STUDIA INFORMATICA 2011
Volume 32 Number 3A (98)

Krzysztof NOWICKI, Adrian OSTROWSKI,
Aurelia POŹNIAK, Łukasz WRZESIŃSKI
Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

WYKORZYSTANIE SPRZĘTU KOMPUTEROWEGO KLASY SOHO DO MODELOWANIA ZŁOŻONYCH ROZWIĄZAŃ SIECIOWYCH[1]

Streszczenie. W artykule dokonano oceny możliwości modelowania złożonych rozwiązań sieciowych, za pomocą modyfikacji otwartoźródłowego sterownika tanich kart ethernetowych. Zaproponowano wykorzystanie koncepcji interfejsów logicznych, znakowania ramek z użyciem pola Tag standardu IEEE 802.1Q – VLAN oraz wielu pierścieni DMA do priorytetyzowania klas ruchu. Techniki te, w połączeniu z implementacją w sterowniku odpowiednich algorytmów, umożliwiają realizację wielu mechanizmów sieciowych, np. autokonfiguracji lub niezawodności. Przedstawiono ograniczenia tak realizowanych sieci. Zaprezentowano realizację przykładowej, dwupierścieniowej sieci komputerowej, wspierającej mechanizmy niezawodności z zaimplementowaną obsługą różnych klas ruchu, automatyczną konfigurację stacji i odtwarzaniem konfiguracji po awarii. Wykazano, że wydajność powstałej sieci w komunikacji między węzłami sąsiednimi nie ustępuje oryginalnej sieci ethernetowej.

Słowa kluczowe: sieć, pierścień, symulator sieci, Ethernet, RPR

MODELLING COMPLEX NETWORK SOLUTIONS USING SOHO CLASS COMPUTER HARDWARE

Summary. Article provides evaluation of capabilities to build complex networks, by modifying an Open Source driver of cheap Ethernet NICs. Methods of using logical interfaces and marking frames with IEEE 802.1Q VLAN Tags are described. These methods combined with implementation of adequate algorithms in driver module provides the use of autoconfiguration and reliability techniques. Constraints of such solution are presented. Advanced computer network, providing reliability and fairness was built. It provides different traffic classes, automatic configuration of the stations and recovering from failures. In created network, efficiency of communication between neighbors matches up that of original Ethernet network.

Keywords: network, ring, network simulation, Ethernet, RPR

[1] Praca częściowo zrealizowana w ramach projektów badawczych N519 581038 oraz PBZ 02/II/2007 MNiSW

1. Możliwości zastosowania kart ethernetowych do symulacji zaawansowanych sieci komputerowych

Współczesny sprzęt ethernetowy przeznaczony do zastosowań SOHO (ang. *Small Office/Home Office*) może być tanim i elastycznym narzędziem do testowania i badania nowoczesnych rozwiązań sieciowych. Tania karta ethernetowa zawiera zwykle elektronikę realizującą funkcje warstwy fizycznej modelu OSI. Warstwa łącza danych jest zwykle w dużej mierze realizowana programowo przez sterownik tego urządzenia, umożliwiając jednak przeniesienie części zadań na sprzęt (np. liczenie sum kontrolnych), co pozwala dowolnie modyfikować zachowanie tej warstwy, poprzez napisanie odpowiedniego sterownika. Ta cecha, polegająca na przenoszeniu funkcjonalności sprzętu do oprogramowania, jest od bardzo dawna widoczna w urządzeniach komputerowych z niższej półki, szczególnie w kartach sieciowych, modemach oraz układach audio. W przypadku gdy takie urządzenie ma otwarty sterownik programowy, możliwe jest jego dowolne modyfikowanie, zmieniające zachowanie sprzętu. Podobnie jak realizowano już oscyloskopy czy generatory sygnału za pomocą kart dźwiękowych [1, 2], można wykorzystać karty sieciowe do innych celów niż te, do których zostały przeznaczone. Szczególnie obiecująca jest możliwość użycia kart ethernetowych do badania rozwiązań sieciowych zupełnie niezwiązanych z Ethernetem.

Oczywiście technika ta ma pewne ograniczenia. Zwykle nie mamy wpływu na zachowanie mechanizmów realizowanych sprzętowo lub mamy bardzo ograniczone możliwości ich regulacji. Jeśli więc testowane rozwiązanie sieciowe wymaga specyficznego kodowania sygnału, nietypowych długości ramek (bardzo krótkich lub bardzo długich) czy przestrzegania ścisłych zależności czasowych, to ograniczenia narzucane przez sprzęt mogą być nie do obejścia. Często można jednak dopasować lub pominąć te kwestie – odpowiednio modelując środowisko testowe i dobierając zakres spełnianych wymogów.

Pewne ograniczenia (funkcjonalne, czasowe) może wprowadzać wybrana platforma programowa, dla której tworzymy sterownik, jednak w przypadku otwartych systemów ograniczenia te są zwykle możliwe do obejścia poprzez proste modyfikacje.

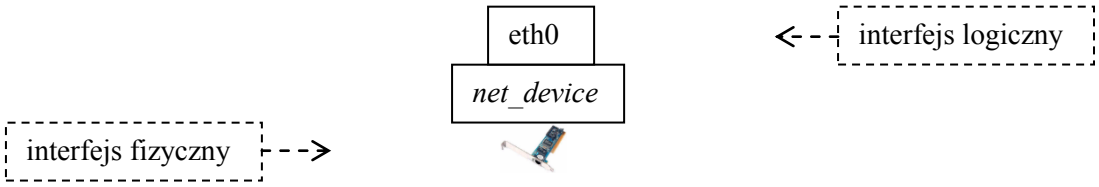
Problemem może być dostępność dokumentacji technicznej lub otwartego sterownika, jednak większość popularnego sprzętu SOHO ma otwarte sterowniki[2], np. dla systemu GNU/Linux, zatem zwykle istnieje możliwość bazowania na już istniejącym kodzie [3].

Można jednak wymienić kilka niepodważalnych zalet stosowania tej techniki. Najbardziej widoczną są niskie koszty. Karta sieciowa i przeznaczony do testów komputer kosztują zwykle ułamek ceny profesjonalnego sprzętu laboratoryjnego. Nie bez znaczenia jest również bardzo duża dostępność takich urządzeń. W parze z dostępnością i niską ceną idzie również popularność, co przekłada się na dostępność otwartych sterowników na popularne platformy.

[2] Na przykład dla kart ethernetowych PCI firmy Realtek Semiconductor

1.1. Wykorzystanie interfejsów logicznych

Oryginalnie każde urządzenie sieciowe w systemie GNU/Linux ma własną strukturę systemową *net_device* w przestrzeni sterownika (rozumianej jako zestaw danych zarządzanych przez sterownik oraz możliwe do podjęcia przez niego działania w jądrze systemu).



Rys. 1. Architektura oryginalnego sterownika w kontekście pojedynczego urządzenia fizycznego
Fig. 1. Original driver's architecture

Po jej zarejestrowaniu funkcją systemową *register_netdev* stają się one dostępne dla systemu operacyjnego pod postacią interfejsów logicznych, widocznych w systemie jako ethX (X jest kolejnym numerem). Droga, jaką przebywa ramka jest w tym przypadku bardzo prosta: ramka otrzymana od systemu operacyjnego za pomocą interfejsu logicznego, np. eth0, trafia do odpowiadającej temu interfejsowi struktury *net_device* i jest przekazywana bezpośrednio do bufora nadawczego powiązanego urządzenia fizycznego. Podobnie ramka odebrana przez dane urządzenie fizyczne zostaje przekazana do systemu operacyjnego za pośrednictwem powiązanego ze strukturą *net_device* tego urządzenia interfejsu logicznego, np. eth0. Upraszczając, interfejs logiczny jest jednoznacznie reprezentowany przez strukturę *net_device*, jednoznacznie powiązaną z danym urządzeniem fizycznym.

Przy budowie sieci o zaawansowanej topologii może zaistnieć potrzeba umieszczenia więcej niż jednego urządzenia sieciowego w danej stacji. W takiej sytuacji warto jest stworzyć nadrzędną, własną strukturę *net_device* (nazwijmy ją *sup_net_device*), którą rejestrujemy w systemie zamiast oryginalnej struktury *net_device* karty sieciowej. Dzięki temu karty nie są bezpośrednio widoczne w systemie, ani dostępne dla niego – są używane jedynie wewnętrznie przez nasz sterownik. Struktura *sup_net_device* zawiera odwołania do oryginalnych struktur *net_device*, oddzielnych dla każdej karty sieciowej, co daje nam pełną kontrolę nad przepływem danych pomiędzy urządzeniami sieciowymi w obrębie jednej stacji.

Tym sposobem można, dla każdej klasy ruchu, zarejestrować po jednym dodatkowym interfejsie lub w dowolny inny sposób tworzyć logiczne interfejsy, używane przez system operacyjny.

1.2. Znakowanie ramek

Aby zapobiec każdorazowemu konwertowaniu ramek ethernetowych, można zachować standardowy format ramki IEEE 802.3 [4], natomiast do przenoszenia razem z ramką dodat-

kowych informacji, koniecznych do pracy sieci, można wykorzystać pole standardu IEEE 802.1Q, czyli VLAN Tag [5]. Jest to wygodne rozwiązanie, umożliwiające dodanie do każdej transmitowanej ramki dodatkowych informacji, bez programowego modyfikowania ramki – tag jest przekazywany do karty równolegle z ramką, a zadaniem karty jest umieścić taga w ramce, obliczyć sumę kontrolną i wysłać całość. Podczas odbierania ramek również zadaniem karty jest sprawdzić sumę kontrolną, wyciąć z ramki taga oraz przekazać parę ramka oraz tag do sterownika. Dzięki temu można również wykorzystać istniejące mechanizmy znajdowania adresu węzła docelowego.

Wadą wykorzystanego rozwiązania jest brak możliwości użycia (zgodnie z założeniem) funkcji tagowania sieci wirtualnych. Inną niedoskonałością tej techniki jest bardzo ograniczona pojemność VLAN Taga – mamy do dyspozycji jedynie 15 bitów, co ogranicza jego zastosowanie jedynie do przenoszenia flag lub niewielkich liczników.

1.3. Klasy ruchu

Nawet najtańsze karty ethernetowe udostępniają przynajmniej dwa nadawcze pierścienie DMA[3] (obszary pamięci służące przekazywaniu danych do i z karty, poprzez mechanizm *Direct Memory Access*). Polega to na tym, że definiujemy dwa niezależne pierścienie DMA do nadawania – normalny i priorytetowy – po czym umieszczamy w nich ramki, zależnie od kolejności, w jakiej powinny zostać wysłane – ważniejsze do bufora priorytetowego, mniej ważne do bufora standardowego. W założeniu obowiązkiem karty jest wysłać całą zawartość bufora priorytetowego przed rozpoczęciem wysyłania z bufora standardowego.

Dzięki zastosowaniu tego mechanizmu w połączeniu z rejestracją oddzielnych interfejsów logicznych można odwzorować działanie klas ruchu, przy czym różnice w obsłudze ruchu różnych klas mogą dotyczyć nie tylko priorytetowego traktowania danego strumienia, ale i zastosowanych algorytmów sprawiedliwości. Klasę ruchu, do której należy dana ramka można zawrzeć np. w opisanym powyżej VLAN Tagu.

2. Implementacja sieci typu RPR na kartach ethernetowych

W celu sprawdzenia efektywności, proponowanego w rozdziale pierwszym, podejścia do budowy sieci, zaimplementowano dwupierścieniową sieć komputerową (sieć autorska), wspierającą niezawodność, klasy ruchu, autokonfigurację i utrzymywanie topologii. Sieć ta wzorowana jest na rozwiązaniu RPR (Resilient Packet Ring).

[3] Na przykład karty oparte na chipie Realtek RTL8169SC

2.1. Sieć RPR

Sieć RPR to światłowodowa sieć dwupierścieniowa, zaprojektowana do wykorzystania głównie w sieciach miejskich i rozległych. Z założeń miała być szybka, wysoce skalowalna, niezawodna i sprawiedliwa oraz gwarantować jakość obsługi [6]. Protokół RPR zatwierdzony został w 2004 roku, standardem IEEE 802.17 [7]. Do tej pory nie doczekał się komercyjnej implementacji.

RPR spośród innych protokołów dwupierścieniowych wyróżnia to, że oba, przeciwnie skierowane pierścienie są jednakowe i równoważne, co oznacza, że oba są wykorzystywane zarówno do obsługi transmisji danych, jak i ramek sterujących [6].

Sieć autorska symuluje niektóre mechanizmy sieci RPR na kartach SOHO. W tabeli 1 dokonano porównania obu rozwiązań (implementacja mechanizmów sprawiedliwości jest w trakcie realizacji).

Tabela 1

Porównanie sieci autorskiej oraz RPR

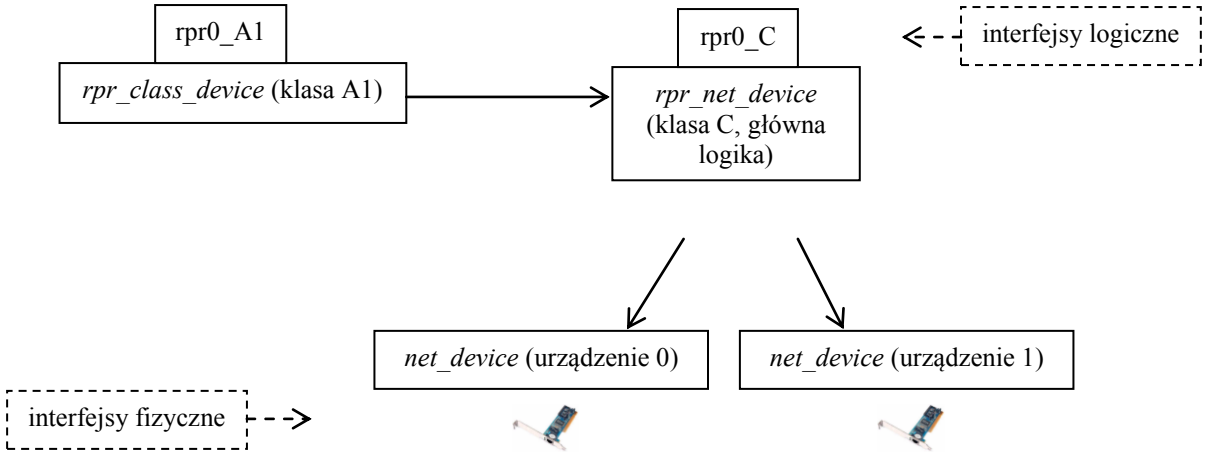
	Sieć autorska	RPR IEEE802.17
Topologia	Dwa przeciwległe, równoważne pierścienie	Dwa przeciwległe, równoważne pierścienie
Format ramki	Zachowano format ramki ethernetowej, ramki są znakowane za pomocą pola VLAN Tag. Wyróżniamy ramki *Topology Protection, Topology Checksum*, ramki mechanizmu sprawiedliwości oraz ramki danych	W protokole RPR wyróżniamy cztery typy ramek: danych, sprawiedliwości, kontrolne oraz *idle*. Rozróżnić je można po dwubitowym polu *ft*
Niezawodność	Obsługuje tryb *steering protection*	Działa w dwóch trybach – *steering protection* oraz *wrapping protection* (opcjonalny)
Klasy ruchu	A, C	A (A0, A1), B (B-CIR, B-EIR), C
Sprawiedliwość	Proaktywna, kontrolująca podział łącza za pomocą ramek sprawiedliwości	Reaktywna, wysyłająca ramki tłumiące

2.2. Sterownik

Autorska implementacja jest modułem jądra dla systemu Linux w wersji 2.6.26 (stabilne jądro systemu Debian „Lenny"). Bazuje na sterowniku r1000, firmy TP-Link Technologies. Sterownik ten, po znacznych modyfikacjach, stanowi bazę dla autorskiego rozwiązania oraz pełni funkcję interfejsu do obsługi fizycznych kart przez dodaną nową warstwę logiki RPR. Poddano go usprawnieniom, naprawiając drobne błędy oraz dodając niezaimplementowane oryginalnie funkcje (np. dwa niezależne bufory nadawcze dla każdego urządzenia).

2.3. Architektura

W autorskiej implementacji zachowano reprezentację fizycznych kart sieciowych w postaci oryginalnych struktur *net_device*, zrezygnowano jednak z rejestrowania ich w systemie. Dzięki temu karty nie są bezpośrednio widoczne w systemie operacyjnym, ani nie są dostępne dla niego – są używane jedynie wewnętrznie przez sterownik. Dla każdej pary urządzeń rejestruje się dodatkową strukturę *net_device* (dalej zwaną *rpr_net_device*), która staje się widoczna w systemie, jako interfejs o nazwie rprX_C (X jest kolejnym numerem, C to klasa ruchu best-effort). Ponadto, rejestrowany jest dodatkowy interfejs logiczny dla każdej dodatkowej klasy ruchu – obecnie jest to interfejs rprX_A1 dla klasy A1. Struktura *rpr_net_device* jest centralnym punktem w architekturze naszego węzła sieci RPR i koordynuje ona pracę kart sieciowych (reprezentowanych przez oryginalne struktury *net_device*), interfejsów logicznych wszystkich klas oraz implementuje podstawowe mechanizmy RPR, a także bezpośrednio klasę ruchu C. Pozostałe klasy ruchu (obecnie klasa A1) korzystają z oddzielnych struktur *net_device* (będziemy je dalej nazywać *rpr_class_device*), stanowiących oddzielne interfejsy logiczne, ale wewnętrznie i tak korzystają z usług *rpr_net_device*.



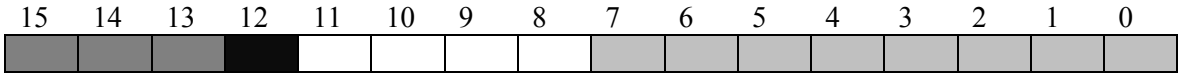
Rys. 2. Architektura zmodyfikowanego w ramach autorskiej implementacji sterownika, w kontekście pojedynczej sieci
Fig. 2. Modified driver's architecture

2.4. Format ramek

Starano się (w miarę możliwości) zachować oryginalny format ramki ethernetowej. Powodem był fakt, iż system operacyjny traktuje interfejsy rprX_Y jako zwyczajne urządzenia ethernetowe i spodziewa się, iż będą one operowały również na ramkach ethernetowych. Aby zapobiec każdorazowemu konwertowaniu ramek ethernetowych na RPR-owe i na odwrót, zachowano standardowy format ramki 802.3, natomiast do przenoszenia razem z ramką dodatkowych informacji, koniecznych do pracy sieci, wykorzystano pole standardu IEEE 802.1Q, czyli VLAN Tag.

VLAN Tag umożliwia wykorzystanie 15 bitów do przenoszenia dodatkowych informacji. 3 bity, wykorzystane w IEEE 802.1Q jako priorytet, reprezentują klasę ruchu ramki, pozostałe 12 bitów, stanowiące VLAN ID zostały wykorzystane do innych celów. Wykorzystano 8 najmłodszych bitów do przenoszenia informacji TTL (ang. *Time to Live)*, na podstawie której węzły podejmują decyzję czy usunąć ramkę z pierścienia. Pozostałe 4 bity traktujemy jako zarezerwowane i obecnie nieużywane.

Identyfikatory klas ruchu są zgodne z tymi zdefiniowanymi dla RPR. Dodano jednak dodatkową klasę: *RPR_CL_CONTROL* dla ramek kontrolnych. Dzięki temu już na podstawie VLAN Taga sterownik decyduje, w jaki sposób powinien obsłużyć ramkę, czego powinien się w niej spodziewać i czy należy analizować jej zawartość. Dzięki temu ogranicza się analizowanie ramek z danymi i dodawanie do nich dodatkowych pól.



Legenda

	Priority Code Point – użyty jako identyfikator klasy ruchu
	Canonical Format Indicator – ustawiony zawsze na 0
	VLAN Identifier (4 najstarsze bity) – zarezerwowany
	VLAN Identifier (8 najmłodszych bitów) – użyty jako TTL

Rys. 3. Wykorzystanie 15 bitów taga IEEE 802.1Q do przekazania podstawowych danych o ramce
Fig. 3. Use of the IEEE 802.1Q VLAN Tag

2.5. Ramki kontrolne

Na potrzeby mechanizmu niezawodności *steering* zdefiniowano dwie ramki kontrolne. Ramka *Topology/Protection* zachowała oryginalną (zdefiniowaną dla RPR) rolę – przenosi informacje, pozwalające pozostałym węzłom zbudować i utrzymywać bazę topologii sieci. Ramka ta wysyłana jest do wszystkich węzłów w sieci. Zawiera ona m.in. informację, czy jej nadawca jest na tym pierścieniu krawędzią, tj. nie jest w stanie nawiązać komunikacji z następnym sąsiadem. Algorytm tworzenia ramki ilustruje rys. 4.

Węzły po odebraniu takiej ramki zachowują identyfikator węzła oraz flagi w tablicy pod indeksem zależnym od wartości TTL odebranej ramki – w ten sposób kolejność wpisów w tablicy topologii odpowiada kolejności węzłów w pierścieniu. Tablice topologii są utrzymywane całkowicie niezależnie dla obu pierścieni. Ramkę TP należy rozesłać, jeśli zmienił się stan nośnej na którymś z interfejsów, kiedy nie możemy nawiązać komunikacji z sąsiadem lub kiedy ramka TP, odebrana od kogoś innego, spowodowała modyfikację w tablicy topologii. Każde takie zdarzenie wywołuje chwilową lawinę ramek TP, po której zakończeniu bazy topologii na wszystkich węzłach są spójne [6].

Do wykrywania problemów komunikacyjnych z sąsiadami służą ramki *Topology Checksum*. Nazwa została zachowana ze specyfikacji RPR, gdzie ramki te służą do kontrolowania

spójności baz topologii pomiędzy sąsiadami. W prezentowanej implementacji nie przenoszą żadnych danych oprócz swojego identyfikatora. Są przesyłane w obie strony, dokładnie co *t* równe 15 milisekund, tylko do najbliższego sąsiada i służą do informowania go o naszej obecności. Nieotrzymanie takiej ramki w czasie 3 *t* powoduje uznanie tego kierunku ruchu za nieczynny i rozesłanie informacji o nowej krawędzi w sieci. Następne otrzymanie takiej ramki przywraca prawidłową komunikację (rozesłana zostaje ramka TP z informacją, że stacja już nie jest krawędzią). Taki mechanizm umożliwia wykrywanie awarii nie tylko, gdy utracimy nośną na łączu fizycznym, ale także w sytuacji, gdy np. sąsiedni węzeł nie odpowiada.

```
static struct sk_buff *rpr_create_topo_frame(struct rpr_net_device *dev, u8 protstatus) {

    struct rpr_co_tp *tp_data;
    struct sk_buff *skb;
    int i;

    //Przygotowujemy SKB
    skb = alloc_skb(sizeof(struct rpr_co_tp) + NET_IP_ALIGN + 8, GFP_ATOMIC);
    skb_reserve(skb, NET_IP_ALIGN);     /* przesuniecie o 2 bajty jakbysmy wyrownywali dla naglowka IP */

    tp_data = (struct rpr_co_tp*) skb_put(skb, sizeof(struct rpr_co_tp) + 8);

    //Czyszczenie
    memset(tp_data,0x00,sizeof(struct rpr_co_tp) + 8);

    //Przygotujmy dane
    for(i=0; i<6; i++)
    {
        tp_data->dst_mac[i] = 0xFF;                 /* dst_mac - broadcast          */
        tp_data->src_mac[i] = (u8) dev->dev_addr[i];    /* src_mac - nasz MAC */
    }
    tp_data->ethertype =        _ETYPE;             /* ethertype = 0xF0CA                   */
    tp_data->co_frame_type =    RPR_CO_TP;          /* co_frame_type = Topology/Protection  */
    tp_data->prot_status =      0x00FF & protstatus;    /* prot_status czyli czy zawijamy */

    // RPR_DBG("Creating new T/P frame:");
    // print_hex(skb->data, skb->len);

    //dane gotowe
    return skb;
}
```

Rys. 4. Implementacja algorytmu tworzenia ramki *Topology Protection*
Fig. 4. Implementation of the *Topology Protection* frame creation algorithm

2.6. Obsługa ramki wychodzącej, wybór pierścienia

Dla każdej ramki wysyłanej do innego węzła w sieci musi zostać podjęta decyzja, którym pierścieniem należy to zrobić.

W przypadku normalnej pracy węzła, dla ramek unicastowych wykorzystuje się bazę topologii sieci. Wybierany jest ten pierścień, na którym odległość stacji od nadawcy do adresata jest mniejsza. Wartość TTL ustawia się na tę odległość. Jeśli adresat jest w takiej samej odległości w obu pierścieniach, wybór następuje drogą losowania. Jeśli natrafiono na krawędź podczas przeszukiwania bazy topologii, wybierany jest przeciwny pierścień, a TTL jest równy odległości lub 255, zależnie od tego, czy adresat znajduje się w tablicy topologii przeciwnego pierścienia.

W przypadku gdy węzeł nadający jest krawędzią, ramka zostaje wysłana w przeciwnym kierunku z TTL równym 255.

W przypadku ramek *Broadcast* i *Multicast* (BC/MC) i w sieci nie ma krawędzi, pierścień jest zawsze losowany, a TTL ustawiany na 255. Jeśli krawędź występuje, wysyłane są oddziel-

ne ramki dla każdego pierścienia. TTL obu ramek jest dostosowany do najbliższej krawędzi w danym pierścieniu. Zapewnia to dostarczenie ramek BC/MC do wszystkich węzłów, z którymi jest czynna komunikacja i jednocześnie zapobiega okrążeniu przez którąkolwiek z kopii całego pierścienia – w przypadku naprawy awarii – co skutkowałoby duplikacją tych ramek.

Istnieje możliwość wyboru domyślnego pierścienia zamiast losowania. Umożliwia to np. testowanie wydajności komunikacji pomiędzy węzłami, znajdującymi się w równej odległości – każdym pierścieniem, z użyciem określonej trasy.

Wybranie pierścienia docelowego dla ramki wychodzącej skutkuje umieszczeniem takiej ramki w odpowiednim buforze nadawczym urządzenia, odpowiedzialnego za wysyłanie ramek w wybranym pierścieniu (w przypadku wszystkich klas poza klasą C) lub w specjalnej kolejce tymczasowej (w przypadku klasy ruchu C), z której ramki są przekazywane do buforów nadawczych, zgodnie z działaniem algorytmu sprawiedliwości.

Ramki usuwane są z sieci po jej okrążeniu i dotarciu do nadawcy lub gdy dekrementowana z każdą stacją wartość TTL będzie równa zeru.

2.7. Obsługa ramki przychodzącej

Ramka przychodząca, po odebraniu przez jedno z urządzeń, jest analizowana pod względem klasy ruchu. Ramki o nieokreślonej lub błędnej klasie ruchu są odrzucane, niezależnie od ich źródła, przeznaczenia i zawartości. Jeśli ramka nie została odrzucona, następuje dalsza analiza jej zawartości, podejmowanie decyzji dotyczących dalszej obsługi oraz dekrementacja wartości TTL. W ramach dalszej obługi możliwe jest przekazanie jej systemowi operacyjnemu, usunięcie lub przekazanie do bufora nadawczego, celem przesłania dalej.

2.8. Klasy ruchu

Zdefiniowano 7 klas ruchu (tabela 2). Według specyfikacji RPR wszelkie ramki kontrolne, sterujące i zarządzające mechanizmem sprawiedliwości są przesyłane w klasie A0. Jednakże klasa ta może również służyć do przekazywania danych, zatem dla tych ramek wydzielono oddzielne klasy ruchu, by łatwiej odróżniać je od ramek transportujących dane. Ostatni bit identyfikatora klasy ruchu w IEEE 802.17 oznacza jej podatność na mechanizmy sprawiedliwości. W tabeli 2 został on podkreślony, jeśli jego znaczenie zostało zachowane. W efekcie, spośród zaimplementowanych klas ruchu, klasa A1 jest priorytetową, klasa C jest klasą best-effort, mogącą podlegać np. mechanizmowi sprawiedliwości, pozostałe są na użytek wewnętrzny (są priorytetowe i nie przenoszą danych użytkowych).

W implementacji sieci wykorzystano dwa nadawcze pierścienie DMA – priorytetowy i standardowy. W założeniu obowiązkiem karty jest wysłać całą zawartość bufora priorytetowego przed rozpoczęciem wysyłania z bufora standardowego. W praktyce okazało się to

niewystarczające do pełnego zagwarantowania klasie A1 oraz klasom dodatkowym pierwszeństwa nad klasą C. Sytuację jednak nieco poprawiło dodanie funkcji oczyszczania pierścieni DMA z wysłanych już ramek przy każdej próbie nadawania.

Tabela 2

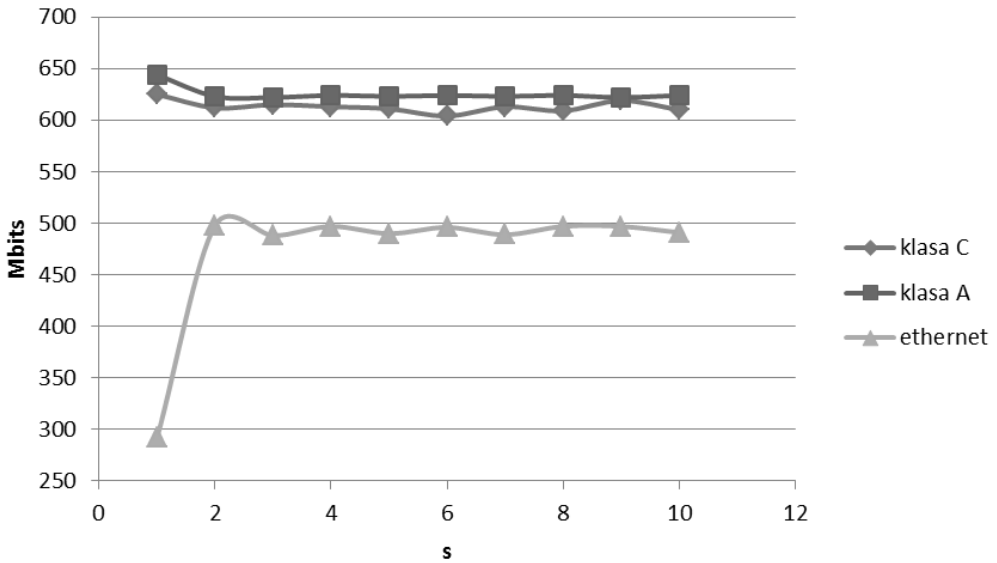
Klasy ruchu

3-bitowy identyfikator	Nazwa wg specyfikacji RPR (klasa / podklasa)	Nazwa w autorskiej implementacji	Pierścień TX DMA	Zaimplementowana
11<u>0</u>	A / subclassA0	RPR_CL_A0	Priorytetowy	nie
10<u>0</u>	A / subclassA1	RPR_CL_A1	Priorytetowy	tak
01<u>0</u>	B / classB-CIR	RPR_CL_BCIR	Normalny	nie
01<u>1</u>	B / classB-EIR	RPR_CL_BEIR	Normalny	nie
00<u>1</u>	C / – (best-effort)	RPR_CL_C	Normalny	tak
101	– (zarezerwowana)	RPR_CL_CONTROL	Priorytetowy	tak (poza specyfikacją)
111	– (zarezerwowana)	RPR_CL_FAIRNESS	Priorytetowy	tak (poza specyfikacją)
00<u>0</u>	– (zarezerwowana)	– (zarezerwowana)	–	nie

2.9. Ocena rozwiązania

Testy wykazały, że stosując niemodyfikowany, oryginalny sterownik ethernetowy stacje osiągały średnią przepustowość 473 Mbit/sec. Rysunek 5 przedstawia wydajność sieci zmodyfikowanej, odpowiednio w klasach ruchu C i A oraz sieci Ethernet, z użyciem oryginalnego sterownika.

Przy komunikacji między sąsiednimi stacjami nie zauważa się spadku wydajności w stosunku do oryginalnego rozwiązania. Poczynione modyfikacje sterownika spowodowały nawet wzrost przepustowości.



Rys. 5. Wydajność komunikacji pomiędzy sąsiadami
Fig. 5. Neighbor communication effectiveness

3. Podsumowanie

W artykule dokonano oceny możliwości modelowania za pomocą modyfikacji otwartoźródłowego sterownika tanich kart ethernetowych, złożonych rozwiązań sieciowych – takich, które istnieją już na rynku, a także takich, które są dopiero projektowane.

Karty ethernetowe oferują duże możliwości symulacji mechanizmów sieciowych. Są bardzo tanie, powszechnie dostępne i gwarantują wysokie przepustowości, rzędu Gb/s. Dzięki popularności Ethernetu dostępnych jest wiele sterowników kart sieciowych, publikowanych na zasadach Open Source. Implementacja mechanizmów sieciowych na poziomie sterownika systemu Linux pozwala na stworzenie struktury, którą można zarejestrować w systemie jako logiczny interfejs w miejsce struktury *net_device*. Rozwiązanie to daje pełną kontrolę nad przepływem ramek w obrębie jednej stacji.

Dzięki alternatywnemu wykorzystaniu pola VLAN Tag można dodawać własne flagi do danych, bez konieczności deklarowania nowego formatu ramki (wymagałoby to każdorazowej konwersji ramek). To rozwiązanie jest korzystne, ponieważ system operacyjny traktuje takie interfejsy jako zwyczajne urządzenia ethernetowe i spodziewa się, iż będą one operowały również na ramkach ethernetowych. Dzięki temu można również wykorzystać istniejące mechanizmy znajdowania adresu węzła docelowego. Wykorzystanie VLAN Tagów daje możliwość tworzenia i rozpoznawania ramek kontrolnych i sterujących.

Wykorzystując powyższe mechanizmy możliwe jest zaimplementowanie symulatora złożonej sieci komputerowej. Można budować sieci o różnych topologiach, zarówno gwiaździstych, jak i pierścieniowych, wykorzystywać w nich mechanizmy autokonfiguracji czy niezawodności.

Przeprowadzone eksperymenty na 32-bitowych magistralach PCI wskazują, iż w przypadku wykorzystania gigabitowych kart sieciowych warto rozważyć użycie urządzeń korzystających z interfejsu PCI Express. Prędkości transferu osiągane przez urządzenia gigabitowe powodują wysycenie możliwości 32-bitowej magistrali PCI, a na wyniki pomiarów mogą mieć nieprzewidziany wpływ inne urządzenia współdzielące magistralę.

BIBLIOGRAFIA

1. http://www.zelscope.com, stan na marzec 2011.
2. http://en.wikipedia.org/wiki/Softmodem, stan na marzec 2011.
3. http://www.realtek.com.tw/downloads/downloadsView.aspx?Langid=1&PNid=13&PFid=4&Level=5& Conn=4&DownTypeID=3&GetDown=false&Downloads=true
4. http://standards.ieee.org/about/get/802/802.3.html, stan na marzec 2011.

5. http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf, stan na marzec 2011.
6. Davik F. i in.: IEEE 802.17 Resilient Packet Ring Tutorial, 2003.
7. http://standards.ieee.org/getieee802/download/802.17-2004.pdf, stan na październik 2010.

Recenzenci: Prof. dr hab. inż. Tadeusz Czachórski
Prof. dr hab. inż. Stanisław Kozielski

Wpłynęło do Redakcji 13 marca 2011 r.

Abstract

Ethernet devices are capable of simulating different network mechanisms. They are cheap, commonly available and guarantee throughput up to 1Gbps. Due to Ethernet's popularity, a lot of Open Source drivers are published. Implementation of network mechanisms in a Linux driver gives the possibility of creating a structure, which can be registered in the OS as logical interface in place of the original *net_device* structure. This technique gives us full control of data flow in stations within the network. The IEEE 802.1Q standard provides the capability to mark frames. By alternative use of the VLAN Tag field flags can be defined and easily added to frames without changing the frame format. An advantage of this solution is the fact that the OS treats devices as Ethernet NICs and expects them to work on Ethernet frames. This provides the capability of using existing destination address resolution mechanisms. Frames can be marked with traffic class flag or as a control frame. The main disadvantage is the fact, that it does not support double tag VLANs. Using the mechanisms mentioned a simulator of an advanced computer network can be implemented. It is possible to build networks based on different topologies, using autoconfiguration and reliability mechanisms.

Given the case of gigabit Ethernet NICs, one may consider using devices based on the PCI Express interface. Transfer speed reached by gigabit NICs may exhaust the capabilities of 32 bit PCI bus. Other devices sharing the bus can affect results of experiments.

Adresy

Krzysztof NOWICKI: Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, ul. Narutowicza 11/12, 80-223 Gdańsk, Polska, krzysztof.nowicki@eti.pg.gda.pl
Adrian OSTROWSKI, Aurelia POŹNIAK, Łukasz WRZESIŃSKI: Politechnika Gdańska Wydział Elektroniki, Telekomunikacji i Informatyki, ul. Narutowicza 11/12, 80-223 Gdańsk, Polska, {aurpozni, lukwrzes, adrostro}@student.pg.gda.pl