DOI: 10.6084/m9.figshare.11969013
УДК 004. 2

Зайцев Володимир Григорович
Доктор технічних наук, професор, професор кафедри системного програмування і спеціалізованих комп'ютерних систем, *orcid.org/0000-0001-9548-1959*
Національний технічний університет України «КПІ ім. Ігоря Сікорського», Київ
Цибаєв Євгеній Ігорович
Кандидат технічних наук, кафедра системного програмування і спеціалізованих комп'ютерних систем, *orcid.org/0000-0002-9115-2346*
Національний технічний університет України «КПІ ім. Ігоря Сікорського», Київ

МОДЕЛЬ ОЦІНКИ ЧАСОВИХ ХАРАКТЕРИСТИК У КОМП'ЮТЕРНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ З ВИКОРИСТАННЯМ СІТОК ПЕТРІ

Анотація. *Робота присвячена проблемі визначення часових характеристик задач у системах реального часу, успішність роботи яких залежить не тільки від їх логічної правильності, а й від часу, за який вони отримують результат. Визначення таких часових характеристик системи на стадії проєктування є досить складною проблемою. Її вирішення на сьогодні засновано на двох основних напрямах: теоретичних розрахунках, пов'язаних з отриманням так званих критеріїв здійсненності і моделюванням роботи системи на статистичних моделях систем масового обслуговування. Однак, як у першому, так і у другому випадках неможливо отримати гарантований результат, що суттєво ускладнює процес проєктування. Запропоновано метод оцінки часових характеристик задач в системах реального часу шляхом аналізу даних, отриманих моделюванням розподілу процесорного часу між задачами згідно обраних алгоритмів планувальника з використанням моделі сіток Петрі. Метод гарантує отримання часових характеристик задач при обранні конкретного типу процесора і планувальника, що потрібно для початку технічного проєктування системи реального часу.*

Ключові слова: модель; задача реального часу; сітка Петрі

Вступ

Системою реального часу (СРЧ) [1] називається система, в якій успішність роботи будь-якої програми залежить не тільки від її логічної правильності, а й від часу, за який вона отримує результат. Якщо часові обмеження не задоволені, то фіксується збій в роботі системи.

Розрізняють сильні (hard) і слабкі (soft) обмеження до вимог реального часу. Якщо запізнення програми призводить до повного порушення роботи системи керування, йдеться про сильне порушення у часі (жорсткі СРЧ). Якщо запізнення призводить тільки до тимчасової втрати продуктивності, то йдеться про слабке порушення у реальному часі.

Часто поняття "система реального часу" плутають з поняттям "швидка система". Це не зовсім правильно. Час затримки реакції СРЧ на подію уже не так і важливий, якщо цього часу достатньо для гарантування часових обмежень щодо певних додатків. Тобто система реального часу реагує у передбачений час на непередбачувані появи зовнішніх подій.

Не існує операційних систем жорсткого або м'якого реального часу. Поняття системи реального часу і операційної системи реального часу (ОСРЧ) часто змішується.

Система реального часу – це конкретна система, пов'язана з реальним об'єктом. Вона включає в себе необхідні апаратні і програмні засоби, операційну систему і прикладне програмне забезпечення. Операційна система реального часу – це тільки інструмент, що допомагає створити конкретну систему реального часу. Тому недоцільно говорити про системи жорсткого або м'якого реального часу. Можна говорити лише про те, чи можна за допомогою певної операційної системи побудувати конкретну систему реального часу [3].

Основні параметри завдань (задач) реального часу

Кожна задача РЧ характеризується такими часовими параметрами $[2]$: r – (Realize Time) – момент часу, коли виникає необхідність у передачі управління виконуваному завданню; d – (Absolute Deadline) – абсолютний крайній термін, момент часу, до якого задача повинна завершити роботу;

s – (Start Time) – момент часу, коли задача фактично починає виконуватись на процесорі; c – (Completion Time) – момент часу, коли задача закінчила роботу, опрацювавши подію (розв'язавши задачу); D – (Relative Time) – відносний крайній термін: $D = d - r$; E – (Execution Time) – час виконання задачі: $e = c - s$; R – (Response Time) – час відгуку: $R = c - r$.

Діаграма на рис. 1 ілюструє ці параметри.

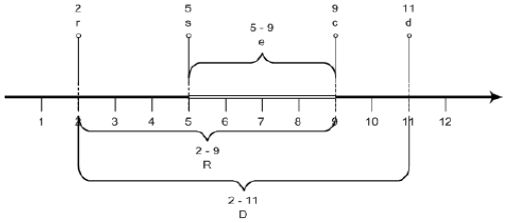

Рисунок 1 – Параметри завдань (задачі)

Наведена на цій діаграмі робота задачі має такі параметри: $r = 2$, $d = 11$, $s = 5$, $c = 9$, $D = 11 - 2 = 9$, $e = 9 - 5 = 4$, $R = 9 - 2 = 7$.

Актуальність дослідження

Згадані параметри визначаються таким чином. Терміни переходу задач у стан готовності, по суті, визначаються природою об'єкта керування. Граничні терміни (d, D) визначає розробник СРЧ, виходячи із властивостей об'єкта керування. Терміни виконання задач *e* визначаються архітектурою процесора, його тактовою частотою і конкретною реалізацією того чи іншого алгоритму. Для визначення останньої величини можна використовувати два підходи:

– підрахунок тактів процесора, необхідних для виконання тієї або іншої задачі. Такий підрахунок надзвичайно ускладнюється у разі, якщо процесор містить механізми типу конвеєрів і хешування;

– часи виконання безпосередньо вимірюються. Однак у випадку процесорів з конвеєрами і кешами такі вимірювання не дають гарантії, що буде виміряно саме максимальний час виконання того чи іншого коду. Крім того, системи, що використовують механізм підкачки сторінок, також є погано передбачуваними. Вважається, що такого роду механізми непридатні для систем реального часу.

Характер виникнення подій на керованому об'єкті відображається у такій класифікації завдань:

– періодичні завдання;

– спорадичні завдання (неперіодичні завдання з жорстким крайнім терміном);

– аперіодичні завдання (неперіодичні завдання з м'яким крайнім терміном).

Оскільки програмне забезпечення (ПЗ) СРЧ має взаємодіяти з процесами зовнішнього середовища і реагувати на цілий ряд подій (у тому числі і таких, що виникають одночасно), СРЧ має виконувати роботу в умовах багатозадачності. У цьому режимі декілька задач мають виконуватись одночасно. Якщо процесор один, то час процесора має поділятись між задачами шляхом утворення і обслуговування черг. Існує два основних принципи організації такого режиму роботи [4–6]:

– кооперативний,

– робота з витисканням.

Згідно першого принципу усі задачі самі приймають рішення на передачу управління, коли їм це слушно. Отже, задача, що виконується, сама приймає рішення про припинення виконання і за власної ініціативи передає управління наступній за чергою задачі.

При роботі з витисканням таким переключенням займається ОССРЧ згідно плану, розробленого планувальником відповідно до особливостей алгоритму його роботи. Алгоритм, у відповідності до якого ОС призначає пріоритети та передає управління від однієї задачі до іншої, називають дисципліною диспетчеризації.

Для дослідження коректності та ефективності дисциплін диспетчеризації, а також для обрання оптимальної дисципліни у кожному випадку реалізації конкретної системи досліджують роботу СРЧ на моделях, які враховують наявність у системі задач трьох основних типів:

– спорадичні – необхідність у їх виконанні виникає у випадкові моменти часу;

– періодичні – необхідність у їх виконанні виникає у моменти часу, що повторюються через рівні інтервали ;

– фонові – задачі, що виконуються за наявності вільного процесорного часу;

– комплекси задач різного типу (включаючи періодичні), виконання яких повторюється за деяким квазіперіодом.

Останні дослідження

Моделі виконання задач

У рамках пропонованого дослідження проаналізуємо процес створення моделі для оцінювання можливості виконання задач СРЧ в однопроцесорній системі.

У найпростішому випадку періодичних задач, кожна задача A_i характеризується двома параметрами:

$$A_i(e_i, T_i), \qquad (1)$$

де e_i – обсяг (час) обчислення; T_i – період.

Це означає, що моменти запуску задачі A_i чередуються з інтервалом T_i. Кожне виконання задачі A_i потребує e_i одиниць процесорного часу. Будемо вважати, що програмний додаток складається з низки незалежних задач $\{A_1, A_2, A_3, \ldots.A_n\}$. Під незалежністю тут

розуміється відсутність взаємозалежності між задачами щодо доступу до поділювальних ресурсів, що потребує синхронізації. Будемо вважати, що *j*-та активізація задачі A_i породжує завдання A_i. Різні завдання, що породжуються за рахунок однієї тієї ж самої задачі, будемо вважати однотипними завданнями. Кожне породження такого завдання збільшує кількість претендентів на ресурс процесорного часу. Завершення кожного із таких завдань, навпаки, зменшує кількість претендентів на процесорний час. Максимально можливий інтервал існування завдань типу A_i – R_i називають часом відгуку завдань типу A_i:

$$R_i = \max\{c_i - r_i\}. \quad (2)$$

У додатку з незалежними завданнями, кожне з активованих на даний час завдань може перебувати в одному з двох системних станів:

– завдання перебуває у стані виконання і володіє ресурсом процесора;

– у конкретний момент часу частина активованих завдань може бути у стані очікування процесорного часу для свого завершення.

Витискання завдань

У рамках створюваної моделі будемо допускати можливість витискання, тобто виконання будь-якої задачі може бути перервано для виконання іншого завдання. Через деякий час виконання перерваної задачі може бути поновлено. Таким чином виникає інтерференція – взаємний вплив активних завдань на час їх виконання.

Критичним сценарієм для задачі A_i називають таку сукупність системних подій, за якою існування чергового завдання типу A_i збільшується до максимального значення R_i.

У загальному випадку, як правило, часові обмеження на виконання завдань типу A_i задаються параметром D_i – відносним крайнім терміном виконання завдання A_i.

Вважається, що необхідно забезпечити виконання умови

$$R_i <= D_i. \quad (3)$$

У найпростіших моделях СРЧ, які засновані на циклічних задачах, ця умова зводиться до

$$C_i - r_i <= T_i. \quad (4)$$

Отже, кожне завдання A_i типу A_i має бути завершене до моменту виникнення чергового завдання цього типу.

Визначимо важливу характеристику u_i, що називається навантаженням на процесор від задачі типу A_i. Значення цього навантаження визначається як відношення потрібного обсягу обчислень (процесорного часу) до періоду активізації задачі,

$$u_i = e_i / T_i. \quad (5)$$

Загальною характеристикою додатка у цілому буде сумарне навантаження усіх задач – складових цього додатка на процесор

$$U = \sum u_j = \sum e_j / p_j \quad (6)$$

Планування

Модельні параметри R_i, T_i, D_i, u_i визначаються на попередніх етапах розроблення системи. На базі цих статистичних параметрів можна виконати апріорний аналіз можливості виконання задач системи в цілому. Але якщо у деякі моменти часу множина активних завдань пуста, то простоює, задарма витрачаючи процесорний час. В іншому випадку необхідно вирішувати, якому з наявних активних завдань слід надати процесор. Порядок надання процесорного часу для обслуговування активних завдань визначається алгоритмом планування і реалізується диспетчером.

Обрання алгоритму і визначення дисципліни планування є надскладною задачею і має бути вирішено на декількох етапах створення системи [1].

Можна вирізнити три таких основних етапи:

– перевірка можливості використання ряду дисциплін планування згідно відомих критеріїв;

– перевірка можливості і ефективності застосування обраних дисциплін планування на моделях СРЧ;

– перевірка працездатності обраного алгоритму планування на конкретному технічному обладнанні і ОС.

Алгоритми планування умовно можна поділити на такі групи [6]:

– статичні планувальники;

– динамічні планувальники із статичними пріоритетами;

– динамічні планувальники із динамічними пріоритетами.

Результатом роботи статичного планувальника є розклад виконання задач. Сукупність задач $\{A_i\}$ називається допускаючим планування деяким алгоритмом планування (АП), якщо цей АП завжди дає коректний розклад для цього набору задач. Наприклад, у випадку системи незалежних періодичних задач $\{A_i\}$, у складі яких можуть бути ще спорадичні та аперіодичні задачі, розкладом буде таблиця, у якій вказано, у якій момент часу яку задачу треба встановити на виконання. Такий розклад, як правило, складається на проміжок часу, що дорівнює гіперперіоду (найменшому загальному кратному періодів) усіх задач. Таким чином, планувальник періодично повторює послідовність

запуску включених до розкладу задач за умови виконання певних часових обмежень.

Планування зі статичними пріоритетами

Планування зі статичними пріоритетами поділяється на дві категорії [4; 5] :

– з витисканням завдань;
– без витискання завдань.

Пріоритети завданням призначаються попередньо і в процесі реалізації алгоритму не змінюються.

Динамічне планування з динамічними пріоритетами

Цей тип планувальників поділяється на два основні види : EDF та LLE [4].

При EDF плануванні (Erliest deadline first) пріоритет завдань призначається за принципом: в кожний момент часу найвищий пріоритет має та задача, у якої залишилось найменше часу до крайнього терміну. Модифікації алгоритму бувають:

– з витисканням завдань;
– без витискання завдань.

При LLF (Least lexity first) пріоритет завдань призначається за принципом: в кожен момент часу найвищий пріоритет має задача з найменшим запасом часу. Є така теорема:

Для того щоб система $\{A_i\}$ незалежних періодичних задач, що витискаються, була планована EDF планувальником, необхідно, щоб коефіцієнт використання процесорного часу системи не перевищував одиниці, тобто

$$u <= 1. \qquad (7)$$

Таку умову ще називають критерієм здійсненності планування або просто критерієм здійсненності.

Динамічне планування із статичними пріоритетами

Є два основних розповсюджених способи надання пріоритетів [5].

1. RMS (Rate monotonic scheduling) – чим менше період задачі, тим вищий у неї пріоритет. Тобто, чим частіше задача переходить у стан готовності, тим вищий у неї пріоритет.

2. DMS (Deadline monotonic scheduling). Чим менший відносний крайній термін виконання задачі, тим вищий у неї пріоритет.

Для системи $\{Ai\}$ незалежних періодичних задач, що витискаються, з періодами, рівними відносним крайнім термінам і таких, що плануються на одному процесорі, запропоновано дещо інший критерій здійсненності:

$$U_{RM} = n(2^{1/n} - 1). \qquad (8)$$

Мета дослідження

Завершуючи короткий огляд різних алгоритмів планування, можна зробити висновок про те, що хоча в деяких випадках і можна сформулювати критерії здійсненності, все ж таки, покладаючись тільки на них, навряд чи можна говорити про створення оптимальної СРЧ, здатної відповідати усім поставленим вимогам. Деякі варіанти реалізації можуть мати надлишкові можливості і процесор буде мати багато вільного часу, а у деяких варіантах навпаки, можуть виникати напружені ситуації. Тому на другій стадії розробки СРЧ бажано промоделювати її роботу з різними типами планувальників і обрати найбільш оптимальний варіант для подальшої реалізації.

Запропонована модель

Загальна схема дослідження може бути представлена, як на рис. 2. Вона складається з генератора потоку задач, алгоритмів планування і диспетчеризації та моделі виконання задач у часі.

Генератор потоку задач генерує послідовності задач в межах періоду (гіперперіоду) надходження задач. Планувальник відповідно до обраного алгоритму планування визначає послідовності (номери) задач, яким слід надати процесорний час і своєю чергою отримує від моделі виконання задач інформацію про моменти надання процесорного часу кожній задачі, проценти та моменти завершення виконання, а також про ступінь завантаженості процесора за період виконання T_i.

У моделі виконання задач попередньо має бути задана інформація про час виконання кожної із задач e_i, а також забезпечена можливість надання обраній на виконання задачі повного часу виконання або надання такого часу окремими порціями (режим з витисканням) за час періоду T_i.

В роботі запропоновано модель виконання задач побудувати за допомогою апарату сіток Петрі.

Рисунок 2 – Загальна схема моделювання

Структура сіток Петрі являє собою сукупність позицій та переходів і складається з двох типів вузлів: позицій (кружечків) та планок (рисок) – переходів. Орієнтовані дуги поєднують позиції і переходи. Дуга, що направлена від позиції P_i до переходу τ_j визначає позицію, що являє собою вхід у перехід. Вихідна позиція вказується дугою від переходу до позиції. Входи та виходи можуть бути кратними, що визначається на схемі або кількістю дуг, або цифрою k на дузі. Кожна позиція може мати певну кількість маркерів, які можуть рухатись з позиції на позицію при виконанні (запуску) переходів [7; 8].

Власне сітка Петрі – це орієнтований дводольний мультиграф G. Формально граф $G = (V, Z)$, де $V = \{v_1, v_2, ...v_s\}$ – множина вершин; $Z = (z_1, z_2, ...z_n)$ – комплект орієнтованих дуг $z_i = \{v_i, v_k\}$, де v_i, v_k є V. Множина V розбита на дві непересічні множини P і θ таких, що $V = PU\theta$, $P \bigcap \theta = 0$.

При аналітичному способі визначення сітки вона може бути представлена як $S = (P, \theta, F, H, \mu_0)$, де додатково визначені вхідна функція F та вихідна H. Через $F(\tau_j)$ позначається множина вхідних позицій, а через $H(\tau_j)$ – множина вихідних позицій переходу (τ_j), μ_0 – початкове маркування (розподіл маркерів по позиціях). Маркування сітки S – це присвоювання певної кількості маркерів, що знаходяться в позиціях (присвоєні) P_i. Маркування μ може бути також представлене n-мірним вектором $\mu = (\mu_0, \mu_1, ...\mu_n)$, де μ_j – розподіл фішок серед всіх позицій P_i у кожному конкретному стані сітки. Тобто, маркована сітка Петрі – сукупність структури сітки $S = (P, \theta, F, H, \mu)$ та маркування μ, або $S = (P, \theta, F, H, \mu)$.

Перехід називають дозволеним, якщо кожна з його вхідних позицій має кількість маркерів не менше ніж кількість дуг із позицій в перехід. Кратні маркери необхідні для кратних вхідних дуг. Маркери (фішки) у вхідній позиції, які дозволяють перехід, називаються дозволяючими фішками. Наприклад, якщо позиції P_1 та P_2 – входи для переходу τ_j, то перехід τ_j дозволено, якщо P_1 та P_2 мають хоча б по одній фішці.

Запуски переходів відповідають деяким подіям, що відбуваються у процесах, що моделюються за допомогою сітки Петрі. Після запуску переходів, що відповідають події, дозволені переходи спрацьовують, а не дозволені – не спрацьовують. У загальному випадку сітка Петрі змінює своє маркування μ на нове μ^1. Тобто

$$\mu_0 \to \mu_1 . \qquad (9)$$

Якщо переходи не запускаються, зберігається попереднє маркування сітки.

Відмітимо, що запустити можна тільки дозволені переходи, і тому кількість фішок у кожній позиції завжди залишається невід'ємною. Запуск переходу ніколи не видалить фішку, яка відсутня у вхідній позиції. Якщо деяка вхідна позиція переходу не має достатню кількість фішок, то перехід не дозволяється і не може бути запущений.

Як приклад, на рис. 3 наведена сітка Петрі, що задана графічним способом.

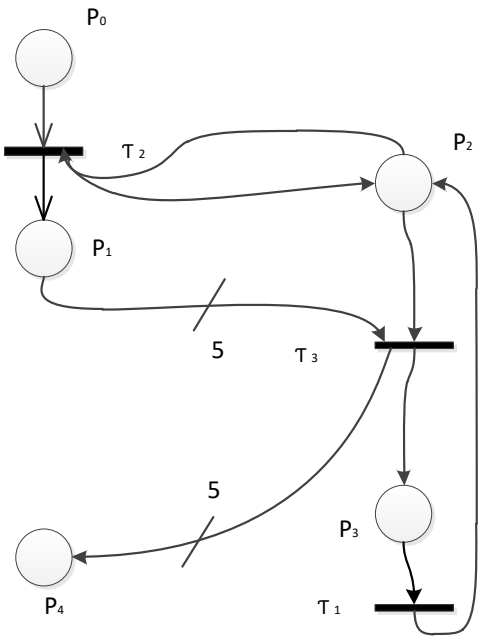

Рисунок 3 – Сітка Петрі

При аналітичному способі окрім множини позицій P_1 та переходів θ також мають бути задані вхідна H та вихідна F функції [8; 9].

Через $F(\tau_j)$ позначається множина вхідних позицій, а через $H(\tau_j)$ – множина вихідних позицій переходу τ_j. Розглянемо це на прикладі. На рис. 3 наведено сітку Петрі. Згідно сітки на цьому рисунку:

$$F(\tau_1) = (P_3); \; H(\tau_1) = (P_2);$$
$$F(\tau_2) = (P_0, P_2); \; H(\tau_2) = (P_1, P_2);$$
$$F(\tau_3) = (P_1, P_1, P_1, P_1, P_1, P_3);$$
$$H(\tau_3) = (P_3, P_4, P_4, P_4, P_4, P_4).$$

Аналітичне представлення сітки у матричній формі може бути представлене так

$$S = (P, \theta, M^-, M^+, \mu_0), \qquad (10)$$

де M^- та M^+ – матриці вхідних та вихідних інцеденцій розміром $m \times n$, де m – кількість переходів; n – кількість позицій.

Кожний елемент матриці M^- дорівнює кратності дуг, що входить в i-й перехід з j-ї позиції, а елемент матриці M^+ дорівнює кратності дуг з i-го переходу в позицію j.

Матриця інцедентності сітки Петрі M визначається як

$$M = M^+ - M^- . \qquad (11)$$

Для наведеної на рис. 3 сітки початкове маркування $M=[5\,0\,0\,1\,0]$, оскільки $[P_0] = 5$, $[P_1] = 0$, $[P_2] = 0$, $[P_3] = 1$, $[P_4] = 0$, де $[P_i]$ – кількість маркерів в позиції P_i, а матриці M^-, M^+ та M представлені, як у таблицях табл. 1 – 3.

Таблиця 1– Матриця M^-

	P0	P1	P2	P3	P4
τ_1	0	0	0	1	0
τ_2	1	0	1	0	0
τ_3	0	5	1	0	0

Таблиця 2 – Матриця M^+

	P0	P1	P2	P3	P4
τ_1	0	0	1	0	0
τ_2	0	1	1	0	0
τ_3	0	0	0	1	5

Таблиця 3– Матриця M

	P0	P1	P2	P3	P4
τ_1	0	0	1	- 1	0
τ_2	- 1	1	0	0	0
τ_3	0	- 5	- 1	1	5

Керування сіткою виконується шляхом запуску переходів. Перехід може бути запущено лише у тому випадку, якщо він дозволений. Як відомо [10] нове маркування після запуску переходу може бути отримане після обчислення нового маркування як

$$\mu_{i+1} = \mu_i + B_j \times M, \qquad (12)$$

де B_j – одиничний вектор – рядок переходу j, усі компоненти якого дорівнюють нулю, за винятком компоненти, що відповідає номеру j та дорівнює одиниці. Якщо перехід спрацьовує, отримується нове маркування μ_{i+1}, в якому не може бути від'ємної кількості фішок у позиціях. Якщо це має місце при обчисленні, то це свідчить про те, що перехід не спрацьовує і маркування залишається незмінним (попереднім), тобто μ_i.

Обчислимо значення величин $B_j \times M$ щодо запуску переходів τ_1, τ_2, τ_3.

Ці величини є константами, які додаються поелементно до кожного елемента попереднього маркування $[P_i]$ після кожного спрацювання переходу, змінюючи попереднє маркування на нове. Якщо перехід не спрацьовує, ця операція не виконується і маркування не змінюється.

Отже:

$$\tau_1 : B_1 \times M = [1\,0\,0] \times M = [0\,0\,1\,-1\,0];$$

$$\tau_2 : B_2 \times M = [0\,1\,0] \times M = [-1\,1\,0\,0\,0];$$

$$\tau_3 : B_3 \times M = [0\,0\,1] \times M = [0\,-5\,-1\,1\,5].$$

Введемо поняття такту запуску сітки, що представлена на рис. 3.

Під цим будемо розуміти спробу запустити кожного разу (такту) послідовність переходів τ_1, τ_2, τ_3. На кожному такті, як це видно з табл. 4, частина з кожних трьох переходів не спрацьовує. Якщо перехід не спрацьовує, він не змінює маркування сітки, і маркування залишається попереднім (як до запуску переходу, який не спрацював).

Аналізуючи зміну маркування сітки у процесі виконання тактів запуску, можна стверджувати, що у кожному такті один маркер з позиції P_0 переходить у позицію P_1, де накопичується N маркерів ($N = 5$ у цьому випадку), і накопичені 5 маркерів після виконання переходу τ_3 на п'ятому такті переходять у позицію P_4. Окрім того маркер з позиції P_2 переходить у позицію P_4, сигналізуючи про завершення передачі N маркерів.

Якщо пов'язати перехід маркера з позиції P_0 з деякою умовною одиницею часу Δt, то можна вважати, що перехід деякої кількості N маркерів буде відповідати T умовним одиницям часу, де

$$T = N\Delta t, \qquad (13)$$

а накопиченню k_i маркерів у позиції P_1 до переходу у позицію P_4 буде відповідати інтервалу $k_i\Delta t$ умовних одиниць часу.

З аналізу наведеної на рис. 3 сітки Петрі видно, що переходи маркерів з позиції P_0 у позицію P_1 і з позиції P_1 у P_4 виконуються за тактами, що складаються з послідовного збудження сукупності переходів τ_1, τ_2, τ_3.

Якщо вважати, що величина умовного часу періоду задач РЧ дорівнює T, а час виконання задачі i в умовних одиницях часу процесора складає $k_i\Delta t$, то можна вважати, що

$$e_i = k_i\Delta t . \qquad (14)$$

І сітка Петрі, що наведена на рис. 3, відповідає моделі виконання однієї задачі за умови, що $e_i = T$. При цьому перехід маркера з позиції P_3 у позицію P_2 відповідає моменту надання задачі процесора, а повернення його у позицію P_3 завершенню виконання задачі.

При розробці загальної моделі роботи n задач слід передбачити можливість переривання

і поновлення видачі задачі процесорного часу. Оскільки функціонування моделі полягає у послідовному збудженні трьох переходів за кожним тактом, ця задача може бути розв'язана шляхом припинення їх видачі на черговому такті і повернення до цього при поверненні задачі процесора на чергових тактах до її повного виконання.

Враховуючи сказане, загальну модель видачі процесорного часу декільком задачам можна представити, як на рис. 4. З цього рисунку видно, що деяка сукупність з n задач, які представлені моделями сітки Петрі, еквівалентними моделі сітки на рис. 3, об'єднані в одну модель шляхом включення позиції P_0 у кожну з моделей – складових. При цьому початкова кількість маркерів у позиції P0 має відповідати T умовним одиницям часу. Надання процесора конкретній задачі буде відповідати збудженню її власних переходів на відповідному такті. Припиненню виконання задачі і постановці на виконання іншої задачі буде відповідати припинення збудження переходів першої і збудження переходів другої, при виконанні чергового такту роботи моделі.

Для того щоб розрізняти позиції (за винятком позиції P_0 , яка є загальною для всіх моделей – складових) та переходи моделей окремих задач, запропоновано ввести подвійну індексацію, де перший індекс додатково вказує на номер задачі, тобто P_{i1} , P_{i2} і т. п. Тобто запропоновано простий спосіб постановки та зняття з виконання конкретної задачі – шляхом запуску відповідних до задачі переходів при виконанні чергового такту.

На завершення опису моделі слід вирішити ще одне питання – як виміряти невикористаний час процесора за час періоду. Цей час можна врахувати, якщо додати в загальну модель ще одну складову (псевдозадачу – A_0), час виконання якої дорівнює одиниці $(k_0 = 1)$. Тоді кількість маркерів, що будуть накопичені у позиції P_{04} за весь проміжок часу T , і буде цією величиною в умовних одиницях.

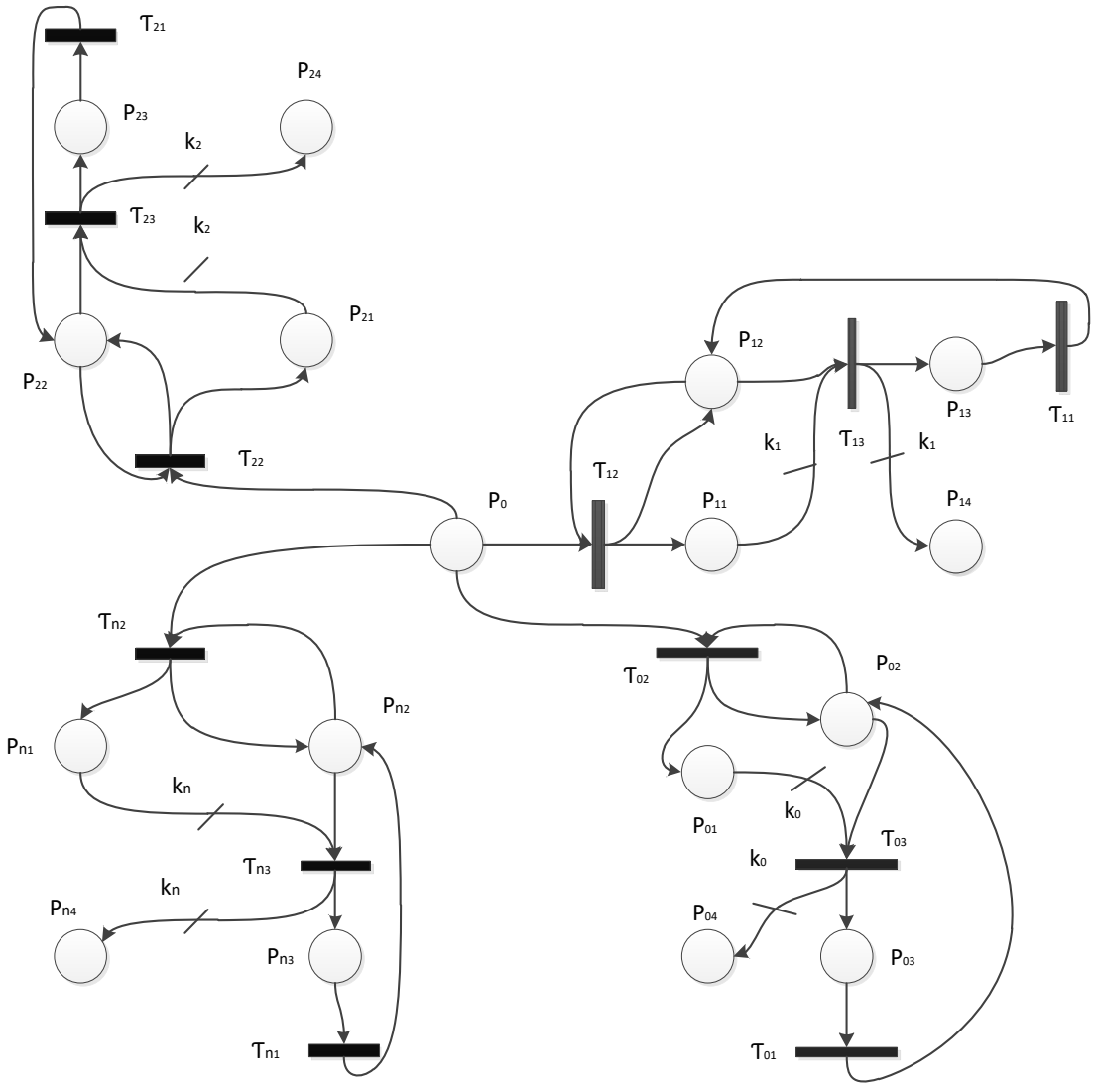

Рисунок 4 – Модель виконання задач

Таблиця 4 – ***Виконання сітки Петрі***

Номер такту	Перехід	Початкове маркування	Результат запуску	Спрацювання переходу	Нове маркування
1	τ_1 τ_2 τ_3	5 0 0 1 0 5 0 1 0 0 4 1 1 0 0	5 0 1 0 0 4 1 1 0 0 4 -4 0 1 5	+ + -	5 0 1 0 0 4 1 1 0 0 4 1 1 0 0
2	τ_1 τ_2 τ_3	4 1 1 0 0 4 1 1 0 0 3 2 1 0 0	4 1 2 -1 0 3 2 1 0 0 3 -3 0 1 5	- + -	4 1 1 0 0 3 2 1 0 0 3 2 1 0 0
3	τ_1 τ_2 τ_3	3 2 1 0 0 3 2 1 0 0 2 3 1 0 0	3 2 2 -1 0 2 3 1 0 0 2 -2 0 1 5	- + -	3 2 1 0 0 2 3 1 0 0 2 3 1 0 0
4	τ_1 τ_2 τ_3	2 3 1 0 0 2 3 1 0 0 1 4 1 0 0	2 3 2 -1 0 1 4 1 0 0 1 -1 0 1 5	- + -	2 3 1 0 0 1 4 1 0 0 1 4 1 0 0
5	τ_1 τ_2 τ_3	1 4 1 0 0 1 4 1 0 0 0 5 1 0 0	1 4 2 -1 0 0 5 1 0 0 0 0 0 1 5	- + +	1 4 1 0 0 0 5 1 0 0 0 0 0 1 5

Розглянемо роботу цієї моделі на прикладі. Нехай циклічно виконуються задачі A_1, A_2, A_3 з такими умовними часовими параметрами:

$$A_1: r_1, e_1, d_1 = 0, 3, 10;$$

$$A_2: r_2, e_2, d_2 = 2, 6, 14;$$

$$A_3: r_3, e_3, d_3 = 4, 4, 12;$$

час циклу $T = 14$. Оберемо планувальник, що використовує алгоритм планування EDF з витисканням.

Визначимо матрицю M кожної із задач $A_i - M_i$, де $i = 1, 2, 3$; i – номер задачі.

У загальному випадку матриця M_i буде мати вигляд, як табл. 5.

Таблиця 5. Матриця M

	P_0	P_1	P_2	P_3	P_4
τ_1	0	0	1	- 1	0
τ_2	- 1	1	0	0	0
τ_3	0	$-k_i$	- 1	1	k_i

Визначимо початкові маркування сіток кожної із задач за умови, що на виконання вона запускається першою:

$$A_1 = [1\,4\,0\,0\,1\,0],$$

$$A_2 = [1\,4\,0\,0\,1\,0],$$

$$A_3 = [1\,4\,0\,0\,1\,0].$$

При цьому слід враховувати те, що позиція P_0 є загальною для всіх моделей задач і тому кількість маркерів у цій позиції буде змінювати кожна із попередньо запущених задач, тобто кожна обрана на виконання задача змінює кількість маркерів у позиції P_0 і впливає на маркування всіх моделей – складових системи.

За аналогією з (12) обчислимо значення констант, які додаються до попереднього маркування сіток моделі після спрацювання відповідного переходу сітки. Константи відповідають таким виразам:

$$\tau_{i1}: [0\,0\,1 - 1\,0],$$

$$\tau_{i2}: [-1\,1\,0\,0\,0],$$

$$\tau_{i3}: [0\ -\kappa_i\ -1\,1\,\kappa_i].$$

Результати роботи моделі наведено у табл. 6.

З аналізу таблиці видно, що першою на виконання обрано задачу A_1. Її було запущено на виконання на початку першого кванту, тобто о нульовій годині умовного часу. Про це свідчить перехід маркера з позиції P_{13} у позицію P_{12}. Задача була виконана повністю, про що свідчить перехід маркера з позиції P_{12} у позицію P_{13}. На виконання задачі витрачено 3 одиниці умовного часу. Наступною на виконання була обрана задача A_2. Вона виконувалась один квант часу, після чого на виконання була обрана задача A_3. Маркер у позиції P_{23} залишився, що свідчить про те, що задачу не завершено. Задача A_3 працювала до завершення на 8 кванті, після чого виконання задачі A_2 було продовжено, її було завершено на 13 кванті (на 13 умовній одиниці часу).

14-й квант залишився невичерпаним і була запущена задача A_0 – підрахунку невикористаного процесорного часу, що становив 1 одиницю умовного часу.

Таблиця 6 – ***Виконання моделі сітки Петрі трьох задач***

Номер такту	Перехід	Початкове маркування	Результат запуску	Спрацювання переходу	Нове маркування
1	τ_{11} τ_{12} τ_{13}	1 4 0 0 1 0 1 4 0 1 0 0 1 3 1 1 0 0	1 4 0 1 0 0 1 3 1 1 0 0 1 3 -2 0 1 3	+ + -	1 4 0 1 0 0 1 3 1 1 0 0 1 3 1 1 0 0
2	τ_{11} τ_{12} τ_{13}	1 3 1 1 0 0 1 3 1 1 0 0 1 2 2 1 0 0	1 2 1 2 -1 0 3 2 1 0 0 1 2 -1 0 1 3	- + -	1 3 1 1 0 0 1 2 2 1 0 0 1 2 2 1 0 0
3	τ_{11} τ_{12} τ_{13}	1 2 2 1 0 0 1 2 2 1 0 0 1 1 3 1 0 0	1 2 2 2 -1 0 1 1 3 1 0 0 1 1 0 0 1 3	- + +	1 2 2 1 0 0 1 1 3 1 0 0 1 1 0 0 1 3
4	τ_{21} τ_{22} τ_{23}	1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0	1 1 0 1 0 0 1 0 1 1 0 0 1 0 -5 0 1 6	+ + -	1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 1 0 0
5	τ_{31} τ_{32} τ_{33}	1 0 0 0 1 0 1 0 0 1 0 0 9 1 1 0 0	1 4 2 -1 0 9 1 1 0 0 9 -3 0 1 4	+ + -	1 0 0 1 0 0 9 1 1 0 0 9 1 1 0 0
6	τ_{31} τ_{32} τ_{33}	9 1 1 0 0 9 1 1 0 0 8 2 1 0 0	9 1 2 -1 0 8 2 1 0 0 8 -2 0 1 4	- + -	9 1 1 0 0 8 2 1 0 0 8 2 1 0 0
7	τ_{31} τ_{32} τ_{33}	8 2 1 0 0 8 2 1 0 0 7 3 1 0 0	8 2 2 -1 0 7 3 1 0 0 7 -1 0 1 4	- + -	8 2 1 0 0 7 3 1 0 0 7 3 1 0 0
8	τ_{31} τ_{32} τ_{33}	7 3 1 0 0 7 3 1 0 0 6 4 1 0 0	7 3 2 -1 0 6 4 1 0 0 6 0 0 1 4	- + +	7 3 1 0 0 6 4 1 0 0 6 0 0 1 4
9	τ_{21} τ_{22} τ_{23}	6 1 1 0 0 6 1 1 0 0 5 2 1 0 0	6 1 2 -1 0 5 2 1 0 0 5 -4 0 1 6	- + -	6 1 1 0 0 5 2 1 0 0 5 2 1 0 0
10	τ_{21} τ_{22} τ_{23}	5 2 1 0 0 5 2 1 0 0 4 3 1 0 0	5 2 2 -1 0 4 3 1 0 0 4 -3 0 1 6	- + -	5 2 1 0 0 4 3 1 0 0 4 3 1 0 0
11	τ_{21} τ_{22} τ_{23}	4 3 1 0 0 4 3 1 0 0 3 4 1 0 0	4 3 2 -1 0 3 4 1 0 0 3 -2 0 1 6	- + -	4 3 1 0 0 3 4 1 0 0 3 4 1 0 0
12	τ_{21} τ_{22} τ_{23}	3 4 1 0 0 3 4 1 0 0 2 5 1 0 0	3 4 2 -1 0 2 5 1 0 0 2 -2 0 1 6	- + -	3 4 1 0 0 2 5 1 0 0 2 5 1 0 0
13	τ_{21} τ_{22} τ_{23}	2 5 1 0 0 2 5 1 0 0 1 6 1 0 0	2 5 2 -1 0 1 6 1 0 0 1 0 0 1 6	- + +	2 5 1 0 0 1 6 1 0 0 1 0 0 1 6
14	τ_{01} τ_{02} τ_{03}	1 0 0 1 0 1 0 1 0 0 0 1 1 0 0	1 0 1 0 0 0 1 1 0 0 0 0 1 0 1	+ + +	1 0 1 0 0 0 1 1 0 0 0 0 0 1 1

Всі задачі завершено у межах циклу, який дорівнює $T=14$ умовних одиниць. Обмеження за часом виконання задач виконані:

$$d_1 = 3 < 10;$$

$$d_2 = 13 < 14;$$

$$d_3 = 8 < 12.$$

На рис. 5 результати вищенаведеного моделювання роботи трьох задач представлені у графічному вигляді. Для підтвердження важливості перевірки правильності вибору алгоритму планування було виконано моделювання процесу виконання тих самих трьох задач, використовуючи алгоритм EDF – планування без витіснення.

Отримані результати моделювання наведено на рис. 6 у графічній формі.

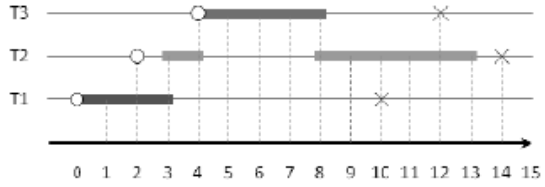

Рисунок 5 – Розклад, складений EDF – планувальником з витісненням

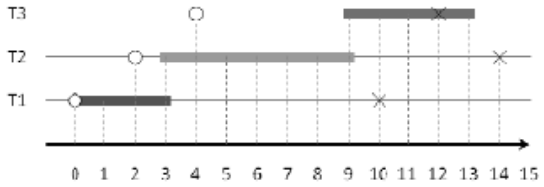

Рисунок 6 – Розклад, складений EDF – планувальником без витіснення

З такого порівняння видно, що хоча всі задачі і виконуються у межах часу тривалості періоду T, вимоги терміну виконання задачі A_3 не витримуються: $d_3 = 12 < 13$, хоча в обох випадках критерій здійсненності (6) виконується.

Висновки

На завершення слід зробити висновок, що запропонована модель є досить ефективним засобом моделювання процесів виконання задач у системах реального часу, проста в реалізації, не потребує значних резервів машинного часу і гарантує отримання необхідної інформації для вибору процесора необхідної продуктивності і типу планувальника.

Список літератури

1 *Зайцев В.Г., Цибаєв Є.І. Комп'ютерні системи реального часу: навчальний посібник // Національний технічний ун-т України "КПІ ім. Ігоря Сікорського". – К.: – 2019. Електронний ресурс КПІ ім. Ігоря Сікорського:* https://ela.kpi.ua

2 *Системы реального времени: конспект лекций / Владим. гос. ун-т; сост. А. С. Голубев. – Владимир: Изд-во Владим. гос. ун-та, 2010 – 127 с.*

3 *Операційні системи: навчальний посібник / Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського":В.Г. Зайцев, І.П. Дробязко. – Київ, 2019. Електронний ресурс КПІ ім. Ігоря Сікорського: https://ela.kpi.ua/handle/123456789/29600*

4 *Baker T. Multiprocessors EDF and Deadline Monotonic Shedulability Analysis // Proceeding of 24 IEEE Real – Time Systems Symposium, 2003, p. 120 – 129.*

5 *Andersen B., Daruah S., Jonson J. Static – Priority Shedulings on Microprocessors // Proccedings of 22 IEEE Real – Time System Symposium, 2001, 193 – 202.*

6 *Ferrari A.D. Real – Time Scheduling Algorithms // Dr. Dobb's Jornal. 1994, N12, p. 60 – 66.*

7 *Сети Петри. Электронній ресурс: http://www.hpc-education.ru/files/lectures/2011/ershov/ershov_2011_lectures05.pdf*

8 *Питерсон Дж. Теория сетей Петри и моделирование систем. – М. : Мир, 1984. – 264 с.*

9 *Фальк В.Н. Введение в сети Петри и моделирование систем. Учебное пособие. Москва. 2009. Электронный ресурс: https://studfiles.net/preview/1529418*

10 *Стеценко І.В. Система імітаційного моделювання засобами сіток Петрі / І.В. Стеценко, О.В. Бойко / Математичні машини і системи, 2009 – № 1. – С.117 – 124.*

Стаття надійшла до редколегії 03.10.2019

Зайцев Владимир Григорьевич
Доктор технических наук, профессор, профессор кафедры системного программирования и специализированных компьютерных систем, *orcid.org/0000-0001-9548-1959*
Национальный технический университет Украины «КПИ им. Игоря Сикорского», Киев
Цибаев Евгений Игоревич
Кандидат технических наук кафедры системного программирования и специализированных компьютерных систем, *orcid.org/0000-0002-9115-2346*
Национальный технический университет Украины «КПИ им. Игоря Сикорского», Киев

МОДЕЛЬ ОЦЕНКИ ВРЕМЕННЫХ ХАРАКТЕРИСТИК В КОМПЬЮТЕРНЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ С ИСПОЛЬЗОВАНИЕМ СЕТЕЙ ПЕТРИ

Аннотация. *Работа посвящена проблеме определения временных характеристик задач в системах реального времени, успешность работы которых зависит не только от их логической правильности, но и от времени, за которое они получают результат. Определение таких временных характеристик системы на стадии проектирования является*

достаточно сложной проблемой. Ее решение сегодня основано на двух основных направлениях: теоретических расчетах, связанных с получением так называемых критериев осуществимости, и моделированием работы системы на статистических моделях систем массового обслуживания. Однако, как в первом, так и во втором случаях невозможно получить гарантированный результат, что существенно усложняет процесс проектирования. Предложен метод оценки временных характеристик задач в системах реального времени путем анализа данных, полученных моделированием распределения процессорного времени между задачами согласно избранных алгоритмов планировщика с использованием модели сетей Петри. Метод гарантирует получение временных характеристик задач при выборе конкретного типа процессора и планировщика, что нужно для начала технического проектирования системы реального времени.

Ключевые слова: модель; задача реального времени; сетка Петри

Zaitsev Vladimir
DSc (Eng.), Professor. Professor of Department of System Programming and Specialized Computer Systems, Kyiv Polytechnic Institute, *orcid.org/0000-0001-9548-1959*
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv
Tsybaev Evgeniy
PhD (Eng.), Department of System Programming and Specialized Computer Systems, Kyiv Polytechnic Institute, *orcid.org/0000-0002-9115-2346*
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv

A MODEL FOR ESTIMATING THE TIMING OF REAL-TIME COMPUTER SYSTEMS USING PETRI NETS

Abstract. *The work is devoted to the problem of determining the temporal characteristics of tasks in real-time systems, the success of which depends not only on their logical correctness but also on the time for which they receive the result. Determining such temporal characteristics of the system at the design stage is a rather difficult problem. Its solution is currently based on two main areas: theoretical calculations related to obtaining the so-called feasibility criteria and modeling the operation of the system on statistical models of queues. However, in both the first and second cases, it is not possible to obtain a guaranteed result, which significantly complicates the design process. The method of estimation of time characteristics of tasks in real time systems by data analysis is offered obtained by simulating the allocation of processor time between tasks according to the selected scheduler algorithms using a Petri net model. The method guarantees the timing of tasks when selecting a specific type of processor and float, which is required to begin the technical design of the real-time system.*

Keywords: model; real-time task; Petri net

References

1. Zaitsev, V.G., Tsybaev, E.I. (2019). Real-time computer systems: a textbook. National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute": Kyiv. Igor Sikorsky KPI Electronic Resource: https://ela.kpi.ua

2. Golubev, A.S. (2010). Real-time systems: lecture notes. Vladim. state un-t; comp. Vladimir, Russia: Publishing houseVladim, state University, 127.

3. Zaitsev, V.G., Drobyazko, I.P. (2019). Operating Systems: Tutorial. Real-time computer systems: a textbook / National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute": Kyiv. Igor Sikorsky KPI Electronic Resource: https://ela.kpi.ua/handle/123456789/29600

4. Baker, T. (2003). Multiprocessors EDF and Deadline Monotonic Shedulability Analysis. Proceeding of 24 IEEE Real – Time Systems Symposium, p. 120-129.

5. Andersen, B., Daruah, S., Jonson, J. (2001). Static – Priority Shedulings on Microprocessors. Proccedings of 22 IEEE Real – Time System Symposium, p. 193–202.

6. Ferrari, A.D. (1994). Real – Time Scheduling Algorithms. Dr. Dobb's Jornal, 12, 60–66.

7. Petri nets. Electronic resource (2011): http://www.hpc-education.ru/files/lectures/2011/ershov/ershov_2011_lectures05.pdf

8. Peterson, J. (1984). Petri net theory and system modeling. M.: Mir, 264.

9. Falk, V.N. (2009). Introduction to Petri Nets and System Modeling. Tutorial. Moscow. Electronic resource: https://studfiles.net/preview/1529418

10. Stetsenko, O.V. Boyko, I.V. (2009). System of simulation modeling by means of Petri nets (2009) / Mathematical Machines and Systems, 1, 117-124.

Посилання на публікацію

APA *Zaitsev, Vladimir, & Tsybaev, Evgeniy, (2019). A model for estimating the timing of real-time computer systems using Petri nets. Management of Development of Complex Systems, 40, 76 – 86, [in Ukrainian]; dx.doi.org\10.6084/m9.figshare.11969013.*

ДСТУ *Зайцев В.Г. Модель оцінки часових характеристик у комп'ютерних системах реального часу з використанням сіток Петрі [Текст] / В.Г. Зайцев, Є.І. Цибаєв // Управління розвитком складних систем. – 2019. – № 40. – С. 76 – 86; dx.doi.org\10.6084/m9.figshare.11969013.*