Математичне та імітаційне моделювання СМО із застосуванням середовища Matlab

Вступ

До основних задач дослідження систем відносять:

- аналіз – вивчення елементів і властивостей функціонування систем;
- синтез – вибір структури і параметрів за заданими властивостями системи.

Часто зустрічаються системи, які мають елементи стохастичності або функціонують як стохастичні. Стохастичний характер можуть носити елементи і процеси, що протікають у системі, як внутрішні так і зовнішні. Найчастіше вплив зовнішнього середовища можна оцінити лише статистичними методами.

На практиці часто виникають задачі, пов'язані із багаторазовим використанням процесів, які описуються однотипними моделми, коли замовлення у систему надходять у випадкові моменти часу і обробляються за допомогою наявних у системі каналів обслуговування. Замовлення – це об'єкт (сутність), який необхідно обслужити і який є носієм замовлення, наприклад: покупці, що хочуть придбати товар, механізми або прилади, які вимагають ремонту, потяги або літаки, які необхідно прийняти на станції і т. д. Такі процеси отримали назву процесів обслуговування, а системи – систем масового обслуговування (СМО). В теорії СМО під вимогою розуміють сам запит на обслуговування. Наведемо декілька прикладів систем, що розглядаються як СМО.

Приклади.

1. Довідкова телефонна служба. Дзвінки абонентів утворюють потік замовлень, а працівники служби – сервіс або прилад обслуговування.
2. Автозаправочна станція. Автомобілі представляють потік замовлень на заправку паливом, а заправочні колонки – сервіси обслуговування.
3. Робота ЕОМ у режимі розділеного часу. Програми – замовлення на обробку, процесор – обробляючий прилад.
4. Організація роботи ремонтної майстерні. Механізм може знаходитись у двох станах: бути справним або вийти з ладу. Коли механізм знахотится у певному стані, на нього діє потік з інтенсивністю λ, який переводить його в інший стан (наприклад, виникнення поломки), і потік з інтенсивністю μ, що повертає його у попередній стан (ремонт). Обидва потоки є Пуассонівськими та незалежними. Поломки – замовлення на ремонт, ремонтний цех – сервіси обслуговування.
5. Обробка деталей на конвейері. Деталі – замовлення на обробку, обробляючий пристрій – сервіс.

Необхідно зазначити, що моделі теорії масового обслуговування є основою багатьох галузей досліджень: теорія надійності, планування виробничих потужностей, теорія розвитку популяцій, логістика та менеджмент.

Розробкою методів дослідження СМО займається спеціальний розділ теорії ймовірностей – теорія масового обслуговування. Теорія масового обслуговування часто називається теорією черг (*англ.* Queue Theorie).

Предметом теорії масового обслуговування є встановлення залежності між факторами, що визначають функціональні можливості системи, і ефективністю її функціонування. В більшості випадків всі параметри СМО є випадковими величинами або функціями, тому такі системи відносяться до стохастичних системам.

Основи теорії масового обслуговування були закладені А. К. Ерлангом на початку 20-го століття, який працював у Копенгагенскій телефонній компанії. У 1909 році він опублікував працю «Теорія ймовірностей і телефонні переговори», де вирішив ряд задач для систем масового обслуговування з відмовами. Термін "теорія масового обслуговування" був введений О. Я. Хінчиним, який вважається її засновником. Значний внесок у розвиток цієї теорії внесли такі математики, як Б. В. Гніденко, А. М. Колмогоров, О. С. Вентцель.

1. Основні поняття, класифікація та параметри СМО

1.1. Основні поняття СМО

Система масового обслуговування – математичний (абстрактний) об'єкт, що складається з одного або декількох приладів (каналів) обслуговування (П) замовлень (З), що надходять у систему через накопичувач (Н) і утворюють чергу (Ч) або без накопичувача безпосередньо на обслуговування (рис. 1).

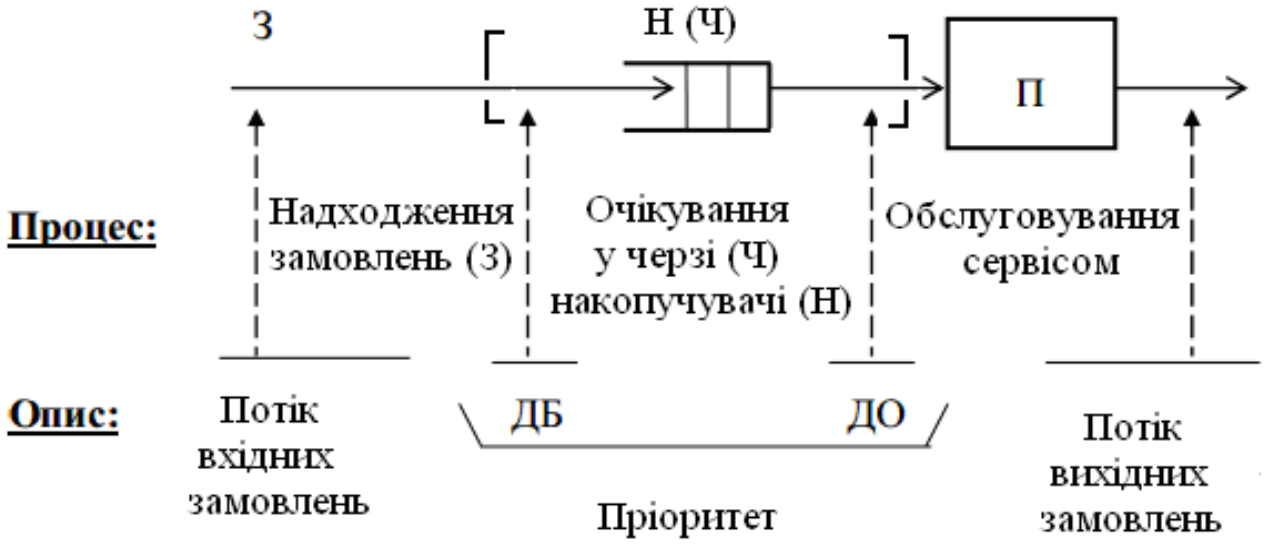


Рис.1. Схема процесів СМО

Замовлення (вимога, запит, виклик, клієнт) – це об'єкт, що надходить у систему і вимагає обслуговування. Сукупність замовлень, розподілених у часі, утворює потік замовлень.

Потоки замовлень поділяють на вхідний та вихідний. Потоки замовлень характеризуються певними властивостями.

Обслуговуючий прилад (пристрій) або сервіс – елемент СМО, призначенням якого є обслуговування замовлень. В кожний момент часу прилад може обслуговувати лише одне замовлення.

Сам процес обслуговування розглядається як затримка замовлення в обслуговуючому приладі.

Час обслуговування – час затримки замовлення у приладі.

Накопичувач (буфер) – сукупність місць для очікування замовлень на обслуговування. Система може мати накопичувач, а може не мати. Якщо накопичувач відсутній і всі сервіси зайняті, то замовлення, що надходить у систему, відхиляється. Якщо накопичувач існує і є вільні місця, то замовлення стає у чергу (перебуває у системі).

Замовлення, що знаходиться у системі, може перебувати у двох станах:

- в стані очікування (у накопичувачі), якщо всі прилади зайняті;
- в стані обслуговування (у приладі).

Кількість місць у накопичувачі визначає його ємність і визначає довжину **черги**.

Дисципліна буферизації (ДБ) – правило занесення замовлень у накопичувач (буфер).

Дисципліна обслуговування (ДО) – правило вибору замовлень із черги для обслуговування приладом.

1.2. Типи потоків замовлень

Потік являє собою сукупність об'єктів, що сприйнмаються як єдине ціле, та існує як процес на деякому часовому інтервалі і вимірюється в абсолютних одиницях за певний період. Параметри потоку – це параметри, що характеризують процес. Основними параметрами потоку є: початкова і кінцева точки руху, траєкторія руху, довжина шляху (міра траєкторії), швидкість і час руху, проміжні точки, інтенсивність.

Потоки класифікуються за різними ознаками:

1) по відношенню до даної системи:

внутрішні потоки – циркулюють усередині системи;

зовнішні потоки – знаходяться поза системою;

2) за ступенем неперервності:

- неперервні потоки – в кожен момент часу по траєкторії потоку переміщується певна кількість об'єктів;
- дискретні потоки – утворюються об'єктами, що переміщуються з інтервалами;

3) за ступенем стабільності:

- стабільні потоки – характеризуються сталістю значень параметрів протягом певного проміжку часу;
- нестабільні потоки – характеризуються змінами параметрів потоку.

Також розрізняють за характером переміщення, періодичністю, синхроністю і таке інше.

Розглянемо типи потоків, які виділяють в теорії масового обслуговування.

За характером замовлень потоки поділяють на:

- ***однорідний потік*** – це потік у якому всі замовлення мають однакові властивості.

- ***неоднорідний потік*** – це потік, коли властивості замовлень відрізняються (наприклад, пріоритетами).

Приклади.

1. Нехай на станцію надходять потяги. Якщо ми їх не розрізняємо, то можна вважати, що маємо справу з однорідним потоком. Якщо ми їх розрізняємо на пасажирські та вантажні, то потік буде неоднорідним, але його можна розділити на два однорідних потоки.

2. Бібліотека університету обслуговує студентів та викладачів, причому, викладачі можуть обслуговуватись в першу чергу. Тоді ми маємо справу з неоднорідним потоком, у якому замовлення відрізняються пріоритетами.

Також потоки розрізняються за ступенем регулярності, а саме розподілом часу між надходженням замовлень. Розглянемо потік однорідних подій. Нехай t_1, t_2,… – моменти часу появи подій, причому $t_{i+1} > t_i$, $i \geq 1$.

Регулярним (детермінованим) потоком називається потік, у якому замовлення надходять послідовно одне за одним через однакові проміжки часу: $\tau = t_{i+1} - t_i - const$.

Стохастичним (ймовірностним) називають потік, що характеризується випадковим характером параметрів, які в кожен момент часу приймають певне значення з певним ступенем ймовірності. Можна сказати, що це послідовність випадкових величин. Стохастичний потік характеризується середнім інтервалом часу між появою подій (математичним сподіванням) та середнім відхиленням від середньої довжини інтервалу (середньоквадратичним відхиленням).

Рекурентним називають стохастичний потік у якому всі функції розподілу інтервалів часу між замовленнями співпадають (підпорядковані одному закону розподілу $P_i(\tau) = P(\tau_i < \tau),\ i > 1$). Рекурентний потік представляє таку послідовність подій, у якого всі часові інтервали між подіями підпорядковані однаковому закону розподілу, що дозволяє дослідити один інтервал і розповсюдити його статистичні характеристики на всі інші. Потік називається рекурентним в силу того, що при його моделюванні використовують рекурентні процедури: спочатку інтервал часу τ_1, потім τ_2 і т.д.

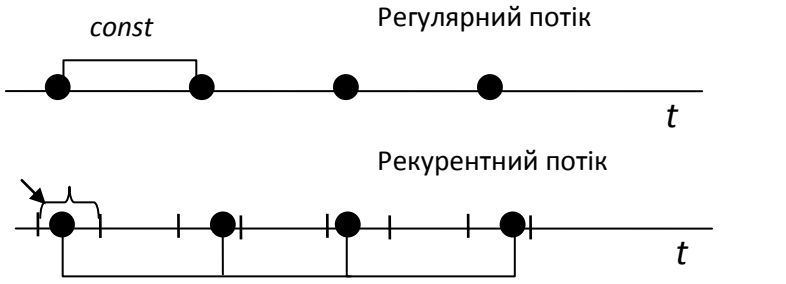


Рис. 2. Приклади графічного представлення регулярного та стохастичного потоків

Стаціонарний ординарний рекурентний потік із запізненням називають потоком ***Пальма***.

Приклади.

1. Потяги метрополітену надходять через однакові проміжки часу і утворюють регулярний потік. Прихід тролейбусів або автобусів на зупинку, яка не є кінцевою, вже не буде утворювати регулярний потік. Проміжки часи між їх приходами будуть становти визначений інтервал плюс-мінус похибку, яка обумовлена наявністю світлофорів та іншого транспорту на маршруті.
2. Час між метрологічним оглядом приладу чітко визначений, сукупність оглядів буде утворювати регулярний потік.
3. Нехай деякий пристрій неперервно працює до моменту відмови, після чого миттєво заміщується. Строк безвідмовної роботи є випадковим, але окремі пристрої виходять з ладу незалежно одна від одної. Тоді потік відмов представляє собою потік Пальма.
4. Група літаків летить у бойовому порядку з однаковою швидкістю, дотримуючись «строю», – певної відстані від іншого літака, що летить попереду. Зрозуміло, що ця відстань дотримується з певною похибкою в силу різих причин. Моменти часу перетину заданого рубежу будуть утворювати потік Пальма. Якщо кожний літак буде намагатись дотримуватись певної відстані від ведучого літака, а не від сусіднього, то тоді це вже не буде потік Пальма.

Потоком ***без наслідків*** називають потік у якому ймовірності появи подій на часових проміжках, що не перетинаються, незалежні (незалежність процесу від передісторії). Потік без наслідків характеризується тим, що кількість подій на кожному проміжку часу не залежить від кількості подій на інших проміжках.

Потоком ***з обмеженими наслідками*** (потік Пальма) називається потік, для якого інтервали часу між подіями незалежні.

Ординарним називають потік, для якого ймовірність появи двох і більше подій на малому проміжку часу ($\tau \to 0$) значно менше, ніж поява однієї події на цьому проміжку.

Потік називають ***стаціонарним***, якщо всі його ймовірностні характеристики не змінюються з часом. Для стаціонарного потоку ймовірність

появи певної кількості подій на інтервалі часу залежить тільки від довжини цього інтервалу і не залежить від його розташування на часовій осі.

Стаціонарний ординарний потік без наслідків називається ***Пуасонівським***, де ймовірність появи n подій на проміжку часу τ з інтенсивністю λ (середньою кількістю подій, що з'являється за одиницю часу):

$$P_n(\tau)=\frac{(\lambda\cdot\tau)^n\cdot e^{-\lambda\cdot\tau}}{n!}$$

і має наступні харакеристики:

- $f(t)=\lambda\cdot e^{-\lambda\cdot t}$ – щільність розподілу;
- $m_x=\lambda\cdot\tau$ – математичне сподівання;
- $D_x=\lambda\cdot\tau$ – дисперсія.

Для розподілу Пуассона математичне сподівання дорівнює дисперсії.

Самі проміжки часу мають експоненціальний закон розподілу:

$$F(t)=1-e^{-\lambda t},\ t>0,$$

де щільність розподілу: $f(t)=dF(t)/dt=\lambda e^{-\lambda t}$.

Якщо у простому потоці розглядати лише кожну k-ту подію, то отримаємо ***потік Ерланга*** k-того порядку. Наприклад, якшо розглядати кожну третю подію, то будемо мати потік Ерланга 3-го порядку (рис. 3).

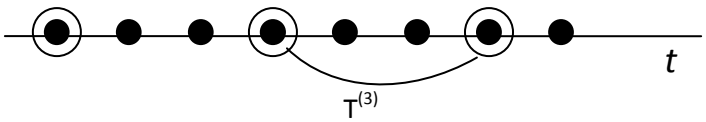


Рис. 3. Перетворення простого потоку у потік Ерланга 3-го порядку

Просіювання подій призводить до того, що між подіями з’являються наслідки, детермінація яких збільшується із зростанням k. При збільшенні значення k, події все рівномірніше розташовуються на часовій вісі, їх розкид зменшується, а регулярність збілшується. Основою цього є той факт, що сума випадкових величин є величиною невипадковою (центральна гранична теорема), причому із збільшенням кількості доданків результат буде більш передбачуваним.

Щільність ймовірності розподілу інтервалів між випадковими величинами в потоці Ерланга k-го порядку становить:

$$f(T)=\frac{\lambda\cdot(\lambda\cdot t)^{k-1}}{(k-1)!}e^{-\lambda t}.$$

$\lambda_k=\lambda/k$ – інтенсивність потоку Ерланга k-го порядку, де λ — інтенсивність простійшого потоку Пуасона, а λ_k – інтенсивність просіяного k разів потоку, (в k разів менше). Параметри закону Ерланга обчислюються за формулами:

$$M_k = 1/\lambda_k \,,\ \sigma_k = 1/\left(\lambda_k \sqrt{k}\right).$$

При $k \to \infty$ події відбуваються строго розміряно, так як $\sigma \to 0$. Можна зробити висновок, що порядок потоку Ерланга – є мірою наслідків потоку.

Для стохастичних потоків існує два типи перетворень: їх сума та розріджування.

Сумарним називається потік, що утворюється шляхом додавання випадкових подій вихідних потоків, що сумуються. Необхідно відзначити дві основні властивості сумарних потоків.

1) Для сумарного потоку існує гранична теорема: сума незалежних, ординарних і стаціонарних випадкових потоків збігається до пуассонівського потоку при необмеженому збільшенні числа доданків. Інтенсивності потоків, що додаються, повинні при цьому бути однакової розмірності. Ця теорема аналогічна центральній граничній теоремі для випадкових величин.

2) При додаванні *n* як стаціонарних, так і нестаціонарних потоків інтенсивність сумарного потоку дорівнює сумі інтенсивностей потоків-доданків і виражається співвідношенням:

$$\lambda(t) = \sum_{i=1}^{n} \lambda_i(t).$$

Розрідженим (рідким) потоком називається потік, що виходить з вихідного потоку шляхом випадкового видалення з нього подій з постійною ймовірністю *q*. Іншими словами, подія вихідного потоку лишається в розрідженому потоці з ймовірністю *p = 1-q*.

Слід розрізняти операцію детермінованого "просіювання", за допомогою якої виходять потоки Ерланга, і операцію випадкового розрідження. Розріджені потоки маєть наступні основні властивості:

1) Для рідкого потоку існує гранична теорема: якщо послідовно розріджувати вихідний стаціонарний ординарний потік Пальма з ймовірністю *p1, p2, ..., pn*, $n \to \infty$, то багато разів розріджений потік прямує до Пуассонівського.

2) Інтенсивність розрідженого потоку дорівнює інтенсивності вихідного потоку, помноженій на ймовірність збереження події в потоці: $\lambda_p = p\lambda$.

Приклади.

1. Середній інтервал часу між послідовним надходженням подій становить 0,5 години, тоді інтенсивність потоку становить $\lambda = 2$ *(зам/год.)*

2. Розглянемо виклики швидкої допомого, які надходять випадковим чином (всі виклики вважаються однаковими і утворюють однорідні події). За довгий час спостережень можна сказати, що у середньому надходить вісім викликів за добу. Отже інтенсивність потоку буде становити: $\lambda = 8/24 (од./год.)$ (значення інтенсивності можна вважати сталим, а потік стаціонарним). Середня довжина інтервалу часу між окремими викликами становить $\mu = 1/\lambda = 3(год.)$ і на кожному інтервалі лише один виклик (ординарність). Виклики надходять

незалежно один від одного (події без наслідків). Отже ми маємо справу з Пуасонівським потоком.

Алгоритм моделювання процесу на проміжку часу $T=100(год.)$ наведений на рис.4.

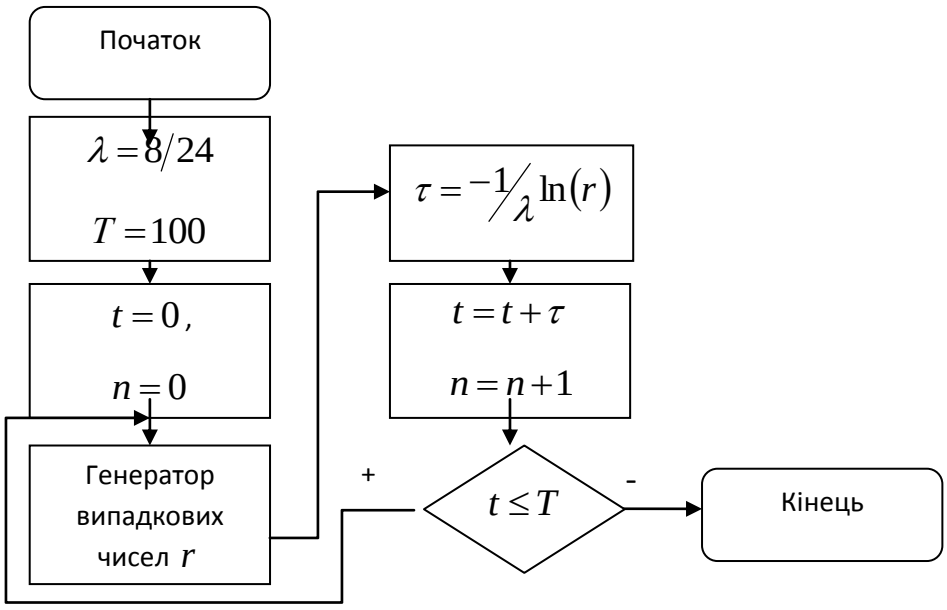


Рис. 4. Алгоритм моделювання надходження викликів швидкої

3. На комп'ютер надходять програми з інтенсивністю $\lambda = 1{,}5\left(завд./сек.\right)$ (потік завдань є простим). Знайти ймовірність, що за 2 секунди:

а) не надійде жодної програми;

б) надійде рівно одна программа;

в) надійде хоча б одна программа.

Розв'язок. Випадкова величина X – кількість програм в секунду має закон розподілу Пуассона з параметром $\lambda \cdot \tau = 1{,}5 \cdot 2 = 3.$

а) Ймовірність, що не надійде жодної програми ($m=0$): $P(X=0) = e^{-3} \approx 0{,}498.$

б) Ймовірність, що надійде рівно одна программа ($m=1$): $P(X=1) = \frac{3^1}{1!} e^{-3} \approx 0{,}1494.$

в) Ймовірність, що надійде хоча б одна программа ($m \ge 1$): $P(X \ge 1) = 1 - P(X=0) = 1 - 0{,}498 = 0{,}502.$

4. В результаті спостережень визначені статистичні оцінки інтервалів часу між подіями:

- $m_t = 2\ хв$ – середня довжина інтервалу часу між подіями;
- $\sigma_t = 0{,}9\ хв$ – середньоквадратичне відхилення інтервалу часу.

Необхідно підібрати потік Ерлангу, з приблизно подібними характеристиками, визначити його інтенсивність та порядок.

Розв'язок. Інтенсивність потоку Ерланга – це величина, обернена до середньої довжини інтервалу часу між подіями: $\lambda_k = 1/M_k = 1/2 = 0{,}5\ подій/год.$

Знайдемо порядок потоку Ерланга: $k = 1/(\sigma \cdot \lambda_k)^2 = 1/(0{,}9 \cdot 0{,}5)^2 \approx 4{,}9$.

В якості значення порядку вибираємо найближче ціле значення: $k = 5$.

Отже потік можна наблизити потоком Ерланга 5-го порядку із щільністю розподілу: $f_5(t) = \frac{(5 \cdot 0{,}5)^5}{4!} t^4 e^{-5 \cdot 0{,}5t} = 4{,}1t^4 e^{-2{,}5t}$, $t > 0$/

5. Нехай проводились спостереження за роботою ламп освітлення вуличних ліхтарів на протязі довгого періоду часу $T = 100$ років. Відомо, що середній час безвідмовної роботи лампи становить $M_k = 1{,}5$ року із середньоквадратичним відхиленням $\sigma_k = 0{,}5$ року.

Розв'язок. Так як $M_k \neq \sigma_k$, то $k \neq 1$ і ми маємо справу з потоком Ерланга з інтенсивністю $\lambda_k = 1/M_k = 1/1{,}5 = 0{,}67$. Отже за рік перегоряє 0,67 лампочки або 67 ламп за 100 років. Обчислимо порядок потоку: $k = 1/(\sigma \cdot \lambda_k)^2 \approx 9$.

Обчислимо інтенсивність відповідного Пуассонівського потоку:

$$\lambda = \lambda_k \cdot k = 0{,}67 \cdot 9 \approx 6.$$

Вважається, що потік подій є заданим, якщо:

а) задано скінченновимірний розподіл послідовності проміжків часу між подіями $\{\tau_i, i \geq 1\}$ або для $\forall n \in N, \cdots n \geq 1$ задано розподіл випадкового вектору $(\tau_1, \ldots, \tau_n)$;

б) задано скінченновимірний розподіл процессу $\nu(t)$ або $\forall n \in N, \cdots n \geq 1$ та набору не відємних чисел $\tau_1, \ldots, \tau_n$ задано розподіл випадкового вектору $(\nu_1, \ldots, \nu_n)$.

1.3. Структура СМО

Структура СМО визначається заданим потоком замовлень, кількістю обслуговуючих приладів, тривалістю обслуговування, кількістю місць очікування на обслуговування. Структура СМО та її основні елементи прийнято представляти схемою, наведеною на рис. 5.

Генератор (джерело замовлень) – об'єкт, що породжує замовлення. Генератор характеризується множиною можливих замовлень та визначає характеристики вхідного потоку. Генератор може мати скінченну або нескінченну потужність.

В залежності від характеру замовлень розрізняють замкнені та розімкнуті СМО. У розімкнених системах множина замовлень, що виробляється генератором, вважається нескінченною і поведінка генератора не залежить від

стану системи. Для замкнених систем характерним є скінченна кількість замовлень, яка циркулює у системі. Оброблені замовлення можуть знову повернутись у систему. Поведінка джерела у замкнених СМО є функцією від її стану.

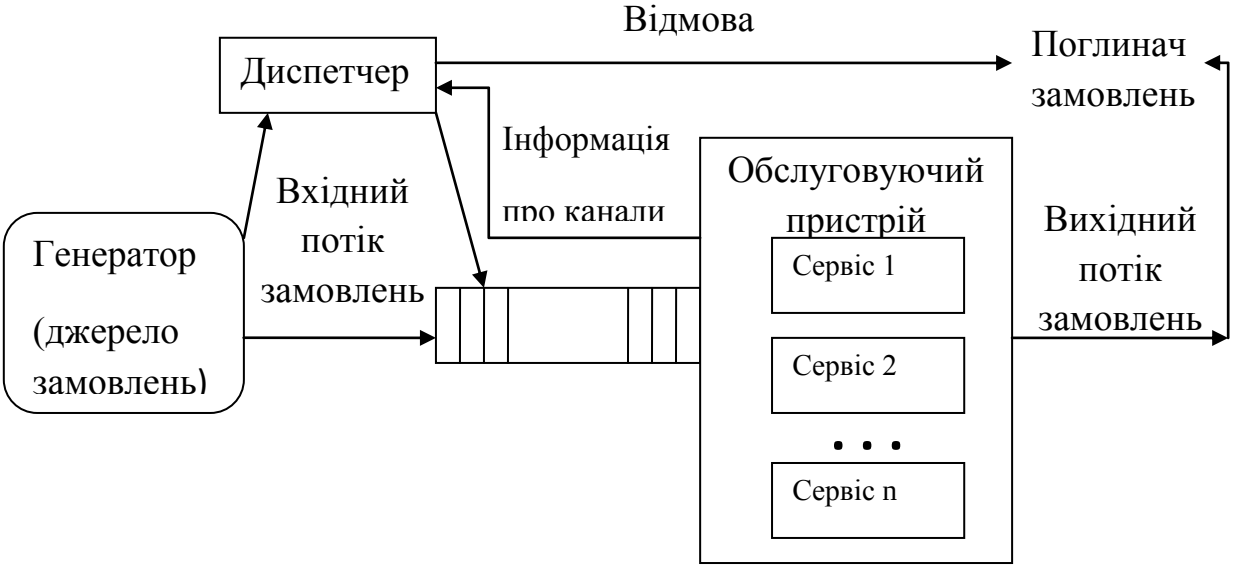


Рис. 5. Структурна схема СМО

Процес генерації замовлень в загальному випадку є випадковим і розглядається як потік однорідних подій, що виникають через випадкові проміжки часу, які можуть бути підпорядковані різним законам розподілу. В теорії масового обслуговування найбільше поширення має простий потік замовлень, у якому часові інтервали між сусідніми подіями описуються експоненціальним законом розподілу:

$$f(\tau) = \lambda \cdot e^{-\lambda\tau}, \text{ де } \tau = t_{i+1} - t_i .$$

Припущення про простий потік дозволяє отримати аналітичні залежності характеристик СМО від параметрів вхідного потоку, що значно складніше для інших видів потоку замовлень. Появу замовлень можна оцінювати за допомогою розподілу Пуассона:

$$p(x) = \frac{e^{-\lambda} \cdot \lambda^x}{x!},$$

де:

$p(x)$ – ймовірність надходження x замовлень за одиницю часу;

x – число замовлень за одиницю часу;

λ – середня кількість замовлень за одиницю часу *(інтенсивність).

Диспетчер – це елемент системи, який розподіляє замовлення по каналах обслуговування. Його основне призначення:

- приймати чи відхиляти замовлення;
- формувати чергу, якщо канали зайняті;
- направляти замовлення на обслуговування, коли є вільний канал;

- приймати інформацію від пристроїв обслуговування про стан каналів;
- слідкувати за часом роботи системи.

Черга – накопичувач замовлень. Основні характеристики черги:
- довжина (кількість місць);
- дисципліна обслуговування.

Виділяють наступні принципи побудови черги:
- FIFO (first in-first out) – першим прийшов-першим вийшов (черга);
- LIFO (last in-first out) – останнім прийшов-першим вийшов (стек);
- випадковий відбір замовлень;
- з пріоритетами, де пріоритет – це переважаюче право на занесення замовлення у накопичувач або його вибір з черги на обслуговування;
- з обмеженим часом очікування (цей випадок можна звести до випадку обмеженої черги)

Обслуговуючий пристрій – пристрій обробки замовлень. Основними характеристиками обслуговуючого пристрою є:
- конфігурація системи обслуговування;
- режим обслуговування.

Система обслуговування може складатись з одного або декількох пристроїв в залежності від чого розрізняють одноканальні та багатоканальні СМО. Кількість каналів визначає кількість замовлень, які можуть одночасно почати оброблятись. Інша характеристика конфігурації – це кількість фаз обслуговування або кількість послідовних етапів обробки одного замовлення (рис. 6).

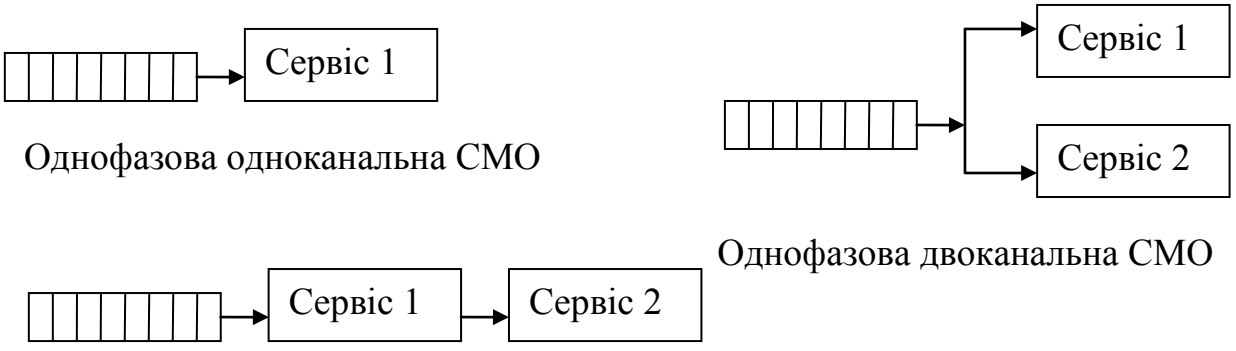


Рис. 6. Приклади конфігурацій СМО

Процес функціонування СМО можна представляти у вигляді послідовності фаз обслуговування, які можуть мати свою структуру та характеристики. Тоді таку багатофазову систему представляють як стохастичну мережу.

Стохастичною мережею (СМО) називається сукупність взаємозв'язаних систем масового обслуговування, для якої визначені ймовірності переходів замовлень між кожною окремою підсистемою СМО (рис. 7).

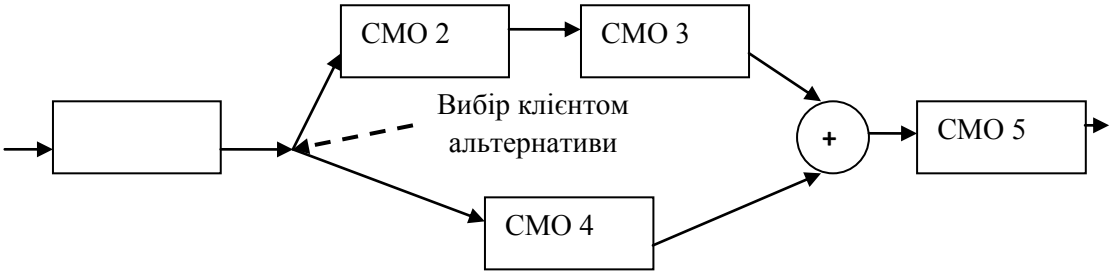


Рис. 7. Приклад мережі масового обслуговування

Режим обслуговування може характеризуватись або постійним або випадковим часом обслуговування. При постійному часі, на обслуговування кожного замовлення витрачається однаковий час, наприклад, на миття автомобіля. У випадку випадкового часу (наприклад, заправка автомобіля) найчастіше застосовується експоненціальний розподіл:

$$F(t) = P(t < T) = 1 - e^{-\lambda t},$$

де $P(t < T)$ – ймовірність того, що фактичний час обслуговування замовлення не буде перевищувати заданої величини T при середній інтенсивності λ

Кожний канал обслуговування може знаходитись в одному з трьох можливих станів:

- канал вільний;
- канал зайнятий;
- канал не працює.

Відмова обслуговування відбувається, якщо всі канали зайняті (при цьому деякі з них можуть бути непрацездатними).

Крім наведених основних елементів в СМО також часто виділяють наступні складові:

- термінатор – поглинач замовлень;
- склад – накопичувач ресурсів та готових замовлень;
- бухгалтерський рахунок – для виконання фінансових операцій;
- менеджер – розпорядник ресурсів.

Приклад. Існує міська мережа ремонту та обслуговування побутових пристроїв, яка складається з центрального call-центру, куди надходять замовлення. Всі замовлення розподіляються між районними відділеннями. Кожне відділення має свій штат персоналу, між яким розподіляються пакети замовлень. Кожний ремонтник має свою кваліфікацію, яка визначає середній час на виконання замовлення, а отже і виробничу потужність відділення.

1.4. Класифікація СМО

Існує велика кількість різноманітних систем масового обслуговування, які розрізняються як структурою, так і організацією процесу функціонування. Класифікація СМО за структурними елементами наведена на рис. 8.

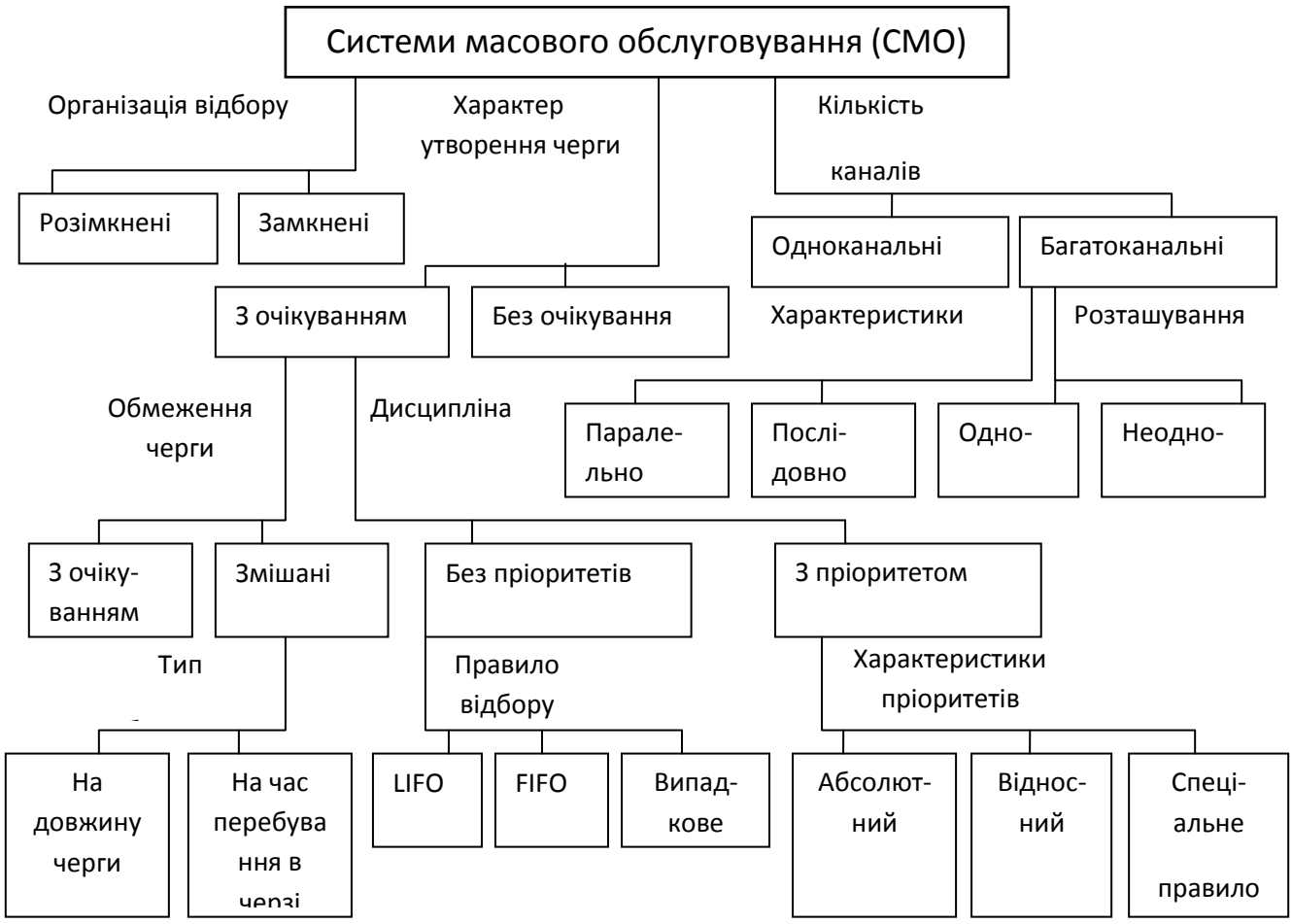


Рис. 8. Класифікація СМО

В залежності від характеру процесів надходження та обслуговування замовлень в систему класифікація СМО та їх мереж наведена на рис. 7. Якщо часові проміжки між замовленнями та час обслуговування є випадковими величинами, то такі системи є стохастичними, якщо вони визначені – то детермінованими. За виглядом залежності, що пов’язує інтенсивності потоків в різних вузлах системи, їх поділяють на лінійні та нелінійні, за типом вхідних та вихідних потоків – на однорідні та неоднорідні.

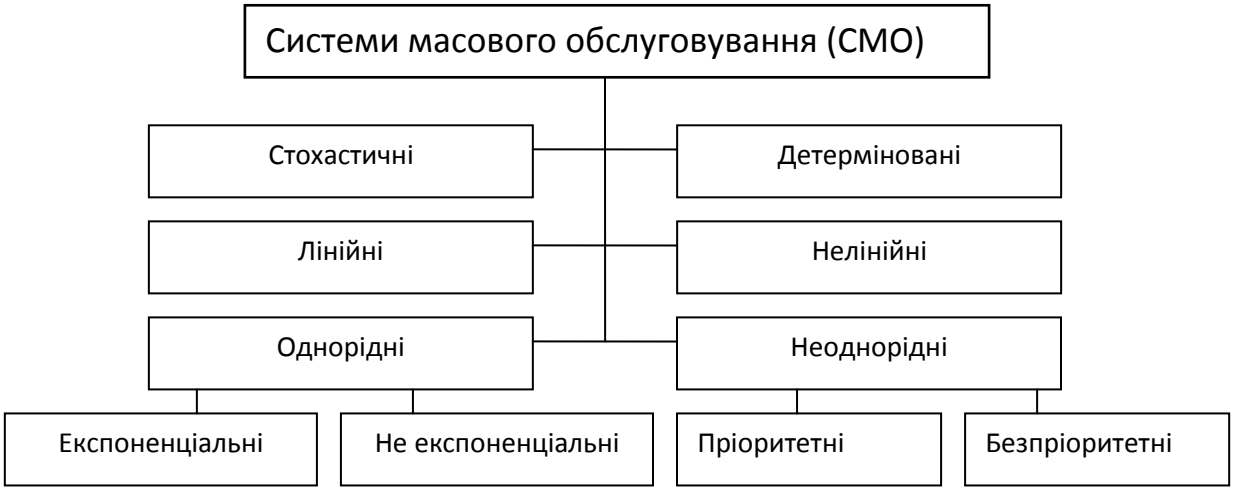


Рис. 9. Класифікація мереж СМО

В лінійних мережах інтенсивність потоків замовлень між *i*-м та *j*-м вузлами описується лінійною залежністю:

$$\lambda_i = k_{ji}\lambda_j,$$

де k_{ji} – коефіцієнт пропорційності (передачі), який показує у скільки разів відрізняються потоки у вузлі *i* та вузлі *j*.

Якщо така залежність справедлива для всіх пар вузлів, то її можна записати як:

$$\lambda_i = k_i\lambda_0,$$

де λ_0 – інтенсивність вхідного потоку мережі СМО.

Коефіцієнт передачі можна трактувати як середню кількість попадання замовлення в даний вузол за час його перебування у мережі.

Розглянувши структуру СМО, можна зазначити, що її функціонування визначається наступними основними факторами:

- *a* –ймовірностним розподілом моментів надходження замовлень (вхідний потік);
- *b* – ймовірностним розподілом часу обслуговування (вихідний потік);
- *c* – кількістю та продуктивністю каналів обслуговування та їх конфігурацією (паралельне, послідовне, комбіноване);
- *e* –максимальною ємністю системи;
- *d* –дисципліною черги, довжина черги визначається як різниця *e* – *c*;
- *f* – потужністю генератора замовлень.

Для скороченого позначення типу СМО застосовується символіка введена Кендаллом-Лі:

(a/ b/ c)(d/ e/ f),

де використовуються наступні стандартні позначення:

- для вхідного та вихідного потоку (*a, b*):
- M – марківський процес (експоненціальний розподіл);
- D – детермінований потік;

- E_k – потік Ерланга або гама розподіл (суми незалежних випадкових величин мають експоненціальний розподіл);
- GI – довільний тип розподілу моментів надходження замовлень;
- G – довільний тип розподілу часу обслуговування;
- для дисципліни черги:
- FCFS – першим прийшов – першим вийшов (FIFO);
- LCFS – останнім прийшов – першим вийшов (LIFO);
- SIRO – випадковий відбір замовлень;
- GD – довільний тип дисципліни.

Значення параметрів *c*, *e*, *f* – натуральні числа, включаючи нескінченність.

1.5. Режими та параметри СМО

Для розробки аналітичних методів розрахунку СМО в багатьох випадках висувається ряд припущень:

- замовлення, що надходить у систему миттєво попадає на обслуговання, якщо є вільні канали;

- замовлення вибирається з черги миттєво (сервіси не простоюють);

- сервіс може в кожний момент часу обслуговувати лише одне замовлення;

- надходження замовлень в систему та час їх обслуговування не залежить від кількості замовлень в системі;

- час обслуговування не залежить від швидкості (інтенсивності) надходження замовлень.

Система масового обслуговування може працювати у двох режимах:

- стаціонарному, коли характеристики СМО не змінюються;

- нестаціонарному, коли характеристики системи змінюються з часом, що може бути обумовлено наступними чинниками:

- перехідним процесом (наприклад, початком роботи), коли характеристики функціонування змінюються і прямують до граничних значень, що визначають стаціонарний режим;
- нестаціонарним вхідним або вихідним потоком замовлень.

Також, у деяких системах з необмеженою чергою може виникнути ситуація перевантаження системи, коли інтенсивність вхідного потоку перевищує інтенсивність процесу обслуговування і довжина черги прямує до нескінченності.

Як правило, розглядається робота СМО у стаціонарному режимі, а для систем з необмеженою чергою висувається умова, що надходження замовлень не перевищує інтенсивності обслуговування (відсутня ситуація перевантаження системи).

Для опису СМО використовують три групи параметрів:

- структурні;
- навантаження;

- функціональні.

До структурних параметрів відносяться:

- кількість обслуговуючих пристроїв c ($c=1$ для одноканальної системи обслуговування і $c>1$ для багатоканальної);
- довжина черги k;
- спосіб взаємозв'язку черги та обслуговуючих пристроїв (для багатоканальних СМО), наприклад у вигляді матриці зв'язків.

Параметри навантаження включають:

- кількість класів замовлень, що надходять у систему (H=1 для систем з однорідним потоком і H >1 для випадку неоднорідних потоків);
- закон розподілу інтервалів часу між
- надходженням замовлень, який характеризується інтенсивністю $\lambda = 1/n$, де n – середня кількість замовлень, що надходить за одиницю часу;
- закон розподілу часу обслуговування або вихідного потоку, який характеризується інтенсивністю $\mu = 1/\tau$, де τ – середній час обслуговування одиничним пристроєм. Для багатоканальної системи $N\mu$ – швидкість обслуговування, коли зайняті всі сервіси.

Функціональні параметри представляють собою конкретні стратегії управління потоками замовлень, які визначають правила управління чергою.

1.6. Випадкові процеси. Граф станів. Марківські процеси

Процес роботи СМО представляє собою випадковий процес, який полягає в тому, що у випадкові моменти часу система переходить з одного стану в інший, а саме змінюється кількість зайнятих каналів, кількість замовлень в черзі і таке інше.

Процес називається ***процесом із дискретними станами***, якщо всі можливі стани можна перерахувати і перехід з одного стану в інший виконується миттєво (стрибок).

Процес називається ***процесом із неперервними станами***, якщо моменти переходу системи з одного стану в інший є випадковими (не фіксовані заздалегідь).

Процес роботи СМО представляє собою випадковий процес з дискретними станами та дискретним або неперервним часом. Системи з дискретними станами та дискретним часом називаються ланцюгами Маркова. Такі системи зручно графічно представляти у вигляді графу станів: орієнтованого графу, де вершини (відображаються прямокутниками) – це множина станів системи $\{S_i\}_{i=\overline{1,n}}$, а дуги – можливі переходи між станами. Кожній дузі відповідає ймовірність переходу системи p_{ij} із стану i в стан j, $i, j = \overline{1,n}$ або для найпростішого потоку – інтенсивність λ_{ij}. Формально система може бути описана за допомогою матриці переходів:

$$\left\| p_{ij} \right\| = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix}.$$

Так як система в кожний момент часу може знаходитись лише в одному стані, то для кожного рядка ймовірності становлять повну групу подій:

$$\sum_{j=1}^{n} p_{ij} = 1.$$

Приклад. Розглянемо технічну систему для якої визначені наступні стани:

- S_1 – система справна і виконує певні операції;
- S_2 – система справна і знаходиться у стані очікування;
- S_3 – система несправна, але факт несправності не встановлений;
- S_4 – система несправна, факт несправності встановлений;
- S_5 – виконуються ремонтні роботи після яких система знову стає до ладу.

Граф станів системи наведений на рис. 10.

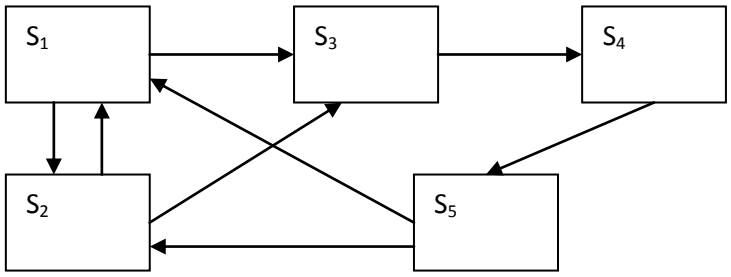


Рис. 10. Граф станів системи

Відповідна матриця перехідних ймовірностей буде мати вигляд:

$$\left\| p_{ij} \right\| = \begin{pmatrix} 0 & p_{12} & p_{13} & 0 & 0 \\ p_{21} & 0 & p_{23} & 0 & 0 \\ 0 & 0 & 0 & p_{34} & 0 \\ 0 & 0 & 0 & 0 & p_{45} \\ p_{51} & p_{52} & 0 & 0 & 0 \end{pmatrix}.$$

Випадковий процес називається марківським або процесом без наслідків, якщо в будь-який момент часу його характеристики залежать лише від поточного стану і не залежать від того як система у цей стан потрапила (не залежать від передісторії). Аналіз СМО значно спрощується, якщо процес її роботи – марківський.

У випадку систем з дискретними станами та дискретним часом для визначення граничних ймовірностей роботи системи у стаціонарному режимі складається система рівнянь за наступними правилами:

в кожному рівнянні ліворуч стоїть гранична ймовірність i-го стану p_i помножена на сумарну інтенсивність вихідних потоків, а праворуч – сума добутків інтенсивностей вхідних потоків на ймовірності тих станів, з яких ці потоки виходять.

Так для прикладу, наведеного на рис. 10, система рівнянь буде мати наступний вигляд:

$$\begin{cases} p_1 + p_2 + p_3 + p_4 + p_5 = 1 \\ p_1(\lambda_{12} + \lambda_{13}) = p_2\lambda_{21} + p_5\lambda_{51} \\ p_2(\lambda_{21} + \lambda_{23}) = p_1\lambda_{12} + p_5\lambda_{52} \\ p_3\lambda_{34} = p_1\lambda_{13} + p_2\lambda_{23} \\ p_4\lambda_{45} = p_3\lambda_{34} \\ p_5(\lambda_{52} + \lambda_{51}) = p_4\lambda_{45} \end{cases}$$

Маємо систему з шести рівнянь з п’ятьма невідомими. Залишивши перше рівняння, одне з наступних рівнянь можна виключити.

Гранична ймовірність показує середній відносний час перебування системи у відповідному стані.

У випадку систем з дискретними станами та неперервним часом для визначення граничних ймовірностей роботи системи у стаціонарному режимі складається система диференційних рівнянь, які називаються рівняннями Колмогорова:

у кожному рівнянні ліворуч стоїть похідна ймовірності відповідного стану, а праворуч – сума вхідних потоків мінус сума вихідних потоків, які описуються подібно правилу для випадку систем з дискретним часом.

Для випадку неперервного часу система рівнянь Колмогорова для графу, представленого на рис. 5, буде мати вигляд:

$$\begin{cases} p_1 + p_2 + p_3 + p_4 + p_5 = 1 \\ \dfrac{dp_1}{dt} = -p_1(\lambda_{12} + \lambda_{13}) + (p_2\lambda_{21} + p_5\lambda_{51}) \\ \dfrac{dp_2}{dt} = -p_2(\lambda_{21} + \lambda_{23}) + (p_1\lambda_{12} + p_5\lambda_{52}) \\ \dfrac{dp_3}{dt} = -p_3\lambda_{34} + (p_1\lambda_{13} + p_2\lambda_{23}) \\ \dfrac{dp_4}{dt} = -p_4\lambda_{45} + p_3\lambda_{34} \end{cases}$$

Так як граничні ймовірності є постійними, то їх похідні в рівняннях Колмогорова можна замінити нульовими значеннями в результаті чого будемо мати систему лінійних алгебраїчних рівнянь, що описують роботу системи у стаціонарному режимі.

Приклади.

1. Розглянемо роботу багатозадачної програмної системи, яка в будь-який момент часу може знаходитись в одному з трьох можливих станів: S_1 – виконувати приорітетну програму, S_2 – виконувати фонову програму, S_3 – знаходитись у стані очікування. Тривалість знаходження системи у кожному із станів є величиною фіксованою і становить Δt. В результаті довгострокового спостереження були визначені перехідні ймовірності перебування системи в кожному із можливих станів:

$$\|p\| = \begin{pmatrix} 0,7 & 0,2 & 0,1 \\ 0,8 & 0,1 & 0,1 \\ 0,8 & 0,05 & 0,15 \end{pmatrix}.$$

Побудувати граф станів системи, скласти систему рівнянь для визначення граничних ймовірностей та визначити коефіцієнт використання процесора.

Розв'язок. За заданою матрицею перехідних ймовірностей побудуємо граф станів (рис. 11).

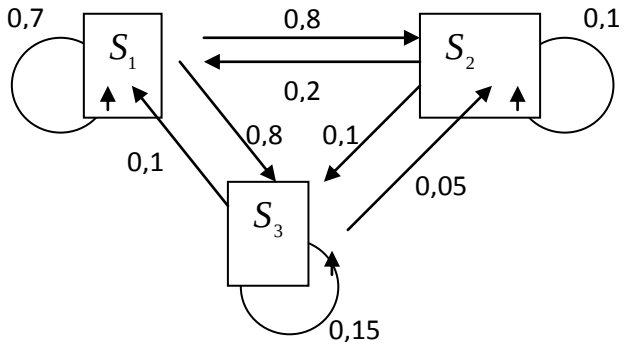


Рис. 11. Граф станів багатозадачної програмної системи

Так як наша система відноситься до систем з дискретним часом та дискретними станами для встановленого режиму роботи рівняння балансу будуть мати вигляд системи лінійних алгебраїчних рівнянь:

$$S_1 = 0,7S_1 + 0,8S_2 + 0,8S_3$$
$$S_2 = 0,2S_1 + 0,1S_2 + 0,05S_3$$
$$S_3 = 0,1S_1 + 0,1S_2 + 0,15S_3$$

Замінимо будь-яке з цих рівнянь умовами нормування: $S_1 + S_2 + S_3 = 1$, наприклад останнє. Тоді отримаємо наступну ситему рівнянь:

$$S_1 = 0,7S_1 + 0,8S_2 + 0,8S_3$$
$$S_2 = 0,2S_1 + 0,1S_2 + 0,05S_3$$
$$S_1 + S_2 + S_3 = 1$$

Розв´язавши цю систему, отримаємо значення граничних ймовірностей:

$$S_1 = 0{,}73,\ \ S_2 = 0{,}17,\ \ S_3 = 0{,}1.$$

Отже ймовірність простою становить $p_3 = 0{,}1$, тоді коефіцієнт використання процесору буде становити: $K_{еф} = 1 - p_3 = 0{,}9$.

2. Розглянемо деяку технічну систему, яка може знаходитись у трьох можливих станах:

- S_1 – система справна і працює;
- S_2 – система має незначні несправності і працює;
- S_3 – система несправна і не працює.

В результаті довгострокових спостережень визначені інтенсивності переходів системи з одного стану в інший (рис. 12).

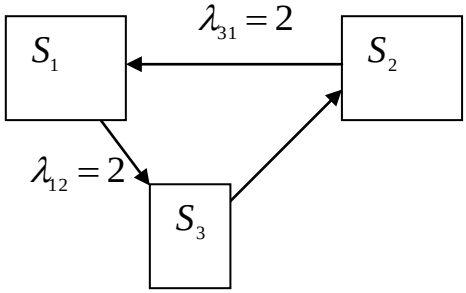


Рис. 12. Граф станів технічної системи

Скласти систему рівнянь роботи системи та знайти граничні ймовірності.

Розв'язок. Переходи системи з одного стану в інший виконуються неперервно у часі, тому ми маємо справу із системою з дискретними станами та неперервним часом. Робота такої системи буде описуватися рівняннями Колмогорова:

$$\begin{cases} \dfrac{dp_1}{dt} = -\lambda_{12} p_1 + \lambda_{31} p_3 \\ \dfrac{dp_2}{dt} = -\lambda_{23} p_2 + \lambda_{12} p_1 \\ \dfrac{dp_3}{dt} = -\lambda_{31} p_3 + \lambda_{23} p_2 \end{cases}.$$

Для знаходження граничних ймовірностей прирівняємо значення похідних до нуля і отримаємо наступну систему лінійних алгебраїчних рівнянь:

$$\begin{cases} -2p_1 + 2p_3 = 0 \\ -p_2 + 2p_1 = 0 \\ -2p_3 + p_2 = 0 \end{cases}.$$

Замінимо останнє рівняння рівнянням балансу і отримаємо:

$$\begin{cases} -2p_1 + 2p_3 = 0 \\ -p_2 + 2p_1 = 0 \\ p_1 + p_2 + p_3 = 1 \end{cases}.$$

Розв’язавши систему, отримаємо наступні значення для граничних ймовірностей: $p_1 = 0{,}25$, $p_2 = 0{,}5$, $p_3 = 0{,}25$. Отже повністю справною система буде лише чверть часу своєї роботи і чверть часу – знаходитись у ремонті. Половину часу роботи система буде працювати, маючи незначні несправності.

1.7. Наближення немарківських процесів марківським методом «псевдостанів»

У реальних практичних задачах пуасонівські потоки зустрічаються не так часто, частіше ми маємо справу з процесами з «наслідками», наприклад, час виходу приладу з ладу залежить від того, скільки він вже пропрацював. Тому важливим є питання про можливість наближення потоків пуасонівськими та оцінки похибки в обчисленні граничних ймовірностей станів системи.

Якщо кількість станів системи не є значною, а потоки представляють (точно або наближено) потік Ерланга, то введенням фіктивних «псевдостанів» вдається звести немарківський процес до марківського і записати для нього систему рівнянь балансу для визначення граничних ймовірностей.

Розглянемо застосування методу «псевдостанів» на прикладі.

Приклади.

1. Нехай система S – деякий технічний прилад, який виходить з ладу під впливом простішого потоку несправностей з інтенсивністю λ. Після виходу з ладу прилад зразу починають відновлювати. Час ремонту T має не пуасонівський закон розподілу, а закон Ерланга 3-го порядку:

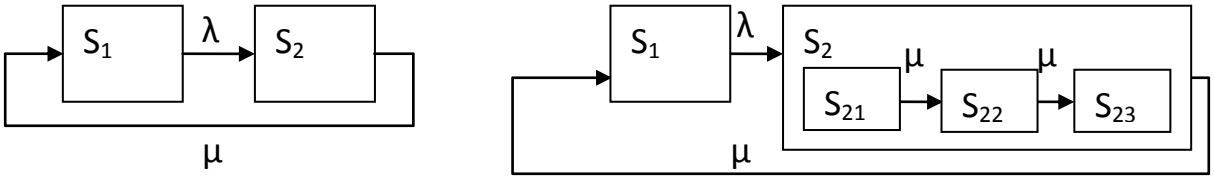
$$f_3(t) = \frac{\mu \cdot (\mu \cdot t)^2}{2} e^{-\mu t},\ t > 0.$$

Необхідно процес наблизити до марківського і знайти для нього граничні ймовірності.

Розв'язок. Випадкова величина T розподілена за законом Ерланга 3-го порядку, а отже, її можна представити як суму трьох випадкових величин T_1, T_2, T_3, розподілених за показниковим законом з параметром μ: $f(t) = \mu e^{-\mu t}$.

Система має лише два істинних стани:

- S_1 – прилад справний;
- S_2 – прилад ремонтується.



а) б)

Рис. 13. Граф станів системи: а) істинні стани; б) із введенням псевдо станів

Що б звести процес до марківського розіб’ємо стан S_2 на три фази (введемо псевдостани):

- S_{21} – початок ремонту;

- S_{22} – продовження ремонту;

- S_{23} – завершення ремонту.

Будемо вважати, що кожний псевдостан має показниковий закон розподілу, тоді процес в системі можна вважати марківським. Для кожного псевдостану введемо відповідні ймовірності: $p_2 = p_{21} + p_{22} + p_{23}$ та введемо позначення для середнього часу перебування у кожному стані: $\bar{t}_1 = 1/\lambda$, $\bar{t}_{21} = 1/\mu$.

Тоді можна одразу записати систему рівнянь для граничних ймовірностей (як для циклічної системи):

$$\begin{cases} p_1 = \dfrac{\bar{t}_1}{\bar{t}_1 + 3\bar{t}_2} \\ p_{21} = p_{22} = p_{23} = \dfrac{\bar{t}_2}{\bar{t}_1 + 3\bar{t}_2} \\ p_2 = p_{21} + p_{22} + p_{23} = \dfrac{3\bar{t}_2}{\bar{t}_1 + 3\bar{t}_2} \end{cases}.$$

Розв´язавши систему рівнянь, отримаємо: $p_1 = \mu/(\mu + 3\lambda)$, $p_2 = 3\lambda/(\mu + 3\lambda)$.

2. Зробимо деяке ускладнення системи. Нехай система S – деякий технічний прилад, що складається з двох однакових вузлів, де кожен з них може виходити з ладу під впливом простішого потоку несправностей з інтенсивністю λ. Після виходу з ладу прилад зразу починають відновлювати. Час ремонту T має не пуасонівський закон розподілу, а закон Ерланга 2-го порядку: $f_3(t) = \mu^2 \cdot t \cdot e^{-\mu t}$, $t > 0$. Знайти граничні ймовірності.

Розв'язок. Визначимо істинні стани системи:

- S_0 – працюють обидва вузли;

- S_1 – один вузол працює, другий – ремонтується;

- S_2 – обидва вузли ремонтуються.

Розіб’ємо ремонт на дві фази: ремонт починається і ремонт завершується. Визначимо псевдостани:

- S_{11} – один вузол працює, другий – починає ремонтуватися;

- S_{12} – один вузол працює, другий – завершує ремонтуватися;

обидва вузли ремонтуються

- $S_{2,11}$ – обидва вузли починають ремонтуватися;

- $S_{2,12}$ – один вузол починає ремонтуватись, а другий – завершує;

- $S_{2,22}$ – обидва вузли завершують ремонт.

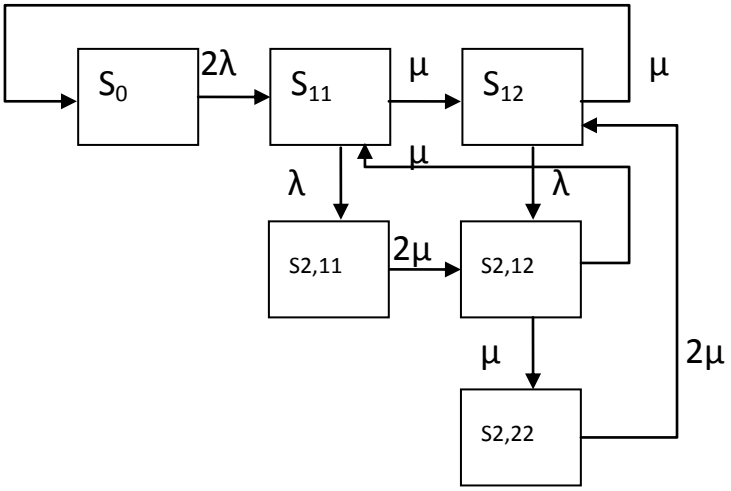


Рис. 14. Граф станів системи з врахуванням псевдостанів

Переходи між станами $S_{2,11}$-$S_{2,12}$ та $S_{2,22}$-S_{12} мають інтенсивність 2μ, так як перейти в наступну фазу (завершення ремонту) може будь-який з двох вузлів.

Система рівнянь для граничних ймовірностей буде мати вигляд:

$$\begin{cases} 2\lambda \cdot p_0 + \mu \cdot p_{2,12} = (\lambda + \mu) \cdot p_{11} \\ \mu \cdot p_{11} + 2\mu \cdot p_{2,22} = (\lambda + \mu) \cdot p_{12} \\ \lambda \cdot p_{11} = 2\mu \cdot p_{2,11} \\ 2\mu \cdot p_{2,11} + \lambda \cdot p_{12} = 2\mu \cdot p_{2,12} \\ \mu \cdot p_{2,12} = 2\mu \cdot p_{2,22} \\ \mu \cdot p_{12} = 2\lambda \cdot p_0 \end{cases}$$

Додамо рівняння того, що ймовірності утворюють повну групу подій:

$$p_0 + p_{11} + p_{12} + p_{2,11} + p_{2,12} + p_{2,22} = 1$$

і виключимо будь-яке рівняння із системи, отримаємо розв'язок спочатку для псевдостанів:

$$p_{11} = \frac{2\lambda\mu}{\mu^2 + 4\lambda\mu + 4\lambda^2};\ p_{12} = \frac{2\lambda\mu}{\mu^2 + 4\lambda\mu + 4\lambda^2};\ p_{2,12} = \frac{2\lambda^2}{\mu^2 + 4\lambda\mu + 4\lambda^2};$$

$$p_{2,11} = \frac{\lambda^2}{\mu^2 + 4\lambda\mu + 4\lambda^2};\ p_{2,22} = \frac{\lambda^2}{\mu^2 + 4\lambda\mu + 4\lambda^2},$$

а потім і для істинних станів:

$$p_0 = \frac{\mu^2}{\mu^2 + 4\lambda\mu + 4\lambda^2};\ p_1 = p_{11} + p_{12} = \frac{4\lambda\mu}{\mu^2 + 4\lambda\mu + 4\lambda^2};$$

$$p_2 = p_{2,11} + p_{2,12} + p_{2,22} = \frac{4\lambda^2}{\mu^2 + 4\lambda\mu + 4\lambda^2}.$$

Необхідно зауважити, що метод «псевдостанів» доцільно застосовувати лише тільки для простих випадків, коли кількість станів є незначною.

Можливість методу розширюється, коли розглядаються узагальнені потоки Еланга.

1.8. Основні характеристики СМО

Визначимо основні цілі для компонентів СМО. Мета клієнта–замовлення, на яке витратити мінімум часу на перебування у черзі. Мета обслуговування – мінімізувати час простою сервісів. Мета аналізу СМО – досягти розумного компромісу між вимогами клієнтів та потужністю системи. Остання характеристика називається показником ефективності системи.

Нехай:

- λ – інтенсивність вхідного потоку;
- μ – інтенсивність вихідного потоку.

До основних характеристик. що визначають процес функціонування системи масового обслуговування відносять:

- *завантаженість* або *коефіцієнт використання системи* ρ – це відношення інтенсивності вхідного потоку до інтенсивності обслуговування $\rho = \lambda / N\mu$,де N – кількість каналів, (для одноканальної системи $\rho = \lambda / \mu$). Якщо система з відмовами (не всі замовлення можуть потрапити до системи) то $\rho = \lambda_{еф} / \mu$, де $\lambda_{еф}$ – ефективна інтенсивність надходження клієнтів у систему $(\lambda_{еф} < \lambda)$. Це відношення показує наскільки задіяні її ресурси. Значення коефіцієнта завантаження визначає необхідну і достатню умови існування стаціонарного режиму, а саме $\rho \le 1$ або $\lambda \le \mu$. В протилежному випадку система буде працювати у режимі перевантаження. Завантаженість ρ характеризує:
  - середня кількість замовлень m, що надходять до системи, за середній час обслуговування одного замовлення τ ($\rho = m \cdot \tau$);
  - частку від інтервалу часу, протягом якого сервіс зайнятий;
  - ймовірність зайнятості сервісу (ймовірність простою сервісу буде складати: $1-\rho$);
  - середню кількість замовлень, що знаходиться в обслуговуючому пристрої (так як сервіс може обслуговувати лише одне замовлення з ймовірністю ρ і простоювати з ймовірністю $1-\rho$, то середня кількість замовлень в обслуговуючому пристрої: $1 \cdot \rho + 0 \cdot (1-\rho) = \rho$).
- *відносна пропускна здатність системи* Q – відносне середнє число замовлень (таблиця 1).

Таблиця 1.

Тип СМО	З відмовами	З чергою	З необмеженою чергою
Однокан альна	$Q = \mu / (\mu + \lambda)$	$Q = 1 - \rho^{m+1} \cdot p_0$	$Q = 1$

Багатоканальна	$Q = 1 - p_{відмови}$	$Q = 1 - p_{відмови}$	$Q = 1$

p_0 – ймовірність того, що канал вільний, $p_{відмови}$ – ймовірність відмови.

- *абсолютна пропускна здатність системи* A – середнє число замовлень, яке може бути оброблене за одиницю часу (таблиця 2).

Таблиця 2.

Тип СМО	З відмовами	З чергою	З необмеженою чергою
Одноканальна	$A = Q \cdot \lambda = p_0 \cdot \lambda$	$A = Q \cdot \lambda$	$A = \lambda$
Багатоканальна	$A = Q \cdot \lambda$	$A = Q \cdot \lambda$	$A = \lambda$

p_0 – ймовірність того, що канал вільний, Q – відносна пропускна здатність.

- *ймовірність відмови* p_0 – ймовірність, що замовлення не потрапить у систему (гранична ймовірність того, що всі канали зайняті):

- $p_{відмови} = \dfrac{\lambda}{\lambda + \mu}$ – для одноканальної СМО;
- $p_{відмови} = \dfrac{\rho^n}{n!} \cdot p_0$ – для багатоканальної СМО.

- *середня кількість зайнятих сервісів* k (для багатоканальних СМО):

$$k = \sum_{i=0}^{n} i p_i = A/\mu = \rho\left(1 - \frac{\rho^n}{n!} p_0\right),$$

n – кількість сервісів, p_i – граничні ймовірності станів.

- *ймовірності станів системи* – найбільш важлива характеристика в тому сенсі, що знаючи ймовірності, можна визначити всі інші характеристики. При цьому під станом системи розуміють кількість замовлень, що знаходиться у системі. Ймовірність стану системи, коли в ній знаходиться k замовлень позначається як p_k (розрахункові формули наведені нижче)

- *час очікування* W_q – середня величина випадкового часу, який замовлення перебуває у черзі в стані очікування W_q;

- *час перебування у системі* W_s – середній час перебування замовлення у системі з моменту надходження до завершення обслуговування:

$$W_s = W_q + 1/\mu$$

(сума середнього часу перебування у черзі та середнього часу обслуговування);

- *середня кількість замовлень у черзі* L_q – середня довжина черги

$$L_q = \sum_{n=m+1}^{\infty} (n - m) p_n = \lambda_{еф} \cdot \mathrm{W_q}$$

- *середня кількість замовлень* m, що знаходяться *у системі* L_s:

$$L_s = \sum_{n=1}^{\infty} n \cdot p_n = \lambda_{еф} \cdot \mathrm{W_s}$$

Залежності між середньою кількістю замовлень в системі L_s і черзі L_q та середнім часом перебування відповідно у системі W_S та черзі W_q називаються *формулами Літтла*.

Середню кількість зайнятих сервісів можна визначити як:

$$m = \lambda_{еф} \Big/ \mu = L_s - L_q .$$

Контрольні питання

1. Дайте визначення поняттю «система масового обслуговування».
2. Які основні складові СМО?
3. Що таке потік замовлень і чим він характеризується?
4. Перерахуйте основні класифікаційні ознаки потоку?
5. Які типи потоків розглядають в теорії систем масового обслуговування? Наведіть приклади потоків різних типів.
6. Який потік називають простішим?
7. Який потік називають потоком Пальма? Наведіть приклади.
8. Що таке Пуасонівський потік? Наведіть приклади.
9. Який потік називають потоком Ерланга? Що означає порядок потоку Ерланга? Наведіть приклади.
10. Коли можна вважати, що потік є заданим?
11. Що таке генератор замовлень і чим він характеризується?
12. Яке призначення диспетчера замовлень?
13. Що таке черга і які типи черг виділяють в СМО?
14. Що таке сервіс, які його характеристики? Які типи обслуговуючих пристроїв розглядають в теорії СМО?
15. Що таке мережа СМО? Наведіть приклад.
16. За якими основними класифікаційними ознаками поділяють СМО?
17. Як класифікують СМО за характером черги?
18. Як класифікують СМО кількістю каналів та способів їх підключення?
19. Наведіть класифікацію мереж СМО.
20. Якими основними факторами визначається функціонування СМО?
21. Що таке символіка Кендалла-Лі?
22. За яких припущень розробляються аналітичні методи розрахунку СМО?
23. Коли режим роботи СМО є стаціонарним?
24. Які характеристики СМО відносяться до структурних параметрів?
25. Які характеристики СМО відносяться до параметрів навантаження?
26. Які сновні типи випадкових процесів розглядають в теорії СМО?
27. Які існують способи представлення випадкоаого процесу?
28. Який процес називається харківським?
29. Як визначаються граничні ймовірності для систем з дискретними станами та дискретним часом?
30. Яку систему рівнянь називають рівняннями Колмогорова?
31. В чому полягає метод наближення не-марківських процесів марківськими методом «псевдостанів».
32. За якими основними характеристиками оцінюється робота СМО?

2. Основні моделі СМО
2.1. СМО з відмовами

Випадок СМО з відмовами відповідає таким системам, коли при надходженні замовлення всі місця в системі зайняті (немає вільних місць в черзі і зайняті всі сервіси) замовлення виходить із системи (відхиляється).

2.1.1. Одноканальна система з відмовами

Розглянемо випадок, коли в системі існує лише один сервіс, на який надходять замовлення з інтенсивністю λ і черга не передбачена (якщо сервіс зайнятий, то замовлення відхиляється). Замовлення обслуговуються з інтенсивністю μ. Середній час обслуговування є оберненою величиною інтенсивності: $t_{обсл} = 1/\mu$.

Знайдемо граничні ймовірності станів системи та показники ефективності.

Система може мати два стани (рис. 15):

- S_0 – сервіс вільний; - S_0 – сервіс зайнятий.

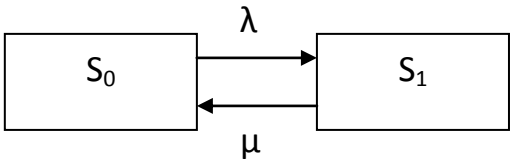


Рис. 15. Граф станів одноканальної СМО з відмовами

У граничному стаціонарному режимі система рівнянь для ймовірностей буде мати вигляд:

$$\begin{cases} \lambda \cdot p_0 = \mu \cdot p_1 \\ p_1 + p_0 = 1 \end{cases}.$$

Розв'язавши систему знайдемо ймовірності:

$$p_0 = \frac{\mu}{\mu + \lambda}, \quad p_1 = \frac{\lambda}{\mu + \lambda},$$

які визначають середній відносний час перебування системи у відповідних станах. Відносна пропускна здатність, ймовірність відмови та абсолютна пропускна здатність становлять:

$$Q = \frac{\mu}{\mu + \lambda} = 1 - p_1, \quad p_{відмови} = \frac{\lambda}{\mu + \lambda} \quad A = \frac{\mu \cdot \lambda}{\mu + \lambda} = \lambda Q.$$

Приклади.

1. В приймальній офісу встановлений телефон на який дзвонять замовники. Якщо телефон зайнятий, то дзвінок не приймається (відбувається відмова замовлення). Дзвінки утворюють простий потік з інтенсивністю $\lambda = 30$ дзвінків/год. Час однієї розмови є випадковою величиною з

експоненціальним законом розподілу із середньою тривалістю $t_{об} = 3$ хв. необхідно визначити показники ефективності системи.

Розв'язок. Перед розвязуванням задачі необхідно всі значення привести до однакових розмірностей. Визначимо середню триваліст розмови (час обслуговування) у годинах $t_{об} = 3/60 = 0.05$ год. Тоді можна визначити інтенсивність обслуговування:

$$\mu = 1/t_{об} = 1/0.05 = 20 \text{ дзвінків/год.}$$

Представимо нашу систему обслуговування у вигляді графа (рис. 16). Вона має два стани: S_0 – система вільна з ймовірністю p_0; S_1 – система зайнята з ймовірністю p_1.

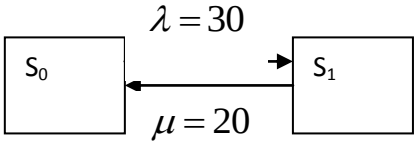


Рис. 16. Граф станів системи (приклад 1)

Визначимо ймовірності перебування системи у відповідному стані:

$$p_0 = \frac{\mu}{\mu + \lambda} = \frac{20}{20 + 30} = 0.4, \quad p_1 = \frac{\lambda}{\mu + \lambda} = \frac{30}{20 + 30} = 0.6,$$

Визначимо показники ефективності системи:

- відносна пропускна спроможність $Q = \frac{\mu}{\mu + \lambda} = 0.4$

- -абсолютна пропускна спроможність $A = \frac{\mu \cdot \lambda}{\mu + \lambda} = \lambda Q = 30 \cdot 0.4 = 12$ дзвінків/год.

- ймовірність відмови $p_{відмови} = \frac{\lambda}{\mu + \lambda} = 0.6$.

З розв'язку видно, що номінальна пропускна спроможність $\mu = 20$ відрізняється від абсолютної $A = 12$ в силу того, що потік дзвінків має випадковий характер і випадковий час обслуговування.

2.1.2. Багатоканальна система з відмовами

Розглянемо задачу Ерланга. Існує *n* каналів (сервісів), на які надходять замовлення з інтенсивністю λ і обслуговуються з інтенсивністю μ (черга не передбачена). Система може мати множину станів $\{S_i\}_{i=\overline{1,n}}$, де S_i – стан, коли зайнято *i* каналів (в системі знаходиться *I* замовлень) (рис. 17).

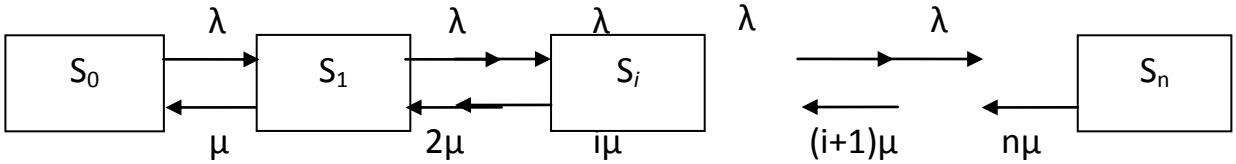


Рис. 17. Граф станів багатоканальної СМО (процес розмноження та загибелі)

Інтенсивність вхідного потоку є постійною, інтенсивність вихідного потоку змінюється в залежності від стану системи (кількості зайнятих каналів).

Граничні ймовірності станів системи будуть обчислюватись за наступними формулами, які називаються формулами Ерланга:

$$p_0 = \left(1 + \frac{\lambda}{\mu} + \frac{\lambda^2}{2!\mu^2} + \ldots + \frac{\lambda^i}{i!\mu^i} + \ldots + \frac{\lambda^n}{n!\mu^n}\right)^{-1} = \left(1 + \rho + \frac{\rho^2}{2!} + \ldots + \frac{\rho^i}{i!} + \ldots + \frac{\rho^n}{n!}\right)^{-1}$$

$$p_1 = \rho \cdot p_0,\ p_1 = \frac{\rho^2}{2!} \cdot p_0, \ldots,\ p_k = \frac{\rho^k}{k!} \cdot p_0, \ldots,\ p_n = \frac{\rho^n}{n!} \cdot p_0;$$

Ймовірність відмови буде відповідати ситуації, коли всі n каналів системи зайняті:

$$p_{відмови} = \frac{\rho^n}{n!} p_0.$$

Відносна та абсолютна пропускні здатності відповідно становлять:

$$Q = 1 - p_{відмови} = 1 - \frac{\rho^n}{n!} p_0, \quad A = \lambda \cdot Q = \lambda \cdot \left(1 - \frac{\rho^n}{n!} p_0\right).$$

Середнє число зайнятих (вільних) каналів $k_{зайнятих}$ ($k_{вільнх}$)– це математичне сподівання числа зайнятих (вільних) каналів:

$$k_{зайнятих} = \sum_{i=0} (i \cdot p_i) = \frac{A}{\mu} = \rho \cdot \left(1 - \frac{\rho^n}{n!} p_0\right), \quad k_{вільних} = \sum_{i=0}^{n} (n - i) \cdot p_i$$

Коефіцієнт зайнятостім (простою) каналів:

$$K_{зайнятих} = \frac{k_{зайнятих}}{n} \qquad K_{простою} = \frac{k_{простою}}{n}.$$

2. Служба ремонту має три телефонні лінії для виклику майстра. Якщо всі телефони зайняті, клієнт не може дозвонитись, отже отримує відмову. Дзвінки надходять зінтенсивністю $\lambda = 60^{дзвінків/год.}$ Час на прийом замовлення (обслуговування) є випадковою величиною з експоненціальним законом розподілу, середня тривалість розмови $t_{обсл.} = 3хв.$ Визначити показники ефективності роботи системи.

Розвязок. Система відноситься до типу багатоканальних СМО з відмовами, граф станів якої наведений на рис. 18.

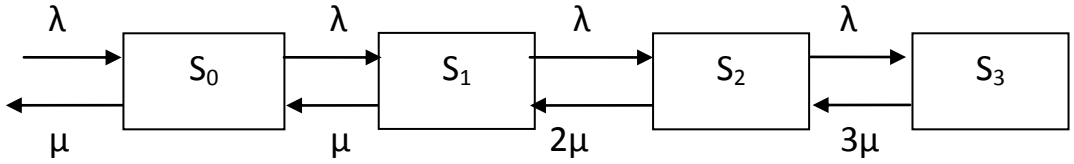


Рис. 18. Приклад трьохканальної СМО з відмовами

Приведемо всі параметри до одних одиниць вимірювання часу (годин):

$$t_{обсл.} = 3хв. = 3/60 = 0{,}05\ год.$$

Тоді інтенсивність обслуговування: $\mu = 1/t_{обсл.} = 1/0{,}05 = 20\ дзвінків$,

приведена інтенсивність потоку:

$$\rho = \lambda/\mu = 60/20 = 3.$$

Обчислимо граничні ймовірності станів:

$$p_0 = \left(1 + \rho + \frac{\rho^2}{2!} + \frac{\rho^3}{3!}\right)^{-1} = \left(1 + 3 + \frac{9}{2} + \frac{27}{6}\right)^{-1} = \frac{1}{13} = 0{,}077$$

$$p_1 = \rho \cdot p_0 = 0{,}231, \quad p_2 = \frac{\rho^2}{2!} \cdot p_0 = 0{,}346, \quad p_3 = \frac{\rho^3}{3!} \cdot p_0 = 0{,}346$$

Ймовірність відмови буде становити: $p_{відмови} = 0{,}346$.

Обчислимо відносну та абсолютну пропускні здатності:

$Q = 1 - p_{відмови} = 1 - 0.346 = 0{,}654$, $A = \lambda \cdot Q = 60 \cdot 0{,}654 = 39{,}24$.

Середнє число зайнятих (вільних) каналів $k_{зайнятих}$ ($k_{вільнх}$):

$$k_{зайнятих} = \frac{A}{\mu} = \frac{39{,}24}{20} = 1{,}962, \qquad k_{вільних} = 1{,}038.$$

Відповідні коефіцієнти зайнятості будуть становити:

$$K_{зайнятих} = \frac{k_{зайнятих}}{n} = \frac{1{,}962}{3} = 0{,}654 \qquad K_{простою} = \frac{k_{простою}}{n} = \frac{1{,}038}{3} = 0{,}094.$$

3. Розглядається робота автозаправної станції (АЗС) з трьома колонками. Якщо зайняті всі три колонки, то машина не стає в чергу, а залишає АЗС. Середній час заправки автомобіля – 3 хв. Інтенсивність потоку автомобілів - 0,25 од./хв. Знайти граничні ймовірності і показники ефективності роботи АЗС.

Розв'язок. Визначимо показники системи:

- $\lambda = 0{,}25$ – інтенсивність вхідного потоку;
- $t_{обсл} = 3$ – час обслуговування, тоді $\mu = 1/t_{обсл}$;
- $\rho = \lambda/\mu = \lambda \cdot t_{обсл} = 0{,}75$.

Знайдемо граничні ймовірності:

$$p_0 = \left(1 + 0{,}75 + \frac{0{,}75^2}{2!} + \frac{0{,}75^3}{3!}\right)^{-1} = 0{,}478, \quad p_1 = 0{,}75 \cdot 0{,}476 = 0{,}357$$

$$p_2 = \frac{0{,}75^2}{2!} \cdot 0{,}476 = 0{,}134 \ , \ p_3 = \frac{0{,}75^3}{3!} \cdot 0{,}476 = 0{,}033$$

Отже в стаціонарному режимі 47,6% часу АЗС не працює і тільки 3,3% працюють всі три колонки.

Ймовірність відмови: $p_{відмови} = p_3 = 0{,}033$.

Відносна пропускна спроможність: $Q = 1 - p_{відмов} = 1 - 0{,}033 = 0{,}967$, отже 96,7% автомобілів буде підлягати обслуговуванню.

Абсолютна пропускна спроможність: $A = \lambda \cdot Q = 0{,}25 \cdot 0{,}967 = 0{,}242$.

Середня кількість зайнятих сервісів: $k = \frac{A}{\mu} = \frac{0{,}242}{1/3} = 0{,}725 < 1.$

Якщо середня кількість зайняти каналів менше 1, то роботу системи не можна вважати ефективною.

4. На вхід багатоканальної системи з відмовами надходить потік замовлень з інтенсивністю $\lambda = 7$ замовлень за годину. Середній час обслуговування замовлення $t_{обсл.} = 0{,}25$ години. Обслуговування замовлення приносить дохід у розмірі 150 грошових одиниць, а утримання каналу обслуговування коштує 120 грошових одиниць за годину. Визначити оптимальну кількість каналів СМО.

Розвязок. Визначимо основні показники СМО:

- середній час обслуговування $t_{обсл.} = 0{,}25$;
- кількість місць в черзі $m = 1$;
- інтенсивність замовлень $\lambda = 7$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 1/0{,}25 = 4$;
- відносне навантаження на систему $\rho = \lambda / \mu = 7/4 = 1{,}75$.

Необхідно визначити кількість каналів.

Якщо система має n каналів, то дохід можна визначити за наступною формулою:

$$D = 150 * A - 120 * n,$$

де A – абсолютна пропускна спроможність СМО.

Для визначення оптимальної кількості каналів складемо розрахункову таблицю.

Таблиця 2.1. Визначення оптимального числа каналів

	Ймовірність вільного стану СМО P_0	Ймовірність відмови $P_{відмови} = \frac{\rho^n}{n!} P_0$	Абсолютна пропускна спромож-ність $A = \lambda(1 - P_{відмови})$	Дохід D
	$P_0 = (1+\rho)^{-1} = 0{,}364$	0,636	2,545	261,75
	$P_0 = (1+\rho+\rho^2/2!)^{-1} = 0{,}234$	0,204	5,569	5

				95,35
	$P_0 = \left(1 + \rho + \rho^2/2! + \rho^3/3!\right)^{-1} = 0{,}193$	0,056	6,61	6 31,5
	$P_0 = \left(1 + \rho + \rho^2/2! + \rho^3/3! + \rho^4/4!\right)^{-1} = 0{,}18$	0,013	6,91	5 56,5

З таблиці видно, що із зростанням кількості каналів, дохід збільшується і оптимальним є три канали.

2.2. СМО з очікуванням (з чергою)

2.2.1. Одноканальна СМО з необмеженою чергою (без відмов)

Розглянемо систему з одним сервісом, коли на чергу не накладаються будь-які обмеження. Потік замовлень має інтенсивність λ, а потік обслуговування – інтенсивність μ. Система може знаходитись у наступних станах (рис. 19):

- S_0 – сервіс вільний;
- S_1 – сервіс зайнятий;
- S_i, $i = 2,3,\ldots$ – канал зайнятий і у черзі стоїть i замовлень.

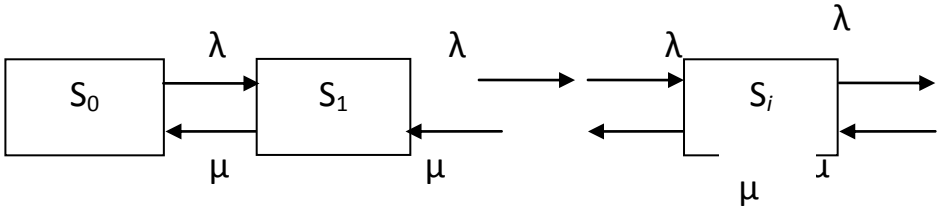


Рис. 19. Граф станів СМО з необмеженою чергою

Така система представляє процес "загибелі та розмноження" з нескінченною множиною станів. Граничні ймовірності визначаються за наступними формулами ($\rho = \dfrac{\lambda}{\mu} < 1$):

$$p_0 = 1 - \rho, \quad p_1 = \rho \cdot p_0, \quad p_2 = \rho^2 \cdot p_0, \ldots, \; p_i = \rho^i \cdot p_0$$

Середня кількість Замовлень в системі та в черзі визначаються як математичне сподівання:

$$L_{сист} = \frac{\rho}{1-\rho}, \quad L_{черги} = \mathrm{L}_{\text{сист}} - \mathrm{L}_{\text{обсл}},$$

де середня кількість замовлень, що обслуговується:

$$\mathrm{L}_{\text{обсл}} = 0 \cdot \mathrm{p}_0 + 1 \cdot (1 - p_0) = 1 - p_0 = P_{зайнято} = \rho$$

і дорівнює ймовірності зайнятості сервісу.

Середня кількість замовлень, що знаходяться в стані очікування:

$$L_{очікування} = \frac{\rho^2}{1-\rho}.$$

Середній час перебування замовлень в системі та в стані очікування:

$$W_{сист} = \frac{1}{\lambda} L_{сист}, \quad W_{очік} = \frac{1}{\lambda} L_{очік}.$$

Відповідний середній час для одного замовлення:

$$W_{сиис} = \frac{1}{\lambda} \cdot \frac{\rho}{1-\rho}, \quad W_{очік} = \frac{1}{\lambda} \cdot \frac{\rho^2}{1-\rho}.$$

Приклади.

1. У закладі встановлено один термінал для проведення платежів. Інтенсивність потоку клієнтів становить 10 осіб протягом робочої доби, яка складає 6 год., що становить чверть доби. Середній час обслуговування одного клієнта становить 30 хв.=0,5 год. Вважається, що черга може бути необмеженої довжини. Знайти показники ефективності роботи терміналу, та ймовірність того, що в черзі не більше 2 осіб.

Розв'язок. Визначимо основні показники СМО:

- середній час обслуговування $t_{обсл.} = 0{,}5$;
- кількість мість в черзі $m = \infty$;
- інтенсивність замовлень $\lambda = 10/6 \approx 1{,}7$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 1/0{,}5 = 2$;
- відносне навантаження на систему $\rho = \lambda/\mu = 1{,}7/2 \approx 0{,}83$.

Так як $\rho = 0{,}83 < 1$, то черга не може нескінченно зростати і граничні ймовірності існують.

Ймовірність того, що термінал вільний:

$$p_0 = 1 - \rho = 1 - 0{,}83 = 0{,}17,$$

а ймовірність, що він зайнятий $p_{зайнятий} = 0{,}83$.

Знайдемо ймовірності, що біля терміналу знаходиться 1, 2, 3 особи:

$p_1 = \rho \cdot p_0 = 0{,}141$, $p_2 = \rho^2 \cdot p_0 = 0{,}117$, $p_3 = \rho^3 \cdot p_0 = 0{,}097$.

Ймовірність того, що в черзі не більше 2 осіб становить:

$$P = p_1 + p_2 + p_3 = 0{,}355.$$

Середня кількість осіб у черзі: $L_{очікування} = \frac{\rho^2}{1-\rho} = 4{,}05$.

Середній час перебування у черзі: $W_{очік} = \frac{1}{\lambda} L_{очік} = 2{,}382$.

Середня кількість осіб у системі: $L_{сист} = \frac{\rho}{1-\rho} = 4{,}882$

Середній час перебування у системі: $W_{сиис} = \frac{1}{\lambda} \cdot \frac{\rho}{1-\rho} = 2{,}872$.

Очевидно, що ефективність обслуговування необхідно підвищити. Для цього необхідно спростити процедуру оплати, що зменшить час на її проведення або поставити ще один термінал.

2.2.2. Багатоканальна СМО з необмеженою чергою (без відмов)

Нехай система складається з n каналів і має необмежену чергу . Вона може знаходитись у наступних станах (рис.20):

- S_0 – всі канали вільні;
- S_i, $i=\overline{1,n}$ – зайнято i каналів;
- S_{n+k} – всі канали зайняті і в черзі знаходиться k замовлень.

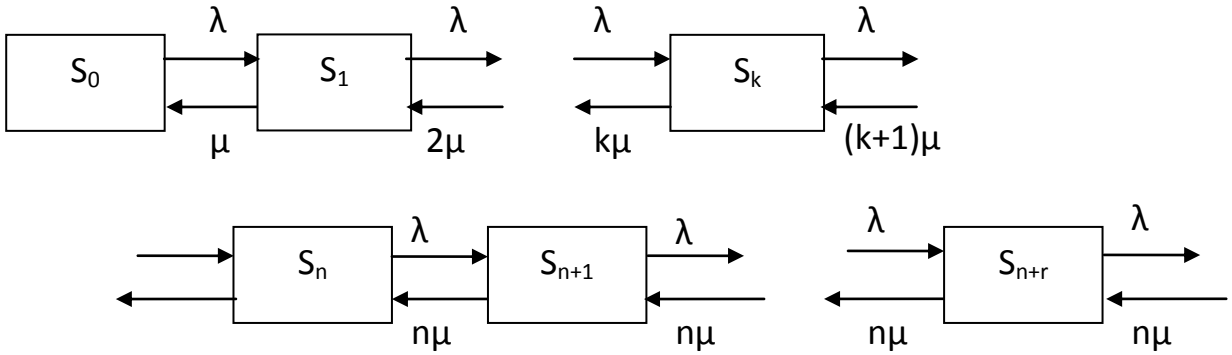


Рис. 20. Граф станів багатоканальної СМО з необмеженою чергою

Якщо $\rho<1$, то граничні ймовірності існують і обчислюються за наступними формулами:

$$p_0=\left(1+\frac{\rho}{1!}+\frac{\rho^2}{2!}+\ldots+\frac{\rho^n}{n!}+\frac{\rho^{n+1}}{n!(n-\rho)}\right)^{-1},$$

$$p_1=\frac{\rho}{1!}p_0,\ldots,\ p_k=\frac{\rho^k}{k!}p_0,\ldots,\ p_n=\frac{\rho^n}{n!}p_0,$$

$$p_{n+1}=\frac{\rho^{n+1}}{n\cdot n!}p_0,\ldots,\ p_{n+r}=\frac{\rho^{n+r}}{n^r\cdot n!}p_0,\ldots$$

Ймовірність того, що замовлення буде у черзі:

$$P_{черга}=\frac{\rho^{n+1}}{n!(n-\rho)}p_0.$$

Середня кількість зайнятих каналів:

$$k=\frac{\lambda}{\mu}=\rho.$$

Середня кількість замовлень в черзі та системі:

$$L_{черга}=\frac{\rho^{n+1}}{n\cdot n!\left(1-\rho/n\right)^2}p_0,\quad L_{сист}=L_{черга}+\rho.$$

Середній час перебування замовлення в черзі та системі обчислюється за формулами Літтла.

Для системи з необмеженою чергою ймовірність відмови $p_{відмови} = 0$, відносна пропускна здатність $Q = 1$, а абсолютна – $A = \lambda$.

Приклади.

1. До касового відділення банку надходить потік людей з інтенсивністю $\lambda = 81 осіб/год$. Середній час обслуговування однієї особи становить 2 *хв*. Необхідно визначити:

1) мінімальну кількість касирів $n_{\min}$, при якому черга не буде рости до нескінченності та відповідні характеристики системи для цього значення;

2) оптимальну кількість касирів n_{op}, при якій відносна величина витрат на обслуговування, яка визначається за формулою $C_{op} = \frac{n}{\lambda} + 3T_{очікування}$ буде найменшою і порівняти характеристики системи при $n_{\min}$ та n_{op};

3) ймовірність того, що в черзі буде не більше трьох осіб.

Розв'язок. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 81/60 = 1{,}35$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 1/2 = 0{,}5$;
- відносне навантаження на систему $\rho = \lambda/\mu = 2{,}7$.

1) Черга не буде зростати до нескінченності за умови $\rho/n < 1$, отже для наших значень $n_{\min} = 3$.

Знайдемо характеристики системи для $n_{\min} = 3$.

Ймовірність відсутності клієнтів у системі буде становити:

$$p_0 = \left(1 + 2{,}7 + \frac{2{,}7^2}{2!} + \frac{2{,}7^3}{3!} + \frac{2{,}7^4}{4!}\right)^{-1} = 0{,}025,$$

що означає, що в середньому 2,5% часу касовий вузол буде простоювати.

Обчислимо ймовірність наявності черги:

$$P_{чер} = \frac{2{,}7^4}{3! \cdot (3 - 2{,}7)} \cdot 0{,}025 = 0{,}735.$$

Середня довжина черги буде становити:

$$L_{черга} = \frac{2{,}7^4}{3 \cdot 3! \cdot (1 - 2{,}7/3)^2} \cdot 0{,}025 = 7{,}35.$$

Середній час очікування в черзі:

$$W_{чер} = \frac{7{,}35}{1{,}35} = 5{,}44 хв.$$

Середня кількість касирів: $k = \rho = 2{,}7$.

Коефіцієнт зайнятих касирів: $k_{від} = \rho/n = 0{,}9$

Абсолютна пропускна спроможність відділення: $A = 1{,}35\left(1/хв\right) = 81\left(1/год\right)$ – 81 клієнт за годину.

Проаналізувавши отримані результати розрахунків, можна зробити висновок, що за наявності трьох касирів $n_{\min}=3$ розрахунковий центр перевантажений.

2) Обчислимо відносну величину витрат при $n_{\min}=3$:

$$C_{op}=\frac{n}{\lambda}+3T_{очікування}=\frac{3}{1{,}35}+3\cdot 5{,}44=5{,}44.$$

Розрахуємо відносну величину витрат при інших значеннях n, та занесемо результати у таблицю (табл. 2.2).

Таблиця 2.2. Значення витрат

Характеристика обслуговування	Кількість касирів				
	3	4	5	6	7
Ймовірність простою p_0	0,025	0,057	0,065	0,067	0,067
Середня довжина черги $L_{черга}$	5,44	0,60	0,15	0,03	0,01
Відносна величина витрат C_{op}	18,54	4,77	4,14	4,53	5,22

З таблиці видно, що мінімальні витрати отримані при наявності п'яти касирів $n=5$. Знайдемо характеристики системи для цього випадку:

$P_{чер}=0{,}091,\ L_{черга}=0{,}198,\ W_{чер}=0{,}146,\ L_{сист}=2{,}9,\ W_{сист}=2{,}15,\ k_{відн}=0{,}54.$

Очевидно, що при $n=5$ значно зменшилась ймовірність виникнення черги та довжина черги, а також середній час перебування клієнтів у черзі.

3) Визначимо ймовірність того, що в черзі буде не більше 3 клієнтів при $n=5$:

$$P(r\le 3)=p_1+p_2+p_3+p_4+p_5+p_{5+1}+p_{5+2}+p_{5+3}=1-P_{черги}+p_{5+1}+p_{5+2}+p_{5+3}=$$

$$=1+\frac{2{,}7^6}{5!\cdot(5-2{,}3)}\cdot 0{,}065+\frac{1{,}7^6}{5\cdot 5!}\cdot 0{,}065+\frac{1{,}7^7}{5^2\cdot 5!}\cdot 0{,}065+\frac{1{,}7^8}{5^3\cdot 5!}\cdot 0{,}065=0{,}986$$

(Можна зазначити, що при $n=3$ ця ймовірність менша $P(r\le 3)=0{,}464$).

2.2.3. СМО з обмеженою чергою (з відмовами). Задача Ерланга

СМО з обмеженою чергою відрізняється від попередніх випадків тим, що якщо черга заповнена, то нове замовлення, що надійшло, отримує відмову.

Різниця у визначені граничних ймовірностей та інших показників системи для СМО з обмеженою чергою полягає в тому, що суми визначаються скінченною прогресією. Нехай m – довжина черги, r – кількість замовлень у черзі. Відповідні формули наведені у таблиці.

Таблиця 2.3. Показники СМО

Одноканальна СМО	Багатоканальна СМО
Граничні ймовірності	
$p_0 = \frac{1-\rho}{1-\rho^{m+2}}$, $p_1 = \rho \cdot p_0$, $p_2 = \frac{\rho^2}{2!} \cdot p_0, \ldots, \ p_k = \frac{\rho^k}{k!} \cdot p_0$	$p_0 = \left(1 + \frac{\rho}{1!} + \ldots + \frac{\rho^n}{n!} + \ldots + \frac{\rho^{n+1}\left(1 - \left(\rho/n\right)^m\right)}{n \cdot n!\left(1-\rho/n\right)} \right)$, $p_1 = \rho \cdot p_0, \ldots, \ p_n = \frac{\rho^n}{n!} \cdot p_0$, $p_{n+1} = \frac{\rho^{n+1}}{n \cdot n!} \cdot p_0, \ldots, \quad p_{n+r} = \frac{\rho^{n+r}}{n^r \cdot n!} \cdot p_0$, $r = \overline{1,m}$
Ймовірність відмови	
$p_{відмови} = p_{m+1} = \rho^{m+1} p_0$	$p_{відмови} = p_{m+n} = \frac{\rho^{m+n}}{n^m \cdot n!} p_0$
Абсолютна пропускна здатність	
$A = \lambda Q = \lambda\left(1 - \rho^{m+1} p_0\right)$	$A = \lambda Q = \lambda\left(1 - \frac{\rho^{m+n}}{n^m \cdot n!} p_0\right)$
Відносна пропускна здатність	
$Q = 1 - p_{відмови} = 1 - \rho^{m+1} p_0$	$Q = 1 - p_{відмови} = 1 - \frac{\rho^{m+n}}{n^m \cdot n!} p_0$
Середня кількість замовлень у черзі	
$L_{черги} = \rho^2 \frac{1-\rho^m\left(m+1-m\rho\right)}{\left(1-\rho^{m+2}\right)\left(1-\rho\right)}$	$L_{черги} = \rho^{n+1} p_0 \left(1 - \left(m+1-m\frac{\rho}{n}\right)\left(\frac{\rho}{n}\right)^m\right)$
Середня кількість зайнятих сервісів	
$L_{обсл} = 1 - \mathrm{p}_0$	$k = \rho\left(1 - \frac{\rho^{n+m}}{n^m n!} p_0\right)$
Середня кількість замовлень у системі	
$L_{сист} = \mathrm{L}_{\text{очікування}} + \mathrm{L}_{\text{обсл}}$	$L_{сист} = \mathrm{L}_{\text{очікування}} + \mathrm{k}$
Середній час перебування у системі	
$W_{сист} = \frac{L_{системи}}{\lambda}$	$W_{сист} = \frac{L_{системи}}{\lambda}$
Середній час перебування у черзі	
$W_{очікування} = \frac{L_{очікування}}{\lambda}$	$W_{очікування} = \frac{L_{очікування}}{\lambda}$

Середній час перебування замовлення в черзі та системі обчислюється за формулами Літтла.

Приклади.

1. Одноканальна СМО з обмеженою чергою.

1) У розрахунковому центр $W_{сист} = \frac{3}{2} = 1{,}5(хв)$ і є лище одна каса. Приміщення допускає перебування не більше шести осіб (один клієнт обслуговується, а інші стоять у черзі, отже черга обмежена довжиною $m = 5$). Якщо приміщення повність зайняте, то клієнти не стають у чергу і відходять. Потік клієнтів має інтенсивність $\lambda = 2\left(особи/хв\right)$. Інтенсивність потоку обслуговування становить $\mu = 2$. Визначити характеристики СМО і зробити висновки стосовно ефективності його роботи.

Розв'язок. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 2$;
- інтенсивність обслуговування $\mu = 2$;
- відносне навантаження на систему $\rho = \lambda / \mu = 1$.

Ймовірність відсутності клієнтів у системі буде становити: $p_0 = \frac{1}{m+2} = \frac{1}{7}$.

Ймовірність, що в системі є відповідна кількість клієнтів: $p_{m+1} = \rho^{m+1} p_0$.

Так як $\rho = 1 \Rightarrow \quad p_1 = p_2 = \ldots = p_6 = \frac{1}{7}$.

Ймовірність відмови: $p_{відмови} = p_6 = \frac{1}{7}$.

Відносна пропускна здатність: $Q = 1 - p_{відмови} = \frac{6}{7}$.

Абсолютна пропускна здатність: $A = \lambda Q = \frac{12}{7} \approx 1{,}7\left(клієнтів/хв\right)$.

Середня кількість клієнтів у системі:

$$L_{сист} = 0 \cdot \frac{1}{7} + 1 \cdot \frac{1}{7} + 2 \cdot \frac{1}{7} + 3 \cdot \frac{1}{7} + 4 \cdot \frac{1}{7} + 5 \cdot \frac{1}{7} + 6 \cdot \frac{1}{7} = 3.$$

Середній час перебування клієнта у системі: $W_{сист} = \frac{3}{2} = 1{,}5(хв)$.

Середня довжина черги: $L_{черги} = \frac{5 \cdot 6}{2 \cdot 7} = 2{,}1(клієнта)$.

Середній час перебування клієнта у черзі: $W_{очікування} = \frac{2{,}1}{2} = 1{,}05(хв)$.

З розрахованих характеристик видно, що кожному сьомому клієнту відмовляється в обслуговуванні, отже ефективність системи не є високою.

2) У невеликому магазині самообслуговування встановлено, що потік покупців є найпростішим з інтенсивністю $\lambda = 1$ покупець в хвилину. В цьому магазині встановлений один касовий апарат, що дозволяє домогтися такої продуктивності праці, при якій середній час обслуговування одного клієнта становить приблизно 1,25 хв. покупця за хвилину.

Визначте характеристики СМО за умови, що черга обмежена контролером при вході в зал самообслуговування: $m = 3$ покупців.

Розв'язок. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 1$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 1/1{,}25 = 0{,}8$;
- відносне навантаження на систему $\rho = \lambda / \mu = 1{,}25$.

Знайдемо граничні ймовірності:

$$p_0 = \frac{1-1{,}25}{1-1{,}25^{3+2}} \approx 0{,}122, \quad p_1 = 1{,}25 \cdot 0{,}122 \approx 0{,}152, \quad p_2 = 1{,}25^2 \cdot 0{,}122 \approx 0{,}191,$$

$$p_3 = 1{,}25^3 \cdot 0{,}122 \approx 0{,}238, \quad p_4 = 1{,}25^4 \cdot 0{,}122 \approx 0{,}297.$$

Ймовірність відмови: $p_{відмови} = p_4 = 0{,}297$.

Відносна пропускна здатність: $Q = 1 - p_{відмови} = 1 - 0{,}297 = 0{,}703$.

Абсолютна пропускна здатність: $A = \lambda Q = 1 \cdot 0{,}703 = 0{,}703\left(клієнтів/хв\right)$.

Середня кількість клієнтів у системі:

$$L_{сист} = 0 \cdot 0{,}122 + 1 \cdot 0{,}152 + 2 \cdot 0{,}191 + 3 \cdot 0{,}238 + 4 \cdot 0{,}297 = 2{,}436$$

Середній час перебування клієнта у системі: $W_{сист} = \frac{2{,}436}{1} = 2{,}436(хв)$.

Середня довжина черги:

$$L_{черги} = 1{,}25^2 \frac{1-1{,}25^3 \cdot (3-3 \cdot 1{,}25+1)}{(1-1{,}25)^2} \cdot 0{,}122 \approx 1{,}56(клієнта).$$

Середній час перебування клієнта у черзі: $W_{очікування} = \frac{1{,}56}{1} = 1{,}56(хв)$.

З отриманих результатів видно, що ймовірність простою касира і середній час очікування клієнта є малими, ймовірність відмови теж незначна і становить приблизно 0,297. Отже, можна вважати, що система працює ефективно.

2. Багатоканальна СМО з обмеженою чргою.

1) На деяку базу в середньому через 30 хв. прибувають автомашини з продукцією. Середній час розвантаження однієї машини становить 1,5 години. Розвантаження виконують дві бригади вантажників. На території бази можуть знаходитися в черзі та очікувати розвантаження не більше 4 автомашин. Визначити показники роботи СМО.

Розв'язок. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 2$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 2/3$;
- кількість сервісів $n = 2$;

- довжина черги $m = 3$;
- відносне навантаження на систему $\rho = \lambda / \mu = 3$;
- приведена інтенсивність потоку $\rho / n = 1{,}5$

Знайдемо ймовірність відсутності клієнтів у системі:

$$p_0 = \left(1 + \frac{3}{1!} + \frac{3^2}{2!} + \frac{3^3}{2 \cdot 2!} \cdot \frac{\left(1 - \left(\frac{3}{2}\right)^4\right)}{\left(1 - \frac{3}{2}\right)} \right)^{-1} = 0{,}0158$$

Ймовірність відмови: $p_{відм} = \frac{\rho^{n+m}}{n^m \cdot n!} \cdot p_0 = \frac{3^6}{2^4 \cdot 2!} \cdot 0{,}0158 = 0{,}36$

Відносна пропускна здатність: $Q = 1 - p_{відмови} = 1 - 0{,}36 = 0{,}64$.

Абсолютна пропускна здатність: $A = \lambda Q = 2 \cdot 0{,}64 = 1{,}28 \left(авто / год \right)$.

Середня кількість зайнятих бригад: $k_{зайн} = \frac{A}{\mu} = \rho Q = 3 \cdot 0{,}64 = 1{,}92$

Середня довжина черги:

$$L_{черги} = \frac{3^3}{2 \cdot 2!} \cdot \frac{1 - (1{,}5)^4 \left(4 + 1 - \frac{4}{2} \cdot 3 \right)}{(1 - 1{,}5)^2} \cdot 0{,}0158 \approx 2{,}6 (авто).$$

Середній час перебування клієнта у черзі: $W_{очікування} = \frac{2{,}6}{2} = 1{,}3 (год)$.

Середня кількість клієнтів у системі: $L_{сист} = 2{,}6 + 1{,}92 = 4{,}52$

Середній час перебування клієнта у системі: $W_{сист} = \frac{4{,}52}{2} = 2{,}26 (год)$.

Задачі для самоконтролю

1. Знайти граничні ймовірності для системи, заданої графом, представленим на рис. 21. *Відповідь*: $p_1 \approx 0{,}706$; $p_2 \approx 0{,}176$; $p_3 \approx 0{,}118$

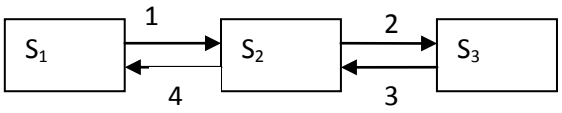


Рис. 21

2. Розглянемо телефонну лінію, на вхід якої надходить простий потік дзвінків з інтенсивністю $\lambda = 0{,}4$ дзвінки/хв. Середня тривалість розмови становить 3 хв. Час розмови має показниковий закон розподілу. Знайти граничні ймовірності станів системи та характеристики обслуговування. *Відповідь*: $p_0 \approx 0{,}455$; $p_{відмови}$=? ; Q=0,455; A=0,182.

3. Є двоканальна найпростіша СМО з відмовами. На її вхід надходить потік заявок з інтенсивністю 4 заявки / год. Середній час обслуговування однієї заявки - 0,8 години. Кожна оброблена заявка приносить дохід 4 грошові одиниці. Експлуатація кожного каналу обходиться 2 грош. од./год. З'ясуйте, вигідно чи невигідно в економічному відношенні збільшити число каналів до трьох. *Відповідь*: при n=2: p_0=0,107; p_2=0,550; Q=0, 450; A=1,8; дохід D=7,2. При n=3: p_0=0,068; p_3=0,371; Q=0, 627; A=2,52; дохід D=10,08. Збільшення числа каналів є вигідним.

4. Магазин відвідує в середньому 90 осіб на годину, які обслуговуються одним касиром. Час на обслуговування однієї особи в середньому становить 1 хв. Черга в зал обслуговування обмежена 5 покупцями. Оцінити ефективність роботи СМО. *Відповідь*: p_0=0,031; $p_{відмови}$=0,354; $L_{очік.} \approx 3{,}457$; $W_{черги}$=2,3. Якщо збільшити чергу до 10, то p_0=0,0039, а $p_{відмови}$=0,036, але для підвищення ефективності краще посадити ще одного касира.

5. На склад в середньому прибуває 3 машини на годину. Розвантаження здійснюють 3 бригади вантажників. Середній час розвантаження машини - 1 год. В черзі на розвантаження може перебувати не більше 4-х машин. Дати оцінку роботи СМО. *Відповідь*: $p_0 \approx 0{,}032$; $p_{відмови}$=0,145; Q=0, 855; $L_{очік.} \approx 1{,}45$; $W_{системи}$=1,34. Можна сказати, що робота системи ефективна.

6. Один оператор, обслуговує в середньому двох клієнтів за одну хвилину. Щогодини в середньому приходять 90 відвідувачів. Провести аналіз роботи СМО. *Відповідь*: $p_0 \approx 0{,}25$; $L_{очік.} \approx 2{,}25$; $L_{системи.} \approx 3$; $W_{системи}$=2.

7. Інтенсивність потоку відвідувачів становить 150 осіб за годину. Є 3 касири, кожен з яких обслуговує в середньому 1 відвідувача за хвилину. Знайти характеристики СМО. *Відповідь*: $p_0 \approx 0{,}0555$; $L_{очік.} \approx 4{,}35$; $L_{системи.} \approx 6{,}35$; $W_{системи}$=2,16.

3. Інші типи СМО
3.1. СМО з обмеженим часом очікування

На практиці зустрічаються задачі з так званими "нетерплячими" замовленнями, які можуть вийти з системи, якщо час очікування перевищує деяке граничне значення. В простих математичних моделях припускають, що замовлення може перебувати у черзі випадковий час, розподілений за показниковим законом з деяким параметром $\nu = 1/\bar{t}_{очікування}$ (замовлення, що перебуває у черзі, може вийти із системи з інтенсивністю ν), де $\bar{t}_{очікування}$ – середній час очікування у черзі, який є обмеженим. Якщо в черзі знаходиться m замовлень, то нтенсивність потоку замовлень, що виходять з черги буде становити $k\nu$. З урахуванням замовлень, що проходять обслуговування, інтенсивність вихідного потоку буде дорівнювати сумі інтенсивності виходу з черги та інтенсивності обслуговування:

$$\mu_{заг} = n\mu + m\nu,$$

де n – кількість каналів обслуговування.

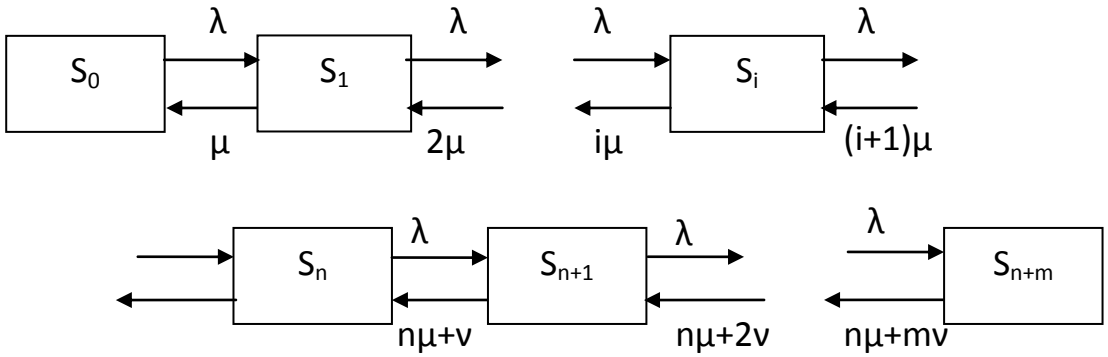


Рис. 22. Граф станів *n*-канальної СМО з обмеженим часом очікування у черзі

Показники ефективності таких систем визначаються за формулами, отриманими для процесу "загибелі та розмноження".

Граничні ймовірності такой системи визначаються за формулами:

$$p_i = \frac{p_0}{i!} \cdot \rho^i,\ i = \overline{1,n}.$$

Якщо враховувати чергу, то для $i > m$ середнє навантаження визначається як: $\rho = \frac{\lambda}{\mu_{заг}} = \frac{\lambda}{n\mu + m\nu}$. Тоді:

$$p_{m+k} = \frac{\rho^n}{n!} \cdot \frac{\lambda^m}{\prod_{j=1}^{m}(n\mu + j\nu)}.$$

З урахуванням того, що всі ймовірності становлять повну групу подій $\left(\sum_{i=0}^{n} p_i = 1\right)$:

$$p_0 = \left(\sum_{i=1}^{n} \frac{\rho^i}{i!} + \frac{\rho^n}{n!} \sum_{k=1}^{\infty} \frac{\lambda^m}{\prod_{j=1}^{m} (n\mu + j\nu)} \right)^{-1},$$

де в практичних розрахунках використовують лише декілька членів нескінченного ряду, так як він швидко спадає.

Для СМО з "нетерплячими" замовленнями поняття ймовірності відмови не має сенсу, бо замовлення може вийти з черги не дочекавшись обслуговування.

Середня кількість замовлень в черзі: $L_{черги} = \frac{\rho}{\beta}$, де $\beta = \frac{\nu}{\mu}$.

Абсолютна пропускна спроможність: $A = \lambda - \nu \cdot L_{черги}$.

Відносна пропускна спроможність: $Q = \frac{A}{\lambda} = 1 - \frac{\nu}{\lambda} \cdot L_{черги}$.

Середня кількість зайнятих каналів: $k = \frac{A}{\mu} = \rho - \beta \cdot L_{черги}$.

Середня кількість вільних каналів: $k_{вільні} = n - k$.

Ефективність обслуговування: $E = k/n$.

Приклади.

1. Пункт обслуговування має три пристрої обслуговування. Інтенсивність потоку замовлень становить 6 осіб за годину. Інтенсивність обслуговування – 3 замовлення за годину. Середня кількість замовлень, що виходять з черги, не дочекавшись обслуговування – 1 замовлення за годину. Визначити абсолютну пропускну спроможність пункту.

Розв'язок. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 6$;
- інтенсивність обслуговування $\mu = 3$;
- кількість сервісів $n = 3$;
- довжина черги $m = 3$;
- відносне навантаження на систему $\rho = \lambda / \mu = 2$;
- приведена інтенсивність потоку $\rho/n = 2/3$.

Знайдемо ймовірність того, що система вільна:

$$p_0 = \left(1 + \frac{2}{1!} + \frac{2^2}{2!} + \frac{2^3}{3!} + \frac{2^3}{3!} \cdot \left(\frac{6}{3 \cdot 3 + 1} + \frac{6^2}{(3 \cdot 3 + 2 \cdot 1)} \right) \right)^{-1} \approx 0{,}13$$

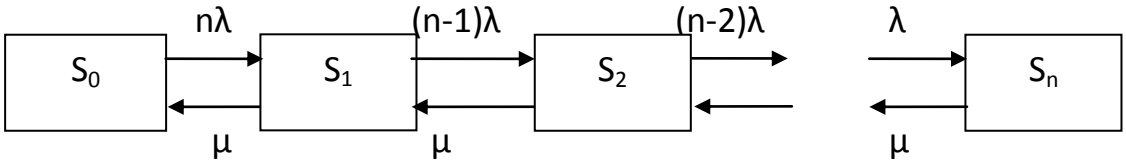
Ймовірність зайнятості всіх сервісів: $p_{зайнятості} = 1 - p_0 = 0{,}87$.

Абсолютна пропускна спроможність: $A = n \cdot p_{зайнятості} = 3 \cdot 0{,}87 = 2{,}61$.

Отже, система обслуговує 2,61 замовлення за годину, що можна вважати непоганим показником. довжина черги $m = 3$;

3.2. Замкнені системи масового обслуговування

В замкнених СМО інтенсивність потоку замовлень залежить від самої системи і джерела замовлень розглядаються не як зовнішні елементи, а як компоненти самої СМО. Для таких систем характерним є те, що джерело замовлень є обмеженим.:



,

Рис.23. Граф станів замкненої СМО

Для замкнених систем щільності ймовірностей дорівнюють:

$$\lambda_{01} = n\lambda,\ \lambda_{12} = (n-1)\lambda, \ldots, \lambda_{n-1,n} = \lambda,$$

$$\lambda_{10} = \lambda_{21} = \ldots = \lambda_{n,n-1} = \mu.$$

Тоді з рівнянь Колмогорова формули граничних ймовірностей будуть мати вигляд:

$$p_0 = \left(1 + n\rho + n(n-1)\rho^2 + \ldots + n(n-1)(n-2)\ldots 1\cdot\rho^n\right)^{-1},$$

$$p_i = n\cdot(n-1)\cdot\ldots\cdot(n-i+1)\rho^i p_0,\ i = \overline{1,n}$$

В якості абсолютної пропускної спроможності виступає ймовірність зайнятості системи (середня кількість несправностей за одиницю часу):

$$p_{зайнятості} = 1 - p_0$$

Якщо виконується обробка замовлення, то абсолютна пропускна спроможність буде становити: $A = (1 - p_0)\cdot\mu$.

Відносна пропускна спроможність не обчислюється, тому що кожне замовлення в результаті буде виконане: $Q = 1$.

Ймовірність того, що обслуговуючий пристрій буде незайнятий:

$$p_{вільний} = 1 - p_{зайнята} = p_0.$$

Середня кількість замовлень буде становити: $L_{сист} = \sum_{i=1}^{n} i p_i = n - \dfrac{1-p_0}{\rho}$

Так як кількість замовлень, що обслуговується дорівнює 1, то середня ймовірність зайнятості сервісу буде становити: $p_{зайнятості} = 1 - p_0$, якщо цю величину відняти від середньої кількості замовлень, то отримаємо середню кількість замовлень, що очікують обслуговування у черзі:

$$L_{черги} = n - \frac{1-p_0}{\rho} = n - (1-p_0)\left(1 + \rho^{-1}\right).$$

Якщо замовлення представляє собою виробничу одиницю з продуктивністю w і, знаючи середню кількість несправних виробничих одиниць, можна оцінити середні втрати групи пристроїв за одиницю часу за рахунок несправностей: $B = w \cdot L_{системи}$

Приклади.

1.Працівник обслуговує групу з трьох верстатів. Кожний верстат виходить з ладу в середньому 2 рази за годину. Процес відновлення становить 10 хвилин. Визначити: ймовірність зайнятості працівника, абсолютну пропускну спроможність, середню кількість несправних верстатів, середню відносну втрату продуктивності за рахунок несправностей.

Розв'язок. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 2$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 1/1/6 = 6$;
- відносне навантаження на систему $\rho = \lambda / \mu = 1/3$;

Обчислимо характеристики системи.

Ймовірність, що всі станки справні:

$$p_0 = \left(1 + 3 \cdot \frac{1}{3} + 3 \cdot 2 \cdot \frac{1}{3^2} + 3 \cdot 2 \cdot 1 \cdot \frac{1}{3^3}\right)^{-1} \approx 0{,}346.$$

Ймовірність зайнятості працівника: $p_{зайн} = 1 - p_0 = 0{,}654$.

Абсолютна пропускна спроможність: $A = 0{,}654 \cdot 6 = 3{,}94$.

Середня кількість несправних верстатів: $L_{черги} = 3 - \frac{0{,}654}{1/3} = 1{,}04$.

Середня відносна втрата продуктивності за рахунок несправностей: $B = 0{,}347$ або 35%.

3.3. Системи з різними дисциплінами підключення каналів обслуговування

Під дисципліною підключення каналів до обслуговування розуміють правила, за якими сервіси підключаються до обслуговування замовлень, при цьому всі замовлення вважаються рівноцінними. Розрізняють два різновиди таких СМО: системи з "взаємодопомогою" між каналами обслуговування і системи "без взаємодопомоги".

Раніше були розглянуті тільки системи, в яких кожне замовлення обслуговувалось одним каналом; одночасне обслуговування одного замовлення кількома каналами не допускалось. Однак зустрічаються СМО, в яких для прискорення процесу обслуговування допускається підключення декількох каналів до роботи над одним замовленням. Наприклад, два або кілька робочих можуть одночасно ремонтувати один верстат або автомашину, по одному літаку може стріляти кілька зенітних знарядів, робота багатопроцесорної ЕОМ

із системою розподіленої пам’яті. В цих випадках виникає задача найкращого розподілу каналів обслуговування.

Зрозуміло, що при "взаємодопомозі" декількох каналів, час обслуговування одного замовлення зменшується. Це означає, що інтенсивність потоку обслуговування є неспадною функцією числа каналів, що підключаються до обслуговування. При цьому, починаючи з деякого критичного числа $K_{кр}$ інтенсивність потоку обслуговування не буде зростати, тобто не можна підключати нескінченно велике число сервісів до виконання однієї роботи. Тому будемо вважати, що інтенсивність потоку обслуговувань зростає лінійно зі збільшенням числа каналів обслуговування до певного рівня, вище якого вона піднятися не може ні при якому збільшенні k. Обмежимося розглядом СМО тільки для лінійної ділянки цієї залежності: тобто будемо вважати, що в кожному конкретному випадку можна знайти число $K_{кр}$, яке буде максимально можливим числом каналів, які об'єднуються для "взаємодопомоги". Можливі дві дисципліни "взаємодопомоги": зосереджена і рівномірна.

У першому випадку при появі одного замовлення його починають обслуговувати всі n каналів одночасно до закінчення процесу обслуговування. Якщо під час обслуговування приходить ще одне замовлення, то воно або отримує відмову, або стає в чергу. Після закінчення обслуговування всі канали перемикаються на інше замовлення (якщо є черга) або чекають його появи (якщо черги немає).

У другому випадку, якщо замовлення приходить в момент часу, коли всі n каналів вільні, то всі вони починають його обслуговування. Якщо в цей час приходить нове замовлення, то частина каналів перемикається на його обслуговування. Якщо за той час, поки обслуговуються ці два замовлення, приходить третє, то на його обслуговування виділяється частина каналів з числа обслуговуючих перших два і т.д. до тих пір, поки не прийде n замовлень, і тоді кожен канал буде обслуговувати тільки одне замовлення. При цьому нове замовлення або отримує відмову, або стає в чергу.

Така дисципліна обслуговування передбачає, що $n < K_{кр}$. Якщо $n > K_{кр}$, то рівномірна взаємодопомога полягає в тому, що якщо всі канали вільні і надходить замовлення, то його починають обслуговувати не n, а m каналів ($m<n<K_{кр}$). Наступне замовлення буде обслуговуватись наступними m вільними каналами. Якщо в системі знаходиться i замовлень, так що $n - i \cdot m < m$, то замовлення, що надходить, приймається на обслуговування, а канали перерозподіляються таким чином, що кожне замовлення обслуговується кількістю каналів $< m$. Якщо замовлення надходить в момент, коли в СМО перебувають n замовлень, то воно або залишає систему, або стає в чергу.

Таким чином, при рівномірній взаємодопомозі можливі два режими: повної взаємодопомоги (перше замовлення обслуговується всіма n каналами) і часткової взаємодопомоги (перше замовлення обслуговується m каналами, причому $m < n$).

Граф станів СМО з повною рівномірною допомогою наведено на рис.24, де видно, що він такий самий як і граф для одноканальнох СМО з чергою.

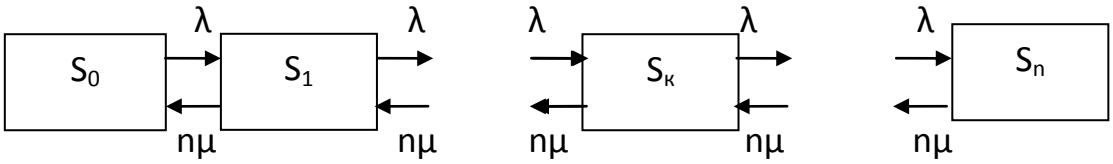


Рис. 24. Граф станів СМО з рівномірною допомогою

Отже система з рівномірною допомогою подібна до одноканальної СМО. Якщо розглядати випадок, коли всі канали обслуговують одне замовлення і лише після завершення обробки приймають на виконання інше, то така система працює як одноканальна з більшою інтенсивністю, але при цьому зменшується її пропусна спрможність. Основні характеристики такої системи будуть визначатись за тими ж формулами, як і для одноканальної СМО при інтенсивності обслуговування:

$$\mu^* = n \cdot \mu,$$

де n – загальна кількість каналів, μ – інтенсивність кожного каналу.

При наявності черги, для системи, коли всі канали працюють як один, середня довжина черги і час перебування у черзі будуть більшими, ніж для простої одноканальної системи, але час перебування у системі буде меншим.

Приклади.

1.Розлянемо триканальну систему $(n=3)$ з необмеженою чергою та інтенсивністю вхідного потоку $\lambda = 4\left(зам./хв\right)$ і середнім часом обслуговування $t_{об.} = 0{,}5(хв.)$. Необхідно встановивити, зважаючи на:

- середню довжину черги;
- середній час очікування у черзі;
- середній час обслуговування замовлення системою;

чи встановлювати взаємодопомогу?

Розв'язок. Розглянемо окремо випадок без взаємодопомоги і з взаємною допомогою.

а) Без взаємодопомоги. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 4$;
- інтенсивність обслуговування $\mu = 1/t_{обсл} = 1/0{,}5 = 2$;
- кількість каналів $n = 3$;
- відносне навантаження на систему $\rho = \lambda / \mu = 2$.

Обчислимо характеристики:

- ймовірність, що система вільна: $p_0 = \left(1 + \frac{2}{1!} + \frac{2^2}{2!} + \frac{2^3}{3!} + \frac{2^4}{3!\cdot(3-2)}\right)^{-1} = \frac{1}{9}$;

- середня довжина черги: $m = \frac{2^4 \cdot 1/9}{3 \cdot 3! \cdot (1/3)^2} = 0{,}889$;

- середній час очікування у черзі: $W_{черги} = m/\lambda = 2/9 = 0{,}222$

- середній час обслуговування замовлення системою:

- $W_{системи} = W_{черги} + t_{об} = 2/9 + 1/2 = 0{,}722$

б) Із взаємодопомогою. Визначимо основні показники СМО:

- інтенсивність замовлень $\lambda = 4$;
- інтенсивність обслуговування $\mu = 3 \cdot 1/t_{обсл} = 3/0{,}5 = 6$;
- кількість каналів $n = 1$;
- відносне навантаження на систему $\rho = \lambda / \mu = 2/3.$

Обчислимо характеристики:

- середня довжина черги: $m = \frac{(2/3)^2}{1/3} = 1{,}333$;

- середній час очікування у черзі: $W_{черги} = \frac{1}{6} \cdot \frac{2/3}{1/3} = 0{,}333$

- середній час обслуговування замовлення системою:

- $W_{системи} = W_{черги} + t_{об} = 1/3 + 1/6 = 0{,}500$

З отриманих результатів видно, що при введені взаємодопомоги хоча час обслуговування і збільшується, але інші показники погіршуються, що не сприяє покращанню ефективності роботи системи.

Розглянемо системи з рівномірною взаємодопомогою з відмовами. Система може перебувати у станах, коли всі канали обслуговують одне замовлення S_1, всі канали обслуговують два замовлення S_2, всі канали обслуговують k замовлень S_k, канали обслуговують n замовлень S_n. Граф станів системи буде мати такий самий вигляд, як і для СМО з рівномірною допомогою. Отже основні характеристики системи для цього випадку будуть обчислюватись за формулами одноканальної СМО з інтенсивністю обслуговування $\mu^* = n \cdot \mu$ та чергою довжиною $m = n - 1.$

Приклади.

1.Розлянемо систему із попереднього прикладу і порівняємо відносну та абсолютну пропускні спроможності для випадків без взаємодопомоги та з розподіленою взаємною допомогою.

Розв'язок.

а) Без взаємодопомоги. Визначимо основні показники СМО:

- відносна пропускна спроможність: $Q = \frac{1 - \rho^3}{1 - \rho^4} = 0{,}79$;

- абсолютна пропускна спроможність: $A = \lambda \cdot Q = 3{,}16$;

- середня кількість зайнятих каналів: $c = A/\mu = 1{,}58.$

б) Із розподіленою взаємною допомогою: $\rho^* = \lambda/(\mu \cdot n) = 2/3$;

- відносна пропускна спроможність: $Q = \dfrac{1-\rho^3}{1-\rho^4} = 0{,}887$;

- абсолютна пропускна спроможність: $A = \lambda \cdot Q = 3{,}51$;

- середня кількість зайнятих каналів: $c = 1{,}76$.

З отриманих результатів видно, що при правильно організованій взаємній допомозі пропускна спроможність системи підвищилась і збільшилась зайнятість каналів.

Розглянемо випадок, коли є рівномірно розподілена взаємна допомога і є черга довжиною m. Інтенсивність обслуговування буде становити $\mu(k) = k \cdot \mu$?, де k – кількість замовлень, що обслуговується всіма n каналами. Граф станів системи наведений на рис. , де S_i, $i \le n$ – кількість замовлень, що обслуговуються всіма n каналами, S_{n+j}, $j \le m$ – загальна кількість замовлень в системі.

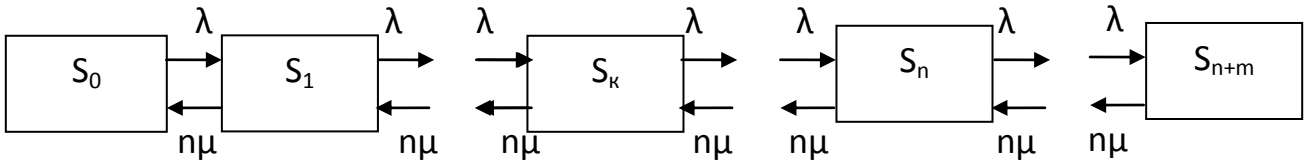


Рис.25. Граф станів СМО з рівномірною допомогою і чергою

З графу станів видно, що система подібна до одноканальної СМО з навантаженням на систему $\rho^* = \lambda / n \cdot \mu$, та чергою. Отже основні характеристики будуть визначатись за наступними формулами:

- $p_{відмови} = \dfrac{(\rho^*)^{n+m}(1-\rho^*)}{1-(\rho^*)^{n+m+1}}$ – ймовірність відмови;

- $Q = \dfrac{1-(\rho^*)^{n+m}}{1-(\rho^*)^{n+m+1}}$ – відносна пропускна спроможність;

- $A = \lambda \cdot Q = \lambda \cdot \dfrac{1-(\rho^*)^{n+m}}{1-(\rho^*)^{n+m+1}}$ – абсолютна пропускна спроможність.

Приклади.

1.Розлянемо триканальну систему $(n=3)$ з чергою $m=2$ та інтенсивністю вхідного потоку $\lambda = 4\left(зам./хв\right)$ і середнім часом обслуговування $t_{об.} = 0{,}5(хв.)$. Порівняти характеристики системи для випадку наявності взаємодопомоги та при її відсутності.

Розв'язок. Визначимо основні показники системи:

- $\lambda = 4$ – інтенсивність вхідного потоку;

- $\mu = 2$ – інтенсивність вихідного потоку;

- навантаження на систему:

- $\rho = 2$ – без взаємодопомоги;

- $\rho^* = 2/3$ – із взаємодопомогою.

а) Без взаємодопомоги:

- відносна пропускна спроможність: $Q = \frac{1-\rho^5}{1-\rho^6} = 0{,}49$;

- абсолютна пропускна спроможність: $A = \lambda \cdot Q = 1{,}96$.

б) З рівномірною взаємодопомогою:

- відносна пропускна спроможність: $Q = \frac{1-(\rho^*)^5}{1-(\rho^*)^6} = 1{,}42$;

- абсолютна пропускна спроможність: $A = \lambda \cdot Q = 5{,}68$.

Очевидно, що при введені взаємодопомоги показники системи покращуються.

3.4. СМО з помилками обслуговування

На практиці зустрічаються випадки, коли замовлення обслуговується не повністю, а з деякою ймовірністю $p \neq 1$, що обумовлена «браком» в роботі системи. Наприклад, неправильне з'єднання телефонної станції з абонентом,

помилка, пропущена коректором в тексті, помилка у платіжній квитанції, невірне повідомлення в розкладі руху транспорту.

У розімкнених системах поява помилки практично не впливає на потік замовлень: якщо число джерел замовлень є значним, то з появою помилки інтенсивність потоку практично не змінюється. Для розімкнених систем врахування помилки виражається лише в тому, що відносна та абсолютна пропускна спроможності системи зменшуються за рахунок того, що вона домножується на $p < 1$, де p – ймовірність помилки. Інші характеристики системи (час очікування, довжина черги і т.д.) не змінюються.

Для замкнених систем замовлення оброблене з помилкою знову стає у чергу, а отже збільшується навантаження на СМО.

В якості прикладу СМО з помилкою розглянемо одного працівника, який обслуговує n верстатів. Нехай інтенсивність потоку несправностей для нього становить λ, середній час налагодження $t_{обсл.} = 1/\mu$, з ймовірністю p ремонт верстату завершується успішно і верстат знову починає працювати, з ймовірністю $1-p$ ремонт невдалий і верстат повертається у чергу на обслуговування. Необхідно визначити граничні ймовірності.

Визначимо стани системи:

- S_0 – всі верстати справні;

- S_1 – один верстат насправний, він ремонтується, черги немає;

- S_k – k верстатів несправні, один ремонтується, а $k-1$ очікують у черзі;

- S_n – всі n верстатів несправні, один ремонтується, $n-1$ утворюють чергу.

Граф станів системи наведений на рис. 26.

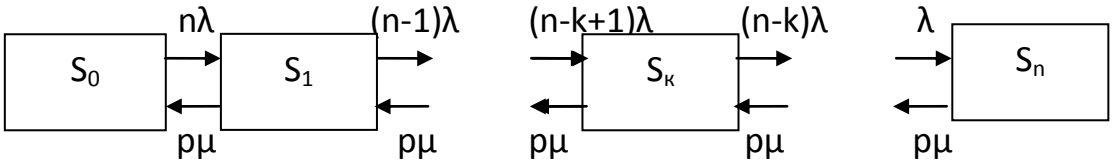


Рис.26. Граф станів СМО з помилкою

Граф станів даної системи подібний до графу замкненої системи (рис. 23) з тією відмінністю, що інтенсивність вихідного потоку буде становити $\mu^* = p \cdot \mu$. Отже характеристики СМО з помилками можуть бути обчислені за формулами замкненої системи для μ^*.

Приклади.

1.Робітник обслуговує три верстати. Верстат виходить з ладу у середньому 2 рази за годину. Ремонт займає 10 хв., причому ймовірність виправлення поломки 2/3. Визначити характеристики системи: ймовірність зайнятості, абсолютну пропускну спроможність, середню кількість несправних верстатів.

Розв'язок. Система, що розглядається, є замкненою. Визначимо основні показники системи за годину:

- $\lambda = 2$ – інтенсивність вхідного потоку;
- $\mu = 6$ – інтенсивність вихідного потоку без помилки;
- $\mu^* = p \cdot \mu = 4$ інтенсивність вихідного потоку з урахуванням помилки;

$\rho^* = \lambda / \mu^* = 1/2$ - навантаження на систему.

Обчислимо основні характеристики системи:

- $p_0 = \left(1 + 3 \cdot \frac{1}{2} + 3 \cdot 2 \cdot \frac{1}{2^2} + 3 \cdot 2 \cdot \frac{1}{2^3}\right)^{-1} \approx 0{,}211$ – ймовірність, що всі верстати працюють;
- $p_{зайн.} = 1 - p_0 = 0{,}789$ – ймовірність відмов;
- $A = 0{,}789 \cdot 4 = 3{,}16$ – абсолютна пропускна спроможність;
- $Q = 3 - \frac{0{,}789}{1/2 = 1{,}42}$ – відносна пропускна спроможність.

Існують системи (наприклад, при наявності людського фактору), де характер обслуговування залежить від довжини черги: коли черга збільшується, то сервіс починає поспішати, що збільшує ймовірність помилки.

В якості прикладу розглянемо одноканальну СМО з n каналами (робітник обслуговує n верстатів). Нехай у спокійному режимі роботи час ремонту становить $t_{обсл.}$. Якщо в черзі з'являються пошкоджені верстати, робітник починає поспішати і інтенсивність потоку обслуговування зростає. Ця

інтенсивність залежить від ймовірності помилки, яка є функцією від довжини черги: $\mu^*(k)=\mu(k)\cdot p(k),\ k=\overline{0,n-1}$ (рис. 27).

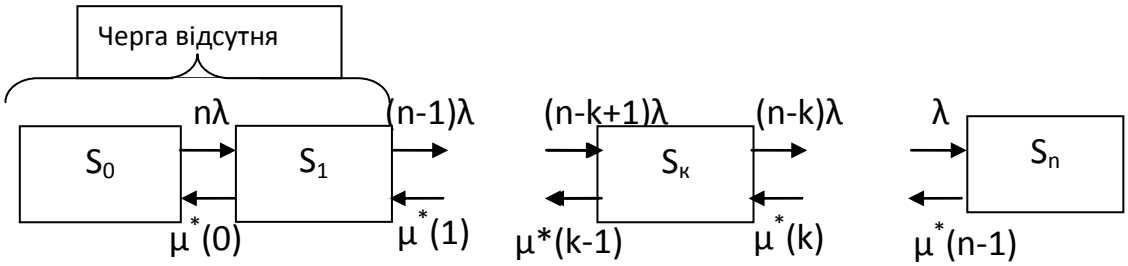


Рис.27. Граф станів СМО з помилкою, що «поспішає»

Взявши формули граничних ймовірностей схеми «загибелі-розмноження» за основу, отримаємо:

$$p_1=\frac{n\cdot\lambda}{\mu^*(0)}p_0,\quad p_2=\frac{n\cdot(n-1)\cdot\lambda^2}{\mu^*(0)\cdot\mu^*(1)}p_0,\quad ,\ p_k=\frac{n\cdot(n-1)\cdot\ldots\cdot(n-k-1)\cdot\lambda^k}{\mu^*(0)\cdot\mu^*(1)\cdot\ldots\cdot\mu^*(k-1)}p_0,\ldots$$

$$p_n=\frac{n\cdot(n-1)\cdot\ldots\cdot1\cdot\lambda^n}{\mu^*(0)\cdot\mu^*(1)\cdot\ldots\cdot\mu^*(n-1)}p_0,$$

$$p_0=\left(1+\frac{n\cdot\lambda}{\mu^*(0)}+\frac{n\cdot(n-1)\cdot\lambda^2}{\mu^*(0)\cdot\mu^*(1)}+\ldots+\frac{n\cdot(n-1)\cdot\ldots\cdot1\lambda^2}{\mu^*(0)\cdot\mu^*(1)\cdot\ldots\cdot\mu^*(n-1)}\right)^{-1}$$

3.5. Моделі систем з непуассонівськими потоками замовлень

В попередніх розділах розглядались випадки, коли процес, що протікає в СМО є марківським. При отриманні характеристик також вважалось, що потік має постійну інтенсивність (є найпростішим), коли інтервали часу між замовленнями мають показниковий закон розподілу (рис. 28):

$$f(t)=\lambda\cdot e^{-\lambda t},\ t>0.$$

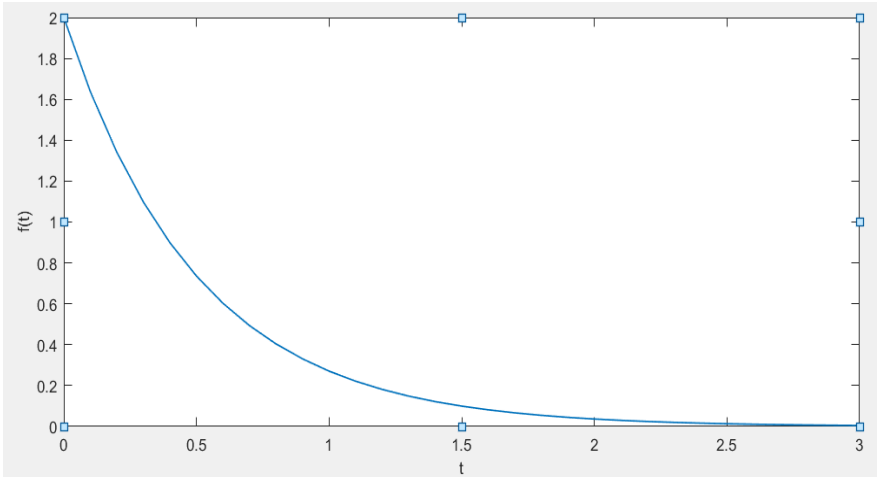


Рис. 28. Графік пуаассонівського потоку

У практичних задачах часто зустрічаються випадки, коли проміжки часу між надходженням замовлень мають інші закони розподілу. Зрозуміло, що для таких випадків розглянуті раніше випадки СМО не підходять, для їх опису необхідно застосовувати більш складний математичний апарат, а отже і результати будуть представлятись складними математичними виразами. Тому формули для характеристик таких СМО отримані тільки для найпростіших випадків, які представлені нижче.

3.5.1. СМО з відмовами.

Б. А. Севастьяновим була розглянута система Ерланга з пуассонівським потоком замовлень з інтенсивністю λ та потоком обслуговування, у якому закон розподілу часу обслуговування одного замовлення є довільним. Якщо позначити цей закон як $f(t_{обсл.})$, то середній час обслуговування (математичне сподівання) буде визначатись за наступною формулою:

$$\bar{t}_{обсл.} = 1/\mu = \int_0^{\infty} t_{обсл.} \cdot f(t_{обсл.}) dt_{обсл.}.$$

Б. А. Севастьянов довів, що формули Ерланга для визначення граничних ймовірностей станів справедливі і для цього випадку:

$$p_k = \frac{\rho^k}{k!} p_0,\ k = \overline{0,n},\ \ p_0 = \left(1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \ldots + \frac{\rho^n}{n!}\right)^{-1},\ \text{де}\ \rho = \frac{\lambda}{\mu} = \lambda \cdot \bar{t}_{обсл.}.$$

3.5.2. Одноканальна СМО з очікуванням

Розглянемо одноканальну СМО ($n=1$) з необмеженою чергою ($m=\infty$). На вхід системи надходить пуассонівський потік замовлень з інтенсивністю λ, закон розподілу часу обслуговування є довільним $f(t)$ з математичним сподіванням $t_{обсл.} = 1/\mu$ та середньоквадратичним відхиленням $\sigma_{toобс} = 1$.

Введемо величину: $v = \frac{\sigma_{toобс.}}{t_{обсл.}}$, яка називається коефіцієнтом варіації часу обслуговування і характеризує величину розсіювання часу обслуговування відносно середнього значення. Для показникового закону розподілу $v = 1$

Для одноканальної СМО з найпростішим вхідним потоком і довільно розподіленим часом обслуговування доведено, що середня довжина черги визначається за формулою:

$$L_{черги} = \frac{\rho^2 \cdot (1+v^2)}{2 \cdot (1-\rho)},\ \text{де}\ \rho = \frac{\lambda}{\mu},\quad (1)$$

а середній час очікування у черзі:

$$W_{черги} = \frac{\rho^2 \cdot (1+v^2)}{2 \cdot \lambda \cdot (1-\rho)} = \frac{L_{черги}}{\lambda}. \qquad (2)$$

Формули (1)-(2) називаються формулами Поллачека-Хінчина.

Якщо час обслуговування не випадковий і дорівнює математичному сподіванню: $t_{обсл.} = 1/\mu$, то $\sigma_{обсл.} = 0$ і $v = 0$ (випадок регулярного потоку), тоді формули (1)-(2) спрощуються:

$$L_{черги} = \frac{\rho^2}{2 \cdot (1-\rho)}, \quad W_{черги} = \frac{\rho^2}{2 \cdot \lambda \cdot (1-\rho)}.$$

Видно, що середня довжина черги і середній час очікування у черзі в два рази менші, ніж у випадку показникового закону розподілу.

Приклади.

1. Нехай потік клієнтів до сервісного центру має інтенсивність $\lambda = 2 год$. Середній час обслуговування $t_{обсл} = 20 хв.$ із середньоквадратичним відхиленням $\sigma_{toобс} = 8 хв$. Визначити середню кількість клієнтів та середній час очікування у черзі.

Розв'язок. Визначимо основні показники системи у год.

- $\lambda = 2$ – інтенсивність вхідного потоку;
- $\mu = 3$ – інтенсивність вихідного потоку;
- $\rho = \lambda / \mu = 2/3$ - навантаження на систему;
- $v = \sigma_{toобс.} / t_{обсл.} = 8/20 = 0,4 \; v =$

Обчислимо основні характеристики системи:

- середня довжина черги: $L_{черги} = \frac{\rho^2 \cdot (1+v^2)}{2 \cdot (1-\rho)} = \frac{(2/3)^2 \cdot (1+0,4^2)}{2 \cdot (1-2/3)} \approx 0,77;$
- середній час очікування у черзі: $W_{черги} = \frac{\rho^2 \cdot (1+v^2)}{2 \cdot \lambda \cdot (1-\rho)} \approx 0,385;$
- середня кількість клієнтів в системі: $L_{системи} = L_{черги} + \rho = 0,77 + 2/3 \approx 1,437.$

2. Нехай є шість типів повідомлень з часом обслуговування 15, 20, 25,30, 35, 40 і 300. Число повідомлень кожного типу однакове. Стандартне відхилення для часу трохи вище свого середнього. Значення останнього часу обслуговування набагато більше інших. Це призведе до того, що повідомлення будуть перебувати в черзі значно довше, ніж, якби час обслуговування був однаковий. У такому випадку при проектуванні доцільно вжити заходів для зменшення довжини черги. Наприклад, якщо зазначені цифри пов'язані з довжинами повідомлень, то, можливо, дуже довгі повідомлення варто розділити на частини.

Розв'язок. Визначемо кількісні оцінки системних показників для початкового варіанта організації інформаційного обміну і для двох варіантів, в яких довжини повідомлень вирівняні.

Обчислимо основні характеристики та величини, що входять в формулу для середньої довжини черги:

- математичне спдівання: $\bar{t}_{обсл} = \frac{15+20+25+30+35+40+300}{7} = 66,43;$

- середньоквадратичне відхилення:

$$\sigma = \sqrt{M\left(t_{обсл}^2\right) - M\left(t_{обсл}\right)^2} = \sqrt{\frac{15^2 + 20^2 + 25^2 + 30^2 + 35^2 + 40^2 + 300^2}{7} - 66{,}43^2} = 95{,}68$$

;

- коефіцієнт варіації часу обслуговування: $v = \sigma_{toобс.} \Big/ t_{обсл.} = \frac{95{,}68}{66{,}43} = 1{,}44$

Будемо вважати, що середня тривалість для всієї групи повідомлень становить 600 одиниць часу, тоді: $\rho = \frac{15 + 20 + 25 + 30 + 35 + 40 + 300}{600} = 0{,}775.$

- середня довжина черги: $L_{черги} = \frac{\rho^2 \cdot \left(1 + v^2\right)}{2 \cdot \left(1 - \rho\right)} = \frac{0{,}775^2 \cdot \left(1 + 1{,}44^2\right)}{2 \cdot \left(1 - 0{,}755\right)} \approx 4{,}10;$

Обчислимо цей показник, якщо останнє повідомлення розбивається на дві однакові частини:

- математичне спдівання:

$$\bar{t}_{обсл} = \frac{15 + 20 + 25 + 30 + 35 + 40 + 150 + 150}{8} = 58{,}13;$$

- середньоквадратичне відхилення:

$$\sigma = \sqrt{\frac{15^2 + 20^2 + 25^2 + 30^2 + 35^2 + 40^2 + 150^2 + 150^2}{8} - 58{,}13^2} = 53{,}56;$$

- коефіцієнт варіації часу обслуговування: $v = \frac{53{,}56}{58{,}13} = 0{,}92$

- середня довжина черги: $L_{черги} = \frac{0{,}775^2 \cdot \left(1 + 0{,}92^2\right)}{2 \cdot \left(1 - 0{,}92\right)} \approx 2{,}47;$

Якщо останнє довге повідомлення розбити на три частини, то $L_{черги} \approx 1{,}94.$

З отриманих результатав видно ефект від розбиття довгого повідомлення на частини.

Для більш складних випадків не пуассонівських потоків (наприклад, багатоканальних систем, систем з особливостями обслуговування) аналітичні формули досить складні або взагалі їх не вдається отримати.

Розглянемо приклад одноканальної СМО з необмеженою чергою.

Приклад.

1. На вхід одноканальної СМО (*n=1*) без відмов ($m = \infty$) надходить потік замовлень з інтенсивністю λ. Час обслуговування $t_{обсл.}$ має закон розподілу Ерланга 2-го порядку з математичним сподіванням $1/\mu$ і представляє собою суму двох незалежних випадкових величин з однаковим показниковим законом розподілу, параметри яких позначимо μ'. Згідно до теореми суми математичних сподівань отримаємо: $1/\mu = 2/\mu'$ або $\mu' = 2 \cdot \mu$.

Час обслуговування такого потоку можна представити як суму двох незалежних випадкових величин $t_{обсл.}^{(1)}$ і $t_{обсл.}^{(2)}$ з показниковим законом розподілу і

параметром 2μ. Величини $t_{обсл.}^{(1)}$ і $t_{обсл.}^{(2)}$ можна представити як дві послідовні «фази» процесу обслуговування.

Визначимо стани системи:

- S_0 – система вільна, замовлень немає;

- $S_{1,1}$ – в системі одне замовлення і знаходиться в першій фазі обслуговування, черги немає;

- $S_{1,2}$ – в системі одне замовлення і знаходиться в другій фазі обслуговування, черги немає;

- $S_{2,1}$ – в системі два замовлення, перше обробляється (перша фаза), а друге стоїть у черзі;

- $S_{2,2}$ – в системі два замовлення, перше обробляється (друга фаза), а друге стоїть у черзі;

……..

- $S_{k,1}$ – в системі k замовлень, перше обробляється (перша фаза), інші стоять у черзі;

- - $S_{k,2}$ – в системі k замовлень, перше обробляється (друга фаза), інші стоять у черзі;

……

Із стану S_0 в стан $S_{1,1}$ систему переводить потік замовлень з інтенсивністю λ, з $S_{1,1}$ в $S_{1,2}$ – потік замовлень з інтенсивністю 2μ і. т.д.

Граф станів такої системи наведений на рис. 29.

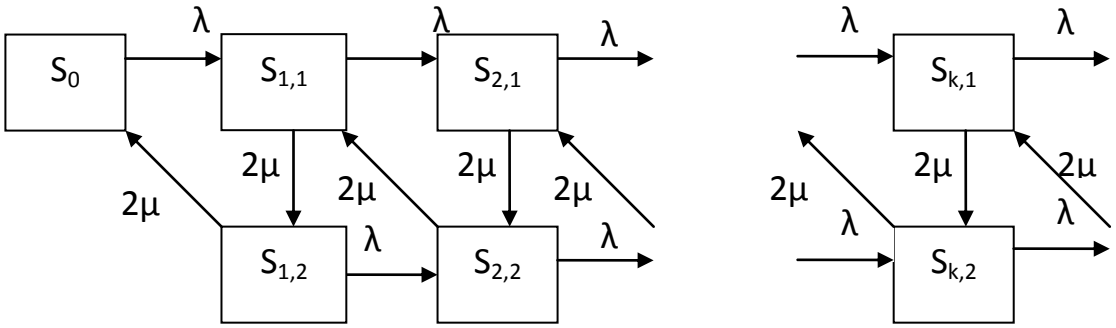


Рис. 29. Граф станів одноканальної СМО із законом Ерланга 2-го порядку

Спираючись на граф станів, запишемо систему лінійних алгебраїчних рівнянь для ймовірностей станів:

$$
\begin{cases}
\lambda \cdot p_0 = 2\mu \cdot p_{1,2} \\
(\lambda + 2\mu) \cdot p_{1,1} = \lambda \cdot p_0 + 2\mu \cdot p_{2,2} \\
(\lambda + 2\mu) \cdot p_{1,2} = 2\mu \cdot p_{1,1} \\
(\lambda + 2\mu) \cdot p_{2,1} = \lambda \cdot p_{1,1} + 2\mu \cdot p_{3,2} \\
(\lambda + 2\mu) \cdot p_{2,2} = \lambda \cdot p_{1,2} + 2\mu \cdot p_{2,1} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\lambda + 2\mu) \cdot p_{k,1} = \lambda \cdot p_{k-1,1} + 2\mu \cdot p_{k+1,2} \\
(\lambda + 2\mu) \cdot p_{k,2} = \lambda \cdot p_{k-1,2} + 2\mu \cdot p_{k,1} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{cases}
$$

Ввівши позначення $\rho = \lambda / \mu$, отримаємо:

$$
\begin{cases}
\rho \cdot p_0 = 2 \cdot p_{1,2} \\
(\rho + 2) \cdot p_{1,1} = \rho \cdot p_0 + 2 \cdot p_{2,2} \\
(\rho + 2) \cdot p_{1,2} = 2 \cdot p_{1,1} \\
(\rho + 2) \cdot p_{2,1} = \rho \cdot p_{1,1} + 2 \cdot p_{3,2} \\
(\rho + 2) \cdot p_{2,2} = \rho \cdot p_{1,2} + 2 \cdot p_{2,1} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
(\rho + 2) \cdot p_{k,1} = \rho \cdot p_{k-1,1} + 2 \cdot p_{k+1,2} \\
(\rho + 2) \cdot p_{k,2} = \rho \cdot p_{k-1,2} + 2 \cdot p_{k,1} \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{cases}
$$

Отже, система з потоком Ерланга методом вкладених ланцюгів зведена до вкладених марківських ланцюгів. При збільшенні порядку потоку Ерланга збільшується і кількість псевдостанів системи і її розв'язок стає досить складним. Існують певні вказівки на те, як можна розв'язати таку систему при конкретних значеннях λ і μ.

Перш за все оцінюється максимально можлива довжина черги, для цього закон розподілу часу обслуговування можна прийняти як показниковий. Це дає змогу обмежити розмірність системи, зробивши її скінченною. Для отримання розв'язку скінченної системи застосовують чисельні методи, найчастіше ітераційні, де за початкові значення приймаються значення ймовірностей, отримані для показникового закону розділені на порядок потоку Ерланга.

У випадках, коли розв'язання системи є складним, для моделювання немарківських процесів застосовують метод статистичних іспитів (метод Монте-Карло).

Контрольні питання та задачі для самоконтролю

1. Дайте характеристику СМО з обмеженим часом очікування.
2. Представте граф станів системи з обмеженим часом очікування. Як визначаються граничні ймовірності для цього випадку?
3. Які СМО називають замкненими. Як визначаються граничні ймовірності для замкнених систем?
4. Як визначається абсолютна пропускна спроможність замкненої СМО?
5. Що розуміють під дисципліною підключення каналів?
6. Дайте характеристику СМО із "взаємодопомогою" між каналами обслуговування і "без взаємодопомоги".
7. Які особливості обчислення характеристик системи із "взаємодопомогою"?
8. Наведіть приклади СМО «з помилкою в обслуговуванні».
9. Як обчислюються характеристики СМО з помилками?

Дописати

4. Багатофазні системи та мережі СМО
4.1. Моделі багатофазних систем

Багатофазними називаються СМО, які складаються з декількох послідовно з'єднаних окремих підсистем масового обслуговування, причому вхідний потік кожної наступної підсистеми є вихідним потоком попередньої.

Будемо вважати, що виконуються умови стабілізації черги, тобто $m=\infty$. Багатофазні СМО призначені для виконання деякого повного циклу обслуговування одного замовлення, коли з різних причин цей цикл доцільніше розділити на кілька фаз (наприклад, обслуговування в магазині, де покупці спочатку стоять в черзі до кас, а потім обслуговуються у продавців). Випадок введення додаткових фаз описаний в розділі «Метод псевдостанів». Типовим прикладом багатофазної системи може бути послідовний процес обробки деталі. Точний розрахунок характеристик таких СМО можливий тільки в разі, якщо всі потоки замовлень є пуассонівськими, а час обслуговування має експоненціальний закон розподілу. В інших випадках можливий лише наближений розрахунок характеристик СМО.

В якості підсистем СМО на кожній фазі обслуговування будемо розглядати одно- і багатоканальні системи без черги або з чергами, в яких число місць не обмежене ($\rho=\lambda/\mu<1, \rho^{*}=\rho/n<1$, де n – кількість каналів).

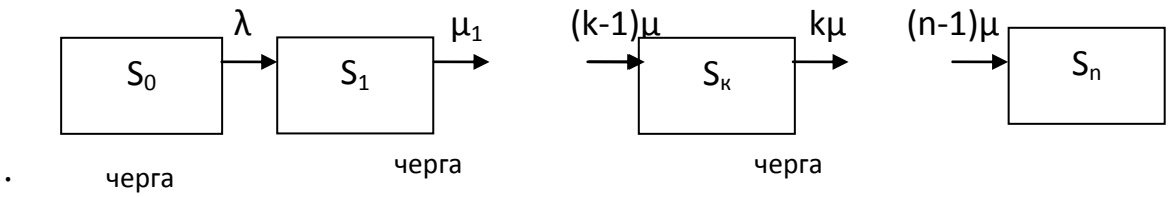


Рис.30. Граф багатоканальної СМО

Можливі стани багатофазної одноканальної СМО можуть бути описані вектором $\overline{S}=(k_1\alpha_1,k_2\alpha_2,...,k_n\alpha_n)$, де k_i – довжина черги на вході i-го сервісу.

$$\alpha_i=\begin{cases}0, & i-й\ прилад\ \ вільний\\ 1, & i-й\ прилад\ \ зайнятий\end{cases}.$$

Якщо черга перед кожною фазою не допускається, то система буде називатися системою з нульовою місткістю блоків очікування. Кожна фаза може бути зайнята обслуговуванням або вільна. Оскільки перед фазою черга не допускається, то приймається, що перша фаза обслуговування заблокована, якщо друга фаза не готова до прийому замовлення з тієї причини, що в ній не закінчено обслуговування. Приймається також, що якщо перша фаза зайнята, то чергове замовлення отримує відмову. В системі можуть бути наступні стани: "фаза вільна" – 0, "фаза зайнята" – 1, "фаза заблокована" – *b*.

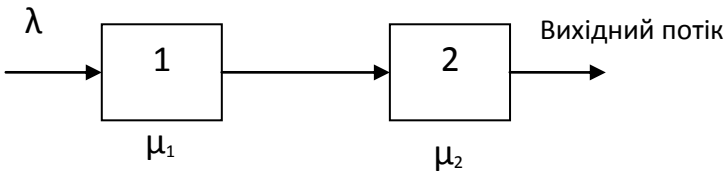


Рис. 31. Двофазна одноканальна СМО без черги

Позначимо стан першої фази через *i*, другої – *j*. Тоді множина всіх можливих станів це всі можливі перестановки пар з нулів і одиниць та врахування стану з блокуванням:

$$\{(i,j)\}=\{(0,0);(1,0);(0,1);(1,1);(b,1)\}.$$

Враховуючи, що на вході системи пуассонівський потік, а час обслуговування має експоненціальний розподіл, для системи можна записати рівняння Колмогорова відносно ймовірностей станів $p_{ij}(t)$:

$$\begin{cases} \dfrac{dp_{00}(t)}{dt}=-\lambda\cdot p_{00}(t)+\mu\cdot p_{01}(t) \\ \dfrac{dp_{01}(t)}{dt}=-(\lambda+\mu)\cdot p_{01}(t)+\mu\cdot p_{10}(t)+\mu\cdot p_{b1}(t) \\ \dfrac{dp_{10}(t)}{dt}=\lambda\cdot p_{00}(t)-\mu\cdot p_{10}(t)+\mu\cdot p_{11} \\ \dfrac{dp_{11}(t)}{dt}=-\lambda\cdot p_{01}(t)-2\mu\cdot p_{11}(t) \\ \dfrac{dp_{b1}(t)}{dt}=\mu\cdot p_{11}(t)-\mu\cdot p_{b1}(t) \end{cases},$$

де початкові умови визначаються як: $p_{00}=1,\ p_{10}=p_{01}=p_{11}=p_{b1}=0$.

В матричній формі цю систему можна записати як: $\dfrac{d\bar{p}(t)}{dt}=A\cdot\bar{p}(t)$, де

$\bar{p}(t)$ – вектор з п’яти елементів, що відповідають ймовірностям станів *i, j*;

$$A=\begin{pmatrix} -\lambda & \mu & 0 & 0 & 0 \\ 0 & -(\mu+\lambda) & \mu & 0 & \mu \\ \lambda & 0 & -\mu & \mu & 0 \\ 0 & \lambda & 0 & -2\mu & 0 \\ 0 & 0 & 0 & \mu & -\mu \end{pmatrix}$$ – матриця коефіцієнтів.

Для визначення граничних ймовірностей ліва частина приймається рівною нулю.

Якщо у трифазній системі без черг стани пешої фази позначити *i*, другої – *j*, а третьої – *k*, то множина всіх можливих станів буде мати вигляд:

$$\{i,j,k\}=\begin{Bmatrix} (0,0,0);(1,0,0);.(0,1,0);(0,0,1);(1,0,1);(0,1,1);(1,1,1); \\ (1,1,0);(b,1,0);(1,b,1);(b,b,1);(b,1,1);(0,b,1) \end{Bmatrix}.$$

Система рівнянь Колмогорова буде представляти собою систему звичайних лінійних диференціальних рівнянь розмірності 13 х 13.

Розглянемо більш складний випадок, коли між фазами може існувати черга (рис. 32). Вважається, що коли на вході i-го сервісу є черга, то якщо сервіс звільняється, до нього миттєво надходить замовлення з черги.

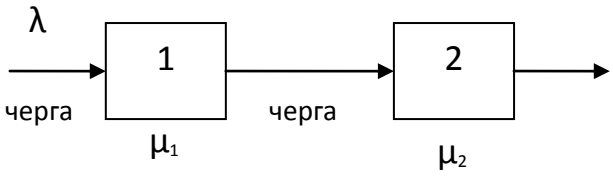


Рис. 32. Двофазна одноканальна СМО з чергою

Стани системи будуть описуватись чотирьохмісними векторами, де парні місця представляють сервіси, а непарні – віповідні черги до них.

Визначимо стани системи:

- $S_0(0,0,0,0)$ – система вільна;
- $S_1(0,1,0,0)$ – в системі одне замовлення, що обслуговується першим сервісом;
- $S_2(,1,1,0,0)$ – в системі два замовлення, перше обслуговується першим сервісом, а друге у черзі до нього і т. д.

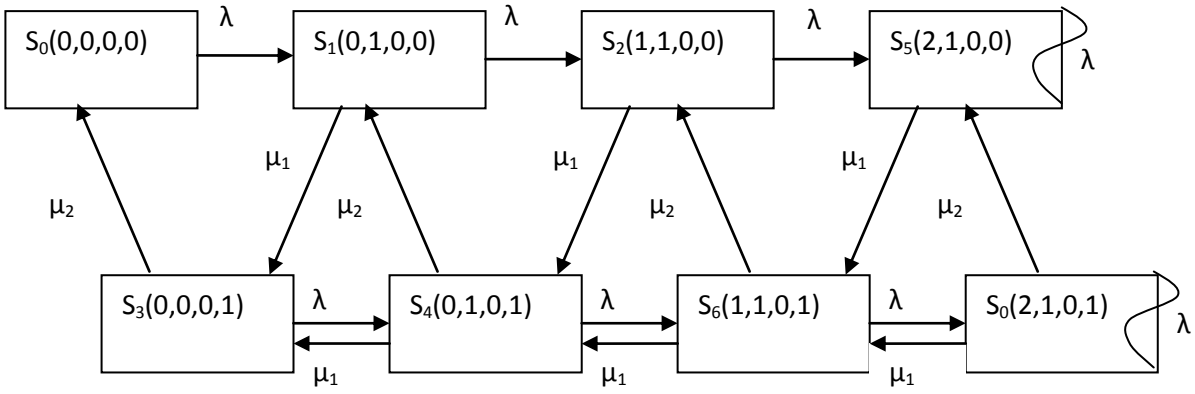


Рис. 33. Граф станів двофазної одноканальної моделі

За графом станів складемо систему рівнянь Колмогорова для граничних ймовірностей:

$$\begin{cases} -\lambda \cdot p_0 + \mu_2 \cdot p_3 = 0 \\ \lambda \cdot p_0 - \mu_1 \cdot p_1 - \lambda \cdot p_1 + \mu_2 \cdot p_4 = 0 \\ \lambda \cdot p_1 - \mu_1 \cdot p_2 - \lambda \cdot p_2 + \mu_2 \cdot p_6 = 0 \\ \mu_2 \cdot p_1 - \mu_2 \cdot p_3 - \lambda \cdot p_3 + \mu_1 \cdot p_4 = 0 \\ \mu_1 \cdot p_2 + \lambda \cdot p_3 - \lambda \cdot p_4 - \mu_1 \cdot p_4 - \mu_2 \cdot p_4 + \mu_1 \cdot p_6 = 0 \\ \lambda \cdot p_2 - \mu_1 \cdot p_3 + \mu_2 \cdot p_7 = 0 \\ \mu_1 \cdot p_5 + \lambda \cdot p_4 - \mu_1 \cdot p_6 - \mu_1 \cdot p_6 - \mu_2 \cdot p_6 + \mu_1 \cdot p_7 = 0 \\ \lambda \cdot p_6 - \mu_1 \cdot p_7 - \mu_2 \cdot p_7 = 0 \end{cases}$$

В цій системі будь-яке рівняння можна замінити умовою, що ймовірності утворюють повну групу подій: $p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 = 1$.

Розв'язавши систему рівнянь, знайдемо граничні ймовірності станів і на їх основі можна визначити показники ефективності системи:

- ймовірність простою системи: p_0;
- ймовірність простою першого приладу: $p_0 + p_3$ (друга координата =0);
- ймовірність простою другого приладу: $p_0 + p_1 + p_2 + p_5$ (четверта координата =0);
- ймовірність повної відмови: $p_5 + p_7$ (сума ймовірностей станів з петлями);
- ймовірність відмови з частковою обробкою дорівнює сумі зважених за інтенсивністю всіх можливих виходів з тих станів, для яких відмова обумовлена зайнятістю одного приладу (але не першого):

$$p_{відмови\,част} = \frac{\mu_1}{\lambda + \mu_1 + \mu_2} p_6 + \frac{\mu_1}{\mu_1 + \mu_2} p_7 .$$

Різні підсистеми в багатофазній СМО можна розглядати незалежно одну від одної як системи, що знаходяться під впливом одного і того ж потоку заявок інтенсивності i (i = 1, ..., k). Стан СМО характеризується ймовірністю деякого розподілу замовлень по кожній з підсистем:

$$p(n_1,\ldots,n_k) = \prod_{i=1}^{k} (1-\rho)\rho_i^{n_i} .$$

За умови нормування:

$$\sum_{n_1=0}^{\infty} \ldots \sum_{n_k=0}^{\infty} \prod_{i=1}^{k} (1-\rho_i)\rho_i^{n_i} .$$

Застосовуючи формулу для середнього числа замовлень в окремій СМО, отримаємо вираз для математичного сподівання числа замовлень багатофазової системи:

$$c = \sum_{i=1}^{k} \frac{\rho_i}{1-\rho_i} .$$

Окремі підсистеми (фази) СМО можна розраховувати за формулами для систем відповідного типу, які розглянуті раніше.

4.2. Моделі мереж масового обслуговування

Мережею масового обслуговування називаються складні послідовно-паралельні з'єднання окремих СМО. Розрізняють замкнені і розімкнені мережі. Для замкненої ймовірнісної мережі не існує зовнішніх джерел замовлень, тобто в ній завжди знаходиться одна і таж кількість замовлень. Для розімкненої мережі є джерела замовлень і стоки замовлень.

Кожний окремий вузол мережі є розімкненою СМО і відображає функціонально самостійну частину реальної системи. Необхідно зазначити, що чим більше окремих СМО утворює мережу і складнішою є структура, тим більш точними є результати, отримані на заміні реальних потоків простішими, в силу граничної теореми для сумарних потоків.

Приклади.

1. Розглянемо організацію обчислювального процесу сучасною ЕОМ. Схема окремих блоків системи представлена на рис. 34, де:

- C_1 – процесор ОЗП та черга до нього O_1 (СМО-1);
- C_2 – селекторний канал із зовнішнім накопичувачем O_2 (СМО-2);
- C_3 – мультиплексний канал з пристроями вводу-виводу та чергою O_3 (СМО-3).

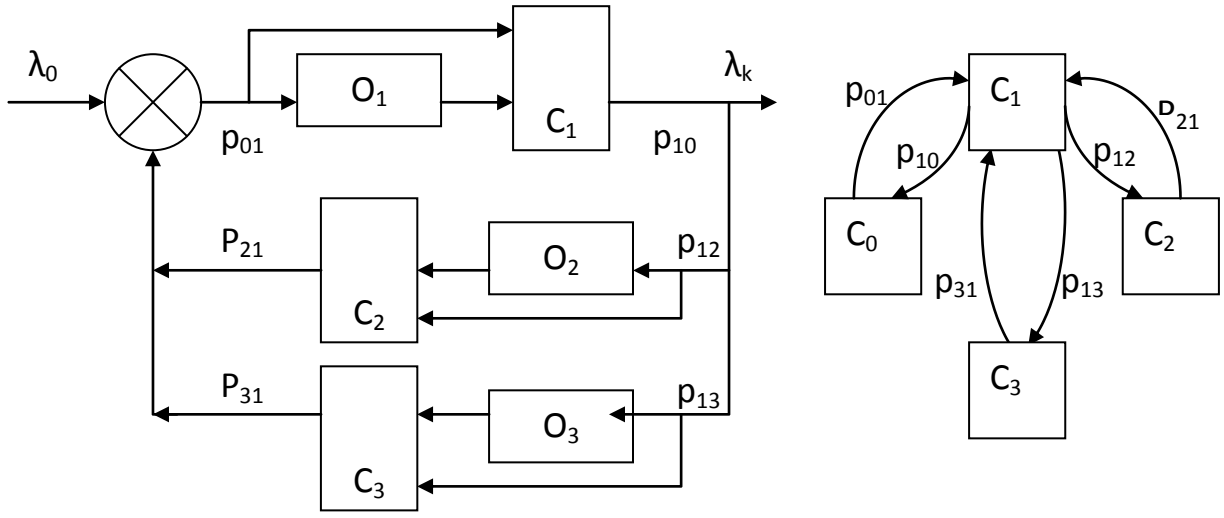


Рис. 34. Схема окремих блоків ЕОМ та її граф станів

На вхід системи подається потік завдань з інтенсивністю λ, які починають обслуговуватись СМО O_1- C_1, після чого з відповідною ймовірністю подаються на інші СМО, які можуть бути як одноканальними, так і багатоканальними.

Мережі масового обслуговування є зручною моделлю для вивчення інформаційних процесів в структурно складних системах. Порядок проходження вузлів може бути різним для різних замовлень. Для частини замовлень необхідне обслуговування у всіх вузлах, а для інших – тільки в деяких з вузлів.

Точний розрахунок характеристик таких СМО можливий тільки в разі, якщо всі потоки замовлень є пуассонівськими, а час обслуговування –

експонентні випадкові величини. В інших випадках можливий лише наближений розрахунок характеристик СМО.

Мережу СМО можна представити як систему, де кожний вузол є окремою СМО із своїми характеристиками. Вихідний потік деякого вузла мрежі буде вхідним потоком наступного вузла.

Потік замовлень на вході окремого вузла складається з вхідного потоку мережі (можливо, нульової інтенсивності) і з потоків, що надходять з виходів інших вузлів. Вхідний потік вузла в експоненційній мережі в загальному випадку пуассонівським не є, тому вузли мережі в загальному випадку не експоненціальні. Проте, вузли все ж часто вважають експоненціальними. Це дозволяє знайти з рівнянь балансу значення інтенсивностей λ1, ..., λN вхідних потоків замовлень і скористатися для розрахунку показників мережі відповідними аналітичними моделями теорії СМО.

При розрахунку показників мереж СМО, як і для багатофазних СМО, необхідно враховувати наступне:

- якщо на вхід СМО надходить кілька потоків замовлень, то інтенсивність повного потоку дорівнює сумі інтенсивностей окремих потоків;
- якщо на вхід СМО надходить частина замовлень з деякого потоку з інтенсивністю λ, то інтенсивність вхідного потоку замовлень в СМО можна визначити за формулою: $\lambda_{вх} = p \cdot \lambda$, де p – - ймовірність попадання замовлення у вхідний потік;
- інтенсивність вихідного потоку в СМО (тобто потоку оброблених замовлень) дорівнює інтенсивності вхідного потоку.

Щоб задати розімкнену експоненціальну мережу СМО необхідно задати значення наступного набору параметрів:

- N – число вузлів;
- n_i – число каналів і-ого вузла;
- $P = \|p_{ij}\|, i = \overline{1,N},\ j = \overline{0,N}$ матрицю ймовірностей передач;
- $\Lambda_i, i = \overline{i,N}$ інтенсивності вхідних потоків замовлень;
- μ_i – інтенсивності вихідних потоків або середній час обслуговування замовлень у вузлах $\bar{t}_{Si},\ i = \overline{1,N}$.

Ітенсивності вхідних потоків у вузлах $\lambda_i, i = \overline{1,N}$ знаходять з рівнянь балансу мержі з врахуванням властивостей злиття та розгалуження потоків.

Основне завдання аналізу мереж СМО полягає у визначенні таких показників, як середній час перебування замовлень в окремих вузлах мережі, завантаження пристроїв в вузлах, середні довжини черг до вузлів і т.п.

Середній час перебування замовлення в мережі.

Час перебування замовлення в мережі визначається як час, що минув з моменту приходу замовлення в мережу до моменту її виходу з мережі. Середній час перебування розраховується за формулою:

$\bar{T} = \frac{1}{\Lambda} \sum_{j=1}^{N} \lambda_j t_{Sj}$, де

де $\Lambda = \Lambda 1 + ... + \Lambda N$; $\bar{t}_{Si}$ - середній час перебування заявки в j-му вузлі.

Передаточні коефіцієнти.

Під передаточним коефіцієнтом розуміється середнє значення числа надходжень α_{ij} заявки *i*-го вхідного потоку в *j*-ий вузол за час перебування цієї заявки в мережі. У стаціонарному режимі при будь-яких Λ_1, ... Λ_N для λ_1, ... λ_N справедливо:

$$\begin{cases} \lambda_1 = \alpha_{11}\Lambda_1 + \alpha_{21}\Lambda_2 + ... + \alpha_{N1}\Lambda_N \\ \lambda_2 = \alpha_{12}\Lambda_1 + \alpha_{22}\Lambda_2 + ... + \alpha_{N2}\Lambda_N \\ ..... \\ \lambda_N = \alpha_{1N}\Lambda_1 + \alpha_{2N}\Lambda_2 + ... + \alpha_{NN}\Lambda_N \end{cases}$$

Інтенсивності надходження замовлень в *j*-ий вузол λ_1, ... λ_N виражені через інтенсивності вхідних потоків мережі Λ_1, ... Λ_N.

В матричній формі систему рівнянь можна представити як $\overline{\lambda} = A \cdot \overline{\Lambda}$, де матриця $A = \left\| \alpha_{ij} \right\|$.

Поклавши $\Lambda_1 = 1$ и $\Lambda_2 = ... = \Lambda_N = 0$, рівняння балансу для α_{1j}:

$$\begin{cases} \lambda_1 = \alpha_{11} \\ \lambda_2 = \alpha_{12} \\ ..... \\ \lambda_N = \alpha_{1N} \end{cases}$$

Інші коефіцієнти α_{kj}, $i \neq k$ знаходять як розв'язок рівнянь балансу для Λ_k=1 і Λ_i = 0,

Середній вхідний час перебування в мережі.

Середнім вхідним часом перебування в мережі $\overline{T_i}$ називається середній час перебування в мережі замовлення, що надходить з *i*-го вхідного потоку, $i = \overline{1, N}$:

$$\overline{T_i} = \sum_{j=1}^{N} \alpha_{ij} t_{Sj} .$$

Абсолютні пропускні спроможності.

Абсолютну пропускну спроможність по *i*-му входу A_i можна знайти безпосередньо за її визначенням. Записавши умову стаціонарності мережі у вигляді: $\frac{\lambda_i \bar{t}_{Si}}{n_i} \leq 1$, що еквівалентно $\lambda_i \leq \frac{n_i}{t_{Si}}$ при $i = \overline{1, N}$, і виразивши λ_i через Λ_i, отримаємо розгорнутий запис умови стаціонарності:

$$\begin{cases} \alpha_{11}\Lambda_1 + \alpha_{21}\Lambda_2 + \ldots + \alpha_{N1}\Lambda_N \le n_1 / t_{S1} \\ \alpha_{12}\Lambda_1 + \alpha_{22}\Lambda_2 + \ldots + \alpha_{N2}\Lambda_N \le n_2 / t_{S2} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \alpha_{1N}\Lambda_1 + \alpha_{2N}\Lambda_2 + \ldots + \alpha_{NN}\Lambda_N \le n_N / t_{SN} \end{cases}$$

Деякі з нерівностей виявляються зайвими: такі нерівності можна виключати, не змінюючи рішення системи. Якщо всі вхідні інтенсивності мережі, крім Λ_i, покласти рівними нулю, то, використовуючи розгорнуту форму записи умов стаціонарності, отримаємо умову стаціонарності:

$$\begin{cases} \alpha_{i1}\Lambda_1 \le n_1 / t_{S1} \\ \alpha_{i2}\Lambda_2 \le n_2 / t_{S2} \\ \ldots\ldots\ldots\ldots \\ \alpha_{iN}\Lambda_1 \le n_N / t_{SN} \end{cases} \text{ або } \begin{cases} \Lambda_1 \le n_1 / (t_{S1} \cdot \alpha_{i1}) \\ \Lambda_2 \le n_2 / (t_{S2} \cdot \alpha_{i2}) \\ \ldots\ldots\ldots\ldots \\ \Lambda_1 \le n_N / (t_{SN} \cdot \alpha_{iN}) \end{cases}$$

Тоді абсолютна пропускна спроможність визначається як мінімум значень, що стоять в правих частинах нерівностей:

$$A_i = \min_j \frac{n_j}{t_{Sj} \cdot \alpha_{ij}}.$$

Умовні пропускні спроможності.

Умовної пропускною спроможністю по *i*-му входу B_i називають максимальне значення інтенсивності Λ_i, при якому мережа залишається стаціонарною. При заданих Λ_k *(k≠i)* мережа стаціонарна для будь-яких значень $\Lambda_i \le \beta_i$. Для знаходження значень β_i необхідно розв'язати систему нерівностей:

$$\begin{cases} \Lambda_1 \le \beta_1 \\ \Lambda_2 \le \beta_2 \\ \ldots\ldots\ldots\ldots \\ \Lambda_1 \le \beta_N \end{cases}.$$

B_i визначається як найменше значення серед усіх β_i.

Запаси за пропускною здатністю.

Запас за пропускною здатністю $D_i = \mathrm{B}_i - \Lambda_i$, $i = \overline{1, N}$ показує, наскільки може бути збільшена інтенсивність приходу замовлень на *i*-му вході при фіксованих інших інтенсивностях без порушення умови стаціонарності.

Приклад.

1.На виробничу дільницю У3 надходять для обробки деталі з двох інших дільниць (У1 і У2). Схема дільниці У3 приведена на рис. 35.

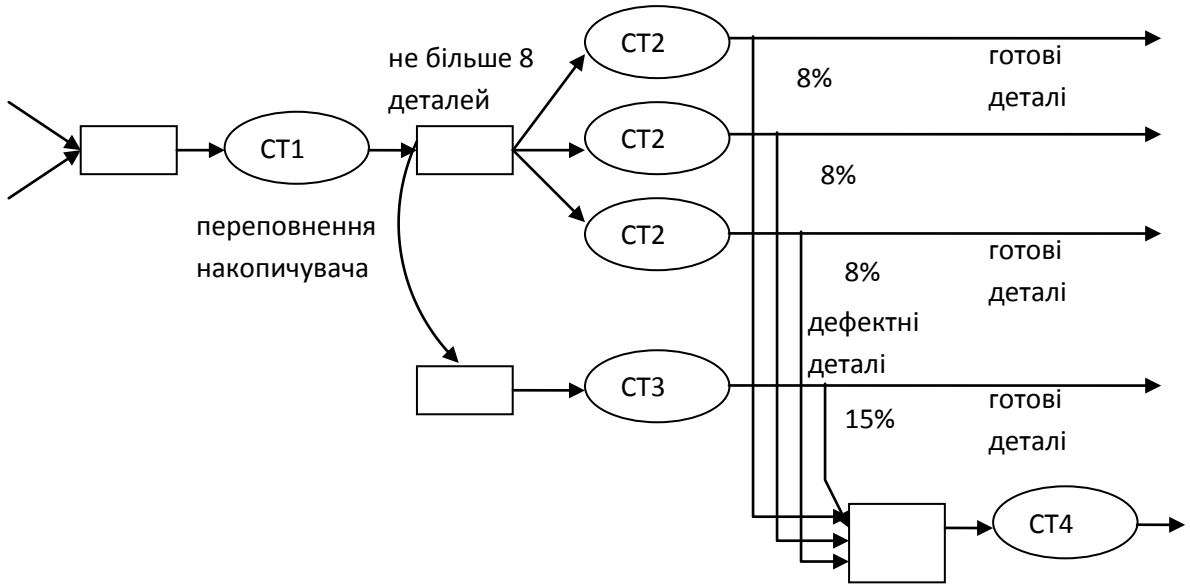


Рис.35. Схема виробничої дільниці У3

Деталі з дільниці У1 надходять в середньому через кожні 10 хв, з дільниці У2 – через кожні 15 хв. Всі деталі проходять обробку на верстаті СТ1. Обробка деталі на цьому верстаті займає від 3 до 7 хв. З верстата СТ1 деталі направляються на групу з трьох однакових верстатів СТ2. Перед цими верстатами знаходиться накопичувач для деталей, які очікують обробки (загальний для всіх трьох верстатів). Деталь направляється на будь-який вільний верстат СТ2. Обробка деталі на верстаті СТ2 займає в середньому 20 хв. Накопичувач перед верстатами СТ2 вміщує 8 деталей. При його заповненні деталі направляються для обробки на верстат СТ3. Обробка на цьому верстаті займає в середньому 25 хв.

При обробці деталей можливі дефекти. На верстатах СТ2 дефект допускається приблизно в 8% випадків, на верстаті СТ3 – в 15% випадків. Всі дефектні деталі направляються на верстат СТ4 для усунення дефекту; це займає в середньому 10 хв.

Потрібно знайти характеристики роботи всіх верстатів, а також визначити середній час перебування деталі на ділянці У3.

Розв'язок.

Розрахунок характеристик верстата СТ1. Щоб виконати цей розрахунок, будемо вважати потоки деталей з цехів У1 і У2 пуассонівськими. Тоді верстат СТ1 можна розглядати як одноканальну СМО, де потік замовлень є пуассонівським, а час обслуговування розподілено по рівномірному закону, тобто СМО типу (M / G / 1). Інтенсивності потоків з ділянок У1 і У2 відповідно дорівнюють $\lambda_1 = 0{,}1,\ \lambda_2 = 0{,}07$ деталей за хв. Інтенсивність вхідного потоку для верстата СТ1 дорівнює сумі інтенсивностей ділянок $\lambda_{CT1} = 0{,}17$. Середній час обробки деталі $t_{обр} = (3+7)/2$ хв., отже, інтенсивність обробки $\mu_{CT1} = 1/5 = 0{,}2$.

Навантаження на систему $\rho = \lambda/\mu = 0{,}85$. Тоді $p_0 = 1 - \rho = 0{,}15$.

Для рівномірного закону розподілу коефіцієнт варіації становить $\varepsilon = \frac{b-a}{\sqrt{3}(b+a)} = \frac{7-3}{\sqrt{3}(7+3)} \approx 0{,}23$. Тоді середня довжина черги $L_{черги} = \rho^2 \frac{(1+\varepsilon^2)}{2(1-\rho)} = 2{,}54$. Всі інші характеристики будуть визначатись як для одноканальної СМО з необмеженою чергою.

Розрахунок характеристик верстатів СТ2. Будемо вважати потік деталей, що виходять з верстата СТ1, пуассонівським (хоча це не зовсім точно, так як час обробки на верстаті СТ1 – не експоненціальна, а рівномірна випадкова величина). Будемо також вважати, що час обробки деталей на верстатах СТ2 є експоненціальна випадкова величина. Тоді групу верстатів СТ2 можна розглядати як трьохканальну марківську СМО (М / М / 3) з обмеженням на довжину черги (так як накопичувач вміщує лише 8 деталей, і при його заповненні деталі направляються на інший верстат). Інтенсивність вхідного потоку $\lambda_{СТ2} = 0{,}17$, так як на цей блок надходять всі деталі з першої ділянки. Середній час обробки деталі $t_{обр} = 20$ хв., отже, інтенсивність обробки $\mu_{СТ2} = 0{,}05$, довжина черги $m = 8$. Розрахунки виконуються як для багатоканальної системи з обмеженою чергою.

Розрахунок характеристик верстата СТ3. На цей верстат надходять деталі, що не потрапили на верстати СТ2 із-за заповнення накопичувача (тобто деталі, які отримали відмову в обслуговуванні). Тому інтенсивність вхідного потоку для верстата СТ3 можна знайти наступним чином: $\lambda_{СТ3} = p_{відмови} \cdot \lambda_{СТ2} = 0{,}16 \cdot 0{,}17 = 0{,}027$, де $p_{відмови}$ – це ймовірність відмови в обслуговуванні для групи верстатів СТ2.

Будемо вважати потік деталей на верстат СТ3 пуассонівським, а час обробки на цьому верстаті – експоненціальним. Тоді верстат СТ3 можна розглядати як одноканальну марківську СМО (М / М / 1) без обмежень на чергу, де $\lambda_{СТ3} = 0{,}027$, середній час обробки деталі $t_{обр} = 25$ хв., отже, інтенсивність обробки $\mu_{СТ3} = 0{,}04$, $\lambda_{СТ3} = 0{,}027$ деталі / хв, $t_{обр} = 25$ хв, $\mu_{СТ3} = 0{,}04$ деталі / хв. Розрахунок характеристик верстата СТ3 виконується згідно порядку розрахунку одноканальної СМО.

Розрахунок характеристик верстата СТ4. На цей верстат надходить 8% деталей, оброблених на верстатах СТ2, і 15% деталей, оброблених на СТ3. Тому інтенсивність вхідного потоку для верстата СТ3 можна знайти наступним чином: $\lambda_{СТ4} = 0{,}08\mu_{СТ2}\bar{c}_{СТ2} + 0{,}15\mu_{СТ3}\bar{c}_{СТ3} = 0{,}08 \cdot 0{,}14 + 0{,}15 \cdot 0{,}027 = 0{,}015$, ($\mu\bar{c}$ – пропускна спроможність системи). Будемо вважати потік деталей на верстат СТ4 пуассонівським, а час обробки на цьому верстаті - експоненціальним. Тоді верстат СТ4 можна розглядати як одноканальний марківську СМО (М / М / 1) без обмежень на чергу, де $\lambda_{СТ4} = 0{,}015$ деталі/хв, $t_{обр} = 10$ хв, $\mu_{СТ4} = 0{,}1$ деталі/хв. Характеристики верстатів приведені в таблиці.

Таблиця 4.1. Характеристики ділянок виробничого процесу

Характеристики	Т1	Т2	Т3	Т4
ρ – навантаження	,85	,13	,68	,15
p_0 – ймовірність простою	,15	,009	,32	,85
$L_{чер}$ – к-ть замовлень у черзі	,54	,38	,4	,027
$p_{відм}$ – ймовірність відмови		,16		
$p_{обсл}$ – ймовірність обслуговування		,84		
k – к-ть замовлень на обслуговуванні	,85	,86	,68	,15
$L_{сист}$ – к-ть замовлень у системі	,39	,24	,08	,177
Пропускна спроможність	,17	,14	,027	,015
$W_{чер}$ – середній час перебування у черзі	4,9	1,29	1,9	,76
$W_{сист}$ – середній час перебування у системі	9,9	1,29	6,9	1,8

Розрахунок середнього часу перебування деталі на ділянці УЗ. Всі деталі проходять обробку на верстаті СТ1. Середній час перебування деталі на цьому верстаті (включаючи час очікування обробки і саме час обробки) становить 19,9 хв. Потім 84% деталей проходять обробку на одному з верстатів СТ2; середній час перебування деталі на цих верстатах становить 51,29 хв. Решта 16% деталей обробляються на верстаті СТ3; середній час перебування деталі на цьому верстаті – 76,9 хв. Крім того, для 8% деталей, оброблених на верстатах СТ2, і 15% деталей, оброблених на верстаті СТ3, необхідно усунення дефекту на верстаті СТ4. Це займає в середньому 11,8 хв. Таким чином, середній час перебування деталі на ділянці можна знайти так:

$$t_{УЗ} = 19,9 + 0,84 \cdot 51,29 + 0,16 \cdot 76,9 + (0,84 \cdot 0,08 + 0,16 \cdot 0,15) \cdot 11,8 = 76,36$$

хв.

За результатами аналізу показників верстатів можна відзначити такі недоліки в роботі ділянок і запропонувати способи їх усунення:

• перевантаження групи верстатів СТ2 і недостатнє завантаження верстата СТ3. Для усунення цього недоліку можна запропонувати зменшити розмір накопичувача перед верстатами СТ2. В такому випадку більша кількість

деталей буде направлятися на верстат СТ3. В результаті завантаження верстатів СТ2 знизиться, а верстата СТ3 – підвищиться;

• явне недовантаження верстата СТ4: верстат простоює 85% робочого часу. Для усунення цього недоліку можна запропонувати використовувати верстат СТ4 не тільки для усунення дефектів, але і для інших робіт.

Контрольні питання.

1. Які системи масового обслуговування відносяться до багатофазних?
2. Наведіть приклади багатофазних СМО?
3. Які особливості розрахунку характеристик багатофазних СМО?
4. Що називають мережами СМО?
5. Яка різниця між мережами та багатофазними СМО?
6. Як виконується розрахунок показників мережі СМО?
7. Наведіть приклади мереж СМО.

Частина2.

1. Імітаційне моделювання СМО із застосуванням середовища Matlab

1.1. Розробка імітаційної моделі СМО в середовищі Matlab

Для побудови імітаційних моделей СМО в середовищі Matlab+Simulink передбачена бібліотека SimEvents, за допомогою інструментів якої можна проектувати та моделювати випадкові динамічні системи з неперервними та дискретними компонентами з дискретними подіями та дискретним часом, до яких відносяться розподілені системи управління, апаратні конфігурації, мережі збору та передачі інформації і таке інше.

Для реалізації складних моделей іноді виникає необхідність застосування інших основних бібліотек графічної мови Simulink, таких як Math Operations (математичні операції), Signals (сигнали), Ports & Subsystems (порти та підсистеми) та інших. В даному розділі розглянемо основні компоненти бібліотеки SimEvents, які є базовими для реалізації імітаційної моделі систем масового обслуговування (рис 1).

Основним поняттям дискретного моделювання на основі подій є:

- entity – сутність (замовлення);
- event – подія – миттєва дискретна подія, що змінює стан і/або є причиною інших подій.

До складу бібліотеки SimEvents входять наступні інструментальні бібліотеки блоків:

- Attributes – визначення атрибутів (параметрів) сутностей;
- Event Translation – перетворення сигналу події у одну або декілька функцій;
- Generators – бібліотека генераторів замовлень;
- Queues – бібліотека черг;
- Servers – бібліотека сервісів (каналів);
- SimEvents Ports and Subsystems – бібліотека портів та підсистем;
- SimEvents User Defined Funct – визначення атрибутів сутності за допомогою функції;
- Entity Management – управління потоками (об'єднання, розподілення) сутностей;

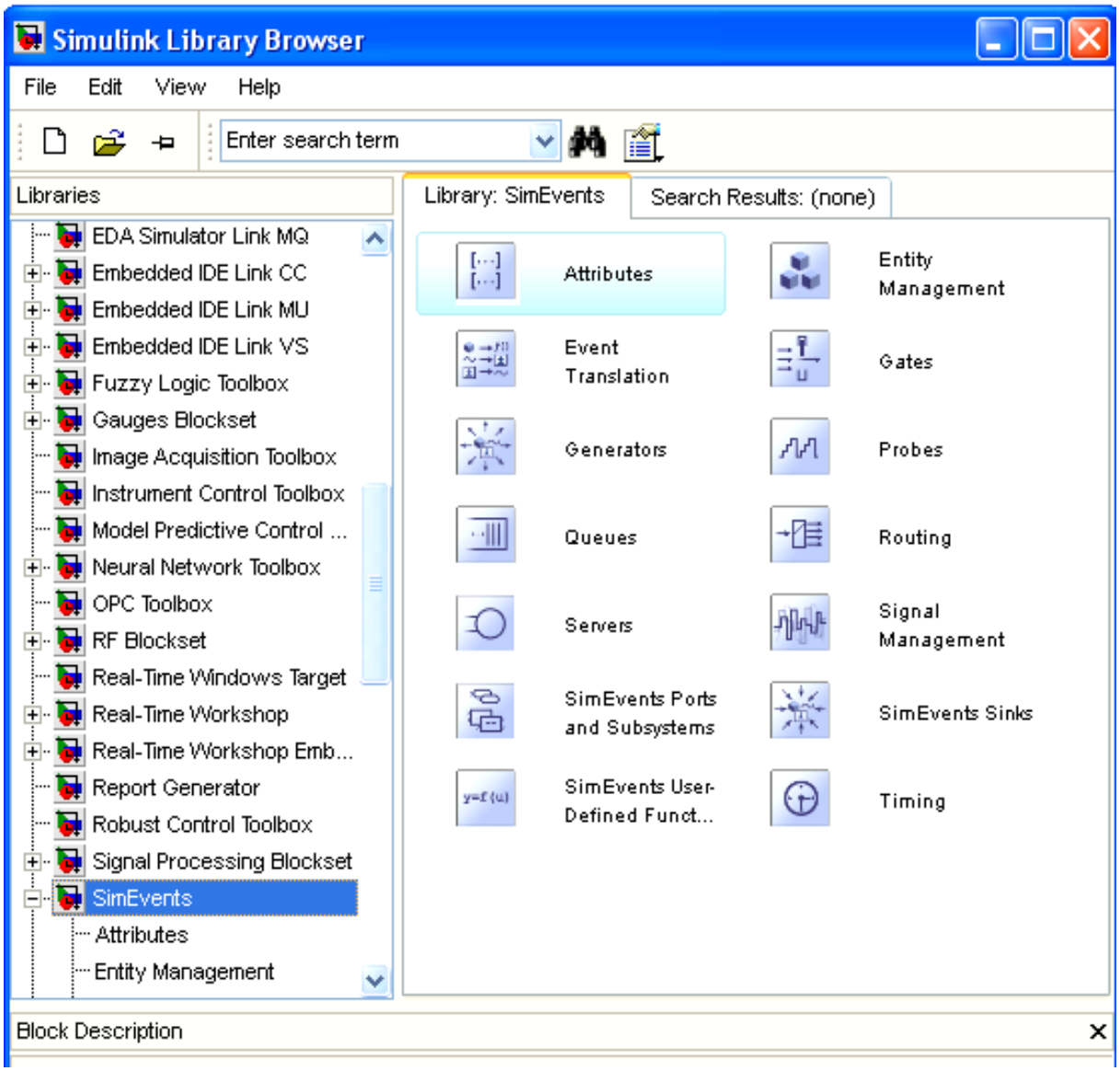


Рис. 1. Компоненти бібліотеки SimEvents

- Gates – управління потоками в залежності від умов, що накладаються на вхідний компонент;
- Probes – лічильник вихідних сутностей із записом у порт і/або до атрибутів;
- Routing – блоки перемикачів та управління потоками;
- Signal Management – управління сигналами, що визначають події;
- SimEvents Sinks – блоки поглинання замовлень та графічного представлення результатів;
- Timing – блоки управління часом.

При моделюванні дискретних подій сутності можуть переміщуватись через мережі черг (queues), серверів (servers) и перемикачів (switches). Графічні блоки бібліотеки SimEvents представляють набір компонентів, які обробляють сутності, але самі сутності не мають графічного представлення.

Загальний вигляд моделі СМО представлений на рис.2.

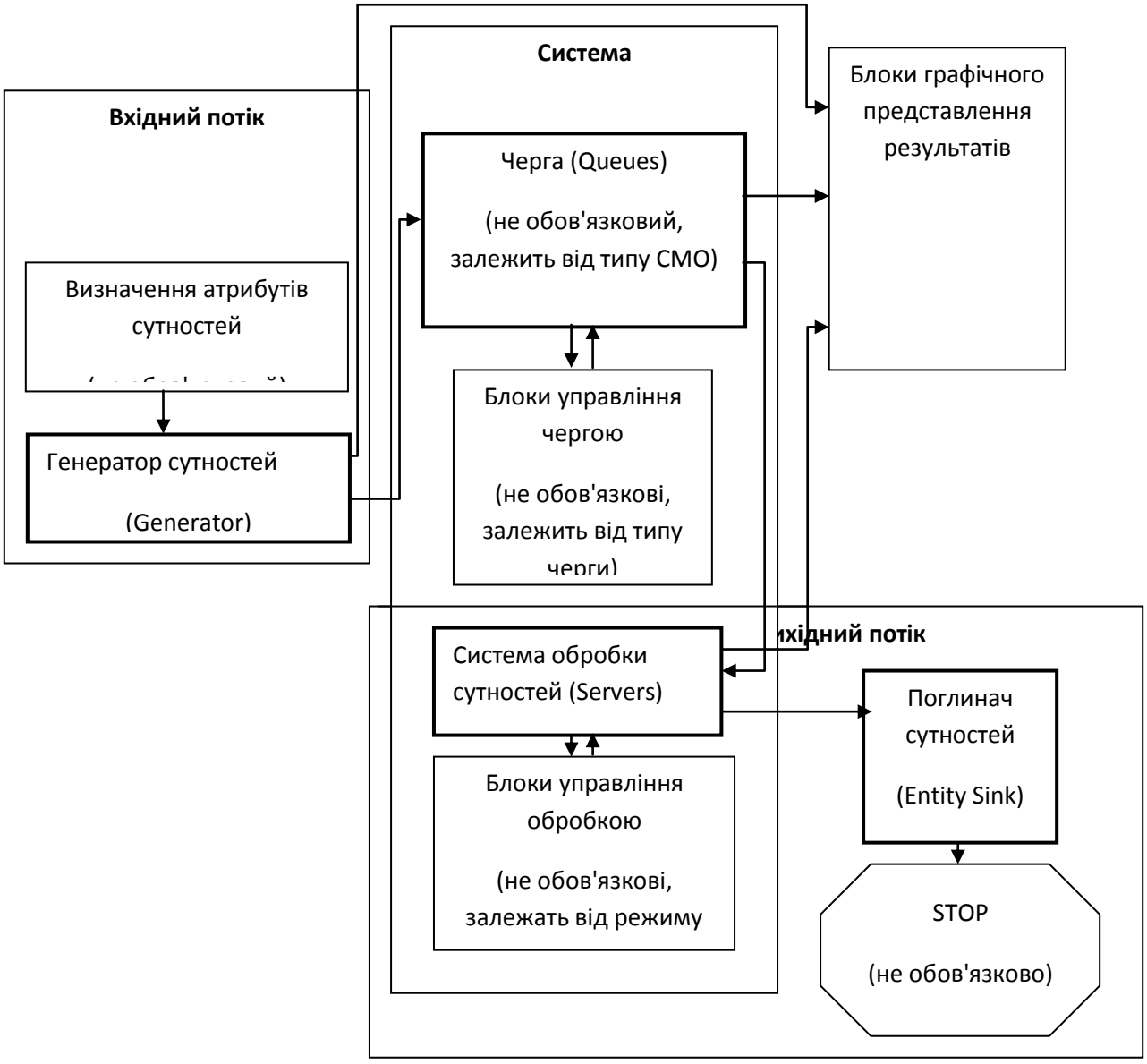


Рис. 2. Структура імітаційної моделі СМО

Графічні блоки мають декілька закладок для їх налаштування. Для отримання статистики передбачена закладка Statistics, яка містить поля основних статистичних даних для відповідного блоку, які можна зробити доступними, встановивши їх у режим On або недоступними, встановивши їх у режим Off. Встановлення поля в режим On створює додатковий вихід для виводу.

До складу сучасних версій середовища Mftlab входить розширений набір компонентів бібліотеки SimEvents, що дозволяє створювати складні імітаційні моделі мереж СМО, але для розробки простої моделі часто достатньо лише таких основних блоків, як генератори, черги та сервіси.

1.2. Генератори сутностей (замовлень) (Generators)

Бібліотека генераторів сутностей містить три типи генераторі:

- генератори сутностей (замовлень)(Entity Generators);

- генератори подій (Event Generators);
- генератори сигналів (Signal Generators),

де до кожної з груп входить два типи інструментів (рис.3)

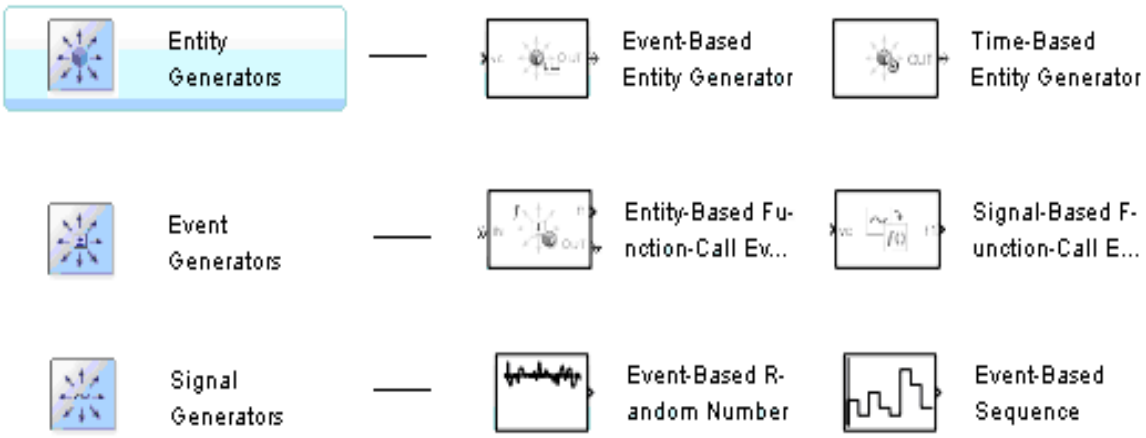


Рис. 3. Компоненти бібліотеки Generators

Генерація сутностей може відбуватись під впливом різних умов (подій, сигналів, функцій), що обумовлює тип вхідних параметрів блоків генераторів, які наведені у таблиці 1.

Блоки бібліотеки Entity Generators є основними для створення потоків замовлень (сутностей), тоді як блоки генерації подій та сигналів використовують для управління іншими блоками.

Таблиця 1. Умови генерації сутностей

Умови генерації сутностей	**Імпорт**
Генерація сутностей відбувається коли додаток повторно обчислюється і видає значення сигналу	trigs
Генерація сутностей управляється тригером	trigr
Генерація сутностей управляється вхідним сигналом	trigc
Генерація сутностей управляється функцією	trigcn

1.2.1. Entity Generators – генератори сутностей

Генератори сутностей породжують потік замовлень під впливом умов та з характеристиками, що визначаються користувачем. Бібліотека генераторів сутностей містить два блоки:

- Event Based Entity Generator;
- Time Based Entity Generator.

1.2.1.1 Event Based Entity Generator

Event Based Entity Generator – блок генерує сутності за умови виконання певних подій (таблиця 1).

Вигляд вікна налаштування параметрів блоку наведено на рис 4. Це вікно має дві закладки:

- Entity Generation – генерація сутності;
- Statisyics – статистика.

Розглянемо налаштування генерації сутності. Блок генерує сутності двох типів **Entity type**:

- Blank – тип, що не включає атрибутів сутностей;
- Standard – тип, що враховує атрибут Priority (Пріоритет) і встановлює значення параметра Count (Кількість) від 1 до 10 за замовчуванням.

Allow OUT port blocking – якщо вибирається цей пункт, то моделювання зупиняється, якщо з'являється повідомленням про помилку і вхідний порт стає недоступний для прийому сутності (замовлення).

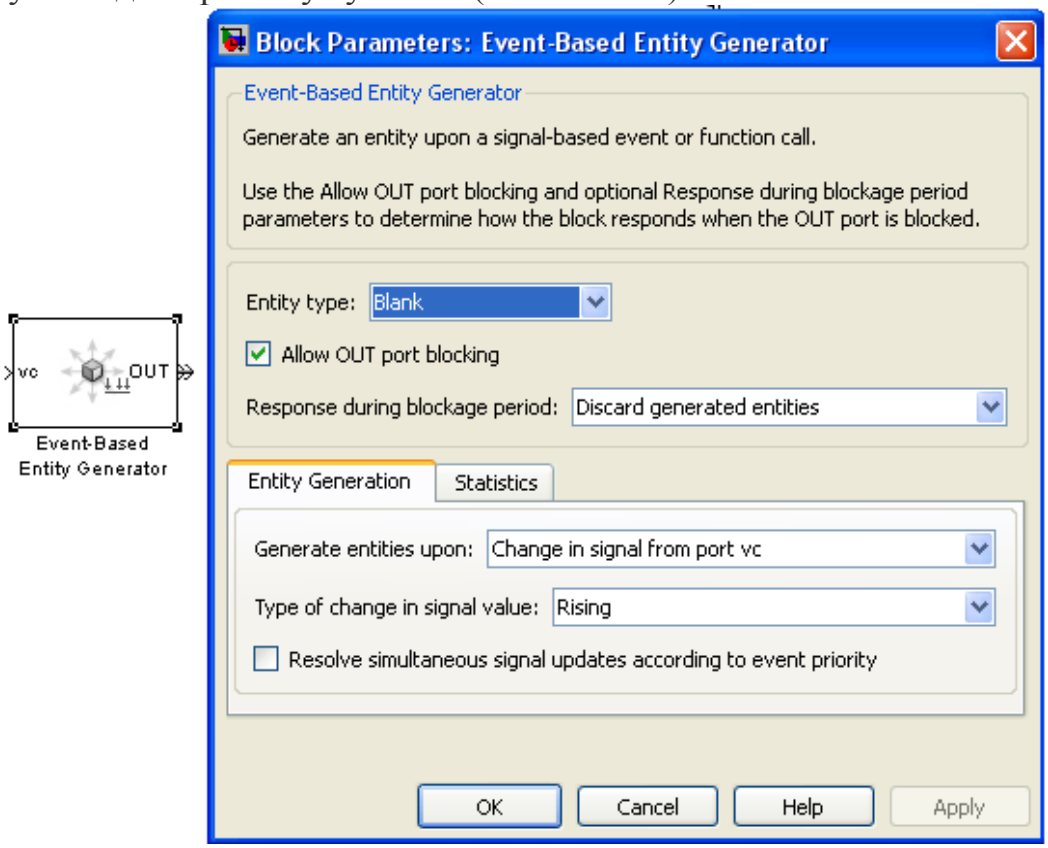


Рис. 4. Діалогове вікно налаштування блоку Event Based Entity Generator

Response during blockage period – визначається причина недоступності вхідного порту. Цей пункт працює, коли вибраний пункт **Allow OUT port blocking.**

Generate entities upon – визначає порядок генерації сутностей. Може приймати значення, наведені на рис. 5 (див. таблицю 1).

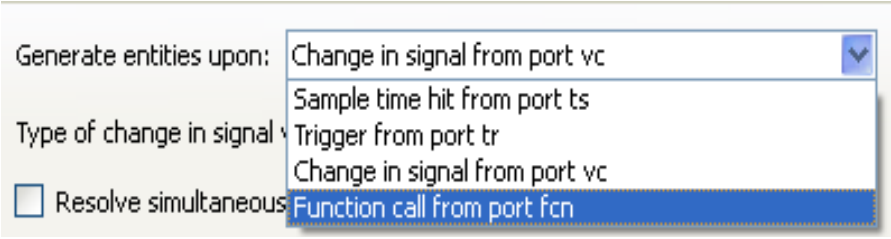


Рис. 5. Значення для налаштування порядку генерації сутностей

Type of change in signal value – тип зміни значення сигналу. Сигнал може генеруватись за збільшенням значень (rising), за спаданням (fallinf) або іншим чином (either). Це поле стає доступним лише коли поле **Generate entities upon** встановлене на порт vc.

Resolve simultaneous signal updates according to event priority – встановлює вибір з одночасних синалів згідно до встановленого пріоритету.

Параметри налаштування статистики наведені на рис. 6. Для збору статистики необхідно у відповідні поля встановити значення On, тоді у блоку з'являється новий вихідний порт для відображення результатів.

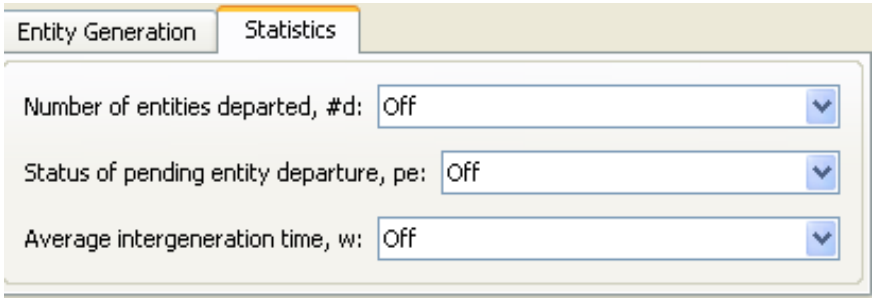


Рис. 6. Налаштування статистики

Number of entities departed: #d – кількість сутностей, що вийшли з блоку після початку моделювання.

Status of pending entity departure – статус сутності, що виходить з блоку через порт **pe.**

Average intergeneration time w – середній час між генераціями сутностей.

Приклади. На рис. 7 наведено приклад фрагменту простого застосування блоку Event Based Entity Generator, де генерацією сутностей управляє подія зміни дискретного часу. На графіку відображається кількість замовлень, що виходить з генератора за часом.

Сутності, які виходять з генератора можуть направлятись до іншого блоку моделі, наприклад, до черги або сервісів обслуговування.

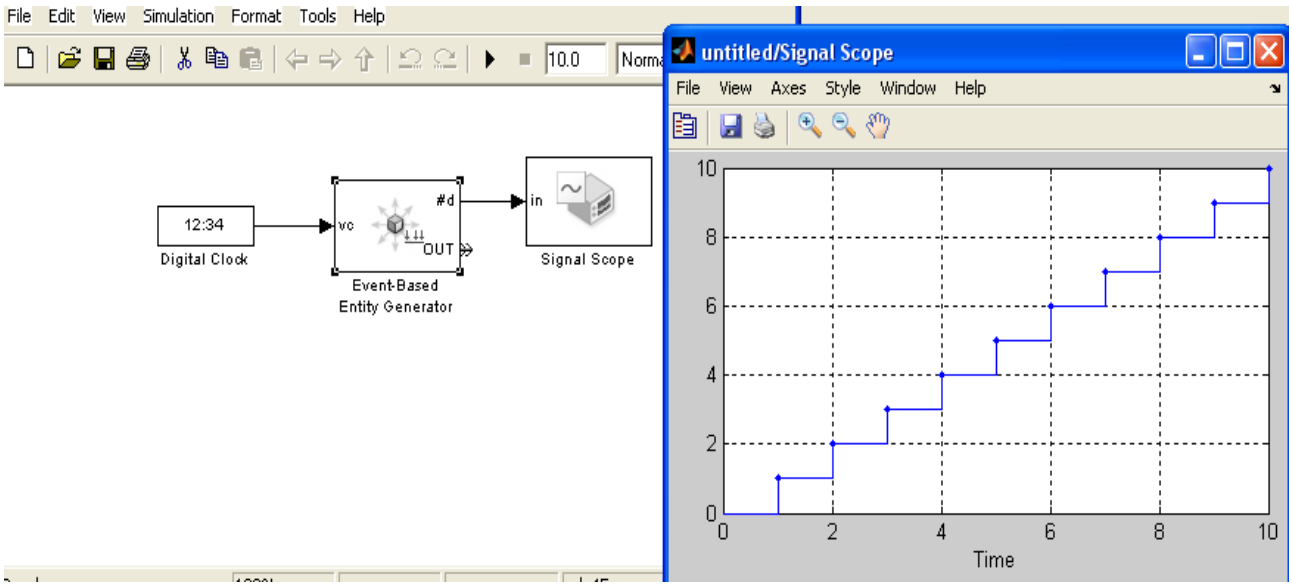


Рис. 7. Приклад простого використання блоку Event Based Entity Generator

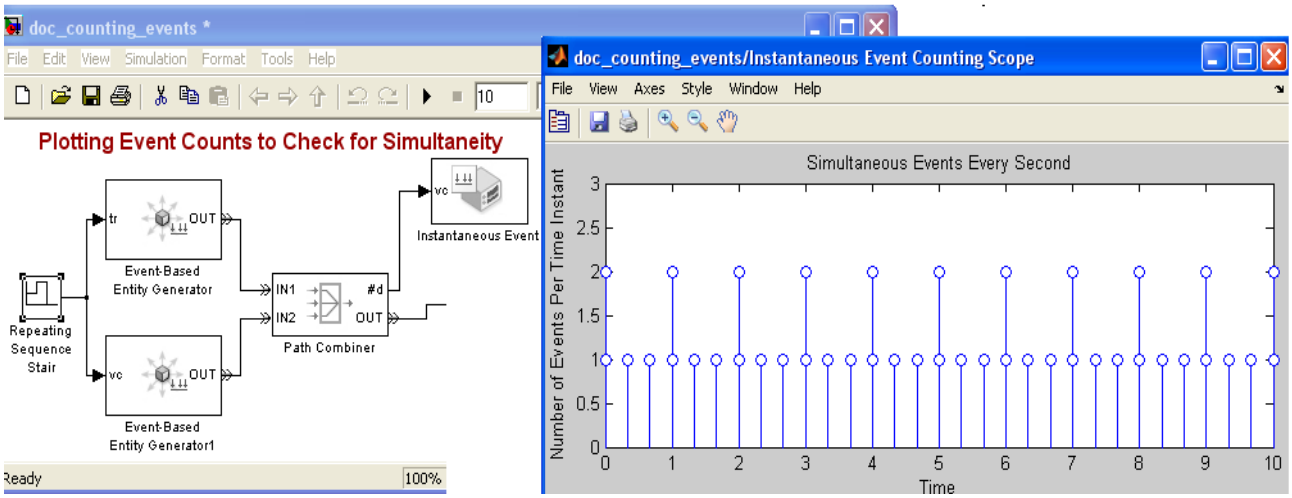


Рис. 8. Приклад фрагменту моделі з двома генераторами

Більш складний приклад наведено на рис. 8. Генерацією сигналів управляє блок Repeating Sequence Stairs, який генерує періодичні сигнали, що приймають значення 0, 1, 2 з періодом 1 і 1/3. До складу моделі входить два генератори, вихідні потоки яких об'єднані в один блоком Path Combiner. Кількість вихідних замовлень за часом відображена на графіку.

1.2.1.2. Time-Based Entity Generator

Time-Based Entity Generator – блок генерує сутності у моменти часу, які визначаються вхідним сигналом або статистичним розподілом (рис. 9).

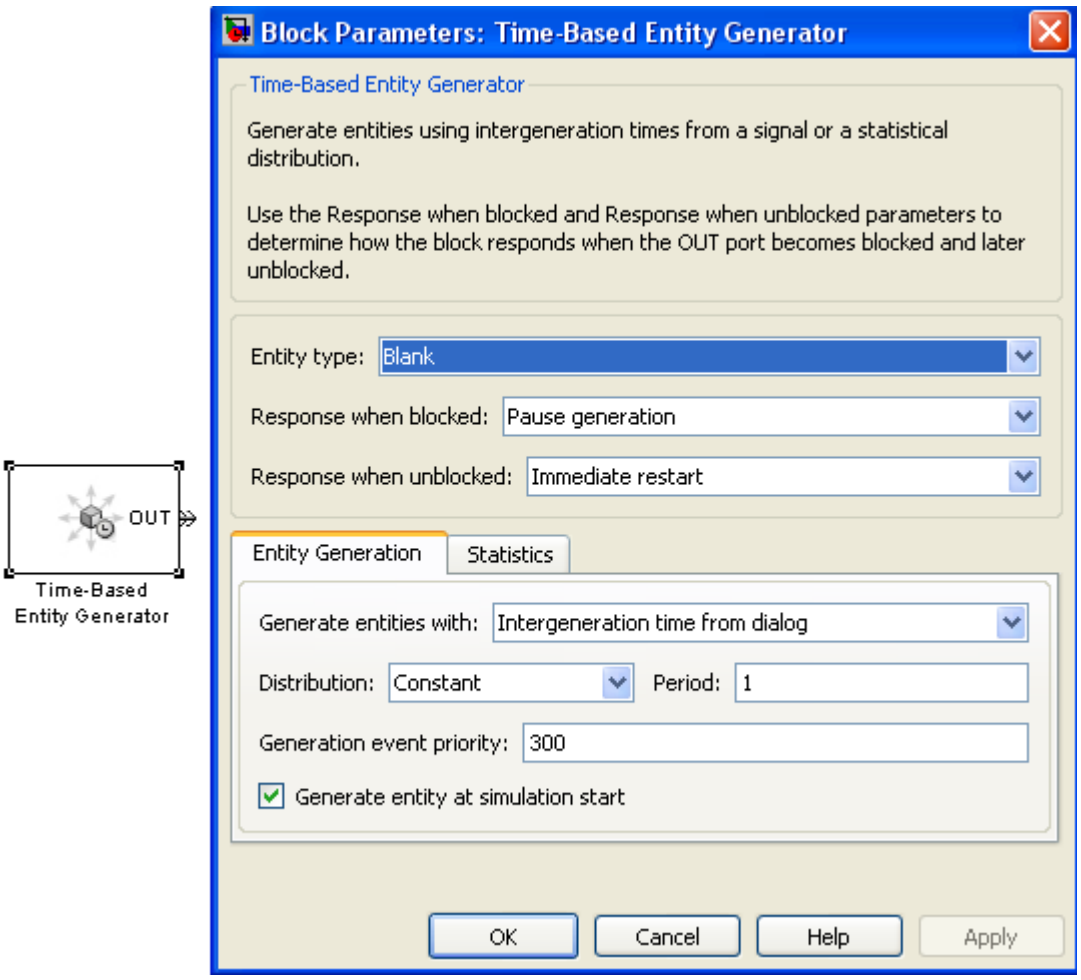


Рис. 9. Блок Time-Based Entity Generator та його діалогове вікно

Структура блоку Time-Based Entity Generator подібна до структури блоку Event Based Entity Generator. Так поля на закладці статистики аналогічні до полів попереднього блоку **Event Based Entity Generator**. Розглянемо поля налаштування генерації, які відрізняються від полів попереднього блоку.

Response when blocked – поле визначає коли згенерована сутність не може вийти з блоку. Поле може приймати два значення:

- Pause Generation – призупинення генерації;
- Error – помилка.

Response when unblocked – визначається порядок розблокування:

- Immediate Restart – термінове відновлення генерації;
- Delayed Restart – відновлення генерації із затримкою.

Generate entities with – визначає, де блок отримує вказівку коли генерувати сутності або з діалогу або із сигнального порту t.

Distribution (розподіл) – визначає статистичний розподіл часу генерації сутностей. Поле доступне тільки при встановленні поля Generate entities with у режим Integration time from dialog і може приймати наступні значення:

- Constant – сталі рівні проміжки часу;
- Uniform – рівномірний розподіл часу;
- Exponential – експоненціальний розподіл часу.

Якщо поля Generate entities with встановлене у режим Integration time from dialog та вибрано розподіл Constant, то необхідно визначити період **Period** – довжину часового інтервалу між генерацією сутностей у секундах.

Для довільного та експоненціального розподілу встановлюється початкове значення для генератора випадкових чисел:

Initial seed – невід`ємне ціле значення. Як правило, це число визначається як непарне число з великим значенням.

У випадку рівномірного розподілу задається часовий інтервал парою значень у секундах – **Minimum**, **Maximum.**

У випадку експоненціального розподілу задається математичне сподівання – **Mean (** $1/\lambda$ **).**

Generation event priority – пріоритет суності у процесі моделювання.

Generate entity at simulation start – при виборі цього поля перша суність генерується на початку процесу моделювання, а наступна – на початку встановленого часового проміжку. У протилежному випадку перша сутність генерується на початку встановленого часового проміжку.

Приклад. На рис. 10 наведений приклад простої моделі з генератором замовлень, підпорядкованим експоненціальному закону розподілу з математичним сподіванням 0.4, чергою "перший-прийшов-перший-вийшов" довжиною 5 та одним сервісом з часом обслуговування 1.5. На графіках наведено кількість сгенерованих замовлень, довжина черги та кількість оброблених замовлень на протязі імітації моделі (t=10).

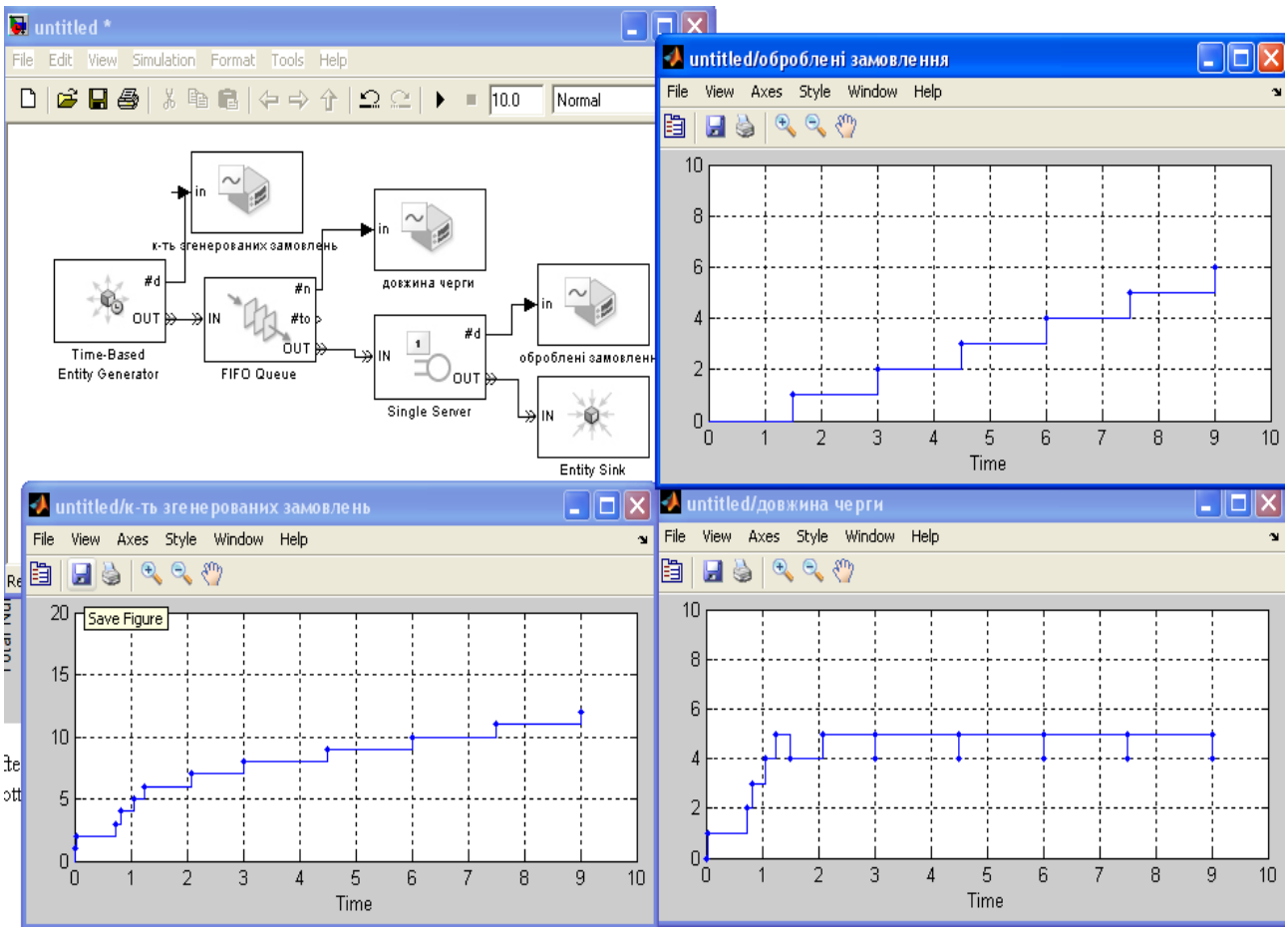


Рис. 10. Прикдад моделі з використанням блоку Time-Based Entity Generator

1.2.2. Event Generators – генератори подій

1.2.2.1. Entity Based Function Call Generator

Entity Based Function Call Generator – генерує виклик функції-події, що відповідає сутності, яка надходить у блок (рис. 11).

Поле **Generate function call** дозволяє задавати виклик функції до або після надходження сутності у блок.

Закладка статистики містить два поля:

- **Number of entities departed** (кількість сутностей, що вийшли з блоку) – управляє доступом та поведінкою вихідного порту сигналів **#d.**
- **Number of f1 function calls** (кількість викликів функції) – управляє доступом та поведінкою вихідного порту **#f1.**

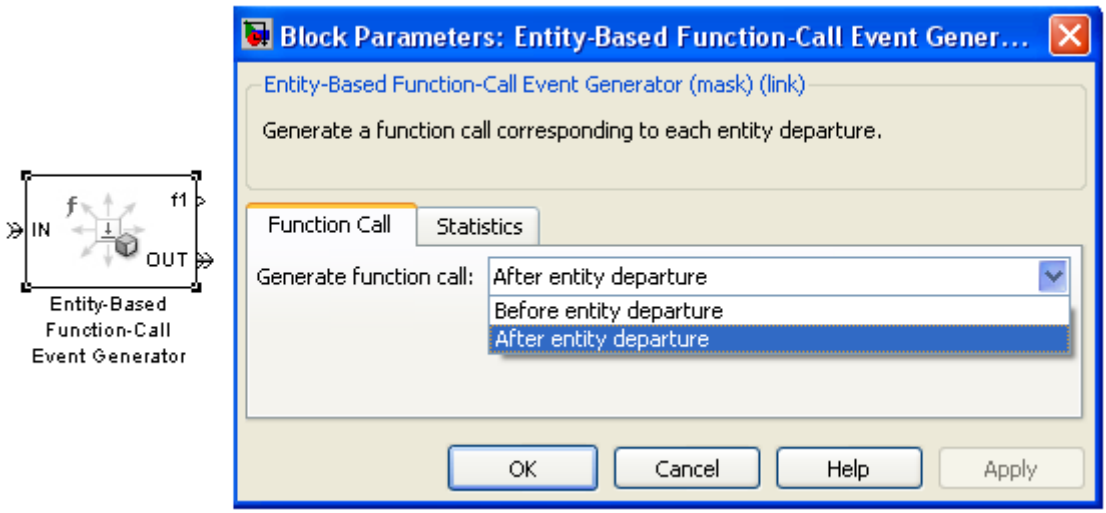


Рис. 11. Блок Entity Based Function Call Generator та його діалогове вікно

Приклад. На рис. 13 наведено приклад моделі, яка виконує обчислення, коли сутність надходить на вхід блоку обєднання сигналів (Path Combiner) IN2 або IN3 (з другої обо третьої черги), але не на вхід IN1 (з першої черги). Обчислення виконуються у середині блоку Function-Call Subsystem, який знаходиться у бібліотеці **Ports & Subsystems**. Вхідним сигналом для функції є блок із бібліотеки **Sources** (Джерела), який визначає вхідний сигнал на основі дискретної послідовності часу, яка задається користувачем у вигляді вектора, як параметр даного блоку.

Рис. 12. Структура блоку Function-Call Subsystem

Блок Mux об'єднує два сигнали виклику функції, створюючи сигнал виклику функції підсистеми, яка називається подвійною для певного момену часу. Блок підпрограми Function Call Sybsystem має просту структуру (рис. 12) і гарантує збереження замовлення на сервері до його надходження у блок обє'днання замовлень Path Combiner (отже для об'єднання двох замовлень він виконується двічі).

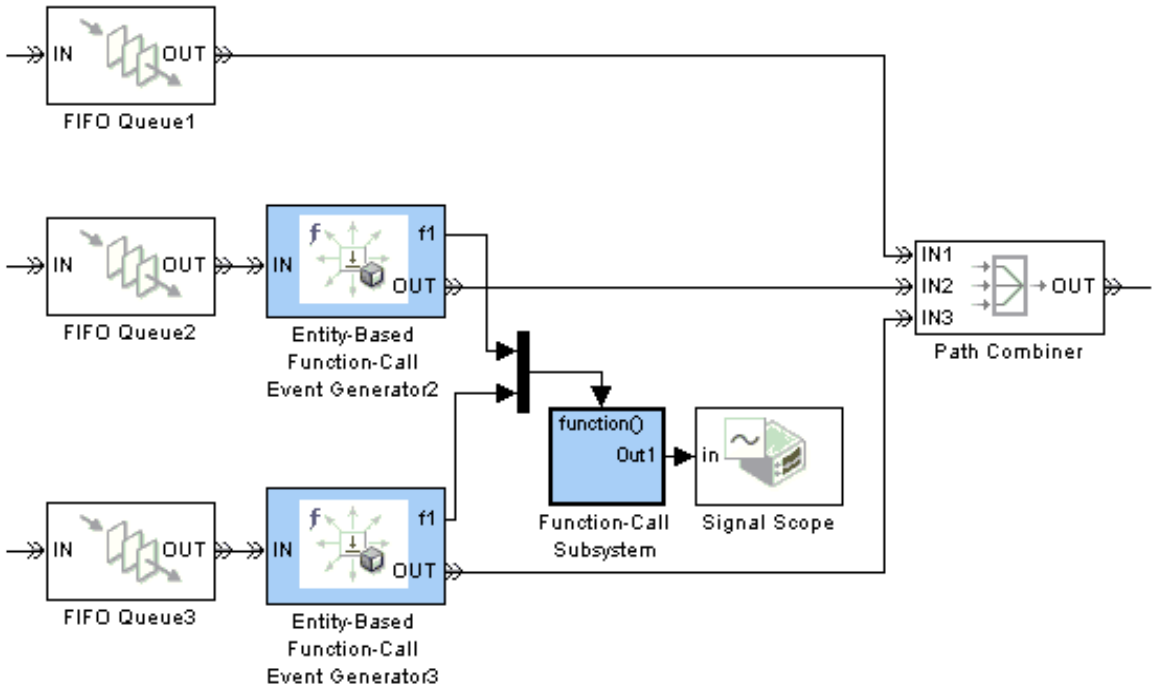


Рис. 13. Приклад фрагменту моделі з використанням блоку Entity Based Function Call Generator

1.2.2.2. Signal Function Call Event Generator

Signal Function Call Event Generator – генерує функцію виклику події у відповідь на сигнал події. Блок звертається до функції, яка визначає подію для генерації сутності. В якості події можна використовувати будь-який сигнал, функцію або блоки бібліотеки **Stateflow (**бібліотека моделювання дискретних систем**)**. За допомогою цього блоку можна затримувати виходи подій згідно до певних умов.

Діалогове вікно блоку містить дві закладки, подібно блоку Entity Based Function Call Generator, який описаний вище. Відмінність полягає у структурі полів першої закладки, яка містить поля (див. блок Event Based Entity Generator):

- **Generate function call only upon** – визначення типу вхідного сигналу (таблиця 1);
- **Type of change in signal value** – тип зміни значення сигналу.

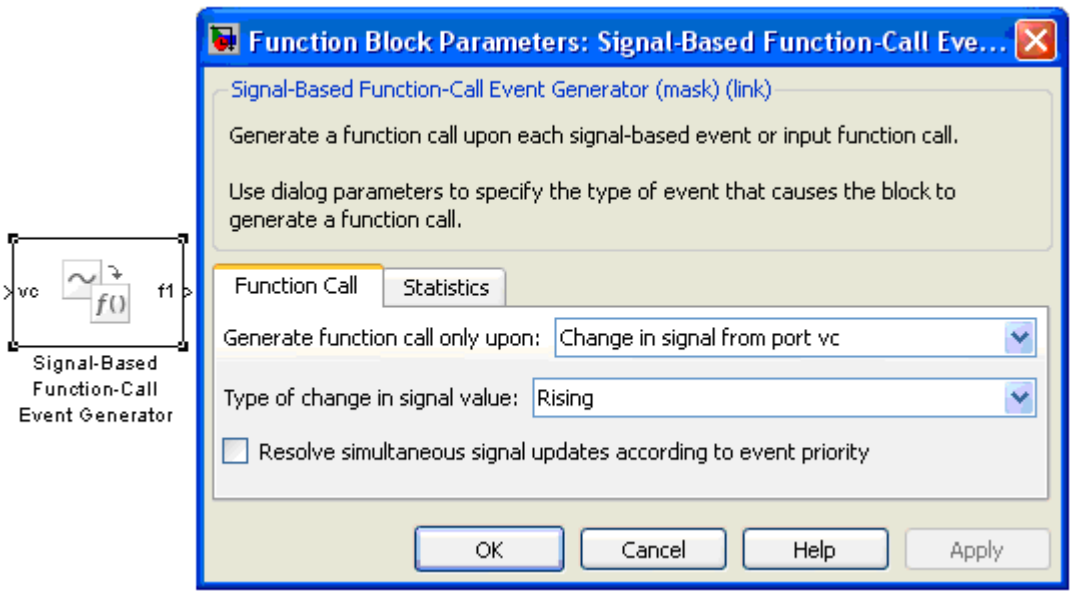


Рис. 14. Блок Signal Function Call Event Generator

Опція **Resolve simultaneous signal updates according to event priority** (Рішення для одночасного поновлення сигналу відповідно до пріоритету події) слугує для управління послідовністю подій виклику функцій відносно до інших одночасних подій у даній симуляції. Якщо ця опція не відмічена, то програма викликає функцію моментально як тільки отримує сигнал події, яка викликає цю функцию. Якщо блок має два типи вводу (виклик функції та сигнальний), вибір опції запобігає затримці сигналу.

Закладка статистики містить одне поле:

- **Number of f1 function calls** (кількість викликів функції)– управляє доступом та поведінкою вихідного порту **#f1.**

Приклад. На рис. 15 наведений фрагмент моделі СМО з врахуванням відмов у випадку поломки та ремонту сервера. Якщо значення сигналу на вихідному порту #n блоку сервера підвищений, то блок Signal-Based Function-Call Event Generator звертається до функції, яка викликає діаграму Stateflow для зміни стану сервера з робочого на неробочий. Завершення ремонту сервера моделюється як вихід з блоку Repair Work. Коли значення вихідного сигналу на порт #n зменшується, то блок Signal-Based Function-Call Event Generator звертається до Stateflow для зміни стану сервера з неробочого на робочий.

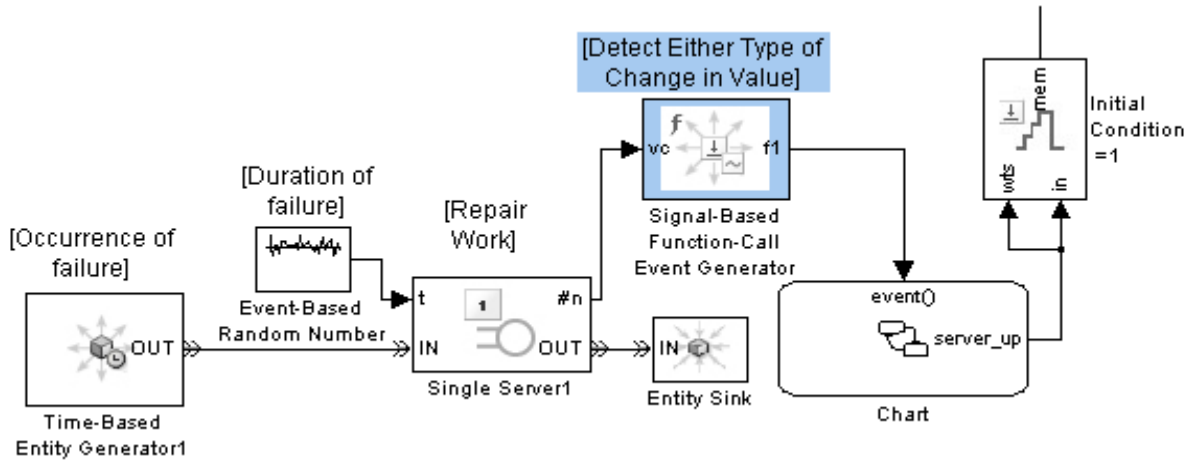


Рис. 15. Приклад фрагменту моделі з використанням блоку Signal Function Call Event Generator

Використання блоку Signal-Based Function-Call Event Generator замість тригеру дозволяє генерувати сутності з нульовим часом.

1.2.3. Signal Generators – генератори сигналів

1.2.3.1. Event Based Random Number

Event Based Random Number – блок генерування випадкових чисел. Блок генерує випадкове число в залежності від події, пов'язаної із наступним блоком, наприклад, з блоком Single Server, кожного разу, як подія надходить на вхідний порт #t, генерується нове випадкове число.

Вікно налаштування містить наступні поля:

- Distribution – дозволяє визначити закон розподілу для генерування послідовності випадкових чисел (таблиця 2);
- Initial seed – початкове значення генератора випадкових чисел (велике непарне число);
- поля для визначення характеристик відповідного закону розподілу (таблиця 2).

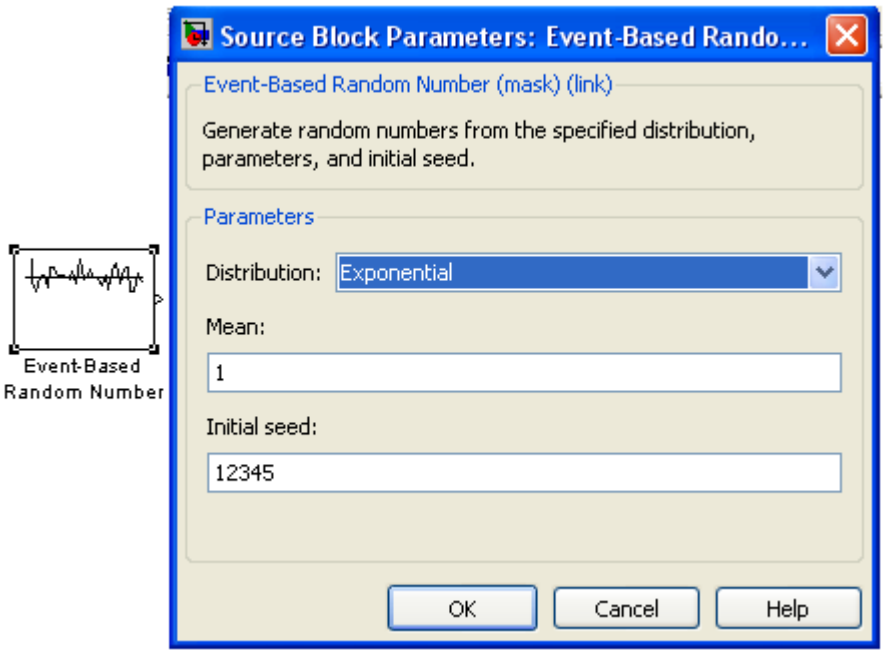


Рис. 16. Блок Event Based Random Number

Таблиця 2

Закон розподілу	Параметри, функція щільності розподілу
Exponential (експоненціальний)	Mean – μ **(**математичне сподівання**)** $f(x)=\begin{cases}\frac{1}{\mu}\exp\left(-\frac{x}{\mu}\right) & x\geq 0\\ 0 & x<0\end{cases}$
Uniform (рівномірний)	Minimum – L, Maximum – U $f(x)=\begin{cases}\frac{1}{U-L} & L\leq x\leq U\\ 0 & \textit{інакше}\end{cases}$
Bernoulli (Бернуллі)	Probability of 1 – $p\in[0,1]$ $f(x)=\begin{cases}p^x(1-p)^{1-x} & x=\overline{0,1}\\ 0 & \textit{інакше}\end{cases}$
Binomial (біноміальний)	Probability of success in a single trial – p (ймовірність події), Number of trials n – (кількість подій) $f(x)=\begin{cases}\frac{n!}{x!(n-x)!}p^x(1-p)^{n-x} & x=0,1.2,\ldots,n\\ 0 & \textit{інакше}\end{cases}$
Triangular	Minimum – L, Maximum – U, Mode – m (мода)

	$$f(x)=\begin{cases}\dfrac{2(x-L)}{(U-L)(m-L)} & L\le x\le m\\ \dfrac{2(U-x)}{(U-L)(U-m)} & m\le x\le U\\ 0 & \textit{інакше}\end{cases}$$
Gamma (гамма)	Threshold – θ, Scale – b, Shape – a, $\Gamma(\gamma)$ – гамма ф-ція $$f(x)=\begin{cases}\dfrac{\left(\dfrac{x-\theta}{b}\right)^{a-1}\exp\left(\dfrac{x-\theta}{b}\right)}{b\Gamma(\gamma)} & x\ge 0\\ 0 & \textit{інакше}\end{cases}$$
Gaussian (нормальний)	Mean – a (математичне сподівання), Standard deviation – σ (середньоквадратичне відхилення) $$f(x)=\frac{\exp\left[-(x-a)/(2\sigma^2)\right]}{\sigma\sqrt{2\pi}}$$
Geometric (геометричний)	Probability of success in a single trial – p (ймовірність події) $$f(x)=\begin{cases}p(1-p)^x & x=0,1,2,...\\ 0 & \textit{інакше}\end{cases}$$
Poisson (Пуассона)	Mean – λ (мат. сподівання) $$f(x)=\begin{cases}\dfrac{\exp(-\lambda)\lambda^x}{x!} & x=0,1.2,...\\ 0 & \textit{інакше}\end{cases}$$
Lognormal (логарифмічно-нормальний)	Threshold – θ, Mu – μ, Sigma – σ $$f(x)=\begin{cases}\dfrac{\exp\left(-\left(\dfrac{\ln(x-\theta)-\mu}{2\sigma^2}\right)^2\right)}{(x-\theta)\sigma\sqrt{2\pi}} & x\ge 0\\ 0 & \textit{інакше}\end{cases}$$
Log-logistic ()	Threshold – θ , Scale – b $$f_{logistic}(x)=\frac{1}{b}\cdot\frac{\exp((x-\theta)/b)}{[1+\exp((x-\theta)/b)]^2}$$
Beta (бета)	Minimum – L, Maximum – U, Shape parameter – a, Shape parameter – b $$f(x)=\begin{cases}\dfrac{(x-L)^{a-1}(U-x)^{b-1}}{(U-L)^{a+b+1}B(a,b)} & L\le x\le U\\ 0 & \textit{інакше}\end{cases},$$ $$B(a,b)=\int_0^1 t^{a-1}(1-t)^{b-1}dt$$

Discrete uniform (дискретний рівномірний)	Minimum – L, Maximum – U, Number of values – K $f(x)=\begin{cases}\frac{1}{K} & x=L+k\frac{U-l}{K-1},\ k=0,1,2,...,K-1\\ 0 & \textit{інакше}\end{cases}$
Weibull (Вейбулла)	Threshold – θ, Scale – α, Shape – γ $f(x)=\begin{cases}\frac{\gamma}{\alpha}\cdot\left(\frac{ч-\theta}{\alpha}\right)^{\gamma-1}\exp\left[-\left(\frac{x-\theta}{\alpha}\right)^{\gamma}\right] & x\geq 0\\ 0 & \textit{інакше}\end{cases}$
Arbitrary continuous (довільний неперервний)	Value vector – вектор значень, Cumulative probability function vector
Arbitrary discrete (довільний дискретний)	Value vector – вектор значень, Probability vector – вектор відповідних ймовірностей

Приклад. Застосування блоку Event Based Random Number з дискретним розподілом часових проміжків між подіями. Для налаштування блоку Event Based Random Number вибраний дискретний розподіл Arbitrary discrete з наступними значеннями:

- Value vector – вектор значень Δt – [1; 1.5; 2];
- Probability vector – вектор відповідних ймовірностей – [0.25; 0.5; 0.25], а саме $P(\Delta t=1)=0.25$, $P(\Delta t=1.5)=0.5$, $P(\Delta t=2)=0.25$.

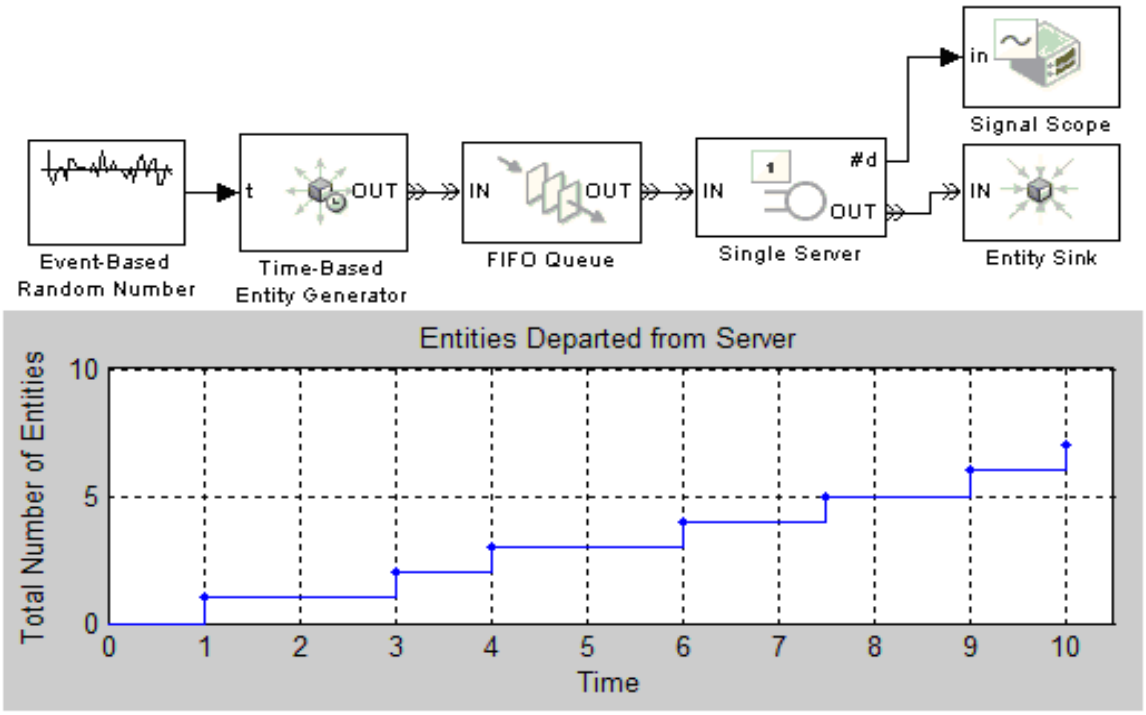


Рис. 17. Приклад застосування блоку Event Based Random Number

Для генерації замовлень використовується блок Time-Based Entity Generator з наступними налаштуваннями: Generate entities with – порт t, який з'єднує даний блок з блоком Event Based Random Number (рис.17).

На графіку, наведеному на рис.17, видно, що оброблені замовлення виходять з сервера з часовими проміжками 1, 1.5, 2 секунди.

1.2.3.2. Event Based Sequence

Event Based Sequence – генерує події на основі сигналів згідно до заданої послідовності у вигляді вектора стовпчика. Якщо блок під'єднати до одиничного сервера (Single Server), то він буде видавати нове значення кожного разу, коли замовлення буде надходити на обслуговування.

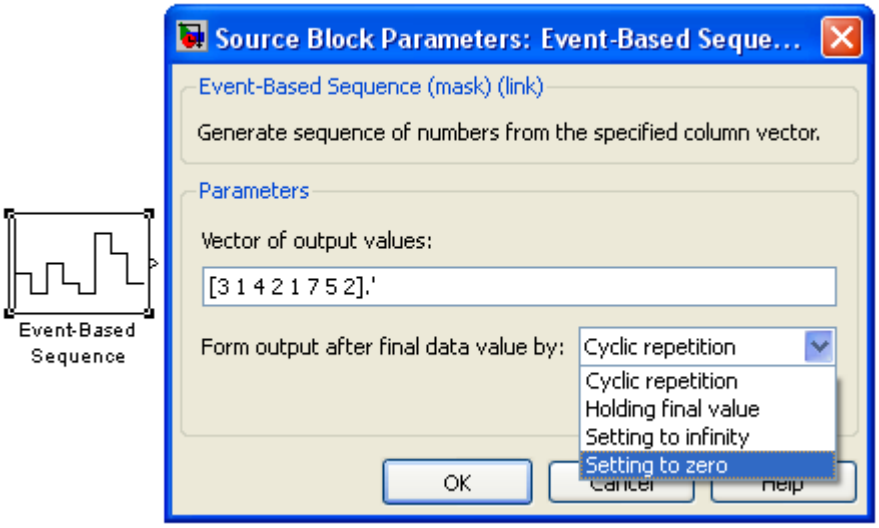


Рис. 18. Блок Event Based Sequence

Вікно налаштування містить да поля:

- **Vector of output values** – вхідний вектор-стовпчик, що задає послідовність;
- **Form output after final data value by** – метод генерації виводу після того, як послідовність буде вичерпана. Це поле може приймати наступні значення:
  - Cyclic repetition – циклічний повтор послідовності;
  - Holding final value – після того як послідовність буде вичерпана, утримується останнє значення;
  - Setting to infinity – після того як послідовність буде вичерпана, утримується значення нескінченності;
  - Setting to zero – після того, як послідовність буде вичерпана, утримується значення нуль.

Блок має обмежений набір допустимих з'єднань з іншими блоками, так як він призначений для виводу з наступних блоків при генерації нового числа. З'єднання повинно задовольняти наступним умовам:

• рівно одна лінія повинна підключатися до портів, перелічених у таблиці 3;

• жодної або декілька ліній можуть підключатися до портів моніторингу (таблиця 4);

• жодна лінія може не підключатися до інших портів; зокрема портів реакції (таблиця 5).

Таблиця 3. Перелік призначення портів

Вхідний порт Signal Input Port	Блок	Генерація нового вихідного значення при
A1, A2, A3 і т. д.	Set Attribute	надходженні сутності
in	Signal Latch	записі події
e1, e2	Entity Departure Event to Function-Call Event	надходженні сутності
	Signal-Based Event to Function-Call Event	відповідному сигналі, на основі подій, в залежності від конфігурації блоку
t	Signal-Based Event to Function-Call Event	відповідному сигналі, на основі подій, в залежності від конфігурації блоку
t	Infinite Server	надходженні сутності
	N-Server	
	Single Server	
t	Time-Based Entity Generator	старті процесу моделювання та подальшому виході сутностей
ti	Schedule Timeout	надходженні сутності
x	X-Y Signal Scope	надходженні проміжку часу від вхідного порту сигналу

Таблиця 4. Перелік портів моніторингу

Вхідний порт Signal Input Port	Блок
Unlabeled	Discrete Event Signal to Workspace
in	Signal Scope X-Y Signal Scope
ts, tr, vc	Instantaneous Event Counting Scope

Таблиця 5. Перелік портів дії

Вхідний порт Signal Input Port	Блок	Відповідне оновлення
en	Enabled Gate	зміна значення від відємного до додатнього, і навпаки
p	Input Switch Output Switch Path Combiner	зміна значення
ts, tr, vc	Entity Departure Counter	проміжок часу з порту ts відповідний тригер з порту tr відповідне значення з порту vc
	Event-Based Entity Generator	
	Release Gate	
	Signal-Based Event to Function-Call Event	
	Signal-Based Function-Call Event Generator	
wts, wtr, wvc, rts, rtr, rvc	Signal Latch	проміжок часу з порту wts або rts відповідний тригер з порту wts або rts відповідна зміна значення з порту wts або rts
Input port corresponding to Discrete Event Inport block in subsystem	Discrete Event Subsystem	проміжок часу з вхідного порту
Unlabeled input port	Initial Value	проміжок часу

Блок має один вихідний порт для чисел заданої послідовності. Початкове значення виходу, яке діє з початку моделювання до першого оновлення, дорівнює 0. Блок не має порту для сутностей , або введення сигналу.

Приклад. У наведеному нижче пикладі блок з нескінченою кількістю серверів може завершувати обробку для декількох замовлень одночасно.

Блок Instantaneous Entity Counting Scope (підрахунок обсягу сутностей) показує скільки замовлень вийшло в кожний фіксований момент часу на протязі прогону моделі.

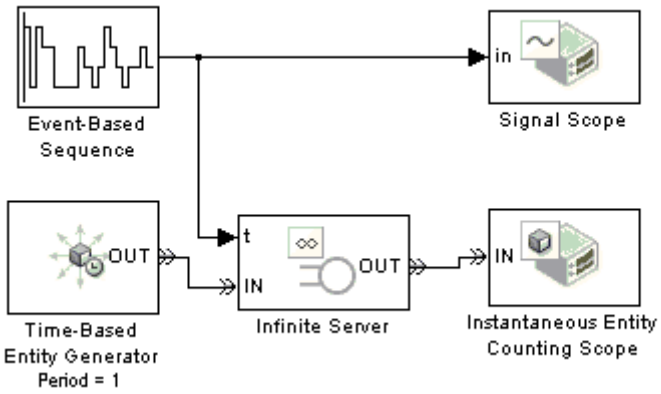


Рис. 19. Приклад використання блоку Event Based Sequence

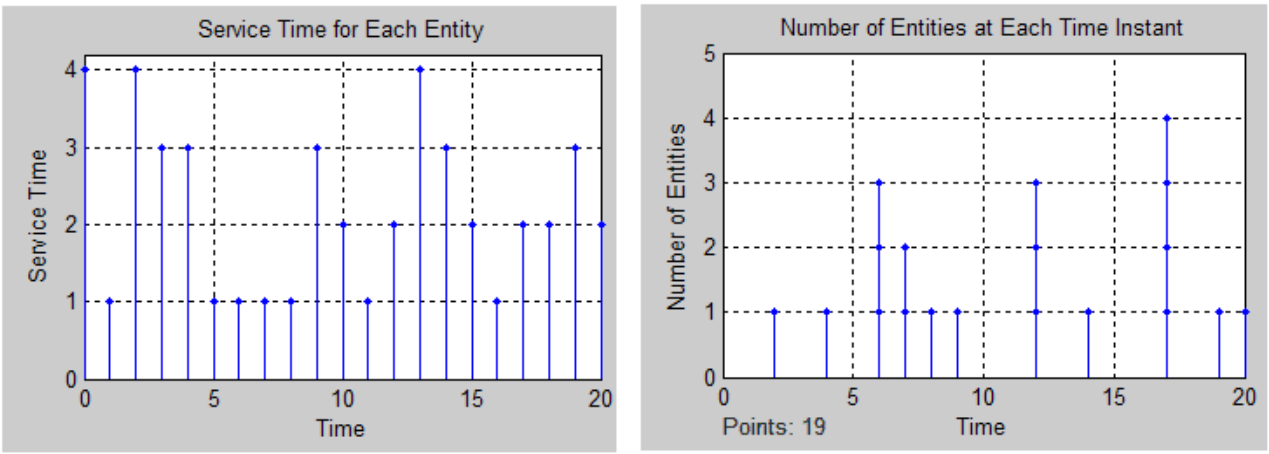


Рис. 20. Результат роботи моделі з використанням блоку Event Based Sequence

Управляюча послідовність: [4 1 4 3 3 1 1 1 1 3 2 1 2 4 3 2 1 2 2 3 2]' циклічно повторюється. Блок Time Based Entity Generator починає роботу з моменту старту, визначений як генератор з паузами і має розподіл Constant з періодом 1.

1.3. Черга (Queue)

Для моделювання черги застосовуються блоки, наведені на рис. 21, які відрізняються порядком (дисципліною) обробки черги:

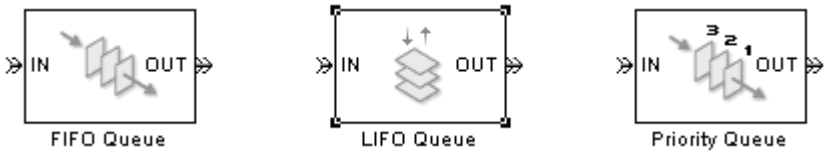


Рис. 21.

- FIFO (first input first ounrun) – черга, яка обслуговується за принципом " перший прийшов – перший вийшов";
- LIFO (last input first output) – черга, що обслуговується за принципом стеку – "перший прийшов – останній вийшов;

- Priority Queue – черга з пріоритетом обслуговування.

Блоки мають однакову структуру і однакові поля для налаштування. Тільки блок Priority Queue містить на першій вкладці два додаткові поля, де визначається дисципліна пріоритету та порядок сортування (рис. 22).

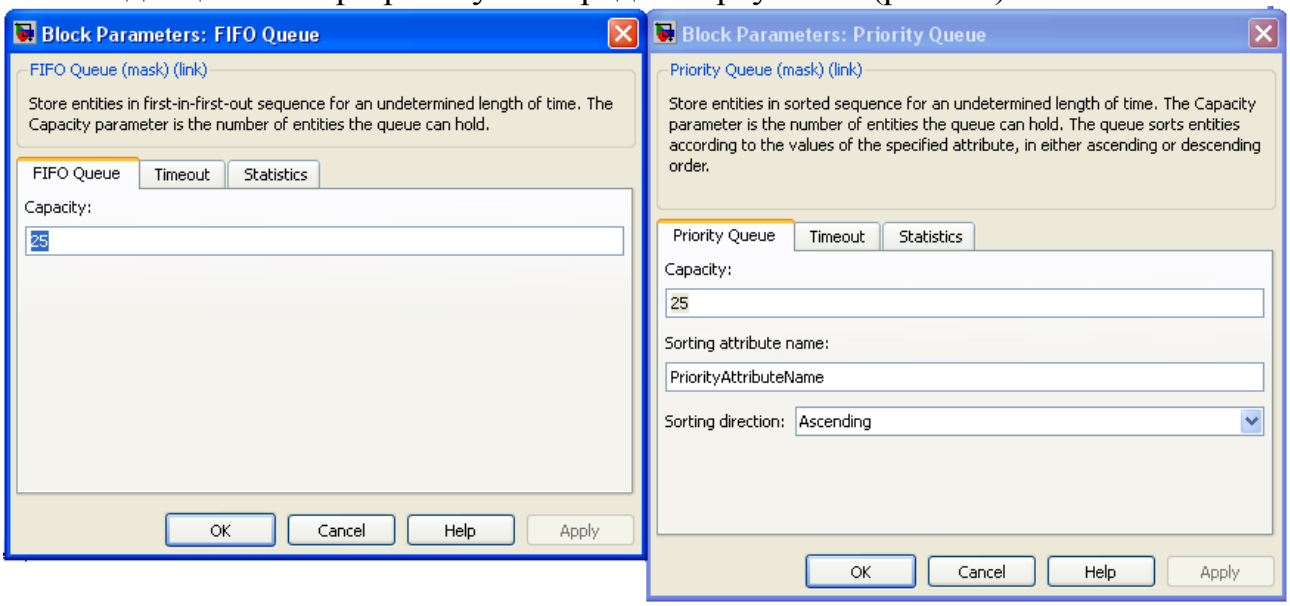


Рис. 22. Головна закладка діалогового вікна блоків черги

Для налаштування блоків необхідно задати довжину черги **capacity** – натуральне число. Якщо всі місця черги зайняті, то вхідний порт IN недоступний і замовлення відхиляється, якщо вихідний порт заблокований (наприклад, всі сервіси зайняті), то замовлення лишається в блоці. Блок пропускає замовлення згідно до дисципліни черги.

Друга закладка Timtout містить одне поле Enable TO port for timed-out entities, вибір якого дозволяє зробити доступним порт TO (timed-out). Це поле актуальне у випадку, коли передбачається обмеження часу перебування замовлення у черзі. Якщо час перебування замовлення у черзі закінчується, замовлення виходить з блоку.

Закладка налаштування статистики наведена рис. 23 і містить наступні поля:

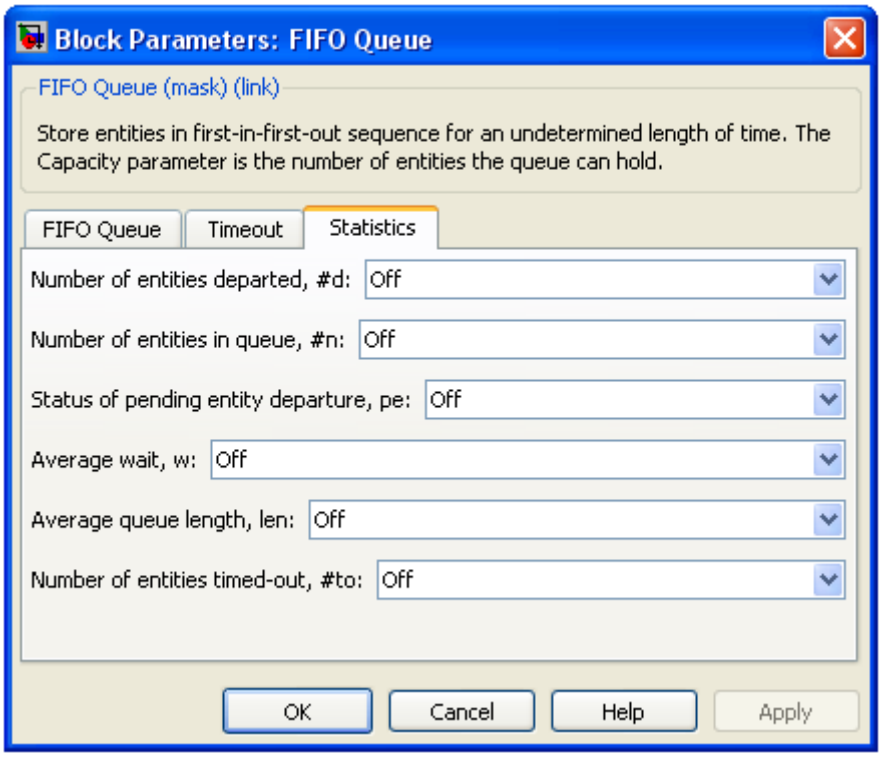


Рис. 23. Налаштування статистики блоків черги

- **Number of entities departed, #d** – число замовлень, що вийшли через вихідний порт OUT після початку моделювання;
- **Number of entities in queue, ; #n** – кількість замовлень у черзі;
- **Status of pending entity departure pe** – контролює наявність з вихідного порту сигналу з написом **pe.**
- **Average wait, w** – середній час очікування у черзі;
- **Average queue length, len** – середня довжина черги;
- **Number of entities timed out, #to** – контролює присутність та поведінку вихідного порту.

Приклад 1. В наведеному нижче прикладі (рис. 24) порівнюються черги FIFO та LIFO для СМО nbge (D/D/1) з часом генерації замовлень 0.3 та часом обслуговування 1.

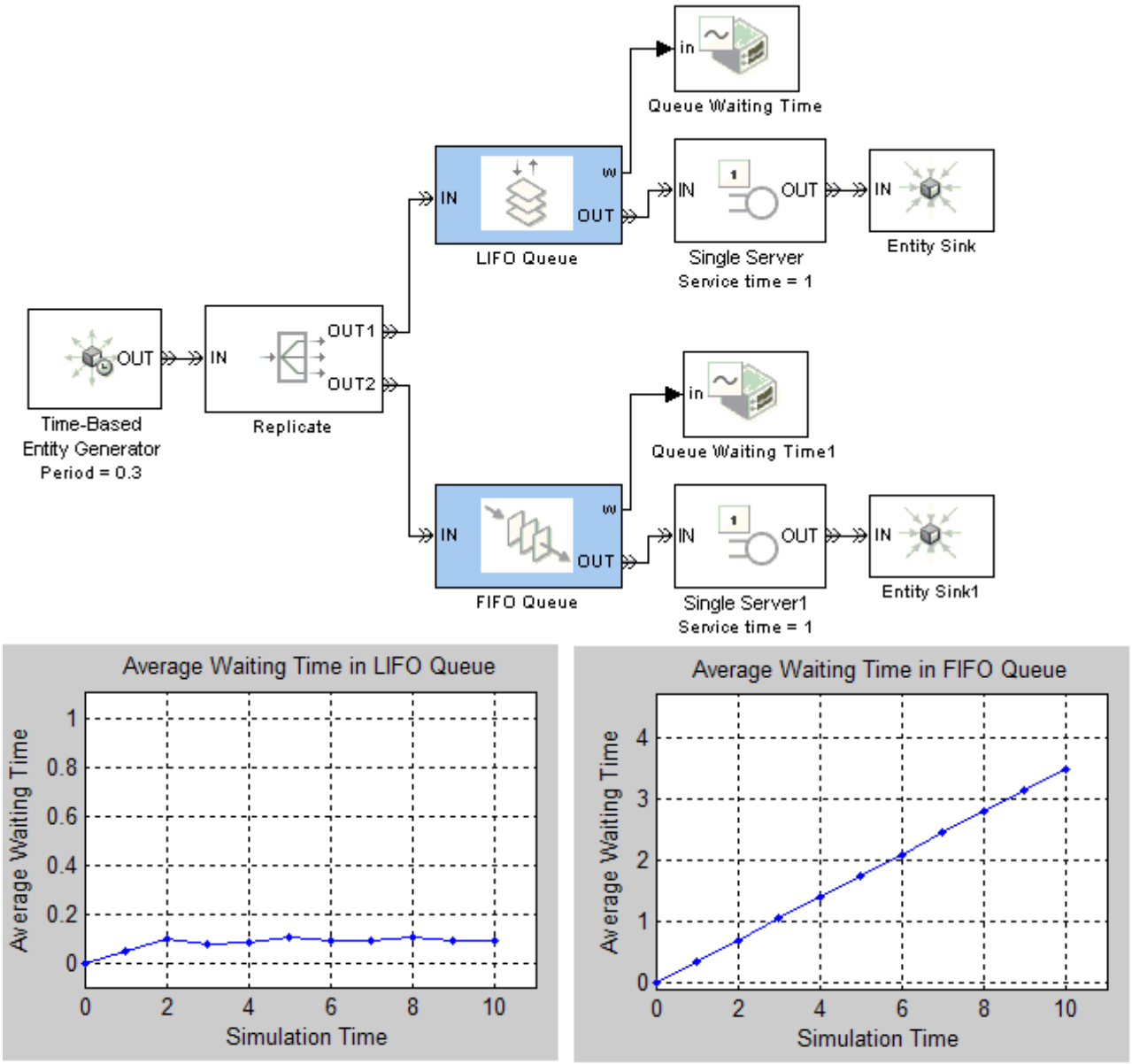


Рис. 24. Приклад моделі з застосуванням черг та результатів її роботи

Приклад 2. В даному прикладі в чергу надходить два типи замовлень, де одне є привілейованим, але відрізок часу його генерації більший. Для задання привілейованих замовлень використовують блоки встановлення атрибутів, які описані нижче. Привілейовані замовлення в черзі розташовуються попереду звичайних замовлень.

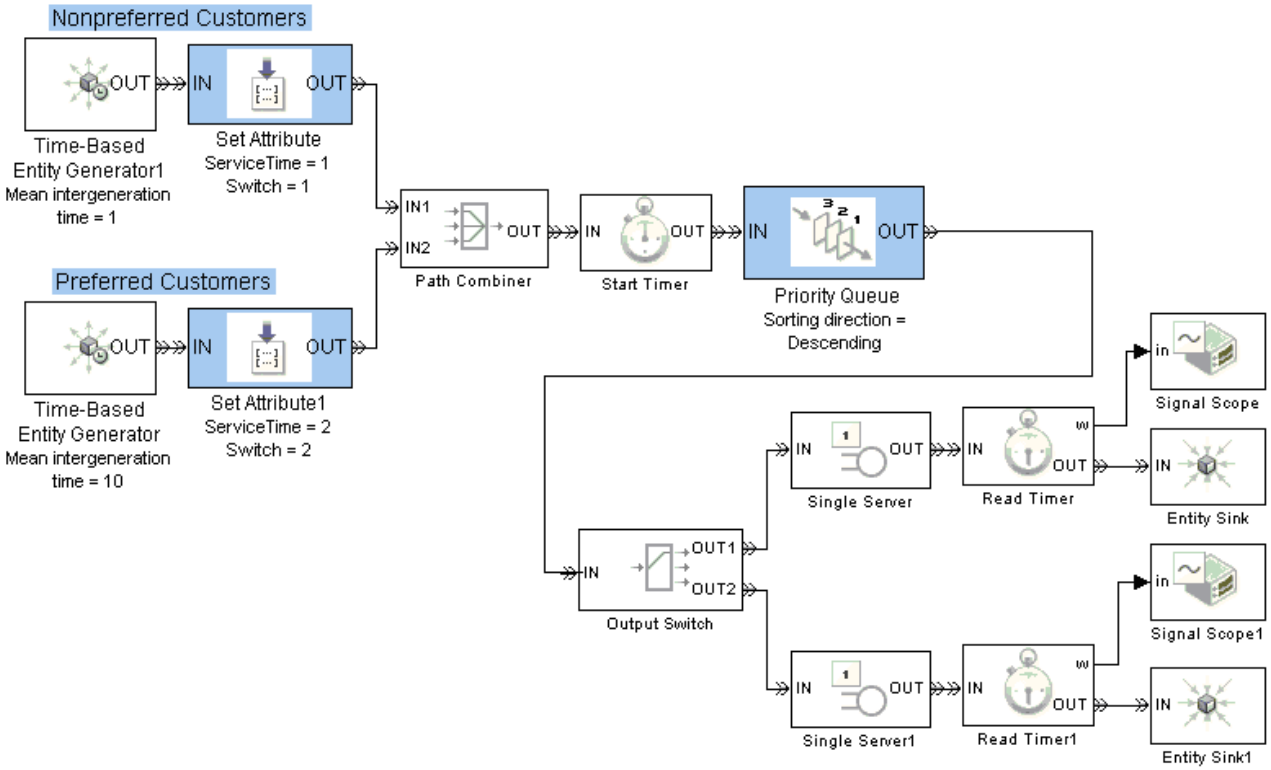


Рис. 25. Приклад моделі з чергою замовлень з пріоритетами

Результатом роботи моделі є графіки середнього системного часу для непривілейованих та привілейованих замовлень. Параметри налаштувань наведені під відповідним блоком.

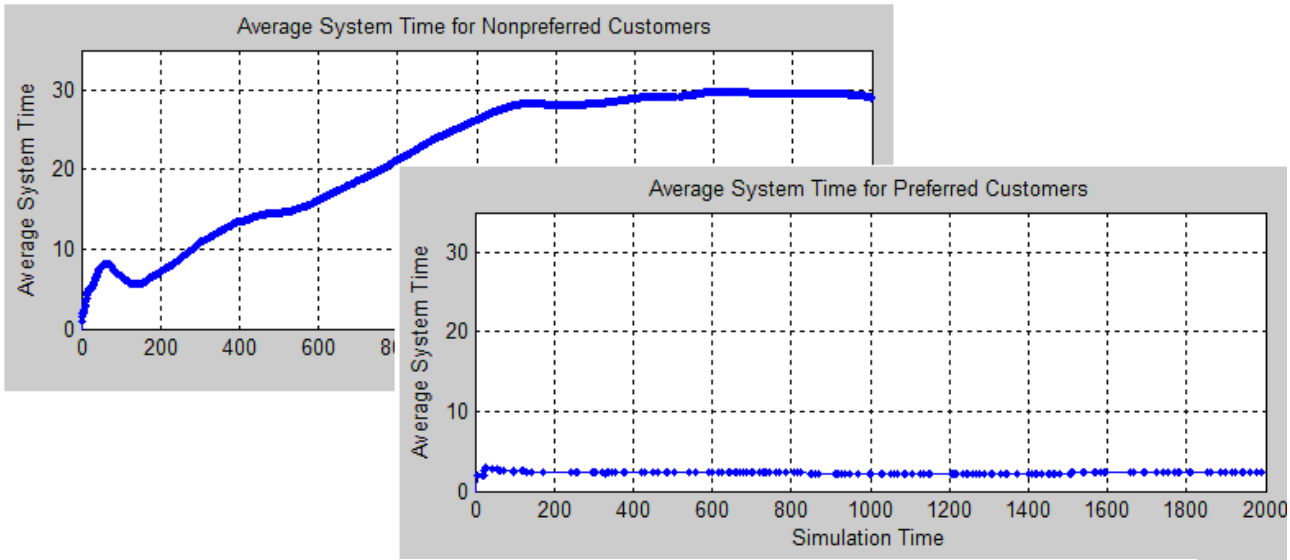


Рис. 26. Результат прогону моделі з чергою замовлень з пріоритетами

1.4. Атрибути (Attributes)

Використання атрибутів дозволяє управляти чергами з пріоритетами, подіями з обмеженим часом (тайм-аут) та потоками сутностей.

Бібліотека містить два графічних блоки (рис.27):

- **Get Attribute** (взяти атрибути) – блок дозволяє визначити атрибути сутності та їх значення для подальшого використання, лишаючи їх без змін (рис. 28);

- **Set Attribute (**встановити атрибути**) –** блок дозволяє призначити атрибути сутностям **(**рис. 29**).**

Кожний атрибут має своє і'мя та значення.

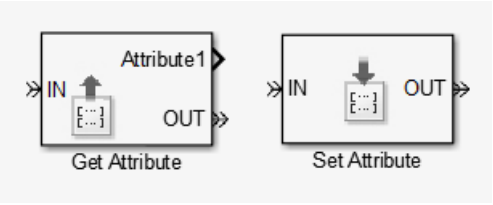


Рис. 27. Блоки управління атрибутами

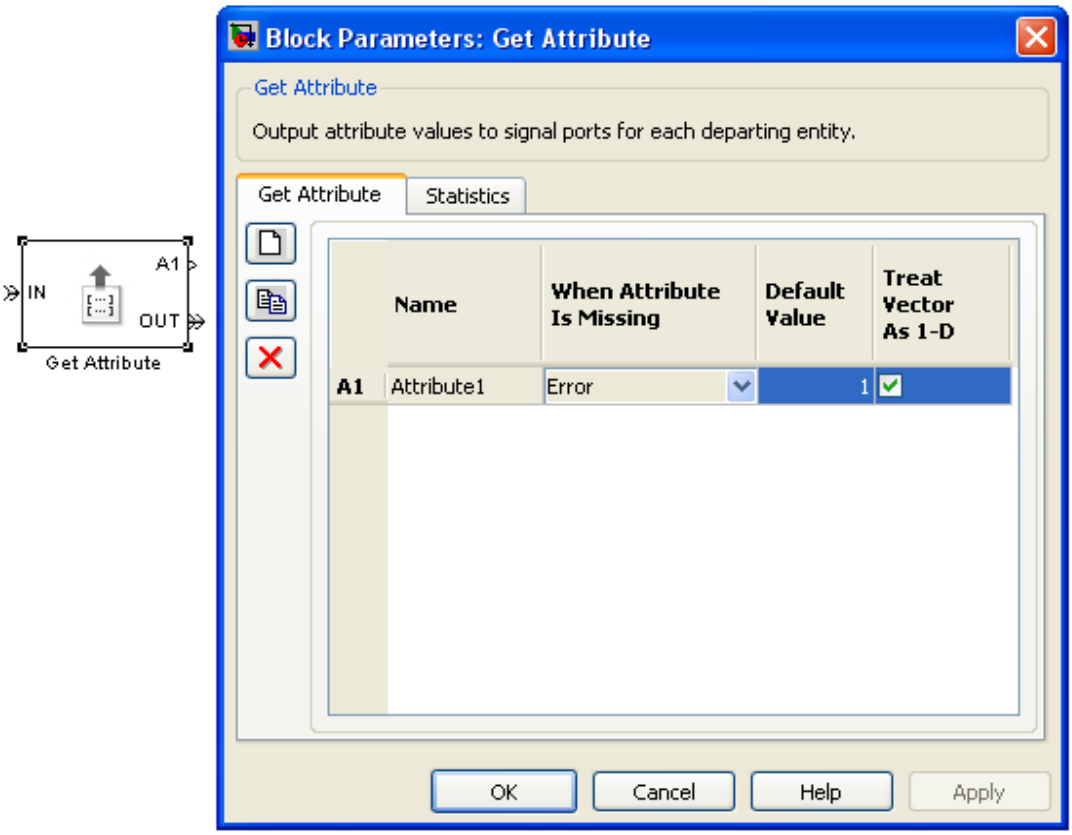


Рис. 28. Блок визначення атрибутів Get Attribute та його діалогове вікно

На панелі блоків є наступні клавіші налаштування:

 – створити поле атрибута;

 – копіювати поле атрибута;

 – видалити поле атрибута

та дві закладки:

- налаштування параметрів атрибуту;
- статистика.

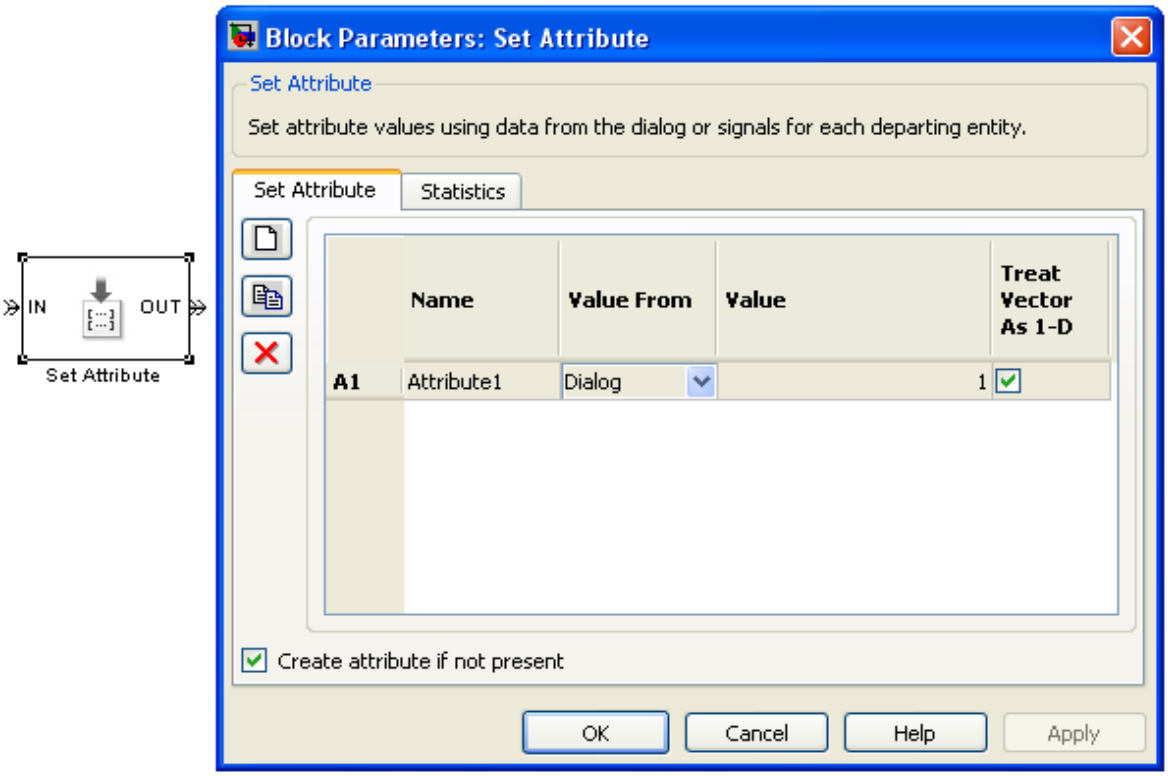


Рис. 29. Блок встановлення атрибутів Set Attribute та його діалогове вікно

Параметри налаштування представлені у вигляді таблиці з наступними полями:.

- **Name** – ім'я атрибуту;
- **When Attribute Is Missing (блок Get Attribute)** – це поле дозволяє управляти атрибутами сутностей, які не зазначене у таблиці і може встановлюватись у наступні значення:
  - Error – блок видає повідомлення про помилку і зупиняє моделювання. У цьому випадку значення за замовчуванням (Default Value – значення вихідного сигналу**)** і вектор корегування (Treat Vector as 1-D – цей параметр управляє відсутніми атрибутами, встановлюючи їх значення за замовчуванням або попереджаючи про помилку) розміщуються у рядку таблиці атрибутів, які є неправильними.
  - **Default Value** – встановлює значення вихідного сигналу, коли сутність відсутня у таблиці імен атрибутів та продовжує процес моделювання;
  - **Warn** – блок виводить встановлене користувачем значення за замовчуванням Default Value та значення вектора Treat Vector as 1-D та продовжує процес моделювання;
- **Value From (блок Set Attribute)** – визначає джерело отримання атрибуту, може приймати значення:
  - **Dialog** – діалог;
  - **Signal Port**– сигнальний порт;
- **Default Value** – значення вихідного сигналу, коли сутність відсутня у таблиці імен атрибутів. (Див. більш докладну інформацію в Attribute Value

Support). В цьому випадку необхідно задати значення полів Default Value та Treat Vector as 1-D.

- **Treat Vector as 1-D** – параметр управляє відсутніми або помилковими атрибутами. Якщо параметр є вибраним, то встановлюється довжина N вектора значень за замовчуванням.

Закладка статистики має лише одне поле:

- **Number jf entities departed #d** – кількість замовлень, що пройшли через блок.

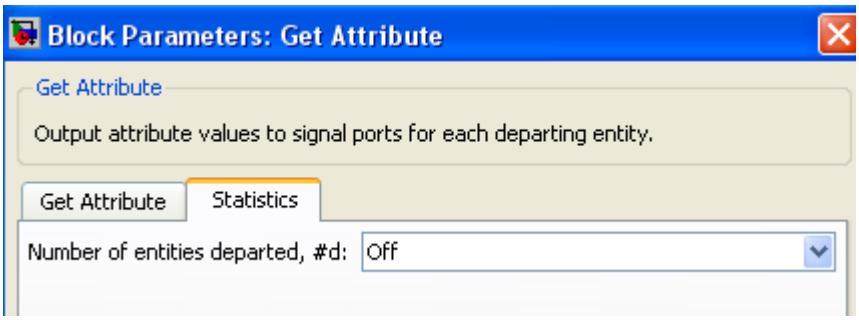


Рис. 30. Закладка статистики блоку атрибутів

Блоки мають один вхідний (**IN**) та один вихідний (**OUT**) порт для замовлень. Блок Get Attribute має додатковий вихід (а Set Attribute – вхід) A*x*, де *x* = 1, 2, 3,… для визначення значення атрибуту, розташованого у A*x*-рядку таблиці атрибутів. Початкове значення для вихідного порту дорівнює нулю.

Приклад підпрограми з використанням блоків управління атрибутами сутностей (рис. 31).

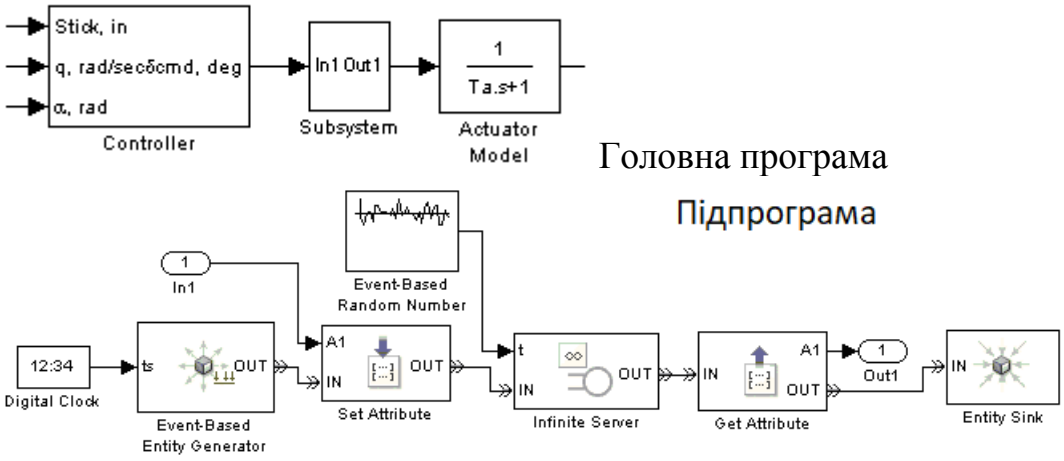


Рис. 31. Фрагмент моделі СМО з використанням блоків атрибутів

В даному прикладі на основі події зміни дискретного часу генеруються замовлення та встановлюються їх атрибути, що задаються із зовні (рис. 32).

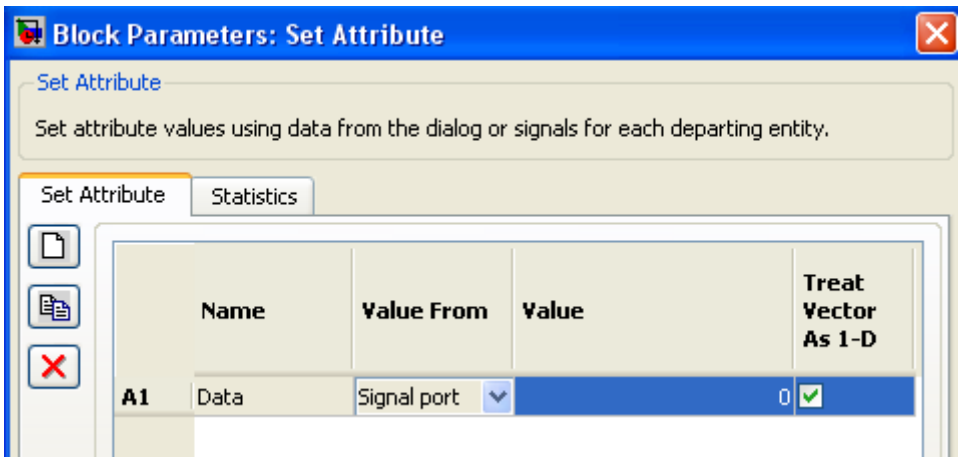


Рис. 32. Приклад задання атрибуту

Замовлення надходять на обслуговування до блоку Infinite Server (необмежена кількість серверів), який встановлює графік обслуговування на основі послідовності випадкових чисел. Після обслуговування атрибути замовлень передаються у головну програму.

1.5. Управління потоками (Entity Management)

У сучасних версіях середовища Matlab для управління потоками замовлень пердбачено шість графічних блоків (рис.33).

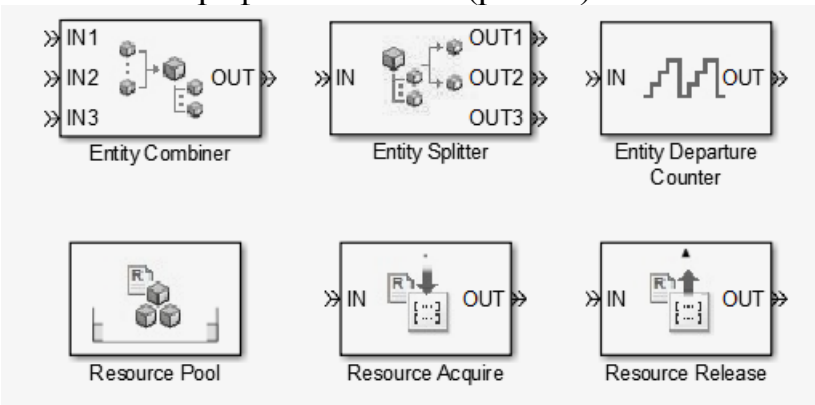


Рис. 33. Графічні блоки управління потоками

В даному розділі розглянемо основні блоки:

- **Entity Combiner** – злиття потоків;
- **Entity Splitter** – роз'єднання потоків.

1.5.1. Entity Combiner (злиття потоків)

Блок генерирує один новий об'єкт (дане типу запис) для кожного набору сутностей, що одночасно надходять на декілька вхідних портів IN, який є вихідним для порту OUT, що визначає кількість сутностей, що пройшли через блок. Вхідна сутність повинна включати інформацію про структуру, атрибути та часові характеристики. Обрані поля в цьому блоці визначають, чи можуть інші блоки звернутись до параметрів сутності і можливість зворотньої операції

– роз'єднання. Деякі значення параметрів вимагають унікальності імен атрибутів або тегів таймеру.

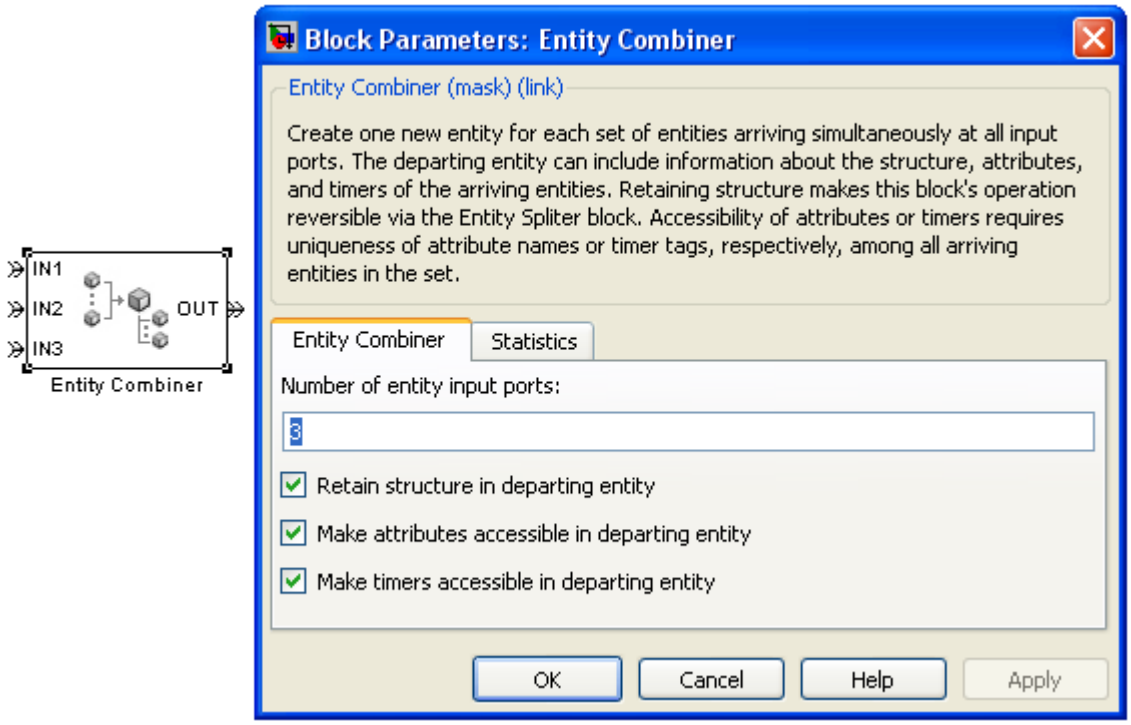


Рис. 34. Блок Entity Combiner

На головній закладці блоку Entity Combiner передбачені наступні поля:

- **Number of entity input ports** – визначає кількість вхідних портів;
- **Retain structure in departing entity** – вибір цього блоку означає, що вихідна сутність несе інформацію про складові елементи запису, яка дає можливість відновити складові об'єкти компоненту за допомогою блоку Entity Splitter;
- **Make attributes accessible in departing entity** – вибір цієї опції дає доступ до атрибутів компонентів вихідних сутностей. Ім'я поля залежить від вибору поля Retain structure in departing entity;
- **Make timers accessible in departing entity** – вибір цього поля дозволяє використовувати таймери вихідних сутностей. Ім'я поля залежить від вибору поля Retain structure in departing entity.

Закладка статистики містить лише одне поле – **Number jf entities departed #d** – кількість замовлень, що пройшли через блок.

Приклад використання блоку Entity Combiner наведений нижче у розділі Entity Splitter (роз’єднання потоків).

1.5.2. Entity Splitter (роз’єднання потоків)

Блок розділяє складену сутність, утворену за допомогою блоку об'єднання, яка надходить на порт IN на компоненти, які виходять через відповідні порти OUT (рис. 35).

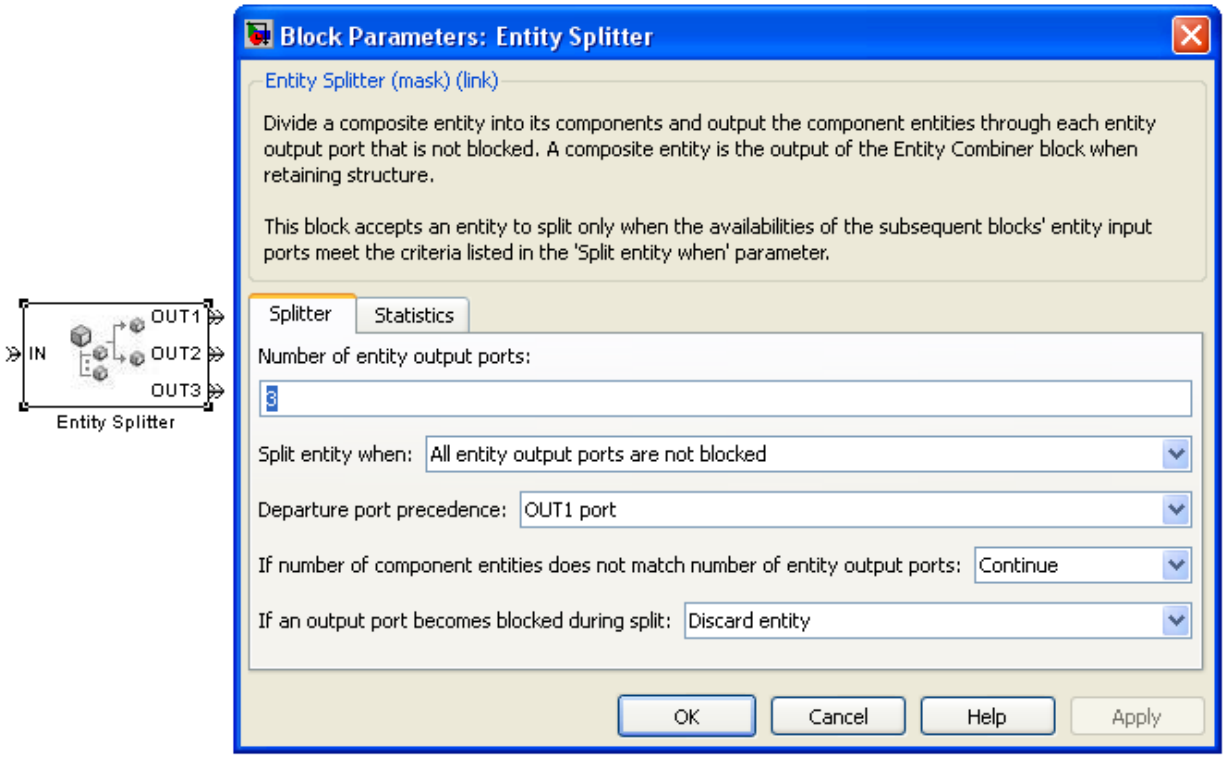


Рис. 35. Блок Entity Splitter

Кількість вихідних портів повинна відповідати кількості входів блоку **Entity Combiner**. Атрибути та параметри часу зберігаються і визначаються вхідними значеннями сутностей, які надійшли на блок об'єднання.

На головній закладці розташовані наступні поля:

- **Number of entity output ports** – визначає кількість вихідних портів, параметр має символьний тип;

- **Split entity when (розбиття потоків)** – визначає доступність блоку для сутностей. Блок доступний, коли хоча б одна із сутностей виходить або всі вихідні порти вільні. Поле може приймати наступні значення:

- **All entity output ports are not blocked** – розбиття потоків виконується лише в тому випадку, коли всі сутності мають можливість виходу;

- **Any entity output port is not blocked** – розбиття потоків виконується в Departure port precedence – визначає початок послідовності при роз'єднанні потоків. Значення поля наведені у таблиці.

Таблиця 6.

Значення	Призначення	Приклад
OUT1 port	При розбитті потоків вони послідовно надходять на вихідні порти OUT1, OUT2, OUT3,..., у заданому порядку	Потоки надходять на порти OUT1, OUT2, OUT3
Round robin	При розбитті потоків перший раз сутності	Для структури з трьох сутностей перший раз

	послідовно надходить на вихідні порти, а далі їх послідовність виходів змінюється	послідовність виходів буде OUT1, OUT2, OUT3 другий раз – OUT2, OUT3, OUT1, третій – OUT3, OUT1, OUT2 і т. д.
Equipr obable	При розбитті потоків перший вихідний порт вибирається випадковим чином, а наступні – згідно до встановленої послідов-ності. Всі порти мають однакову ймовірність вибору	Для структури з чотирьох сутностей, якщо першим був обраний третій порт, то виходи будуть встановлені у порядку OUT3, OUT4, OUT1, OUT2. Якщо першим визначений другий порт, то порядок виходів буде OUT2, OUT3, OUT4, OUT1.

- **If number of component entities does not match number of entity output ports** – визначає дію у випадку, коли кількість вхідних потоків не відповідає кількості виходів. Значення "Continue" означає, що блок ігнорує додаткові вихідні порти та відкидає зайві обєкти.

- **If an output port becomes blocked during split** – визначає, чи буде видаватись повідомлення у випадку, коли вихідний порт блокується в процесі розділення потоків. Поле доступне тільки у випадку, коли поле Split entity when має значення All entity output ports are not blocked (всі вихідні порти не блокуються). Значення поля наведені у таблиці.

Таблиця 7.

Значення	Призначення
Discard entity	виконується блокування компонентів, які повинні виходити через попередньо заблоковані виходи
Warn and discard entity	Видається попереджувальне повідомлення про блокування компонентів, які повинні виходити через попередньо заблоковані виходи
Error	Припинення симуляції з виведенням попереджувального повідомлення

Закладка статистика містить наступні поля:

- **Number of entities arrived** – #a;

- **Number of entities departed** – контролює вихідний порт сигналів з позначкою #d.

Призначення портів наведено у таблиці.

Таблиця 8.

Позначення	Призначення	Час поновлення статистики	Порядок поновлення
#a	Кількість сутностей, що	Після	1

	надійшли у блок з початку процесу моделювання	надходження сутності	
#d	Кількість сутностей, що вийшли з блоку з початку процесу моделювання	Після виходу сутності	2

Початкові значення для всіх виходів на початку процесу моделювання встановлюється рівними 0.

Приклад. У наведеній нижче моделі об'єднуються три сутності (header – заголовок, payload – вартість, trailer – вантаж) у одну комбіновану структуру. Атрибутом кожного компонента є його довжина. Розділення структури виконується з урахуванням цього атрибуту, але в середині самої структури атрибут є недоступним, бо у випадку різної довжини імена атрибутів сутностей співпадають, що призведе до неоднозначності атрибуту усередині структури .

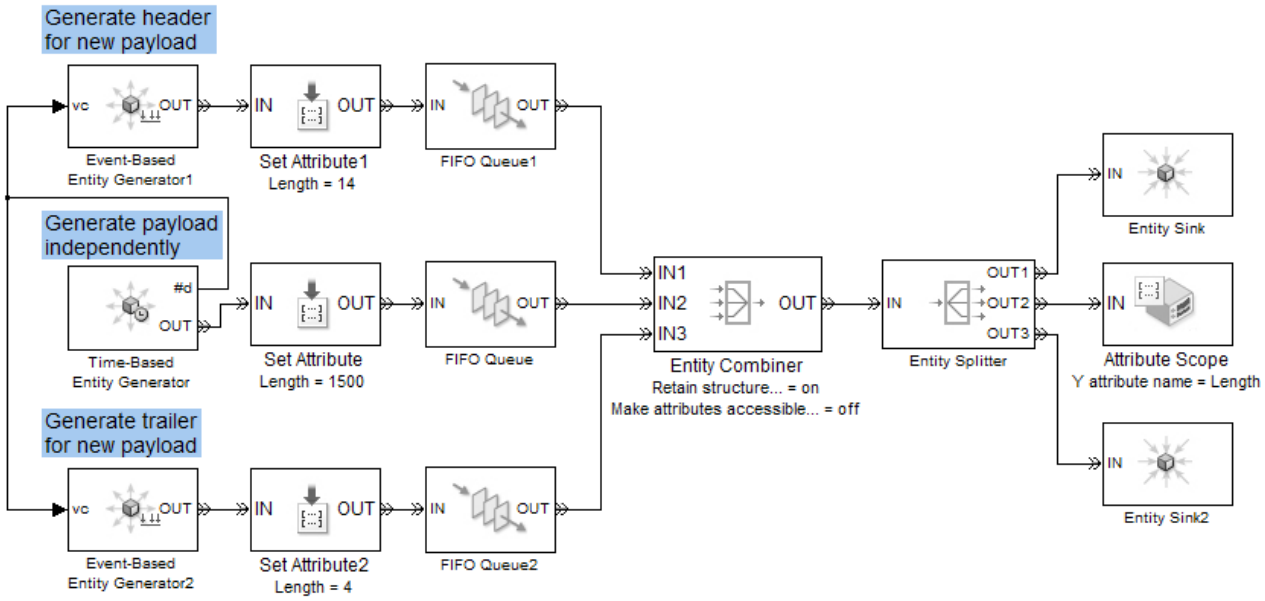


Рис. 36. Приклад використання блоків об'єднання та роз'єднання потоків

1.5.3. Управління сигналами (Signal Management)

Блоки цієї бібліотеки дозволяють управляти сигналами. Бібліотека складається з двох блоків:

- **Ininial value** – початкове значення;
- **Signal Latch** – фіксація сигналу.

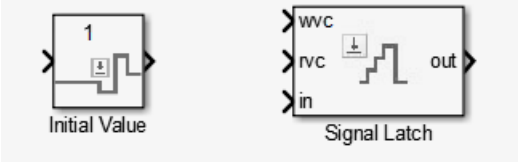


Рис. 37. Блоки управління сигналами

1.5.3.1. Ininial value (Початкове значення)

Блок встановлює початкове значення для сигналу на основі події (рис. 38). Для першого значення входу та всіх наступних надходжень вихідний сигнал ідентичний вхідному сигналу.

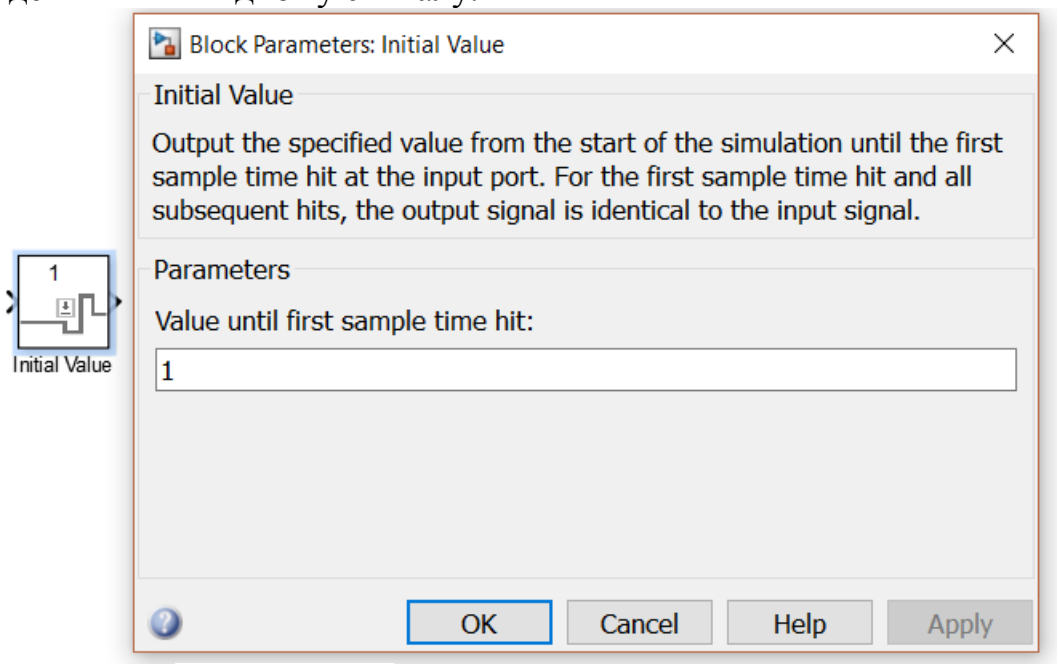


Рис. 38. Блок Ininial value та його діалогове вікно

Блок має один вхідний і вихідний порти без назв. Сигнал, що надходить в блок зупиняє використання початкового значення. Сигнал на виході приймає або значення, зазначене у діалоговому вікні блоку, або значення вхідного сигналу, залежно від того, чи надходить вхідний сигнал повторно.

Діалогове вікно містить лише одну вкладку з одним полем: **Value until first time hit** – значення до першого надходження.

Приклад.

Наступний фрагмент моделі ілюструє використання блоку у зворотньому зв'язку (рис. 39). Коли розпочинається симуляція, блок початкового значення забезпечує вихідне значення 1, яке відкриває ворота, щоб перша сутність могла перейти до циклу зворотнього зв'язку. Без ненульового початкового значення, жодна сутність не надійде на сервери, а блок блокування сигналу ніколи не зазнає жодних подій.

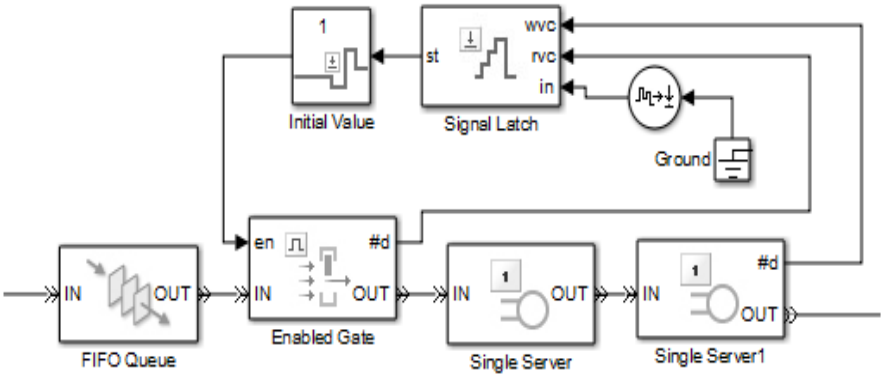


Рис.39. Фрагмент моделі з використанням блоку Ininial value

1.5.3.2. Signal Latch

Signal Latch (фіксація сигналу) – є універсальним блоком для керування сигналами на основі подій і може використовуватись для затримки або повторного використання сигналу на основі подій, а не часу. Цей блок зберігає та виводить значення вхідного сигналу на основі подій.

Блок записує значення сигналу у внутрішній пам'яті, коли відбувається подія "**write to memory**" та зчитує значення пам'яті та оновлює сигнал у вихідному порту, якщо він присутній, коли відбувається подія "**read fron memory**".

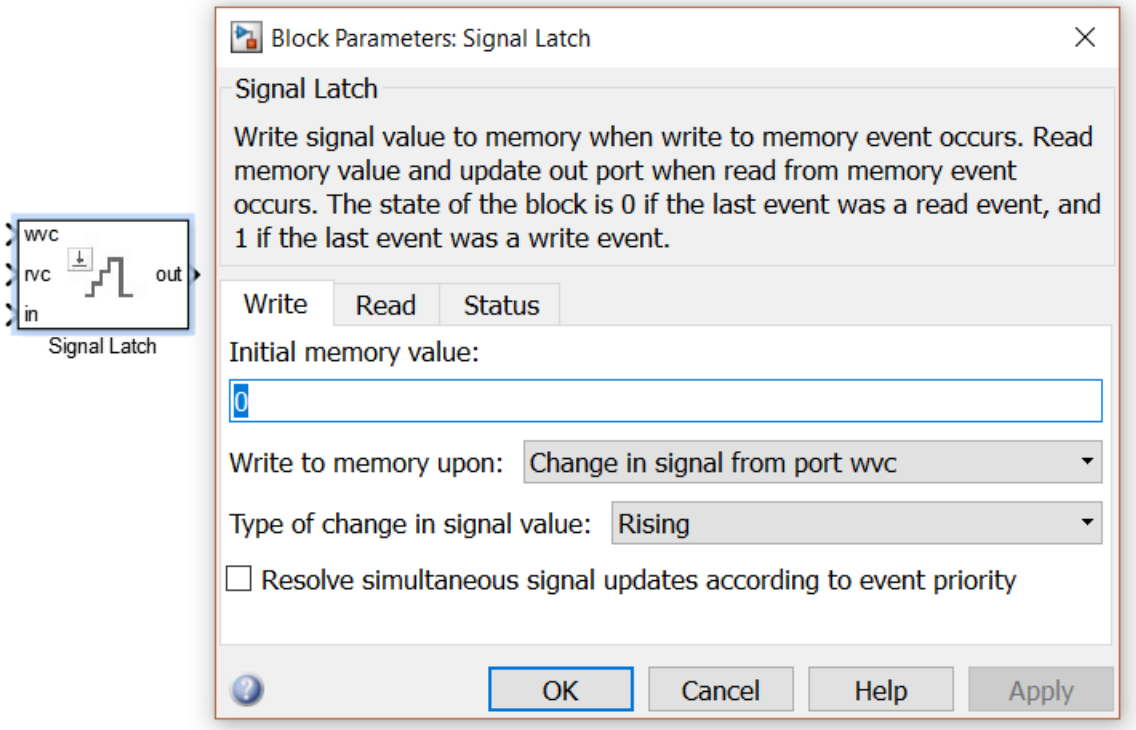


Рис. 40. Блок Signal Latch та його діалогове вікно

Цей блок корисний для моделювання циклів зворотнього зв'язку в системах дискретних подій, в яких вихід з одного компонента є входом до іншого компонента. Оскільки компоненти працюють незалежно, то оновлення

вхідного та вихідного для них є також незалежним. Блок дозволяє контролювати причинність та час, пов'язаний із збереженням вихідного сигналу з одного компонента та оновленням значення, отриманого іншим компонентом.

Блок може працювати з широким набором вхідних та вихідних портів (табл.9 , табл.10).

Таблиця 9. Вхідні порти блоку Signal Latch

Позначення	Опис
wts	Сигнал, що ініціює запис подій. Цей сигнал повинен бути сигналом на основі події. Порт доступний тільки в тому випадку, якщо ви встановите **Write to memory upon** (Записати у пам'ять), щоб отримати значення встановленого часу із порту wts.
wtr	Тригерний сигнал, межа якого викликає записи подій. Цей сигнал повинен бути сигналом на основі події. Порт доступний, якщо встановлено **Write to memory.**
wvc	Сигнал, зміна значення якого призводить до запису подій. Цей сигнал повинен бути сигналом на основі події. Порт доступний, якщо встановлено **Write to memory**, щоб змінити сигнал з порту wvc.
wfcn	Заданий функцією сигнал, який викликає записи подій. Цей сигнал повинен бути викликом функції на основі події. Порт доступний, якщо встановлено **Write to memory.**
rts	Сигнал, чиє оновлення викликає читання подій. Цей сигнал повинен бути сигналом на основі події. Порт доступний, якщо встановлено **Read from memory upon.**
rtr	Тригерний сигнал, що викликає читання подій. Цей сигнал повинен бути сигналом на основі події. Порт доступний, якщо встановлено **Read from memory upon.**
rvc	Сигнал, зміна значення якого призводить до читання подій. Цей сигнал повинен бути сигналом на основі події. Порт доступний, якщо встановлено **Read from memory upon.**
rfcn	Сигнал заданий функцією, що викликає читання подій. Цей сигнал повинен бути викликом функції на основі події. Порт доступний, якщо встановлено **Read from memory upon.**
in	Сигнал для повторного відтворення та / або затримки. Цей сигнал повинен бути сигналом на основі події.

Таблиця 10. Вихідні порти блоку Signal Latch

Позначення	Опис	Час оновлення	Початкове значення
st	0 або 1, залежно від операції, яку виконує блок: читання чи запис.	При читанні або записі події	0

mem	Значення внутрішньої пам'яті блоку, коли відбувається запис.	При читанні події	Значення парамет-ра **Initial memory value**
out	Значення внутрішньої пам'яті блоку при читанні.	При записі події	

Порядок оновлення всії вихідних портів дорівнює 1. Оновлення виконується в довільній послідовності відносно один одного. Початкове значення набуває чинності з початку імітації до першого оновлення блоку.

Діалогове вікно блоку містить три вкладки (рис. 41):

- **Write** – запис;
- **Read** – читання;
- **Status** – статус.

Структура та парметри першої вкладки **Write** та другої **– Read** подібні і містять наступні поля:

- **Initial memory value** (початкове значення, занесене у пам'ять) – значення у внутрішній пам'яті блоку перед початком першого запису;

- **Write to memory upon** (запис у пам'ять) – тип виклику події або функції на основі сигналу, що викликає подія запису. Поле може приймати значення наведені у таблиці вхідних портів;

- **Type of change in signal value** (тип зміни значення сигналу) – тип тригера, який визначає напрям зміни сигналу: за зростанням, спаданням чи запис події. Поле доступне лише коли встановлено **Write to memory** для тригеру з порту **wtr** або **wvc.**

- **Resolve simultaneous signal updates according to event priority** (вирішити одночасне оновлення сигналів відповідно до пріоритету події) – контролює послідовність події запису, відносно інших одночасних подій у симуляції. При виборі цього параметру програма виконає подію запису негайно після її появи на основі сигналу. Також у діалоговому вікні з'явиться додаткове поле **Event priority for reading from me-mory** (пріоритет події для запису в пам'ять), яке приймає цілочисельне значення. Якщо на вкладках **Write** та **Read** вибрати **Resolve simultaneous signal updates according to event priority,** то програма ігнорує пріоритет події для читання.

Параметри вкладки **Status** дозволяють контролювати стан блоку та пам’яті (рис. 41). На вкладці розташовані наступні поля:

- **Report state of the block** (повідомляти стан блоку) – дозволяє використовувати вихідний порт сигналу зі значком st;
- **Report memory value upon write event** (повідомляти значення пам'яті під час написання події) – дозволяє використовувати вихідний порт сигналу з позначкою mem;
- **Report memory value upon read event** (повідомляти про значення пам'яті після прочитання події) – дозволяє використовувати вихідний порт сигналу з позначкою **out.**

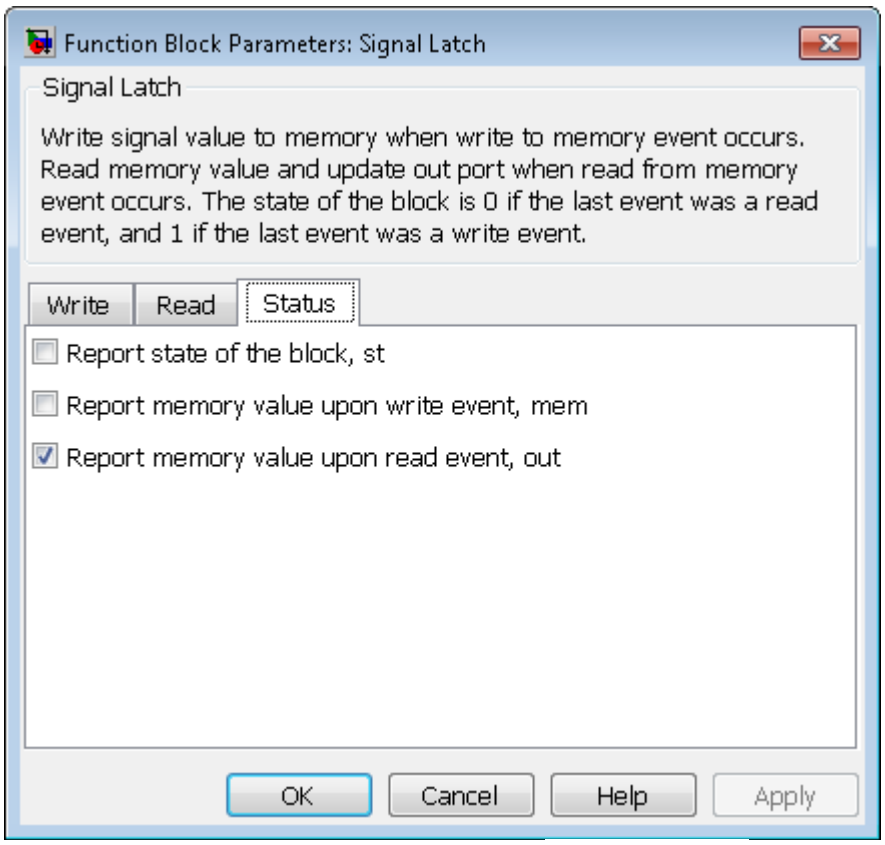
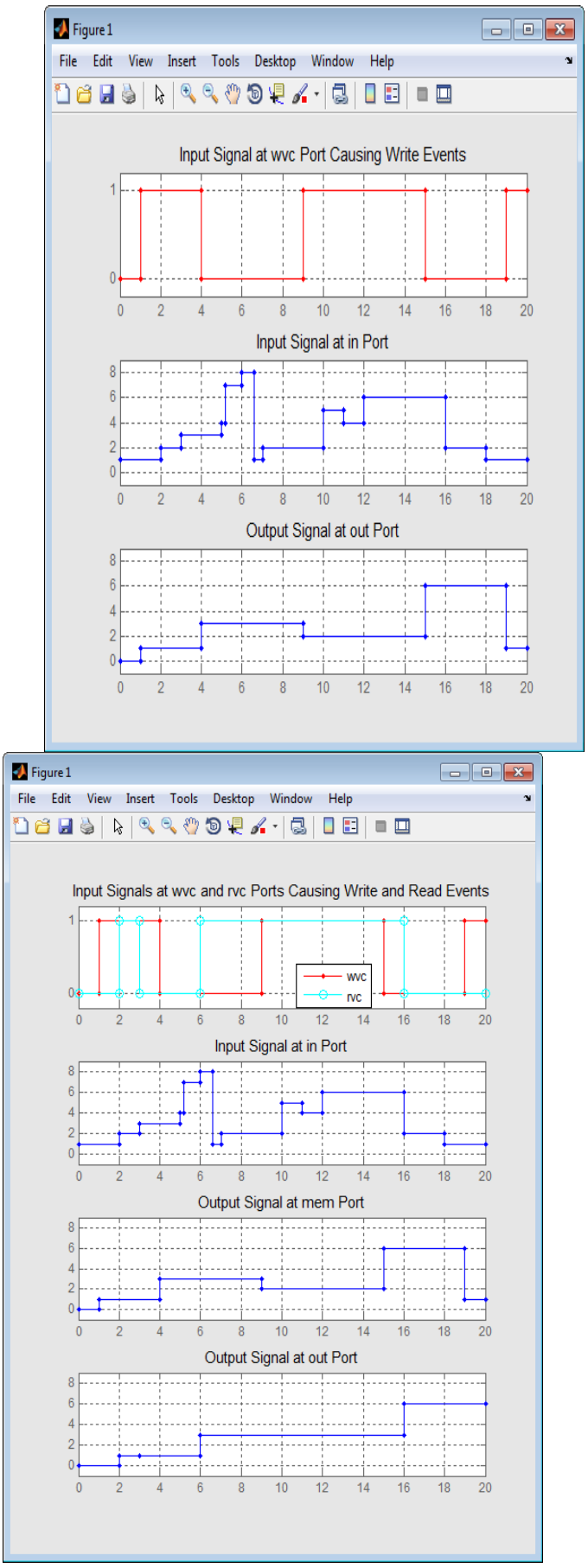


Рис. 41. Закладка Status блоку Signal Latch

Приклади.

1. Читання з пам'яті після кожного запису. У наведеному нижче графіку вихідний сигнал відображає значення вхідного сигналу при кожному зростанні або падінні сигналу **wvc**. Між послідовними записами події вихідний сигнал підтримує значення останньої події запису. Перед початком першого запису, вихідний сигнал становить 0.



а) б)

Рис. 42. Графік до прикладів: а) прикдад 1; б) приклад 2

2. Незалежні події для читання та запису. У наведених нижче графіках сигнал **mem** відображає значення вхідного сигналу для кожного значення, що зростає чи падає, сигналу **wvc**, тоді як значення вихідного сигналу відображають значення сигналу **mem** при кожному зростанні або падінні сигналу **rvc**.

1.6. Ворота (Gates)

Блоки цієї бібліотеки дозволяють управляти потоками в залежності від заданих умов і складається з двох блоків (рис. 43):

-Enabled Gate – дозволяючі ворота;

- Release Gate – пропускаючі ворота.

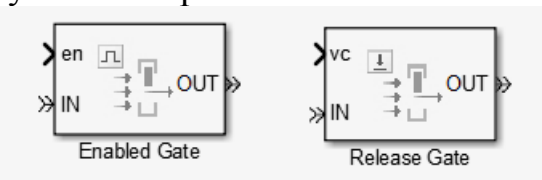


Рис. 43. Блоки бібліотеки Gates

1.6.1. Enabled Gate (Дозволяючі ворота (вентиль))

Блок дозволяє проходження замовлення, коли вхідний сигнал має позитивне значення en і закриває потік, якщо значення сигналу нуль або негативне згідно до встановленого приорітету. Параметр en має числове значення подвійної точності. Дозволяючі ворота дають можливість проходження замовленням одразу до наступного блоку або, коли вони закриті – блокують проходження.

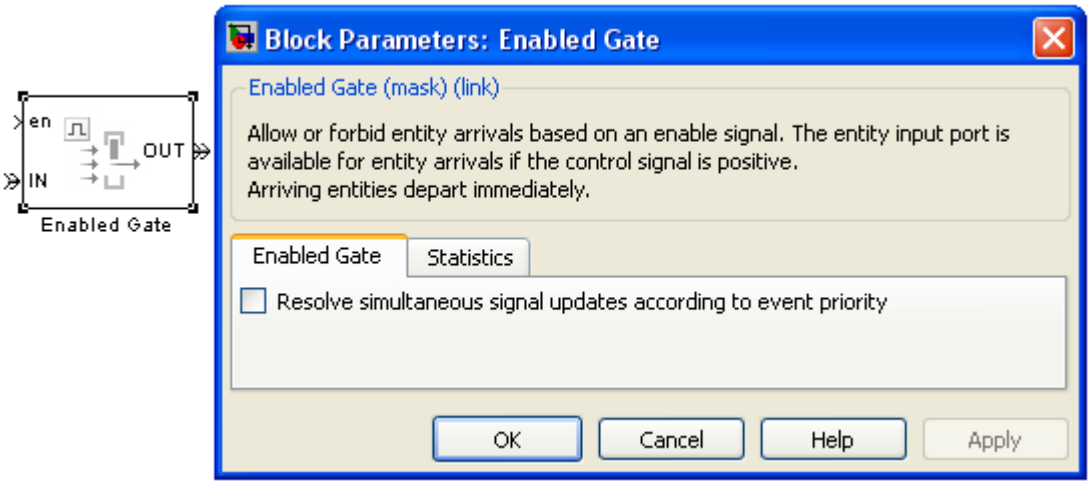


Рис. 44. Блок Enabled Gate

Значення параметра *en* може лишатись позитивним протягом інтервалу часу довільної довжини, а отже потік може бути відкритим на протязі цього часу.

Вхідний порт IN – порт для надходження замовлень.

Вихідний порт OUT – порт для виходу потоку замовлень.

Діалогове вікно блоку має дві закладки:

- **Enabled Gate** – закладка для встановлення режиму роботи. Якщо опція **Resolve simultaneous signal updates according to event priority** є вибраною, то відкриття та закриття потоків узгоджується відносно інших подій згідно до встановленого приорітету і стає доступним значення **Event priority** – приорітет події.

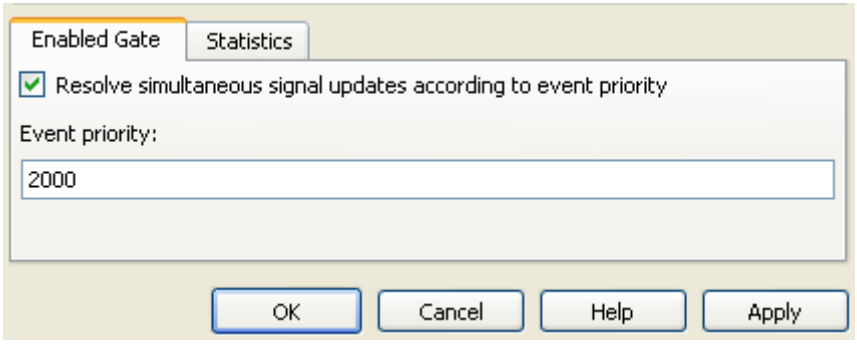


Рис. 45. Блок Enabled Gate із встановленням приорітету

В протилежному випадку події будуть мати на календарі подій приорітет SYS1.

- **Statistics** – закладка статистики дозволяє отримувати інформацію про кількість замовлень, що пройшли через блок з початку процесу моделювання. Для отримання інформації порт #d необхідно встановити у режим *On.*

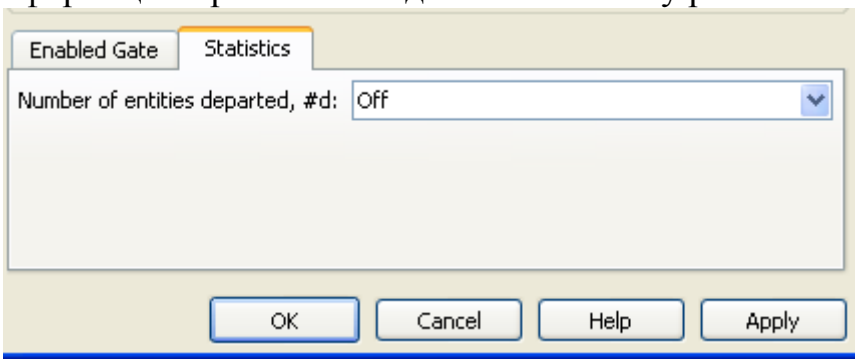


Рис. 46. Закладка статистики блоку Enabled Gate

Приклад. Управління двома серверами.

Нехай кожне замовлення послідовно обробляється двома серверами, а загальний час обслуговування визначається як сума часу обслуговування кожним сервером окремо. Якщо сервери з'єднані послідовно, то може бути така ситуація, коли перший сервер почав обробку нового замовлення, а другий – ще не завершив обробку попереднього. Для уникнення такої ситуації, а саме, щоб обробка наступного замовлення починалась лише тоді, коли завершиться обробка попереднього обома серверами, необхідно встановити блок **Enabled Gate**, який би відкривав потік з початку процесу моделювання і закривав, коли замовлення надходить до першого серверу до моменту часу завершення його обробки другим сервером. Після завершення обробки замовлення обома серверами блок воріт знову відкриває потік. Модель такої системи наведена на рис. 47.

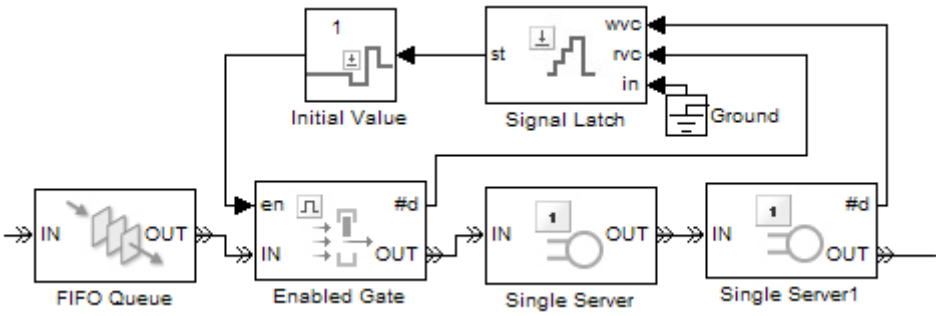


Рис. 47. Модель системи управління двома серверами

Для встановлення значення параметра *en* передбачений блок-засув Signal Latch (бібліотека **Signal Management**), вихідний сигнал якого *st* набуває значення 0, коли замовлення проходить на обслуговування (вхідне значення – *rvc*) і значення – 1, коли замовлення покидає блок обслуговування (другий сервер – вхідне значення *wvc*).

1.6.2. Release Gate (пропускаючі ворота)

Блок відкриває ворота на основі події-сигналу або виклику функції (рис. 48). Замовлення для проходження через канал повинні бути вже в режимі очікування, перед подією відкриття каналу. Відкриття каналу дозволяє одному замовленню відразу перейти до наступного блоку (перехід виконується миттєво без витрат часу). Якщо в стані очікування замовлень немає, то канал закривається без виконання обробки.

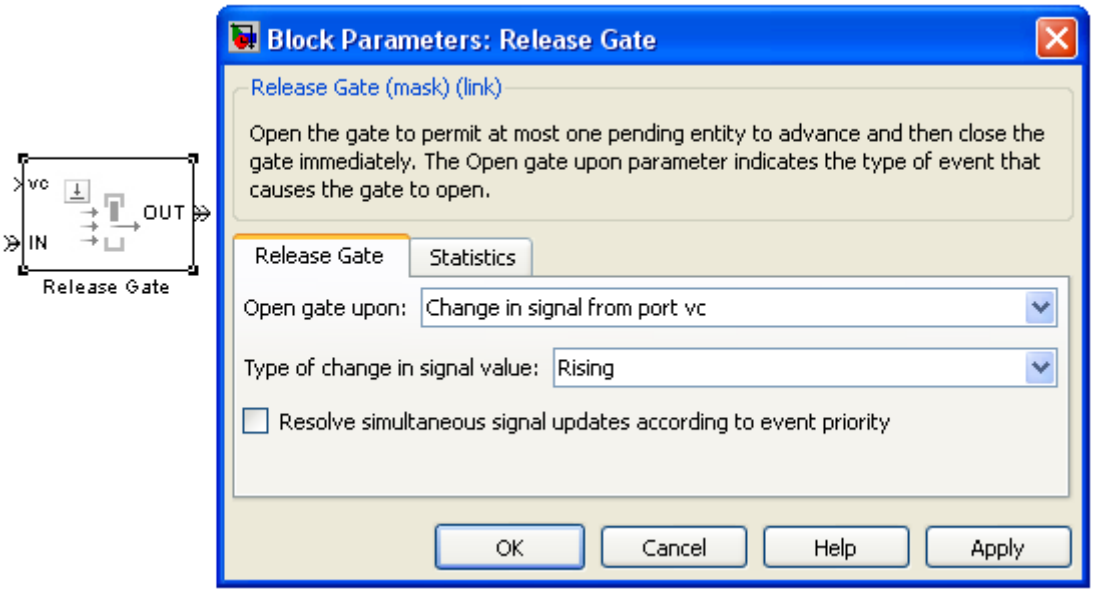


Рис. 48. Блок Release Gate

Блок має два вхідних та один вихідний сигнали.

Вхідний сигнал IN та вихідний сигнал OUT мають такі ж значення, як і для блоку **Enabled Gate**. Для визначення другого вхідного сигналу та настроювання блоку застосовується діалогове вікно (рис. 48), яке подібне до

діалогового вікна блоку **Enabled Gate**. Відмінність полягає у настроюванні вхідного сигналу, для чого передбачено два поля:

- **Open gate upon** – визначення типу події для відкриття потоку:
  - tr – тригер сигнал, який вказує коли відкрити канал;
  - vc – керуючий сигнал, чисельна зміна якого визначає, коли відкрити канал;
  - fcn – функція виклику сигналу, що вказує, коли відкрити канал.
- **Type of change in signal value** – визначення типу управляючого сигналу:
  - Rising – за зростанням сигналу;
  - Falling – за спаданням сигналу;
  - Either – інше.

Приклад із синхронізацією початку відліку часу за годиником.

У наведеному нижче прикладі, блок Release Gate управляється вхідним сигналом від блоку генератора імпульсів (Pulse Generator), який гарантує, що сутності надходять з фіксованим часовим кроком в 1 секунду, хоча сутності перебувають в асинхронному режимі. У цьому прикладі для блоку Реалізація каналу (Realize Gate) значення поля Open gate upon встановлене у значення Change in signal from port vc and Type of change in signal value set to Rising (тип зміни значення сигналу заданий в Rising), в той час як блок генератора імпульсів (Pulse Generator) має встановлений період в 1.

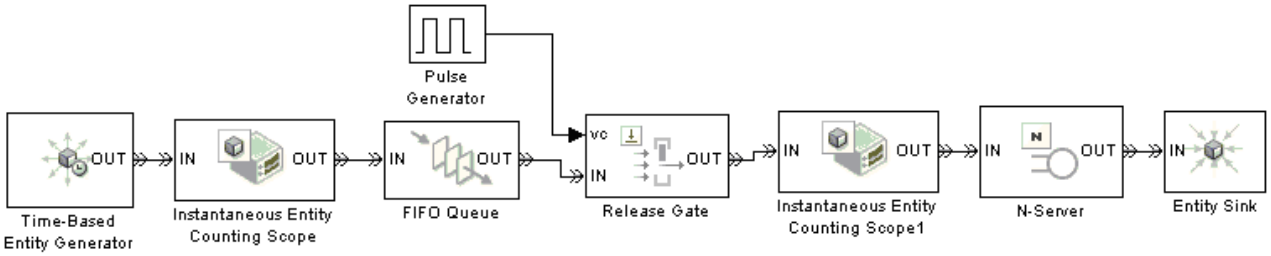


Рис. 49. Модель синхронізації початку видліку часу за годинником

Графіки нижче показують, що Час Генерування Сутностей може бути не цілим числом, проте Час Початку Сервісу завжди цілі числа.

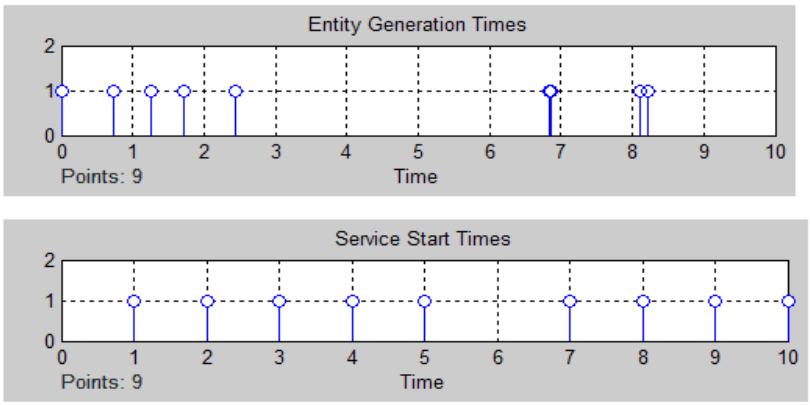


Рис. 50. Графіки часу генерування сутностей та початку їх обслуговування СМО

Приклад відкриття каналу під час виходу сутностей.

У наведеній нижче моделі, дві пари черг сервера працюють паралельно і сутність виходить від верхньої черги тільки у відповідь на вихід від нижньої черги. Зокрема відхилення від нижнього блоку черги викликає блок Entity Departure Event to Function-Call Event – подію, яка закінчує виклик функції, яка в свою чергу відкриває канал. Блок Release Gate в цій моделі має значення поля Open gate upon – Function call from port fcn.

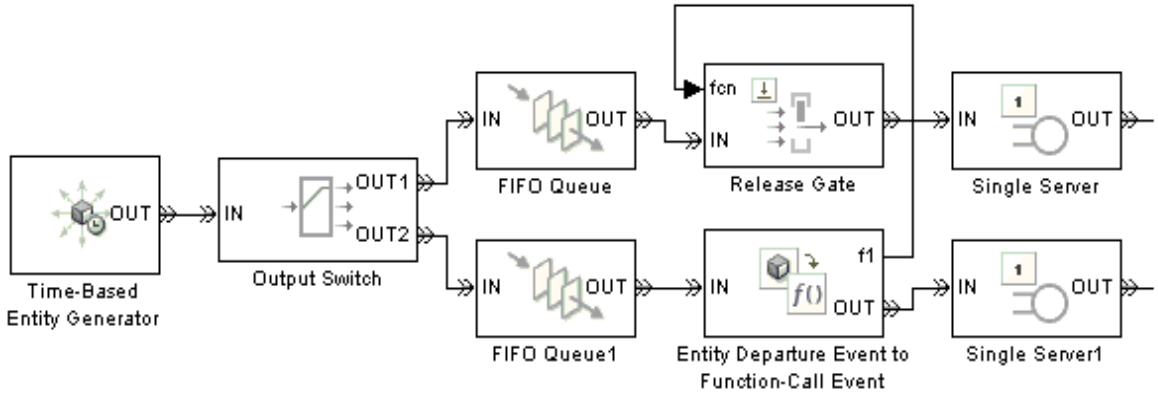


Рис. 51. Модель з двома парами черг сервера, що працюють паралельно

Якщо верхня черга в моделі порожня, коли нижня черга має дані на вихід, то канал відкриваються, але жодна сутність не надходить.

При налаштуванні каналу на відкриття при виході сутностей, переконайтеся, що в ваших намірах є логіка. Наприклад, дивлячись на модель, яка показана вище, можна подумати, що пари сутностей проходять через сервер-чергу під час симуляції. Тим не менш, якщо вихідний вимикач (Output Switch) блоку виконаний з можливістю вибору першого вихідного порту не заблокованої сутності і якщо верх черги має велику ємність по відношенню до числа сутностей, отриманих протягом терміну симуляції, то може виявитись, що всі сутності переходять до початку черги, а не до кінця. У результаті немає сутності, яка відходить від кінця черги і канал ніколи не відкривається для виходу сутностей з початку черги. На відміну від цього, якщо вихідний вимикач (Output Switch) встановлений з урахуванням можливості його вибору випадковим чином між двома вихідними портами сутностей, то цілком ймовірно, що деякі сутності дійдуть до серверів, як очікувалося.

1.7. Сервери (Servers)

Бібліотека містить три блоки, які симулюють сервіси СМО:

- **Single Server** – одиничний сервіс;
- **N-Server** – N-сервісів, де число N визначається користувачем;
- **Infinite Server** – нескінченна кількість сервісів.

Блоки мають подібну структуру, тому детально розглянемо блок Single Server, а для інших блоків зазначимо лише особливості їх застосування та налаштування.

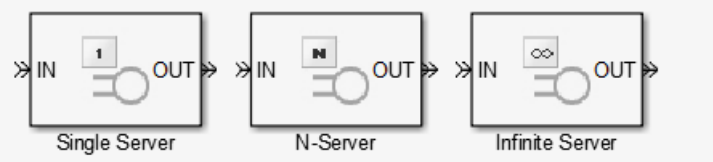


Рис. 52. Блоки серверів

1.7.1. Single Server

Single Server – блок обслуговує одночасно одну сутність за деякий інтервал часу і далі намагається пропустити її через вихідний порт OUT. Якщо порт OUT заблокований, то сутність залишається в середині блоку до тих пір, поки вихідний порт не разблокується.

Час обслуговування (Service Time) визначається через параметри, атрибути або сигнали в залежності від значення параметра Service Time From, яке знаходиться на головній закладці і може приймати наступні значення:

- **Dialog** – параметри задаються на основі діалогу у полі Service Time From. Поле Service Time From визначає час обслуговування у секундах;
- **Signal port t** – параметри отримуються із сигнального порту t, причому сигнал повинен визначати подію (event-based). Якщо виставлено це значення, то до блоку додається додатковий вхід – сигнальний порт t;
- **Attribute** – параметри задаються як атрибут, задний у полі Attribute name, яке з'являється при виборі значення поля Attribute.

Час обслуговування (Service Time) є цілим числом і задається у секундах.

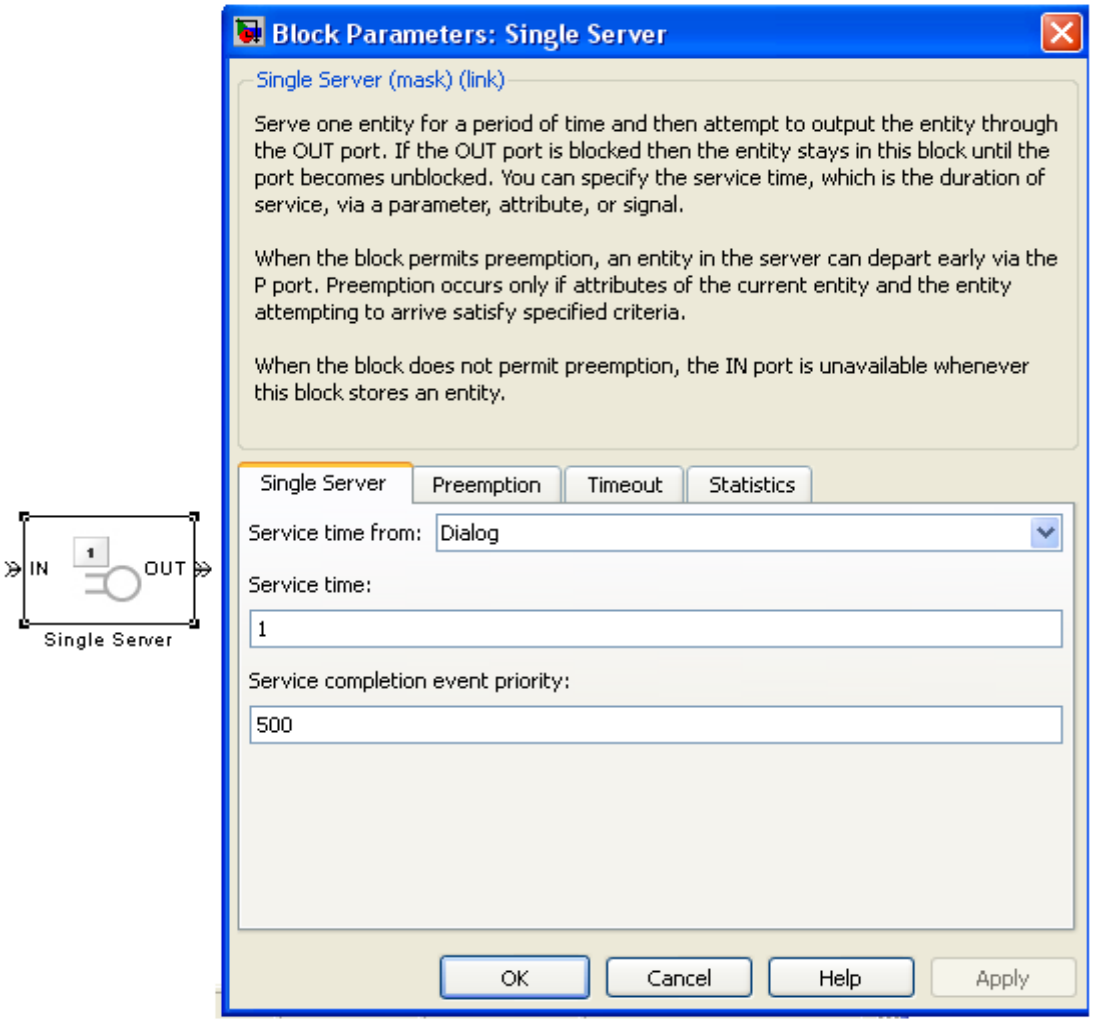


Рис. 53. Блок Single Server

Поле Service completion event priority (завершення обслуговування пріоритетних подій) визначає пріоритетну подію, обслуговування якої завершується раніше від інших подій в процесі симуляції.

Наступна закладка **Preemption** (попередження) містить встановлення опції **Permit preemption based on attribute** (дозвіл на випередження в залежності від атрибута). Вибір опції дозволяє надходження сутності з вищим пріоритетом замість сутності з нижчим і на вкладці статитистики параметр **Average wait** (очікування надходження) встановлюється у положення Off (стає недоступним). При виборі опції з'являються наступні поля:

- **Sorting attribute name** – задає ім'я приорітетного атрибуту;
- **Sorting direction** – визначає порядок сортування (за зростанням або спаданням значень приорітету).

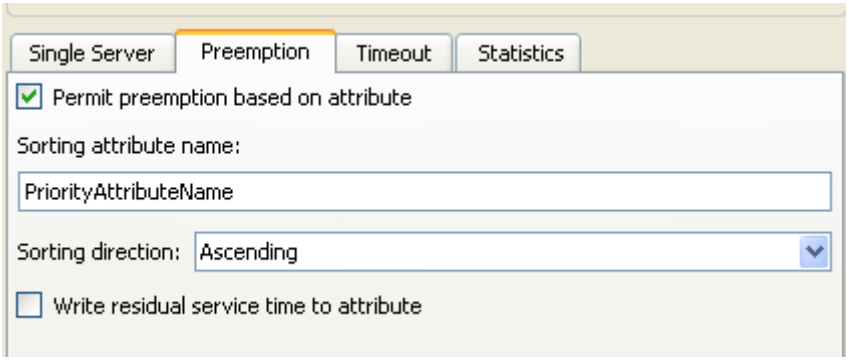


Рис. 54. Закладка Preemption блоку Single Server

Також доступною стає опція **Write residual service time to attribute** (записати додатковий час обслуговування), вибір якої у випадку появи сутності з вищим пріоритетом встановлює атрибут поточної сутності, що обслуговується і має нижчий пріоритет, у стан перерваного і записує залишковий час. Значення залишкового часу задається у полі Residual service time attribute name.

Наступна закладка **Timeout** використовується для підключення порту для сутностей з перерваним часом обслуговування, якщо такі передбачені.

Для збору статистики використовується закладка Statistics (рис. 54) з наступними полями:

- **Number of entities departed**, **#d** (кількість сутностей, що вийшли з блоку) – контролює наявність і поведінку вихідного сигналу порту #**d**;
- **Number of entities in block**, **#n (**кількість сутностей, що обслуговуються**) –** контролює наявність і поведінку вихідного сигналу порту #**n**;
- **Number of entities preempted**, **#p** (кількість сутностей з перерваним обслуговуванням) – контролює наявність і поведінку вихідного сигналу порту з написом #p. Це поле доступне, тільки якщо вибрано опцію **Permit preemption based on attribute** (дозвіл переривань на основі параметра атрибута) на вкладці **Preemption** (попередження);
- **Status of pending entity departure**, **pe** – контролює наявність і поведінку вихідного сигналу порту **pe**;
- **Average wait, w** (середній час очікування) – контролює наявність і поведінку вихідного сигналу порту з міткою w. Це поле доступне, тільки якщо очистити опцію **Permit preemption based on attribute** (дозвіл переривань на основі параметра атрибута) на вкладці **Preemptio**n (попередження);
- **Utilization, util** (утилізація або доля часу моделювання, яка використана на зберігання сутності) – контролює наявність і поведінку вихідного сигналу порту **util**;
- **Number of entities timed out, #to** (кількість сутностей, час обслуговування яких сплив) – контролює наявність і поведінку вихідного сигналу порту **#to**.

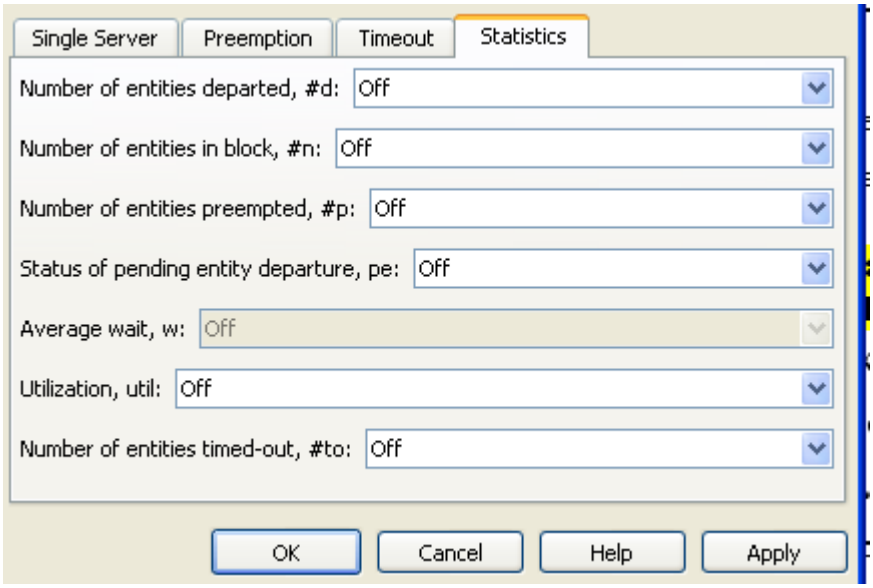


Рис. 55. Вигляд закладки Statistics блоку Single Server

Щоб зробити доступним порт для отримання відповідної статистики, його значення необхідно встановити у положення On.

Приклади.

1. Послідовне підключення одиничних сервісів з чергою.

Дві пари черга-сервіс представляють собою послідовні операції. Наприклад, частини на конвеєрі обробляються послідовно двома машинами.

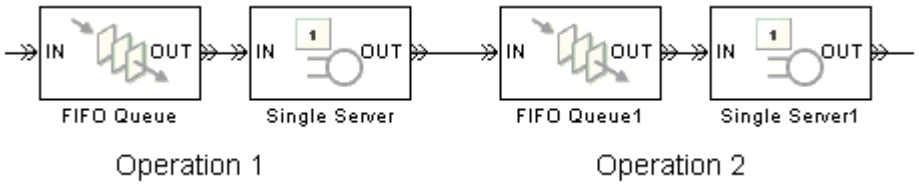


Рис. 56. Послідовне підключення пар черга-сервіс

Можна моделювати ситуацію по-іншому, як пару серверів без черги між ними, відсутність черги означає, що якщо перший сервер завершить обслуговування сутності до того як другий сервер буде доступний, то сутність повинна залишитися на першому сервері і перший сервер не може прийняти новий об'єкт для обробки, поки другий сервер не стане доступним.

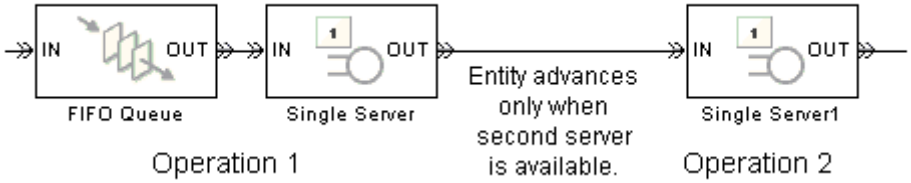


Рис. 57. Послідовне підключенняодиничних сервісів без черги між ними

2. Паралельні пари черга-сервер як альтернативи. Дві пари черга-сервер, паралельно з'єднані представляють собою альтернативні операції. Наприклад,

автобуси чекають завантаження на платформі, куди пасажири підходять з одного входу.

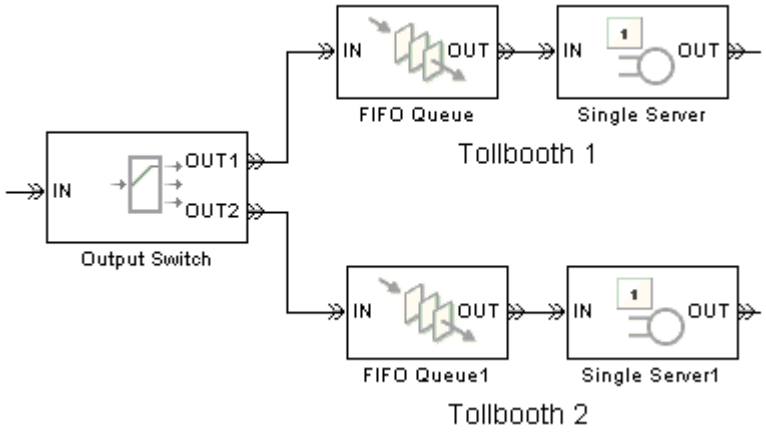


Рис. 58. Паралельні пари черга-сервер як альтернативи.

3. Паралельні пари черга-сервер у багатоадресному повідомленні. Дві пари черги-сервера, паралельно з'єднані, копія кожного об'єкта надходить до обох підсистем. Така система може представляти ситуацію багатоадресної передачі, наприклад, надсилання повідомлення кільком одержувачам.

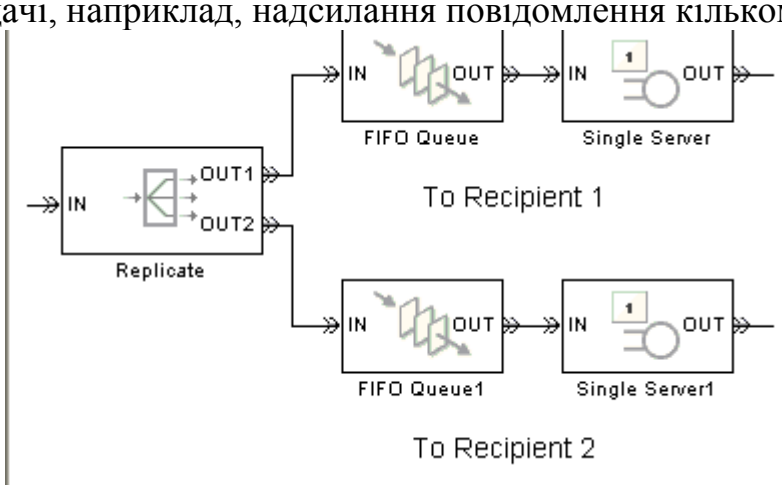


Рис. 59. Паралельні пари черга-сервер у багатоадресному повідомленні

4. Контроль двох серверів. Припустимо, що кожний об'єкт проходить два процеси, по одному за раз, і що перший процес не починається, якщо другий процес все ще виконується для попереднього об'єкта. Для цього прикладу краще моделювати два процеси, використовуючи два блоки Single Server, а не один, час служби якого є сумою двох окремих періодів обробки (рис. 60).

Якщо підключити чергу, сервер та інший сервер послідовно, то перший сервер може почати обслуговувати новий об’єкт, тоді як другий сервер все ще обслуговує попередній об'єкт. Це не відповідає поставленій меті. Модель потребує воріт, щоб запобігти першому серверу приймати об'єкт занадто рано, тобто, коли другий сервер оброблює попередній об'єкт.

Один із способів реалізації цього полягає в тому, щоб перед першим блоком Single Server розташувати блок Enabled Gate, який налаштований так,

що ворота закриваються, коли об'єкт знаходиться на одному з серверів. Ворота виконують наступні процедури:

• відкриті від початку моделювання до надходження першої сутності;

• закриваються кожного разу, поки об'єкт не досягне першого сервера, тобто коли вихідний сигнал блоку gateway збільшує кількість

• повторно відкриваються кожного разу, коли цей об'єкт виходить з другого сервера, тобто при збільшенні вихідного сигналу #d другого серверного блоку.

Таке розташування показано нижче.

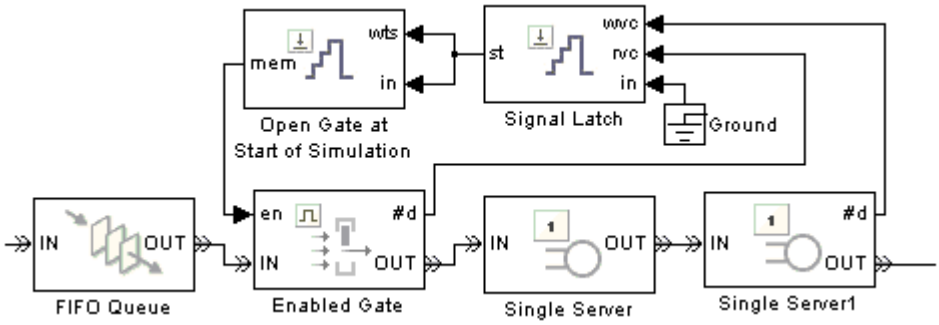


Рис. 60. Контроль двох серверів

Значення вихідного сигналу блоку, приймає значення 0, коли вхідний сигнал **rvc** блоку збільшується і стає рівним 1, коли вхідний сигнал **wvc** збільшується. Тобто, **st**-сигнал стає 0, коли об'єкт відходить від воріт і стає 1, коли об'єкт відходить від другого сервера. Блок з позначкою "Відкриті ворота на початку моделювання" – ще один блок засувки сигналу; його метою є зміна сигналу **st** лише шляхом визначення початкового стану 1. Таким чином, об'єкт, який очолює чергу, переходить на перший блок одиничного сервера тоді і тільки тоді, коли обидва сервери порожні.

1.7.2. N-Server

N-Server – блок забезпечує незалежну обробку N замовлень, де кожне замовлення виконується протягом певного періоду часу, а потім намагається вивести його через порт OUT. Якщо порт OUT заблоковано, то замовлення залишається в цьому блоці, доки порт не буде розблоковано. Якщо замовлення повинно виконуватись згідно розкладу, то воно може надійти наперед через додатковий порт TO.

N-сервер нагадує набір N окремих серверів, підключених паралельно, за яким слідує комбінатор шляхів, який визначає послідовність замовлень до розблокованого шляху, де обробка замовлення завершилась і воно вийшло з блоку.

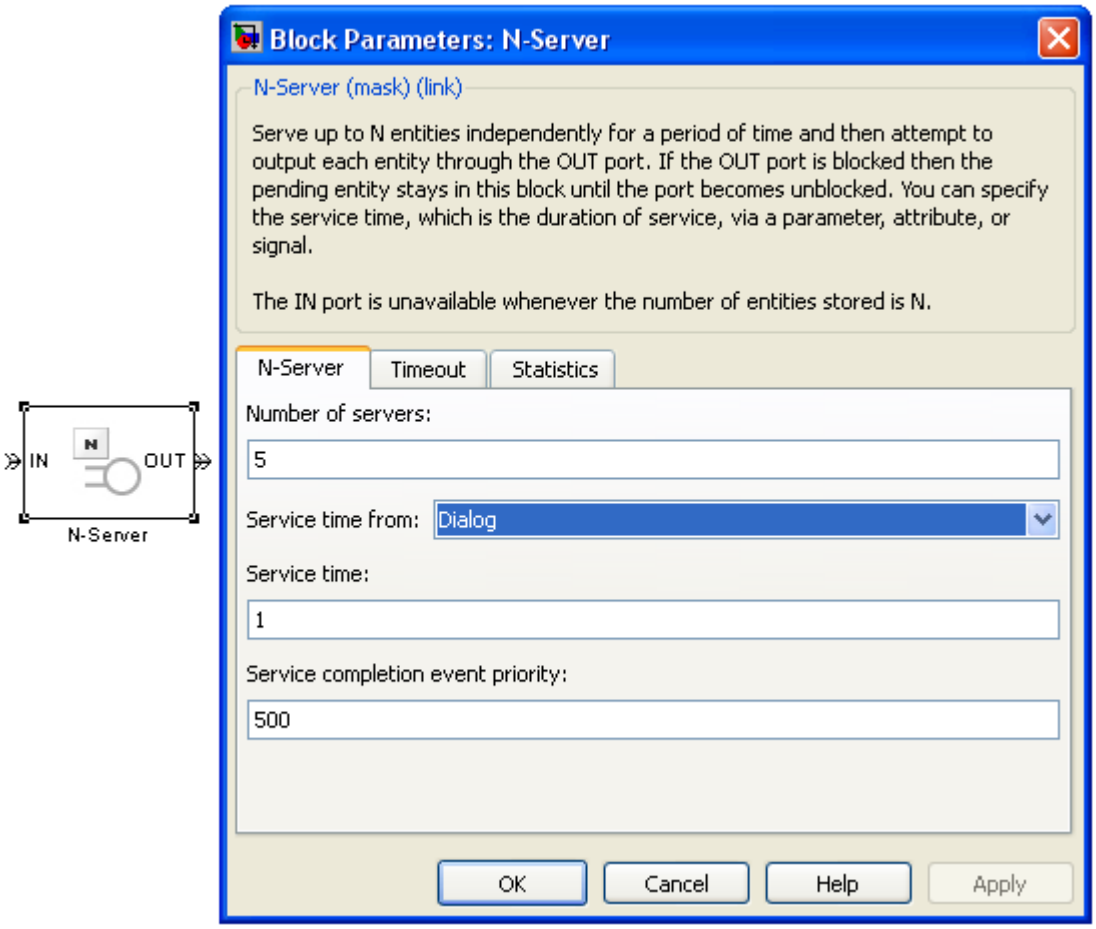


Рис. 61. Блок N-сервер та його діалогове вікно

На головній вкладці діалогового вікна визначається кількість сервісів через параметр Number of servers, який може приймати значення з множини натуральних чисел.

Всі інші поля подібні до полів блоку Single Server, описаних вище.

Для збору статистики використовується закладка Statistics (рис.62) з наступними полями:

- **Number of entities departed, #d** (кількість сутностей, що вийшли з блоку) – контролює наявність і поведінку вихідного сигналу порту #**d;**
- **Number of entities in block, #n (**кількість сутностей, що обслуговуються**) –** контролює наявність і поведінку вихідного сигналу порту #**n;**
- **Pending entity present in block** (замовлення в стані очікування у блоці) – дозволяє використовувати вихідний порт сигналу #**pe.**
- **Number of pending entities** (кількість замовлень в стані очікування) – дозволяє використовувати вихідний порт сигналу із позначкою **#pe.**

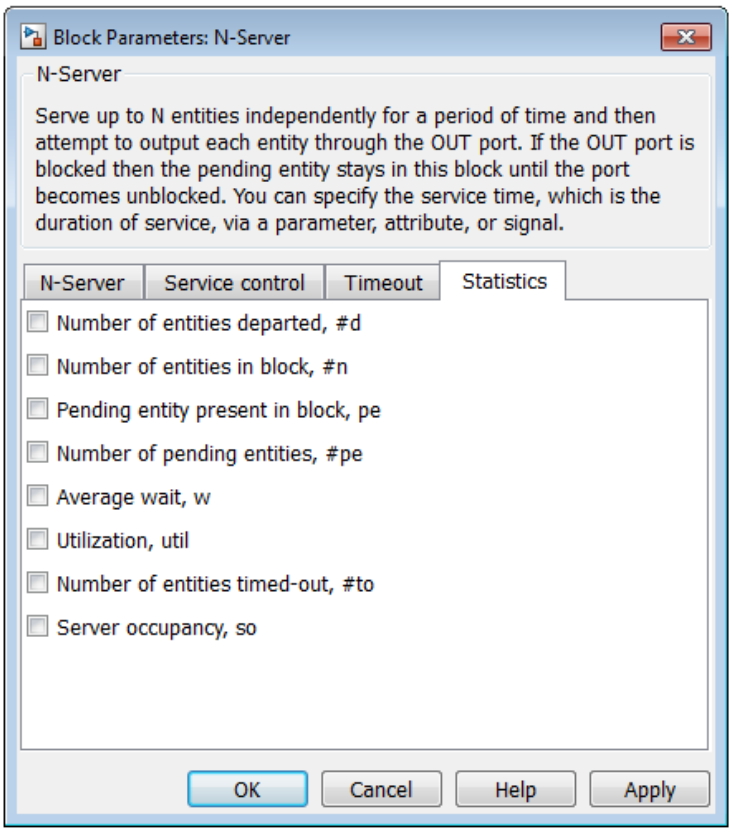


Рис. 62. Вкладка статистики блоку N-сервер

- **Average wait, w** (середній час очікування) – контролює наявність і поведінку вихідного сигналу порту з міткою w. Це поле доступне, тільки якщо очистити опцію **Permit preemption based on attribute** (дозвіл переривань на основі параметра атрибута) на вкладці **Preemption** (попередження);
- **Utilization** (утилізація або доля часу моделювання, яка використана на зберігання сутності) – контролює наявність і поведінку вихідного сигналу порту **util**;
- **Number of entities timed out** (кількість замовлень з вичерпаним часом обслуговування) – дозволяє використовувати вихідний порт сигналу із позначкою **#to**;
- **Server occupancy, so** (максимальна кількість задіяних серверів) – дозволяє використовувати вихідний порт сигналу із назвою **so.**

Приклад. Система M/M/5 з нескінченною чергою. Наведена нижче модель показує систему без відмов та п'ятьох ідентичних серверів.

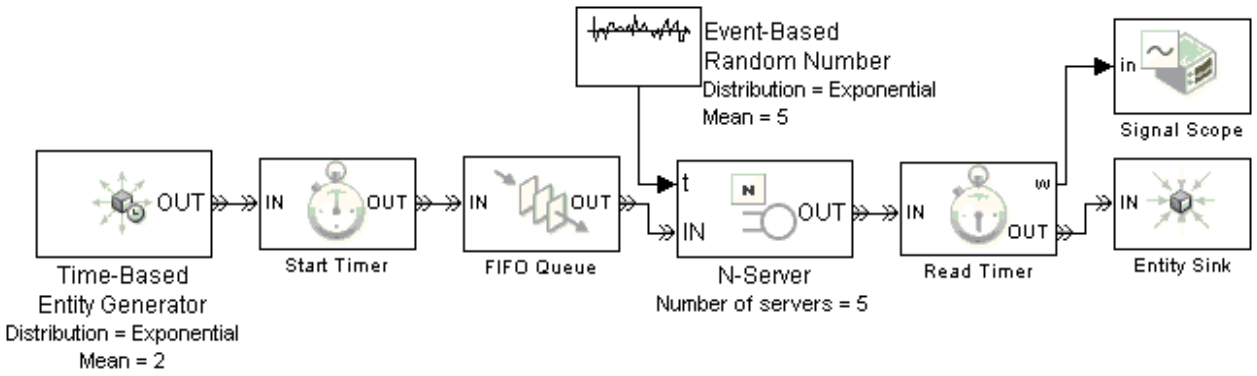


Рис. 63. Система M/M/5 без відмов

Наведений нижче графік показує час очікування в системі (рис. 63)

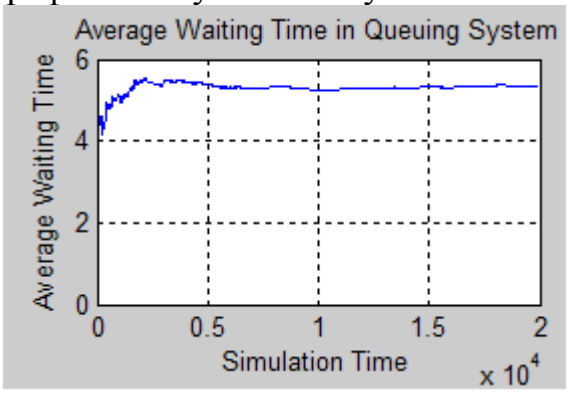


Рис. 64. Графік розподілу часу очікування в системі

Можна порівняти емпіричні значення, показані на графіку, з теоретичним значенням $W_{сист}$ середнього системного часу для моделі M/M/m з вхідною інтенсивністю λ = 1/2 та швидкістю обслуговування μ = 1/5:

$$\rho = \frac{\lambda}{m \cdot \mu} = \frac{(1/2)}{5 \cdot (1/5)} = \frac{1}{2},\ p_0 = \left(1 + \sum_{i=1}^{m-1} \frac{(m \cdot \rho)^i}{i!} + \frac{(m \cdot \rho)^m}{m!} \cdot \frac{1`}{1-\rho}\right)^{-1} \approx 0{,}080,$$

$$W_{сист} = \frac{1}{\mu} + \frac{1}{\mu} \cdot \frac{(m \cdot \rho)^m}{m!} \cdot \frac{p_0}{m(1-\rho)^2} \approx 5{,}26.$$

Середній час перебування в системі становить приблизно 5,26 одиниць часу, що відповідає лінії на графіку (рис. 64).

1.7.3. Infinite Server

Infinite Server – блок обслуговує будь-яку кількість замовлень протягом певного періоду, який називається службовим часом, а потім намагається вивести їх через порт OUT. Якщо порт OUT заблоковано, тоді блок зберігає об'єкти до розблокування порту. Якщо блок заплановано на роботу за розкладом на тайм-аут, то замовлення може вийти через додатковий порт TO.

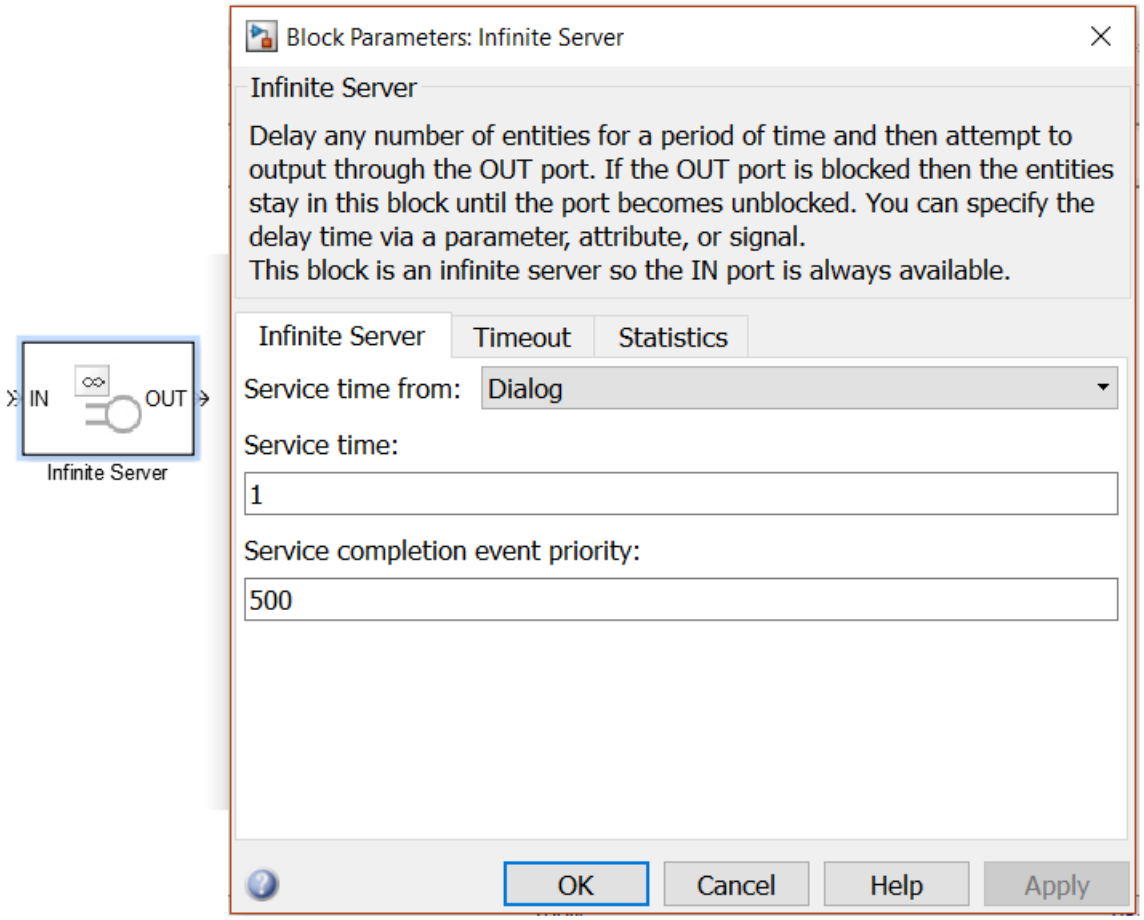


Рис. 65. Блок Infinite Server та його діалогове вікно

Нескінченний сервер подібний до нескінченного набору одиничних серверів, підключених паралельно, а потім до комбінатора шляхів, який розблоковує шлях у послідовності, в якій замовлення завершили свій час обробки, доки одне замовлення не відійде з блоку.

Вхідними даними є час обслуговування, який задається через параметр, атрибут або сигнал. Блок визначає час обслуговування замовлення після його прибуття. Передбачається, що час обслуговування визначається в секундах.

Примітка. Якщо час обслуговування вказаний за допомогою сигналу на основі події, переконайтеся, що його оновлення відбуваються до того, як замовлення прибуде.

Закладки і поля подібні до полів блоку Single Server, описаних вище.

Для збору статистики використовується закладка Statistics (рис. 66).

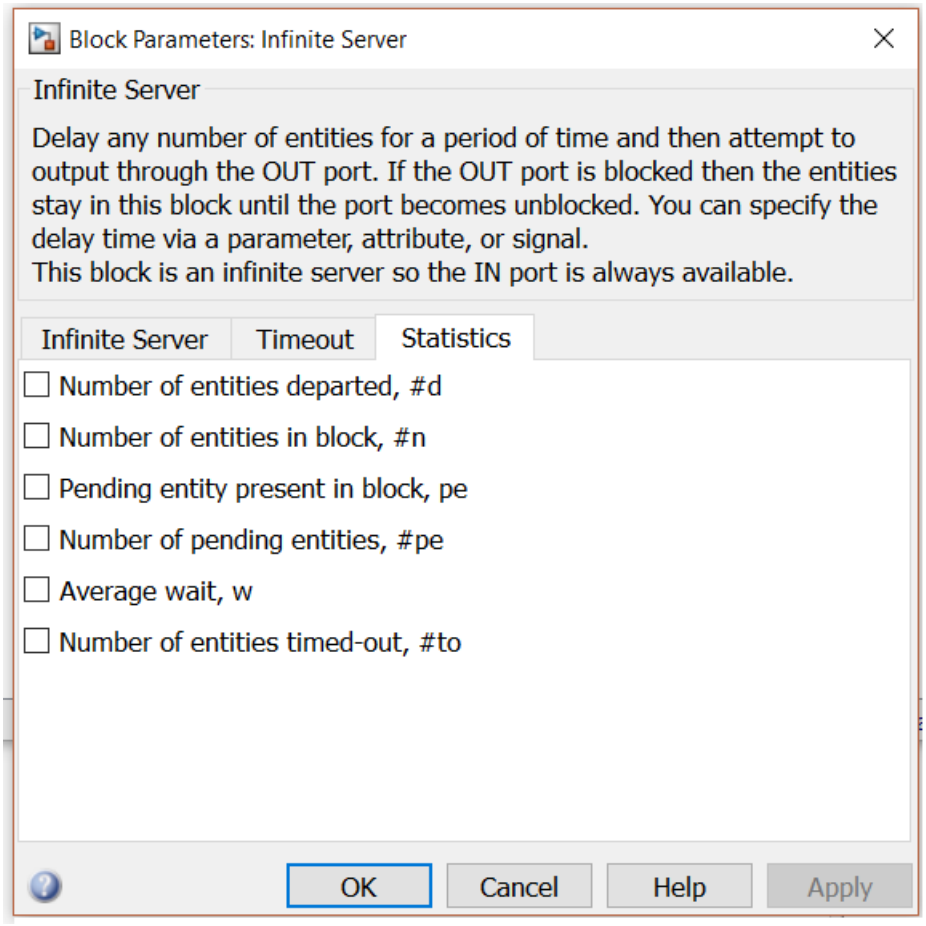


Рис. 66. Вкладка статистики блоку Infinite Server

1.7.4. Блоки відображення результатів моделювання та поглинання сутностей SimEvents Sinks

Бібліотека відображення результатів містить вісім блоків:

- Attribute Scope – область атрибутів подій;
- Discrete Event Signal to Work – дискретний сигнал події для роботи;
- Entity Sink – поглинач сутностей;
- Instantaneous Entity Counting Scope – область підрахунку сутностей;
- Instantaneous Event Counting Scope – область підрахунку подій;
- Signal Scope – область сигналу;
- X-Y Attribute Scope – область X-Y атрибутів;
- X-Y Signal Scope – область X-Y сигналів.

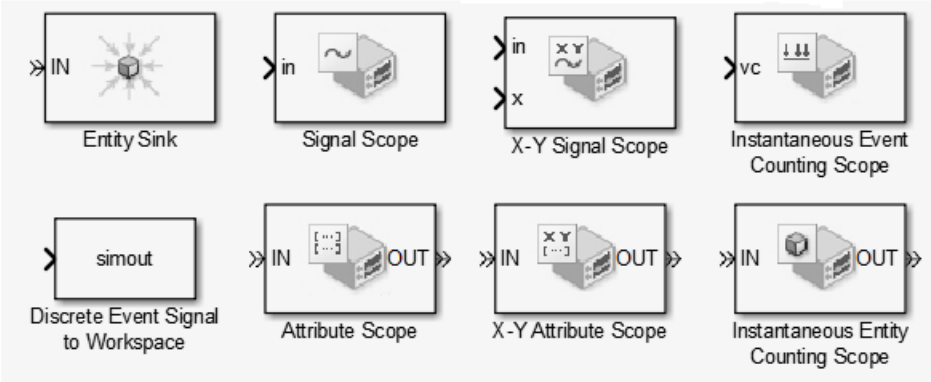


Рис. 67. Блоки бібліотеки SimEvents Sinks

За допомогою блоків Scope можна виводити графіки необхідних характеристик.

1.7.4.1. Attribute Scope

Attribute Scope – блок виводу графіка даних вхідних атрибутів (параметрів)
сутностей.

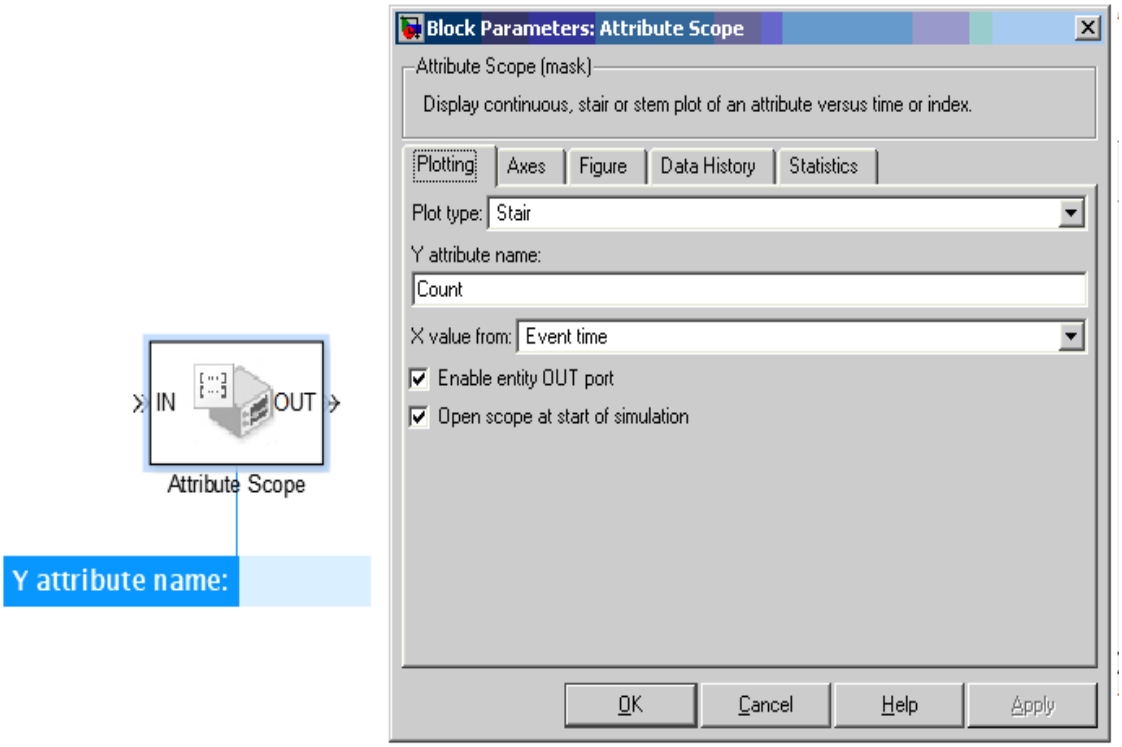


Рис. 68. Блок Attribute Scope та його діалогове вікно

Цей блок формує графік сутностей, що вийшли із системи на основі атрибутів вхідних значень і задаються дійсними скалярними значеннями. Щоб відкрити вікно налаштування, відкрийте випадаюче меню, натиснувши ліву клавішу «миші», і виберіть пункт Block Parameters (параметри блоку).

Головна закладка **Plotting** (Побудова графіку) містить наступні поля:

- **Plot type** (Тип графіку) – дозволяє вибрати неперервний (Continuous) або дискретний (Stair) тип графіку для представлення даних;
- **Y attribute name** – назва атрибуту вздовж осі Y, для визначення за яким саме атрибутом будувється графік;
- **X value from** – визначає джерело даних для осі X (табл..)

Таблиця 8. Графіки вкладки Plotting

Джерело даних X (Source of X Data)	Опис графіку (Description of Plot)
Event time Час події	Графік спеціалізованого атрибуту та час симуляції.
Index Індекс	Графік успішних значень спеціалізованого атрибуту по горизонтальній осі, що показують індекс значень. Значення першого атрибуту сутності має індекс 1, значення другого атрибуту сутності має індекс 2, і так далі. Наприклад, можна використовувати цю опцію, коли багато сутностей можуть надходити на вхід одночасно, щоб допомогти визначити точну послідовність серед одночасних значень атрибутів.

Використання різних джерел даних для горизонтальної осі наведені на рис. 69. Графіки виглядають схожими, крім того, що другий графік має рівномірні горизонтальні відступи, а перший має відступи, побудовані на основі промжків часу між сусідніми точками.

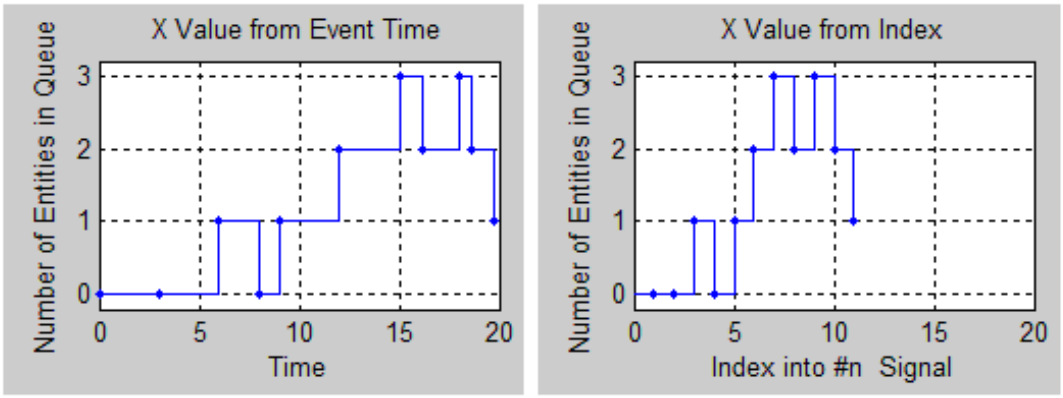


Рис. 69. Приклад побудови графіків за різними джерелами вхідних даних

- **Enable entity OUT port** (Включити вихідний порт для сутності) – вибір опції активізує вихідний порт, через який замовлення можуть виходити з блоку, в протилежному випадку – блок поглинає сутності;
- **Open scope at start of simulation** (включити вікно графіку на початку симуляції) – при виборі цієї опції графічне вікно відкривається з початком запуску моделі, в протилежному випадку - графічне вікно можна відкрити подвійним кліком на значок блоку.

Опис портів блоку поданий у табл.11.

Таблиця 11. Порти блоку Attribute Scope

Позначення	Призначення	Опис
IN	Вхідний порт сутностей	Вхідний порт сутностей, чиї атрибути містять у собі дані для графіку.
OUT	Вихідний порт сутностей	Вихідний порт сутностей. Цей порт можна побачити тільки після вибору команди Enable entity OUT port (Включити вихідний порт сутності)
#a	Вихідний порт сигналів	Номер сутності, що прибула у блок після старту симуляції.

Початкове значення кількості сутностей у блоці, за замовчуванням, дорівнює 0. Діалогове вікно блоку містить декілька закладок для налаштування.

До складу вкладки вісей **Axes** ходять наступні поля (рис. 70):

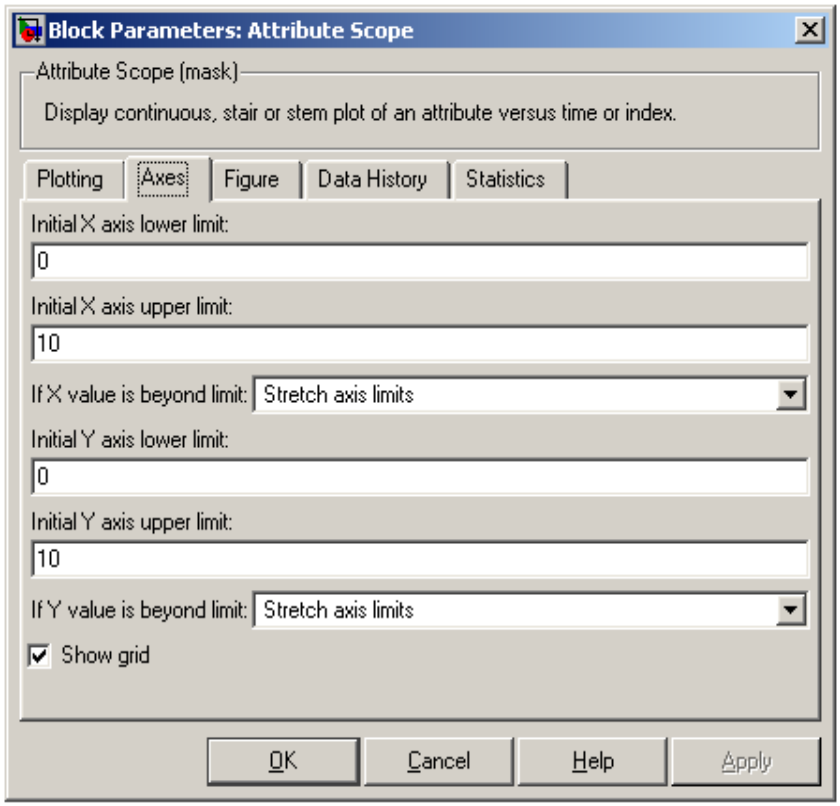


Рис. 70. Закладка налаштування координатних вісей

- **Initial X axis lower limit** (мінімальне значення X), **Initial X axis upper limit** (максимальне значення X) – встановлення інтервалу для даних по осі X. Дані можна автоматично масштабувати в мехаж цього інтервалу;
- **If X value is beyond limit** (Якщо значення X знаходиться за межами інтервалу) – визначає зміну графіка за межами встановленого інтервалу X. Для отримання детальної інформації дивіться **Varying Axis Limits Automatically**.

- **Initial Y axis lower limit** (мінімальне значення Y), **Initial Y axis upper limit (**максимальне значення Y) – визначає межі побудови графіка за віссю Y.
- **If Y value is beyond limit** (Якщо значення Y знаходиться за межами інтервалу) – інтервал може змінюватися від цих початкових налаштувань за допомогою функцій наближення, автоматичного масштабування або якщо значення Y знаходиться поза межами встановленого ліміту.
- **Show grid** (Показати сітку) – вмикає та вимикає сітку на графіку.

Figure Tab (Вкладка фігур) (рис.71)застосовується для створення підписів на графіку:

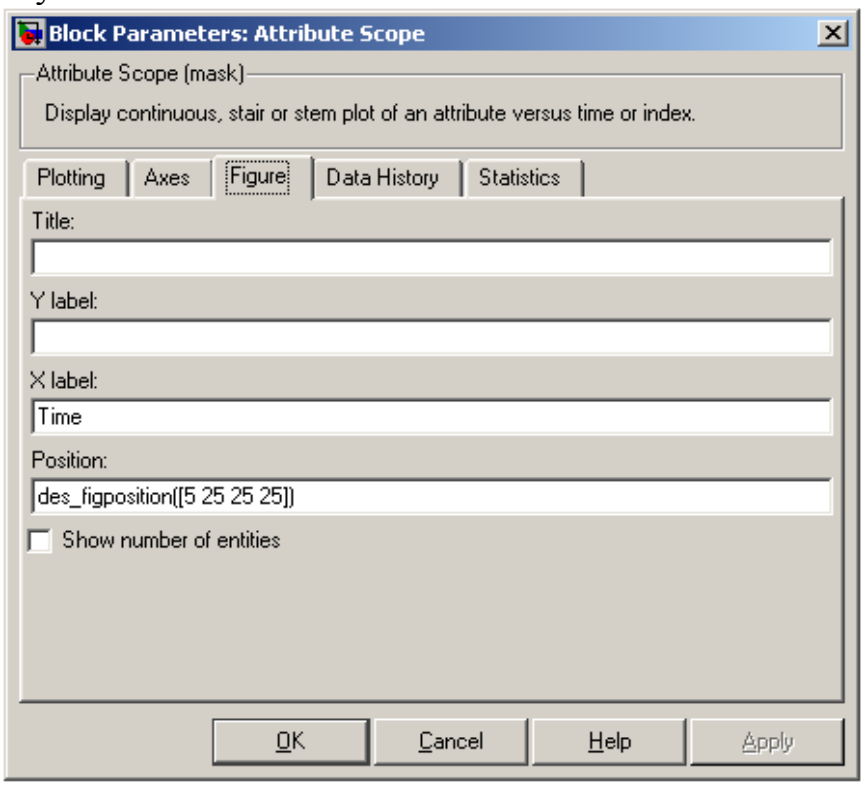


Рис. 71. Закладка налаштування графіка

- **Title** (Назва) – текст, який з'являється в якості назви графіку, над осями;
- **Y Label** (Позначення осі Y), **X Label** (Позначення осі X) – текст, який є підписом для відповідної осі координат;
- **Position** (Позиція) – вектор з чотирьох елементів, що визначає положення вікна графіка. Перші два значення вказують координати лівого нижнього кута, два останні – відповідно ширину і висоту вікна.
- **Show number of entities (**Показати кількість сутностей) – вибір опції дозволяє відображати кількість точок у вікні графіка.

Для управління збереженням даних передбачена вкладка **Data History Tab** (Таблиця історії даних) (рис.72):

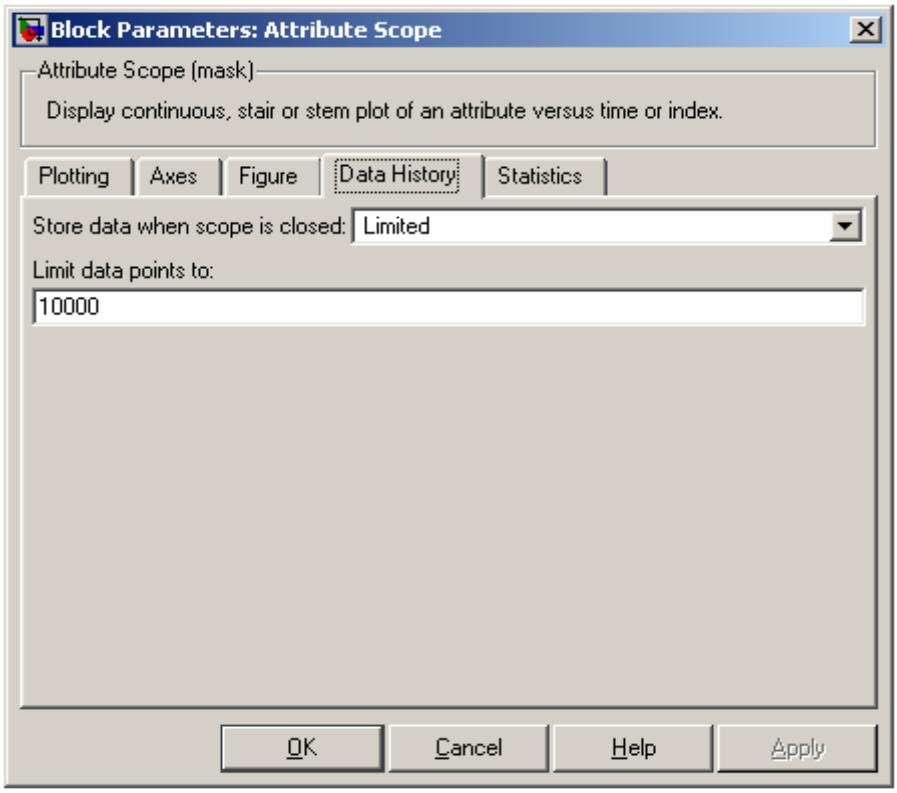


Рис. 72. Закладка налаштування даних

- **Store data when scope is closed** (Збереження даних по завершенню події) – визначає об'єм даних для збереження і може приймати наступні значення:
  - **Unlimited** – збереження всіх даних;
  - **Limited** – збереження обмеженої частини даних;
  - **Disabeled** – відключення опції збереження даних;
- **Limit data points to** (Обмеження кількості точок) – визначає кількість точок для збереження. Поле доступне лише при встановленні значення Limited в полі **Store data when scope is closed**.

Вкладка Статистика **(Statistics Tab)** містить лише одне поле **Number of entities arrived** (Кількість вхідних сутностей) – кількість сутностей, що вийшли з блоку через порт #a.

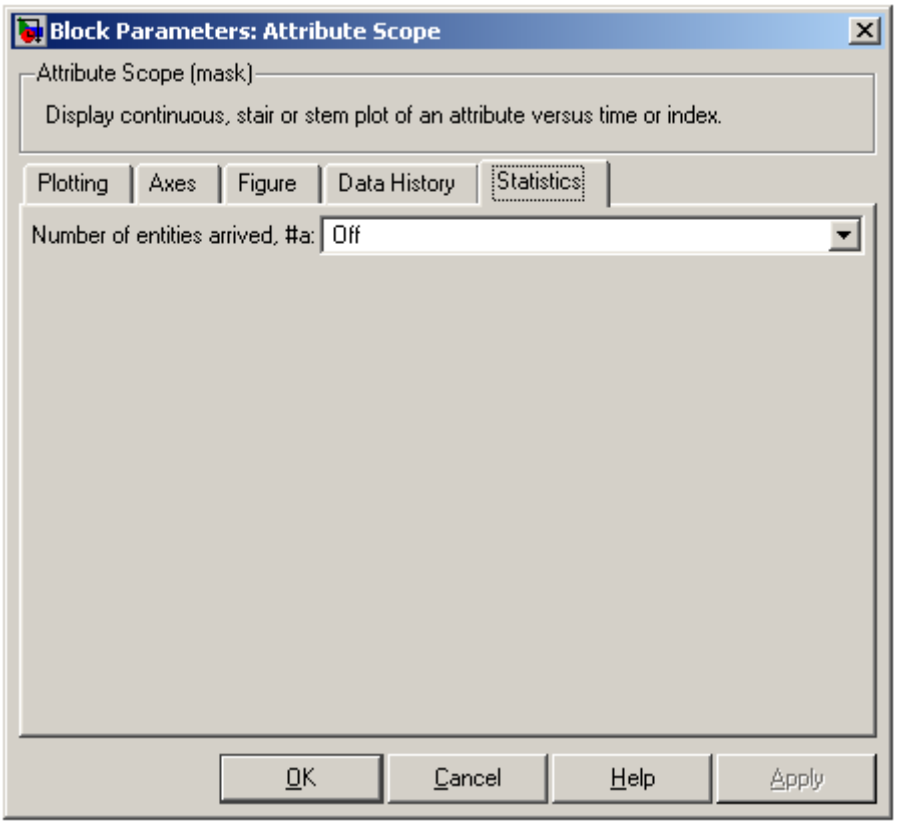


Рис. 73. Закладка налаштування статистики

Приклад. Круговий підхід до вибору входів.

Система містить три вхідні потоки, які послідовно-циклічно стають доступними для надходження сутностей у систему обслуговування, яка складається з одиничного сервісу. Управління вхідними потоками виконується за допомогою перемикача Input Switch, який вибирає наступний вхідний порт після кожного вибування сутності. Коли перемикач вибирає вхідний порт сутності, це робить інші вхідні порти сутності недоступними. Циклічний підхід реалізований за рахунок встановлення параметру Switching criterion (критерій перемикання) в режим Round Robin.

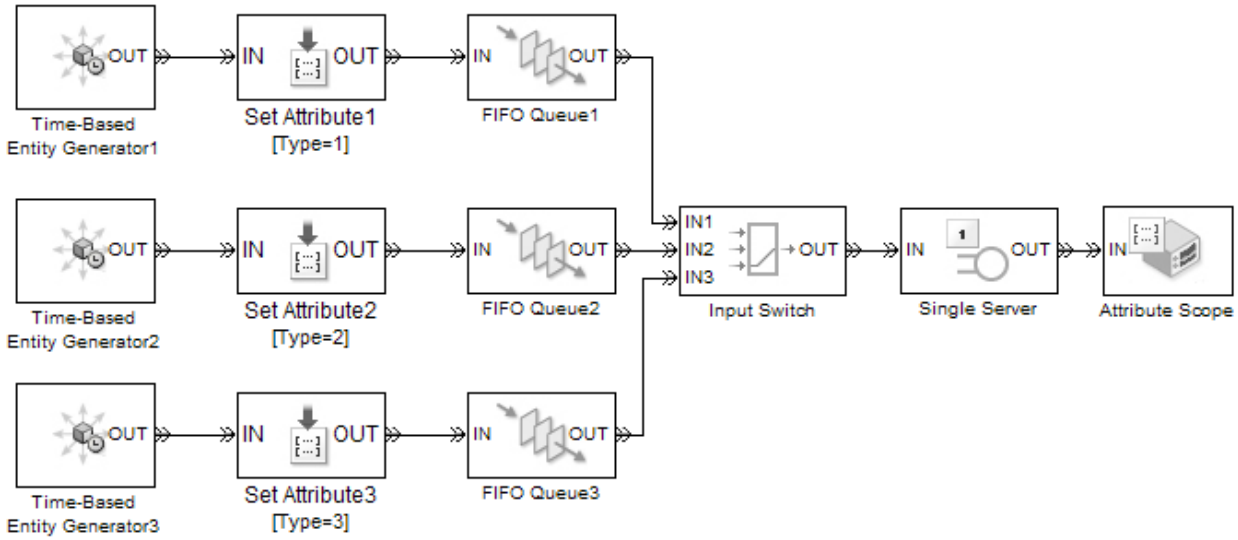


Рис. 74. Модель системи з круговим перемикачем вхідного потоку на сервер

Три набори блоків Set Attribute призначають кожній сутності атрибут, який залежить від генератора сутності, яким вона створенoа. Блок FIFO Queue утворює чергу і блокує сутності перед входом в блок Input Switch. Блокування обумовлене наступними чинниками:

- вхідний комутатор очікує надходження сутності в іншому вхідному порті, відповідно до критерію перемикання round-Robin.
- єдиний блок Сервер зайнятий обслуговуванням сутності, тому його вхідний порт недоступний.

Блок Attribute Scope створює графік проходження сутностей за значенням атрибутів, в даному випадку, за часом (рис.75). Кожному типу сутностей відповідає значення певної висоти.

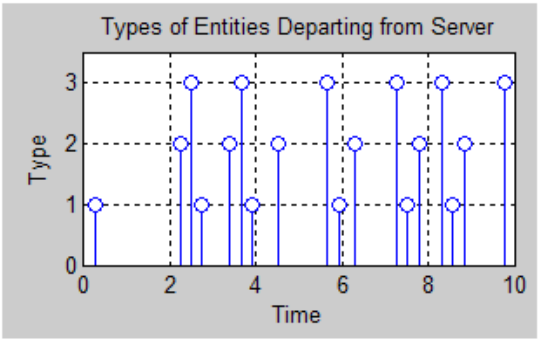


Рис. 75. Графік Attribute Scope за часом

1.7.4.2. X-Y Attribute Scope

X-Y Attribute Scope **(**X-Y атрибути вибірки) – блок діаграми, який будує криву, використовуючи дані з двох реальних скалярних атрибутів вхідних сутностей (рис.76).

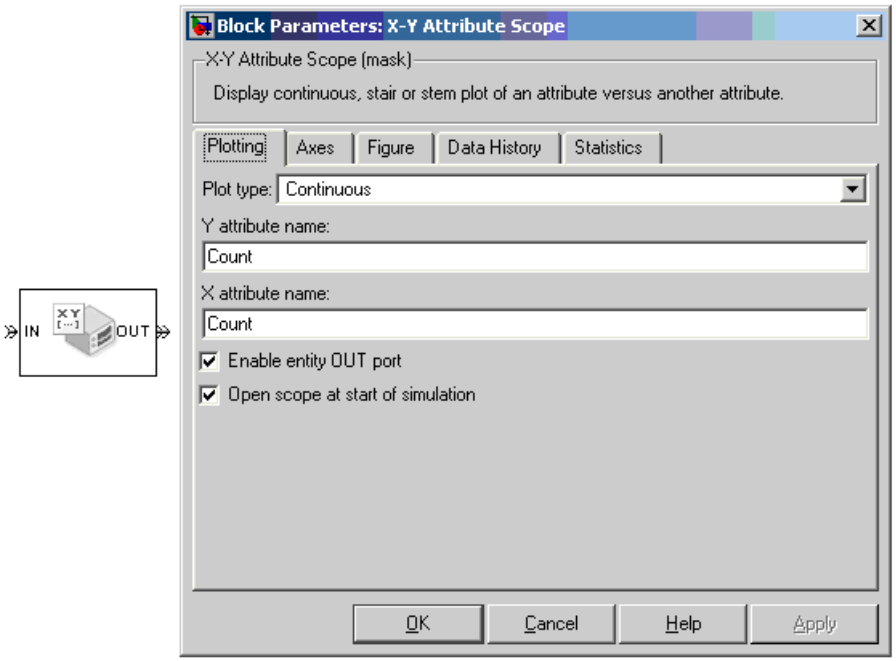


Рис. 76. Блок ***X-Y*** Attribute Scope та його діалогове вікно

Структура та поля блоку такі ж самі, як і для блоку Attribute Scope, який описаний вище. Різниця полягає лише у виборі джерела даних для побудови графіка для осі X. Поле **X attribute name** (і'мя атрибуту X) знаходиться на головній вкладці діалогового вікна **Plotting** і дозволяє у якості значень за віссю X використовувати вибраний атрибут.

1.7.4.3. Discrete Event

Discrete Event (Дискретна подія) – блок для запису дискретних подій вхідного сигналу у робочий простір Workspace середовища Matlab при моделюванні зупинок або пауз. Цей блок аналогічний блоку Sinks, але призначений для використання подієвих сигналів (рис.77).

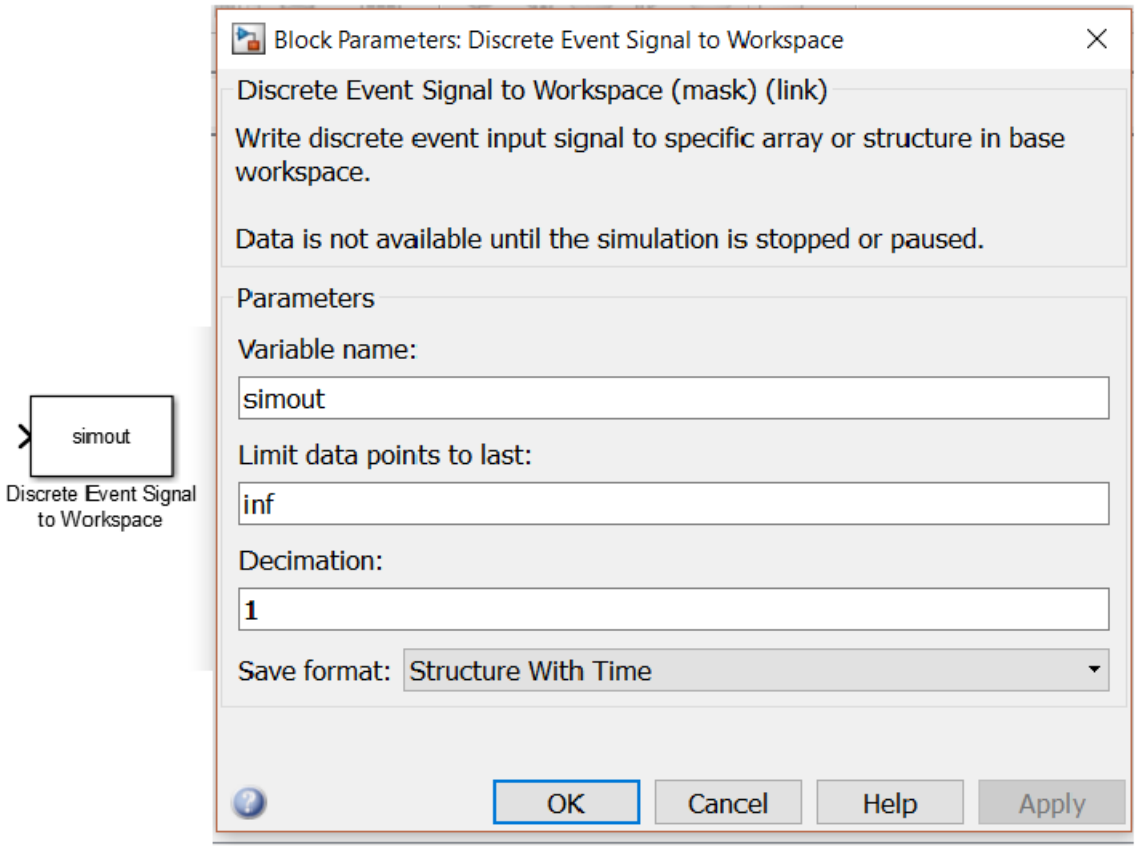


Рис. 77. Блок Discrete Event та його діалогове вікно

Діалогове вікно блоку має наступні параметри:

- **Variable name** (Ім'я змінної) – ім'я структури або масиву для збереження даних;

- **Limit data points to last** (Ліміт даних) – максимальне число вхідних вибірок для збереження;

- **Decimation** (Децимація) – додатнє ціле число N, що визначає коефіцієнт децимації. Блок ігнорує останні N–1 з кожних N вхідних вибірок;

- **Save format** (Формат збереження) – Формат для збереження вихідних даних моделювання в робочий простір. Рекомендований формат для подієвих сигналів є "структура з часом", тому що він показує, коли сигнал приймає кожне значення. Можливі значення формату:

- Structure With Time – структура з часом;
- Structure – структура;
- Array – масив.

Цей блок підтримує тільки типи даних Double і не перетинається із даними, які заносяться у робочу область налаштуванням параметрів імпорт/експорт у конфігурації налаштувань.

Блок може виявляти нульові значення тривалості вхідного сигналу, а також сигналу оновлення, які не обов'язково відповідають часу дії, що визначається динамікою на основі часу. Він не має жодного параметру часу, тому що події на основі сигналів не є істинними.

Цей блок має один вхідний порт для запису в робочий простір і не має портів-сутностей і вихідного сигналу.

1.7.4.4. Entity Sink

Entity Sink – блок для накопичення або блокування сутностей. Блоку доступний лише один вхідний порт **IN** – порт для надходження сутностей та вихідний порт **#a** для виведення кількості сутностей.

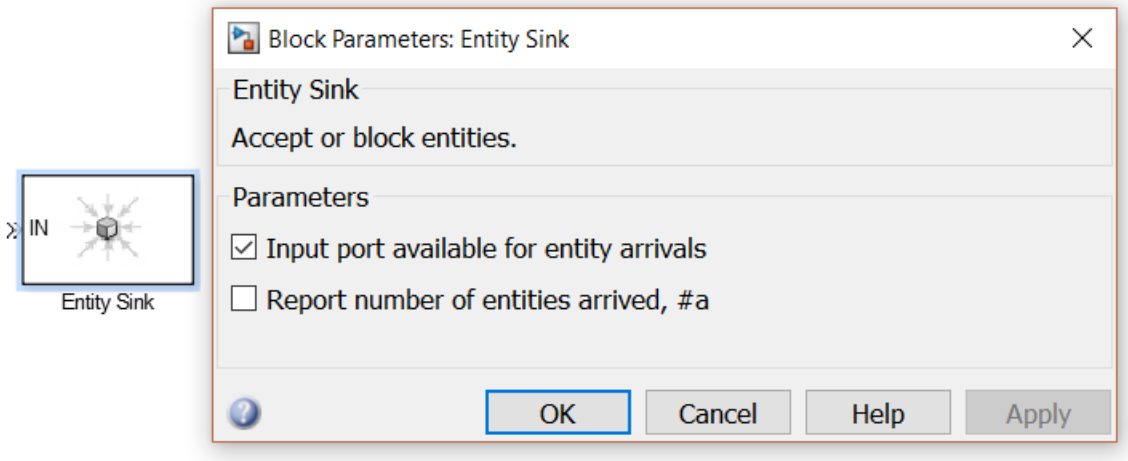


Рис. 78. Блок Entity Sink та його діалогове вікно

Діалогове вікно має лише одну вкладку з двома полями:

- **Input port available arrivals** – вхідний порт відкритий для сутностей. Якщо вибрана ця опція, то блок завжди приймає сутності, що надходять;
- **Report number of entities arrived** – кількість сутностей, що надійшли. Це поле доступне лише коли вхідний порт відкритий. Початкове вихідне значення, яке діє з початку моделювання до першого оновлення блоком, дорівнює 0.

Приклад 1. На рис. 79 представлений фрагмент імітаційної моделі, яка складається з трьох одиничних сервісів. Для кожного сервісу передбачений свій накопичувач вихідних сутностей.

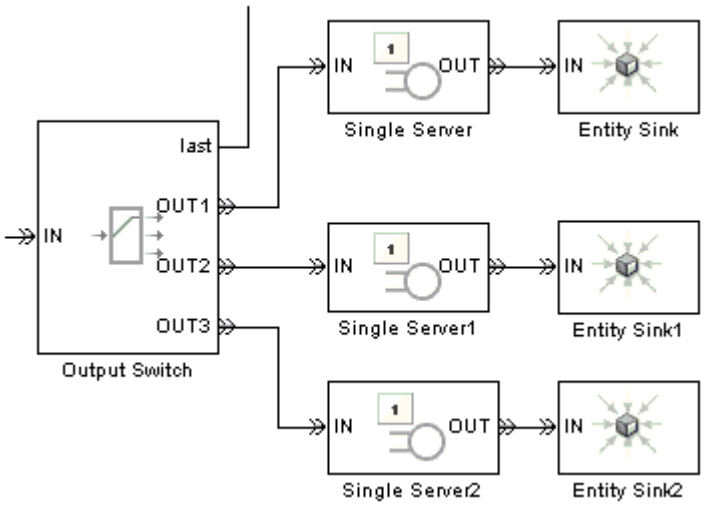


Рис. 79. Фрагмент моделі СМО з використанням блоку Entity Sink

Приклад 2. Використання атрибуту для вибору вихідного порту (рис.80). Нехай замовлення необхідно доставити в одну з трьох можливих зон. Для цього кожному замовленню надається відповідний атрибут (в даному прикладі – випадковим чином). Після обслуговування, замовлення необхідно відсортувати, тому, після виходу із сервера замовлення потрапляють у перемикач (Output Switch), який розподіляє замовлення по різним накопичувачам Entity Sink. Активізувавши вихідний порт **#a**, можна бачити кількість сутностей у кожному з накопичувачів.

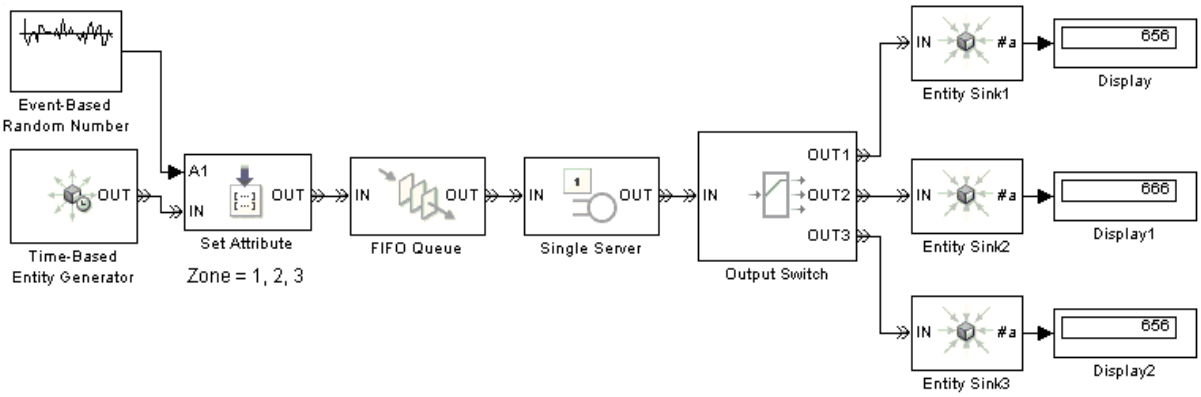


Рис. 80. Модель до прикладу 2

1.7.4.5. Instantaneous Entity Counting Scope

Instantaneous Entity Counting Scope(Обчислення кількості сутностей) – блок обчислення об’єму сутностей використовується для побудови діаграми підрахунку вхідних сутностей в кожний момент досягнутого часу. При зміні часу блок перезапускається із значенням кількості сутностей рівним 1. Початкове значення вихідного сигналу дорівнює 0. Кількість сутностей накопичується протягом заданого часу, але не підсумовується протягом усього моделювання.

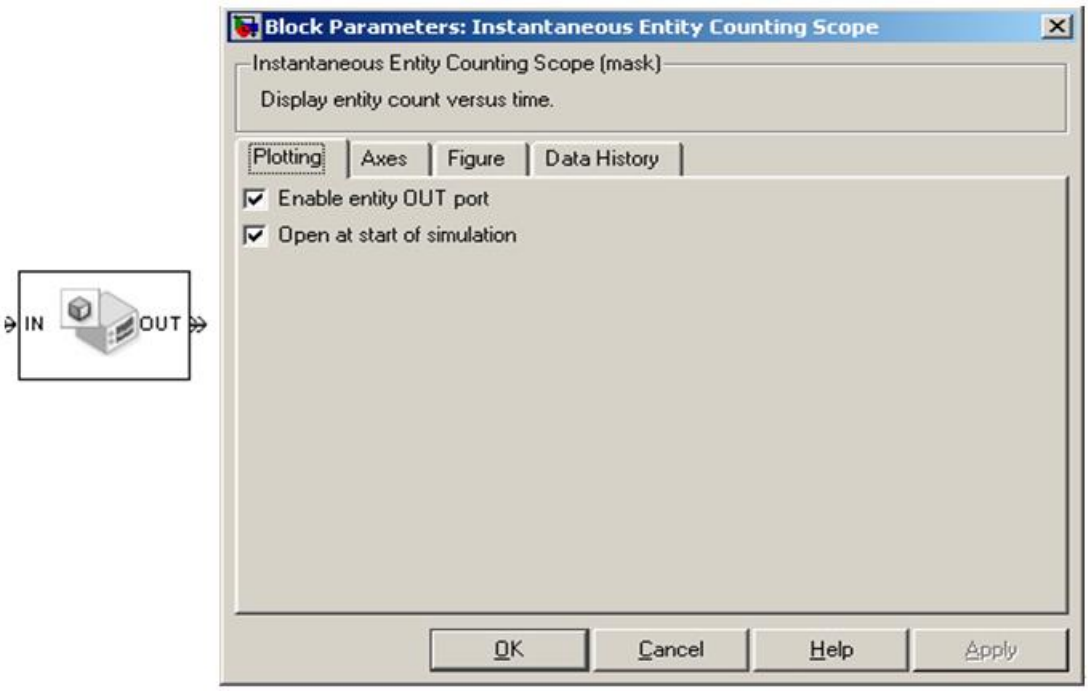


Рис. 81. Блок Instantaneous Entity Counting Scope та його діалогове вікно

Примітка. Якщо ви хочете побудувати загальну кількість номерів вхідних сутностей за весь час моделюванню, підключіть вихід **#d** блоку_Entity Departure Counter до блоку Signal Scope.

Блок має один вхідний порт **IN** для надходження сутностей і один віхідний порт **OUT** для вихідних сутностей. Порт **OUT** стає доступним лише коли активоване поле “Enable entity OUT port”.

Для того щоб відкрити блок діалогове вікно, натисніть на Параметри (Parameters) на панелі інструментів у вікні діаграми.

Діалогове вікно містить декілька вкладок. Головна вкладка **Plotting** (Побудова графіка) містить два поля:

- **Enable entity OUT port** (Активувати вихідний порт сутностей) – дозволяє подавати накопичені сутності на вихідний порт. Якщо цей прапорець зняти, блок поглинає вхідні сутності;

- **Open scope at start of simulation** (Відкрити накопичувач на початку моделювання) – при виборі цієї опції на початку симуляції викликається вікно діаграми. Якщо цей прапорець зняти, то запустити вікно діаграми можна за допомогою подвійного кліку на зображенні блоку.

Вкладка **Axes Tab** (координатні вісі**)** представлена на рис. 82 і містить наступні поля:

- **Initial X axis lower limit** (початкова нижня межа X)**, Initial X axis upper limit** (початкова верхня межа X) – визначають інтервал на вісі X на початку моделювання. Інтервал може змінитися за рахунок масштабування, автоматичним масштабуванням, або якщо значення X лежить поза встановленим лімітом;

- **- Initial Y axis lower limit** (початкова нижня межа Y), **Initial Y axis upper limit** (початкова верхня межа Y) – визначають інтервал на вісі Y. Інтервал може змінитися масштабуванням, автоматичним масштабуванням, або якщо значення Y лежать поза встановленим лімітом;
- **If Y value is beyond limit** (якщо значення Y за рамками ліміту) – визначає межі зміни значення Y, якщо вони за межами ліміту;
- **Show grid** (відобразити сітку) – вмикає і вимикає сітку.

Вкладка **Figure** (фігура) (рис.82) дозволяє встановлювати підписи графіка та його вісей і містить наступні поля:

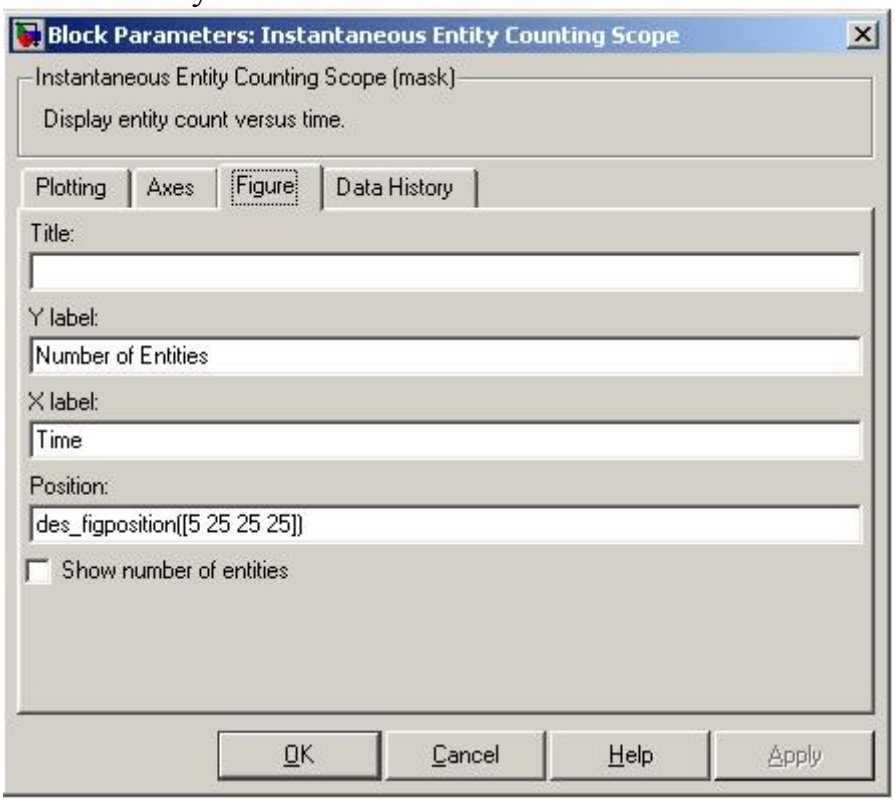


Рис. 82. Закладка Figure блоку Instantaneous Entity Counting Scope

- **Title** (заголовок) – текст, який з'являється в якості назви діаграми, над вісями;
- **Y label** (вісь Y) – текст, який з'являється зліва від вертикальної вісі;
- **X label** (вісь X) – текст, який з'являється нижче горизонтальної вісі;
- **Position** (позиція) – вектор з чотирьох компонентів [координати лівого нижного кута, ширина, висота], що визначає область видимості;
- **Show number of entities** (показувати номер сутностей) –відображає кількість точок на графіку.

Вкладка **Data History** (історія даних) (рис.83) включає два поля:

- **Store data when scope is closed** (зберегти дані, коли накопичувач закритий) – визначає об’єм даних для перегляду. Поле може приймати два значення: Limited – збереження останніх даних, кількість яких визначена наступним полем **Limit data points to**; Unlimited – дозволяє переглядати всі дані;

- **Limit data points to** (обмеження кількості точок) – поле визначає кількість точок для збереження, задану натуральним числом.

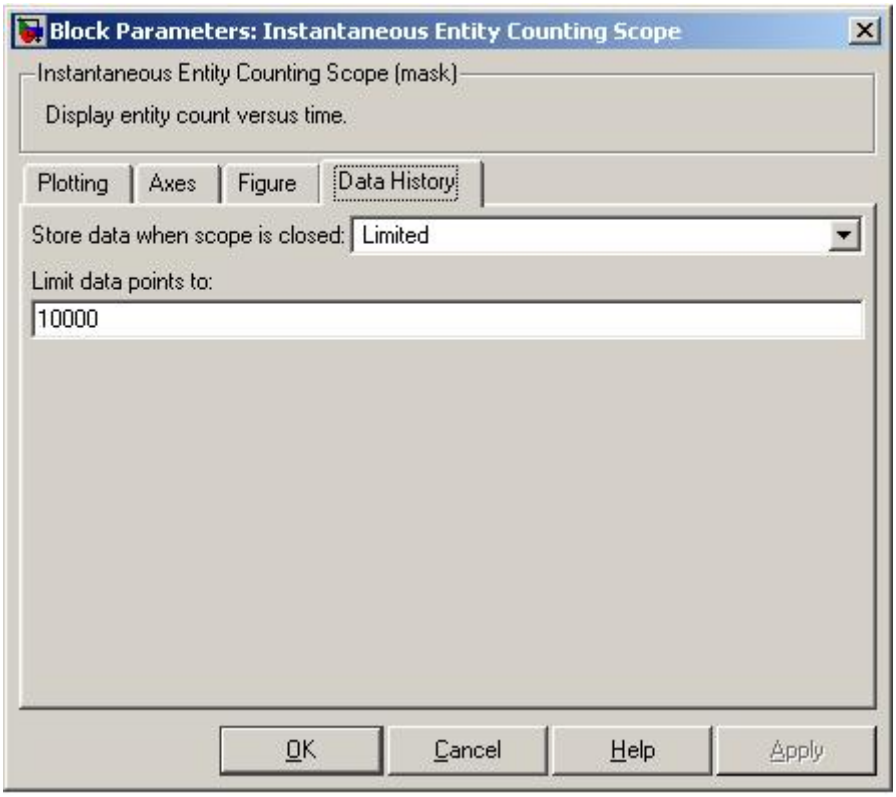


Рис. 83. Закладка Data History блоку Instantaneous Entity Counting Scope

Приклад. У наведеному прикладі блок Infinite Server в деякі моменти часу завершує обслуговування декількох сутностей. Блок Instantaneous Entity Counting Scope виконує підрахунок кількості сутностей, для яких одночасно завершилось обслуговування під час моделювання.

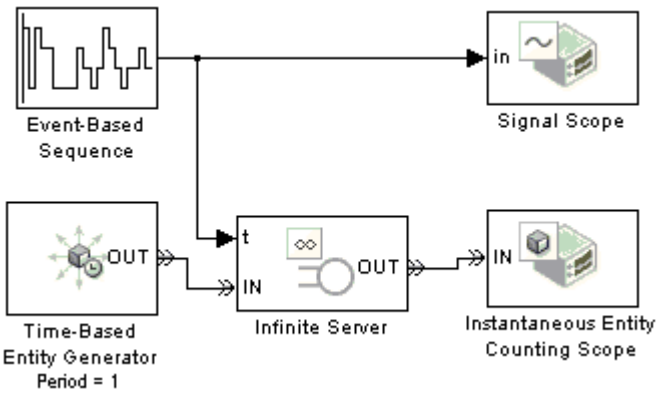
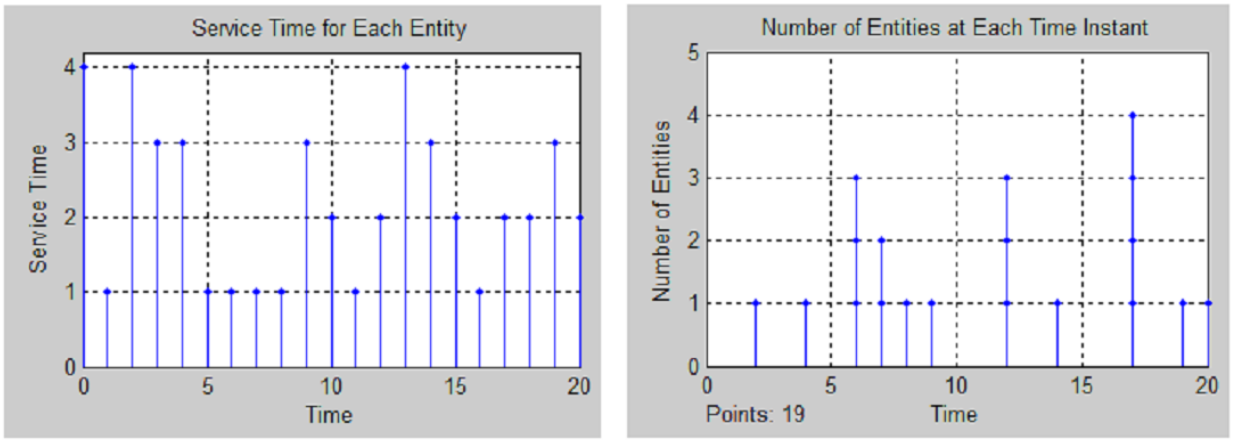


Рис. 84. Приклад використання блоку Instantaneous Entity Counting Scope

Результати моделювання наведені на рис. 85, де на лівому графіку відображено час обслуговування кожного замовлення, який видається блоком Signal Scope. Графік, наведений праворуч, є результатом роботи блоку і відображає скільки сутностей в певні моменти часу одночасно вийшли з блоку обслуговування.



а) б)

Рис. 85. Результати роботи моделі з використанням блоків а)Signal Scope б) Instantaneous Entity Counting Scope

1.7.4.6. Instantaneous Event Counting Scope

Instantaneous Event Counting Scope (Події графіку розраховуються в залежності від часу) – блок відображає графік подій (рис.86).

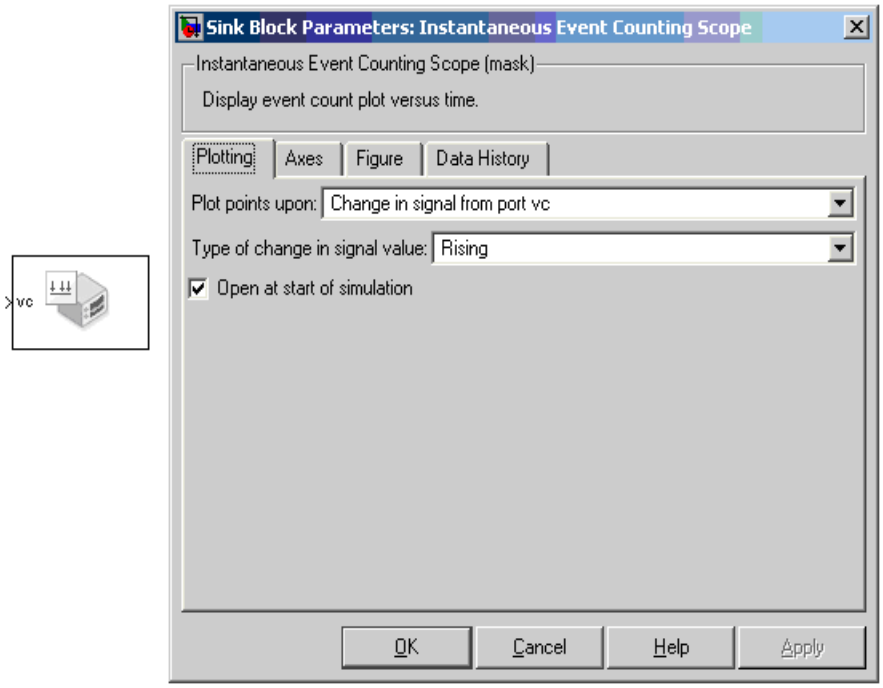


Рис.86. Блок, що відображає графік подій

Блок оновлює розрахунок, починаючи з 1, коли відбувається зміна у часі. Розрахунок є кумулятивним для кожної миті часу, але некумулятивним для всієї симуляції.

Вхідні порти доступні для блоку представлені у табл.12.

Таблиця 12. Вхідні порти блоку Instantaneous Event Counting Scope

Позначення	Опис
ts	збільшення значення лічильника при зміні сигналу. Порт доступний лише коли встановлено **Plot points upon** -> Sample time hit from port ts.
tr	збільшення значення лічильника при досягненні тригерного критерію. Порт доступний лише коли встановлено **Plot points upon** -> Trigger from port tr.
vc	збільшення значення лічильника при досягненні значення змінної. Порт доступний лише коли встановлено **Plot points upon** -> Change in signal from port vc.
fcn	збільшення значення лічильника при надходженні сигналу, заданого функцією. Порт доступний лише коли встановлено **Plot points upon** ->Function call from port fcn.

Діалогове вікно

Для відкриття діалогового вікна натисніть «Parameters» на панелі інструментів у вікні графіку. Діалогове вікно містить декілька вкладок, призначення та структура яких подібна до відповідних компонентів блоку Instantaneous Entity Counting Scope, описного вище. Розглянемо головну вкладку **Plotting Tab**, на якій визначаються основні параметри графіку, який формується даним блоком.:

- Plot points upon (Побудова за точками за умови) – визначає тип події, коли блок збільшує свій лічильник. В полі можуть бути встановлені наступні режими (табл.10):

- **Trigger from port tr**;
- **Change in signal from port vc** ;
- **Sample time hit from port ts**;
- **Function call from port fcn.**

- Type of change in signal value – визначає напрямок зміни вхідного сигналу, при якому збільшується значення лічильника.

- Open scope at start of simulation – встановлення цієї опції відкриває вікно графіку при запуску моделювання. В випадку коли опція не вибрана, вікно графіку можна відкрити шляхом подвійного кліку по значку блоку.

Приклад. Наведений нижче приклад демонструє використання блоку **Instantaneous Event Counting Scope** для визначення одночасних подій, які б ви хотіли бачити одночасно.

Нехай система включає два генератори сутностей з періодами 1 і 1/3 для створення одночасних замовлень кожну секунду, так що пріоритет подій визначає які сутності надходить в чергу першими.

Модель використовує два блоки Event-Based і Event Generator, які отримують однаковий вхідний сигнал. З графіка можна бачити, що одночасні

події відбуваються кожну секунду, згідно до заданого періоду і в моменти їх співпадіння – одночасно.

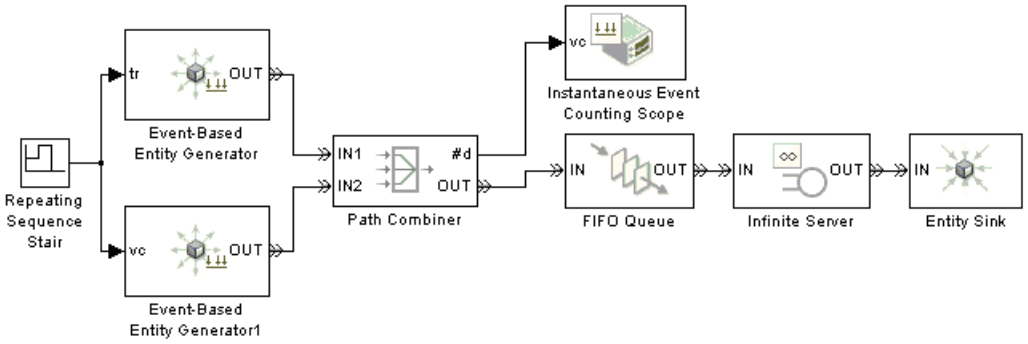


Рис. 87. Модель з одночасними подіями

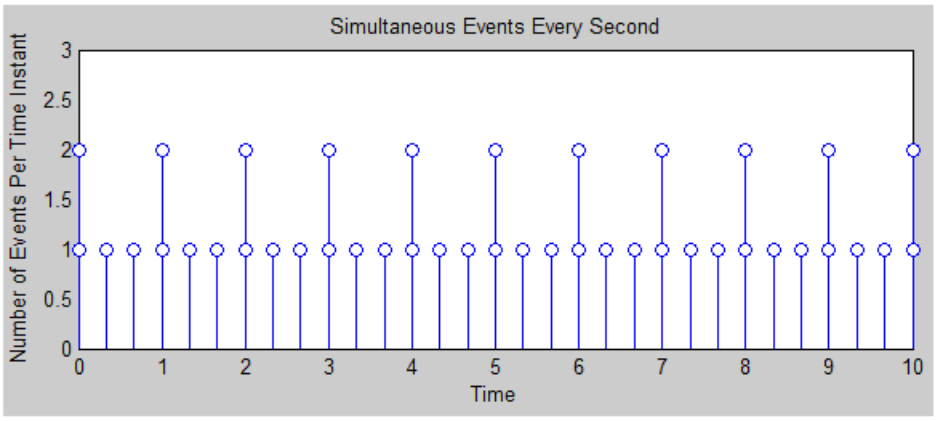


Рис. 88. Графік надходження подій

В даному прикладі замість блоку Instantaneous Event Counting Scope можна було використати блок Instantaneous Entity Counting Scope.

1.7.4.7. Signal Scope

Signal Scope (Область сигналу) – блок створює сюжет, використовуючи дані сигналу. Сюжет особливо підходить для даних, пов'язаних з моделюванням дискретних подій або даних, пов'язаних з особами, тому що сюжет може включати значення нульової тривалості. Дані для вертикальної вісі виходять із сигналу, підключеного до вхідного порту **in** (рис.89).

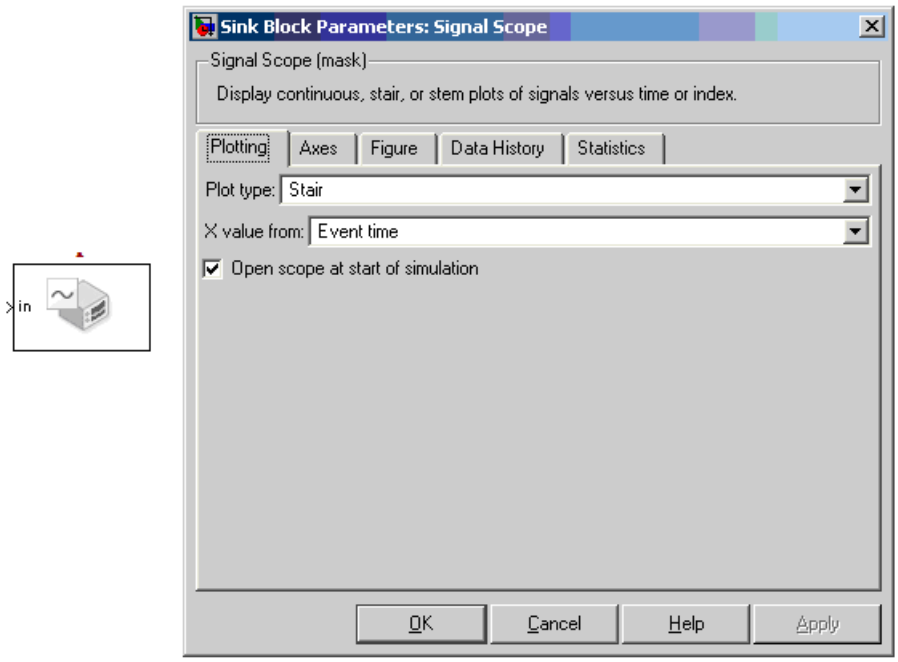


Рис. 89. Діалогове вікно Signal Scope

Для блоку доступний один вхідний порт **in** (вхідний сигнал), який є джерелом значеня для графіку за віссю Y, та вихідний порт **#c**, який виводить кількість точок для побудови графіку.

Діалогове вікно

Головна вкладка **Plotting** містить поля, які визначають вигляд графіка:

- **Plot type** (тип графіка) – поле визначає тип графіка і може мати значення: Stair (ступінчатий), Stem (дискретний «кружечок на ніжці»), Continuos (неперервний);
- **X value from** (значення X з) – вибір даних для горизонтальної вісі. Поле може приймати наступні значення:

• **Event time** (час події) – графік сигналу залежить від часу моделювання, наприклад, графік можна застосовувати для того, щоб бачити як з часом зростає черга;

• **Index** (Індекс) – графік представляє індекс значення. Перше значення сигналу протягом моделювання має індекс 1, друге значення –індекс 2, і так далі. Цю опцію можна використовувати для сигналу, який має значення нульової тривалості, щоб визначити точну послідовність одночасних сигналів.

Наведені нижче графіки ілюструють різні джерела даних для горизонтальній вісі. Ділянки схожі, за винятком того, що друга ділянка має рівномірну горизонтальну відстань, а не на основі часу відстань між сусідніми точками.

Наведені нижче графіки, що ілюструють використання різних джерел для горизонтальної вісі X. Графіки схожі, за винятком того, що на другому графіку горизонтальні відстані рівномірні, а не на основі часу.

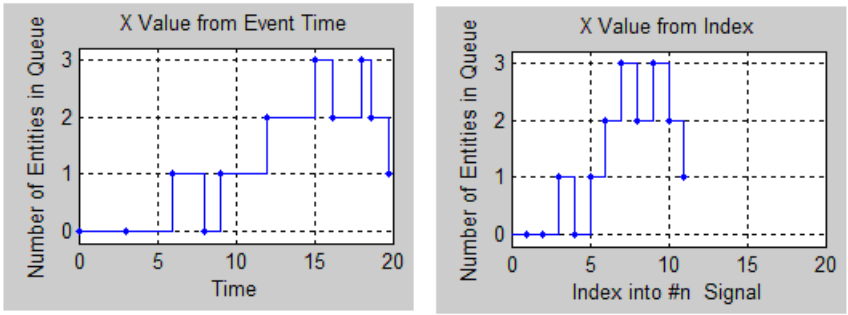


Рис. 90. Графіки використання різних джерел для горизонтальної вісі X

Структура та значення полів закладок Axes, Figure, Data History така ж сама, як і для блоку Instantaneous Entity Counting Scope, описаного вище.

Вкладка **Statistics** (Статистика) призначена для отримання статистичних характеристик блоку має лише одне поле: **Number of points plotted #c,** встановлення якого у режим on відкриває відповідний вихідний порт для отримання кількості точок для побудови графіка після призупинення або завершення роботи блоку.

1.7.4.8. X-Y Signal Scope

X-Y Signal Scope (графік сигналів X-Y) –. будує криву, використовуючи дані двох вхідних сигналів. Графік особливо підходить для даних, пов'язаних з моделюванням дискретних подій або даних, пов'язаних із сутностями (рис. 91).

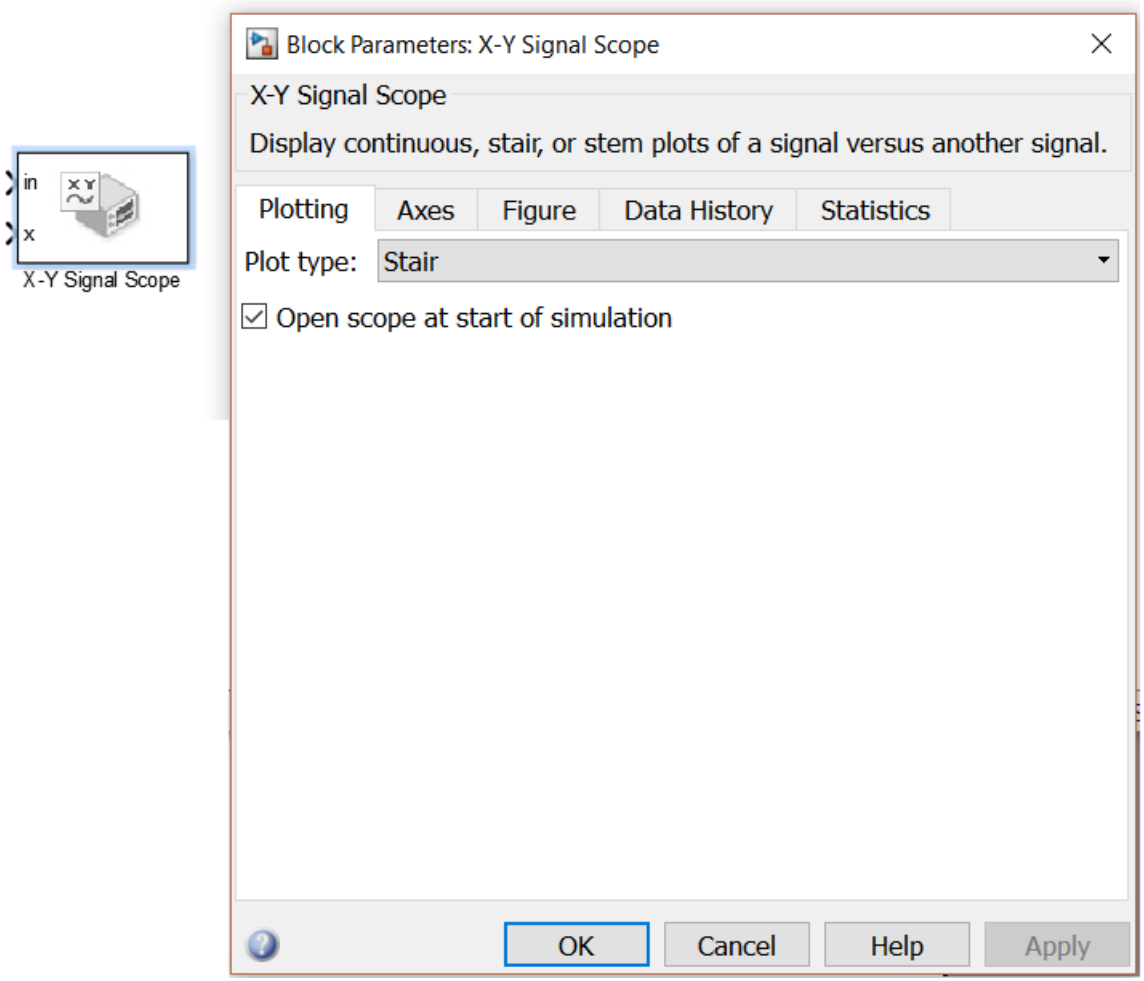


Рис.91. Вигляд блоку X-Y Signal Scope та його діалогове вікно

Блок має два вхідних порти: **in** – для надходження даних для вертикальної вісі; **x** – для задання даних за віссю X, та один вихідний порт #c для виведення кількості точок для побудови графіка. Початкове вихідне значення, що діє з початку моделювання до першого його оновлення блоком, дорівнює 0.

Діалогове вікно

Щоб відкрити діалогове вікно блоку, натисніть праву кнопку миші на блоці та виберіть Parameters.

Структура та поля вкладок блоку X- Signal Scope аналогічні структурі та полям блоку Signal Scope, який описаний вище.

Приклад. На рис. 92 представлена імітаційна модель з використанням блоку X-Y Signal Scope, де у якості значень за віссю X береться лінійний сигнал Ramp, а даними для вісі Y є оброблені замовлення одиничним сервісом.

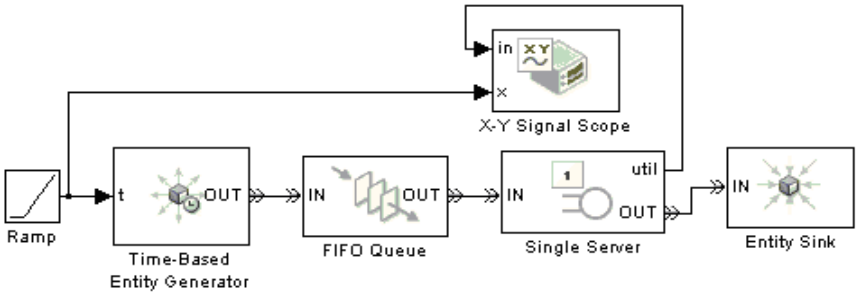


Рис. 92. Модель з використанням блоку X-Y Signal Scope

Результати моделювання представлені графіком, сформованим блоком X-Y Signal Scope представлені на рис. 93.

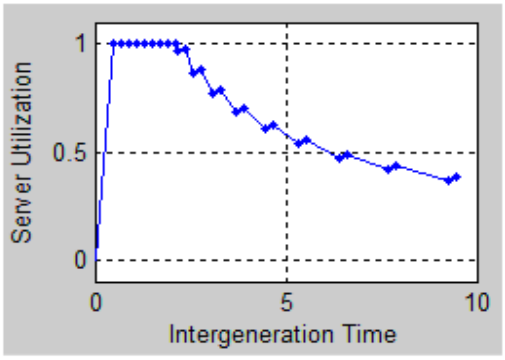


Рис. 93. Результати моделювання з використанням блоку X-Y Signal Scope

1.8. Управління часом (Timing)

Блоки бібліотеки Timing застосовують для управління часом. За їх допомогою можна встановлювати часові обмеження на перебування сутності у черзі або на час обслуговування сервером. Бібліотека містить чотири блоки:

- Start Time – початок відліку часу;
- Read Time – зчитування поточного часу;
- Shedule Timeout – встановлення обмеження часу;
- Cancel Timeout – відміна часових обмежень.

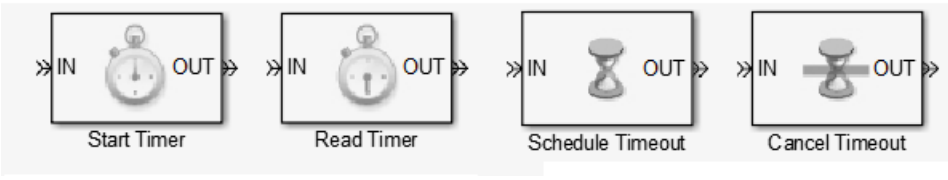


Рис. 94. Блоки бібліотеки Timing

1.8.1. Start Timer

Start Timer – призначає кожному вхідному об’єкту незалежний таймер та починає відлік часу. Якщо об’єкт має таймер з таким самим ім’ям, таймер продовжуватиме працювати або буде перезапущений в залежності від

налаштувань в полі **«If timer was already started»**: Можливість попередження та продовження може бути корисна для налагодження та запобігання помилок при моделюванні. Будь–які інші таймери, пов’язані зі вхідним об’єктом залишаються невикористаними.

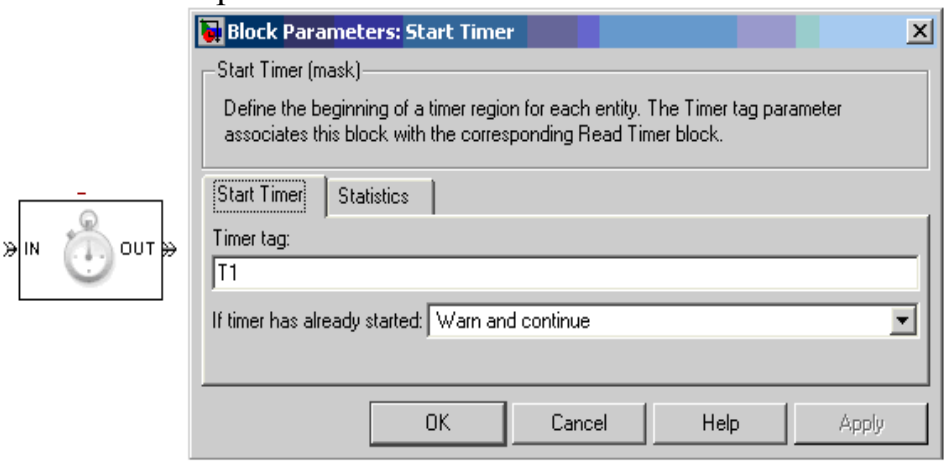


Рис. 95. Блок Start Timer

Даний блок працює з блоком Read Timer (таймер, що зчитує). Щоб дізнатися значення таймеру, вказаного в блоці, потрібно знайти таймер з таким же ім’ям в блоці Read Timer. Для більшої інформації про використання даної пари блоків див. Using Timers.

Блок має один вхідний порт **IN** для надходження сутностей та один вихідний порт **OUT** для виходу, а також підключення вихідного порту **#d** для виводу кількості вихідних сутностей з початку роботи, значення оновлюється після виходу кожного об’єкту. Початкове значення сигналу вихідного порту до моменту першого оновлення дорівнює 0.

Діалогове вікно

Діалогове вікно містить дві вкладки. На вкладці **Start Timer** (запуск таймеру) розташовано два поля:

- **Timer Tag** (тег таймеру) – задає ім’я таймеру для кожного об’єкта;
- **If timer was already started** (якщо таймер вже стартував) – визначає поведінку системи при співпаданні назв таймерів. Поле може приймати наступні значення:

• **Warn and continue** – попереджати та продовжувати;

• **Continue** – продовжувати;

• **Restart** – перезапустити.

Закладка **Statistics** (Статистика) має лише одне поле **Number of entities departed** (Кількість вихідних сутностей) дозволяє відкрити вихідний порт #d для виводу кількості вихідних об’єктів (рис. 96).

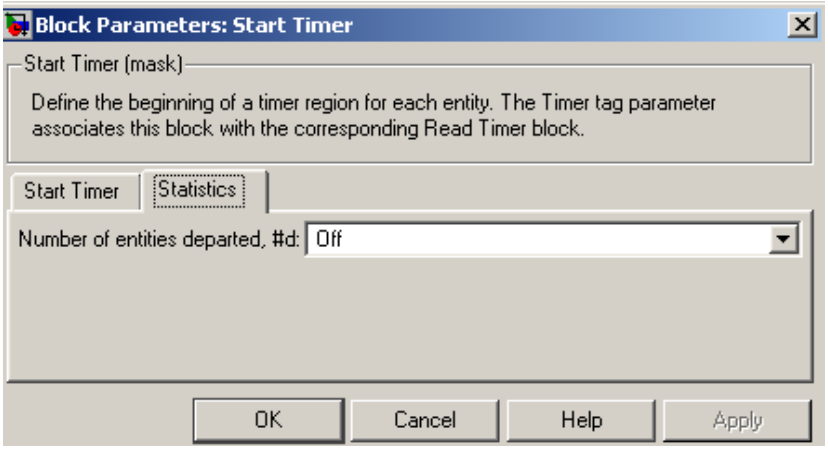


Рис. 96. Діалогове вікно блоку Start Timer

1.8.1.1. Стандартні процедури встановлення блоку

Для застосування блоку необхідно виконати певний порядок дій:

1. Визначити точки моделі, з яких необхідно починати таймінг та задати значення таймеру.

2. Встановити Start Timer блок в точку моделі, звідки потрібно розпочати таймінг.

3. В діалоговому вікні Start Timer блоку задати назву таймера в полі Timer tag. Значення даного поля відрізнятиме даний таймер від інших таймерів, які вже можуть бути пов'язані з цим же об'єктом. Коли виконання процесу опиняється в точці зі встановленим Start Timer блоком, блок прив'язує назву таймера до об'єкта та починає відлік часу (таймінг).

4. Встановити Read Timer блок в точці моделі, де необхідно зчитувати значення таймеру.

5. В діалоговому вікні Read Timer блоку встановити таку назву таймеру (що була введена в поле Timer Tag), яка була задана в відповідному Start Timer блоці.

Коли об'єкт з відповідним таймером надходить до блоку Read Timer, блок зчитує значення таймеру. Використовуючи вкладку Statistics діалогового вікна Read Timer блоку, можливо налаштувати блок таким чином, щоб миттєво дізнаватися значення або середні значення всіх таймерів, пов'язаних з даним блоком.

Якщо є необхідність в декількох незалежних таймерах на кожний із об'єктів, тоді процедуру необхідно повторювати для кожного з таймерів. Для більш детальної інформації див. Timing Multiple Processes Independently.

1.8.1.2. Таймінг декількох об'єктів за допомогою одного таймеру

Якщо модель включає декілька блоків обробки, тоді різні об'єкти можуть використовувати різні шляхи. Для того, щоб таймер охоплював декілька шляхів, необхідно додати відповідну кількість Start Timer та Read Timer блоків, вказуючи однакову назву таймерів (поле Timer Tag) для кожного таймеру.

Приклад. Якщо модель містить блоки маршрутизації, то різні об'єкти можуть використовувати різні шляхи. Щоб таймер охоплював кілька шляхів, можна включити кілька блоків Start Timer або кілька блоків читання таймера в моделі, використовуючи один і той же параметр параметра Tag Timer у всіх блоках таймера.

1. На рис. 97 наведена модель, де сутності розділяються у два потоки за допомогою блоку Output Switch і надходять на обслуговування до різних серверів. Таймер продовжує відлік часу, незалежно від обраного шляху. Нарешті, кожен об'єкт переходить до одного з двох блоків читання таймера, який читає значення таймера.

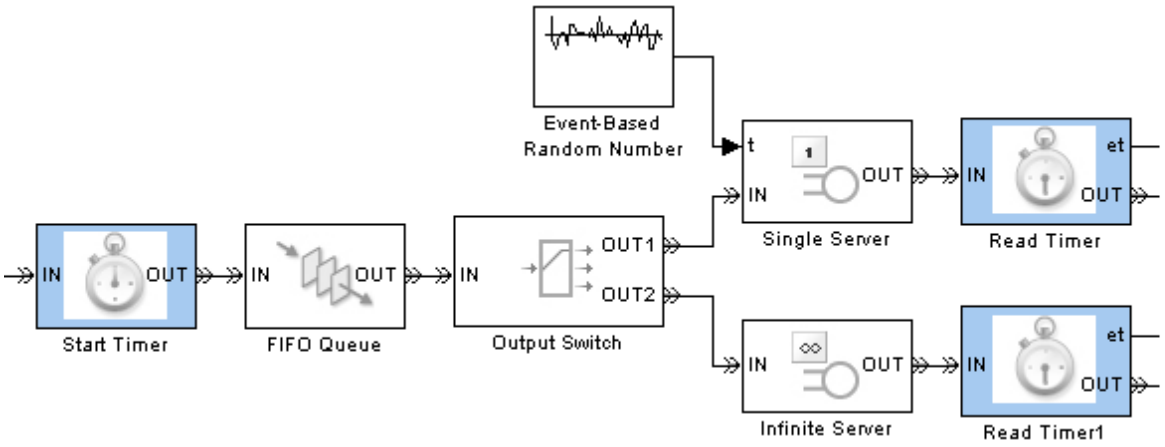


Рис. 97. Модель, де сутності розділяються у два потоки за допомогою блоку Output Switch

2. На рис. 98 об'єкти чекають у двох різних чергах перед переходом до одного сервера. Блоки таймера вимірюють час, який кожний об'єкт витрачає у відповідній парі черги-сервера. Два блоку запуску таймера, налаштовані з однаковим значенням параметра тегу таймера, гарантують, що всі об'єкти керуються таймером, незалежно від шляху, який вони виконують перед досягненням сервера.

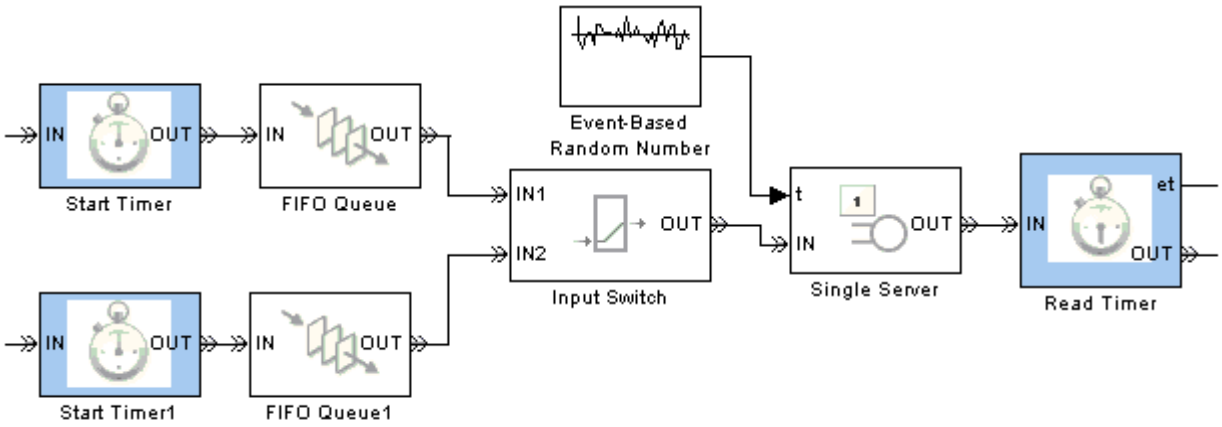


Рис. 98. Модель, де об'єкти чекають у двох різних чергах перед переходом до одного сервера

1.8.2. Зчитуючий Таймер (Read Timer)

Read Timer – виведення статистичних даних про вказаний (названий) таймер, пов'язаний з надходженням сутностей (рис. 99). Блок зчитує значення таймера, відповідного блоку Start Timer (Запуску таймера).

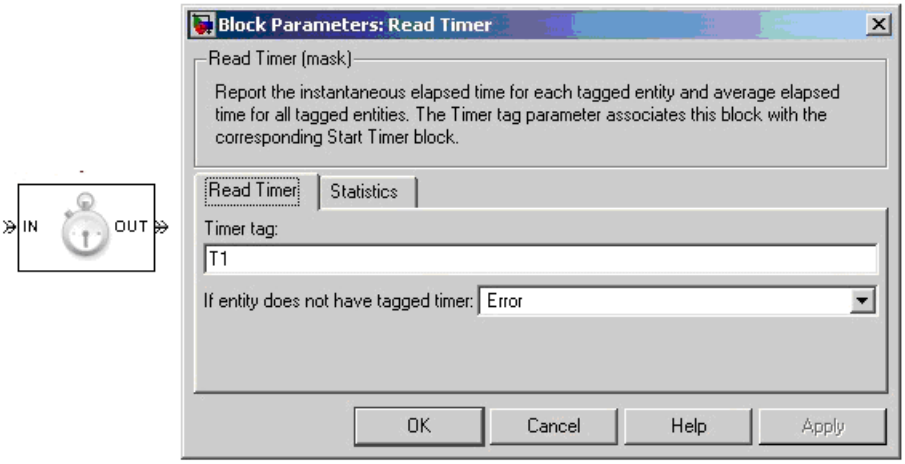


Рис. 99. Блок Read Timer та його діалогове вікно

Використовуючи час, що минув у звіті, та відзвітувавши про середні проміжкові параметри часу, можна через вихідні порти et та w відповідно, налаштувати параметри статистики блоку.

Примітка. Якщо вхідна сутність не відповідає таймеру з вказаним іменем, то існує можливість налаштувати блок, або видавати помилку, або ігнорувати відсутність таймера. В останньому випадку, вихідні сигнали зберігають свої колишні значення.

Блок має один вхідний порт **IN** для вхідних сутностей і один вихідний **OUT** – для виходу. При відповідних налаштуваннях полів закладки статистики (Statistics) можна активізувати вихідні порти сигналів (табл.13), час поновлення яких після виходу сутності.

Таблиця 13.

Назва	Опис	Порядок поновлення
#d	Кількість сутностей, які пройшли через блок з початку моделювання.	3
#t	Загальна кількість сутностей, які пройшли через блок і відповідають таймеру з вказаним ім’ям.	2
et	Миттєвий відлік часу для прибуваючої сутності, якщо вона відповідає таймеру з вказаним ім'ям.	2
w	Середнє значення **et** всіх сутностей, що надійшли в цей блок і відповідають таймеру з вказаним ім'ям.	1

Початкове значення вихідного сигналу, який діє з початку моделювання (симуляції) до першого оновлення блоком, дорівнює 0 для всіх сигналів.

Діалогове вікно.

Діалогове вікно містить дві закладки. На першій закладці Read Timer (Зчитування таймеру) розташоване два поля:

- **Timer Tag** (Тег таймеру) – визначає ім'я тегу поточного таймеру;
- **If entity does not have tagged timer** (Якщо сутності не призначений таймер) – дозволяє управляти поведінкою таймера, коли він не призначений відповідній сутності. Поле може приймати наступні значення:

• **Error** – повідомлення про помилку;

• **Ignore** – ігнорувати.

На вкладці **Statistics** (Статистика) розташовано чотири поля:

- **Number of entities departed , #d** – кількість сутностей, які пройшли через блок;
- **Number of entities departed with specified tag, #t** – кількість сутностей, які пройшли через цей блок з вказаним ім'ям. Якщо значення поля **If entity does not have tagged timer** визначене як **Ignore**, то значення **#t** може бути менше, ніж значення **#d**;
- **Elapsed time, et** – миттєвий відлік часу;
- **Average elapsed time, w** – середнє значення часу що пройшов.

Ці поля визначають доступ до відповідних портів. Для того, щоб порт був доступний, необхідно встановити значення **on.**

Приклади використання блоку Read Timer наведені вище в описі блоку Start Timer.

.

1.8.3. Schedule Timeout

Schedule Timeout (Задання режиму очікування) – блок задає режим очікування для кожної сутності (тайм-аут). Події затримки дозволяють обмежити час, який сутність витрачає на шляхи під час моделювання. Топологічно цей блок позначає початок шляху сутності, яка відповідає встановленому тегу

(рис. 100).

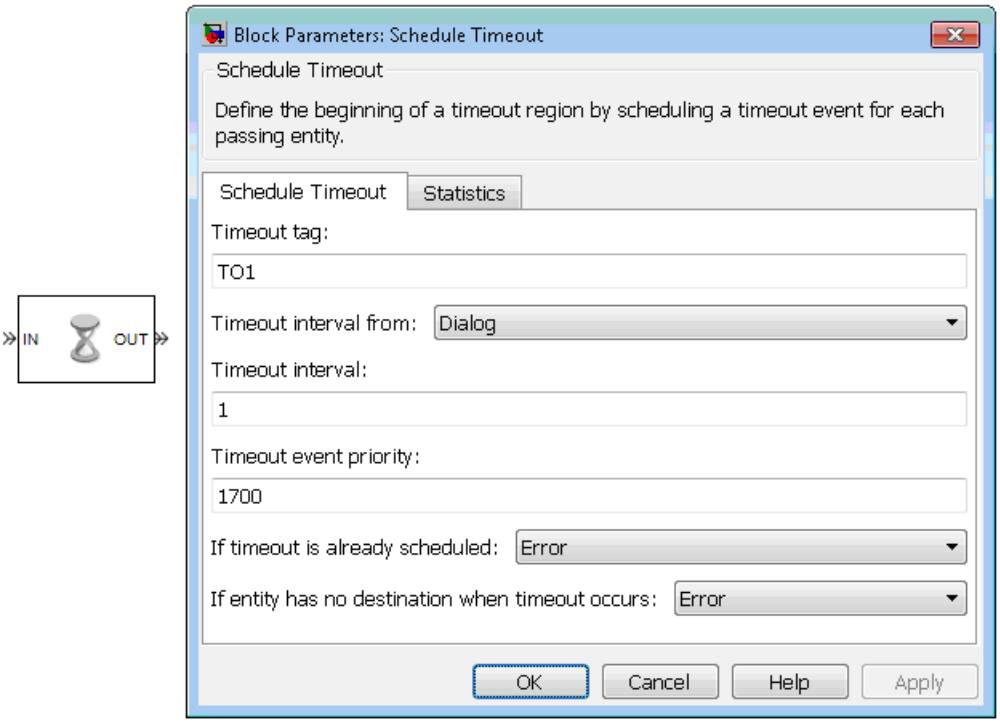


Рис. 100. Блок Schedule Timeout та його діалогове вікно

Подія тайм-аут знаходиться в календарі подій. Час події дорівнює часу прибуття сутності плюс інтервал тайм-ауту. Інтервал тайм-ауту задається через параметр, атрибут або сигнал. Блок визначає абсолютний час події з тайм-аутом сутності після її прибуття. Наприклад, якщо сутність прибуває в T=5, а інтервал тайм-ауту 3 секунди, тоді графік очікування тайм-аутів подій буде дорівнювати T=5+3=8.

Календар подій може містити кілька незалежних тайм-аутів для однієї сутності, якщо вони мають різні теги тайм-ауту. Цей блок не впливає на тайм-аути подій, що мають інші теги тайм-ауту.

Пріорітет подій, визначається за допомогою параметра **Timeout event priority**. Зверніть увагу, що, якщо тайм-аут подій для двох сутностей має різні пріоритети подій і планується за тим же значенням, або досить близьким до нього, тоді пріоритетні значення будуть визначитись для сутності, чий тайм-аут був першим.

Примітка. Якщо інтервал тайм-ауту вказаний за допомогою сигналу на основі події, то необхідно переконатися, що його оновлення відбуваються до прибуття сутності.

Визначення тайм-аутів подій

Якщо для конкретної сутності виникає подія тайм-аут, вона намагається вийти з блоку, у якому знаходиться, через порт **TO** (тайм-аут). Щоб налаштувати блок так, щоб він мав порт **TO**, необхідно вибрати **Enable TO port for timed-out entities** у діалоговому вікні відповідного блоку. Якщо сутність знаходиться в блоці, який немає порту **TO**, тоді режим очікування блокується. Якщо сутність не має місця призначення, то в момент завершення

або зупинки виконання моделі при виникненні події тайм-аут з'являється повідомлення про помилку.

Для скасування тайм-ауту подій, застосовується блок **Cancel Timeout.** Неможна безпосередньо змінювати запланований час або пріоритет події тайм-ауту, який вже є у календарі подій. Однак можна скасувати тайм-аут подію, а потім запланувати нову.

Блок Schedule Timeout має один вхідний порт **IN** для надходження сутностей, а також можна активізувати сигнальний вхідний порт **ti** для визначення інтервалу часу. Вхідний сигнал повинен бути сигналом на основі подій. Відкриття порту виконується у полі **Timeout interval from** на головній вкладці діалогового вікна.

Для виходу сутностей передбачений вихідний порт **OUT,** а також можна відкрити вихідний порт сигналів **#d** для виводу кількості сутностей, що пройшли через блок.

Діалогове вікно

На діалоговому вікні розташовано дві вкладки:

- **Schedule Timeout** – режими тайм-ауту;
- **Statistics** – статистика.

Розглянемо поля та їх значення вкладки **Schedule Timeout**:

- Timeout tag (Тег тайм-ауту) – ім'я, що зв'язує кожний об'єкт з подією тайм-аут.
- **Timeout interval from** (Часовий інтервал на основі) – визначає джерело надходження значення інтервалу часу очікування. Поле може приймати наступні значення:
  - **Dialog** –діалог, означає, що значення встановлюється у діалоговому вікні;
  - **To Signal port ti** – через вхідний сигнальний порт **ti;**
  - **Attribute** – на основі атрибутів.
- **Timeout interval** (Часовий інтервал) – визначає довжину проміжку часу між часом прибуття сутності та запланованою подією тайм-ауту, поле доступне, якщо встановлено **Timeout interval from to Dialog**;
- **Attribute name** (Ім'я атрибуту) – визначає ім'я атрибуту, значення якого використовується блоком як інтервал тайм-аутів. Це поле доступне тільки, якщо встановлено **Timeout interval from to Attribute**;
- **Timeout event priority** (Пріоритет подій тайм-ауту) – визначає пріоритет тайм-ауту подій, в порівнянні з іншими одночасними подіями у симуляції;
- **If timeout is already scheduled** (Якщо подія тайм-ауту вже у списку) – визначає поведінку блоку, якщо тайм-аут подія з вказаним тегом очікування вже запланована. Поле може приймати наступні значення:
  - **Error** – помилка;
  - **Warn and reshelule** – попередження та зміна списку;
  - **Reshedule** – зміна списку.

- **If entity has no destination when timeout occurs** (Якщо сутність не має призначення, коли відбувається тайм-аут) – визначає поведінку блоку, якщо сутність не має призначення, коли відбувається тайм-аут. Значення поля:

• **Error** – помилка;

• **Warn and discard the entity** – попередження та відхилення об’єкту.

Вікно **Statistics (**Статистики) (рис. 101) має лише одне поле **Number of entities departed** для визначення кількості подій, що вийшли з блоку через сигнальний порт **#d**. Початкове вихідне значення 0.

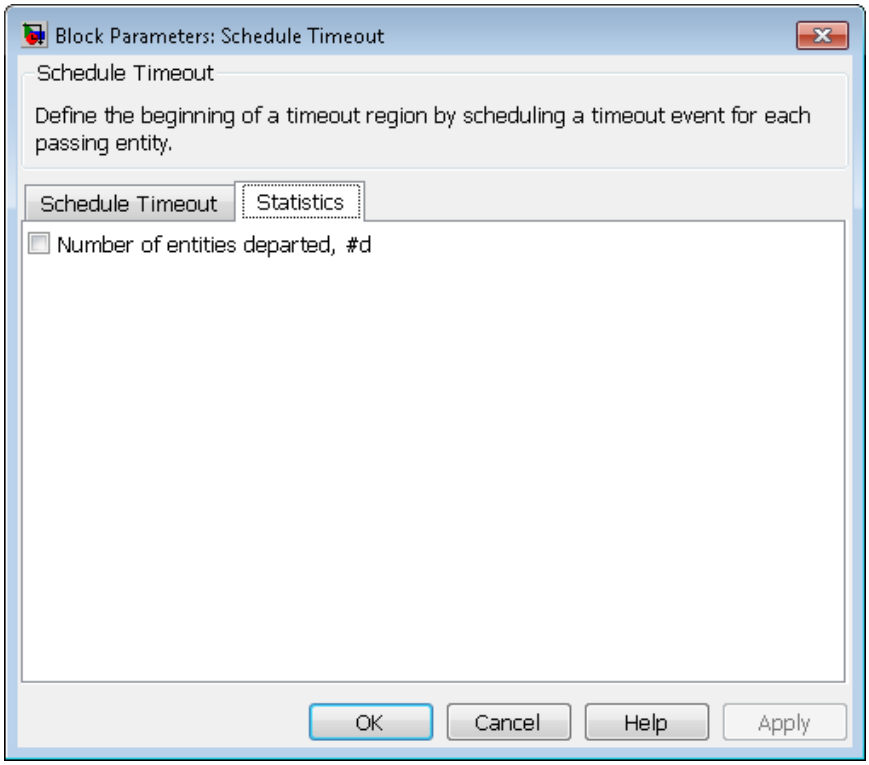


Рис. 101. Закладка статистики блоку Schedule Timeout

Приклад використання блоку Schedule Timeout наведений нижче в описі блоку Cancel Timeout

1.8.4. Cancel Timeout

Cancel Timeout (Відміна події тайм-аут) – відміняє подію Timeout (завершення часу) для кожного об'єкта, що попередньо була ініційована блоком Schedule Timeout (рис. 102). Цей блок позначає завершення часового обмеження для сутностей. Можливість скасування події Timeout перш, ніж вона відбудеться дозволяє продовжити життя сутності.

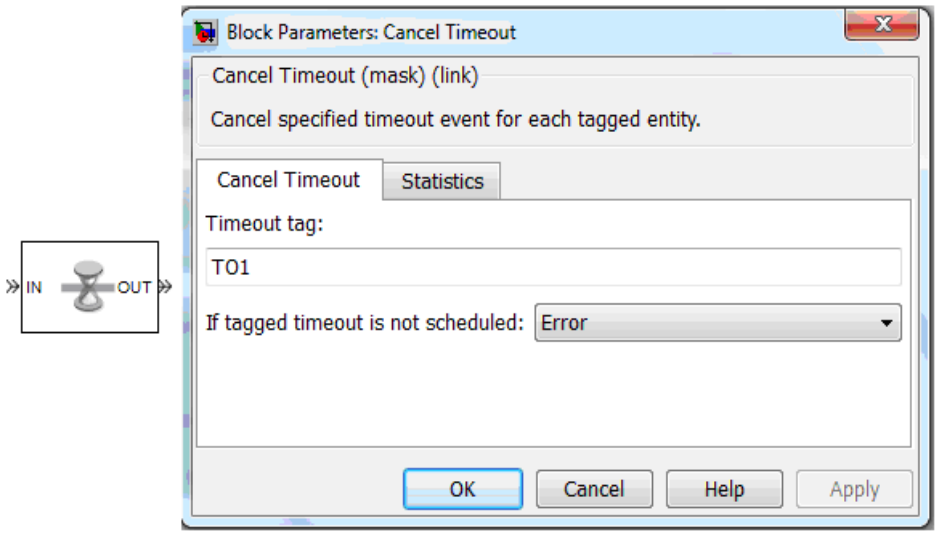


Рис. 102. Блок Cancel Timeout та його діалогове вікно

Блок Schedule Timeout має один вхідний порт **IN** для надходження сутностей.

Для виходу сутностей передбачений вихідний порт **OUT**, а також можна відкрити вихідні порти сигналів (табл. 13).

Таблиця 14. Сигнали вихідних портів

Назва	Опис	Порядок оновлення
#d	Кількість сутностей, які вийшли з цього блоку від початку моделювання	3
#t	Кількість сутностей, які вийшли з цього блоку і пов'язані з ім'ям події Timeout	2
rt	Час між час прибуття в цей блок і часом блоку Scheduled time пов’язаного з ім'ям події Timeout	2
w	*Average residual time* середнє значення серед всіх значень RT, які надійшли в цей блок під час моделювання і пов’язані з ім’ям події Timeout	1

Час оновлення всіх портів при включені на вкладці статистики – після виходу сутності.

Діалогове вікно

На діалоговому вікні розташовано дві вкладки:

- **Cancel Timeout** – скасування режиму тайм-ауту;
- **Statistics** – статистика.

На головній вкладці **Cancel Timeout** розташовані наступні поля:

- **Timeout tag** (Тег тайм-ауту) – задає ім’я події, що відповідає назві параметру **Timeout tag** блоку **Schedule Timeout** для якої відміняється тайм-аут. Якщо вхідна сутність не пов’язана з подією Timeout, то

можна налаштувати блок для отримання повідомлення про помилку, попередження або ігнорувати відсутність події Timeout.

- **If tagged timeout is not scheduled** (Об'єкт для події тайм-аут відсутній у списку) – визначає поведінку блоку, якщо прибуває сутність не пов'язана з подією Timeout.
  - **Error** – помилка;
  - **Warn and continue** – попередити і продовжувати;
  - **Continue** – продовжувати.

На вкладці **Statistics** (Статистика) розташовані наступні поля:

- **Number of entities departed** – виводить кількість сутностей, що пройшли через блок і контролює поведінку вихідного сигналу порту **#d;**

- **Number of entities departed with specified tag** – виводить кількість сутностей що пройшли через блок з режиму тайм-аут і контролює поведінку вихідного сигналу порту **#t;**

- **Residual time** – визначає проміжок часу між прибуттям сутності до блоку і часом існування сутності, який зазначений в блоці і дозволяє використовувати вихідний сигнал порту **rt;**

- **Average residual time** – визначає середнє значення часу перебування у блоці всіх сутностей і контролює поведінку вихідного сигналу порту **#w.**

Для всіх сигналів початкове вихідне значення, яке діє з початку моделювання до першого оновлення блока, дорівнює 0.

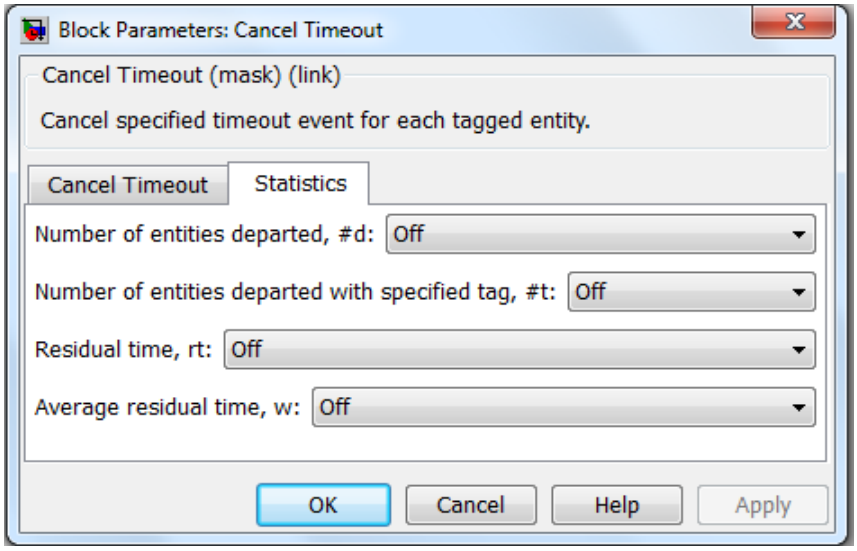


Рис. 103. Вкладка Statistics

Приклад. Використиння тайм-аутів для обмеження часу перебування сутностей у черзі.

Модель зображена на рис. 104 обмежує час існування кожної сутністі в черзі, але не обмежує час існування на сервері. Черга видаляє будь-яку сутність, яка перевищує ліміт часу. Наприклад, якщо кожна сутність є клієнтом, який намагається дозвонитись до оператора в центр підтримки викликів, то ця модель описує клієнта, який чекає відповіді оператора.

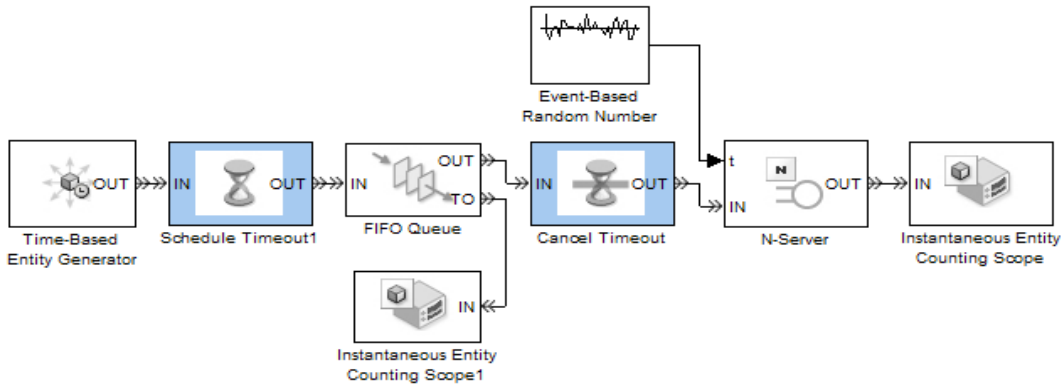


Рис. 104. Модель, що обмежує час існування кожної сутності в черзі

Для кожного клієнта, який прибув до блока Schedule Timeout, встановлюються часові обмеження. Наступним кроком для клієнта може бути:

Entity Times Out – якщо клієнт все ще перебуває в черзі, коли годинник досягає ліміт часу, клієнт кладе трубку телефону, не додзвонившись до оператора. В загальному випадку, сутність виходить з блоку FIFO Queue через порт TO і не доходить до сервера.

Entity Advances to Server – якщо клієнт виходить з черги, перш ніж годинник досягає ліміту часу, клієнт не кладе трубку телефону і починає говорити з оператором. В загалі, якщо сутність надходить до блоку Cancel Timeout, перш ніж годинник досягає ліміту часу, з сутності знімається її часове обмеження. Сутність переходить до сервера.

1.9. Блоки перемикачів та управління потоками (Routing)

Блоки бібліотеки Routing дозволяють об’єднувати або розподіляти потоки сутностей (рис. 105).

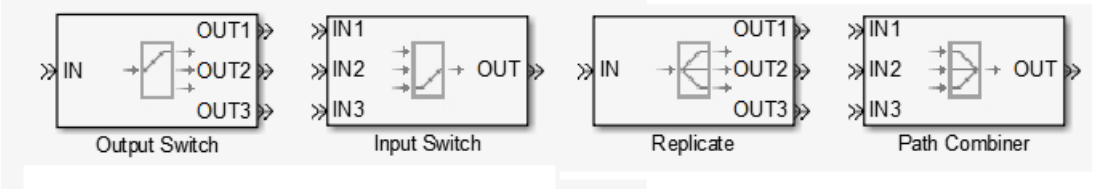


Рис. 105. Блоки бібліотеки Routing

1.9.1. Output Switch

Output Switch (Перемикач виходів) – блок організовує вихід сутностей, які на нього надходять через декілька вихідних портів. Управління вихідними портами може змінюватися під час моделювання.

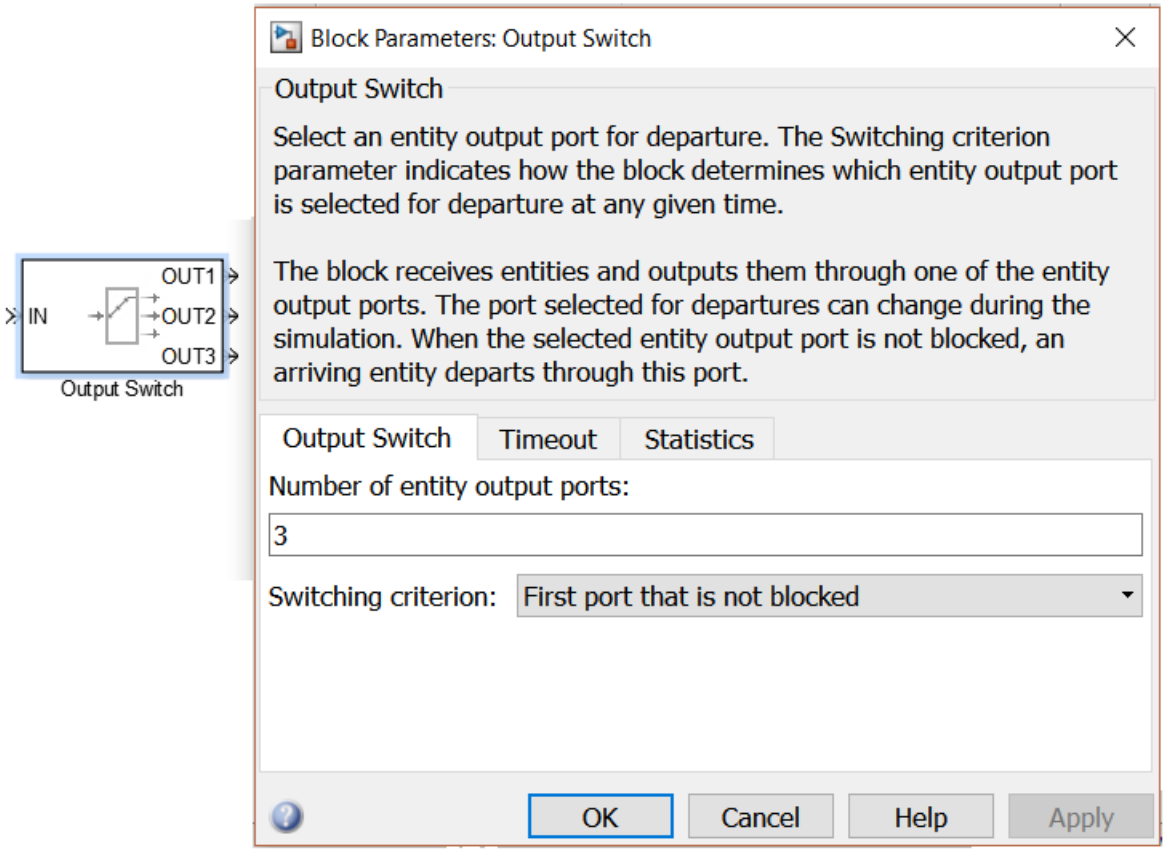


Рис. 106. Блок Output Switch та його діалогове вікно

Блок має один вхідний порт **IN** для надходження сутностей. Додатково можна відкрити сигнальний порт **p**, який визначає індекс вихідного порту сутності. Значення мають бути цілими числами від 1 до числа вихідних портів. Цей сигнал повинен бути сигналом на основі події. Цей порт доступний тільки в тому випадку, якщо критерій перемикання (Switching criterion) встановлено на From signal port p.

Вихідних портів **OUT** може бути декілька. Їх кількість визначається користувачем. Також для сутностей з обмеженим часом можна зробити доступним порт **TO**. Порт стає доступним лише коли поле **Switching criterion** головної вкладки встановлене як **From signal port p** і на другій вкладці **Timeout** (тайм-аут) встановлено режим **Enable TO port for timed-out entities**.

Відповідними налаштуваннями вкладки **Statistics** (Статистика) можна активізувати сигнальні порти (табл. 14).

Таблиця 15. Вхідні порти блоку Output Switch

Назва	Опис	Час оновлення при виборі статистика	Порядок
#d	Кількість сутностей, які вийшли з блоку, починаючи з початку моделювання	Після виходу сутностей через порт, крім TO	3
#	Кількість сутностей з	Після виходу	2

to	режимом тайм-аут, які вийшли з блоку, починаючи з початку моделювання	сутностей через порт, крім TO	
e	Значення 1 означає, що блок зберігає об'єкт, який прибув і не зміг відійти. У такому випадку сутність є незавершеною. Значення 0 означає, що блок не зберігає будь-які очікувані об'єкти.	Відлік часу починається з 1 і відбувається після того, як блок зберігає об'єкт, який намагався і не зміг відійти. Відлік часу починається з 0, після виходу незавершеного об'єкта через будь-який порт	1
last	Індекс вихідного порту, через який вийшов останній об'єкт, за винятком тих, що мають обмежений час. Значення цього сигналу - 1, 2, 3, ..., Кількість вихідних портів.	Після виходу сутностей через порт, крім TO	2

Початкове вихідне значення, яке діє від початку моделювання до першого оновлення блоку, становить 0 для всіх сигналів.

Діалогове вікно блоку містить три вкладки:

- **Output Switch** – управління перемиканням;
- **Timeout** – управління подіями таймоут;
- **Statistics** – статистика.

Головна вкладка **Output Switch** містить два поля:

- **Number of entity output ports** (Кількість вихідних портів) – ціле число, яке задає кількість вихідних портів;
- **Switching criterion** (Критерій перемикання) – визначає правило, через який порт сутність буде виходити з блоку. Можливі значення поля наведені у табл. 15. Для обраного правила необхідно задати відповідні параметри, які з'являються на вкладці після встановлення режиму.

Таблиця 16. Критерії перемикання

Критерій	Опис
Round robin круговий критерій	Перемикання портів відбувається циклічно послідовно. Перша сутність виходить через порт OUT1, наступна – OUT2 і т. д., поки не вичерпаються всі порти. Після цього блок повертаєься до порту OUT1.

Equiprobable рівноможливий	Блок випадковим чином вибирає вихідний порт сутностей, через який виходить черговий об'єкт. Всі вихідні порти є рівноможливими. Параметр **Initial seed** (вхідна сутність) ініціалізує процес генерації випадкових чисел.
First port that is not blocked перший порт, який не заблоковано	Виведення сутностей починається з порту OUT1. Якщо цей порт заблоковано, тоді блок намагається вивести об'єкт через OUT2 і так далі. Якщо усі вихідні порти заблоковані, то вхідний порт IN блоку стає недоступним.
From signal port p Через сигнальний порт p	Створюється додатковий порт виведення сигналу, позначений як p. Сигнал на цьому порту приймає цілі значення від 1 до кількості вихідних портів. Блок контролює значення сигналу p протягом всього моделювання та реагує на зміни, вибравши відповідний вихідний порт.
From attribute Згідно до атрибуту (пріоритету)	Сутність виводиться через вихідний порт, що відповідає значенню заданого атрибута. Атрибут задається за іменем. Значення атрибута повинно бути цілим числом від 1 до значення "Кількість вихідних портів". Якщо вихідний порт зазначеного об'єкта заблоковано, цей блок не приймає сутність, доки вихідний порт не буде розблоковано.

Примітка. Якщо встановлюється критерій переключення на потік p, то блок пропонує кілька варіантів, які допоможуть забезпечити актуальність та вірність сигналу при його використанні для визначення способу обробки прибуваючого об'єкта. Необхідно бути особливо обережним, коли сигнал знаходиться в зворотньому зв'язку, або коли сигнал може змінюватися одночасно із надходженням сутності.

Наступна вкладка діалогового вікна **Timeout** блоку Output Switch містить лише одне поле **Enable TO port for timed-out entities** (Активізувати порт **TO** для виділених об'єктів) (рис. 107). Параметр доступний, лише при встановленні критерію перемикання на потік з сигналу p та вибору об'єкту Store.

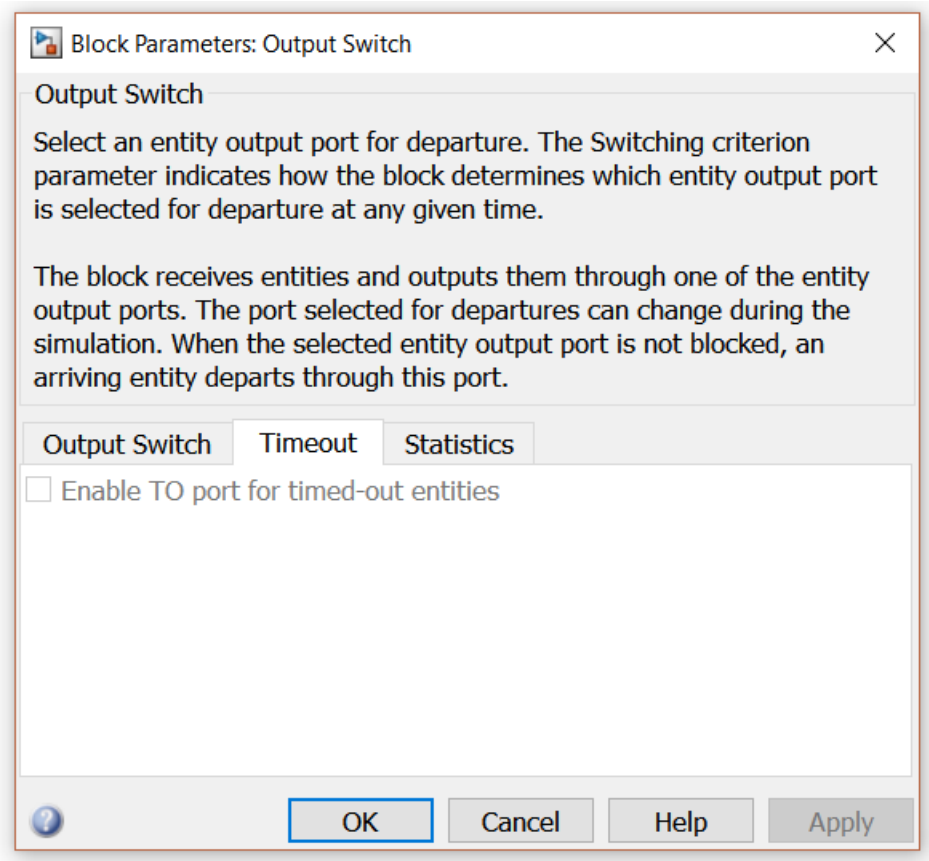


Рис. 107. Закладка статистики блоку Output Switch

Опція стає доступною, якщо час перебування об'єкта в цьому блоці вичерпано.

На останній вкладці Statistics передбачені наступні поля (рис. 108):

- **Number of entities departed** (Кількість вихідних сутностей) – дозволяє використовувати вихідний порт сигналу з позначкою #d;(дозволяє використовувати вихідний порт сигналу із позначкою #to;
- **Pending entity present in block, pe** (Очікуючі сутності у блоці, pe) – дозволяє використовувати вихідний порт сигналу з позначкою pe. Порт доступний лише, якщо встановлено критерій перемикання **Switching criterion** на From signal port p;
- **Last entity departure port** (Останній вихідний порт) – дозволяє використовувати вихідний порт сигналу, позначений як останній.

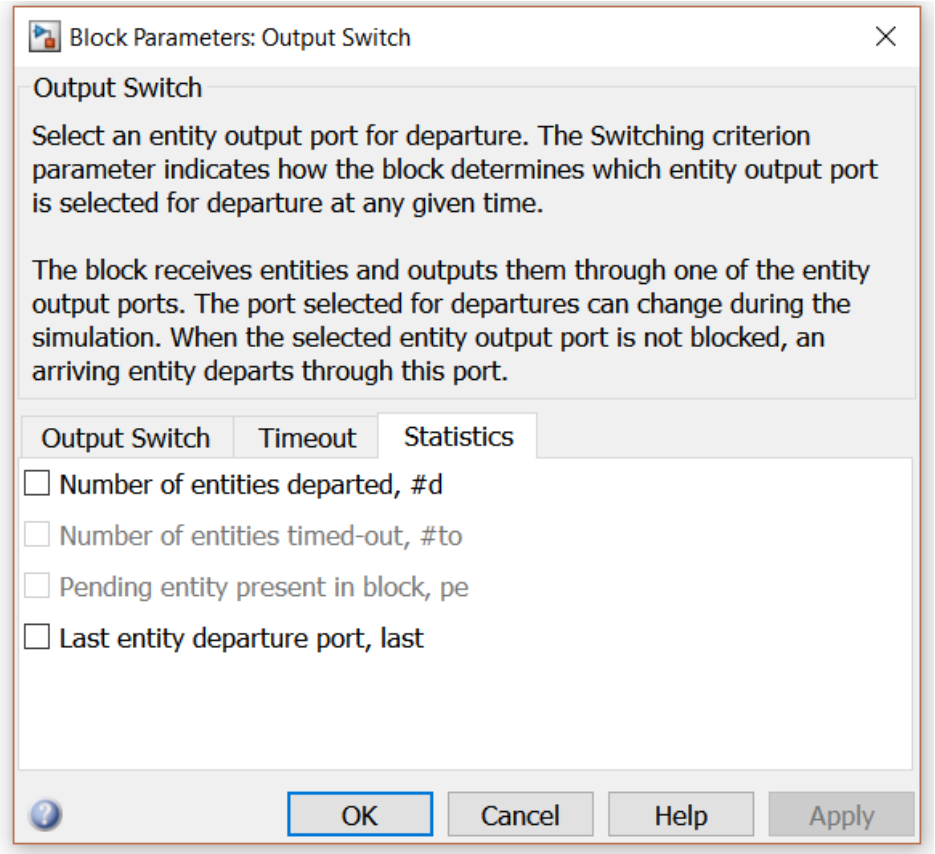


Рис. 108. Закладка статистики діалогового вікна блоку Output Switch

Приклад. Вибір першого доступного сервера. У цьому прикладі об'єкти, що надходять на блок вихідного комутатора, виходять через перший вихідний порт, який не заблоковано, якщо принаймні один такий вихідний порт існує. Прикладом може бути одна черга людей, які чекають на обслуговування одним з декількох банківських касирів, представників колл-центрів тощо. Кожна людина в черзі хоче якнайшвидше перейти до першого доступного постачальника послуг. Реалізувати цей підхід можна, встановивши параметр критерію перемикання у блоці Output Switch на **First port that is not blocked** (Перший порт, який не заблоковано).

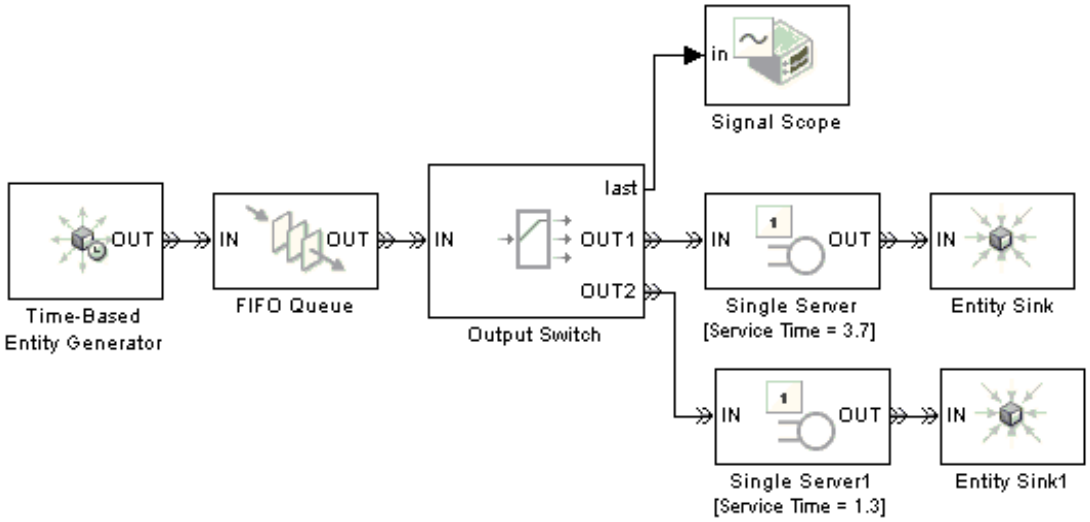


Рис. 109. Модель з вибором першого доступного сервера

Ця детермінована модель створює одну сутність кожну секунду і намагається провести її до одного з двох серверів. Ці два сервери мають різний час обслуговування, обидва більше 1 секунди. Сервер з більшим часом обслуговування стає доступним рідше і має меншу пропускну здатність. Блок FIFO Queue зберігає об'єкти, коли обидва сервери зайняті. Після того, як будь-який сервер стає доступним, об'єкт у черзі переходить до блоку Output Switch, який виводить цей об'єкт на відповідний сервер. Блок Output Switch також виводить сигнал, що містить індекс вихідного порту об'єкта, через який відбувся вихід останнього об'єкта. Блок Signal Scope вказує значення цього сигналу. Можна побачити, що порівняно з першим сервером, другий сервер обробляє більше об'єктів, оскільки його час служби коротший (рис.110).

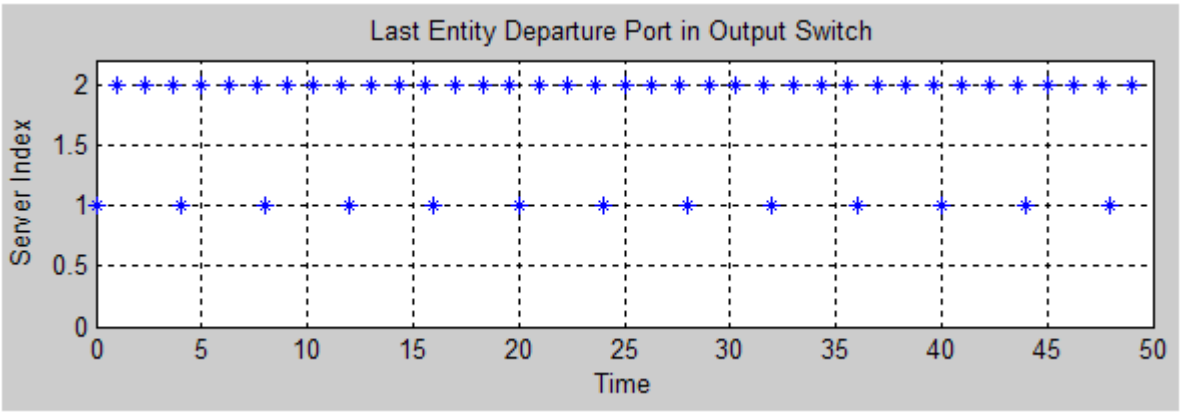


Рис. 110. Графік індексів серверів в процесі роботи

Приклад. Використання атрибута для вибору вихідного порту.

Розглянемо ситуацію, коли посилки сортуються між кількома транспортними засобами доставки залежно від розташування вказаних одержувачів. Якщо кожна посилка є об'єктом, то можна кожний об'єкт наділити маркерами (даними), щоб вказати місце розташування його одержувача. Для реалізації сортування необхідно встановити параметр критерію перемикання в

блоці Output Switch в атрибут From. Наведена нижче модель ілюструє процес сортування (але не самий процес доставки), на три географічні зони. Генератор представляє джерело посилок, адресованих одній з зон за допомогою встановлення відповідного атрибуту. Далі посилки переходять до черги. Блок Single Server моделює малу затримку, що виникає в процесі сортування, і відправляє кожну посилку через блок виводу до одного з трьох вихідних портів. Приклад розглядає відсортовані об'єкти, призначені для кожної зони.

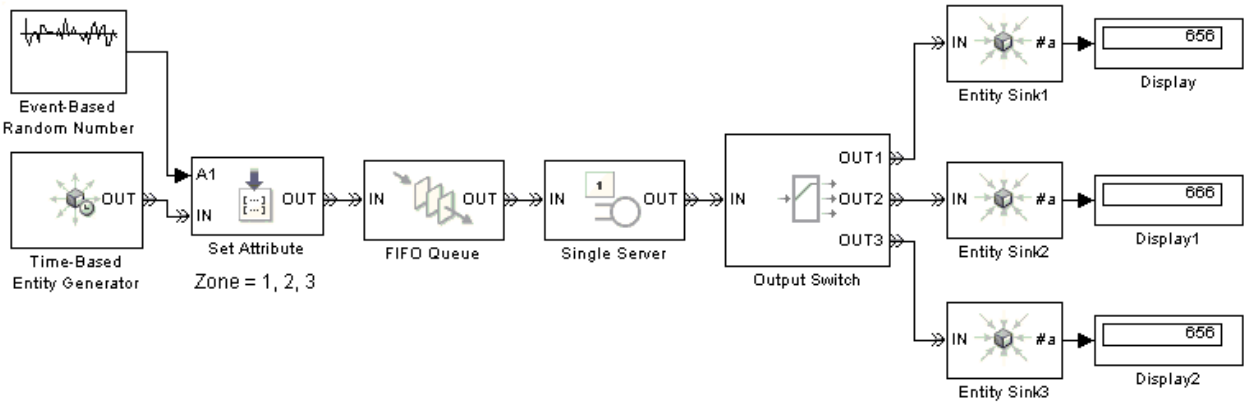


Рис. 111. Модель сортування сутностей на три вихідні потоки

1.9.2. Input Switch

Input Switch (Перемикач входів) – блок визначає вхідний порт для потенційного надходження сутностей. Вибраний порт входу об'єкта під час моделювання може змінюватися. Коли вибирається один вхідний порт, інший стає недоступним (рис. 112).

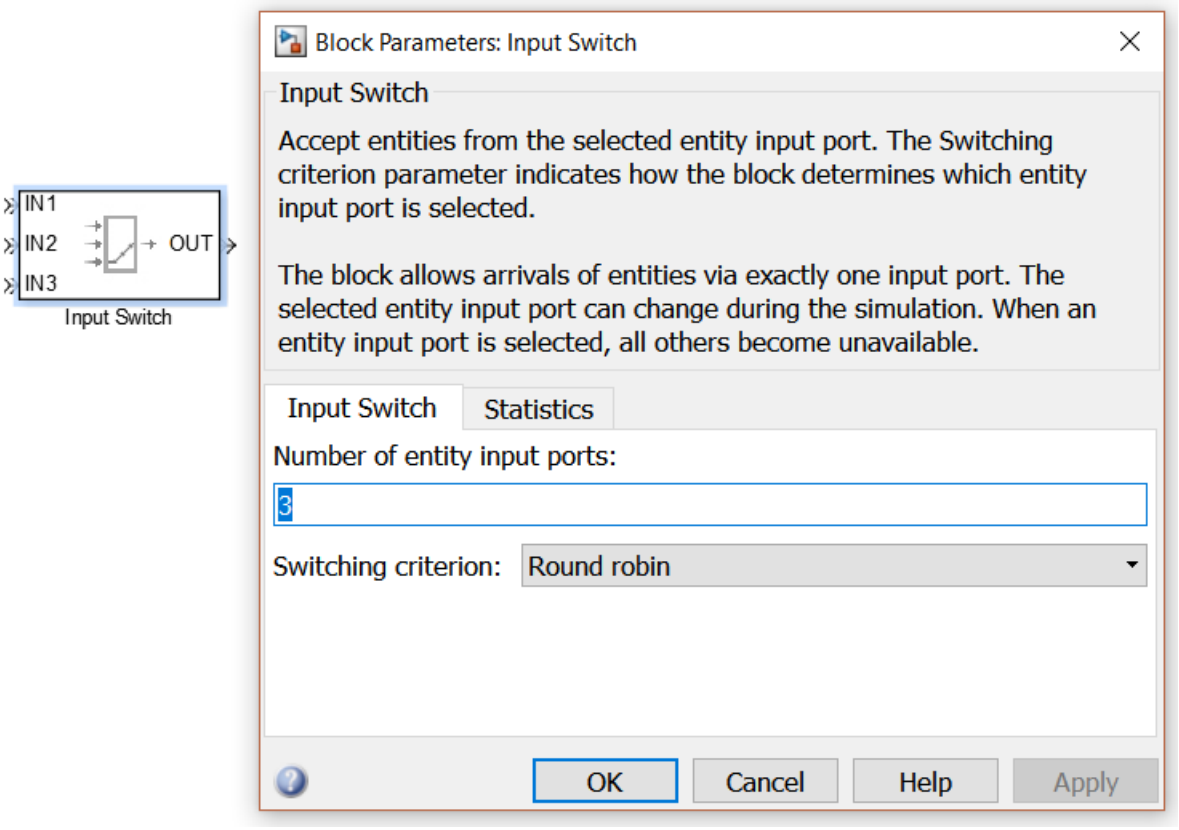


Рис. 112. Блок Input Switch та його діалогове вікно

Блок має наступні вхідні порти:

- **IN1, IN2, IN3**, і т.д. – вхідні порти. У будь-який момент часу вибирається один вхідний порт, а інші недоступні. Параметр **Number of entity input ports** (Кількість вхідних портів) вхідних даних визначає, скільки з цих елементів вхідних портів має блок;
- **P –** вхідний сигнальний порт. Індекс доступного порта вводу об'єкта. Цей сигнал повинен бути сигналом на основі події. Порт доступний тільки в тому випадку, якщо ви встановите критерій перемикання **Switching criterion** на From signal port p.

Вихідні порти:

- **OUT –** основний вихідний порт сутностей;
- **#d** – виведення кількості вихідних сутностей від почаку симуляції, пріоритет оновлення порту дорівнює 2;
- **last** – відображення індексу вхідного порту, який був доступний в останній раз, після виходу об'єкту. Початкове значення – 0, пріоритет оновлення порту дорівнює 1.

На головній вкладці діалогового вікна розташовано два поля:

- - **Number of entity input ports** (Кількість вхідних портів) – ціле число, яке задає кількість вхідних портів;
- **Switching criterion** (Критерій перемикання) – визначає правило, за яким визначається порт, через який сутність буде виходити з блоку. Правила, які використовує блок для вибору вхідного порту об'єкта, перелічені у таблиці 15, наведеній вище для блоку Output Switch.

Вкладка статистики **Statistics** дозволяє налаштовувати сигнальні вихідні порти (рис. 113):

- **Number of entities departed** (Кількість вихідних сутностей) – дозволяє використовувати вихідний порт сигналу з позначкою #d.
- **Last entity arrival port** (Останній вхідний порт) – дозволяє використовувати останній вихідний порт сигналу.

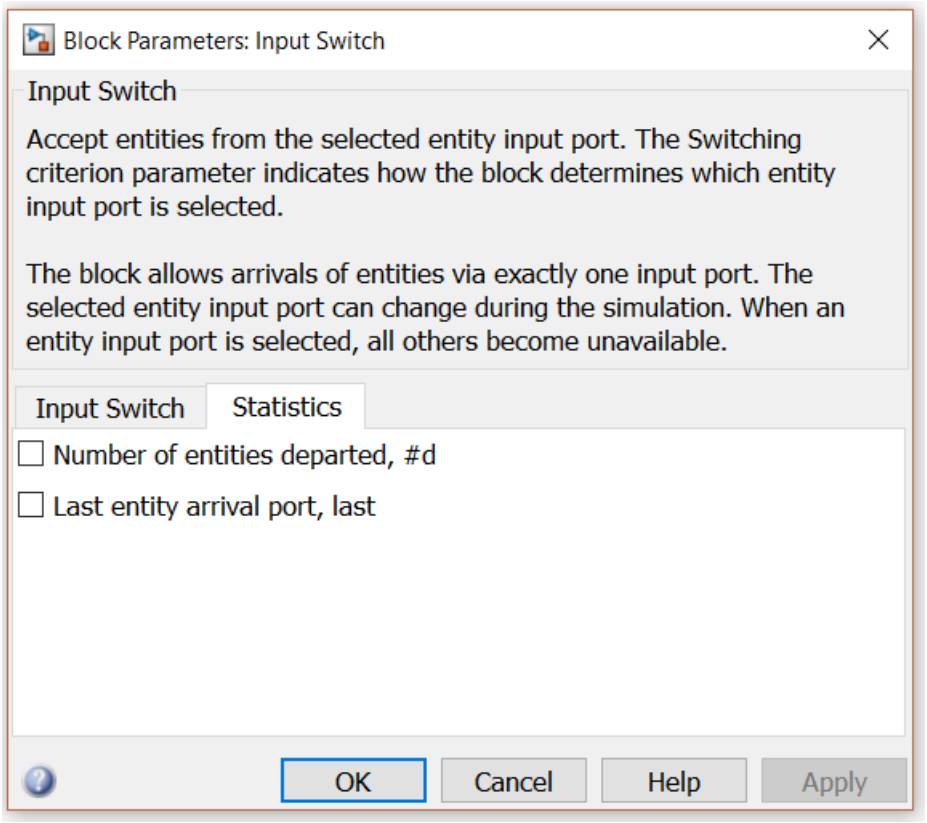


Рис. 113. Вкладка Statistics блоку Input Switch

Приклад. Використання блоку Input Switch.

На рис. 114 наведений фрагмент моделі, у якій об’єкти чекають у двох чергах перш ніж потрапити до серверу. Таймер-блоки вимірюють час, який об’єкти проводять у відповідній черзі. Два Start Timer блоки керуються таймером з однією і тією ж назвою, забезпечуючи наявність для всіх об’єктів таймерів, незалежно від їх маршруту перед досягненням серверу. Блок **Input Switch** управляє процесом надходження замовлень до серверу, об’єднуючи обидва потоки у єдиний потік.

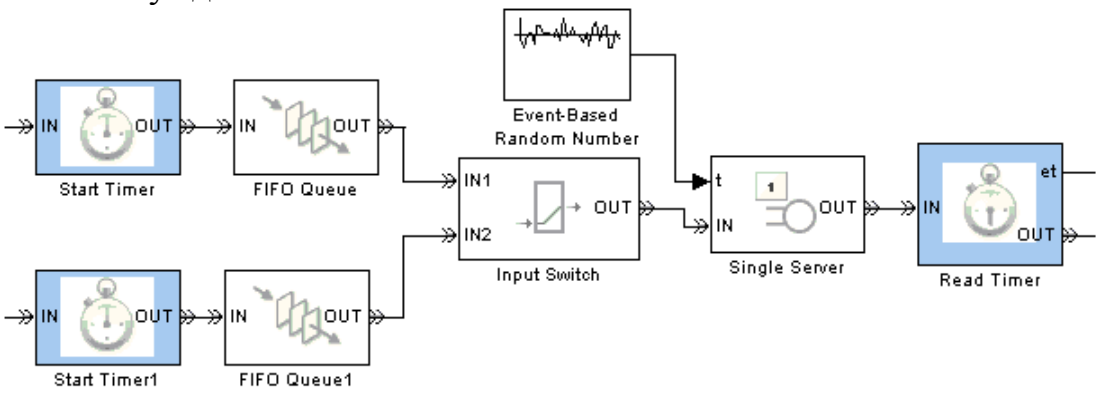


Рис. 114. Приклад використання блоку Input Switch

1.9.3. Path Combiner (Комбінатор шляхів)

Path Combiner – блок приймає об'єкти через будь-який вхідний порт та виводить їх через єдиний вихідний порт. Кількість вхідних портів визначається

параметром **Number of entity input ports** (Кількість елементів вхідних портів), який визначається на головній вкладці діалогового вікна. До блоку може одночасно надійти декілька сутностей, які будуть виводитись по одній згідно послідовності подій, які передують блоку Path Combiner.

Кілька екземплярів об’єктів одного типу, але з різними атрибутами, можуть надходити до блоку Path Combiner. У таких ситуаціях тип скомпільованого об’єкта відображає тип профайлу (блок не змінює атрибути).

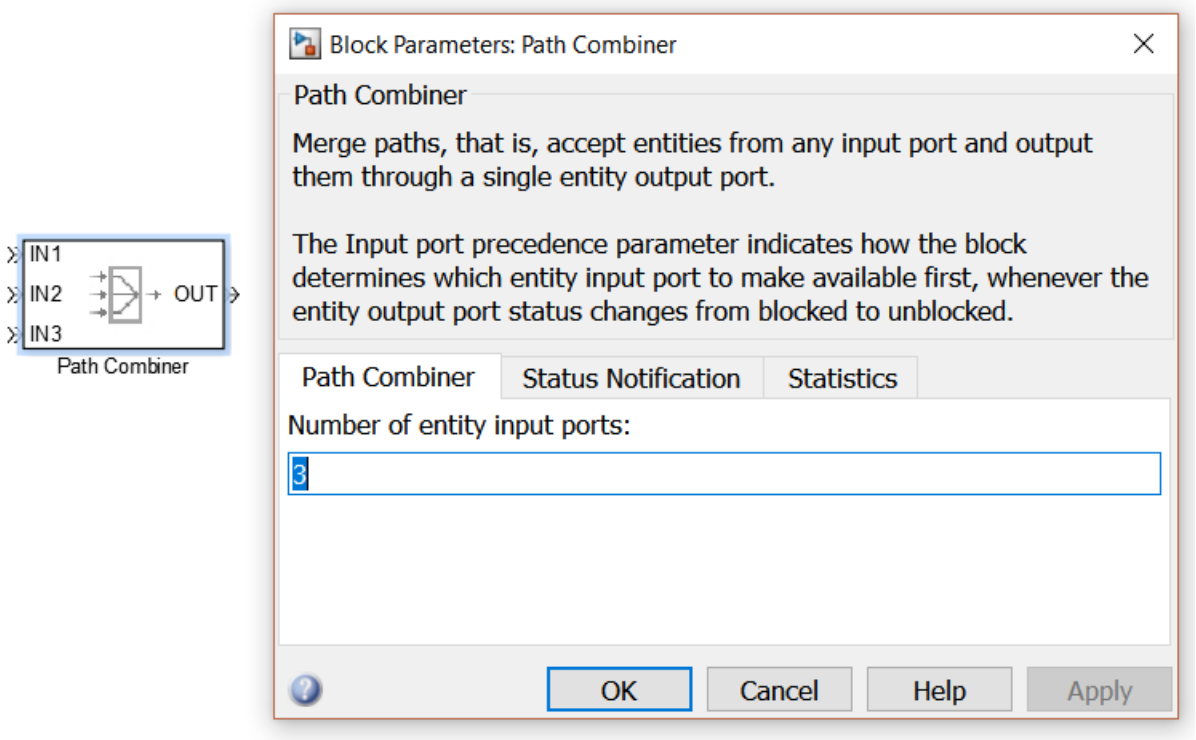


Рис. 115. Блок Path Combiner та його діалогове вікно

На другій вкладці Status Notification також одне поле:

Input port precedence (Передача вхідного порту) – порядок доступу відповідного вхідного порту до вихідного. Значення поля наведені у таблиці.

Приклад.

Розглянемо декілька сценаріїв поєднання шляхів:

- блоки генераторів створюють об'єкти, що мають різні значення для певного атрибуту, зливаються в один потік, але далі розглядаються по-різному, відповідно до значень атрибутів (рис. 116).

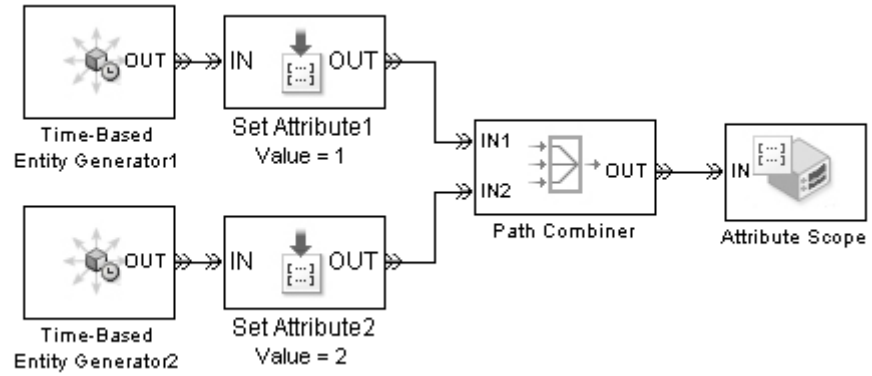


Рис. 116. Обєднання потоків

- кілька черг об'єднуються в одну чергу.

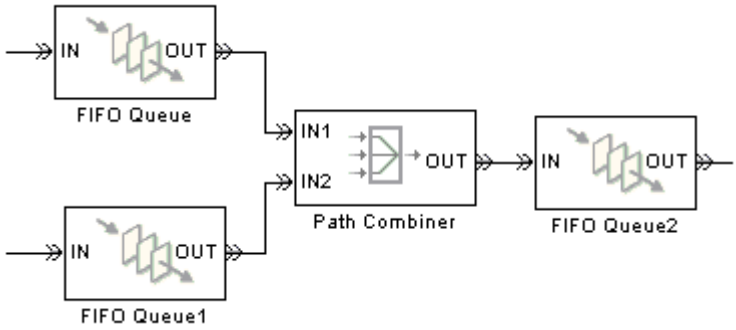


Рис. 117. Обєднання черг

- зворотній потік входить в ту ж чергу, що й звичайний потік.

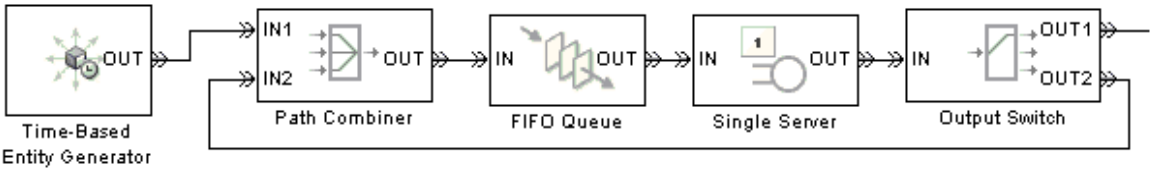


Рис. 118. Об'єднання вхідного і зворотнього потоків

Replicate

Replicate – блок виводить копію (реплікацію) об'єкта, що надходить, через вихідний порт. Кількість копій задається параметром **Number of entity output ports parameter** (Кількість вихідних портів), який знаходиться на головній вкладці діалогового вікна. Якщо для об'єкта встановлені обмеження за часом (таймаут), то всі копії мають однаковий термін завершення. Логічно, блок скасовує подію тайм-ауту для сутності, що надходить та розраховує нові події тайм-аут для об'єктів, що виходять з блоку. Реплікація об'єкта може бути повною або частковою.

Блок має один вхідний порт сутностей **IN**. Кількість вихідних портів для сутностей **OUT1, OUT2, OUT3** визначається користувачем. Також для блоку є доступними два сигнальні порти:

- #a – кількість сутностей, які прибули до блоку з моменту початку моделювання (пріоритет 1);

- **#d** – кількість сутностей, які вийшли з блоку з моменту початку моделювання (пріоритет 2).

Початкове значення для портів за замовчуванням дорівнює 0.

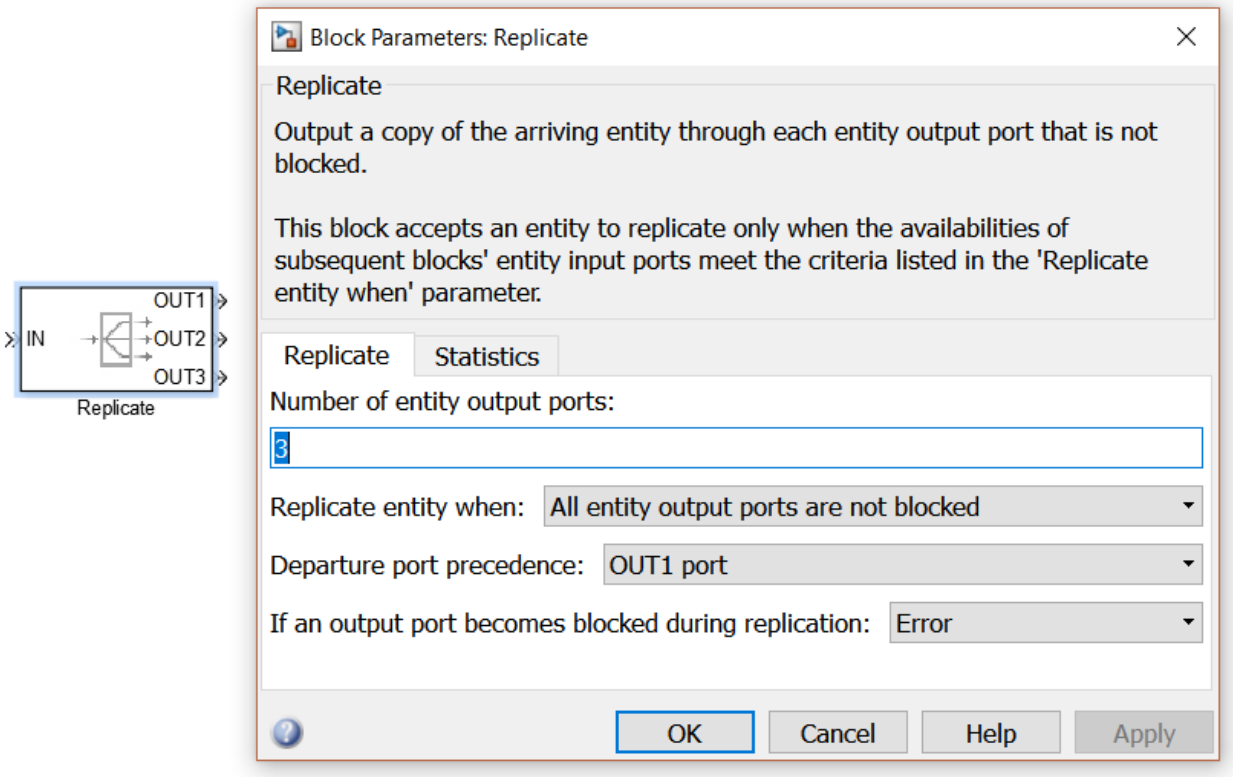


Рис. 119. Блок Replicate та його діалогове вікно

Поля головної вкладки Replicate діалогового вікна дозволяють налаштовувати процедуру копіювання:

- **Number of entity output ports** – визначає кількість копій;
- **Replicate entity when** – визначає доступність блоку для прибуваючих сутностей, коли принаймні один вихідний порт не блокується, або лише тоді, коли всі вихідні порти не заблоковані. Поле може приймати два значення:

• All entity output ports are not blocked – блок приймає сутності для реплікації тільки тоді, коли всі вихідні порти підключені до доступних портів наступних блоків;

• Any entity output port is not blocked – блок приймає сутності для реплікації, коли хоча б один вихідний порт підключений до доступного порту наступного блоку.

- **Departure port precedence –** визначає початок послідовності, в якій блок виводить копії. Значення поля наведені у табл. 16.

-

Таблиця 17.

Значення	Опис	Приклад
OUT1 port	копії відходять через вихідні порти OUT1, OUT2,	послідовність виходу завжди OUT1, OUT2, OUT3, ...

	OUT3, ... у зазначеній послідовності	протягом моделювання.
Round robin	перша копія відправляється через порт, що мав перевагу в останньому випадку. Інші копії відправляються по черзі через наступні порти.	першого разу, копії виходять у послідовності OUT1, OUT2, OUT3. Другий раз – OUT2, OUT3, OUT1. Третій раз –OUT3, OUT1, OUT2. Четвертий раз аналогічний першому і т. д.
Equiprobable	перша копія відправляється через випадково виділений вихідний порт об'єкта. Інші копії відправляються по черзі через наступні порти	Для блоку з 4 виходами, якщо випадкове число дорівнює 3, копії виходять у послідовності OUT3, OUT4, OUT1, OUT2. Якщо – 2, то у послідовності OUT2, OUT3, OUT4, OUT1.

- **Initial seed** – невід'ємне ціле число, яке ініціалізує генератор випадкових чисел. Поле доступне лише при встановленні значення Equiprobable.
- **If an output port becomes blocked during replication –** визначає наявність виданих блоком повідомлень, коли копія не може вийти, тому що вихідний порт блокується в процесі реплікації. Поле доступне, лише якщо встановлено **All entity output ports are not blocked.**

Наступна вкладка статистики Statistics (рис. 120) містить два поля, які визначають доступ до вихідних сигнальних портів (табл.16).

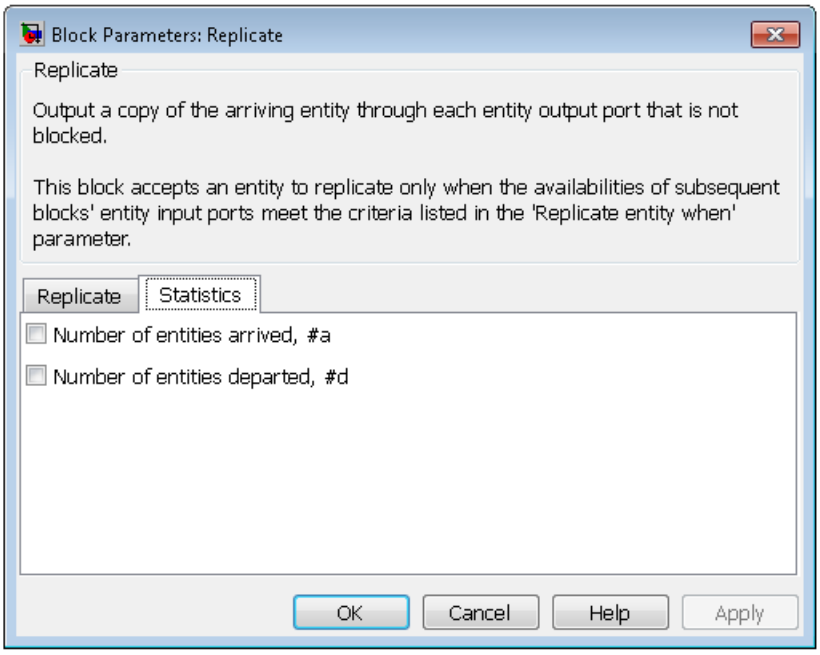


Рис.121. Вкладка статистики блоку Replicate

Приклад.

В прикладі, порівнюються дисципліни FIFO та LIFO в системі типу D / D / 1 з вхідним потоком через детерміновані проміжки часу- 0,3 та часом обслуговування - 1. Подія, що генерується, дублюється блоком Replicate, в результаті на черги різного типу подаються однакові події

.

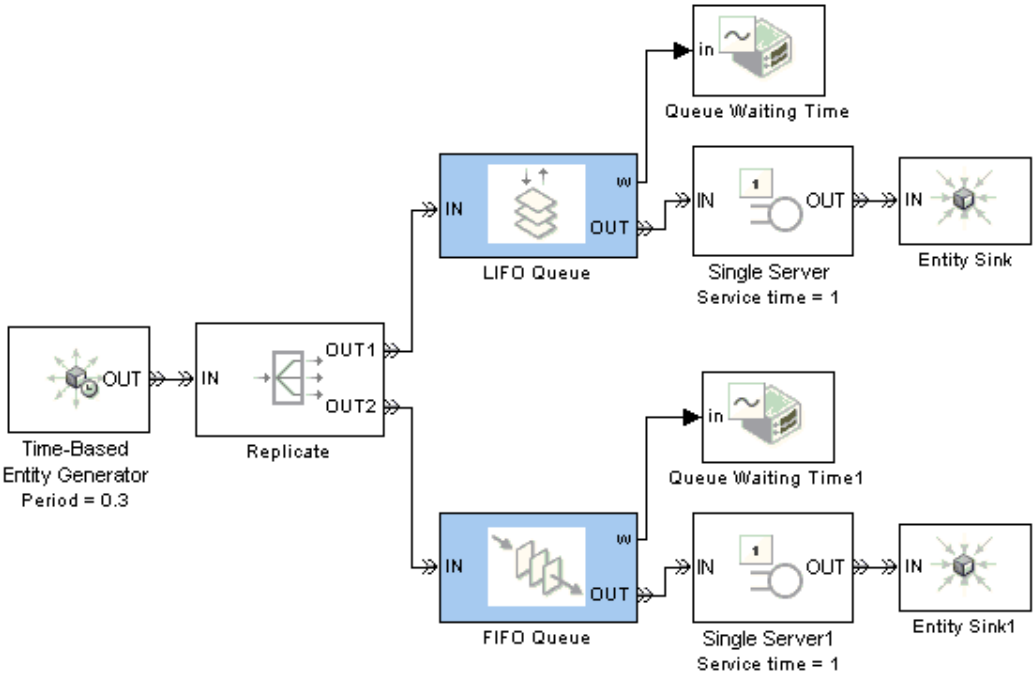


Рис. 122. Модель із застосуванням блоку Replicate для демонстрації різниці між різними типами черги

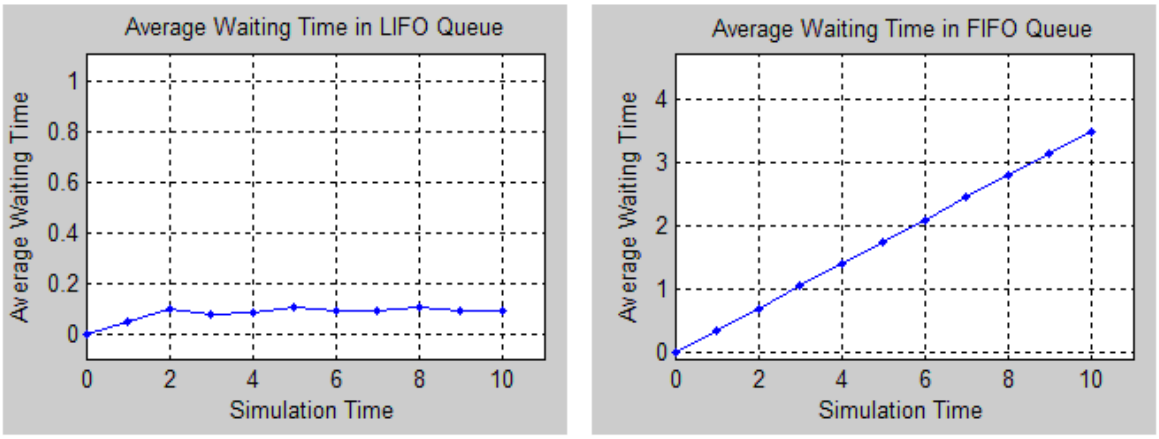


Рис. 123. Результати виконання моделі (рис.122)

Ports & Subsystems

За допомогою блоків портів та підсистем можна створювати підсистеми (підпрограми) для раціонального використання бланку моделі і створення моделі загальної системи як сукупності окремих підсистем. Підсистема складається з тих самих блоків, що і проста модель СМО. Блок підсистеми може представляти віртуальну (**Subsystem**) або невіртуальну (монолітну)

підсистему (**Atomic Subsystem**). Відмінність цих видів підсистем полягає в порядку виконання блоків під час розрахунку.

Монолітна підсистема вважається єдиним (неподільним) блоком і Simulink виконує розрахунок всіх блоків в такій підсистемі, не перемикаючись на розрахунки інших блоків в основній моделі. Невіртуальні підсистеми виконуються як окремі одиниці (атомне виконання). Можна створити умовно виконану невіртуальну підсистему, яка виконується лише тоді, коли відбувається перехід за ініціативою, викликом функції, дією або ввімкненим входом.

Якщо підсистема є віртуальною, то Simulink ігнорує наявність границь, які відокремлюють таку підсистему від моделі при визначенні порядку розрахунку блоків. Іншими словами в віртуальній системі спочатку можуть бути розраховані вихідні сигнали декількох блоків, потім виконаний розрахунок блоків в основній моделі, а потім знову виконується розрахунок блоків, що входять в підсистему. Щоб визначити, що підсистема є віртуальною, використовується функція **get_param** для параметра логічного блоку **IsSubsystemVirtual**.

Підсистеми також можуть бути керованими або некерованими. Керовані підсистеми завжди є монолітними. Керовані підсистеми мають додаткові (керуючі) входи, на які надходять сигнали, які активізують дану підсистему. Керуючі входи розташовані зверху або знизу підсистеми. Коли керована підсистема активізована – вона виконує обчислення. У тому випадку, якщо керована підсистема пасивна, то вона не виконує обчислення, а значення сигналів на її виходах визначаються налаштуванням вихідних портів.

Для створення в моделі підсистеми можна скористатися двома способами:

1. Скопіювати блок підсистеми Atomic Subsystem з бібліотеки в модель (рис. 124). Потім, відкривши його, додати необхідні блоки до підсистеми і зберегти їх в блоці підсистеми.

2. Виділити за допомогою миші потрібний фрагмент моделі і виконати команду Create Subsystem з меню Edit вікна моделі. Виділений фрагмент буде поміщений в підсистему, а входи і виходи підсистеми будуть забезпечені відповідними портами. Даний спосіб дозволяє створити віртуальну некеровану підсистему. Надалі, якщо це необхідно, можна зробити підсистему монолітною, змінивши її параметри, або керованою, додавши керуючий елемент. Скасувати угруповання блоків в підсистему можна командою Undo.

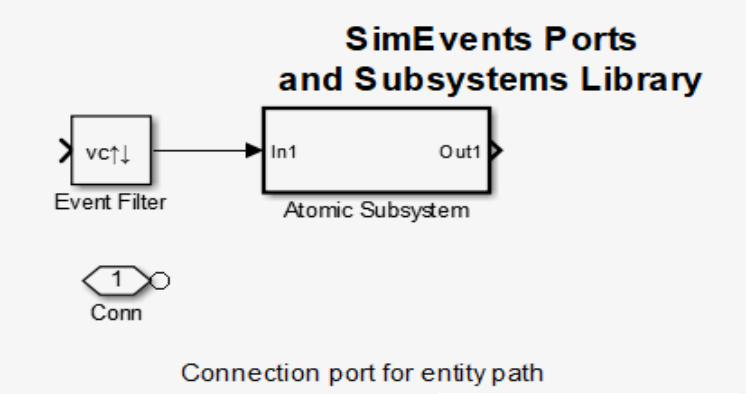


Рис. 124. Блоки бібліотеки Ports & Subsystems

Atomic Subsystem

Atomic Subsystem – блок для створення підсистеми. Обмін даними між підсистемою і моделлю виконується через вхідні (In) та вихідні (Out) порти.

Діалогове вікно блоку Atomic Subsystem має дві закладки. Головна закладка (рис. 125) містить наступні поля:

- Show port labels (Показувати мітки портів) – управляє відображенням міток для портів підсистеми на піктограмі підсистеми. Поле може приймати наступні значення:

- **none** – не відображати мітки портів підсистеми (значення встановлюється за замовчуванням);
- **FromPortIcon** – відображати назви сигналу або порту блоку;
- **FromPortBlockName –** відображати ім'я відповідного блоку портів у блоці підсистеми;
- **SignalName** – якщо ім'я існує, відобразиться назва сигналу, підключеного до порту в блоці підсистеми; інакше – ім'я відповідного блоку портів.

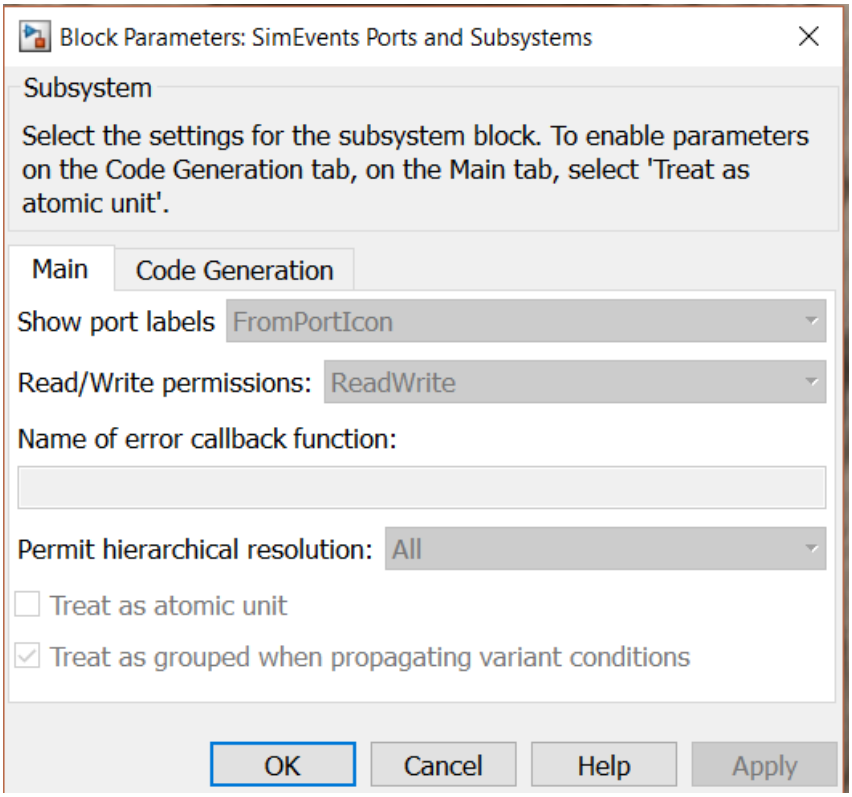


Рис. 125. Головна закладка діалогового вікна блоку Atomic Subsystem

- **Read/Write permission** (Читання/запис) – контролює доступ користувачів до вмісту підсистеми. Поле може приймати наступні значення:
  - **ReadWrite** – дозволяє відкривати та змінювати вміст підсистеми;
  - **ReadOnly** – дозволяє відкривати, але не модифікувати підсистему. Якщо підсистема знаходиться в бібліотеці блоків, можна створювати та відкривати посилання на підсистему, а також створювати та модифікувати локальні копії підсистеми, але не можна змінювати вміст вихідного екземпляру бібліотеки.
  - **NoReadOrWrite** – відключає відкриття або модифікацію підсистеми. Якщо підсистема знаходиться в бібліотеці, можна створити посилання на підсистему в моделі, але не можна відкривати, змінювати, або створювати локальні копії підсистеми.
- **Name of error callback function** – в полі задається ім'я функції, яку потрібно викликати, якщо під час виконання підсистеми програмного забезпечення Simulink виникає помилка. Simulink передає два аргумента функції: вказівник та рядок, який визначає тип помилки. Якщо не вказано жодної функції і при виконанні підсистеми виникає помилка, відображається загальне повідомлення про помилку.
- **Permit hierarchical resolution** – визначає, чи можна використовувати імена змінних робочої області, на які посилається підсистема. Поле може приймати наступні значення:
  - **All** – дає доступ до всіх імен змінних робочої області, що використовуються цією підсистемою, включаючи ті, які використовуються для

значень параметрів блоку та об'єктів даних Simulink (наприклад, об'єктів Simulink Signal);

- **ExplicitOnly** – дає доступ лише до змінних робочої області, які використовуються для значень параметрів блоку пам'яті даних (де немає блоку), сигналів та станів, позначені як "must resolve";
- **None** – забороняє використовувади будь-які дані робочої області.

- **Treat as atomic unit** – визначення типу підсистеми як програмної одиниці при визначенні порядку виконання блочних методів. Поле може бути встановлене в режим **On** або **Off**. За замовчуванням встановлюється значення **Off**.

- **Treat as grouped when propagating variant conditions** – визначає обробку підсистеми як програмної одиниці при змінних умовах в залежності від умов блоків Variant Source або Variant Sink. Якщо режим встановлено, то, наприклад, коли Simulink обчислює варіантний стан підсистеми, він поширює цю умову для всіх блоків у підсистемі. В протилежному випадку, Simulink розглядає всі блоки в підсистемі як такі, що знаходяться на одному рівні в ієрархії моделі.

На другій закладці блоку **Code Generation** (Генерація коду) є лише одне поле (рис.126):

- Function packaging – визначення формата коду, який буде згенерований для атомної (невіртуальної) підсистеми.

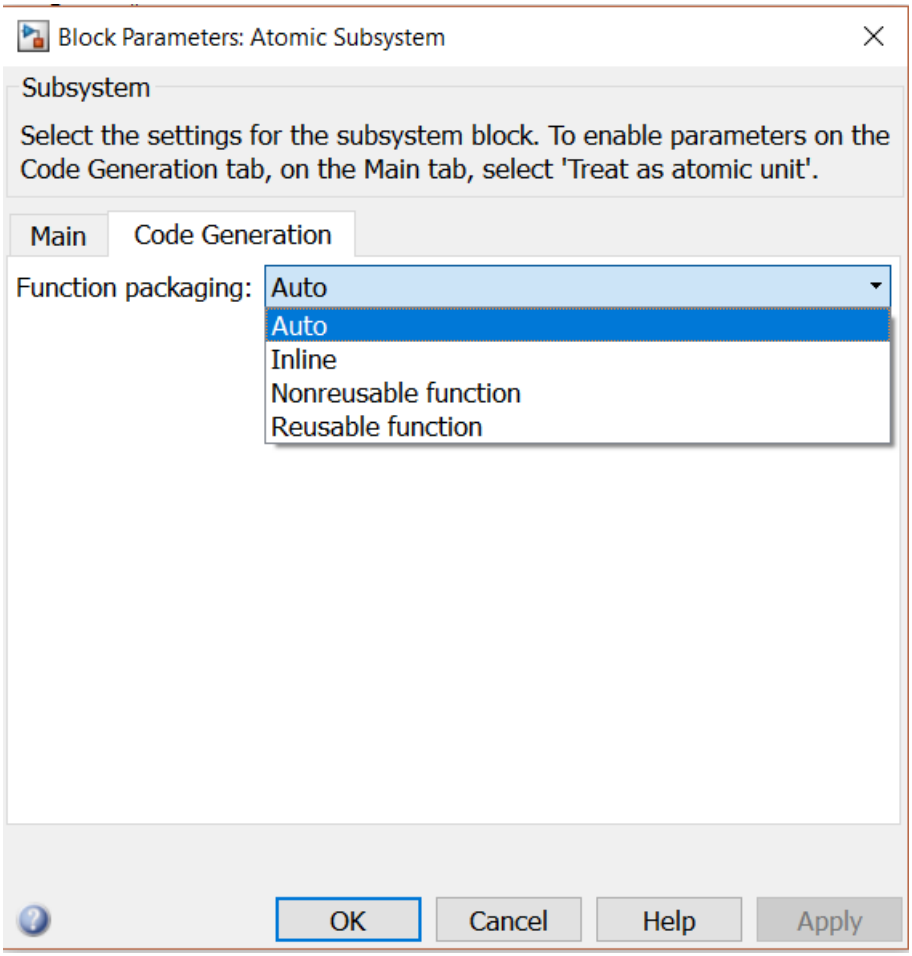


Рис. 126. Закладка Code Generation діалогового вікна блоку Atomic Subsystem

Поле може приймати наступні значення:

- **Auto** – система вибирає оптимальний формат за типом та кількістю екземплярів підсистеми, які існують у моделі;
- **Inline** – підсистема визначається як безумовна;
- **Nonreusable function** – система явно генерує окрему функцію в окремому файлі. Підсистеми з цим параметром генерують функції, які можуть містити аргументи залежно від параметра інтерфейсу функції. Назвати сформовану функцію та файл можна за допомогою параметрів Function nameand (Ім'я файлу) без розширення. Ці функції не є зворотними;
- **Reusable function** (Багаторазові функції) – система генерує функцію з аргументами, що дозволяє повторно використовувати код підсистеми, коли модель включає кілька екземплярів підсистеми.

Приклад: Порівняння довжини двох черг

У моделі, що містить дві черги, логічне порівняння довжини черг змінюється, коли в будь-якій з черг є надходження або відправлення замовлень. Вихідний сигнал #n блоку черги оновлюється після кожного надходження якщо черга не порожня, і після кожного відправлення. На відміну від нього, блок **Relational Operator** (Оператор відношення) є блоком, що працює за часом. Наведена нижче модель виконує порівняння всередині підсистеми

дискретних подій, в обох з яких у **Discrete Event Inport**-блоках поле **Type of signal-based event(тип події)** поставлено у Sample time hit (Встановлений час). Таким чином, порівняння відбувається кожного разу, коли будь-який сигнал #n оновлюється. Якщо обидві черги оновлюють свої значення #n одночасно за годинником симуляції, тоді підсистема викликається двічі.

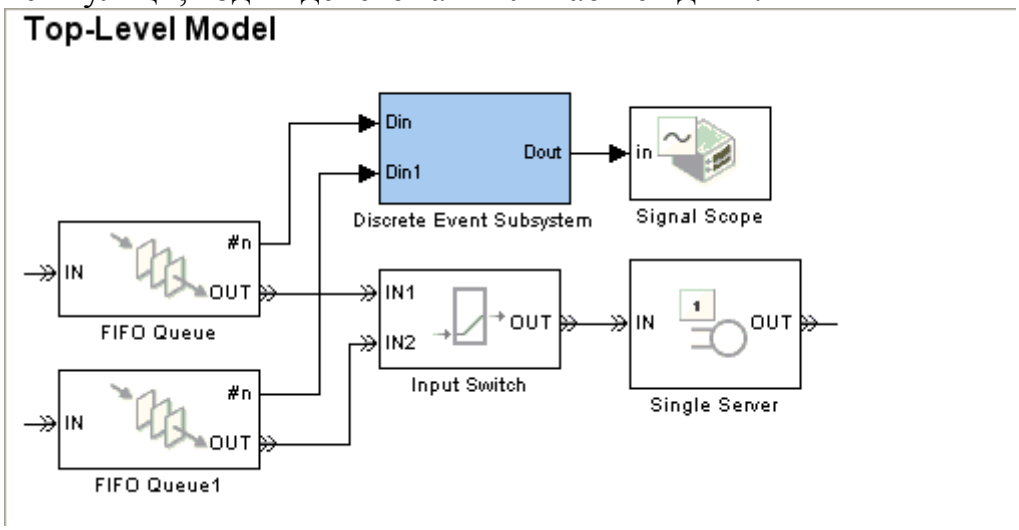


Рис. 127. Представлення імітаційної моделі

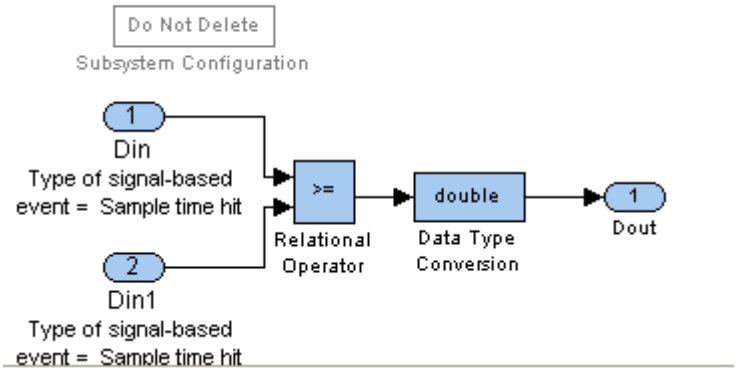


Рис. 128. Підсистема Discrete Event Subsystem

Event Filter

Event Filter (Фільтр подій) – цей блок впливає на блок атомної підсистеми, визначаючи події для виконання підсистеми. Цей блок також може встановити пріоритетність виконання підсистеми стосовно інших подій, що відбуваються одночасно шляхом планування виконання підсистеми в календарі подій. Розглянемо сигнал на основі подій, який є входом до блоку атомної підсистеми. Без блоку фільтрування подій, кожен часовий сигнал призводить до негайного виконання підсистеми. Вставлення блоку фільтра подій на цю сигнальну лінію дає змогу впливати на поведінку підсистеми наступним чином:

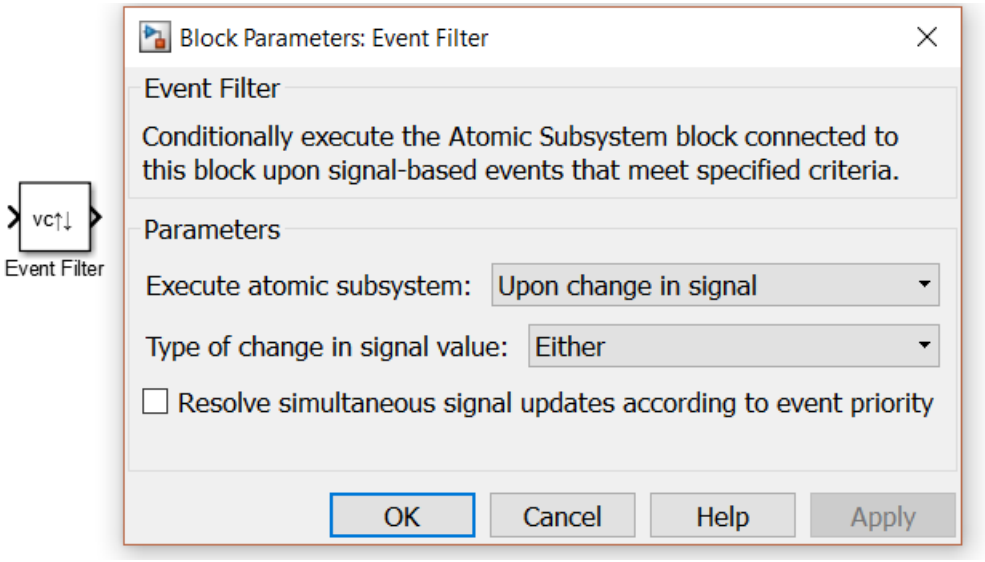


Рис. 129. Блок Event Filter та його діалогове вікно

Для налаштування блоку необхідно вказати тип події на основі сигналу, що викликає виконання підсистеми:

- Sample time hit – за встановленим часом;
- Change in signal value (rising, falling, or either) – за зміною значення сигналу (підйом, падіння або будь-який інший);
- Trigger (rising, falling, or either) – за тригером (підйом, падіння чи будь-який інший).

Якщо вхідний сигнал цього блоку є нескалярним масивом, блок визначає одну кваліфікаційну подію з позицій масиву. Наприклад, зміна значення сигналу з [1 2 3] на [1 5 6] представляє одну кваліфікаційну подію, а не дві.

Використання цього блоку дозволяє:

- запобігти вхідному сигналу викликати виконання підсистеми. У цьому випадку сигнал пасивно забезпечує передачу даних до підсистеми. Підсистема може виконуватись на основі сигнальних подій іншого вхідного сигналу;

- надавати пріоритетність виконання підсистемі щодо інших одночасних подій у симуляції. Замість того, щоб відбуватися відразу після події на основі сигналу, виконання стає запланованою подією в календарі подій.

Блок має один вихідний порт, який може підключатися лише до одного вхідного порту блоку атомної підсистеми. Лінія з'єднання не може відгалужуватися.

Коли вхідний сигнал блоку фільтру подій має встановлений час, виконується наступне:

1. Оновлюється вихідний сигнал за значенням вхідного сигналу. Це значення доступне для блоку атомної підсистеми, до якої підключається блок фільтру подій.

2. Визначає, чи виконувати блок атомної підсистеми, на основі налаштувань в діалоговому вікні блоку фільтру подій. Якщо блок фільтру подій не повинен виконувати блок атомної підсистеми, фільтр подій нічого більше не

робить, доки не надійде наступна вибірка вхідного сигналу. В іншому випадку, обробка продовжується до наступного кроку.

3. Визначає, коли виконувати блок атомної підсистеми.

• Якщо ви не вибрали **Resolve simultaneous signal updates according to event priority** (Оновити одночасне оновлення сигналів відповідно до параметра пріоритету події), блок блокування подій негайно виконує блок атомної підсистеми.

• Якщо ви виберете **Resolve simultaneous signal updates** (Оновити сигнали одночасного оновлення відповідно до параметру пріоритету події), блок складає події у календар подій. Час події – це поточний час моделювання. Пріоритет події – це значення параметра пріоритету події у блоці фільтрів подій. Коли календар події виконує цю подію, блок атомної підсистеми виконує його обчислення.

Діалогове вікно блоку містить одну закладку з наступними полями:

- **Execute atomic subsystem** – визначає кваліфікаційну подію у вхідному сигналі цього блоку. Якщо сигнал складний, потрібно вибрати **Upon sample time hit** (Згідно до встановленого часу) або **Never** (Ніколи).
- **Type of change in signal value** (Тип зміни значення сигналу) – Тип зміни значення сигналу, що додатково обмежує тип події, вказаний в асиметричній підсистемі Execute. Він доступний, лише якщо встановлено **Execute atomic subsystem** (Виконати атомну підсистему) у **Upon change in signal** (Після зміни сигналу).

2. Приклади побудови імітаційних моделей

2.1. Приклад 1

Постановка задачі

В центрі обслуговування банку працюють два оператори. Клієнти обслуговуються в порядку живої черги. Черга необмежена. Потік клієнтів має закон розподілу Пуасона і середня інтенсивність становить 7 клієнтів на годину. Час обслуговування клієнта становить в середньому 15 хвилин і підпорядковується експоненціальному закону. Необхідно побудувати імітаційну модель системи і визначити на її основі основні характеристики:

середній час очікування в черзі; середній час знаходження в системі; довжину черги.

Порядок виконання роботи:

1. Розробити структуру моделі.

2. Побудувати імітаційну модель за допомогою пакету Simulink. Ввести в модель вихідні дані і зробити необхідні налаштування моделі.

Необхідні параметри системи представити у вигляді графіків.

1.1 Розробка моделі

Згідно до позначень Кендалла-Лі наша задача відноситься до задач типу *M/M/.2*. Розглянемо сервер, який одночасно обробляє 2 замовлення (2 оператора), на вхід якого замовлення надходять з інтенсивністю $\lambda = 7$.

Середній час обслуговування замовлень сервером становить $t_{обсл.} = 0{,}25$ год. Припустимо, що сервер обробляє замовлення відразу, як тільки воно надходить на його вхід. Коли сервер завершує обробку поточної заявки, надходить нове замовлення і сервер знову береться за роботу.

1.2 Побудова імітаційної моделі

Для побудови моделі оберемо наступні блоки та виконаємо необхідні налаштування:

- блок генератора подій **Time-Based Entity Generator**. У діалоговому вікні виберемо закон розполілу **Distribution** як експоненціальний та встановимо значення математичного сподівання $1/\lambda = 1/7$;

- блок, який реалізує чергу з відповідною дисципліною обслуговування замовлень (FIFO Queue). На основній вкладці у полі **Capacity** яке визначає довжину черги встановимо значення **Inf** – нескінченність. На закладці статистики поля **Average wait** та **Average queue length** встановимо у режим **on**. Параметр Average wait показує час очікування замовлень на обслуговування, а параметр Average queue length довжину черги очікування на обслуговування. Після того, як ми оберемо відповідні поля для блоку черги відкриється два вихідні порти: w та len;

- обслуговуючий прилад (N Server). У діалоговому вікні встановлюємо кількість сервісів рівним 2 і на вкладці статистики обираємо поле **Utilization** (утилізація або доля часу моделювання на зберігання сутності);

- приймач оброблених замовлень (**Entity Sink**);

- дисплей, що відображає кількість оброблених замовлень;

- три блоки **Signal Scope** для візуалізації процесу моделювання, які з'єднуємо з відповідними вихідними портами блоків черги та сервера;

-. блок осцилографа Scope для виводу графіків в одному вікні;

- шину об'єднання потоків для побудови графіка **Scope.**

Виконаємо необхідні з'єднання блоків і отримаємо схему моделі, представлену на рис. 130.

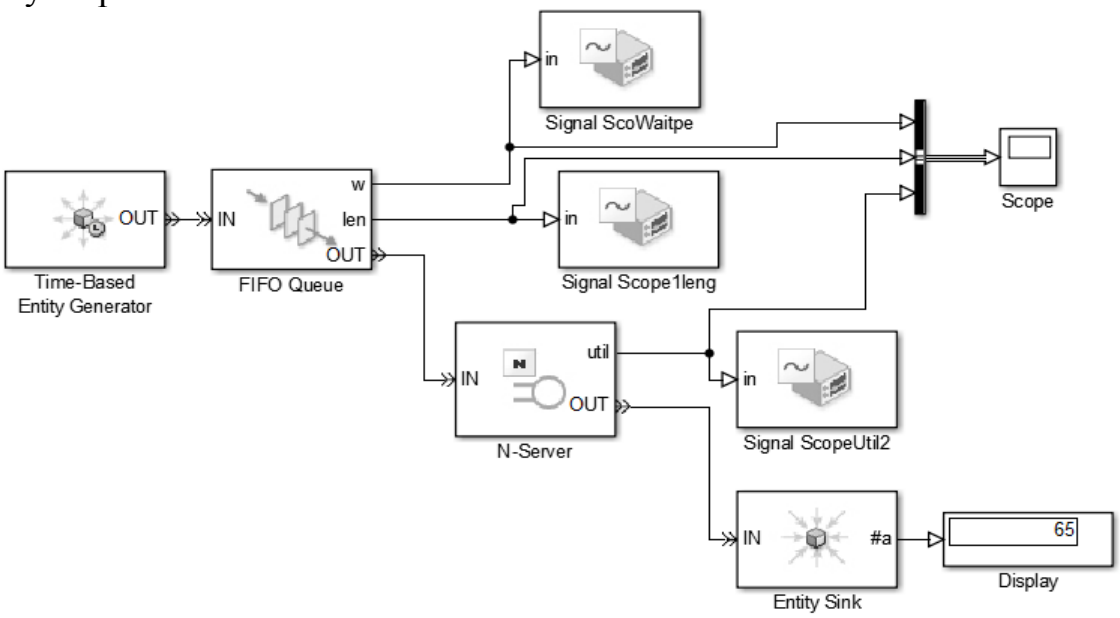


Рисунок 130. Схема імітаційної моделі

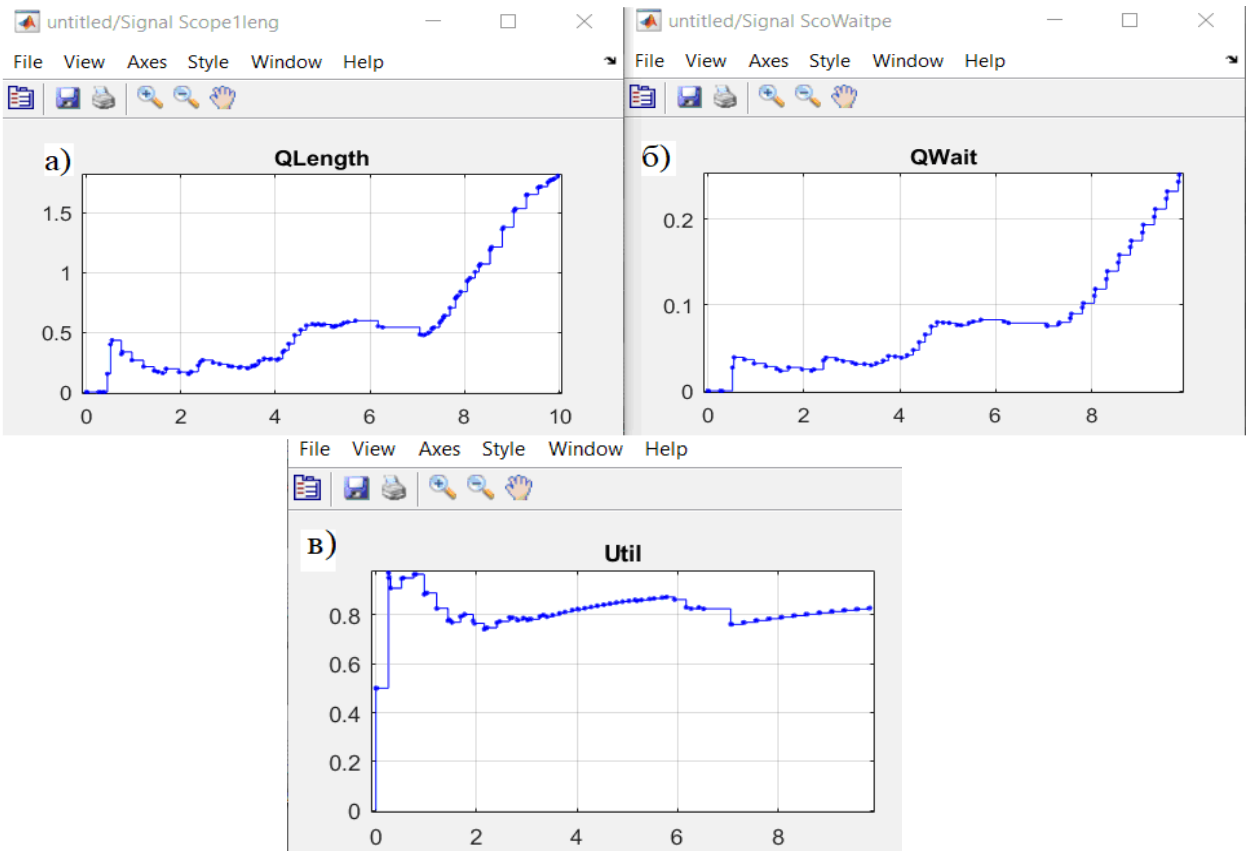


Рис.131. Графічне представлення результатів моделювання: а) середня довжина черги; б) середній час очікування; в) доля часу моделювання, яка використана на зберігання сутності

Результати моделювання представлені на рис. 132. З графіків видно, що на протязі робочого дня (8 год.) черга не перевищує однієї особи і середній час очікування у черзі не перевищує 6 хв. Можна сказати, що система добре збалансована і її робота ефективна.

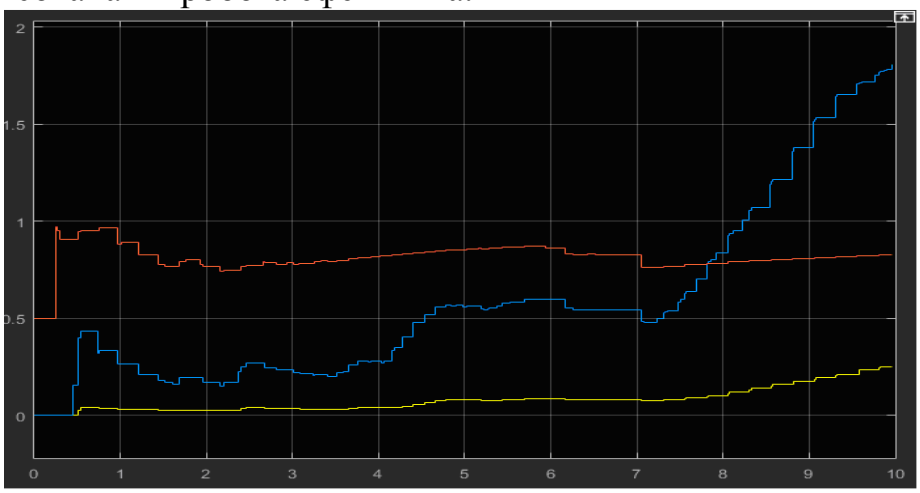


Рис. 132. Представлення графіків у вікні Scope

2.2. Приклад 2

Постановка задачі.

Створити імітаційну модель сервісного центру, де інтервали часу між

надходженням клієнтів до одного майстра становлять 18±6 хв. Час обслуговування розподілений рівномірно на інтервалі 16±4 хв. Клієнти обслуговуються в порядку «черги першим пийшов – першим обслуговується». Виконати моделювання на протязі робочого дня, який становить 8 год. Необхідно визначити параметри функціонування центру: коефіцієнт завантаження майстра; максимальну, середню і поточну кількість клієнтів у черзі; середній час обслуговування.

Порядок виконання роботи:

1. Розробити структуру моделі.

2. Побудувати імітаційну модель за допомогою пакету Simulink. Ввести в модель вихідні дані і зробити необхідні налаштування моделі.

Необхідні параметри системи представити у вигляді графіків.

2.1 Розробка моделі

Згідно до позначень Кендалла-Лі наша задача відноситься до задач типу *D/M/.1*. Розглянемо сервер, який одночасно обробляє 1 замовлення. На вхід замовлення надходять через рівні проміжки часу 18±6 хв. Середній час обслуговування замовлень сервером становить $t_{обсл.} = 16 \pm 4$ хв. Припустимо, що сервер обробляє замовлення відразу, як тільки воно надходить на його вхід. Коли сервер завершує обробку поточної заявки, надходить нове замовлення і сервер знову береться за роботу.

2.2 Побудова імітаційної моделі

Для побудови моделі оберемо наступні блоки та виконаємо необхідні налаштування:

- блок генератора подій Time-Based Entity Generator. У випадкові моменти часу блок генерує події, що моделюють надходження клієнтів. В параметрах блоку вибраний тип розподілу **Uniform** (рівномірний), параметри: Minimum – 12, Maximum 24;
- блок **FIFO Queue** створює чергу клієнтів. Довжину черги приймемо як **inf** (нескінченну);
- Блок **Single Server** моделює обслуговування клієнта майстром. Час обслуговування задається через сигнальний порт **t** (Service time from – Signal port t) блоком **Event Based Random Number**, що генерує рівномірно розподілені випадкові числа з параметрами: Minimum – 12, Maximum 20;
- блок **Entity Sink** поглинає замовлення, обробка яких завершена;
- блоки **Signal Scope** графічно представляють процес імітації.

З'єднавши всі блоки, отримуємо імітаційну модель системи (рис. 133).

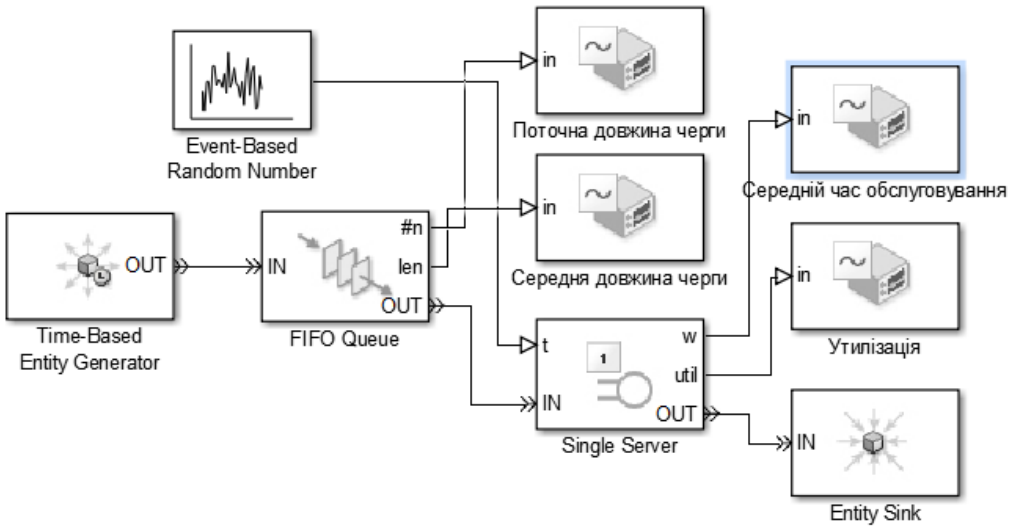


Рис. 133. Імітаційна модель центру обслуговування

Встановимо час моделювання 8 х 60=480 хв. і запустимо модель на виконання. Графічне представлення результатів імітації відображно на рис. 134, 135.

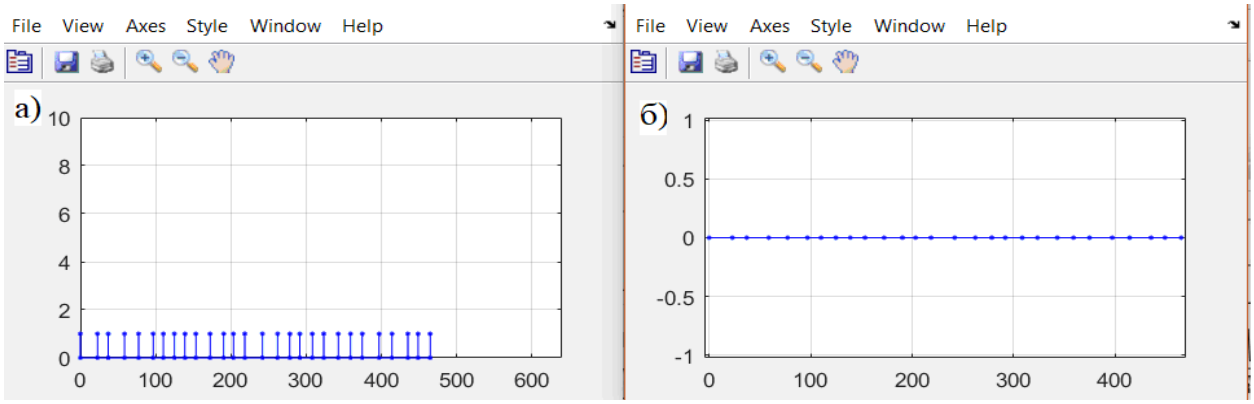


Рис.134. Графіки а) поточної та б) середньої довжини черги

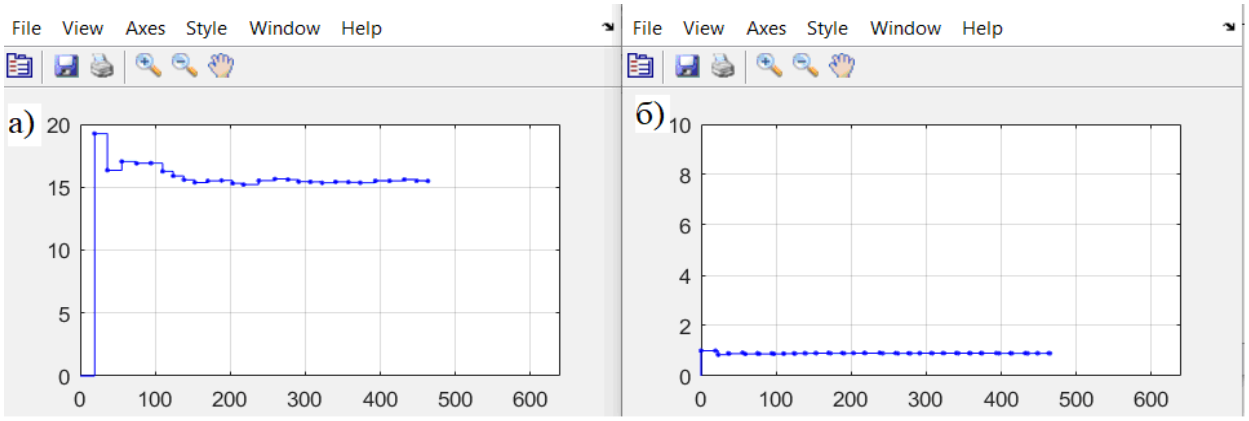


Рис. 135. Графіки роботи сервісу: а) середній час обслуговування; б) зміна завантаження майстра за часом.

2.3. Приклад 3

Постановка задачі.

Розробити імітаційну модель сервісного центру, куди надходять клієнти

з різними пріоритетами, причому клієнти з вищим пріоритетом надходять з інтенсивністю в 5 разів меншою. Клієнти стають у чергу згідно до свого пріоритету (довжина черги становить 5 місць) і обслуговуються двома операторами, які мають різну кваліфікацію, що впливає на час обслуговування (2/3 і 3/4 одиниці часу). Виконати моделювання на протязі робочого дня, який становить 8 год.

2.1 Розробка моделі

Згідно до позначень Кендалла-Лі наша задача відноситься до задач типу *M/M/.2*. Замовлення надходять за експоненціальним законом розподілу $\lambda_1 = 1/5$ (пріортет 2) і $\lambda_2 = 1$ (пріоритет). Замовлення надходять у чергу згідно до пріоритету за спаданням його значення, після чого вони направляються на обробку на вільний сервіс. Час обслуговування сервісів становить відповідно 2/3 і 3/4 одиниці часу.

2.2 Побудова імітаційної моделі

Для побудови моделі оберемо наступні блоки та виконаємо необхідні налаштування:

- блоки генератора випадкових чисел **Event Based Random Number**. У діалоговому вікні виберемо закон розполілу **Distribution** як експоненціальний та встановимо значення математичного сподівання $1/\lambda = 1/5$ і $\lambda = 1$;

- для генерації замовлень за заданими законами встановимо блоки **Time-Based Entity Generator**;

- для встановлення відповідних пріоритетів застосуємо блоки **Set Attribute.** В обох блоках визначимо однакове ім'я атрибуту att, але для клієнтів з $\lambda = 1$ встановимо значення 1, а для клієнтів із значенням $\lambda = 5$ – значення 2.

- для об'єднання потоків використаємо блок **Path Combiner**, встановивши статус циклічного опитування двох вхідних портів Round robin;

- для контролю часу встановимо блок **Start Time**;

- для реалізації черги виберемо блок **Priority Queue**, встановивши в ньому назву атрибуту att та порядок обслуговування черги за зростанням значення атрибуту. Довжину черги встановимо рівну 5;

- згідно до встановлених пріоритетів, замовлення надходять на обслуговування на вільний у поточний момент часу сервіс **Single Server**. Для розподілу замовлень на два потоки встановимо блок **Output Switch**. Для кожого сервісу встановимо відповідний час обслуговування;

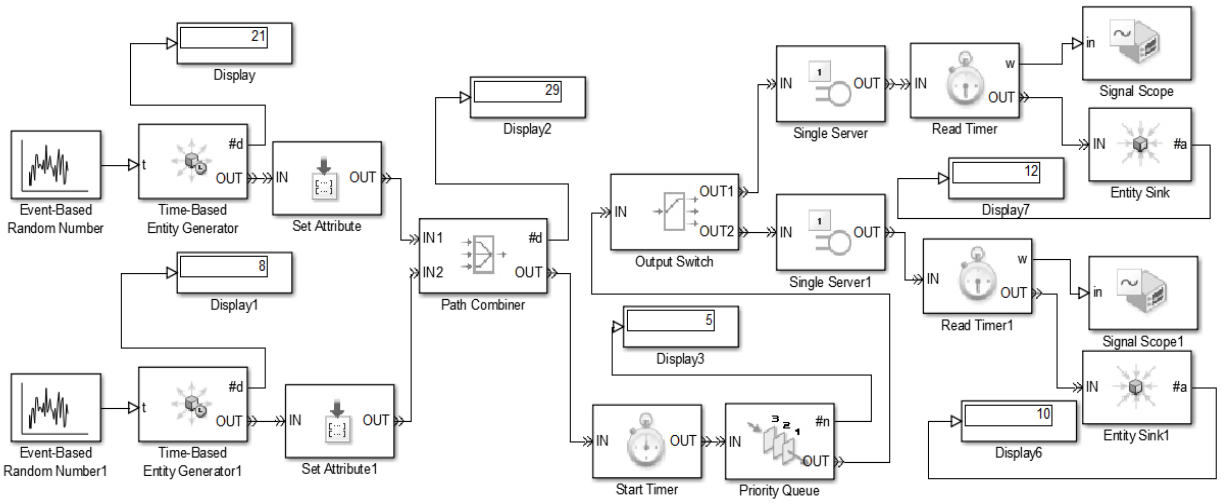


Рис. 136. Імітаційна модель системи (приклад3)

- для побудови графіків вихідних потоків кожного сервісу за часом встановимо блоки **Read Timer**;

- блок **Entity Sink** поглинає замовлення, обробка яких завершена. Кількість замовлень, що пройшли через відповідний сервіс виведемо на блок **Display**;

- блоки **Signal Scope** графічно представляють процес імітації.

Імітаційна модель системи представлена на рис.136. Графіки обслуговування сервісами наведені на рис. 137.

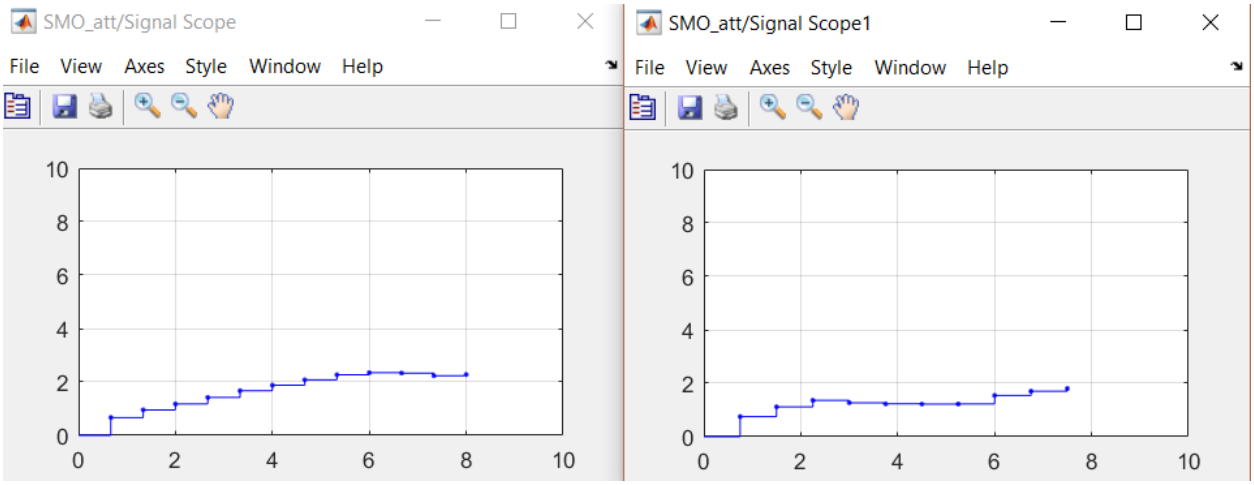


Рис. 137. Графіки обслуговування сервісами

3. Задачі для самостійного виконання

1. Автозаправна станція має 4 бензоколонки. Середній час заправки - 2 хв. Вхідний потік автомашин – найпростіший з інтенсивністю 1,5 авт./хв. Якщо всі колонки зайняті, то машина на станцію не заїжджає. Визначте ймовірність відмови і середнє число зайнятих колонок.

2. Покупці магазину утворюють найпростіший потік вимог з інтенсивністю 150 чол./год. Визначте найменше число продавців, при яких середня кількість покупців, які очікують обслуговування, не перевищить 3.

3. У нафтоналивному порту 4 причали для заправки танкерів, які приходять в середньому через 18 год, а час завантаження становить в середньому дві доби. У черзі можуть стояти не більше 2 танкерів. Визначте пропускну здатність і холостий хід порту.

4. Потік бажаючих оформити виклик лікаря додому – найпростіший. В середньому абоненти телефонують через кожні 10 с. Час прийому виклику розподілено за показниковим законом із середнім значенням 12 с. Визначте найменше число телефонів в реєстратурі, при якому виклик приймається не менш ніж від 90% абонентів. Вважається, що в разі невдачі абонент не робить більше спроб додзвонитися.

5. Автоматична мийка може прийняти на обслуговування одночасно 4 автомашини. В середньому машини прибувають через 2 хв., а середня тривалість мийки - 10 хв. У черзі можуть перебувати не більше 6 машин. Визначте ймовірність того, що в системі знаходиться хоча б одна машина, і завантаженість однієї установки для миття машин.

6. У магазині є 3 довідкових телефони. В середньому звертаються за довідками 40 чол./год. Середня тривалість розмови - 3 хв. Витрати, пов'язані з роботою одного телефону, 0,5 грн./хв. Визначте мінімальну вартість однієї хвилини розмови по телефону, при якій система незбиткова.

7. Платна стоянка для легкових машин має 7 місць. Знайдіть ймовірність того, що машина, яка прибула, знайде вільне місце, якщо машини в середньому прибувають через 10 хв., а займають місце на стоянці в середньому - 1год.

8. Потік деталей, що сходять з конвеєра, найпростіший з інтенсивністю - 2 дет./хв. Час перевірки деталі контролером має показниковий закон розподілу в середньому 2 хв./дет. Визначте частку неперевірених деталей.

9. Місто обслуговують 4 машини швидкої допомоги. Виклики надходять в середньому через 4 год. Імовірність того, що хоча б одна машина зайнята, дорівнює 0,25. Визначте середнє число зайнятих машин і середню частку простою машин.

10. У перукарні працюють два майстри. Час обслуговування розподілено за показниковим законом в середньому 12 хв. Очікувати обслуговування можуть не більше трьох осіб. Потік клієнтів – найпростіший з інтенсивністю 10 клієнтів/год. Знайдіть найважливіші операційні характеристики цієї системи.

11. Система автоматичної посадки літаків одночасно може зберігати дані тільки про 6 літаків, що знаходяться в повітрі. Літаки, що підлітають до

аеродрому, утворюють найпростіший потік з інтенсивністю 6 літаків/год. Якщо в момент запиту посадки система заповнена, то літак відлітає до запасного аеродрому. Аеродром має 3 посадкові смуги, літак займає смугу в середньому 20 хв. Знайдіть пропускну здатність СМО, завантаженість однієї смуги, середнє число зайнятих смуг, середній час очікування початку посадки після запиту.

12. Розглядається робота автозаправної станції (АЗС), на якій є 2 заправні колонки. Припустимо, що вона описується процесом розмноження і загибелі в стаціонарному режимі. Заправка кожної машини триває в середньому 3 хвилини. В середньому на АЗС кожні дві хвилини прибуває машина. Число місць в черзі необмежена. Всі машини, що стали на заправку, терпляче чекають своєї черги. Визначте: ймовірність того, що на заправці знаходиться 5 машин; ймовірність того, що знов прибулій машині доведеться чекати обслуговування.

13. Закусочна на АЗС має один прилавок. Автомобілі прибувають відповідно до пуасонівського розподілу, в середньому 2 автомобілі за 5 хвилин. Для виконання замовлення в середньому досить 1.5 хвилини, хоча тривалість обслуговування розподілено за експоненціальним законом. Знайдіть: ймовірність простою прилавка; середні показники; ймовірність того, що кількість прибулих автомобілів буде не менше 10.

14. Рентгенівський апарат дозволяє обстежити в середньому 7 осіб на годину. Інтенсивність відвідувачів становить 5 осіб на годину. Припускаючи стаціонарний режим роботи, визначте середні характеристики.

15. У річковому порту один причал, інтенсивність вхідного потоку 5 суден в день. Інтенсивність вантажно-розвантажувальних робіт – 6 суден в день. Маючи на увазі стаціонарний режим роботи, визначте всі середні характеристики системи.

16. Яке оптимальне число каналів обслуговування повинна мати СМО, якщо інтенсивність потоку заявок дорівнює 3, середнє число, заявок обслужених в одиницю часу, дорівнює 2, штраф за кожну відмову дорівнює 5, а вартість простою однієї лінії дорівнює 2?

17. Яке оптимальне число каналів обслуговування повинна мати СМО, якщо інтенсивність потоку заявок дорівнює 3, середнє число заявок, обслужених в одиницю часу, дорівнює 1, штраф за кожну відмову дорівнює 7, а вартість простою одній лінії дорівнює 3?

18. Яке оптимальне число каналів обслуговування повинна мати СМО, якщо інтенсивність потоку заявок дорівнює 4, середнє число заявок, обслужених в одиницю часу, дорівнює 2, штраф за кожну відмову дорівнює 5, а вартість простою однієї лінії дорівнює 1?

19. Визначте число злітно-посадочних смуг для літаків з урахуванням вимоги, що ймовірність очікування повинна бути меншою, ніж 0.05. При цьому інтенсивність вхідного потоку - 27 літаків на добу, а інтенсивність їх обслуговування - 30 літаків на добу.

20. Скільки рівноцінних незалежних конвеєрних ліній повинен мати цех, щоб забезпечити ритм роботи, при якому ймовірність очікування обробки виробів повинна бути меншою ніж 0.03 (кожний виріб випускається однією

лінією). Відомо, що інтенсивність надходження замовлень - 30 виробів за годину, а інтенсивність обробки виробу однією лінією – 36 виробів за годину.

21. Середнє число викликів, що надходять на АТС за одну хвилину, дорівнює 3. Вважаючи потік пуасонівським, знайдіть ймовірність того, що за 2 хвилини надійде: два виклики; менше двох викликів; не менше двох викликів.

22. Скільки каналів повинна мати СМО з відмовами, якщо інтенсивність потоку заявок дорівнює 2 замов./год, середнє число заявок, обслужених в одиницю часу, дорівнює 1 замов./год, штраф за кожну відмову складає 8т.грн., Вартість простою одній лінії - 2т. грн. в годину?

23. Система масового обслуговування являє собою автоматичну телефонну станцію, яка може забезпечити не більше п'яти переговорів одночасно. Заявка-виклик, що надійшла в той момент, коли всі канали зайняті, отримує відмову і покидає систему. В середньому на станцію надходить 0,8 викликів в хвилину, а середня тривалість одних переговорів дорівнює 1,5 хвилини. Для стаціонарного режиму функціонування системи необхідно визначити: ймовірності станів системи; ймовірність відмови; абсолютну і відносну пропускні спроможності; середнє число зайнятих каналів.

24. Робітник обслуговує три однотипних верстати. Кожен верстат зупиняється в середньому два рази на годину, а процедура налагодження займає в середньому 10 хвилин. У стаціонарному режимі функціонування системи потрібно визначити: ймовірності станів системи; ймовірність зайнятості робітника; середню кількість несправних верстатів; середнє число верстатів, що налагоджуються.

25. Спеціалізований пост діагностики є одноканальною СМО. Число стоянок для автомобілів, які очікують проведення діагностики, обмежено і дорівнює 3. Якщо всі стоянки зайняті, то черговий автомобіль, що прийшов на діагностику, в чергу на обслуговування не стає. Потік надходження автомобілів – пуассонівський і має інтенсивність 0.85 автомобіля на годину. Час діагностики розподілено за показниковим законом і в середньому становить 1.05 години. Проведіть порівняльний аналіз роботи СМО при S = 3 і S = 4.

26. Потрібно промоделювати роботу невеликого магазину, який має один касовий апарат і одного продавця. Відомі такі параметри функціонування магазину: потік покупців (заявок), що приходять в магазин за покупками, – рівномірний; інтервал часу прибуття покупців коливається в межах від 8,7 хвилини до 10,3 хв. включно, або 9,5 ± 0,8 хв; час перебування покупців у касового апарата становить 2,3 ± 0,7 хв. Після цього покупці підходять до продавця для отримання товару; час, витрачений на обслуговування покупців продавцем, складає 10 ± 1,4 хв. Потрібно визначити параметри функціонування магазину: коефіцієнт завантаження касира; коефіцієнт завантаження продавця; максимальне, середнє і поточне число покупців в кожній черзі; середній час обслуговування в кожному каналі обслуговування.

27. Дисплейний зал має 5 дисплеїв. Потік користувачів - найпростіший. Середнє число користувачів, які відвідують дисплейний зал за добу, становить 140 осіб. Час обробки інформації одним користувачем на одному дисплеї

розподілено за показниковим законом і становить в середньому 40 хвилин. Визначити, чи існує стаціонарний режим роботи залу; ймовірність того, що користувач застане всі дисплеї зайнятими; середнє число користувачів в дисплейному залі; середнє число користувачів в черзі; середній час очікування вільного дисплея; середній час перебування користувача в дисплейному залі.

28. Контроль готової продукції фірми здійснюють три контролера. Якщо виріб надходить на контроль, коли всі контролери зайняті перевіркою готових виробів, то він залишиться неперевіреним. Середнє число виробів, що випускаються фірмою, складає 20 вид./год. Середній час на перевірку одного виробу - 7 хв. Визначити показники ефективності відділу технічного контролю. Скільки контролерів необхідно поставити, щоб ймовірність обслуговування склала не менше 97%?

29. У нафтоналивному порту 4 причали для заправки танкерів, які приходять в середньому через 18 годин, а час завантаження становить в середньому дві доби. У черзі можуть стояти не більше 2 танкерів. Визначте пропускну здатність і холостий хід порту

30. Платна стоянка для легкових машин має 7 місць. Знайдіть ймовірність того, що машина, що прибула, знайде вільне місце, якщо машини в середньому прибувають через 10 хв, а займають місце на стоянці в середньому 1 годину.

31. Потік деталей, що сходять з конвеєра, найпростіший з інтенсивністю 2 дет./хв. Час перевірки деталі контролером має показовий закон розподілу в середньому 2 хв./дет. Визначте частку неперевірених деталей.

32. У пункті хімчистки є три апарати для чищення. Інтенсивність потоку відвідувачів $\lambda = 6$ відвідувачів на годину. Інтенсивність обслуговування відвідувачів одним апаратом $m = 3$ відвідувачів на годину. Середня кількість відвідувачів, які покидають чергу, не дочекавшись обслуговування, $n = 1$ відвідувач на годину. Знайти абсолютну пропускну спроможність пункту.

33. Нехай для обслуговування 10 персональних комп'ютерів (ПК) виділено два інженери з однаковою продуктивністю. Потік відмов (несправностей) одного комп'ютера – пуасонівський з інтенсивністю 0,2. Час обслуговування ПК підпорядковується показниковому закону. Середній час обслуговування одного ПК одним інженером становить 1,25 години. Можливі наступні варіанти організації обслуговування:

- обидва інженери обслуговують всі 10 комп'ютерів, так що при відмові ПК його обслуговує один з вільних інженерів в цьому випадку;
- кожен з двох інженерів обслуговує по п'ять закріплених за ним ПК. У цьому випадку необхідно вибрати найкращий варіант організації обслуговування ПК.

34. Нехай n-канальна СМО являє собою обчислювальний центр (ВЦ) з 3 взаємозамінними ПЕОМ для вирішення завдань. Потік завдань, що надходять на ВЦ, має інтенсивність 1 завдання на годину. Середня тривалість обслуговування 1,8 години. Потік пакетів на вирішення завдань і потік обслуговування цих пакетів є найпростішими. Потрібно обчислити фінальні значення: ймовірності станів ВЦ; ймовірності відмови в обслуговуванні

пакетів; відносної пропускної спроможності ВЦ; абсолютної пропускної здатності ВЦ; середнього числа зайнятих ПЕОМ на ВЦ. Визначте, скільки додатково треба придбати ПЕОМ, щоб збільшити пропускну здатність ВЦ в 2 рази.

35. В обчислювальному центрі працює 5 персональних комп'ютерів (ПК). Найпростіший потік завдань, що надходять на ВЦ, має інтенсивність 10 завдань на годину. Середній час вирішення задачі дорівнює 12 хв. Вимога отримує відмову, якщо всі ПК зайняті. Знайдіть ймовірнісні характеристики системи обслуговування (ВЦ).

http://au.kai.ru/documents/Faizutdinov_Model_slogn_sistem.pdf
https://narfu.ru/university/library/books/1163.pdf