WOJNY TOKENOWE

Projekt przewiduje napisanie programu działającego pod kontrolą systemu czasu rzeczywistego RTEMS umożliwiającego udział w grze Wojny Tokenowe. Program należy napisać używając języka C. Dostępne na laboratorium SMCR płyty prototypowe wyposażone w procesory Freescale MCF5282 dołączone zostaną do wspólnej sieci komputerowej poprzez interfejs Ethernet 10/100. Dołączone urządzenia przesyłają tokeny zgodnie z przedstawionym poniżej protokołem. Urządzenia dołączone zostaną do wspólnego przełącznika sieciowego (ang. switch). Urządzenia komunikują się wykorzystując gniazda sieciowe (ang. sockets) wysyłając datagramy UDP (User Datagram Protocol). W celu dostarczenia pakietów do wszystkich systemów dane przesyłane są w postaci pakietów rozgłoszeniowych (ang. Broadcast). Celem wspólnej zabawy jest przeszukanie wirtualnej pamięci systemu komputerowego przeciwnika i odnalezienie ukrytych liczników energii. Po odnalezieniu liczników należy wyzerować ich zawartość wysyłając rozkazy DECREMENT. System mikroprocesorowy, w którym odnaleziono i wyzerowano wszystkie liczniki kończy grę. Pozostałe urządzenia walczą między sobą dalej. Zwycięzca pozostaje ostatni system, który będzie miał energię w jednym z liczników. W czasie walki, urządzenia pasywne, które nie biorą chwilowo udziału w walce, mogą również odbierać pakiety. Istanieje możliwość "podsłuchania" adresów, pod którymi znajdują się liczniki przeciwników, jednak zakłócanie transmisji nie jest dozwolone.

Informacje podstawowe

System operacyjny RTEMS jest systemem czasu rzeczywistego przeznaczonym dla urządzeń wbudowanych. Pierwotnie nazwa RTEMS oznaczała Real-Time Executive for Missile Systems, następnie Real-Time Executive for Military Systems, a w końcu wersja systemu napisana w języku C została nazwana Real-Time Executive for Multiprocessor Systems. Cechą wyróżniającą system RTEMS wśród szeregu innych komercyjnych rozwiązań dostępnych na rynku jest fakt że system ten jest rozpowszechniany na licencji GNU General Public License. Jedyne odstępstwa od powyższego sposobu licencjonowania dotyczą takich elementów systemu jak stos TCP/IP, RPC/XDR oraz WebServer. System RTEMS charakteryzują następujące cechy:

- Zgodność ze standardem POSIX 1003.1
- Pełna impelmentacja stosu TCP/IP z uwzględnieniem następujących elementów:
  - UDP, TCP
  - ICMP, DHCP, RARP
  - TFTP
  - RPC
  - FTPD
  - HTTPD
  - CORBA
  - SNMP
- Wsparcie dla nastepujących systemów plików:
  - IMFS (In-Memory File System)
  - TFTP Client File System
  - FTP Client File System
  - FAT File System

- Wsparcie dla języka skryptowego Python
- Podstawowe cechy jądra systemu:
  - Wsparcie dla wielozadaniowości
  - Możliwość pracy zadań w trybie z wywłaszczaniem albo bez wywłaszczania
  - Algorytmy przeciwdziałania inwersji priorytetów
  - Wsparcie dla mechanizmów synchronizacji oraz komunikacji międzyprocesowej (semafory, zdarzenia, kolejki komunkatów)
  - Wsparcie dla mechanizmów dynamicznego alokowania pamieci

System RTEMS wspiera ponad 70% standardu POSIX 1003.1b-1996, który definiuje interfejs programistyczny dla systemów opracyjnych UNIX. Z tego względu możliwe jest bezproblemowe przenoszenie wiekszości programów działających na systemie UNIX na platformę RTEMS. Dokładne informacje na temat funkcji standardu POSIX zaimplementowanych w systemie RTEMS znaleść można na stronie www.rtems.org.

Proces konfiguracji

Po włączeniu zasilania wszystkie urządzenia przechodzą w tryb bezczynności (idle). Przed rozpoczęciem wirtualnej "walki" należy wykonać podstawowe czynności mające na celu przygotowanie danego systemu mikroprocesorowego do dalszej pracy.

Konfiguracja zasobów sprzętowych systemu

W czasie walki wykorzystywane zostaną następujące zasoby systemu mikroprocesorowego: timer, interfejs szeregowy (pracujący zgodnie ze specyfikacją EIA RS232) zewnętrzny wyświetlacz LCD, diody LED oraz interfejs sieci Ethernet. Po operacji inicjalizacji systemu operacyjnego należy skonfigurować i zainicjować powyższe urządzenia (konfiguracją niektórych urządzeń zajmuje się system operacyjny RTEMS). Timer programowy dostarczany przez system RTEMS powinien zostać tak zaprogramowany, aby odmierzał przedziały czasowe równe zawartości stałej ROUND. Port szeregowy powinien zostać dostosowany do transmisji 19200 bodów (zawartość komórki TRANSMISSION_SPEED), w trybie 8N1.

Konfiguracja pamięci

Podczas pisania programu należy zadeklarować kilka stałych wartości (ROUND, ROBUST, MASTER, MAX_MEM) służących do łatwej modyfikacji parametrów programu. Należy zarezerwować główny obszar pamięci o wielkości 64 kB (16 kB x 32).

W pamięci tej należy umieścić trzy liczniki Q1, Q2 oraz Q3. W licznikach powinny zostać wpisane wstępne wartości "energii" danego systemu. Dostępna liczba zasobów energii wynosi 10000. Sposób podziału zasobów nie jest określony i zależy tylko od inwencji programisty (najprostszy podział: 1 i 2 licznik 3333, 3-licznik 3334). Przed komórką pamięci każdego licznika musi znajdować się 32-bitowa sygnatura, 0x5A5A5A5A. Dodatkowo należy zarezerwować trzy komórki pamięci, w których zostaną zapisanych adresy powyższych liczników.

Komórka pamięci MASTER wskazuje, czy dany system posiada pakiet kontrolny TOKEN (system pracuje jako serwer sieciowy, który poszukuje swojej ofiary generując tokeny REQUEST). Po restarcie zawartość zmiennej MASTER jest równa 0. W komórce o nazwie TIMER będzie zapisywany czas, który upłynął od momentu rozpoczęcia "walki". Urządzenie nie może pracować dłużej niż czas określony przez stała ROUND. Należy zadeklarować zmienną informująca o tym, czy dany węzeł jest aktywny ROBUST (może brać udział w grze). Po restarcie należy wpisać 1.

Nazwa stałej	Wartość
Wskaźnik do licznika 1	0x10000
Wskaźnik do licznika 2	0x13333
Wskaźnik do licznika 3	0x27777
Sygnatura	0x5A5A5A5A
***(Adres licznika 1)**	Zawartość licznika 1
***(Adres licznika 2)**	Zawartość licznika 2
***(Adres licznika 3)**	Zawartość licznika 3
MASTER	1
MAX_MEM	0x10000
ROBUST	1
ROUND	0
NODE_ADD	1
KodStartu	0x55
KodKonca	0xAA
TRANSMISSION_SPEED	19200
MAX_NODE	15

Tablica 1: Przykłady niektórych parametrów zdefiniowanych jako stałe

Reguły gry

"Zdekrementuj liczniki przeciwnika a wygrasz!"

Każdy z węzłów biorących udział w grze ma przyznany indywidualny adres (zmienna NODEADD). Urządzenie MAIN_MASTER (główne urządzenie rozpoczynające transmisję, może generować nowe tokeny) posiada adres 1. Pozostałe urządzenia posiadają adresy od 2 do MAX_NODE.

Po zakończeniu inicjalizacji urządzeń peryferyjnych, rozpoczyna się proces konfiguracji. Jest on przeprowadzany przez Master-a głównego. Urządzenie o adresie 1 rozpoczyna proces odpytywania kolejnych urządzeń (pozostałe urządzenia znajdują się w stanie idle). W pierwszej kolejności do systemu o adresie 2 wysyłany jest rozkaz REQUEST wymuszający natychmiastową odpowiedź. Jeżeli urządzenie nie prześle pakietu potwierdzenia w ciągu 250 ms (zawartość komórki TIMER) węzeł o danym numerze uznawany jest za nieaktywny. Następny pakiet przesyłany jest do węzła o numerze 3. Jeżeli urządzenie wyśle liczbę pakietów o numerach rosnących równą MAX_NODE (urządzenie odpytuje każdy węzeł dwa razy) i nie otrzyma odpowiedzi gra zostaje zakończona, a urządzenie zostaje zwycięzca (zakładając, że pracuje zgodnie ze specyfikacją). Jeżeli jest to pierwsza runda oznacza to błąd w protokole lub transmisji danych. Jeżeli taka sytuacja nastąpi podczas gry oznacza to, że urządzenie, które jest Master-em zwyciężyło. Jeżeli urządzenie wysyłające token otrzyma pakiet potwierdzenia ACK przesłany od urządzenia o numerze zgodnym z pakietem REQUEST można rozpocząć proces przeszukiwania pamięci w celu obniżenia zawartości liczników Q1-Q3. Od tego momentu urządzenia dołączone do magistrali mogą mierzyć czas w celu wykrycia nieprawidłowości lub próby oszustwa (np. generowanie pakietów potwierdzenia z opóźnieniem).

Urządzenie rozpoczyna proces przeszukiwania pamięci pod kątem znalezienia sygnatury (wysyłając do urządzenia, z którym nawiązano transmisje rozkaz odczytu danej READ z komórki pamięci). Rozkaz spowoduje przesłanie odpowiedzi zawierającej informacje o zawartości komórki pamięci o adresie umieszczonym w rozkazie REQUEST. Master przeszukuje pamięć w zakresie od 0 do zawartości określonej przez stała MAX_MEM. Urządzenia tłumaczą odebrane adresy na adresy pamięci, w której znajdują się liczniki. To, w jaki sposób pamięć będzie przeszukiwana zależy tylko od osoby piszącej program. Im lepszy algorytm przeszukiwania, tym większa szansa na szybsze znalezienie sygnatury

i pokonanie przeciwników. Po znalezieniu sygnatury należy dokonać zmniejszenia licznika wysyłając rozkaz DECREMENT (spowoduje to zmniejszenie licznika znajdującego się zaraz za komórką sygnatury o 1). Instrukcja DECREMENT zwraca błąd wyłącznie w przypadku dostępu do adresu znajdującego się poza dostępną przestrzenia adresową. Jeżeli po zmniejszeniu wartość znajdujące się w tej komórce będzie równa 0, należy usunąć sygnaturę nadpisując ja wartością 0. Jeżeli liczniki są równe 0 należy uznać dany węzeł za martwy wpisując do komórki ROBUST wartość 0. Węzeł martwy nie może odpowiadać na żadne rozkazy. Proces przeszukiwania pamięci musi zostać wykonany w czasie zdefiniowanym przez stała ROUND.

Po zakończeniu walki z jednym urządzeniem lub odmierzeniu czasu należy przesłać rozkaz STORE do węzła, w którym modyfikowano liczniki. Rozkaz spowoduje przesłanie zawartości liczników Q1-Q3. Przesłanie tego rozkazu jest informacją o zakończeniu walki z danym węzłem. Wszystkie pakiety przesyłane przez systemy zapisywane są w pliku przez zewnętrzny komputer. Powyższy proces ma na celu sprawdzenie wiarygodności wykonanych operacji.

Po upłynięciu określonego przez zawartość komórki TIMER (np. 250 ms) system kończący turę, przesyła rozkaz REQUEST do węzła o numerze większym i czeka na odpowiedź. Jeżeli otrzyma odpowiedź przekazuję TOKEN do tego węzła. Jeżeli nie otrzyma odpowiedzi w określonym czasie odpytywany jest kolejny węzeł. Urządzenie wysyłające TOKEN przechodzi w stan uśpienia, a urządzenie odbierające staje się nowym Master-em. Rozpoczyna proces znalezienia kolejnego węzła, z którym będzie toczona walka. Powtarza się sytuacja opisana powyżej. Postępowanie zgodnie z podanymi regułami umożliwi cykliczną walkę kolejnych węzłów. Węzły będą umierały, aż do momentu, gdy pozostanie tylko jeden aktywny węzeł. Przesyłane pakiety będą logowane przez program umieszczone na pomocniczym komputerze PC. Po zakończonej walce zostanie dokonana analiza kluczowych pakietów oraz sprawdzona zostanie poprawność programu zwycięscy. W czasie walki można analizować przesyłane przez Master i Slave pakiety. Nie wolno jednak zakłócać transmisji podczas gry i niszczyć tokenów.

Protokół transmisji

Dane przesyłane są w postaci 11 bajtowych ramek. Każda ramka zaopatrzona jest w sygnaturę początku oraz końca, adres nadawcy i odbiorcy, kod rozkazu oraz przesyłaną daną, patrz tabela 2. Dana składa się z 4 bajtów. Najpierw przesyłany jest najbardziej znaczący bajt MSB. Każdy z węzłów ma przypisany adres (od 1 do MAX_NODE).

Kod startu	**Adres nadawcy**	**Adres odbiorcy**	**Kod rozkazu**	**Dana**	**Kod końca**
0xFFAA (16 bit)	(8 bit)	(8 bit)	0x0000 xxxx (8 bit)	MSB..LSB (32 bit)	(0xFF55) (16 bit)

Tablica 2: Ramka wysyłana przez urządzenie Master i Slave (długość ramki 11 bajtów)

Podstawowe rozkazy

Podstawowe rozkazy, które należy zrealizować wraz z ich krótkim opisem umieszczono w tabeli 3 oraz 4.

Rozkaz	Kod binarny	Opis
REQUEST	0001	Żądanie odpowiedzi węzła o danym adresie
READ	0010	Odczytanie zawartości komórki pamięci o danym adresie (adres przesyłany w polu danej)
DECREMENT	0011	Zwiększenie zawartości licznika należącego do danej sygnatury spod danego adresu
STORE	0100	Przesłanie zawartości wszystkich liczników
TOKEN	0101	Przekazanie Token-u do węzła o danym adresie

Tablica 3: Rozkazy węzła Master

Rozkaz	Kod binarny	Opis
ACK	1000	Potwierdzenie zapytania REQUEST
SEND	1001	Przesłanie zawartości komórki o danym adresie
DECACK	1010	Potwierdzenie wykonania operacji DECREMENT
READ_ERR	1011	Błąd odczytu komórki pamięci (poza dostępną przestrzenią adresową)
ERROR	1111	Błąd w realizacji przesłanego rozkazu

Tablica 4: Rozkazy węzła Slave

W tabeli 5 umieszczono odpowiedzi urządzeń na przesłane rozkazy w zależności od trybu pracy węzła (Master/Slave, aktywny/pasywny).

Rozwinięcie skrótów występujących w tabeli 5.

SA - adres węzła wysyłającego pakiet

TA - adres węzła odbierającego pakiet

CM - rozkaz do wykonania

VALUE - wartość odczytana z komórki pamięci o podanym wczesniej adresie

NEW_VAL - wartość po inkrementacji licznika

Zasady działania uproszczonej sieci Token Ring

Proponowana struktura sieci jest typu master-slave, a dokładniej multimaster z wykorzystaniem tzw. logicznego pierścienia tokenowego (logical token ring), z automatycznym przełączaniem węzłów między trybem aktywnym i pasywnym.

Zasady obowiązujące w sieci:

1. Każdy z węzłów aktywnych dołączonych do wspólnej magistrali pracuje w dwóch trybach transmisji danych: Master lub Slave. Przesyłanie danych odbywa się, zatem według zasady: zapytanie (Master) - odpowiedź (Slave).

2. Węzeł Slave nie ma prawa inicjowania transmisji. Może on jedynie odpowiadać na zapytania węzłów typu Master.

Rozkaz	Warunek	Odpowiedź
REQUEST	ROBUST = ACTIVE MASTER = MASTER	ERROR [|SA|RA|CM|0x00|]
	ROBUST = ACTIVE MASTER = SLAVE	ACK [|0x00|]
	ROBUST = PASIVE MASTER = SLAVE	Brak reakcji
TOKEN	ROBUST = ACTIVE MASTER = MASTER	ERROR [|SA|TA|CM|0x00|]
	ROBUST = ACTIVE MASTER = SLAVE	TOK_ACK [|0x00|]
	ROBUST = PASIVE MASTER = SLAVE	Brak reakcji
READ	ROBUST = ACTIVE MASTER = MASTER	ERROR [|SA|TA|CM|0x00|]
	ROBUST = ACTIVE MASTER = SLAVE	SEND [|VALUE|]/READ_ERR [|0x00|]
	ROBUST = PASIVE MASTER = SLAVE	Brak reakcji
DECREMENT	ROBUST = ACTIVE MASTER = MASTER	ERROR [|SA|TA|CM|0x00|]
	ROBUST = ACTIVE MASTER = SLAVE	DECACK[|NEW_VAL|]
	ROBUST = PASIVE MASTER = SLAVE	Brak reakcji

Tablica 5: Reakcja urządzenia na przesłane rozkazy

3. Tylko aktywny Master posiadający w danej chwili token ma możliwość inicjowania transmisji - generowania zapytania lub wykonywania innych operacji z użyciem tokena.

4. Aktywny węzeł Master (posiadający token) może nawiązywać transmisję z dowolnym aktywnym Slave.

5. Token jest przekazywany tylko między aktywnymi węzłami.

6. Węzeł Master może zostać wyłączony z sieci multimaster (tryb pasywny Master). Taki węzeł nigdy nie może przejąć tokena - staje się, zatem węzłem Slave, odpowiadającym jedynie na zapytania innych węzłów Master.

7. Token może być przejęty przez aktywny węzeł Master tylko na określony czas. Upłynięcie tego czasu uprawnia węzeł do dokończenia bieżącej operacji transmisji i wykonana czynności niezbędnych do przekazania tokena kolejnemu aktywnemu węzłowi Master.

8. Przekazywanie tokena odbywa się zgodnie z góry ustalonym porządkiem logicznym. Token przekazywany jest pod warunkiem potwierdzenia przez inny węzeł Master jego aktywności (zdolności przejęcia tokena), a po przesłaniu tokena następuje ponowne potwierdzenie poprawności transmisji.

9. W przypadku utraty tokena (np. w wyniku zakłócenia transmisji lub uszkodzenia węzła Master posiadającego token) węzeł Master o najwyższym priorytecie

(MAIN_MASTER) po wykryciu przekroczenia ustalonego czasu, po którym powinien otrzymać token, ma prawo do wygenerowania nowego tokena.

10. Węzeł może zostać całkowicie wyłączony z transmisji danych (np. na skutek uszkodzenia). Nie powoduje to zakłócenia lub ustania transmisji pomiędzy pozostałymi węzłami.