ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: INFORMATYKA z. 30 Nr kol. 1315

Joanna TOMASIK

SYMULACYJNY MODEL SIECI STEROWNIKÓW PRZEMYSŁOWYCH

Streszczenie. Projektowanie konfiguracji sieci sterowników przemysłowych wymaga dużego doświadczenia zespołu projektantów i jest czasochłonne. Ostre wymagania czasowe oraz wysoki koszt urządzeń powodują, że przy projektowaniu korzystne jest szacowanie czy sieć spełni warunki czasowe przy jak najniższej swojej cenie. W tym celu buduje się model kolejkowy sieci, a następnie rozwiązuje się ją — w tym przypadku, z uwagi na występujące w systemie w dużej liczbie zdarzenia synchronizujące — symulacją. W artykule przedstawiono pewien typ sieci sterowników i główne idee budowania jego modelu na potrzeby symulatora z pakietu QNAP.

A SIMULATIVE MODEL OF INDUSTRIAL CONTROLLERS NETWORK

Summary. Designing of controllers network is time-consuming. A group of people working on it should be very experienced because the network has to fulfill strict time requirements. A cost of the designed network should be also taken under consideration. At this point it may be useful to build a queueing network model of controllers network and solve it with any numerical method. For a large number of synchronization events occuring in the network a simulation is used. In this article one discusses an example of controllers network and some notions of designing its model in QNAP package.

MODELE SIMULATIF D'UN RESEAU DES CONTROLEURS INDUSTRIELS

Résumé. Elaboration d'un projet d'un réseau des contrôleurs demande du temps et de experience. Les constrains de temps de reponse dans la contrôle doivent être aussi respectés et le coût du réseau. Ces facteurs peuvent être pris en cosidération dans un modèle de files d'attente resolu par des méthodes mathematiques ou par simulation. Dans l'article on discute un simulateur de performance du réseau industriel des contrôleurs preparé avec l'aide de QNAP.

1. Wprowadzenie

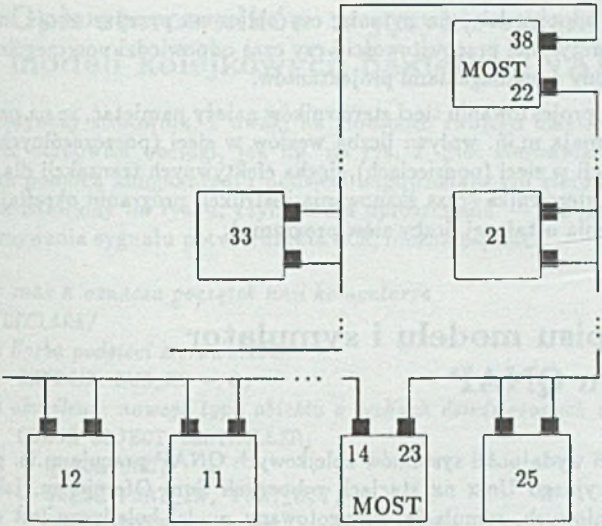
Sieci sterowników przemysłowych znajdują swoje zastosowanie w warunkach wymagających od nich często natychmiastowej reakcji na zaistniałe w kontrolowanym przez nie systemie zjawiska. Wymagania czasowe względem sieci sterowników wynikają z procesu technologicznego i zadaniem zespołu projektującego sieć sterowników i piszących dla nich oprogramowanie jest ich bezwzględne wypełnienie. Na rynku znajdują się sterowniki różnych firm, współpracujące z różnymi magistralami i protokołami, o różnych parametrach czasowych i różnych cenach, toteż zespół projektujący powinien wykazać się dużym doświadczeniem przy wyborze najbardziej optymalnej pod względem wydajności i kosztu konfiguracji. Istotną pomocą dla projektantów systemu sterującego może być przeanalizowanie rozważanych projektów za pomocą narzędzi teorii masowej obsługi, tj. zbudowanie adekwatnych modeli kolejkowych i oszacowanie ich parametrów wydajnościowych. Z uwagi na mnogość uzależnień czasowych oraz na dynamiczny charakter sieci sterowników najbardziej użyteczną do tego celu metodą jest symulacja. Można użyć symulatorów komercyjnych, znajdujących się w pakietach programowych, do szacowania wydajności modeli kolejkowych. Do liczenia parametrów opisanego niżej systemu został użyty symulator z francuskiego pakietu QNAP (Queueing Network Analysis Package) firmy Simulog [1]. Można w tym miejscu wspomnieć, że pakiet ten powstawał początkowo na francuskich uniwersytetach jako narzędzie do badań stricte akademickich, po czym został rozbudowany i rozwinięty do formy komercyjnej.

2. Przykładowa sieć sterowników

Rozważana sieć sterowników wchodzi w skład rzeczywistego urządzenia. Zbudowana jest z ogólnie dostępnych sterowników, wszelkie dane dla symulatora są danymi wynikającymi z katalogów firmowych i sposobu oprogramowania sterownikow.

Protokół dostępu do magistrali obowiązujący w sieci polega na przekazywaniu uprawnienia (ang. *token bus*). Sterownik, który dostaje żeton, jest uprawniony do wysyłania ramek na magistralę. Na czas transmisji uprawniony do niej sterownik blokuje żeton i zwalnia go dopiero po zakończeniu wysyłania informacji, przesyłając go do sterownika o adresie kolejnym w stosunku do własnego adresu. Należy pamiętać, że nawet gdy sterownik nie ma nic do wysłania, to przetrzymuje żeton przez ściśle określony czas, wprowadzając niemożliwe do likwidacji opóźnienie. Poza tym w ramce żetonu znajdować się może globalna informacja (ang. *global data*) dla wszystkich sterowników, na której potwierdzenie nadawca nie oczekuje, ale jej odczyt z ramki żetonu pochłania pewien czas.

Sieć składa się z kilku niezależnych podsieci (por. rys. 1). W każdej z nich krąży dedykowany jej żeton, a ewentualna wymiana informacji między podsieciami odbywa się za pomocą spinającego je mostu. Ramka, która ma zostać przesłana między dwoma podsieciami, niesie ze sobą informacje, przez które mosty musi przejść, aby dostać się do podsieci docelowej. Ramka oczekuje na moście do momentu otrzymania przez most żetonu z sieci tranzytowej (czy docelowej).



Rys. 1. Fragment przykładowej sieci sterowników
Fig. 1. An example of controllers network

3. Zasadnicze problemy przy projektowaniu sieci

Zgrupowanie sterowników w podsieciach ma na celu wyodrębnienie sterowników, które kontaktują się między sobą bardzo często i czas przepływu informacji między nimi jest wartością krytyczną. Poprzez wyodrębnienie podsieci wprowadzamy do niej jej „prywatny" żeton, o czasie przebiegu krótszym niż byłby czas obiegu „globalnego" żetonu w całej sieci. W ten sposób zapewniamy relatywnie szybszą wymianę informacji między sterownikami tej samej podsieci. Oczywiście odbywa się to kosztem względnego spowolnienia przesyłu między sterownikami znajdującymi się w różnych podsieciach, ponieważ konieczne jest wtedy dodatkowe oczekiwanie na żeton na moście (mostach). Wstępna analiza sieci powinna więc uwzględnić ewentualne podzielenie sieci sterowników na podsieci pod kątem priorytetów przepływającej między sterownikami informacji. Może się również zdarzyć, że podział na podsieci będzie wymuszony przez ograniczenia producenta co do możliwej liczby sterowników znajdujących się w jednej podsieci.

Kolejnym krokiem jest przeanalizowanie sposobu adresacji sterowników w obrębie każdej podsieci i jego wpływu na jej parametry wydajnościowe. W tym celu należy zbadać charakter informacji wymienianej między sterownikami. Jeżeli na przykład sterownik A przesyła do sterownika B głównie żądania przysyłania dużych bloków danych, to korzystne jest nie nadawać sterownikowi B adresu bezpośrednio wyższego od adresu sterownika A, lecz adres o tyle wyższy, aby w chwili, gdy B otrzyma żeton, miał już dane zebrane i gotowe do wysłania na magistralę.

Następnie należy odpowiedzieć na pytania, czy obliczona przepustowość magistrali nie przekroczy jej teoretycznej przepustowości i czy czas odpowiedzi poszczególnych sterowników będzie zgodny z wymaganiami projektantów.

Reasumując, przy projektowaniu sieci sterowników należy pamiętać, że na parametry wydajnościowe sieci mają m.in. wpływ: liczba węzłów w sieci (poszczególnych podsieciach), sposób adresacji w sieci (podsieciach), liczba efektywnych transakcji dla każdego sterownika, szybkość sterownika (czas skanowania instrukcji programu określający czas potrzebny do wykonania ustalonej liczby słów programu).

4. Język opisu modelu i symulator z pakietu QNAP

Pakiet do obliczeń wydajności systemów kolejkowych QNAP pracuje m.in. pod kontrolą systemu operacyjnego Unix na stacjach roboczych Sun. Oferuje on, jako jedną z możliwości obliczeniowych, symulację. Przygotowany model kolejkowy jest do niego wprowadzany w specyficznym dla QNAPa języku opisu sieci kolejek (mającego w pewnym stopniu cechy języka obiektowego). Po uruchomieniu zadania opis modelu jest poddawany obróbce przez translator i tłumaczony na wewnętrzną postać, zrozumiałą dla modułow obliczeniowych.

Opis modelu składa się z: części deklaratywnej rozpoczynającej się od /DECLARE/, w której znajdują opisy poszczególnych stanowisk obsługi (/STATION/) i definicje zmiennych różnych typów, z części sterującej rozpoczynającej się od /CONTROL/, w której znajdują się parametry wykonywania programu, takie jak maksymalny czas przebiegu symulacji TMAX, sposób zbierania śladu symulacji i inne oraz z części planującej przebieg eksperymentu rozpoczynającej się od /EXEC/, w której określana jest metoda obliczeń, sposób jej wykonywania (powtarzanie z różnymi parametrami), sposób zbierania informacji statystycznej z obiektów itd. Język opisu modelu pozwala na zamykanie pewnych części kodu w tzw. makrodefinicjach

$MACRO *nazwa makrodefinicji* [(*ewentualne argumenty*)]
⋮
$END

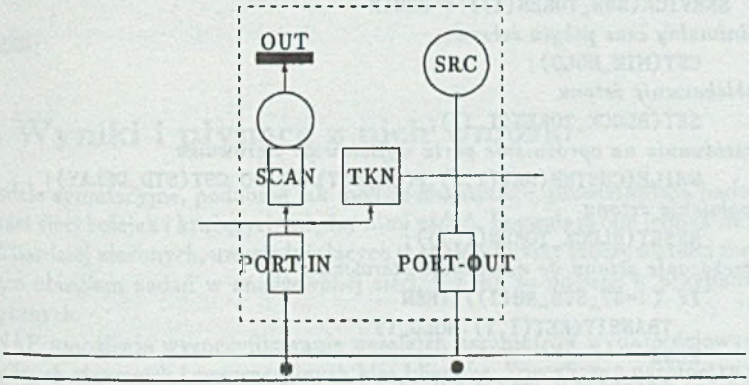
przy czym należy pamiętać, że w języku QNAPa nie ma możliwości zagnieżdżania makr.

Program symulacyjny umożliwia zbieranie danych statystycznych zarówno z poszczególnych zdefiniowanych kolejek, jak i śledzenie zmiennych zdefiniowanych przez użytkownika (ang. *watched variables*). Ocena dokładności może być przeprowadzana za pomocą estymacji przedziału ufności metodą replikacyjną albo regeneracyjną lub za pomocą analizy korelacji ciągów próbek metodą spektralną, która jest metodą domyślną, jeżeli w modelu nie określono inaczej.

5. Opis sterowników w języku opisu modeli kolejkowych pakietu QNAP

Pojedynczy sterownik, z uwagi na złożoność swojego działania, jest przedstawiany jako sieć stanowisk obsługi, jak np. na rys. 2. Sieć sterowników jest zatem modelowana za pomocą konglomeratu podsieci odpowiadającym sterownikom. Sterownik, jak ten przedstawiony na rys. 2, czyli — dla uproszczenia — bez uwzględniania wysyłania i otrzymywania sygnału potwierdzenia ACK, można zapisać:

```
& znak & oznacza początek linii komentarza
/DECLARE/
& liczba podsieci sterowników
  INTEGER SUB_NB = 8;
& określenie nowego typu obiektu o cechach dziedziczonych z obiektu QUEUE
  QUEUE OBJECT CONTROLLER;
    QUEUE SRC;
    QUEUE PORT_IN, PORT_OUT;
    QUEUE SCAN;
    QUEUE HOLD_T;
  END;
```



Rys. 2. Przykładowy sterownik przedstawiony jako sieć kolejek
Fig. 2. An example of controller as a queueing network

Ponieważ most jest modelowany za pomocą bezpośrednio połączonych dwóch obiektów typu CONTROLLER o zdeaktywowanych źródłach i należących do dwóch różnych podsieci, należy mieć możliwość nadania temu obiektowi atrybutu, aby rozróżnić czy wchodzi on w skład mostu, czy też pracuje jako „zwykły" sterownik. Dokonuje się tego poprzez definicję: CONTROLLER INTEGER BRIDGE;. Odwołania do atrybutu dokonuje się tak samo jak do obiektu składowego: <*obiekt*> . <*atrybut*>.

Następnie wyspecyfikować należy parametry poszczególnych składowych typu QUEUE obiektu CONTROLLER już za pomocą standardowych konstrukcji języka QNAPa, jak na przykład zapisane poniżej działanie stanowiska przetrzymującego żeton sterownika znajdującego się w podsieci I o numerze J w tej podsieci, przy czym TKN_INIT inicjuje liczbę klientów danej klasy na danym stanowisku w chwili rozpoczęcia symulacji. Całość opisu jest zawarta w makrodefinicji. Każdemu sterownikowi w każdej podsieci przyporządkowana jest flaga FLAG BLOCK_TOKEN(SUB_NB,CNTR_NB);, która odpowiedzialna jest za przetrzymanie żetonu aż do czasu całkowitego opróżnienia portu wyjściowego danego sterownika. Modelując porty wejściowy i wyjściowy należy pamiętać, że zapewniają one do pewnego stopnia współbieżność zacodzących transakcji, gdyż na portach mamy kilka ścieżek odczytu/zapisu danych (*Data Master Path*, *Data Slave Path*) oraz zleceń programowych (*Program Master Path*, *Program Slave Path*). Zajętość ścieżki nie ma wpływu na wydajność pozostałych ścieżek. Liczba ścieżek każdego typu zależy od konkretnego rodzaju sterownika. Gdy wszystkie ścieżki są zajęte, nadchodzące/wychodzące zgłoszenia są kolejkowane.

Oto przykładowy opis jednej ze składowych obiektu typu CONTROLLER:

```
$MACRO ST_TOKEN(I,J,TKN_INIT)
  /STATION/
     NAME = NET(I,J).HOLD_T;
& ustawienie warunku początkowego
     INIT(SUB_TOKEN(I)) = TKN_INIT;
     SERVICE(SUB_TOKEN(I)) = BEGIN
& minimalny czas pobytu żetonu
        CST(MIN_HOLD);
& zablokowanie żetonu
        SET(BLOCK_TOKEN(I,J));
& oczekiwanie na opróżnienie portu wyjściowego sterownika
        WHILE(CUSTNB(NET(I,J).PORT_OUT)>0) DO CST(STD_DELAY);
& zwolnienie żetonu
        RESET(BLOCK_TOKEN(I,J));
& przekazanie żetonu do następnego sterownika
        IF (J=ST_SUB_NB(I)) THEN
           TRANSIT(NET(I,1).HOLD_T)
        ELSE
           TRANSIT(I,J+1).HOLD_T);
     END;
$END
```

Polem, które musi być obligatoryjnie niepuste, jest pole NAME. Określenie typu obsługi i tranzycji może być w najprostszych przypadkach zapisane explicite, np.

```
SERVICE = EXP(1.0);
TRANSIT = NEXT_STATION, 0.7, OTHER_STATION, 0.3;
```

W przypadkach systemów złożonych istnieje konieczność napisania całego programu opisującego proces obsługi, w którym mogą występować wszystkie konstrukcje języka

(istotna jest możliwość sprawdzania warunków: ustawienia flag, poziomu zasobów itp.), przy czym ostatnią instrukcją procesu obsługi musi być instrukcja TRANSIT, gdyż jej wykonanie kończy proces, a następujące po niej instrukcje nie są wykonywane.

Ponieważ w sieci sterowników przesył ramek może zależeć od ich rodzaju, modelujący ich zachowanie klienci są dzieleni na klasy, które różnicują je z uwagi na czasy obsługi, priorytety obsługi, ścieżki przejścia czy ograniczoną liczebność. Tak na przykład dla poszczególnych podsieci sterowników w opisie modelu definiujemy dedykowaną klasę żetonu, zakładając jednocześnie, że jej populacja nie może przekroczyć jeden.

```
k zdefiniowanie żetonów dla każdej posieci
CLASS SUB_TOKEN(SUB_NB);
k zdefiniowanie ogólnej klasy bloków
CLASS BLOCK;
```

Klasa klientów BLOCK jest określona powyżej bardzo ogólnie. Można ją uszczegóławiać przez dodawanie atrybutów albo przez utworzenie nowego obiektu o cechach dziedziczonych z typu CLASS. W ten sposób klientowi można przypisać liczbę przesyłanych w ramach jednego bloku rejestrów, kto jest nadawcą, kto jest odbiorcą itd. poprzez dodawanie dodatkowych pól do standardowego obiektu klienta:

```
CLASS OBJECT BLOCK;
  INTEGER LENGTH;
    :
END;
```

6. Wyniki i płynące z nich wnioski

Modele symulacyjne, podobnie jak modele analityczne, przedstawiają badany obiekt w postaci sieci kolejek i krążących między nimi zadań. Pozwalają one jednak na tworzenie modeli bardziej złożonych, uwzględniających w sposób bliższy rzeczywistości mechanizmy rządzące obiegiem zadań w analizowanej sieci, niż ma to miejsce w przypadku modeli analitycznych.

QNAP umożliwia wyspecyfikowanie wszelkich parametrów wydajnościowych dla poszczególnych stanowisk i poszczególnych klas klientów. Specyficzne parametry potrzebne do oszacowania wydajności interesującej nas sieci, takie jak czas obiegu żetonu w poszczególnych posieciach czy oszacowanie roboczej przepustowości magistrali można w elementarny sposób obliczyć rozbudowując odpowiednio segment /EXEC/.

Czas symulacji dobiera się w taki sposób, aby móc uważać otrzymane wyniki za wartości w stanie ustalonym, tzn. po osiągnięciu zadanej długości przedziału ufności na określonym poziomie ufności. W każdym razie naczelną zasadą przy projektowaniu modelu jest troska o to, aby nie przeładować go szczegółami, które w niewielkim stopniu będą miały wpływ na dokładność modelu, natomiast w istotny sposób mogą wpłynąć na wydłużenie czasu trwania symulacji.

Przy analizowanej symulacji rozpatrywane były dwa przypadki:

1) średni, w którym rozpoczęcie obsługi żądania następuje co 1/2 czasu cyklu programu zajmującego całą pamięć programu sterownika, gdzie czas cyklu jest równy iloczynowi długości programu i czasu skanowania, stałego dla konkretnego typu sterownika,

2) skrajnie pesymistyczny, kiedy rozpoczęcie obsługi żądania następuje raz na jeden cykl programu zajmującego całą pamięć programu sterownika.

Wyniki otrzymane w tym drugim przypadku dla dobrze zaprojektowanej sieci nie powinny przekroczyć ok. 70% krańcowych parametrów wydajnościowych, gwarantowanych przez producenta sprzętu, trzeba bowiem pamiętać, że otrzymane wyniki mają charakter statystyczny, tymczasem sieć pracuje w sposób asynchroniczny, co powoduje dużą zmienność obciążenia sieci w czasie.

Obecnie w sieci Internet dostępny jest na prawach *public domain* rozbudowany, napisany w obiektowym języku C pakiet Smurph (por. [2]), który umożliwia bardzo szczegółowe modelowanie protokołów sieciowych warstw średnich. Pakiet ten posługuje się pojęciem automatu o skończonej liczbie stanów. Procesy biegnące na poszczególnych elementach (węzłach) modelowanej sieci powodują zmiany stanów tych elementów. Z uwagi za swoje daleko idące wyspecjalizowanie pakiet Smurph mógłby być również zalecany do modelowania sieci sterowników.

LITERATURA

[1] QNAP2, User's Manual & Reference Manual. Simulog, 1994.

[2] Gburzyński P.: Smurph, User's Manual & Reference Manual. Springer, 1995.

Recenzent: Prof. dr hab. inż. Andrzej Grzywak

Wpłynęło do Redakcji dnia 27 grudnia 1995

Abstract

Designing of controllers network is time-consuming. A group of people working on it should be very experienced because the network has to fulfill strict time requirements. A cost of the designed network should be also taken under consideration. At this point it may be useful to build a queueing network model of controllers network and solve it with any numerical method. For a large number of synchronization events occuring in the network a simulation is used. In this article one discusses an example of controllers network consisting of couple subnetworks is presented in the picture 1. Its model is built with the QNAP software package which main features are also described. A simply model of a controller considered as a queueing network is in the picture 2.