe Pol. Śląskiej, s.Informatyka z.32, Gliwice 1997.
22. Carcagno L., dours D., Facca R., Sautet B.: Distributed Hard-Real-Time Systems: from specification to Realisation, Repr. 13th IFAC Workshop on Distributed Computer Control Systems, Toulouse, pp. 49-54, 1995.
23. Cardeira C., Mammeri Z.: Preemitive and Non-Preemitive Real-Time Scheduling Based on Neural Networks, Repr. 13th IFAC Workshop on Distributed Computer Control Systems, Toulouse, pp. 67-72, 1995.
24. Castori P.: An Event-based Algorithm for Distributed Deadlock Detection Resolution, Proc. Of the European Research Seminar on Advances in Distributed Systems, pp. 311-316, 1995.
25. Castori P.: Distributed Concurrency Control Algorithms with an OSI Application Layer Protocol in Hetero geneous Environment, Proc. of 2nd Int. Conf. on Industrial Automotion, Nancy, France, June 1995.
26. Castori P.: Software Design Document for the MMS Event Management Server, Tech. Rep. Laboratoire d'Informatique Technique Ecole Polytechnique Federale de Lausanne, 1991.
27. Chang W.-T., Kalavade A.: Lee E. A., Effective heterogeneous Design And Co-Simulation, NATO Advanced Study Institute Workshop on Hardware/Software Codesign, 1995.
28. Chodorow S. E., Jahanian F. and Donner M.: Run-time monitoring of real-time systems. In Proc. of IEEE 12th Real-Time Systems Symp., San Antonio, TX, pp. 74-83, December 1991.

29. Choiński D.: Data Flow Tokens for Real–Time Simulation, Proc. 7th Symposium System Modelling Control, Vol. 1, Zakopane 1993.
30. Choudhary et al. A. L.: A modified priority-based probe algorithm for distributed deadlock detection and resolution. IEEE Trans. on Software Engineering, 15(1):10-17, January 1989.
31. Comer D.: Sieci komputerowe TCP/IP. WNT, Warszawa 1998.
32. Communications Networks. Hardware description. Dokumentacja techniczna firmy Cegelec, Paryż 1989.
33. Communications Networks. Programming Manual. Dokumentacja techniczna firmy Cegelec, Paryż 1989.
34. Cupek R., Fojcik M.: Budowa modułów komunikacyjnych stacji nadzorczej z sieciami przemysłowymi. Zeszyty Naukowe Pol. Śląskiej, s. Informatyka z.32 Gliwice 1997.
35. Cupek R.: Metody hierarchizacji rozproszonych procesów przemysłowych. ZN Pol. Śl. s. Informatyka z. 28, Gliwice 1995.
36. Cupek R.: Metody wizualizacji rozproszonych procesów przemysłowych. Praca doktorska. Instytut Informatyki Pol. Śląskiej, Gliwice 1998.
37. Cupek R.: Protokół TCP/IP w systemach wizualizacji procesów przemysłowych. Studia Informatica Vol.22 No.3. Gliwice 2001.
38. Czachórski T., Atmaca T., Fourneau J.M., Pekergin F.: Dynamika przepływu strumieni pakietów w sieci, próby sterowania-modele dyfuzyjne. ZN Pol.Śl. s.Informatyka z.32 Gliwice 1997.
39. Czachórski T.: Modele kolejkowe w analizie i ocenie efektywności sieci komputerowych. ZN Pol.Śl. s.Informatyka z. 30, Gliwice 1996.
40. Czajka A.: Statystyczne szeregowanie zadań o okresach binarnych w systemach silnie uwarunkowanych czasowo. Rozprawa doktorska, Poznań 2000.
41. Davoli R., Giachini L.–A.: Schedulability Checking of Data Flow Task in Hard–Real–Time Distributed Systems, Technical Report UBLCS–94–4, University of Bologna, 1994.
42. Deminet J.: System operacyjny RSX-11. WNT, Warszawa 1986.
43. Dieterle W., Kochs H.–D., Dittmar E.: Communication architectures for distributed computer control system, Control Eng, Practice, Vol. 3, No. 8, pp. 1171-1176, 1995.
44. Eager D.I, Zahorjan j., Lazowska E. D.: Speedup versus efficiency in parallel systems, IEEE Transaction on Computers, vol. 38, No. 3, 1989.
45. ECHELON: LonWorks Products databook, 1995.
46. EN50170 General Purpose Field Communication System, CENELEC 1996.
47. Fidge C.J., Wellings A.J.: An action–based formal model for concurrent, real–time systems. Tech. Rap. YCS–95–249, The Univ. Of York, Real–Time Systems Research Group, 1995.

48. Figwer J.: Laboratorium metod optymalizacji. Politechnika Śląska, Gliwice 1994.
49. Filipowicz B.: Analiza i synteza systemów obsługi i sieci kolejkowych. WNT, Warszawa 1996.
50. FIP Bus Controller (FBC) for Alspa C80–35 PLC. Dokumentacja sterowników PLC firmy Cegelec. Paryż 1995.
51. FIP Device Manager. User reference manual. Dokumentacja techniczna firmy Cegelec. Paryż 1992.
52. FIP NETWORK General Introduction; DPS 50249 aA, Clamart 1990.
53. FIP Protocol. World FIP Europe, Nancy 1996.
54. FIP STARTER KIT materiały klubowe WorldFip; Clamart 1995.
55. FIP Toolbox. Centre de Competence WORDFIP, Nancy 1994.
56. FIPCODE Software version 5 User Reference Manual; DPS 50263 b–en, Clamart 1994.
57. FIPGEN. Dokumentacja oprogramowania firmy Cegelec. Paryż 1992.
58. Fojcik M.: Ograniczenia w wymianie informacji w zintegrowanych sieciach przemysłowych. Zeszyty Naukowe Pol. Śląskiej, s. Informatyka z.28, Gliwice 1995.
59. Fojcik M.: Problemy integracji sieci przemysłowych. Zeszyty Naukowe Pol. Śląskiej, s. Informatyka z.36, Gliwice 1999.
60. Gabrielian A. and Franklin M. K.: Multilevel specification of real-time systems. Comm. of ACM, 34(5):51-60, May 1991.
61. Gaj P.: Szybka sieć przemysłowa a system wizualizacji – problem interfejsu. ZN Pol.Śl. s.Informatyka z.36, Gliwice 1999.
62. Goldberg E.D.: Algorytmy genetyczne i ich zastosowanie. WNT, Warszawa 1995.
63. Goldberg E.D.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley Publishing Company 1989.
64. Gommaa H.: A software design method for real-time systems, Communication of the ACM, Vol. 27, No. 9, pp. 938-948, 1984.
65. Gonera P.: Program edukacyjny dla sieci FIP. Praca dyplomowa, Instytut Informatyki Pol. Śl,. Gliwice 1995.
66. Goszczyński J.: Przykłady zastosowań sieci LonWorks na świecie i w kraju. Pomiary Automatyka Robotyka 2/1998, ss. 5-8.
67. Grzywak A., Kwiecień A. i inni: Laboratorium sieci komputerowych. Skrypt Pol.Śl., Gliwice 1999.
68. Grzywak A.: Bezpieczeństwo w sieciach rozproszonych. ZN Pol. Śl. s. Informatyka z. 24, Gliwice 1993.
69. Grzywak A.: Problemy integracji sieci komputerowych. ZN Pol. Śl. s. Informatyka z. 23, Gliwice 1993.

70. Haban D.: DTM-A method for testing distributed systems. In Proc. of 6th Symp. On Reliability in Distributed Software and Database Systems, Williamsburg, VA, pp. 66-73, 1987.
71. Halang W. A.: Real-Time Systems: Another Perspective, Real Time Systems: Abstractions, Languages and Desing Methodologies, K. M. Kavi (ed.) IEEE Computer Society Press, 1992.
72. Jaffe M. S., Leveson N. G., Heimdahl M. P. E. and Melhart B. E.: Software requirements analysis for real-time process control systems. IEEE Trans. on Software Engineering, SE-17:241-258, 1991.
73. Jahanian F. and Mok A.K.: Safety analysis of timing properties in real-time systems. IEEE Trans. on Software Engineering, SE-12(9):890-904, September 1986.
74. Jain R.: The Art. Of Computer Systems Performance Analysis, Techniques for Experimental Design, Measurement, Simulation and Modeling. Wiley, New York 1991.
75. Kavi K.M., Buckles B.P., Bhat U.N.: A Formal Definition of Data Flow Graph Model. IEEE Transaction On Computers, Vol. 35, N0. 11, 1996.
76. Kądziela Z.: Analiza metod optymalizacji makrocykli pracy sieci FIP i implementacja wybranych algorytmów w module kreatora opisu sieci dla systemu wizualizacyjnego KRONOS. Praca dyplomowa, Politechnika Śląska, Wydział Automatyki, Elektroniki i Informatyki, Kierunek Informatyka, Gliwice 1998.
77. Klubowe materiały szkoleniowe; 2 tomy WorldFip, Clamart 1995.
78. Kopetz H.: A communication infrastructure for a fault-tolerant distributed real-time systems, Control Eng. Practice, Vol. 3, No. 8, pp. 1139-1146, 1995.
79. Kopetz H.: Ochsenreiter W.: Clock synchronisation in distributed real-time systems, IEEE Transaction on Computers, Vol. 36, No. 8, pp. 933-940, 1987.
80. Kopetz H.: A communication infrastructure for a fault–tolerant distributed real–time system. Control Eng. Practice, Vol. 3, No. 8, 1995.
81. Kwasenowski P., Zygmunt H., Sieńkowski J., Hayduk G., Jachimski M.: Zastosowanie technologii LonWorks w systemach automatyki budynków i monitoringu mediów. Studia Informatica. Volume 22, Number 3, Gliwice 2001.
82. Kwiecień A, Cupek R.: Kryteria stosowania protokołu TCP/IP w sieciach przemysłowych niskiego poziomu. Materiały konferencyjne SCR 2001 AGH, Kraków 2001.
83. Kwiecień A.: Przemysłowe sieciowe systemy rozproszone czasu rzeczywistego. Cechy i wymagania. Studia Informatica Vol.21 No.1. Gliwice 2000.
84. Kwiecień A., Bigewski Z., Cupek R., Fojcik M., Gaj P.: Dokumentacja kontrolerów sieci FIP po rewizji oraz sprawozdanie z uruchomienia i badań. Praca PC-2/ RAu-2/95. Gliwice 1996.

85. Kwiecień A., Bigewski Z., Cupek R., Fojcik M., Gaj P.: Projekt adaptacji oprogramowania narzędziowego z jego ewentualną rozbudową dla potrzeb sieci FIP. Projekt adaptacji oprogramowania komunikacyjnego dla kontrolera sieci FIP. Praca PC-2/ RAu-2/95. Gliwice 1996.
86. Kwiecień A., Bigewski Z., Gaj P., Mrówka Z.: Analiza funkcjonalna układów FULLFIP i projekt modelu karty sieciowej. Praca PC-2/RAu2/95. Gliwice 1995.
87. Kwiecień A., Bigewski Z., Gaj P.: Optymalizacja wymian w sieci FIP. ZN Pol.Śl. s.Informatyka z.32, Gliwice 1997.
88. Kwiecień A., Bigewski Z., Mrówka Z.: Analiza czasu najgorszego przypadku w sieciach przemysłowych.. ZN Pol.Śl. s.Informatyka z.36, Gliwice 1999.
89. Kwiecień A., Fojcik M.: Kryteria integracji sieci przemysłowych najniższego poziomu. Studia Informatica Vol.22 No.3. Gliwice 2001.
90. Kwiecień A., Gaj P., Grzywak A., Mrówka Z.: Rozwiązania sprzętowe i programowe sieci przemysłowej FIP. ZN Pol. Śl. s. Informatyka z. 30, Gliwice 1996.
91. Kwiecień A., Gaj P., Mrówka Z.: O pewnej implementacji interfejsu sieci typu FIP. ZN Pol. Śl. s. Informatyka z. 34, Gliwice 1998.
92. Kwiecień A., Gaj P.: Kryteria doboru protokołów komunikacyjnych w sieciach przemysłowych. Studia Informatica Vol.22 No.3. Gliwice 2001.
93. Kwiecień A., Gaj P.: Sieć FIP, wstęp do analizy czasowej. ZN Pol. Śl. s. Informatyka z. 28, Gliwice 1995.
94. Kwiecień A., Góźdź M.: Moduł programowy umożliwiający diagnostykę i raportowanie pracy segmentu sieci przemysłowej MODBUS dla stacji kontrolno–nadzorczej KRONOS. Archiwum Informatyki, Gliwice 1999.
95. Kwiecień A., Grzywak A., Gaj P.: Dokumentacja funkcjonalna, techniczna i uruchomieniowa karty sieciowej FIP. Praca PC-2/ RAu-2/95. Instytut Informatyki Pol. Śl., Gliwice 1996.
96. Kwiecień A., Grzywak A.: Komputerowy rozproszony system sterowania dla stacji uzdatniania wody elektrowni „TRZEBOWICE"-Czechy. Wydawnictwo Politechniki Gdańskiej Wydziału Elektroniki, Gdańsk 1994.
97. Kwiecień A., Grzywak A.: Możliwość zastosowania sieci komputerowych w kopalniach. Mechanizacja i Automatyzacja Górnictwa, nr 5-6, 1994.
98. Kwiecień A., Grzywak A.: Perspektywy rozwoju zastosowań sieci komputerowych w górnictwie. Mechanizacja i Automatyzacja Górnictwa nr 5, 6/94.
99. Kwiecień A., Grzywak A.: Rozproszone systemy sterowania i zarządzania procesami technologicznymi. Mechanizacja i Automatyzacja Górnictwa nr 8/94.
100. Kwiecień A., Grzywak A.: Sieci komputerowe w systemach sterowania i zarządzania w górnictwie. Konferencja Międzynarodowa ICAMC'95 World Mining Congress.

101. Kwiecień A.: Analiza czasowa sieci przemysłowej SYCOWAY N10 z protokołem TOKEN-BUS. Zeszyty Naukowe Politechniki Śląskiej, s. Informatyka nr 23, Gliwice 1993.
102. Kwiecień A.: Analiza przepływu informacji w komputerowych sieciach przemysłowych. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 1999.
103. Kwiecień A.: Poprawa parametrów pracy sieci przemysłowych z cyklicznymi transakcjami wymiany informacji. Materiały konferencyjne SCR 2000 AGH, Kraków 2000.
104. Kwiecień A.: Sieć rozległa FIP. ZN Pol. Śl. s. Informatyka z. 24, Gliwice 1993.
105. Le reseau de terrain FIP. Materiały firmowe Cegelec/Red.
106. Leterrier P.: The FIP Protocol. Centre de Competence FIP, Nancy 1991.
107. Lista rozkazów maszyn ODRA 1300. Dokumentacja techniczna ELWRO Wrocław.
108. Lueth T., Laengle T., Heinzman J.: Dynamic Tasc Mapping for Real-Time Controller of Distributed Cooperative Robot Systems, Repr. 13th IFAC Workshop on Distributed Computer Control Systems, Toulouse, pp. 37-42, 1995.
109. Materiały firmowe Open MODBUS/TCP Specification. Schneider Electric, 1999.
110. Michalewicz Z.: Algorytmy genetyczne + struktury danych = programy ewolucyjne. WNT, Warszawa 1996.
111. MicroFIP handler. User Reference Manual. Dokumentacja firmy Cegelec. Paryż 1996.
112. MicroFIP. Dokumentacja firmy Cegelec. Paryż 1996.
113. Mok et al. A. K. Synthesis of a real-time message processing system with data-driven timing constraints. In Proc. of IEEE 8th Real-Time Systems Symp., San Jose, CA, pp. 133-143, December 1987.
114. Morzenti A. and Pietro P. S.: Object-oriented logical specification of time-critical systems. ACM Trans. on Software Engineering and Methodology, 3(1): 56-98, January 1994.
115. Navabi Z. Massoumi M.: Investigating simulation of hardware at various level of abstraction and timing back–notation of dataflow description. Simulation No. 5, 1991.
116. Nawrocki J., Czajka A., Complak W.: Scheduling cyclic tasks with binary periods. Information Processing Letters vol 65 nr 4 1998.
117. Nawrocki J., Czajka A.: Binaryzacja okresów zadań cyklicznych. Materiały konferencyjne SCR2000, AGH, Kraków 2000.
118. Nawrocki J.,Czajka A.: Szeregowanie zadań w systemach silnie uwarunkowanych czasowo metodą cyklicznego obciążenia. Zeszyty Naukowe Pol, Śl. seria Automatyka nr 1389, Gliwice 1998.
119. Network Coprocesor. Multiprotocol. Dokumentacja techniczna firmy Cegelec.

120. Netzer R. H. B. and Miller B. P.: Improving the accuracy of data race detection. In Proc. of 3 d ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming, Williamsburg, VA, pp. 133-144, April 1991.
121. Niederliński A.: Systemy komputerowe automatyki przemysłowej t.1 Sprzęt i oprogramowanie. Wydawnictwo Naukowo–Techniczne, Warszawa 1984.
122. Praca zbiorowa Grzywak A., Kwiecień A., i inni: Rozproszone systemy komputerowe. Pronet, Gliwice 1994.
123. Puchol C., Mok A.K.: The Integration of Control and Dataflow Structures in Distributed Hard Real–Time Systems. Proc. Of the Int. Workshop on parallel and Distributed Real–Time Systems, WPDRTS, 1994.
124. Puschner P., Koza Ch.: Calculating the Maximum Execution Time of Real–Time Systems, Vol. 2, 1989.
125. Qin B., Sholl H.A., Ammar R.A.: RTS: A system to simulate the real time cost behaviour of parallel computations, Software – Practice and Experience, Vol. 18, No. 10, 1988.
126. Rahkonen T.: Distributed industrial control systems – a critical review regarding openness. Control Eng. Practice, Vol. 3, No. 8, 1995.
127. Rasid R. F. and Robertson G.G.: Accent: A communication-oriented network operating system kernel. In Proc. 8th ACM Symp. Op. Syst. Princ., ACM Press, New York, pp. 64-75, December 1981.
128. Rick Igou: Working Implementation Agreements for Open Systems Interconnection-Manufacturing Message Specification (MMS), Open Systems Environment Implementers' Workshop, 1994.
129. Rogowski D.: Zastosowanie serwera internetowego i.LON 1000 do połączenia sieci sterowania LonWorks z sieciami bazującymi na protokole IP. Studia Informatica Volume 22, Number 3, Gliwice 2001.
130. Rozległa sieć przemysłowa typu FIP. Materiały dydaktyczne do laboratorium „Sieci przemysłowe", Instytut Informatyki Pol. Śl., Gliwice 1994-1998.
131. Ruitz L., Raja P., Fischer N., Decotigne J.D.: Self Configuration Protocol for a Hard Real–Time Network. Repr. 13th IFAC Workshop on Distributed Computer Control Systems, Toulouse, 1995.
132. Rumbaugh J.: A Data Flow Multiprocessor. IEEE Transaction on Computers, Vol. 26, No. 2, 1977.
133. Rushby J. and Henke F.: Formal verification of algorithms for critical systems. In Proc. of ACM/SIGSOFT 91 Conf. on Software for Critical Systems, New Orleans, pp. 1-15, December 1991.
134. Rushby J. and Henke F.: Formal verification of algorithms for critical systems. IEEE Trans. on Software Engineering, 19(1):13-23, January 1993.

135. Rushby J.M.: Formal Methods and Certification of Critical Systems. SRI International, Menlo Park, Tech. Rap. No. CSL–93–7, 1993.
136. Russel T.: Telecommunications protocols. McGraw Hill 1999.
137. Sancar R., Edwards G.: CSIM Simulation of Teken Ring LAN, Tran. Of The Society for Computer Simulation, Vol. 9 No. 1, pp. 39-58, 1992.
138. Sanchez M., R-Roda I., Poch M., Lafuente J., Cortes U., DAI-DEPUR Archtecture: Distributed Agents for Real-Time WWTP Supervision and Control, Proc. of IFAC/IFIP/IMACS Symposium of Artificial Intelligence in Real-Time Control (AIRTC' 94) pp. 147-152, Valcancia, 1994.
139. Schweitzer P.J.: The periodic loading problem: Formulation and Heuristics. INFOR nr 1. Vol.26, 1988.
140. SCO-UNIX Driver. FIP Device Manager. Dokumentacja firmy Cegelec. Paryż 1996.
141. Shaw A. C.: Communicating real-time state machines. IEEE Trans. on Software Engineering, 18(9): 805-816, September 1992.
142. Shin K.G. HARTS: A distributed real-time architecture. IEEE Computer, 24(5): 25-35, May 1991.
143. Singhal M. and Casavant T. L.: Distributed computing systems. IEEE Computer, 24(8): 12-15, August 1991.
144. Sinha et al. P. K.: The Galaxy distributed operating system. IEEE Computer, 24(8): 34-41, August 1991.
145. Stacja uzdatniania wody dla Elektrowni „TRZEBOWICE". Dokumentacja firmy „PROLOC" Katowice 1993.
146. Stancovic J. A.: Real-Time Computing, BYTE, pp. 155-160, August 1992.
147. Stankovic J. A., Spuri M., Natale M.Di.: Implications of Classical Scheduling Results for Real–Time Systems, IEEE Computer, Vol. 28, No. 6, 1995.
148. Stothert A. G., McLoad I. M.: Medeling and Verifying Timing Properties in Distributed Computer Control Systems, Repr. 13th IFAC Workshop on Distributed Computer Control Systems, Toulouse, pp. 25-30, 1995.
149. Sysło m., Varsingh D.: Algorytmy optymalizacji dyskretnej. PWN, Warszawa 1995.
150. Tokhi M. O., Hosain M. A.: Homogeneous and heterogeneous parallel architectures in Real-Time signal processing and control, Control Eng. Practice, Vol. 3, No. 12, pp. 1675-1686, 1995.
151. Tomasik J.: Symulacyjny model sieci sterowników przemysłowych. ZN Pol.Śl. s.Informatyka z.30, Gliwice 1996.
152. Torngren M.: Fundamentals of Implementing Real-Time Control Applications in Distributed Computer Systems, Journal of Real-Time Systems (in press), 1996.

153. Torngren M., Lind H.: On decentralisation of control functions for distributed real-time motion control, Proc. ISRAM'94 ASME Press, 1994.
154. Torngren M., Wikander J.: A decentralisation methodology for real-time control applications, Control Eng. Practice, Vol. 4, No. 2, pp. 219-228, 1996.
155. Trybicka K.: Opracować i uruchomić program karty FULLFIP dla komputera IBM PC. Praca dyplomowa, Instytut Informatyki Pol. Śl., Gliwice 1996.
156. Tsai J.J.P., Yaodong Bi, Yang S.J.H., Smith R.A.W.: Distributed Real-Time Systems.,JOHN WILEY&SONS,INC 1996.
157. Tsai. J. J.-P., Yang S. and Bi. Y.: Visualization and debugging of distributed real-time software systems. In Proc. of 1992 IEEE Int.'l Conf. on Automation, Robotics, and Computer Vision, pp. Inv-5.6.1-inv-5.6.5, September 1992.
158. Tsai. J. J.-P., Bi. Y.D. and Yang S. J. H. VDS: A system for debugging distributed real-time systems with monitoring support. Int.'l J. Software Engineering and Knowledge Engineering, 1996.
159. Vickers A. J., McDermid J. A.: An Approach to the Design of Software for Distributed Real-Time Systems, Tech. Rap. YCS-93-211, The Univ. of York, Real-Time Systems Research Group, 1993.
160. Weigman J., Kilian G.: Decentralization With Profibus-Dp: Architecture and Fundamentals, Configurations and Use With Simatic S7. John Wiley&Sons 2001-08-09.
161. Węgrzyn S., Vidal P., Gille J.C.: On the structure of developmental systems and the control of theeir development. Systems Analysis, Modeling Simulation, 1994.
162. Węgrzyn S., Vidal P., Gille J.C.: Przyspieszenie realizacji algorytmów w systemach sterowanych przepływem argumentów. ZN Pol. Śl. s. Informatyka z. 26, Gliwice 1994.
163. Węgrzyn S., Vidal P., Gille J.C.: Some models for developemental systems, part XIII. Multilevel development by parallel programming, Int. J. Systems Scl. Vol. 41, no 3, 1993.
164. Węgrzyn S., Vidal P., Gille J.C.: Systems evolutifs commandes par le flux des messages. Bulletin of the Polish Academy of Sciences Technical Sciences, vol. 43, No 2, 1995.
165. Węgrzyn S.: Podstawy automatyki. Państwowe Wydawnictwo Naukowe, Warszawa 1980.
166. Węgrzyn S.: Systemy sterowane przepływem operacji i systemy sterowane przepływem argumentów. ZN Pol. Śl. s. Informatyka z. 24, Gliwice 1993.
167. Wittie L. D.: Computer networks and distributed systems. IEEE Computer, 24(9): 67-76, September 1991.
168. Wolnica W.: Zastosowanie sieci FIP. Praca dyplomowa. Studium Podyplomowe, Instytut Informatyki Pol. Śl., Gliwice 1996.

169. Woźnica D.: Analiza zagadnień związanych ze zdalną konfiguracją modułów sieciowych sieci FIP i praktyczna implementacja wybranego algorytmu konfiguracji dla systemu wizualizacyjnego KRONOS. Praca dyplomowa, Instytut Informatyki Pol. Śl., Gliwice 1997.
170. Yourdon E.: Współczesna analiza strukturalna. Wyd. WNT, Warszawa 1996.

ANALIZA PRZEPŁYWU INFORMACJI W KOMPUTEROWYCH SIECIACH PRZEMYSŁOWYCH

Streszczenie

Praca niniejsza została ujęta w trzynastu rozdziałach.

We wstępie zamieszczono ogólne informacje dotyczące historycznego rozwoju informatycznych systemów sterowania oraz kierunków ich ewolucji. W podrozdziale 1.1 przedstawiono główne cele pracy.

W rozdziale 2 scharakteryzowano przemysłowe systemy rozproszone czasu rzeczywistego. Zwrócono uwagę na ich cechy i wymagania. Określono pewien model, który w dalszej części pracy będzie przedmiotem analizy. W rozdziale 2.6 opisano również cele, które przyświecały realizacji pracy oraz odniesienia do istniejących w literaturze metod analizy pracy sieci komputerowych.

Rozdział 3 - to szczegółowy opis przyjętego modelu do analizy oraz budowa jego poszczególnych elementów.

Rozdziały: 4, 5 i 6 zawierają pełną analizę czasową deterministycznych protokołów dostępu do łącza wraz z opisem ilościowym i jakościowym zjawisk zachodzących w węźle systemu i podają zależności stanowiące podstawę do dokładnych obliczeń parametrów czasowych systemu komunikacyjnego instalacji.

W rozdziale 7 zaprezentowano metody poprawy parametrów czasowych sieci z cyklicznymi wymianami informacji.

Rozdział 8 - to próba nowego podejścia do metod klasyfikacji protokołów wykorzystywanych w sieciach przemysłowych najniższego poziomu, służących ich właściwemu doborowi w funkcji przeznaczenia i zastosowania.

W rozdziale 9 omówiono metody integracji sieci przemysłowych najniższego poziomu i jej wpływ na parametry czasowe systemu komunikacyjnego. Zaprezentowano w nim również przykład integracji dwóch sieci i podano wyniki testów.

W rozdziale 10 omówiono bardzo aktualny problem użycia protokołu TCP/IP w zastosowaniach przemysłowych. Przedstawiono również możliwości integracji sieci za pomocą tego protokołu. Omówiono zasady konfiguracji sieci najniższego poziomu stosując ten protokół z zachowaniem zasady deterministycznego dostępu do medium.

Rozdział 11 - to krótki przegląd firmowych rozwiązań sieci przemysłowych, współcześnie najczęściej stosowanych. Podano w nim parametry techniczne sieci i dokonano wstępnej ich analizy czasowej oraz zaprezentowano sugestie dotyczące kierunków dalszego rozwoju sieci przemysłowych najniższego poziomu.

W rozdziale 12 zamieszczono wnioski końcowe, jak i spostrzeżenia wyniesione z kilku praktycznych aplikacji rezultatów analizy czasowej przemysłowych rozproszonych systemów czasu rzeczywistego.

Rozdział 13 stanowi skorowidz najczęściej używanych określeń i symboli.

Pracę zamyka spis literatury, który zawiera pozycje cytowane i niecytowane. Ma to na celu ułatwienie ewentualnemu Czytelnikowi szybsze odnalezienie odpowiednich i interesujących pozycji literaturowych.

ANALYSIS OF INFORMATION FLOW IN INDUSTRIAL COMPUTER NETWORKS

Summary

The thesis concerns the methodology of analysing data flow in computer networks of the lowest level, i.e. in the networks which are directly connected with industrial processes control and visualization. In majority of cases, industrial IT systems of the lowest level are Distributed Real Time Systems (DRTSs), which necessitates the development of a method of the network operation analysis, so as to make it possible for such systems to be designed properly.

This thesis consists of thirteen chapters.

The Introduction contains general information about the historical development of computer control systems and the directions of their evolution. Subchapter 1.1. presents the main objectives of the thesis.

Chapter 2 characterizes industrial DRTSs. Their features and requirements are pointed out. A certain model is defined which in the following chapters will be the subject of an analysis. In Chapter 2.6., the nobel purposes of the thesis realization are described and references to the methods of a computer network analysis in the literature are made.

Chapter 3 is a detailed description of the model assumed for the analysis and of the structure of its individual components.

Chapters 4, 5 and 6 contain a full time analysis of link access deterministic protocols together with a qualitative and quantitative description of the phenomena which occur in the system node. They also present the dependencies which constitute the basis for exact calculations of time parameters of the installation communications system.

In Chapter 7, improvement methods are presented of the time parameters of networks with cyclic information exchange.

Chapter 8 is an attempt at finding a new approach to the protocols' classification methods used in lowest level industrial networks. The methods serve their proper selection as a function of their application.

Chapter 9 discusses integration methods of industrial networks of the lowest level and the influence of the integration on the time parameters of the communications system. An example of integration of two networks is also presented and test results are given.

In Chapter 10, a very topical issue of TCP/IP protocol use in industrial applications is discussed. Moreover, the possibilities of a network integration by means of this protocol are indicated. The principles are discussed of the lowest level network configuration by using this protocol, with meeting the principle of deterministic access to the medium.

Chapter 11 is a short review of corporate solutions of the industrial networks which are currently most often applied. The networks' technical parameters are provided and an initial time analysis of the networks is performed. Also, suggestions are presented as to the directions of development of the lowest level industrial networks.

Chapter 12 contains final conclusions as well as observations gathered from a few practical applications of the results of the time analysis of industrial DRTSs.

Chapter 13 is a index off most frequently used definitions and symbols.

The thesis ends with a list of literature items which, apart from the entries the thesis refers to, includes some entries not quoted. This is to facilitate a potential reader finding relevant and interesting literature items.

INFORMATION FOR AUTHORS

The journal *STUDIA INFORMATICA* publishes both fundamental and applied Memoirs and Notes in the field of informatics. The Editors' aim is to provide an active forum for disseminating the original results of theoretical research and applications practice of informatics understood as a discipline focused on the investigations of laws that rule processes of coding, storing, processing, and transferring of information or data.

Papers are welcome from fields of informatics inclusive of, but not restricted to *Computer Science, Engineering*, and *Life and Physical Sciences*.

All manuscripts submitted for publication will be subject to critical review. Acceptability will be judged according to the paper's contribution to the art and science of informatics.

In the first instance, all text should be submitted as hardcopy, conventionally mailed, and for accepted paper accompanying with the electronically readable manuscript to:

Dr. Marcin SKOWRONEK
Institute of Informatics
Silesian University of Technology
ul. Akademicka 16
44-100 Gliwice, Poland
Tel.: +48 32 237-12-15
Fax: +48 32 237-27-33
e-mail: marcins@ivp.iinf.polsl.gliwice.pl

MANUSCRIPT REQUIREMENTS

All manuscripts should be written in Polish or in English. Manuscript should be typed on one side paper only, and submitted in duplicate. The name and affiliation of each author should be followed by the title of the paper (as brief as possible). An abstract of not more than 50 words is required. The text should be logically divided under numbered headings and subheadings (up to four levels). Each table must have a title and should be cited in the text. Each figure should have a caption and have to be cited in the text. References should be cited with a number in square brackets that corresponds to a proper number in the reference list. The accuracy of the references is the author's responsibility. Abbreviations should be used sparingly and given in full at first mention (e.g. "Central Processing Unit (CPU)"). In case when the manuscript is provided in Polish (English) language, the summary and additional abstract (up to 300 words with reference to the equations, tables and figures) in English (Polish) should be added.

After the paper has been reviewed and accepted for publication, the author has to submit to the Editor a hardcopy and electronic version of the manuscript.

It is strongly recommended to submit the manuscript in a form downloadable from web site http://sun10.iinf.polsl.gliwice.pl/publikacje.html.

To subscribe: *STUDIA INFORMATICA* (PL ISSN 0208-7286) is published by Silesian University of Technology Press (Wydawnictwo Politechniki Śląskiej) ul. Akademicka 5, 44-100 Gliwice, Poland, Tel./Fax +48 32 237-13-81. 2002 subscription price: US$30 per number.

Wydano za zgodą
Rektora Politechniki Śląskiej

Wydawnictwo Politechniki Śląskiej
ul. Akademicka 5, 44-100 Gliwice
tel./fax (0 prefiks 32) 237-13-81
Dział Sprzedaży i Reklamy (0 prefiks 32) 237-18-48

www.polsl.gliwice.pl/alma.mater/wps.html
wydawnictwo@polsl.gliwice.pl

Nakł. 100 + 50 Ark. wyd. 16 Ark. druk. 14,375 Papier offset. 70x100, 80 g.
Oddano i podpisano do druku 27. 03. 02 r. Druk ukończono w kwietniu 2002 r.
Zam. 97/02

Fotokopie, druk i oprawę
wykonano w Zakładzie Graficznym Politechniki Śląskiej w Gliwicach, ul. Kujawska 1

INSTYTUT INFORMATYKI prowadzi:

- Dzienne Studia Magisterskie (5-letnie)
- Wieczorowe Studia Inżynierskie (4,5-letnie)
- Uzupełniające Studia Magisterskie (2-letnie)
- Studia Doktoranckie (zaoczne, 4-letnie)
- Studium Podyplomowe z zakresu *Sieci komputerowe, systemy mikrokomputerowe, bazy danych* (2 semestry)
- Studium Podyplomowe z zakresu *Systemy informatyczne* (2 semestry)

Informacje:

POLITECHNIKA ŚLĄSKA
Instytut Informatyki

ul. Akademicka 16, 44-100 Gliwice

tel. (032) 237 24 05; 237 21 51; 237 27 33
tel./fax (032) 237 27 33

e-mail: sekretariat@top.iinf.polsl.gliwice.pl

http://www.iinf.polsl.gliwice.pl (dydaktyka)