WYZNACZANIE BEZKOLIZYJNYCH DRÓG PRZESYŁANIA DANYCH W SIECI TELEINFORMATYCZNEJ O STRUKTURZE TYPU HIPERSZEŚCIANU

Jan CHUDZIKIEWICZ, Krzysztof MURAWSKI

Wojskowa Akademia Techniczna, Wydział Cybernetyki, Instytut Teleinformatyki i Automatyki,
ul. Kaliskiego 2, 00-908 Warszawa; e-mail: j.chudzikiewicz@iar.wat.waw.pl, k.murawski@ita.wat.edu.pl

Streszczenie

W referacie zaprezentowano metodę oraz bazujący na tej metodzie algorytm wyznaczania bezkolizyjnych ścieżek przepływu danych w sieci o strukturze hipersześcianu. Sieci teleinformatyczne o strukturze logicznej hipersześcianu mają możliwość adaptowania (rekonfigurowania) struktury logicznej sieci, do zaistniałych awarii lub wymaganych warunków samodiagnozowania się sieci. Należą one do klasy systemów tolerujących. Przedstawiono również, bazując na systemie operacyjnym klasy Windows®, narzędzia i mechanizmy wbudowane w system, które ułatwią sposób implementacji opracowanego algorytmu.

Słowa kluczowe: hipersześcian, samodiagnozowanie, systemy tolerujące błędy.

DETERMINING A NON COLLISION DATA TRANSFER PATH IN HYPERCUBE TELECOMUNICATION NETWORK

Summary

In this paper author presents the method and the algorithm for determining a non collision data transfer path in hypercube computer network. The hypercube structures have properties of auto-reconfiguration of network structure depending on failures or on requiring conditions for auto-reconfiguration. Hypercube computer networks belong to the class of fault tolerant computer networks. More over, for Windows® operating systems, the tools and mechanism are presented which to makes implementation of the algorithm easier.

Keywords: hypercube, auto-diagnosis, faulty tolerance systems.

1. WPROWADZENIE

W systemach wielokomputerowych (wieloprocesorowych) zapewnienie odpowiedniej szybkości komunikacji, jest krytyczne z punktu widzenia efektywności ich działania. Nie mniej istotnym jest możliwość rekonfiguracji struktury takiego systemu do zaistniałych awarii tak, aby graf opisujący powstałą strukturę był spójny. Przykładem struktury spełniającej te wymagania, jest struktura hipersześcianu [9], [3]. Sieci o strukturze logicznej n-wymiarowego hipersześcianu należą do klasy systemów tolerujących błędy i charakteryzują się dużą złożonością dla $n > 3$. Sposób identyfikacji zaistniałych awarii zależny jest od zastosowanej metody diagnozowania [9], [5], [4], [3].

Problem przesyłania danych w systemach o strukturze hipersześcianu jest szeroko analizowany w literaturze przedmiotu. Między innymi Gordon i Stout przedstawiają metodę nazwaną przez nich „sidetracking" [7]. Metoda ta zakłada, że każdy z węzłów przechowuje informacje o stanie niezawodnościowym swoich sąsiadów. Informacja jest przesyłana przez losowo wybraną ścieżkę, która sąsiaduje z węzłem będącym w stanie zdatności. W przypadku, gdy nie ma ścieżek sąsiadujących ze zdatnymi węzłami, informacja jest blokowana i przesyłana jest z powrotem do węzła, z którego pierwotnie była wysłana. Wadą tej metody jest małe prawdopodobieństwo przesłania informacji dla określonej liczby niezdatnych węzłów w systemie oraz duże opóźnienie czasowe. Inna metoda zaproponowana przez Chena nosi nazwę „backtracking" [1]. Metoda ta zakłada konieczność przechowywania w przesyłanych danych informacji o kolejnych węzłach, które pośredniczyły w jej przesyłaniu. W przypadku, gdy dane dotrą do węzła, który sąsiaduje z węzłami niezdatnymi, informacja ta jest wykorzystywana do zwrotnego przesłania danych do węzła wcześniejszego. Wadą tego rozwiązania jest wprowadzanie do przesyłanych danych nadmiarowej informacji oraz duże opóźnienia czasowe.

W niniejszym referacie przedstawiono metodę przesyłania danych w systemie o strukturze hipersześcianu, bazującą na wyznaczeniu pokrycia grafu opisującego daną strukturę. Metoda zakłada, że każdy z węzłów przechowuje informacje o stanie niezawodnościowym całego systemu. Ponadto w artykule zaprezentowano opracowany na

potrzeby implementacji algorytmu w systemie Windows® sterownik pakietowy oraz omówiono mechanizmy wbudowane w system, które mogą w znacznym stopniu ułatwić realizację postawionego zadania.

2. POJĘCIA PODSTAWOWE

Niech Z^n oznacza zbiór *n*-wymiarowych wektorów binarnych.
Oznaczmy:

$$(s_1,\ldots,s_n)=\{z\in Z^n:((s_i\neq x)\Rightarrow$$
$$\Rightarrow(z_i=s_i))\wedge((s_i=x)\Rightarrow$$
$$\Rightarrow(z_i\in\{0,\ 1\})\ \ (s_i\in\{0,\ 1,\ x\},\ 1\le i\le n),$$

gdzie *x* oznacza wartość nieokreśloną (0 lub 1).

$Z(s)$ -zbiór podsześcianów 0-wymiarowych (zbiór wektorów $z=(z_1,...,z_n)$, $(z_i\in\{0,1\}$, $1\le i\le n)$ podsześcianu s $(s\in S^n))$.

Określenie 1. *n*-wymiarowym hipersześcianem binarnym nazywamy graf zwykły G' $(G'=$ $=<E,\ U'>,|E|=2^n,|U'|=n\cdot 2^{n-1})$ o 2^n węzłach, z których każdy opisany jest odpowiednim wektorem binarnym z $(z=(z_1,\ldots,z_n)$, $z_i\in\{0,\ 1\},\ 1\le i\le n,\ z\in Z^n,|Z^n|=2^n)$ oraz o $n\cdot 2^{n-1}$ krawędziach, łączących te węzły, których opisujące je wektory odległe są o 1 według miary Hamminga.

Strukturę *n*-wymiarowego hipersześcianu binarnego będziemy dalej oznaczać przez H^n, a graf częściowy tej struktury przez H_t^n. Indeks *t* oznacza liczbę krawędzi grafu struktury H_t^n.

Dalej węzły grafu H^n będą reprezentować komputery (procesory), a jego krawędzie - linie transmisji danych między tymi komputerami (procesorami), które są incydentne z określoną krawędzią.

Określenie 2. Łańcuchem τ o długości k $(0\le k\le 2^n)$ w H^n nazywamy spójny podgraf grafu H^n, zawierający k + 1 węzłów, z których tylko dwa są stopnia pierwszego.

Węzeł stopnia pierwszego łańcucha nazywamy biegunem tego łańcucha.

Niech $Z(\tau)$ oraz $B(\tau)$ $(B(\tau)\subseteq Z(\tau))$ oznaczają odpowiednio zbiór węzłów oraz biegunów łańcucha τ.

Łańcuch τ będziemy przedstawiać zarówno w postaci podgrafu $<Z(\tau)>H^n$ jak i w postaci zbioru $S(\tau)$ podsześcianów 1-wymiarowych s $(s\in S_1^n)$ takiego, że:

$$[s\in S(\tau)]\Leftrightarrow[\exists\ \ z',\ z''\in Z(\tau):z'+z''=s].$$

Określenie 3. Mówimy, że łańcuchy τ' i τ'' w H^n są silnie wzajemnie niezależne, jeżeli $Z(\tau')\cap Z(\tau'')=\varnothing$.

Określenie 4. Mówimy, że zbiór P $(P=\{\tau_1,\ldots,\tau_p\},\ 1\le p\le 2^{n-1})$ silnie wzajemnie niezależnych łańcuchów $\tau_1,\ldots\tau_p$, jest p o k r y c i e m H^n, jeżeli $\{Z(\tau_i):1\le i\le p\}=Z^n$.

Określenie 5. Mówimy, że pokrycie P $(P\in\boldsymbol{P}_n)$ jest pokryciem klasy $\boldsymbol{K}(\lambda_1^{\delta_1},\ldots,\lambda_a^{\delta_a})$ $(\boldsymbol{K}(\lambda_1^{\delta_1},\ldots,\lambda_a^{\delta_a})\in\boldsymbol{K}_n)$ jeżeli spełnione są następujące warunki:

$$(|S(\tau_i)|=\lambda_i)\wedge(\delta(b'(\tau_i),\ b''(\tau_i))=$$
$$=\delta_i,\ \tau_i\in P,\ b'(\tau_i),\ b''(\tau_i)\in B(P),\ \lambda_i\in\lambda,$$
$$i=\{1,\ldots,|P|\}$$

gdzie:

K_n - zbiór klas hipersześcianu H^n,

$\boldsymbol{P}_n$ - zbiór pokryć H^n,

$\boldsymbol{\Lambda}^a(b)$ $(2\le a\le b)$ - zbiór addytywnych *a*-członowych podziałów $\lambda=(\lambda_1,\ldots,\lambda_a)$ $(\lambda\in\boldsymbol{\Lambda}^a(b),\ \lambda_i>0,\ 1\le i\le a,\ \lambda_j\ge\lambda_{j+1},$ $1\le j\le a-1)$ liczby wierzchołków b hipersześcianu H^n,

Odległość Hamminga między dwoma wektorami binarnymi $b'(\tau_i)$ i $b''(\tau_i)$, będącymi biegunami łańcucha τ_i, spełnia zależność:

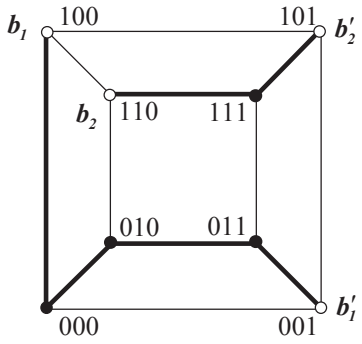
$$\delta(b'(\tau_i),\ b''(\tau_i))=\sum_{k\in\{1,\ldots,n\}}(b'(\tau_i)_k\oplus b''(\tau_i)_k),$$

gdzie: $b(\tau_i)_k$ - *k*-ta składowa wektora binarnego $b(\tau_i)$,

$\oplus$ - suma modulo 2.

3. METODA WYZNACZANIA BEZKOLIZYJNYCH ŚCIEŻEK PRZEPŁYWU DANYCH W SYSTEMIE O STRUKTURZE HIPERSZEŚCIANU

Metoda wyznaczania bezkolizyjnych ścieżek polega na wyznaczeniu zbioru możliwych pokryć struktury H^n silnie wzajemnie niezależnymi łańcuchami prostymi. Efektem końcowym działania metody jest wyznaczenie wszystkich ścieżek pomiędzy elementami, które w danym momencie zamierzają realizować wymianę danych tak, aby nawzajem nie zakłócały one transmisji. Dla zobrazowania działania metody rozpatrzmy przykład wyznaczenia ścieżek pomiędzy dwiema parami elementów w strukturze H^3. Postać jednego z możliwych pokryć dwoma łańcuchami struktury H^3 pokazano na rys. 1.



Rys. 1. Przykład pokrycia $|P| = 2$ struktury H^3

W przedstawionym przykładzie zostało wyznaczone pokrycie dwoma łańcuchami prostymi grafu struktury H^3, umożliwiające bezkolizyjną wymianę danych pomiędzy węzłami $(b_1,\ b_1')$ i $(b_2,\ b_2')$, dla których $\delta(b_1(\tau_1),\ b_1'(\tau_1)) = 2$, $|S(\tau_1)| = 4$ oraz $\delta(b_2(\tau_2),\ b_2'(\tau_2)) = 2$, $|S(\tau_2)| = 2$. Wyznaczone pokrycie należy do klasy $\boldsymbol{K}(4^2, 4^2)$. Dla przykładowej struktury metoda pozwala na wyznaczenie wszystkich 9 możliwych klas pokryć: $\boldsymbol{K}(5^1,1^1);\boldsymbol{K}(5^3,1^1);\boldsymbol{K}(4^2,4^2);\boldsymbol{K}(3^1,3^1);$ $\boldsymbol{K}(3^3,3^3);\boldsymbol{K}(3^1,1^1,1^1);\boldsymbol{K}(3^2,1^1,1^1);\boldsymbol{K}(2^2,2^2,1^1);$ $\boldsymbol{K}(1^1,1^1,1^1,1^1)$.

4. ALGORYTM WYZNACZANIA BEZKOLIZYJNYCH ŚCIEŻEK PRZEPŁYWU DANYCH W SYSTEMIE O STRUKTURZE HIPERSZEŚCIANU

Wyznaczenie klas pokryć jest zadaniem złożonym, szczególnie dla $n \geq 4$, dlatego też należy w tym celu wspomóc się komputerem. Poniżej przedstawiono algorytm wyznaczania pokryć.

Oznaczenia:

$Ł(z',\ z'')$ - zbiór łańcuchów łączących węzły z' oraz z'',

$Z(\tau_j)$ - zbiór węzłów tworzących łańcuch τ_j,

$\boldsymbol{P}$ - aktualnie wyznaczane pokrycie.

Algorytm rozpoczynamy od $i = 1$, czyli pierwszego łańcucha pokrycia.

Krok 1.

Wybierz jako biegun początkowy łańcucha τ_i nie wybrany węzeł $z' \in Z^n \setminus Z(\tau_j),\ \tau_j \in P,$ $j = \{1,\ldots,i-1\},\ i > 1$ o najmniejszej etykiecie. Określ zbiór Z_i, $(Z_i = \{z'' \in Z^n \setminus Z(\tau_j) \cup z' : (\delta(z',\ z'') = \delta_i)\},\ \tau_j \in P,\ j = \{1,\ldots,i-1\}, i > 1)$ biegunów końcowych łańcucha τ_i.

Krok 2.

Wyznacz zbiór $Ł(z',\ z'')$ $(Ł(z',\ z'') = \{\tau : |S(\tau)| = \lambda_i\},\ \forall\ z'' \in Z_i)$ łańcuchów łączących węzły z' oraz z'' tak, aby: $(\forall\ \tau_j \in P : Z(\tau_j) \cap Z(\tau) = \varnothing, j = \{1,\ldots,i-1\},$ $i > 2)$.

Krok 3.

Pobierz łańcuch τ ze zbioru łańcuchów dostępnych $Ł(z',\ z'')$ i dodaj go do pokrycia $\boldsymbol{P}$. $Ł(z',\ z'') = Ł(z',\ z'') \setminus \tau$.

Jeżeli $i = a$, to przejdź do kroku 4 w przeciwnym przypadku $i = i + 1$ i przejdź do kroku 1.

Krok 4.

Sprawdź czy nie wyznaczono wcześniej pokrycia odpowiadającego pokryciu $\boldsymbol{P}$.

$$\boldsymbol{P} \in \boldsymbol{K}(\lambda_1^{\delta_1},\ldots,\lambda_a^{\delta_a}) \Leftrightarrow \neg\exists \boldsymbol{P}' \in \boldsymbol{K}(\lambda_1^{\delta_1},\ldots,\lambda_a^{\delta_a}) : (B(\tau_l) \neq B(\tau_k)) \wedge (Z(\tau_l) \neq Z(\tau_k)),$$
$$\tau_l \in P',\ \tau_k \in P,\ l,k = \{1,\ldots,a\}.$$

Jeżeli nie wyznaczono wcześniej pokrycia odpowiadającego pokryciu $\boldsymbol{P}$, to dodaj $\boldsymbol{P}$ do zbioru $\boldsymbol{P}_n$. W przeciwnym przypadku odrzuć $\boldsymbol{P}$. Przejdź do kroku 5.

Krok 5.

Czy $Ł(z',\ z'') = \varnothing$. Jeżeli nie to przejdź do kroku 3 w przeciwnym przypadku, gdy $i > 1$, $i = i - 1$ powtórz krok 5. Jeżeli i=1 sprawdź czy węzeł, który nie był jeszcze biegunem ma etykietę mniejszą od 2^n. Jeżeli TAK to przejdź do kroku 1. Jeżeli NIE to przejdź do kroku 6.

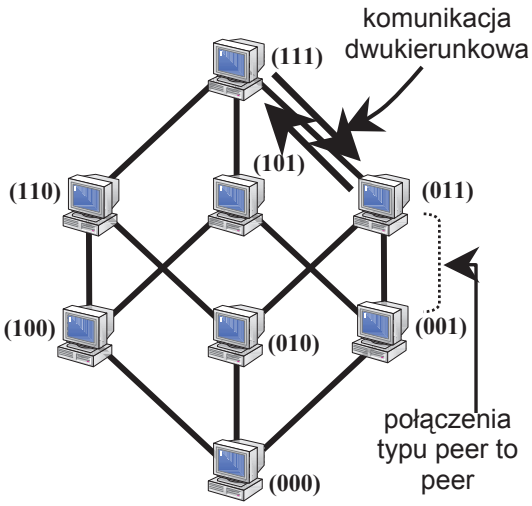
Krok 6.

Koniec działania algorytmu.

5. IMPLEMENTACJA W SYSTEMIE WINDOWS®

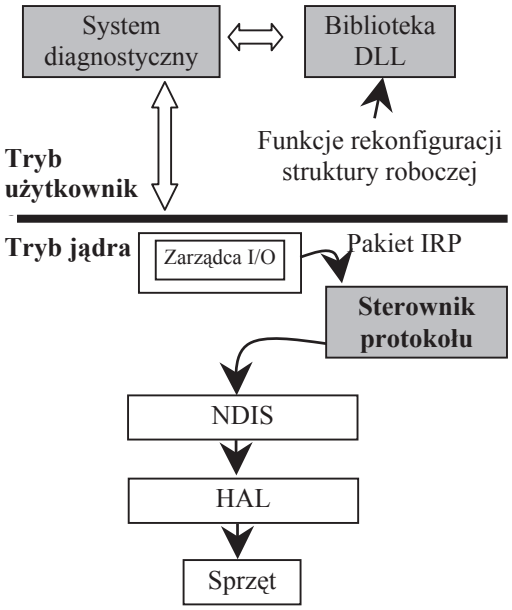
Sieć komputerowa o strukturze hipersześcianu jest zaliczana do sieci typu punkt-punkt (rys. 2), co powoduje, że komunikacja pomiędzy komputerami, których odległość Hamminga jest większa niż 1, wymaga pośrednictwa innych komputerów.

Rozwiązanie takie wymaga zastosowania komputerów wyposażonych w *n*-interfejsów sieciowych, gdzie *n* jest wymiarem struktury hipersześcianu. Aby zapewnić możliwość komunikacji z wykorzystaniem wszystkich interfejsów, należało opracować odpowiedni sterownik i włączyć go w architekturę sieciową systemu Windows® (rys. 3).



Rys. 2. Rozwiązanie komunikacji w sieci komputerowej o strukturze hipersześcianu

Architektura systemu Windows zakłada wykorzystanie dwóch poziomów uprzywilejowania, jak przedstawiono to na rys. 3: trybu użytkownika oraz trybu jądra [6], [10].



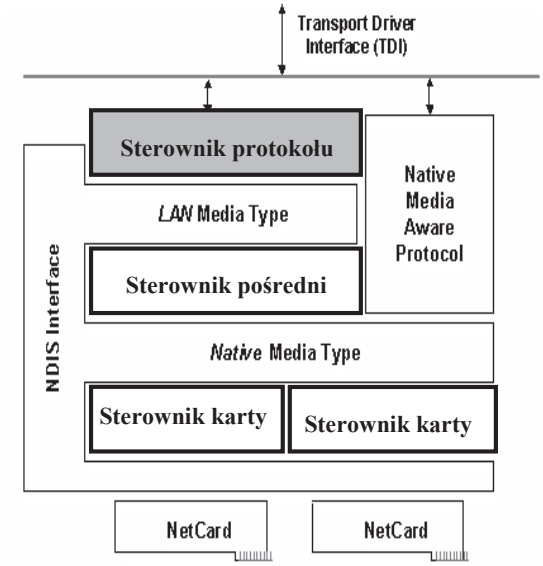
Rys. 3. Architektura systemu Windows klasy NT

W trybie użytkownika uruchamiane są aplikacje użytkowe. W naszym przypadku jest to system diagnostyczny nadzorujący poprawność działania całej struktury.

Specyfikacja interfejsu sterowników sieci (ang. *NDIS - Network Driver Interface Specification*) jest zrealizowana w Windows® jako biblioteka, która definiuje interfejsy między sterownikami poszczególnych warstw, dzięki czemu oddziela sterowniki sprzętowe (niskiego poziomu) od sterowników wyższych warstw takich jak warstwa transportowa. NDIS przechowuje również informacje na temat stanu i parametrów sterowników sieciowych w tym wskaźników do funkcji, uchwytów i innych wartości.

NDIS wyróżnia następujące typy sterowników (patrz rys. 4):

- sterownik miniportu (*ang.Miniport driver*);
- sterownik pośredni (ang. Intermediate driver);
- sterownik protokołu (*ang. Protocol driver*).



Rys. 4. Typy sterowników NDIS

Sterownik miniportu (sterownik karty sieciowej) ma dwie podstawowe funkcje:

- zarządzanie kartą sieciową w tym wysyłanie i odbieranie danych;
- współpraca ze sterownikami wyższego poziomu, takimi jak sterowniki pośrednie czy sterowniki protokołu.

Sterownik miniportu komunikuje się z kartą sieciową i sterownikami wyższego poziomu poprzez funkcje z biblioteki NDIS. Biblioteka ta eksportuje zbiór funkcji (funkcje NdisXXX), który zawiera wszystkie funkcje systemu operacyjnego, wymagane przez sterownik karty sieciowej. Niemniej istotnym jest to, że sterownik karty sieciowej powinien udostępniać oddzielny zbiór funkcji przeznaczony dla biblioteki NDIS (funkcje Miniport*Xxx)*.

Współpraca sterownika miniportu z biblioteką NDIS i sterownikiem wyższego poziomu realizowana jest zgodnie z następującymi zasadami:

- Wysłanie pakietu przez sterownik protokołu powoduje wywołanie funkcji NdisXXX wyeksportowanej przez bibliotekę NDIS. NDIS przekazuje otrzymany pakiet do sterownika miniportu wywołując odpowiednią funkcję MiniportXXX wyeksportowaną przez sterownik miniportu. Sterownik miniportu, będący ostatnim elementem na drodze przesyłania pakietu, przekazuje go do karty sieciowej. Operacja przekazania pakietu pomiędzy

sterownikiem miniportu a kartą sieciową, realizowana jest poprzez wywołanie odpowiedniej funkcji NdisXXX przez sterownik.

- Odebranie przez kartę sieciową pakietu do niej zaadresowanego powoduje wysłanie komunikatu o tym zdarzeniu do sterownika miniportu. Sterownik miniportu korzystając z funkcji biblioteki NDIS wymusza przesłanie odebranego pakietu z karty sieciowej do pamięci podręcznej sterownika miniportu. Kolejną czynnością jest wywołanie przez sterownik miniportu odpowiedniej funkcji NdisXXX, dzięki której sterownik wyższego poziomu zostanie poinformowany o otrzymanym pakiecie.

Sterownik pośredni stosowany jest w hierarchii pomiędzy sterownikiem miniportu a sterownikiem protokołu, dlatego komunikować musi się z dwoma sterownikami:

- Od strony niższej warstwy sterownik pośredni świadczy usługi sterownika protokołu poprzez wyeksportowanie funkcji ProtocolXXX. Funkcje te używane są przez NDIS do komunikacji, gdy sterownik miniportu przesyła informacje w górę stosu protokołów. Dla sterownika miniportu sterownik pośredni wygląda jak sterownik protokołu.
- Od strony wyższej warstwy sterownik pośredni świadczy usługi sterownika miniportu poprzez wyeksportowanie funkcji MiniportXXX, których używają sterowniki protokołu do komunikowania się z kartą sieciową. Dla sterownika protokołu sterownik pośredni wygląda jak sterownik miniportu.

Pomimo tego, że sterownik pośredni eksportuje zestaw funkcji MiniportXXX dla wyższych sterowników stosu sieciowego, nie zarządza on fizycznie kartą sieciową, tworzy zaś wirtualne karty sieciowe, do których może zostać przyłączony sterownik protokołu. Gdy sterownik protokołu wysyła pakiet do sterownika pośredniego ten przesyła go do sterownika miniportu. W przeciwnym kierunku sterownik miniportu przesyła dane otrzymane z karty sieciowej do sterownika pośredniego, który następnie przesyła je do sterownika protokołu.

Sterowniki pośrednie używane są do:

- Translacji pakietów pomiędzy różnymi rodzajami mediów transmisyjnych. Przykładowo sterownik pośredni pomiędzy warstwą transportową Ethernet a sterownikiem miniportu ATM, przetwarza pakiety Ethernet na pakiety ATM i odwrotnie.
- Filtrowania pakietów. Przykładem takiego wykorzystania jest harmonogram pakietów. Moduł ten sprawdza priorytet pakietu wysłanego w dół stosu przez sterownik protokołu, lub w górę przez sterownik miniportu i ustala kolejność przesyłania pakietów ze względu na wartość priorytetu.
- Balansowania transmisją pakietów między większą ilością kart sieciowych. Dla sterownika protokołu udostępnia on pojedynczy wirtualny adapter, jednak ruch sieciowy kieruje na kilka kart sieciowych zwiększając prędkość przesyłania danych.

Sterownik protokołu to najwyżej położony w stosie NDIS sterownik i jest zarazem najniżej usytuowanym komponentem z zaimplementowanym protokołem sieciowym. Przykładem takiego sterownika jest sterownik TCP/IP lub IPX/SPX. Sterownik protokołu alokuje dla pakietu odpowiedni obszar w pamięci, kopiuje dane z aplikacji do przygotowanego pakietu i poprzez wywołanie funkcji NDIS przesyła go do karty sieciowej. Tworzy również interfejs dla przychodzących z karty sieciowej danych i przesyła je do aplikacji.

6. PODSUMOWANIE

Przedstawiony w referacie algorytm, pozwala na wyznaczenie możliwych, bezkolizyjnych ścieżek przesyłania zarówno komunikatów diagnostycznych informujących o stanie niezawodnościowym systemu, jak również testów oraz przetwarzanych danych. Algorytm bazuje na strukturze diagnostycznej systemu i umożliwia wybranie ścieżki, która nie będzie uwzględniała innych elementów realizujących w danym momencie transmisji oraz tych, które znajdują się w stanie niezdatności. Przyjęte założenie można rozszerzyć o elementy realizujące w danym momencie proces przetwarzania danych, ale wymaga to przyjęcia dodatkowych, nie uwzględnionych w aktualnej wersji algorytmu, założeń.

W chwili obecnej prowadzone są pracę nad budową sieci eksperymentalnej, wykorzystującej różne typy interfejsów sieciowych, między innymi połączenia kablowe (karty Ethernet, standard USB) oraz bezprzewodowe (Wi-Fi, Bluetooth). Wykonanie takiej sieci pozwoli na praktyczną weryfikację założeń teoretycznych opracowanego algorytmu oraz sprawdzenie możliwości jego adaptacji, jeżeli sieć podlegać będzie procesowi „*łagodnej degradacji*".

LITERATURA

[1] Chen M.-S., Shin K. G.: *Depth-first search approach for faulttolerant routing in hypercube multicomputers*, IEEE Transactions Parallel and Distributed Systems, vol. 1, no. 2, str. 152-159, Apr. 1990.
[2] Chiu G. M., Wu S. P.: *A Fault-Tolerant Routing Strategy in Hypercube Multicomputers*, IEEE Transactions on Computers, vol. 45, no. 2, str 143-155, february 1996.
[3] Chudzikiewicz J.: *Sieci komputerowe o strukturze logicznej typu hipersześcianu;* Instytut Automatyki i Robotyki, Wydziału Cybernetyki WAT, Warszawa 2002.
[4] Chudzikiewicz J.: *Metoda wyznaczania m-optymalnych struktur opiniowania diagnostycznego dla sieci komputerowych typu hipersześcianu*, V Krajowa Konferencja Diagnostyka Urządzeń i Systemów, Ustroń 2003.
[5] Chudzikiewicz J.: *Wyznaczanie m-diagnozowalnych struktur typu PMC w systemach o zwiększonej odporności na uszkodzenia*, X Konferencja Systemów Czasu Rzeczywistego, Ustroń 2003.
[6] Dokumentacja: *Microsoft Windows 2003 Driver Development Kit*, Microsoft 2003.
[7] Gordon J. M., Stout Q. F.: *Hypercube message routing in the presence of faults*, Proc. Third Conf. on Hypercube Concurrent Computers and Applicafions, vol. 1, str. 318-327, Jan. 1988.
[8] Korzan B.: *Elementy teorii grafów i sieci. Metody i zastosowania*, Wydawnictwo Naukowo-Techniczne, Warszawa, 1978.
[9] Kulesza R.: *Podstawy diagnostyki sieci logicznych i komputerowych*, Instytut Automatyki i Robotyki, Wydział Cybernetyki WAT, Wydanie II, Warszawa 2000.
[10] Oney W.: *Programming the Microsoft Windows Driver Model*, wydanie drugie, Microsoft Press 2003.

Jan CHUDZIKIEWICZ ukończył studia w 1993 na Wydziale Cybernetyki Wojskowej Akademii Technicznej uzyskując tytuł mgr. inż. w specjalności systemy komputerowe. W roku 2001 na tymże samym wydziale obronił pracę doktorską uzyskując tytuł dr inż. w specjalności diagnozowanie sieci komputerowych. Przez cały okres swojej pracy zawodowej związany z Wojskową Akademią Techniczną. W latach 1994-1998 współpracował z Przemysłowym Instytutem Elektroniki w zakresie projektowania systemów diagnostycznych dla układów cyfrowych. Obecnie jego zainteresowania koncentrują się wokół metod diagnozowania systemów komputerowych, sieci komputerowych i systemów tolerujący błędy.

Krzysztof MURAWSKI ukończył w 1997 studia na Wydziale Cybernetyki Wojskowej Akademii Technicznej uzyskując tytuł mgr inż. W specjalności systemy komputerowe. W roku 2002 na tymże samym wydziale obronił pracę doktorską uzyskując tytuł dr inż. w specjalności inżynieria systemów. Przez cały okres swojej pracy zawodowej związany z Wojskową Akademią Techniczną. W latach 2000-2001 odbył staż naukowy w George Mason University. Obecnie jego zainteresowania skupiają się wokół budowy i projektowania systemów komputerowych.