2.1. 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 10).

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;

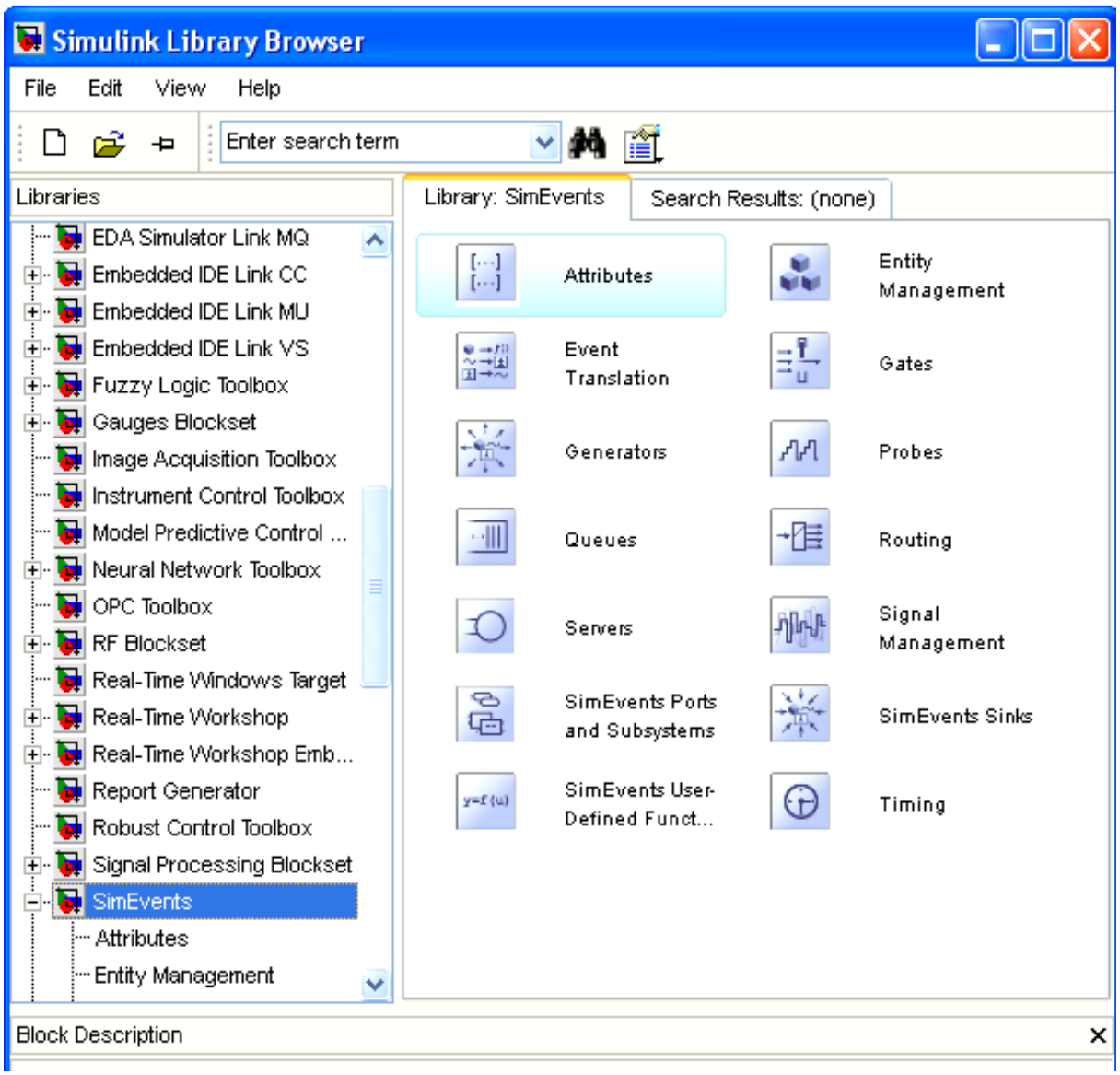


Рис. 10 Компоненти бібліотеки SimEvents

- 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. 11.

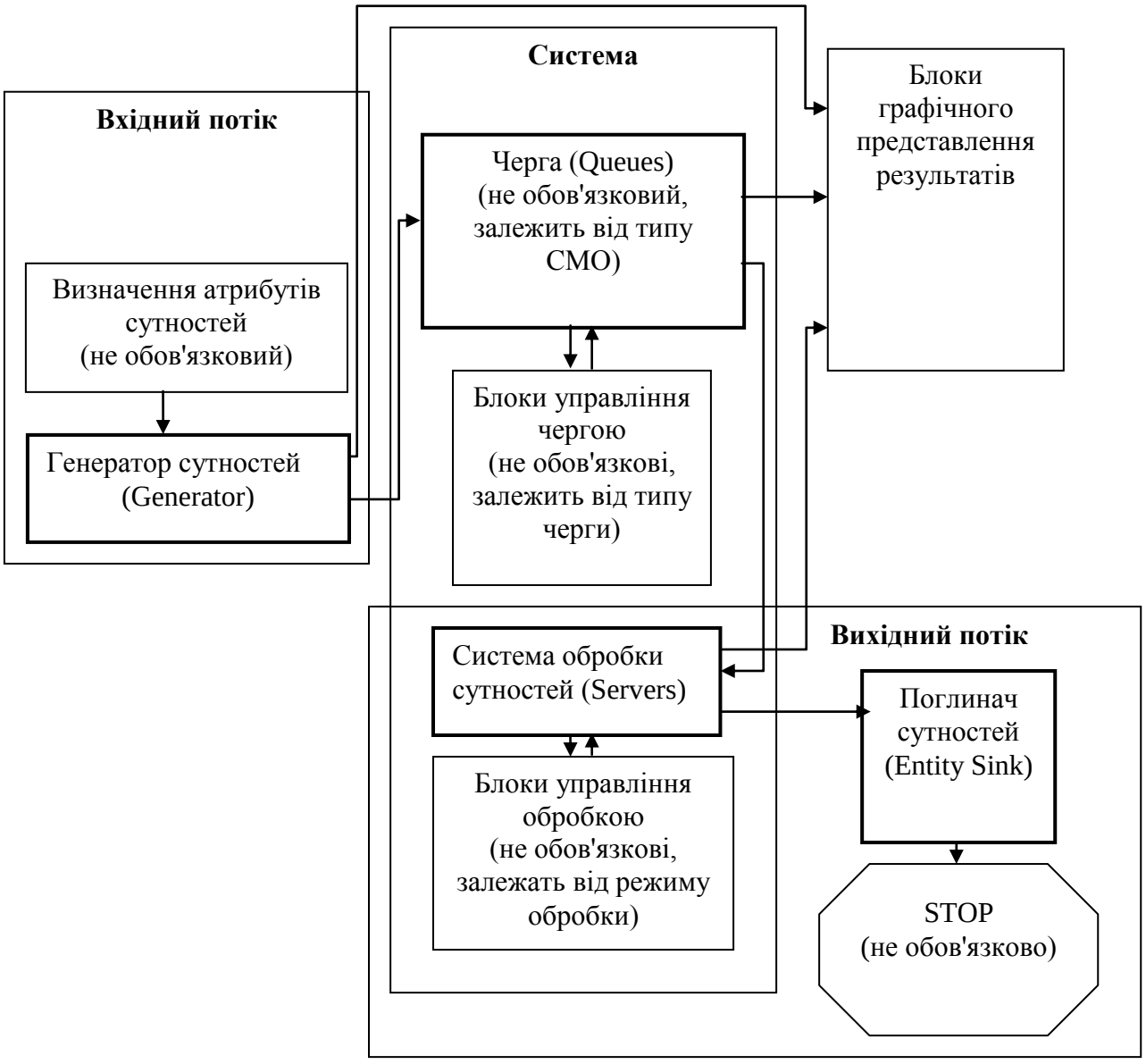


Рис. 11 Структура імітаційної моделі СМО

Графічні блоки мають декілька закладок для їх налаштування. Для отримання статистички передбачена закладка Statistics, яка містить поля основних статистичних даних для відповідного плоку, 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.

2.2. 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. 12)

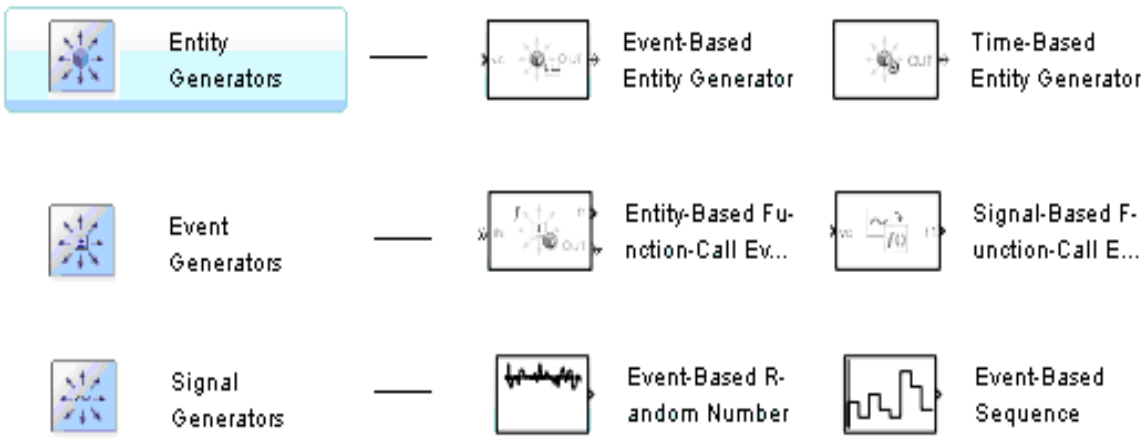


Рис. 12 Компоненти бібліотеки Generators

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 строрення потоків замовлень (сутностей), тоді як блоки генерації подій та сигналів використовують для управління іншими блоками.

Таблиця 1. Умови генерації сутностей

Умови генерації сутностей	**Порт**
Генерація сутностей відбувається коли додаток повторно обчислюється і видає значення сигналу	ts
Генерація сутностей управляється тригером	tr
Генерація сутностей управляється вхідним сигналом	vc
Генерація сутностей управляється функцією	fcn

2.2.1 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.

2.2.1.1 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 13. 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 не доступний для прийому сутності (замовлення).

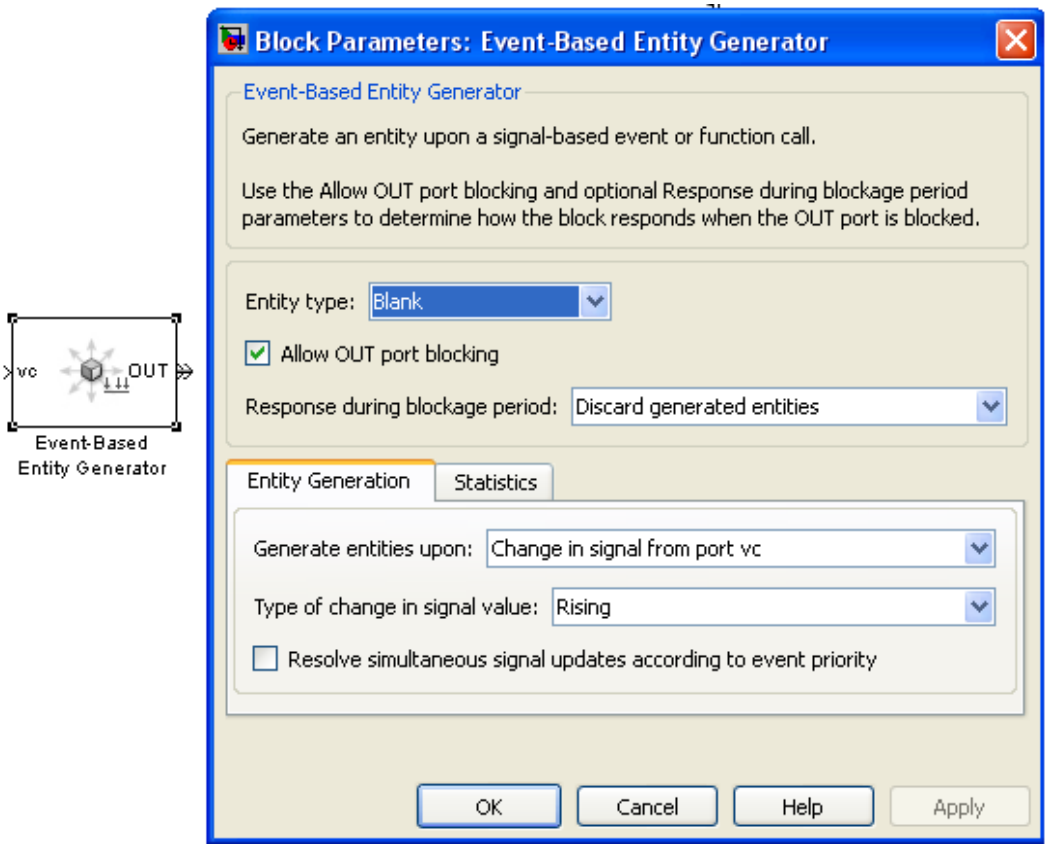


Рис. 13 Діалогове вікно налаштування блоку Event Based Entity Generator

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 14 (див. таблицю 1).

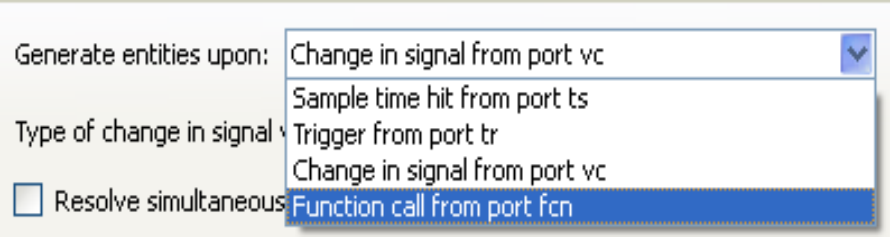


Рис. 14 Значення для налаштування порядку генерації сутностей

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. 15. Для збору статистики необхідно у відповідні поля встановити значення On, тоді у блоку з'являється новий вихідний порт для відображення результатів.

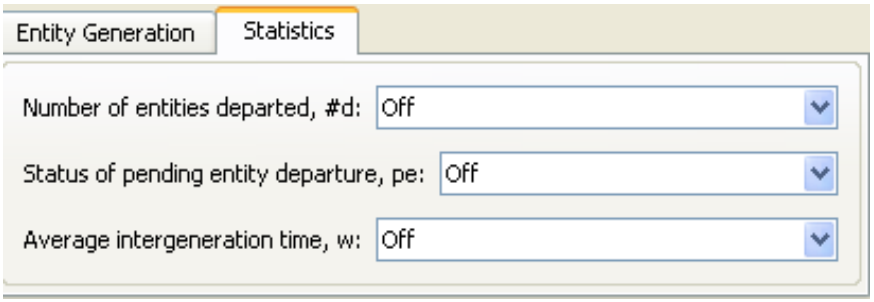


Рис. 15 Налаштування статистики

Number of entities departed: #d – кількість сутностей, що вийшло 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. 16 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.

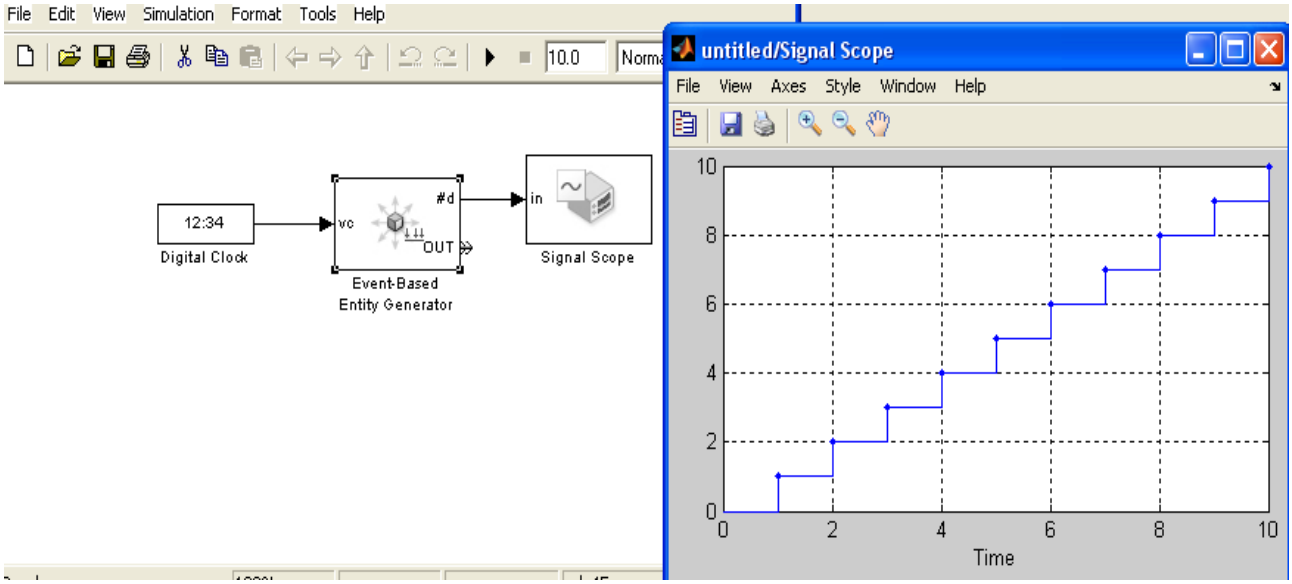


Рис. 16 Приклад простого використання блоку Event Based Entity Generator

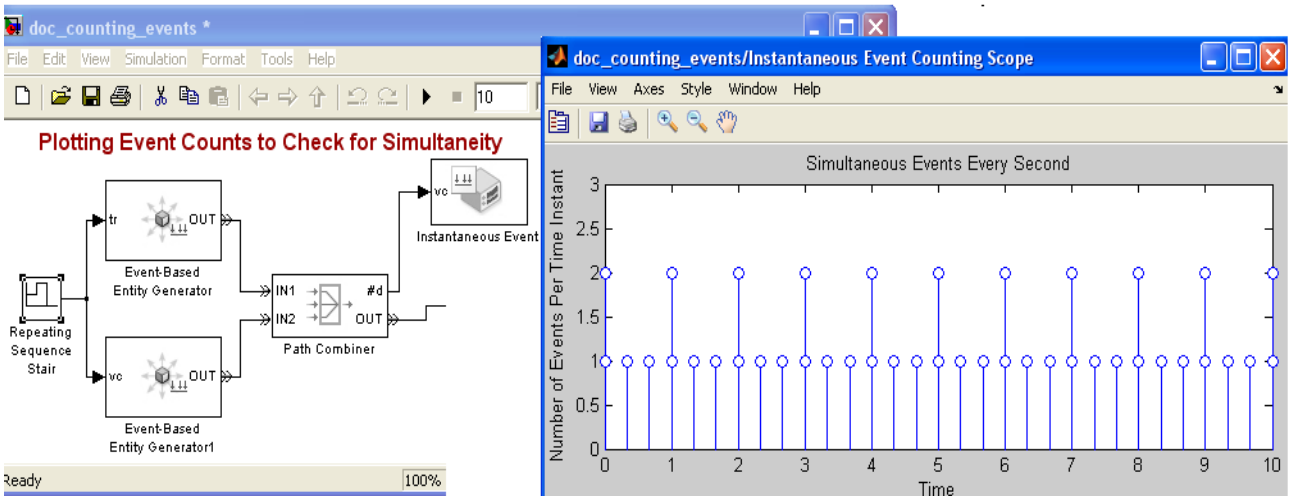


Рис. 17 Приклад фрагменту моделі з двома генераторами

Більш складний приклад наведено на рис. 17. 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.

2.2.1.2 Time-Based Entity Generator

Time-Based Entity Generator – блок генерує сутності у моменти часу, які визначаються вхідним сигналом або статистичним розподілом (рис. 18).

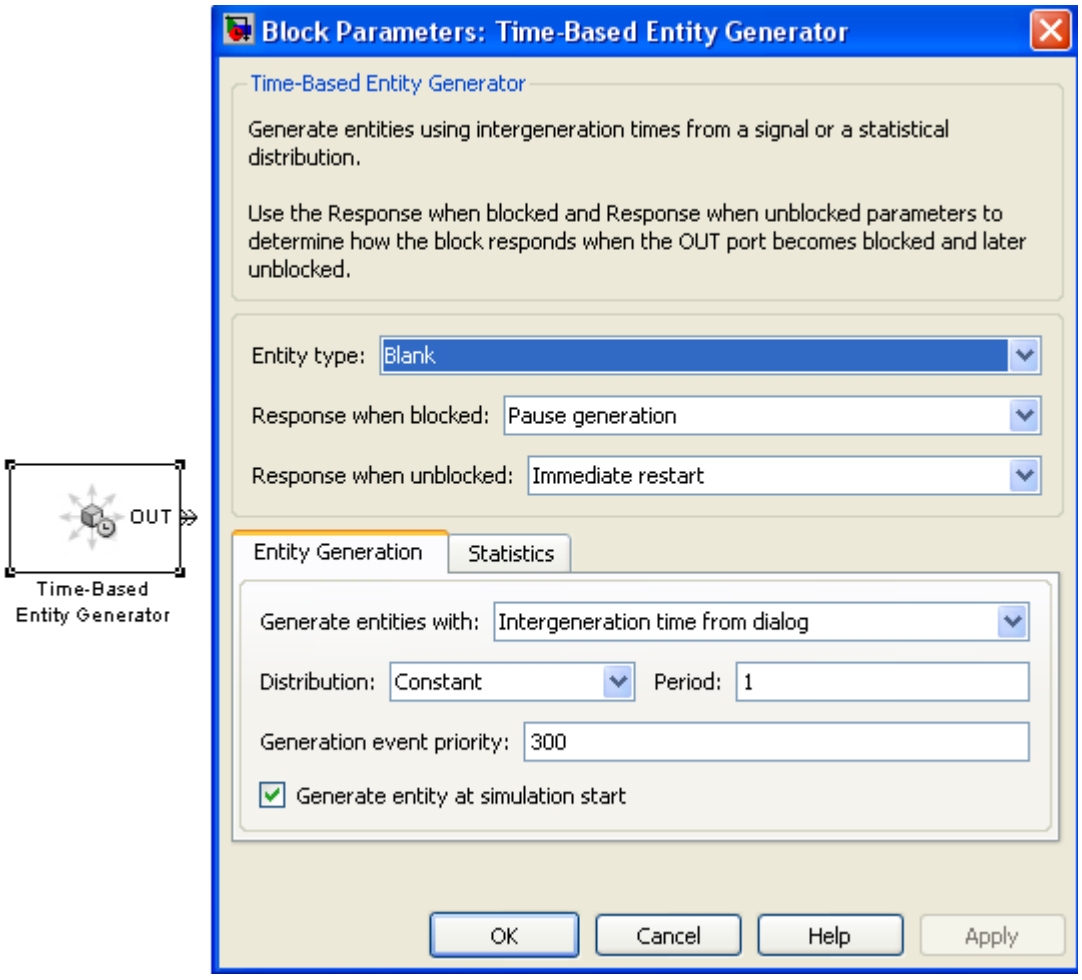


Рис. 18 Блок Time-Based Entity Generator та його діалогове вікно

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 встановлюються початкове значення для генератора випадкових чисел:

Initial seed – невідємне ціле значення. Як правило це число визначається як непарне число з великим значенням.

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

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

Generation event priority – пріорите суності у процессі моделювання.

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

Приклад. На рис. 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).

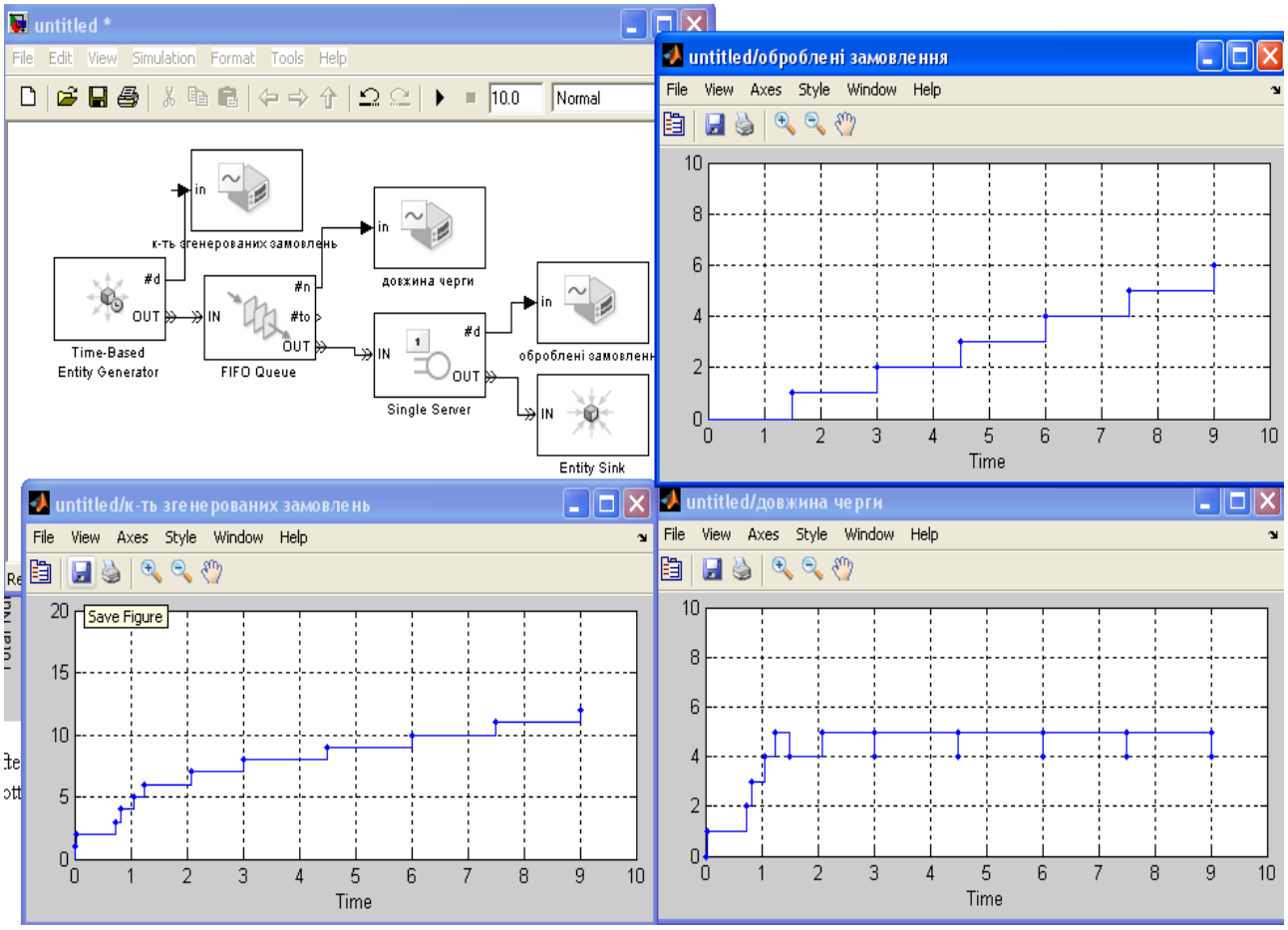


Рис. Прикдад моделі з використанням блоку Time-Based Entity Generator

Event Generators – генератори подій

Entity Based Function Call Generator

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

Поле **Generate function call** дозволяє задавати виклик функція 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**.

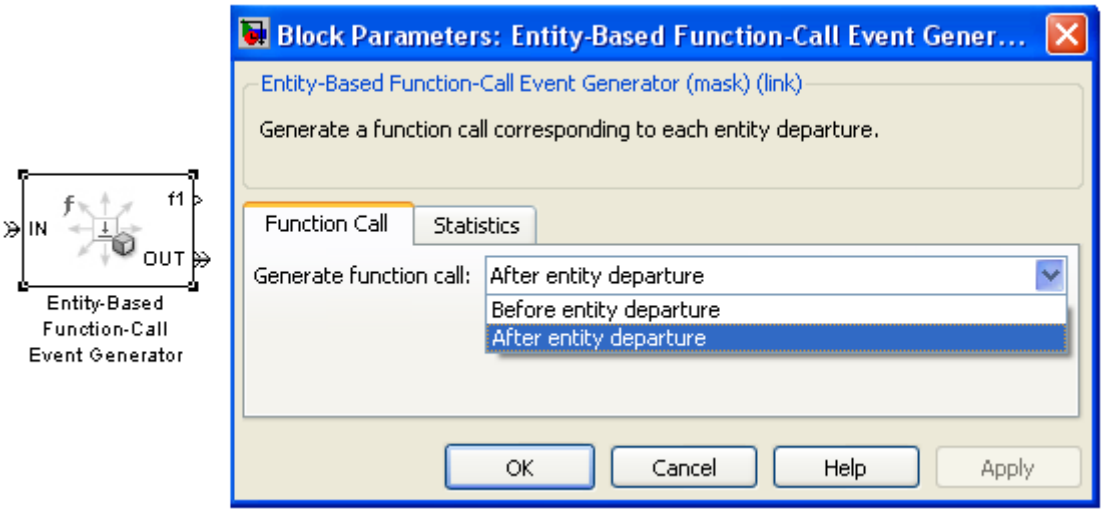


Рис. Блок Entity Based Function Call Generator та його діалогове вікно

Приклад. На рис. наведено приклад моделі, яка виконує обчислення, коли сутність надходить на вхід блоку обєднання сигналів (Path Combiner) IN2 або IN3 (з пдругої 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.

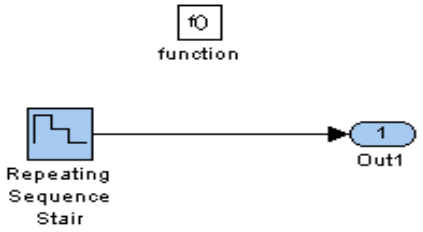


Рис. Структура блоку Function-Call Subsystem

Блок Mux обєднує 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.) 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).

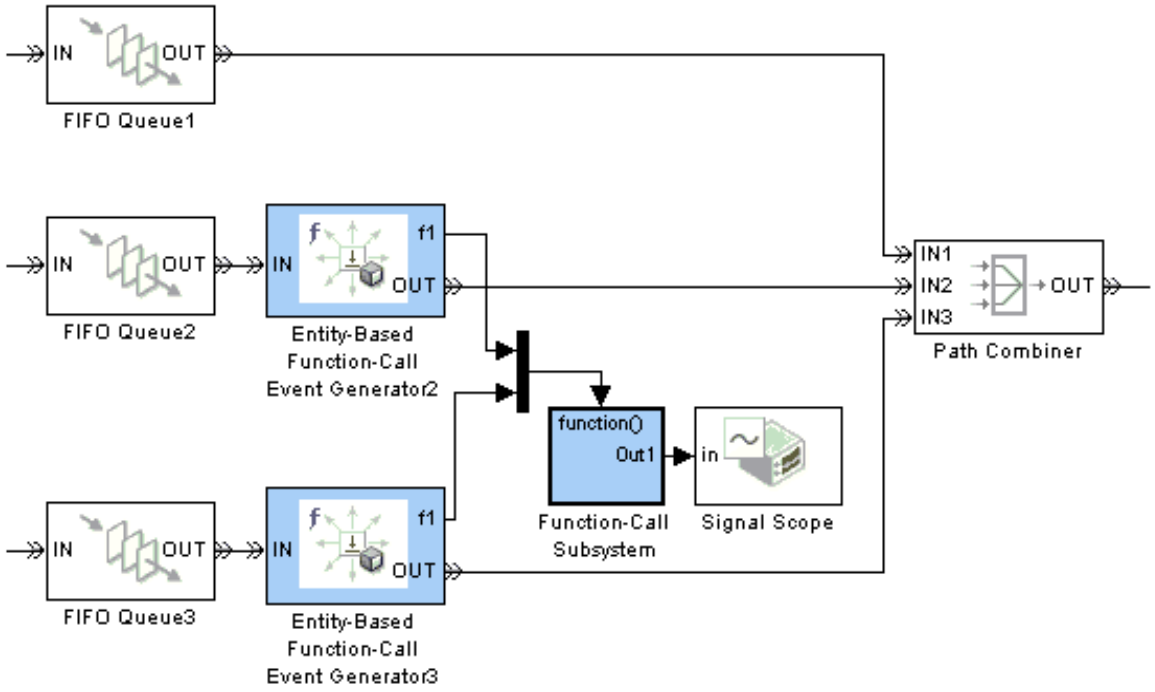


Рис. Приклад фрагменту моделі з використанням блоку Entity Based Function Call Generator

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.

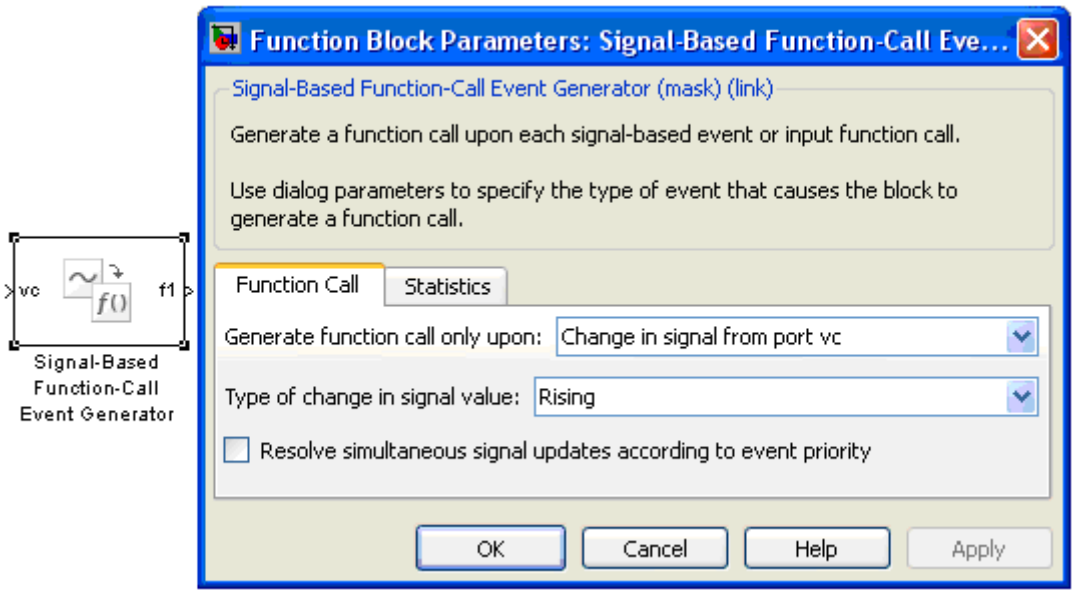


Рис. Блок Signal Function Call Event Generator

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 (виклику 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. 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.

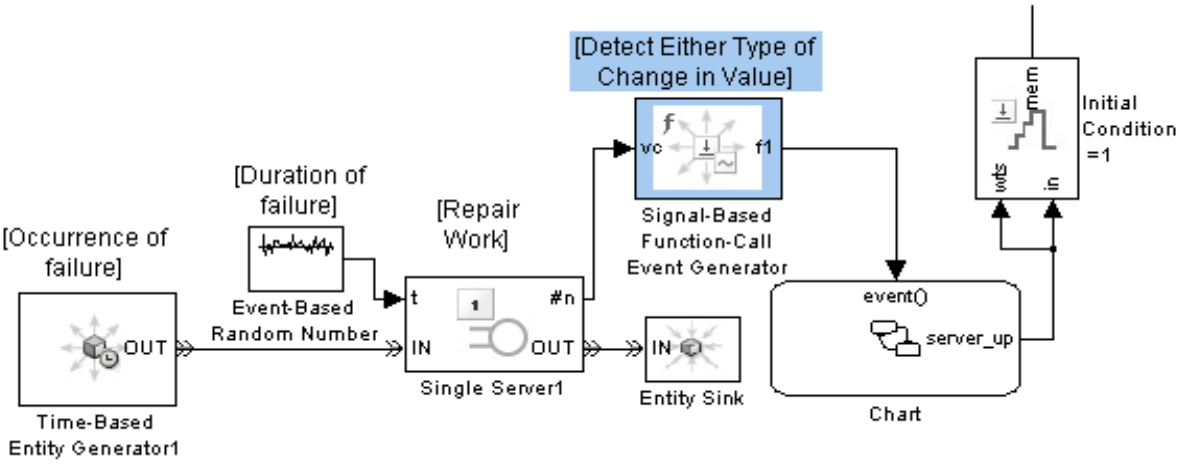


Рис. Приклад фрагменту моделі з використанням блоку Signal Function Call Event Generator

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

Signal Generators – генератори сигналів

Event Based Random Number

Event Based Random Number – блок генерування випадкових чисел. Блок генерує випадкове число в залежності від події, повязаної із натупним

блоком, наприклад, з блоком Single Server, то 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).

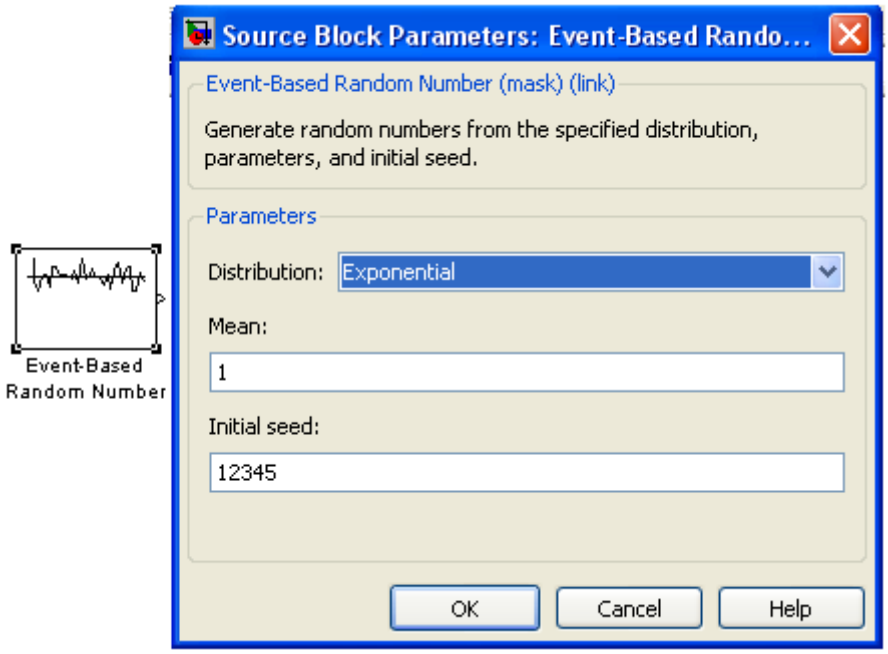


Рис. Блок Event Based Random Number

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) $f(x)=\begin{cases}\frac{n!}{x!(n-x)!}p^x(1-p)^{n-x} & x=0,1.2,...,n\\ 0 & \textit{інакше}\end{cases}$
Triangular	Minimum – L, Maximum – U, Mode – m (мода) $f(x)=\begin{cases}\frac{2(x-L)}{(U-L)(m-L)} & L\le x\le m\\ \frac{2(U-x)}{(U-L)(U-m)} & m\le x\le U\\ 0 & \textit{інакше}\end{cases}$
Gamma (гамма)	Threshold – θ, Scale – b, Shape – a, $\Gamma(\gamma)$ – гамма ф-ція $f(x)=\begin{cases}\frac{\left(\frac{x-\theta}{b}\right)^{a-1}\exp\left(\frac{x-\theta}{b}\right)}{b\Gamma(\gamma)} & x\ge 0\\ 0 & \textit{інакше}\end{cases}$
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) $f(x)=\begin{cases}\frac{\exp(-\lambda)\lambda^x}{x!} & x=0,1.2,...\\ 0 & \textit{інакше}\end{cases}$
Lognormal (логарифмічно-нормальний)	Threshold – θ, Mu – μ, Sigma – σ $f(x)=\begin{cases}\frac{\exp\left(-\left(\frac{\ln(x-\theta)-\mu}{2\sigma^2}\right)^2\right)}{(x-\theta)\sigma\sqrt{2\pi}} & x\ge 0\\ 0 & \textit{інакше}\end{cases}$
Log-logistic ()	Threshold – θ , Scale – b $f_{\text{logistic}}(x)=\frac{1}{b}\cdot\frac{\exp((x-\theta)/b)}{[1+\exp((x-\theta)/b)]^2}$
Beta (бета)	Minimum – L, Maximum – U, Shape parameter – a, Shape parameter – b $f(x)=\begin{cases}\frac{(x-L)^{a-1}(U-x)^{b-1}}{(U-L)^{a+b+1}B(a,b)} & L\le x\le U\\ 0 & \textit{інакше}\end{cases}$, $B(a,b)=\int_0^1 t^{a-1}(1-t)^{b-1}dt$
Discrete uniform	Minimum – L, Maximum – U, Number of values – K

(дискретний рівномірний)	$f(x)=\begin{cases}\frac{1}{K} & x=L+k\frac{U-l}{K-1},\ k=0,1,2,...,K-1\\ 0 & \textit{інакше}\end{cases}$
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 $P(\Delta t=1)=0.25$, $P(\Delta t=1.5)=0.5$, $P(\Delta t=2)=0.25$.

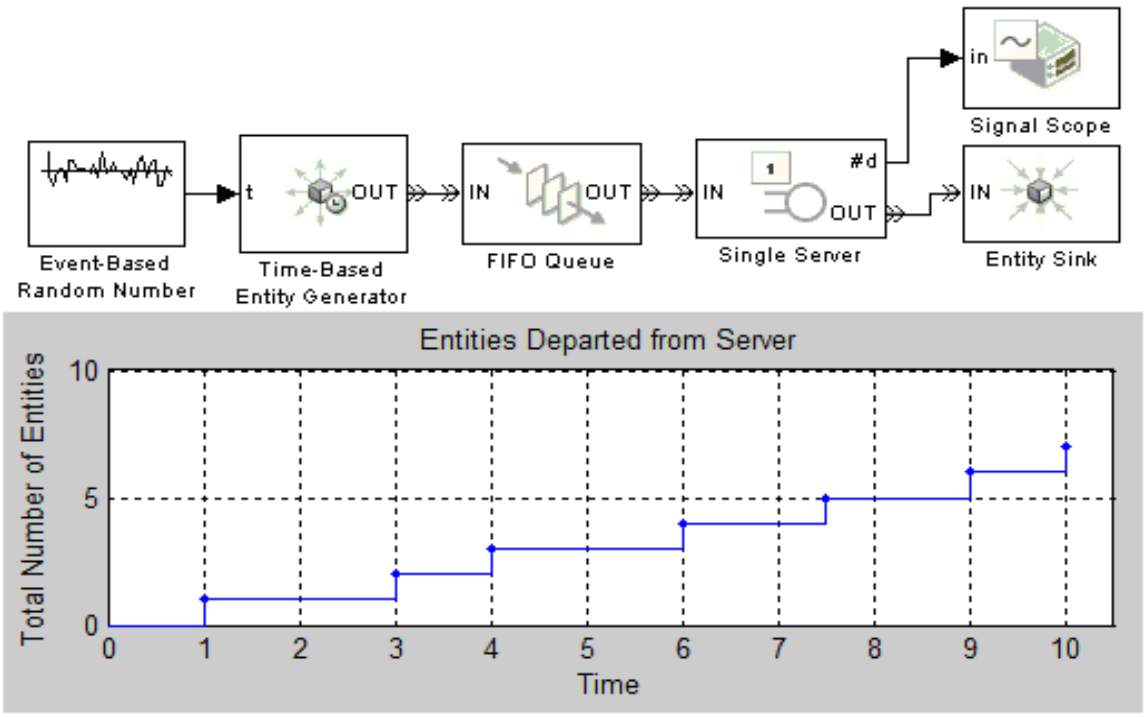


Рис. Приклад застосування блоку Event Based Random Number

Для генерація замовлень використовується блок Time-Based Entity Generator з наступними налаштуваннями: Generate entities with – порт t, який зєднанує даний блок з блоком Event Based Random Number (рис.).

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

Event Based Sequence

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

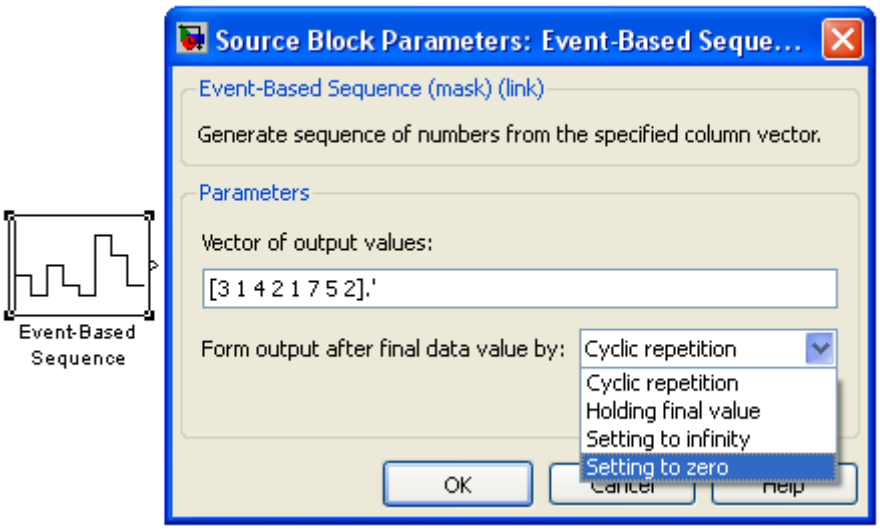


Рис. Блок Event Based Sequence

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 з заходу на основі послідовності блоку 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 ортів 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
	Event-Based Entity Generator	відповідний тригер з порту tr
	Release Gate	відповідне значення з порту vc
	Signal-Based Event to Function-Call Event	
	Signal-Based Function-Call Event Generator	
wts, wtr, wvc, rts,	Signal Latch	проміжок часу з порту wts або

rtr, rvc		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, яка 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.

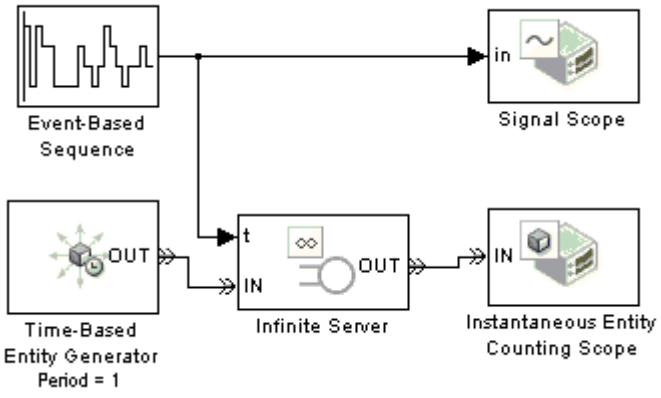


Рис. Приклад використання блоку Event Based Sequence

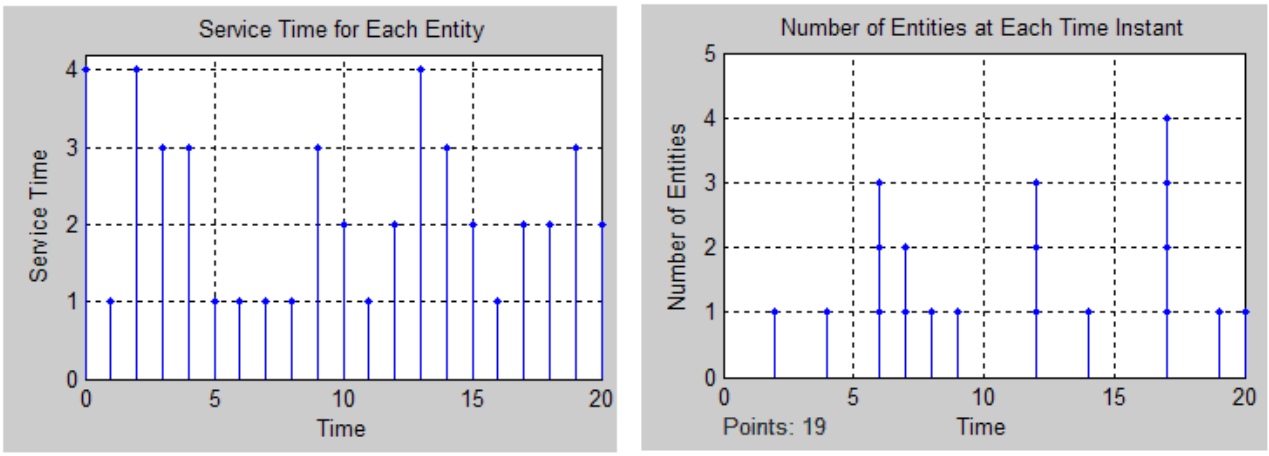


Рис. Результат роботи моделі з використанням блоку Event Based Sequence

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.

Черга (Queue)

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

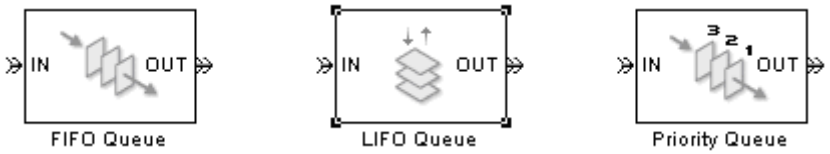


Рис.

- FIFO (first input first ounrun) – черга, яка обслуговується за принципов " 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. Тшльки блок Priority Queue містить на першій вкладці два додаткові поля, де визначається дисципліна пріоритету та порядок сортування (рис.)

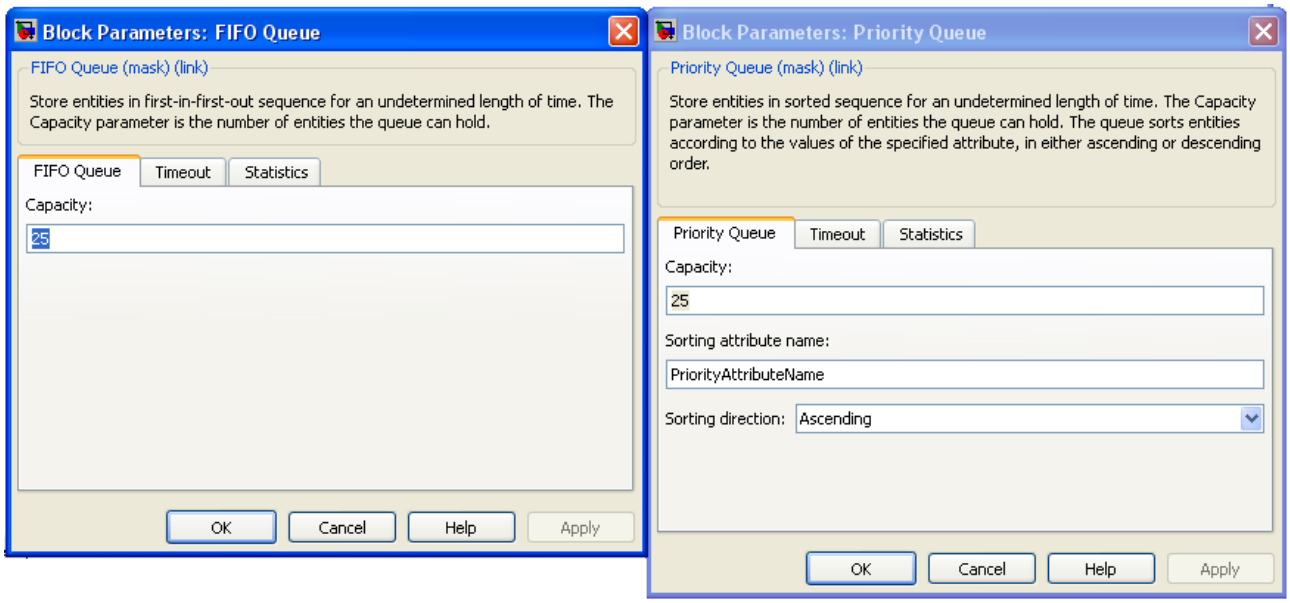


Рис. Головна закладка діалогового вікна блоків черги

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 боле 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. і містить наступні поля:

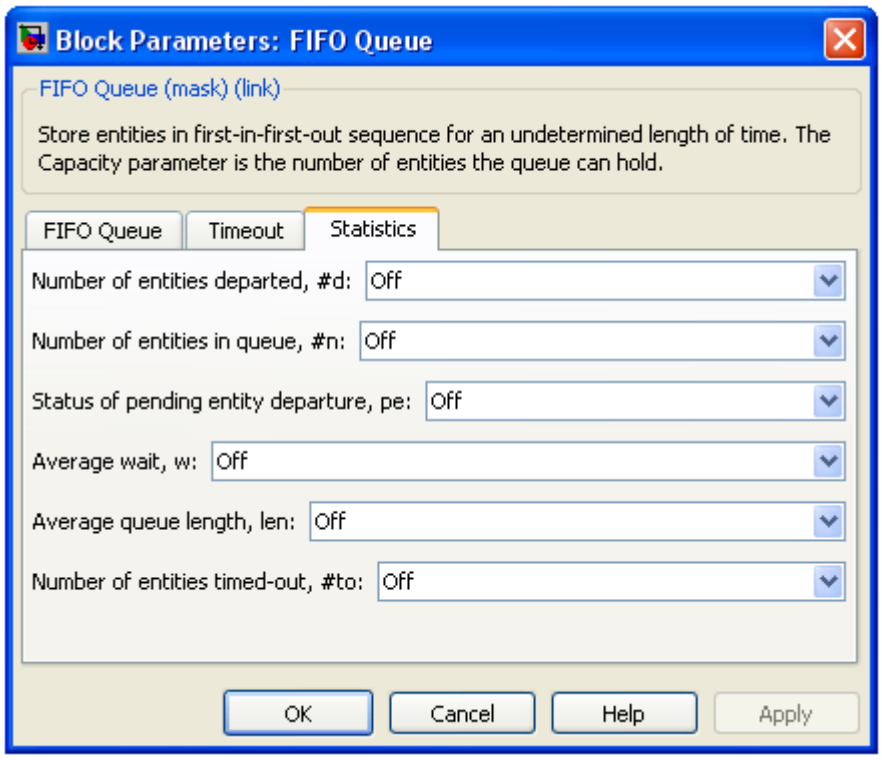


Рис. Налаштування статистики блоків черги

- 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 вихідного порту сигналу з написом **pe.**
- Average wait, w – середній час очікування у черзі;
- Average queue length, len – середня довжина черги на у часі;
- Number of entities timed out, #to – контролює присутність та поведінку вихідного порту.

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

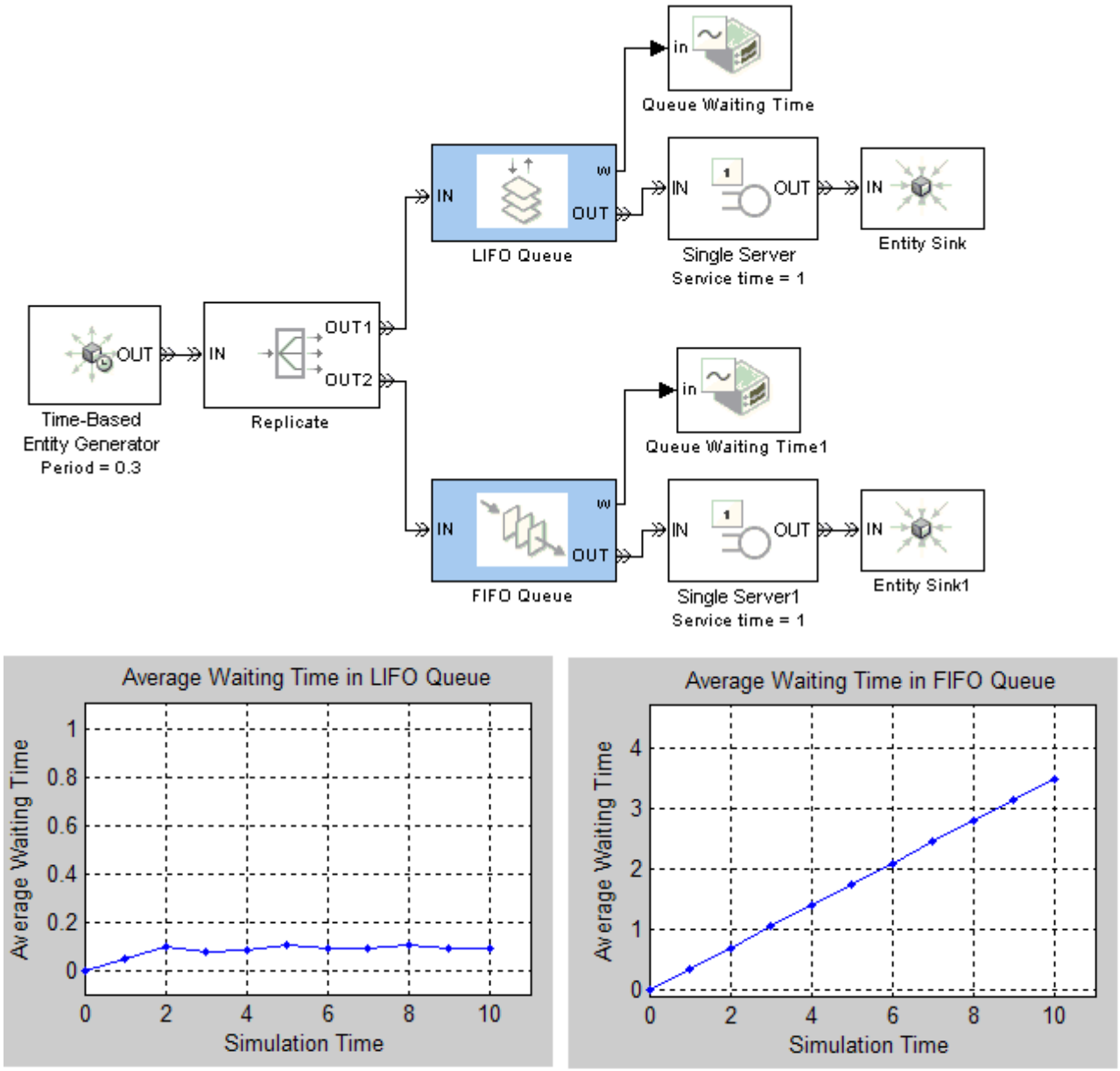


Рис. Приклад моделі з застосуванням черг та результатів її роботи

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

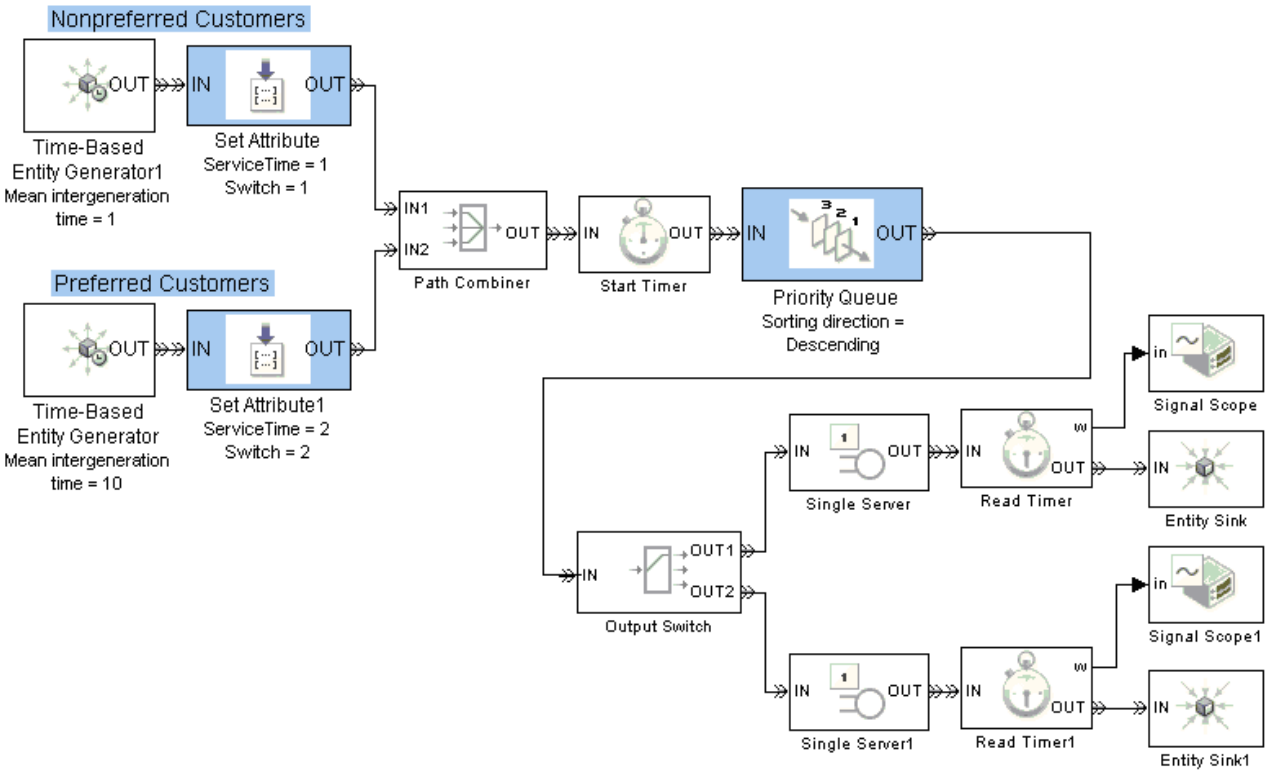


Рис. Приклад моделі з чергою замовлень з пріоритетами

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

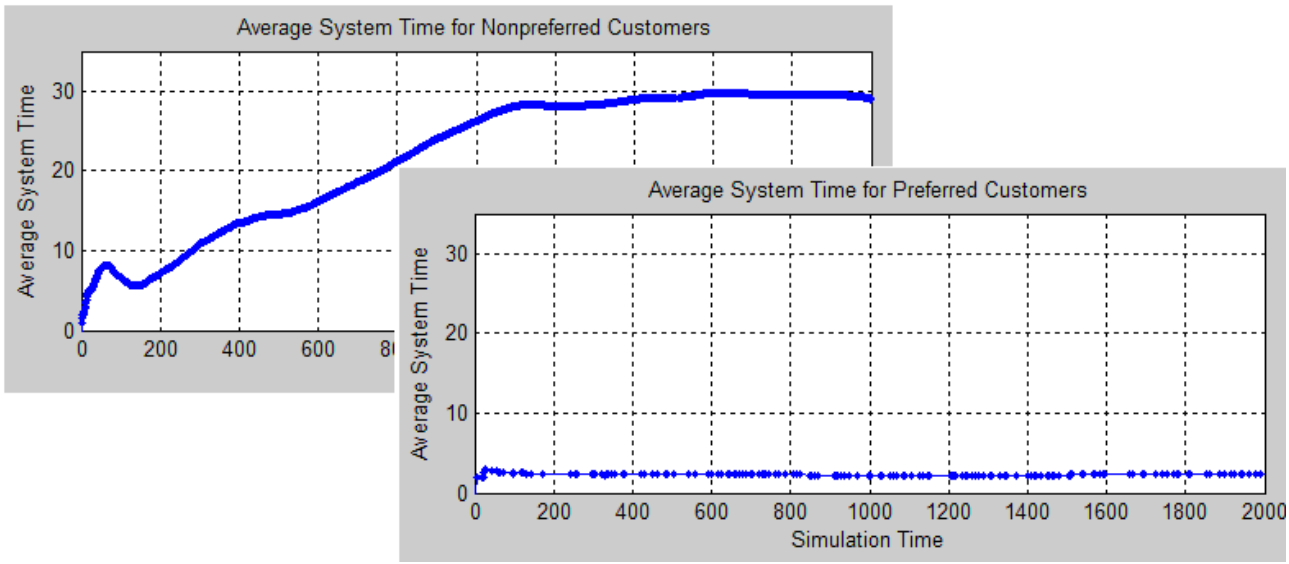


Рис. Результат прогону моделі з чергою замовлень з пріоритетами

Атрибути (Attributes)

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

Бібліотека містить два графічних блоки (рис.):

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

Кожний атрибут має своє ім'я та значення.

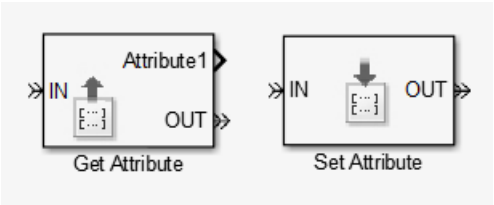


Рис. Блоки управління атрибутами

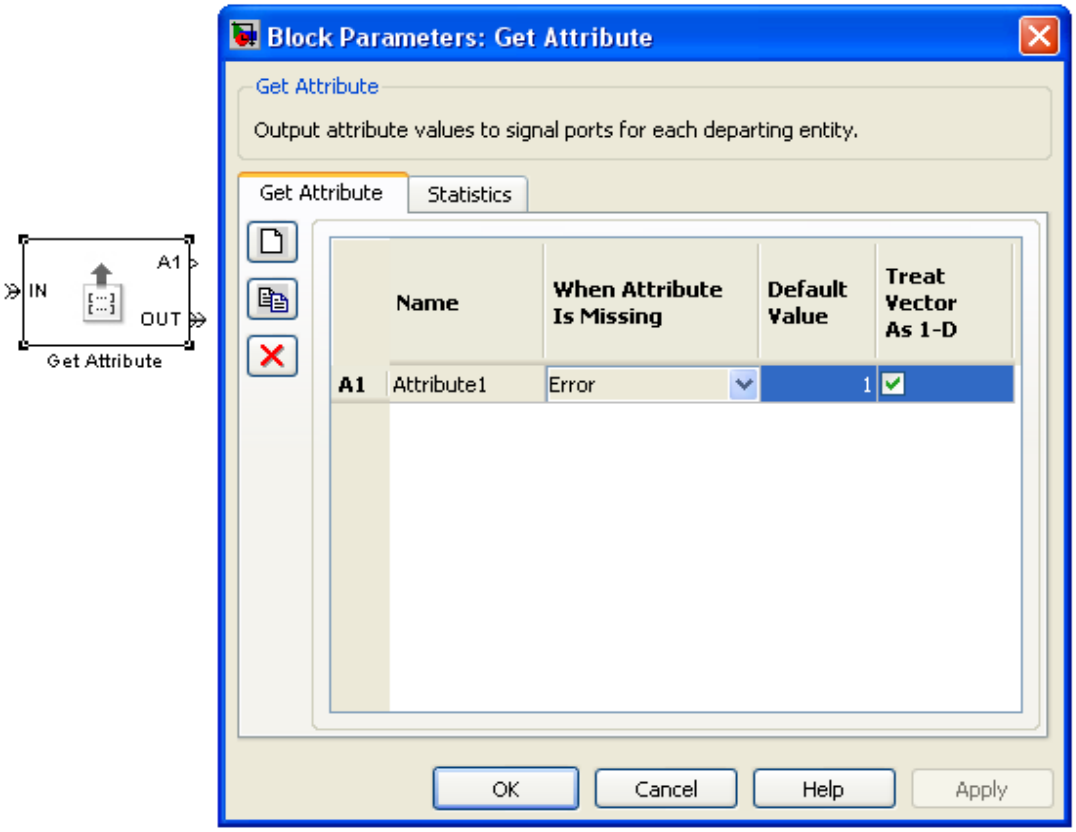


Рис. Блок визначення атрибутів Get Attribute та його діалогове вікно

На панелі блоків є наступні клавіші налаштування:

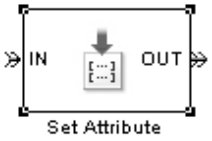
 – створити поле атрибута;

 – копіювати поле атрибута;

 – видалити поле атрибута

та дві закладки:

- налаштування параметрів атрибуту;
- статистика.



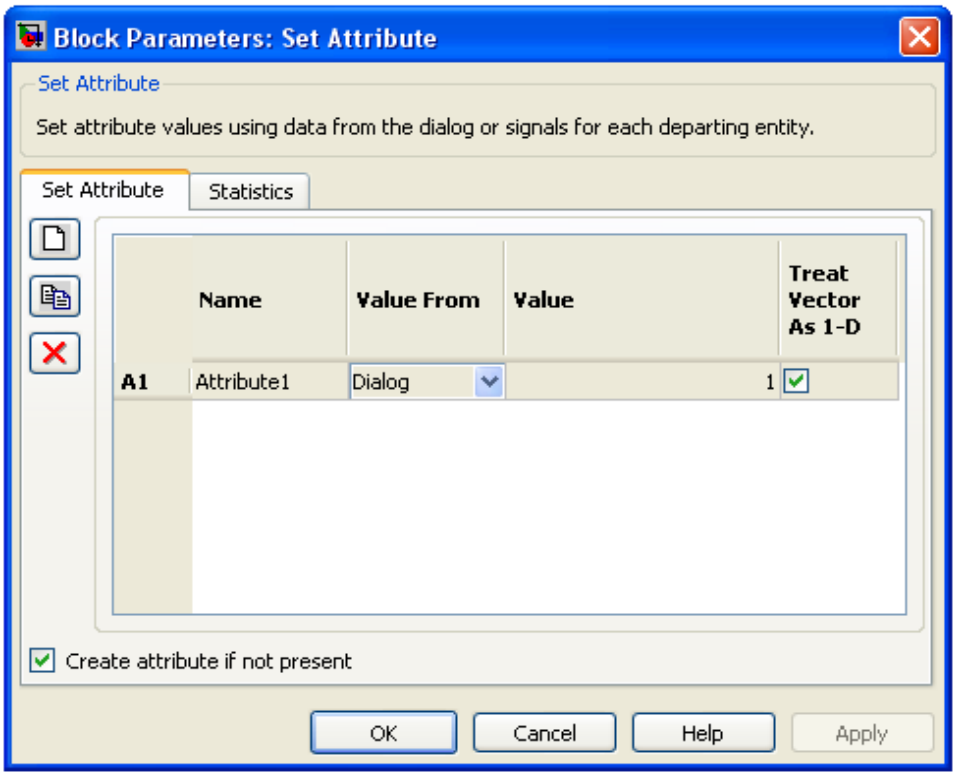


Рис. Блок встановлення атрибутів Set Attribute та його діалогове вікно

Параметри налаштування представлені у вигляді таблиці з наступними полями:

- Name – імя 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.

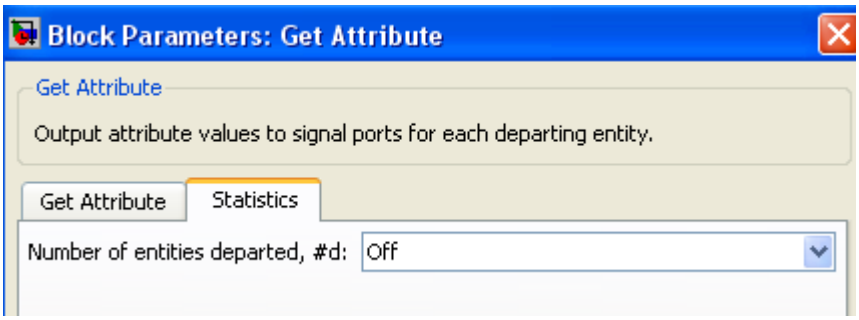


Рис. Закладка статистики блоку атрибутів

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.)

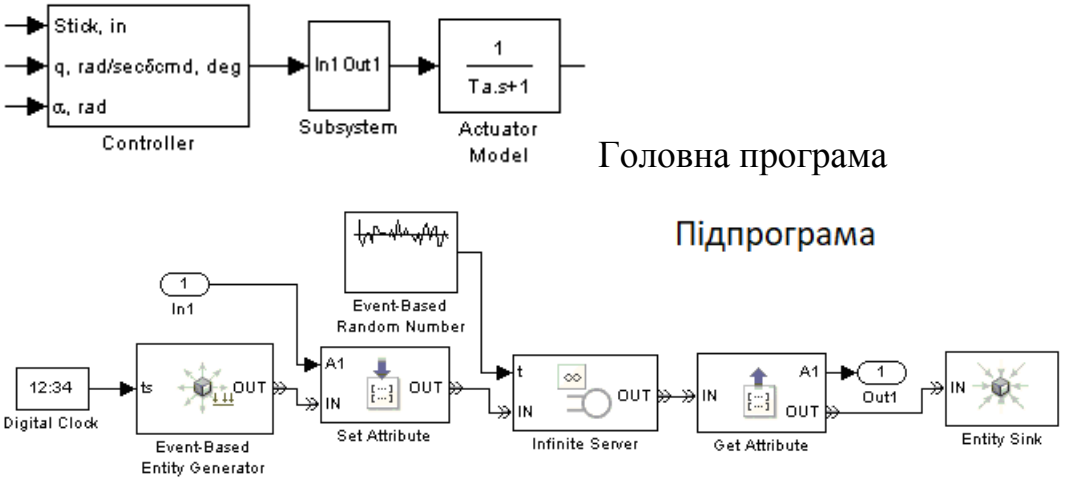


Рис. Фрагмент моделі СМО з використанням блоків атрибутів

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

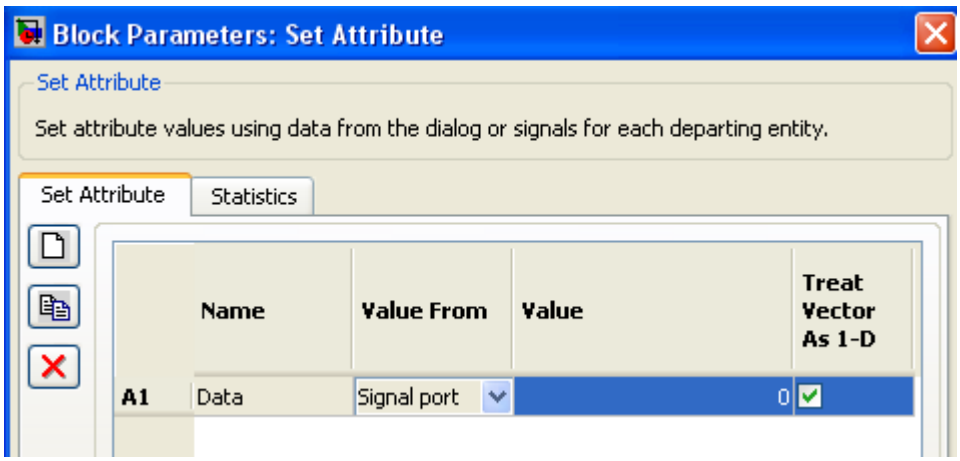


Рис. Приклад задання атрибуту

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.

Управління потоками (Entity Management)

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

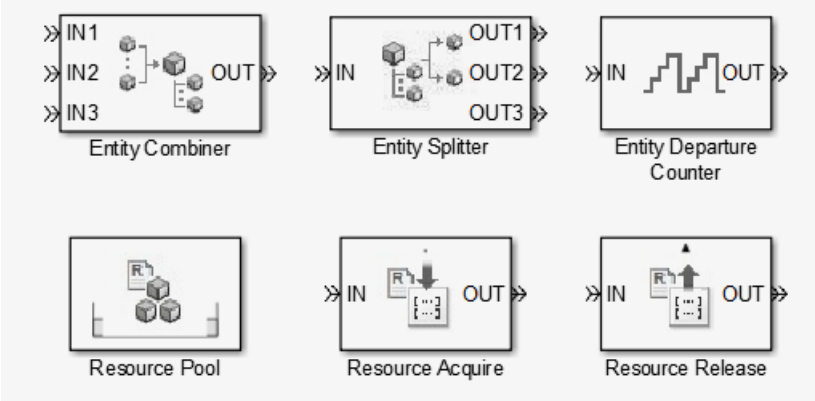


Рис. Графічні блоки управління потоками

В даному розділі розглянемо основні блоки:

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

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 до параметрів сутності і можливість зворотньої операції – роз'єднання. Деякі значення параметрів вимагають унікальності імен атрибутів або тегів таймеру.

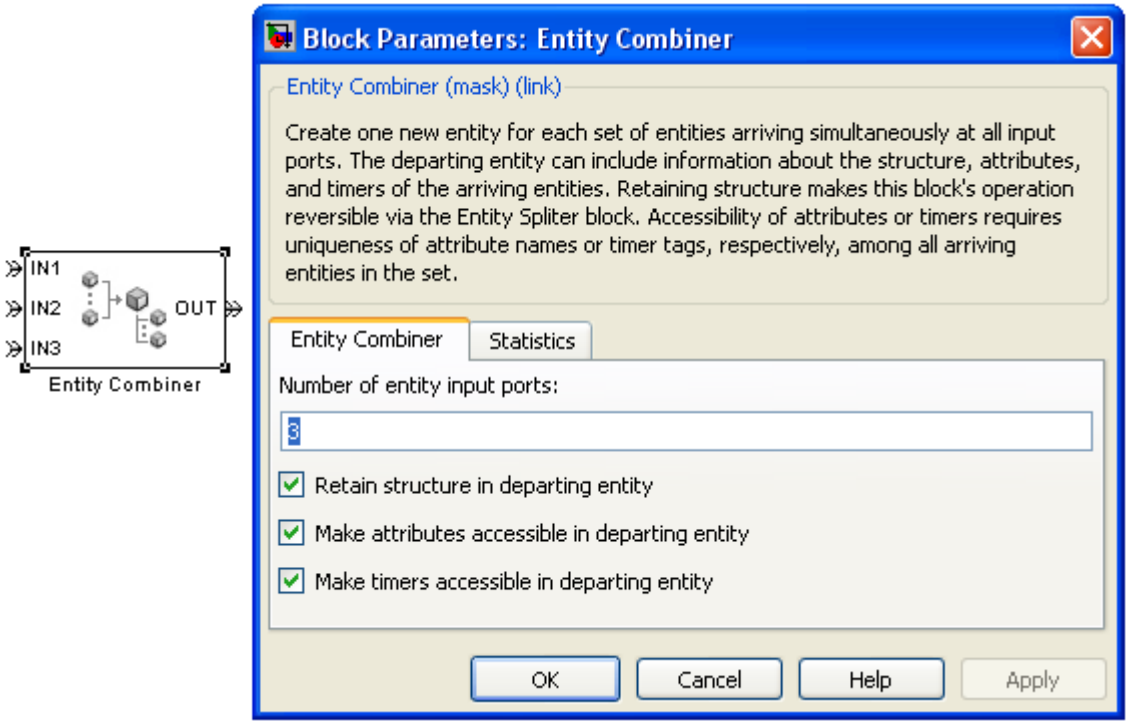


Рис. Блок Entity Combiner

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 обєкти компоненту за допомогою блоку Entity Splitter;

- Make attributes accessible in departing entity – вибір цієї опції дає доступ до атрибутів компонентів вихідних сутностей. Імя поля залежить від вибору поля Retain structure in departing entity;
- Make timers accessible in departing entity, – вибір цього поля дозволяє використовувати таймери вихідних сутнностей. Імя 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).

Entity Splitter (роз'єднання потоків) – блок розділяє складену сутність, утворену задопомогою блоку обєднання яка надходить на порт IN на компоненти, які виходять через відповідні порти OUT (рис.).

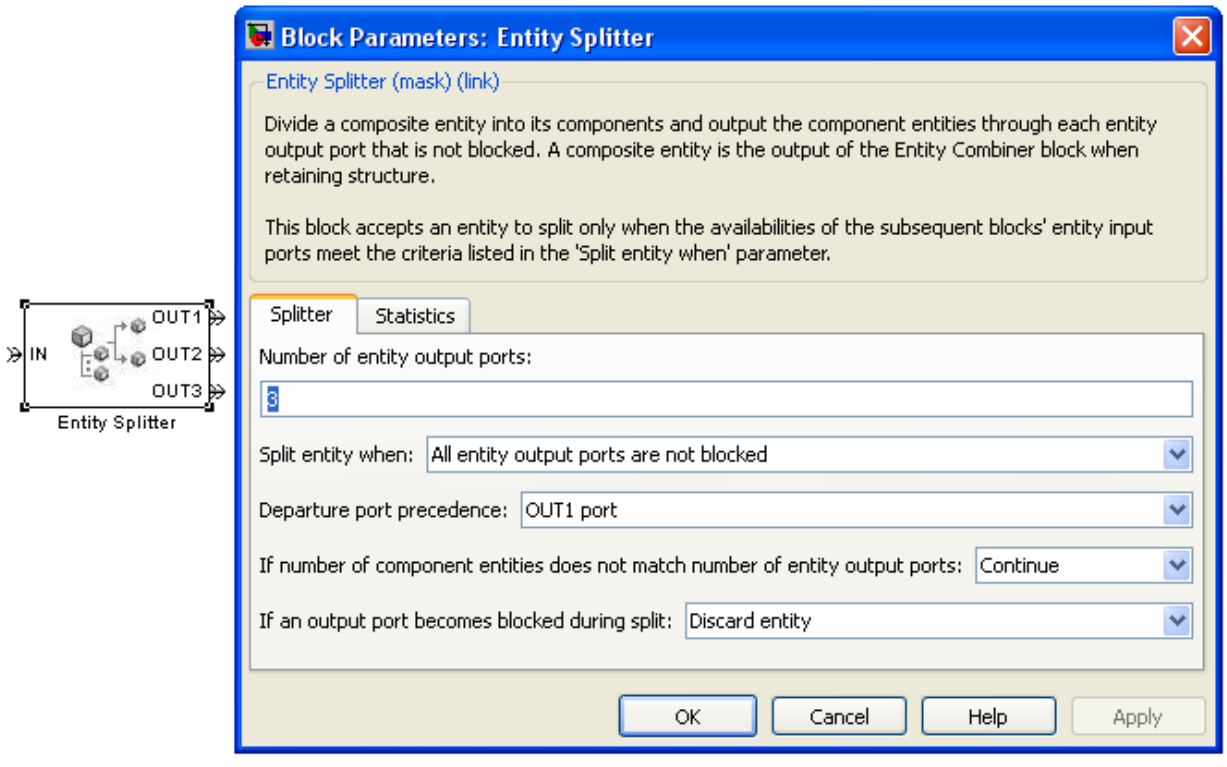


Рис. Блок Entity Splitter

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

На головні й 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 розєднанні потоків. Значення поля наведені у таблиці.

Таблиця

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

	однакову ймовірність вибору	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

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

Позначення	Призначення	Час поновлення статистики	Порядок поновлення
#a	Кількість сутностей, що надійшли у блок з початку процесу моделювання	Після надходження сутності	1

#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 рівним 0.

Приклад. У наведеній нижче моделі обєднуються 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 у середині структури .

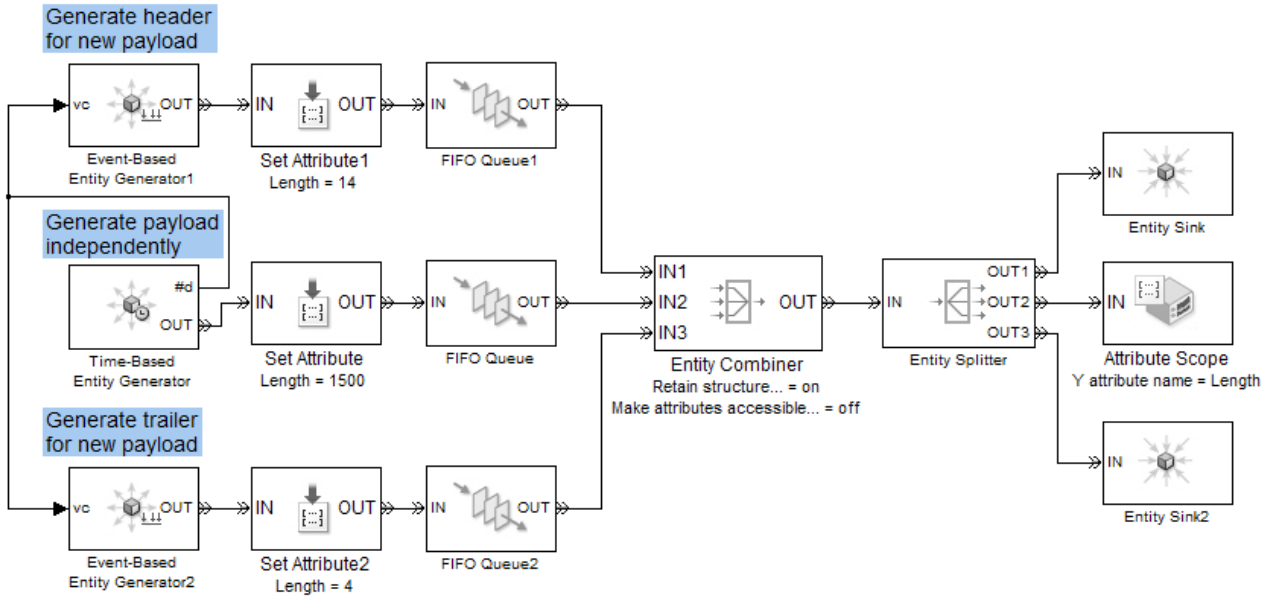


Рис. Приклад використання блоків обєднання та розєднання потоків

Управління сигналами (Signal Management)

Блоки цієї бібліотеки дозволяють управляти сигналами і складається з двох блоків:

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

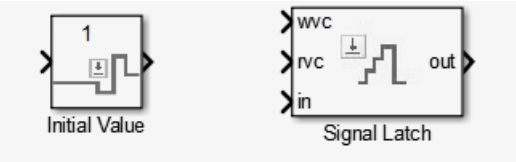


Рис. Блоки управління сигналами

Ininial value (Початкове значення) – блок встановлює початкове значення для сигналу на основі події (рис.). Для першого значення входу та всіх наступних надходжень вихідний сигнал ідентичний вхідному сигналу.

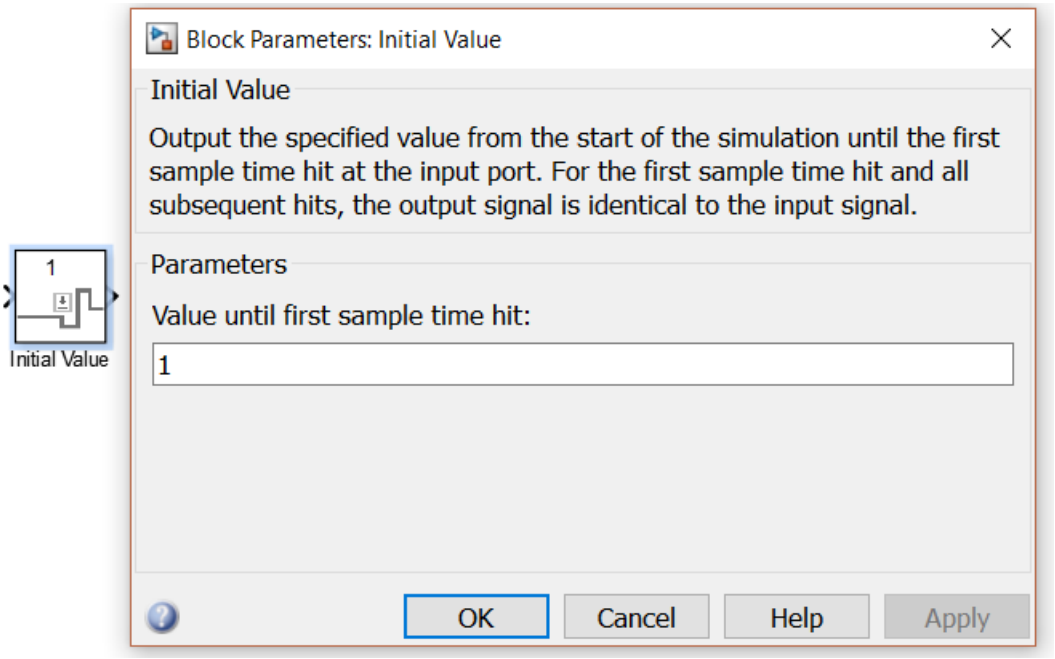


Рис. Блок Ininial value та його діалогове вікно

Блок має один вхідний і вихідний порт 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.). Коли розпочинається симуляція, блок початкового значення забезпечує вихідне значення 1, яке відкриває ворота,

щоб перша сутність могла перейти до циклу зворотного зв'язку. Без ненульового початкового значення, жодна сутність не надійде на сервери, а блок блокування сигналу ніколи не зазнає жодних подій.

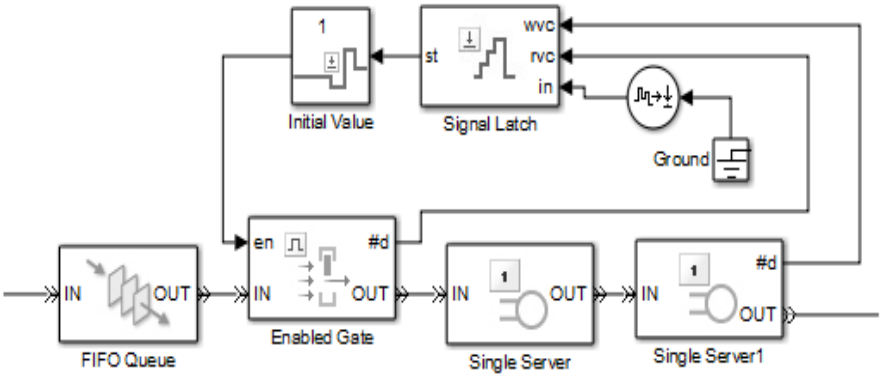


Рис.Фрагмент моделі з використанням блоку Ininial value

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**".

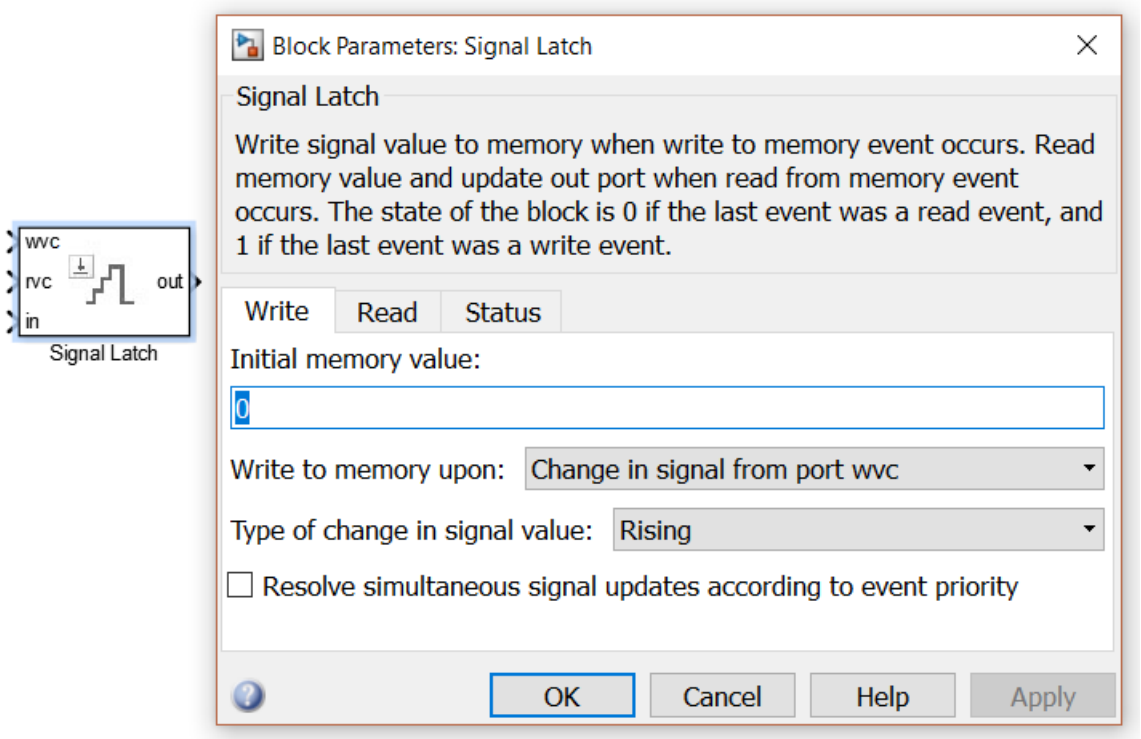


Рис. Блок Signal Latch та його діалогове вікно

Цей блок корисний для моделювання циклів зворотного 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 наборо вхідних та вихідних портів (табл.. , табл..).

Таблиця Вхідні порти блоку Signal Latch

Позначення	Опис
wts	Сигнал, що ініціює запис подій. Цей сигнал повинен бути сигнал 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 сигнал на основі події. Порт доступний, якщо встановлено **Write to memory**.
wvc	Сигнал, зміна значення якого призводить до запису подій. Цей сигнал повинен бути сигнал 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, чиї оновлення викликає читання подій. Цей сигнал повинен бути сигнал на основі події. Порт доступний, якщо встановлено **Read from memory upon**.
rtr	Тригерний сигнал, що викликає читання подій. Цей сигнал повинен бути сигнал на основі події. Порт доступний, якщо встановлено **Read from memory upon**.
rvc	Сигнал, зміна значення якого призводить до читання подій. Цей сигнал повинен бути сигнал 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 сигнал на основі події.

Таблиця Вихідні порти блоку Signal Latch

Позначення	Опис	Час оновлення	Початкове значення
st	0 або 1, залежно від	При читанні	0

	операції, яку виконує блок: читання чи запис.	або записі події	
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.):

- **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, спмданням 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. Такох у діалоговому вікні зявиться додаткове поле **Event priority for reading from me-mory** (Пріоритет події для запису в пам'ять), яке приймає цілочисельне значення. Якщо 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.). 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**.

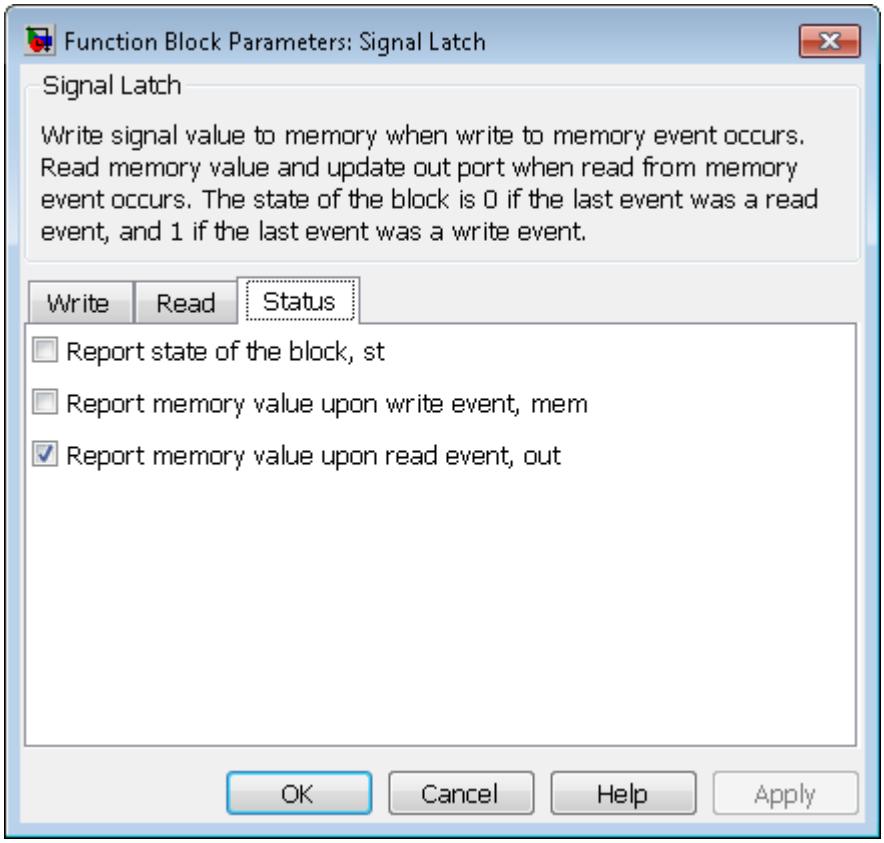
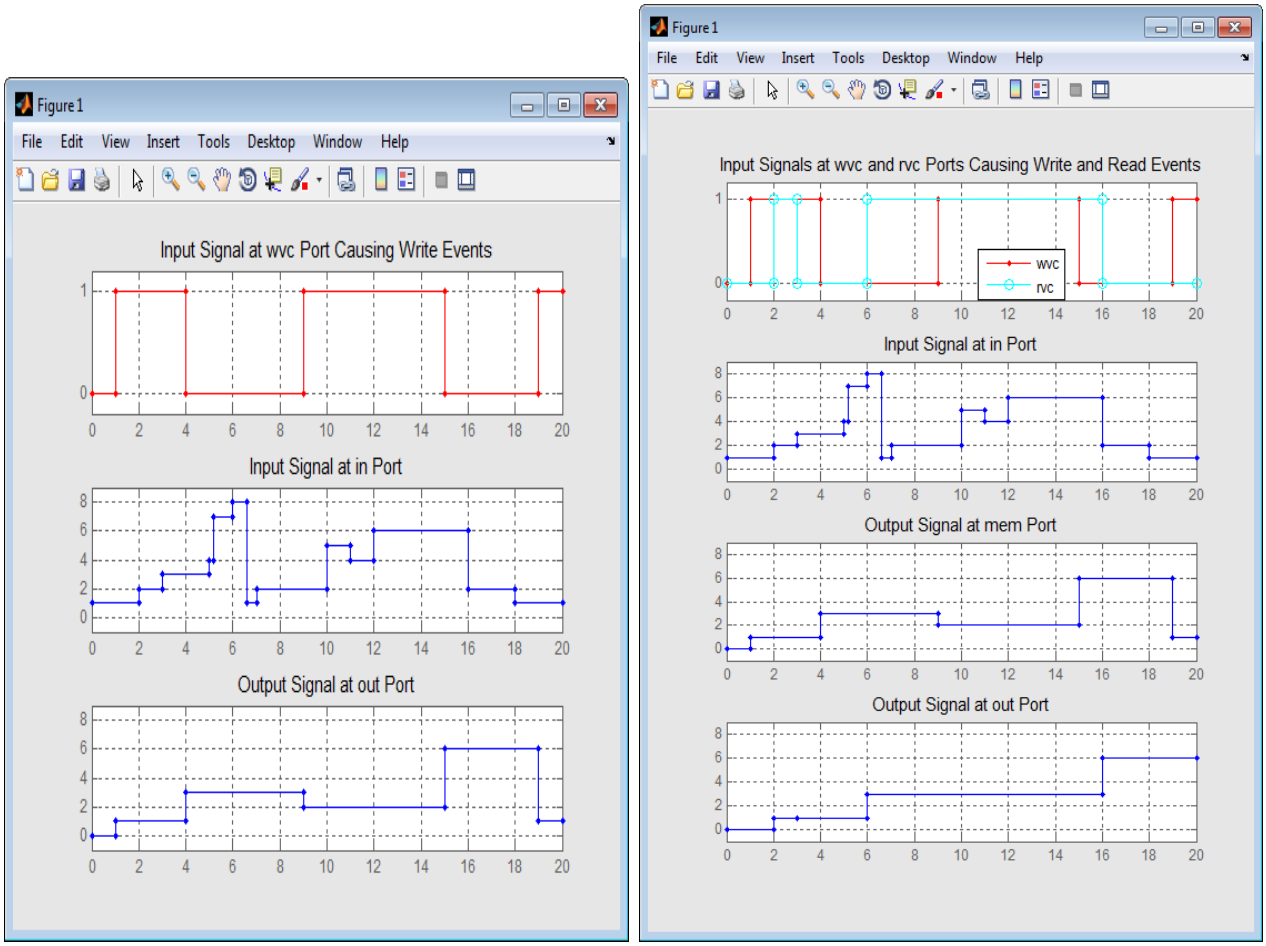


Рис. Закладка Status блоку Signal Latch

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. 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**.

Ворота (Gates)

Блоки цієї бібліотеки дозволяюь управляти потоками в залежності від заданих умов і складається з двох блоків (рис.):

-Enabled Gate – дозволяючі ворота;

- Release Gate – пропускаючі ворота.

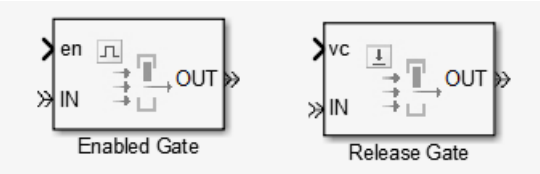


Рис. Блоки бібліотеки Gates

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.

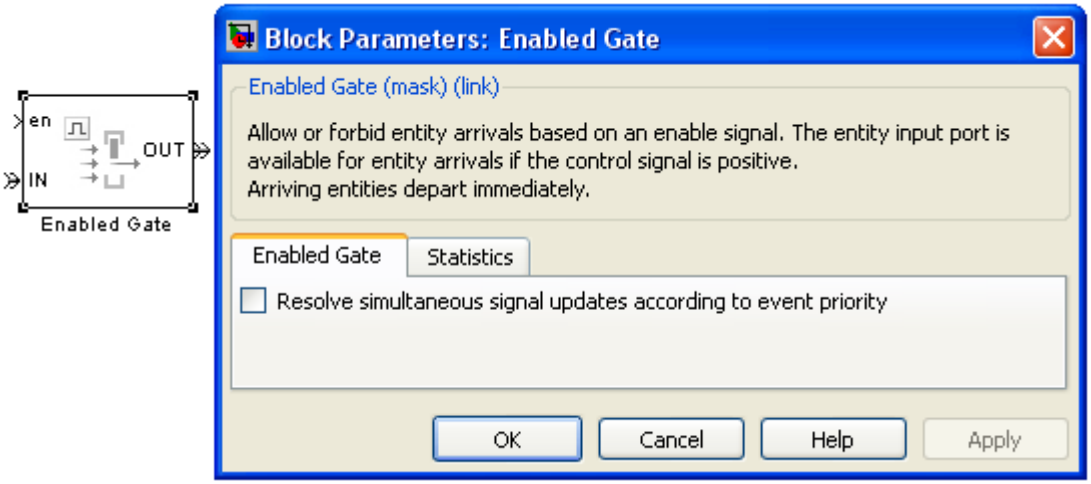


Рис. Блок Enabled Gate

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.

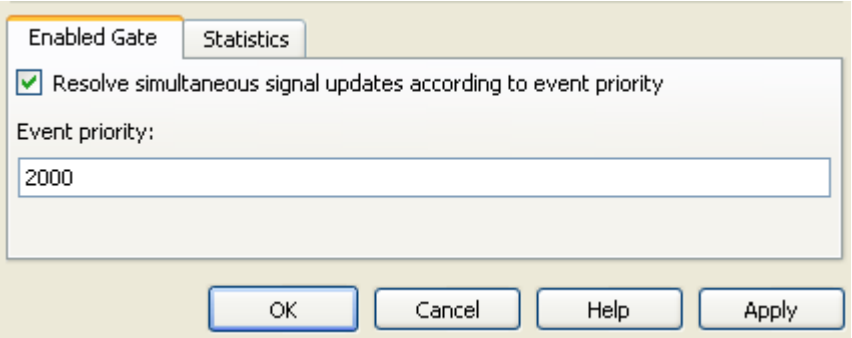


Рис. Блок Enabled Gate із встановленням приорітету

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.*

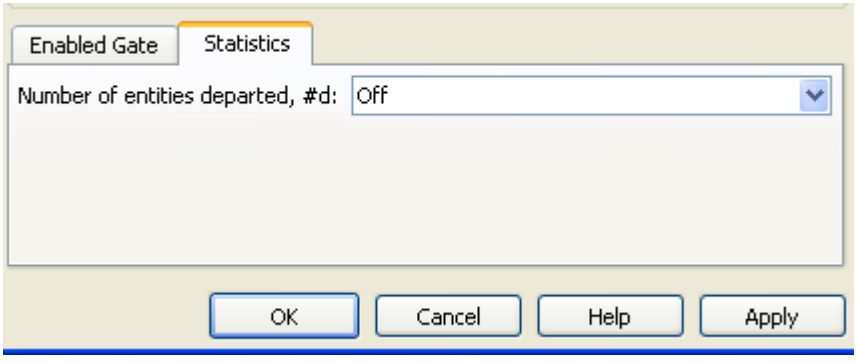


Рис. Закладка статистики блоку Enabled Gate

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 зєднані 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 ворота блок знову відкриває потік. Модель такої системи наведена на рис.

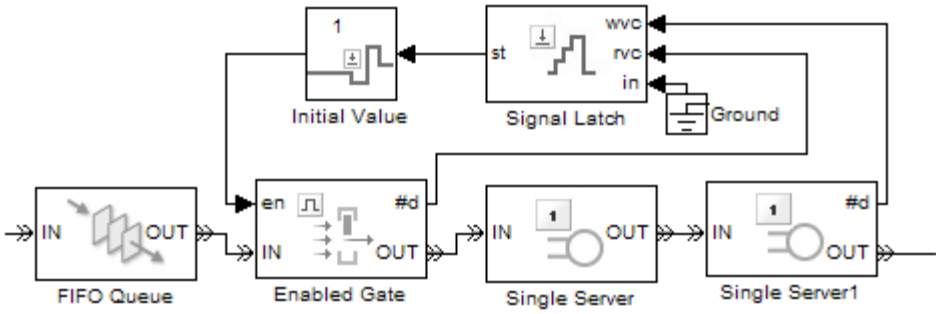


Рис.

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*).

Release Gate (пропускаючі ворота) – блок відкриває ворота на основі події-сигналу або виклику функції (рис.). 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.

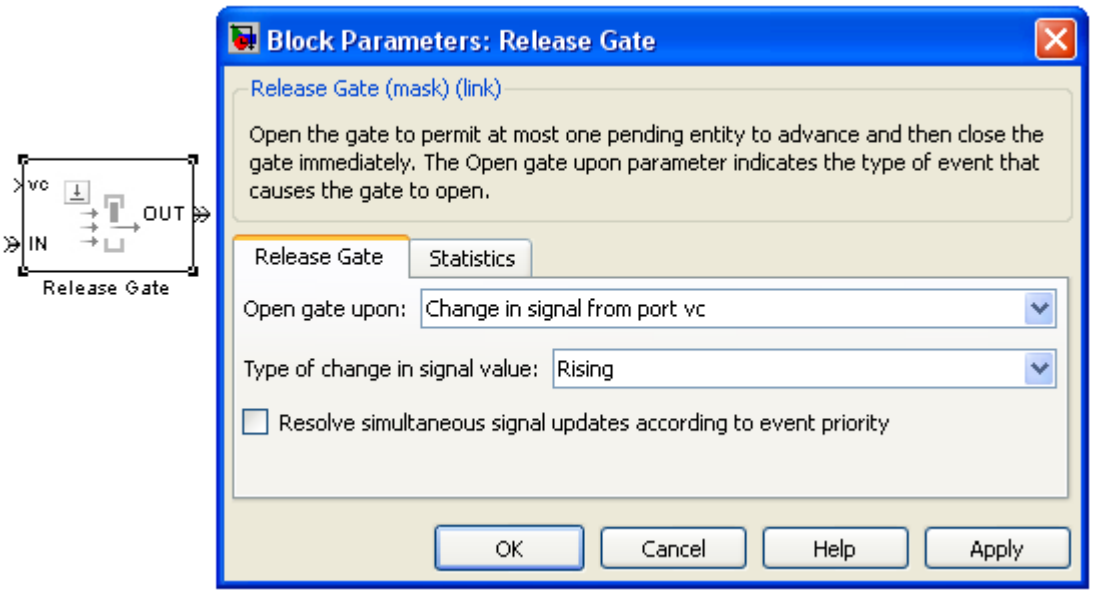


Рис. Блок Release Gate

Блок має два вхідних та один вихідний сигнали.

Вхідний сигнал IN та вихідний сигнал OUT мають такіж значення, як і для блоку **Enabled Gate**. Для визначення другого вхідного сигналу та настроювання блоку застосовується діалогове вікно (рис.), 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 змінна 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 встановленний період в 1.

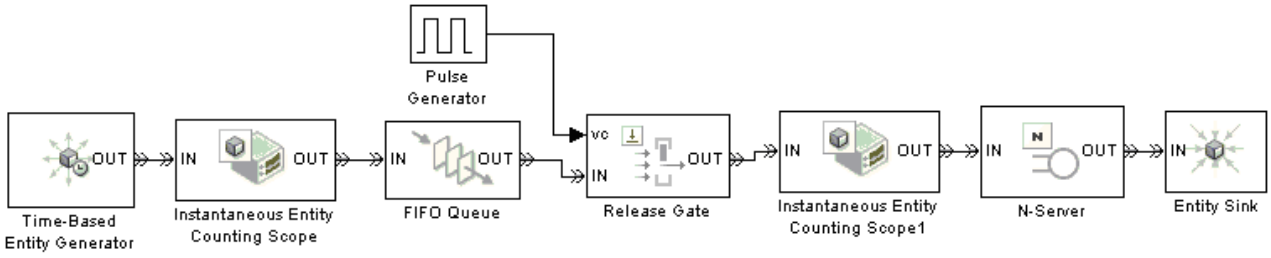


Рис.

Графіки нижче показують, що Час Генерування Сутностей може бути не цілим числом, проте Час Початку Сервісу завжди цілі числа.

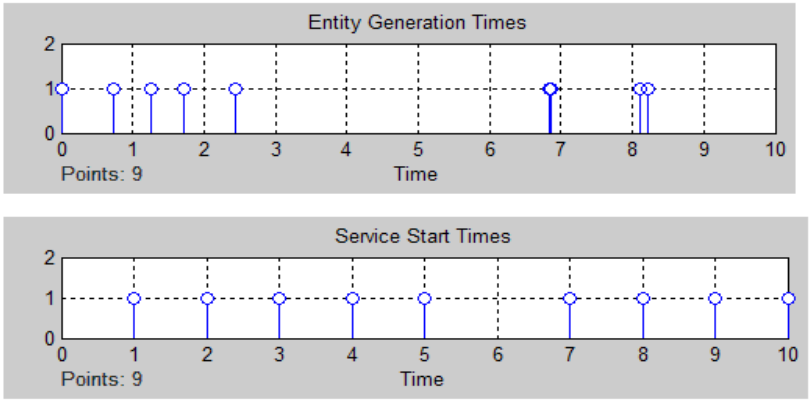


Рис. Графіки часу генерування сутностей та початку їх обслуговування СМО

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.

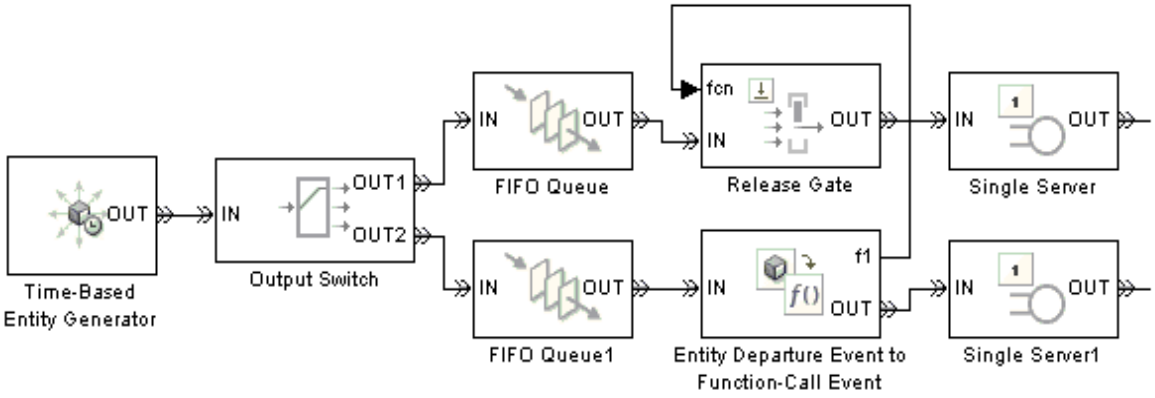


Рис.

Якщо верхня черга в моделі порожня, коли нижня чергу 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 можете 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 врахуванням можливості його вибору випадковим чином між двома вихідними портами сутностей, то цілком імовірно, що деякі сутності дійдуть до серверів, як очікувалося.

Сервери (Servers)

Бібліотека містить три блоки, які симулюють сервіси СМО:

- Single Server – одиничний сервіс;
- N-Server – N-сервісів, де чсло N визначається користувачем;
- Infinite Server – нескінченна кількість сервісі.

Блоки мають подібну структуру, тому детально розглянемо блок Single Server, а для інших блоків зазначимо лише особливості їх застосування та налаштування.

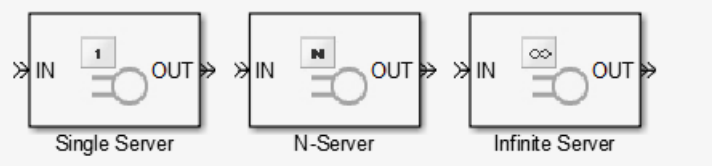


Рис. Блоки серверів

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 зявляється при виборі значення поля Attribute.

Час обслуговування (Service Time) є цілим числом і задається у секундах.

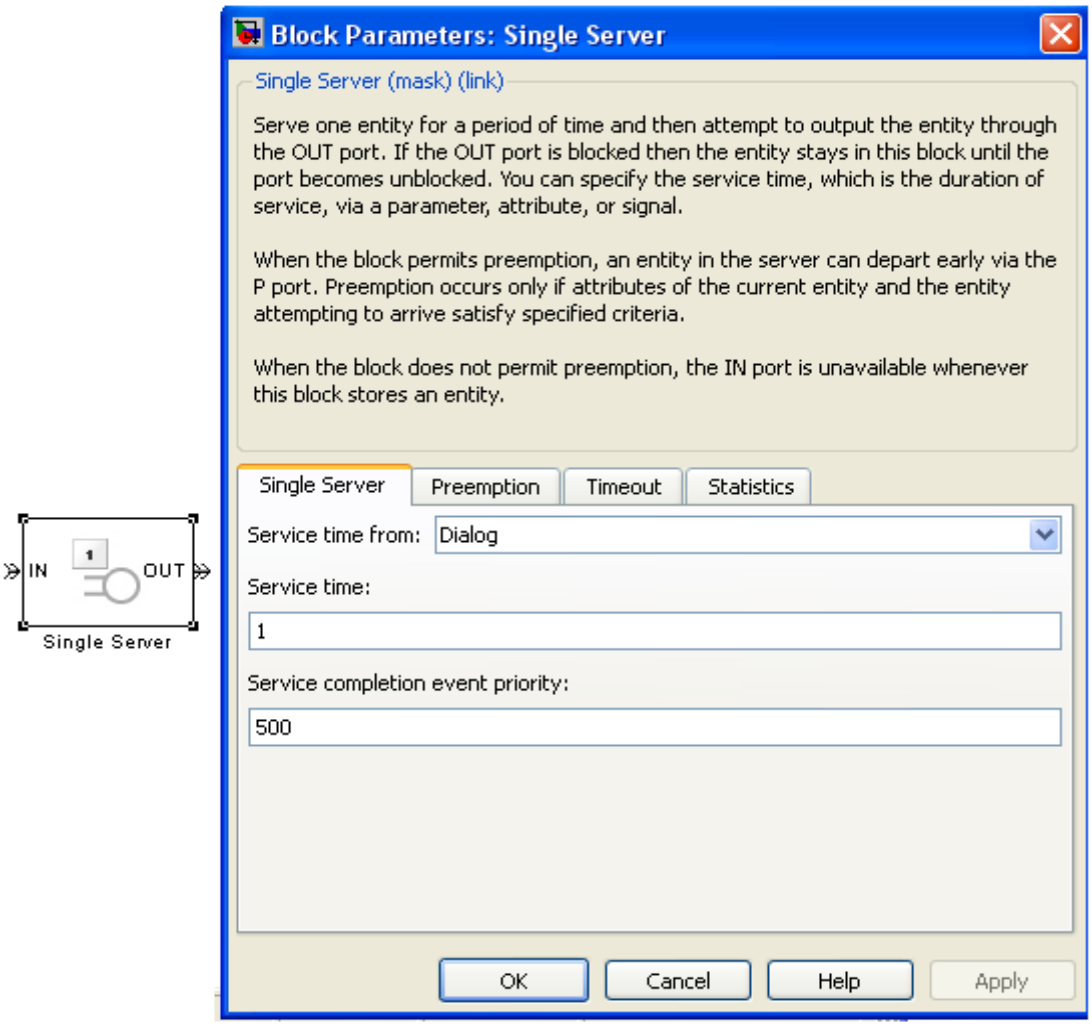


Рис. Блок Single Server

Поле Service completion event priority (завершення обслуговування приорітетних подій) визначає приорітетну 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 приорітетом замість сутності із нижчим і на вкладці статитистики параметр Average wait (очікукування надходження) встановлюється у положення Off (стає недоступним). При виборі опції з'являються наступні поля:

- Sorting attribute name – задає імя приорітетного атрибуту;

- Sorting direction – визначає порядок сортування (за зростанням або спаданням значень приорітету).

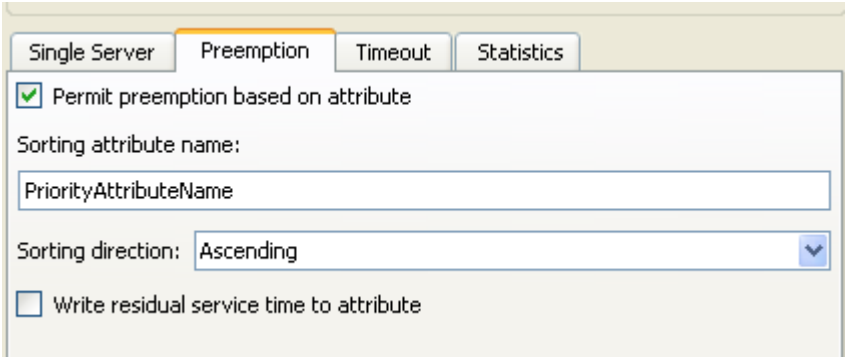


Рис. Закладка Preemption блоку Single Server

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.) 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 переваним 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** (Озвіл переривань на основі параметра атрибута) на вкладці **Preemption** (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**.

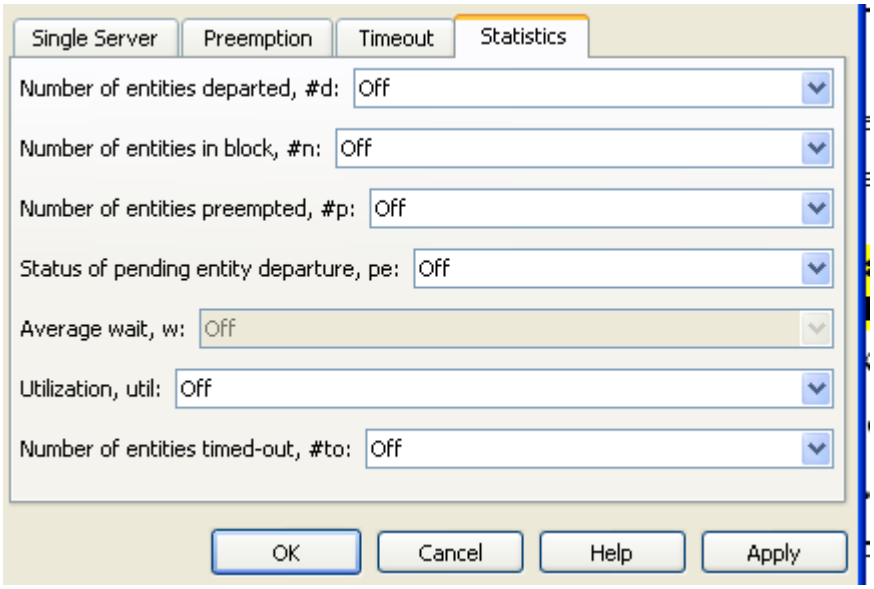


Рис. Вигляд закладки Statistics блоку Single Server

Що б зробити доступним порт для отримання відповідної статистики, його значення необхідно встановити у положення On.

Приклад.

1. Послідовне підключення одиничних сервісів з чергою.

Дві пари черга-сервіс представляють собою послідовні операції. Наприклад, частини на конвеєрі обробляються послідовно двома машинами.

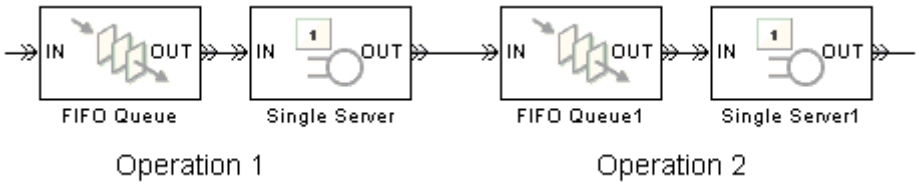


Рис. Послідовне підключення пар черга-сервіс

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.

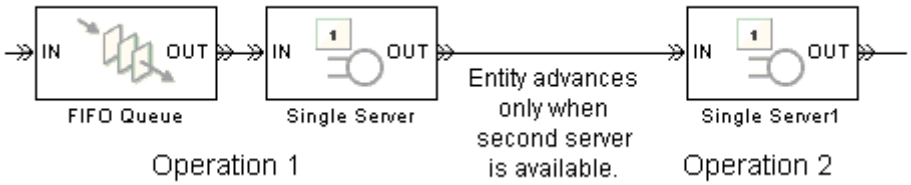


Рис. Послідовне підключенняодиничних сервісів без черги між ними

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.

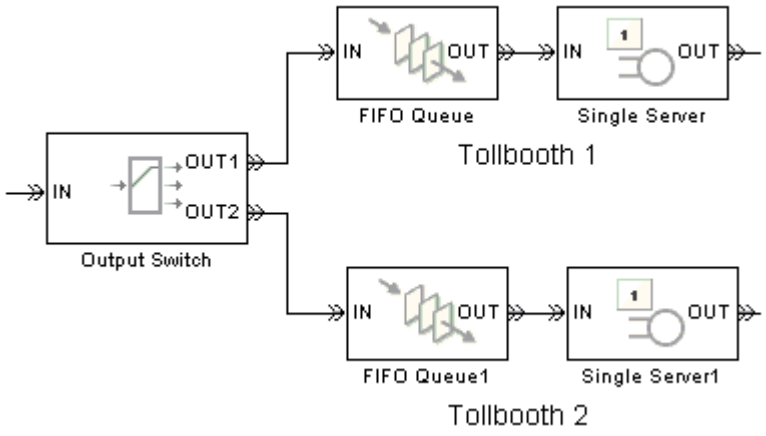


Рис. Паралельні пари черга-сервер як альтернативи.

3. Паралельні пари черга-сервер у багатоадресному повідомленні. Дві парі 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.

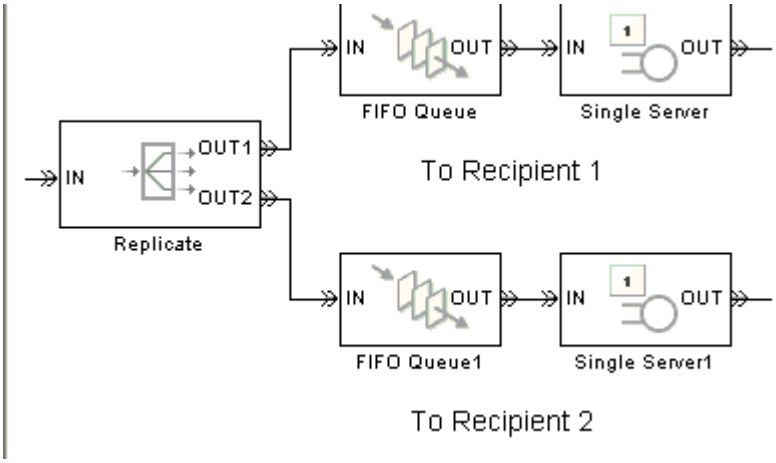


Рис. Паралельні пари черга-сервер у багатоадресному повідомленні

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.).

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.

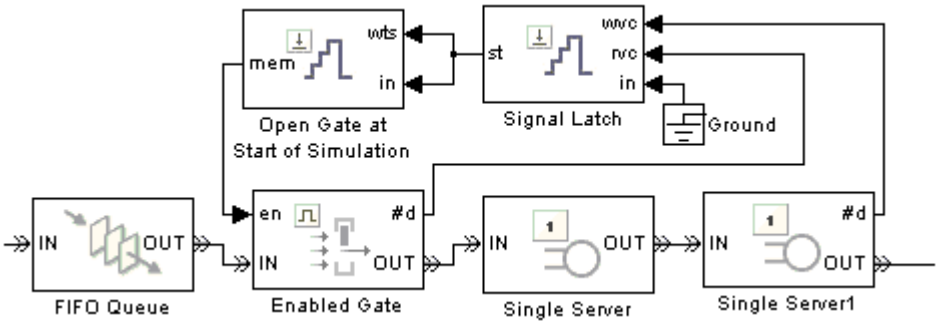


Рис. Контроль двох серверів

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.

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 блокую.

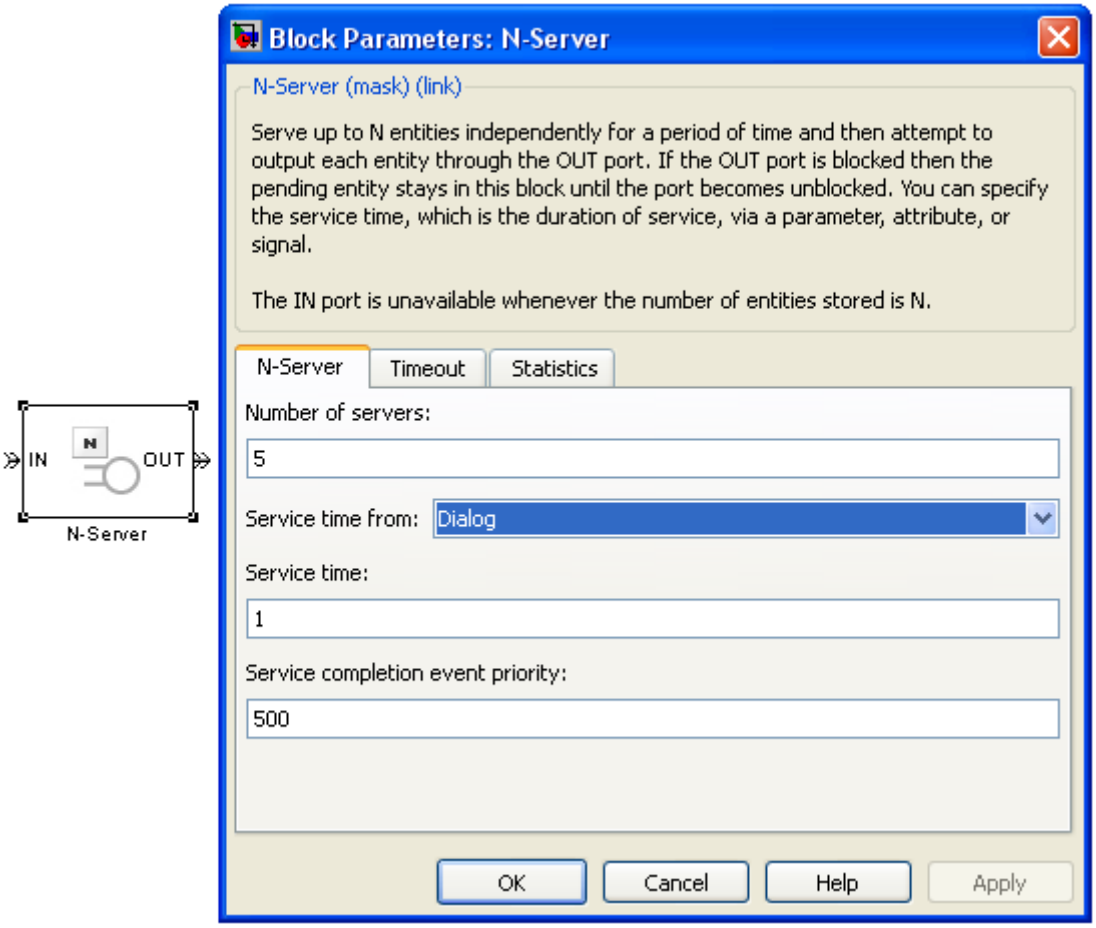


Рис. Блок N-сервер та його діалогове вікно

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.) 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**.

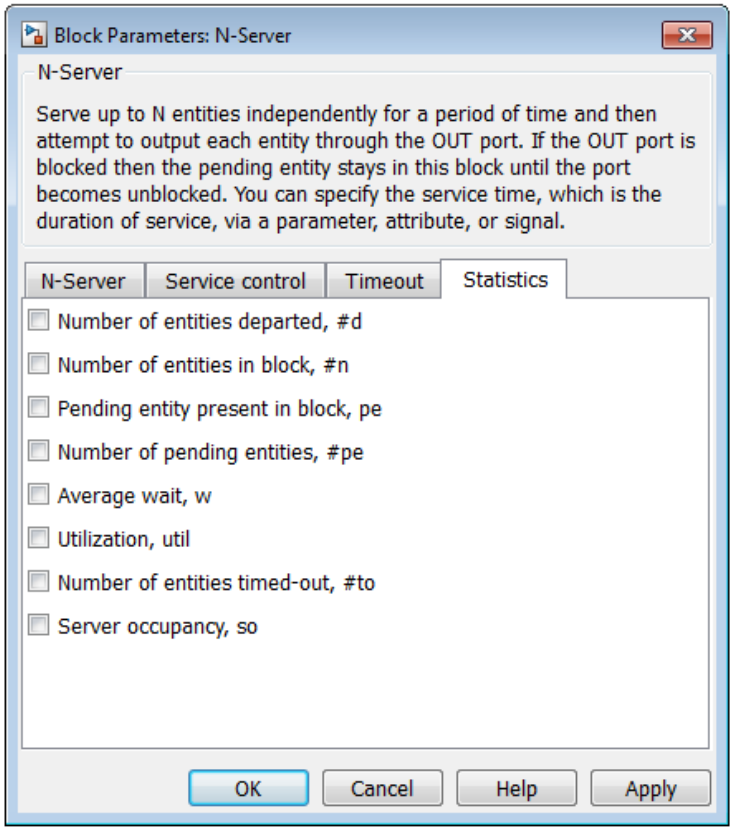


Рис. Вкладка статистики блоку N-сервер

- **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** (Озвіл 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.

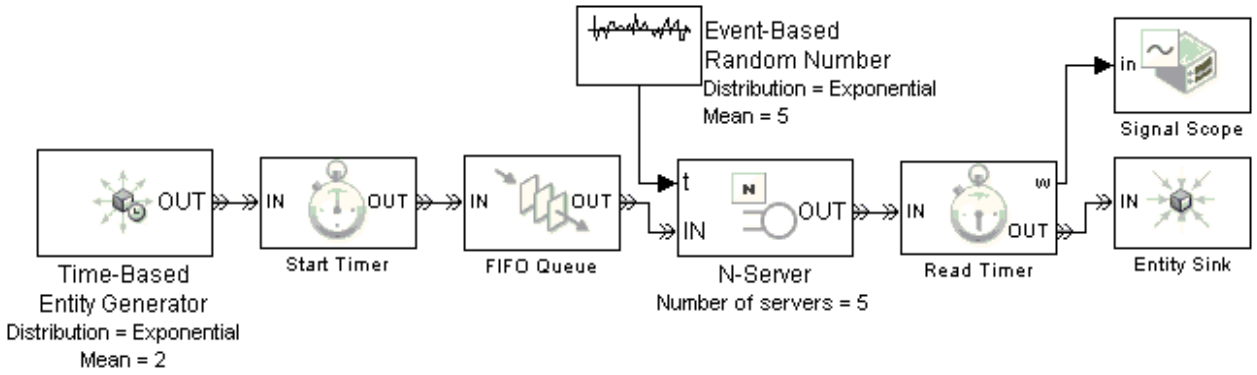


Рис. Система M/M/5 без відмов

Наведений нижче графік показує час очікування в системі (рис.)

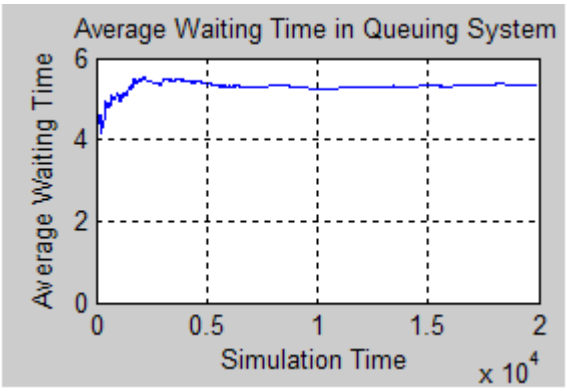


Рис. Графік розподілу часу очікування в системі

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:

$$\rho = \frac{\lambda}{m \cdot \mu} = \frac{(1/2)}{5 \cdot (1/5)} = \frac{1}{2}, \; p_0 = \left(1 + \sum_{i=1}^{m-1} \frac{(m \cdot \rho)^i}{i!} + \frac{(m \cdot \rho)^m}{m!} \cdot \frac{1`}{1-\rho}\right)^{-1} \approx 0{,}080,$$

$$W_{cucm} = \frac{1}{\mu} + \frac{1}{\mu} \cdot \frac{(m \cdot \rho)^m}{m!} \cdot \frac{p_0}{m(1-\rho)^2} \approx 5{,}26.$$

Середній час перебування в системі становить приблизно 5,26 одиниць часу, що відповідає лінії на графіку (рис.).

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 блоку заплановано на роботу за розкладом на тайм-аут, то замовлення може вийти через додатковий порт TO.

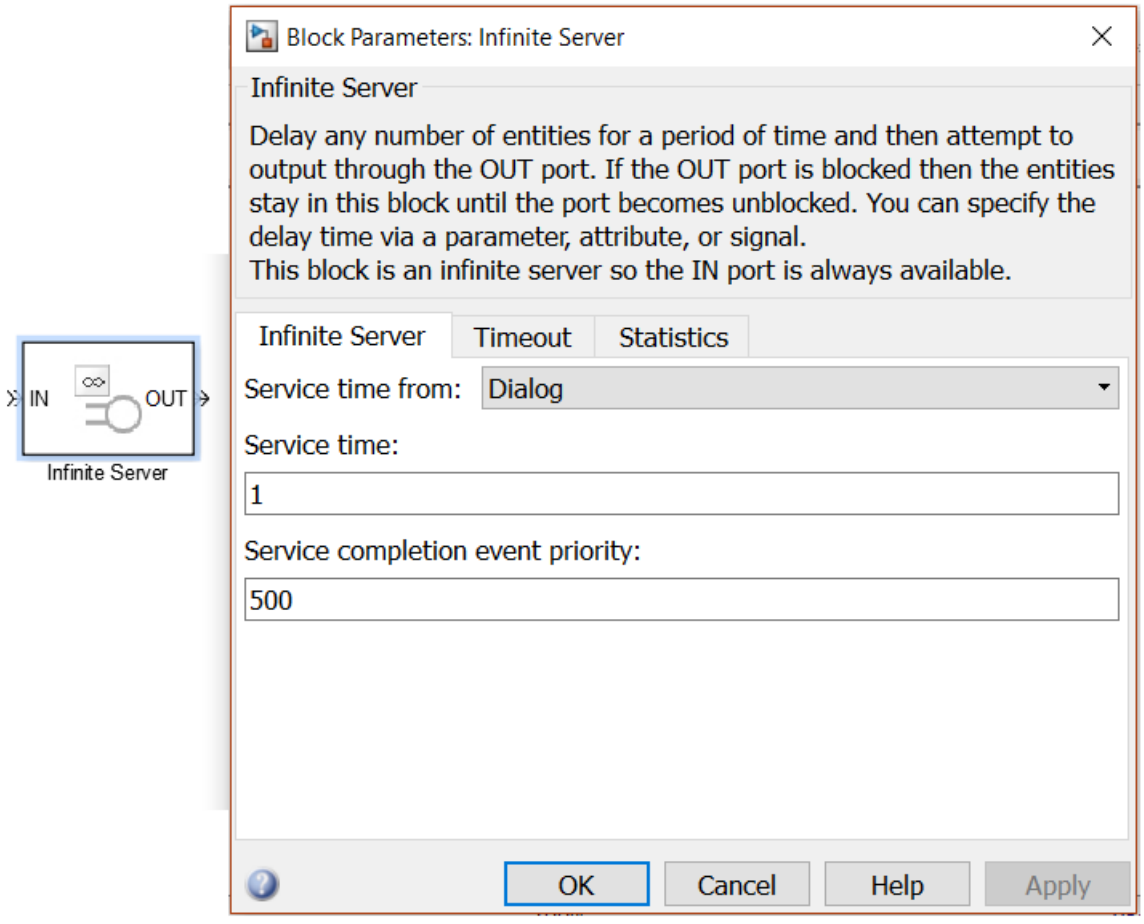


Рис. Блок Infinite Server та його діалогове вікно

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.).

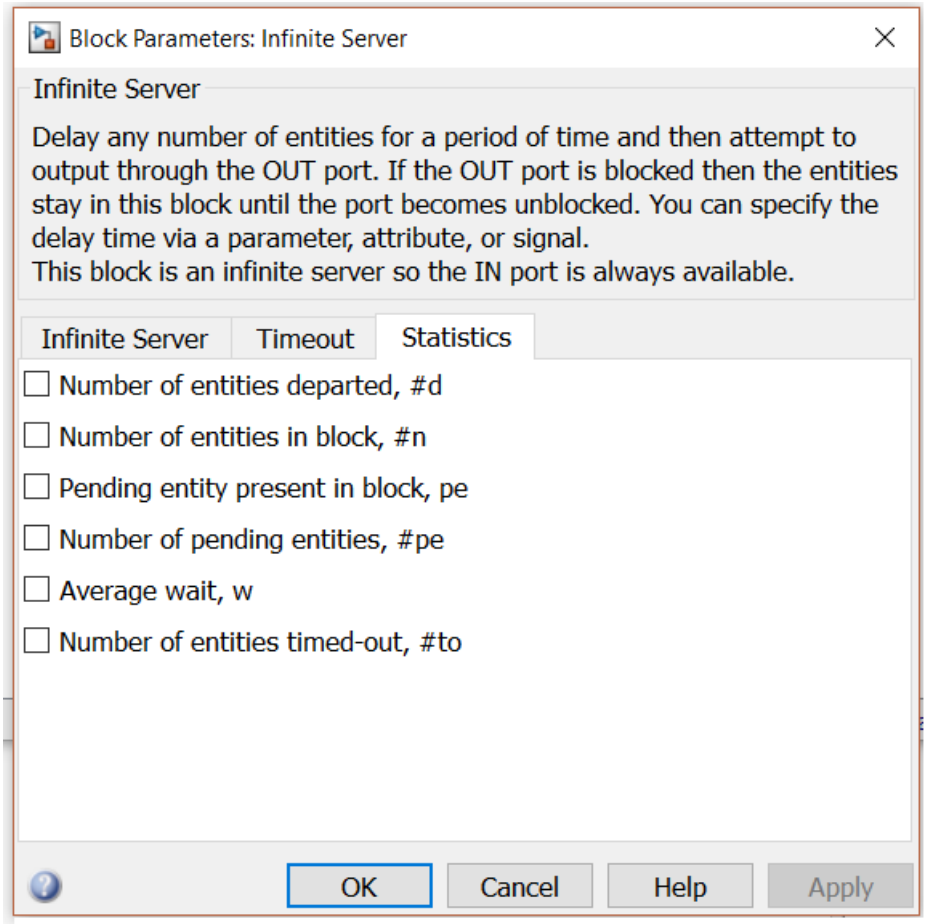


Рис. Вкладка статистики блоку Infinite Server

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.

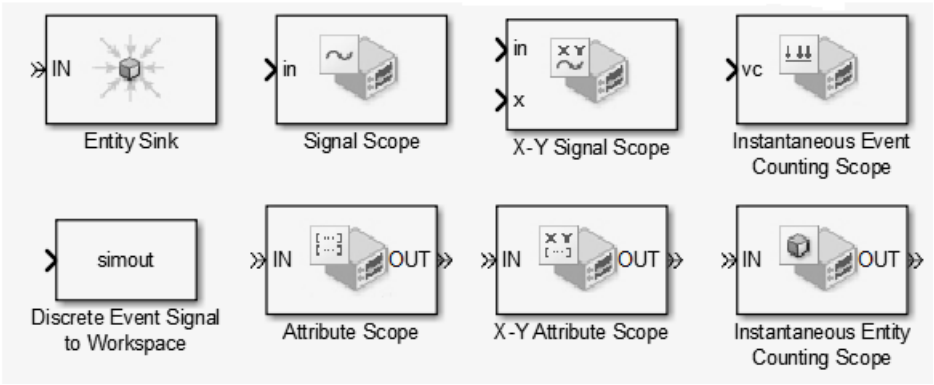


Рис. Блоки бібліотеки SimEvents Sinks

За допомогою блоків Scope можна виводити графіки необхідних характеристик.

Attribute Scope

Attribute Scope – блок виводу графіка даних вхідних атрибутів (параметрів) сутностей.

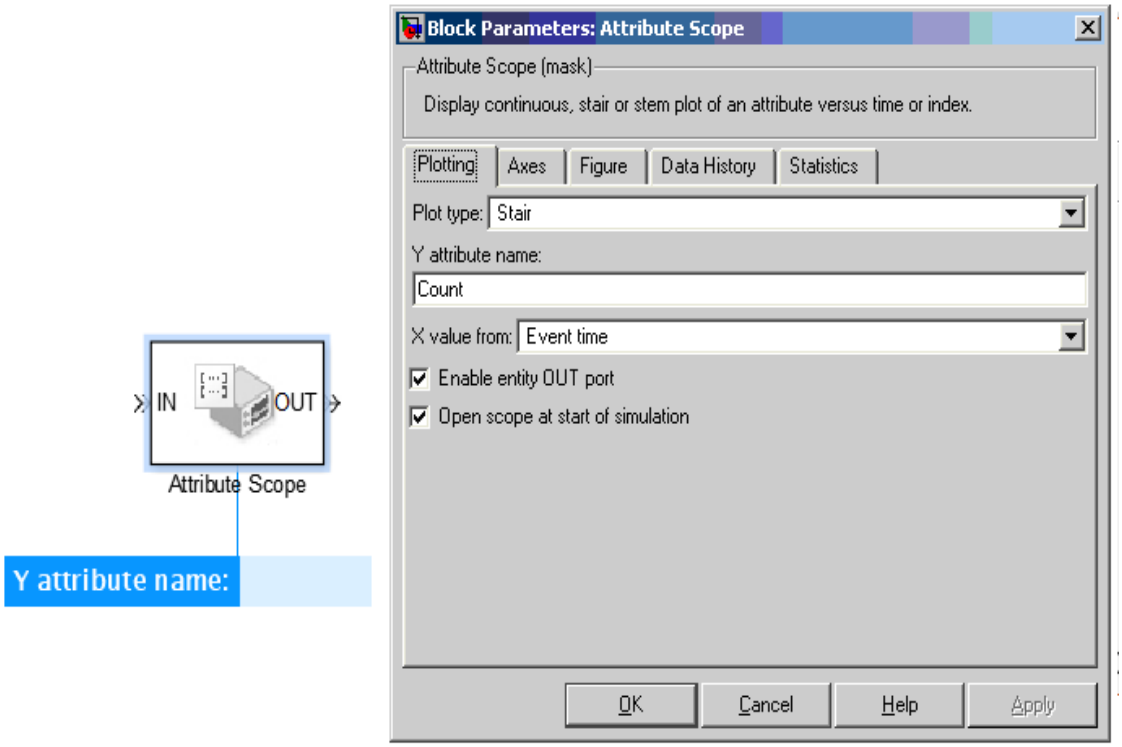


Рис. Блок Attribute Scope та його діалогове вікно

Цей блок формує графік сутностей, що вийшли із системи на основі атрибутів вхідних значень, що задаються дійсними скалярними значеннями. Що б 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 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. Графіки виглядають схожими, крім того що другий графік має рівномірні горизонтальні відступи, а перший має відступи, побудовані на основі промжків часу між сусідніми точками.

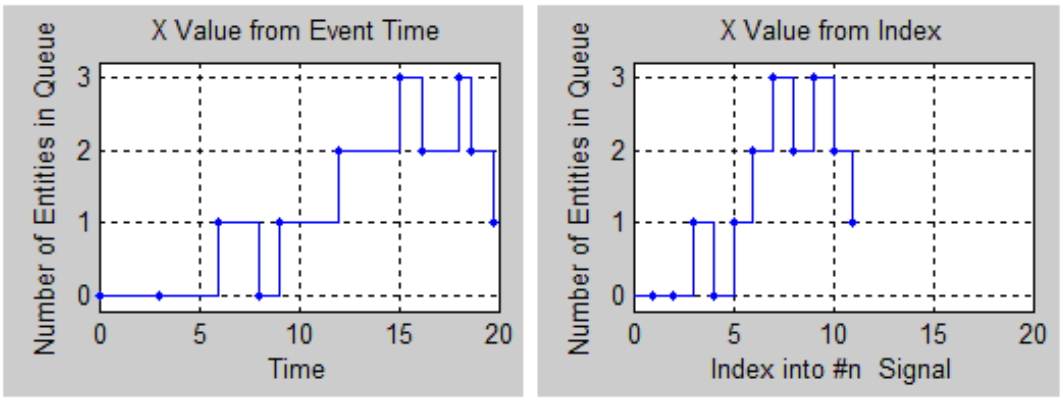


Рис. Приклад побудови графіків за різними джерелами вхідних даних

- **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..

Таблиця 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.):

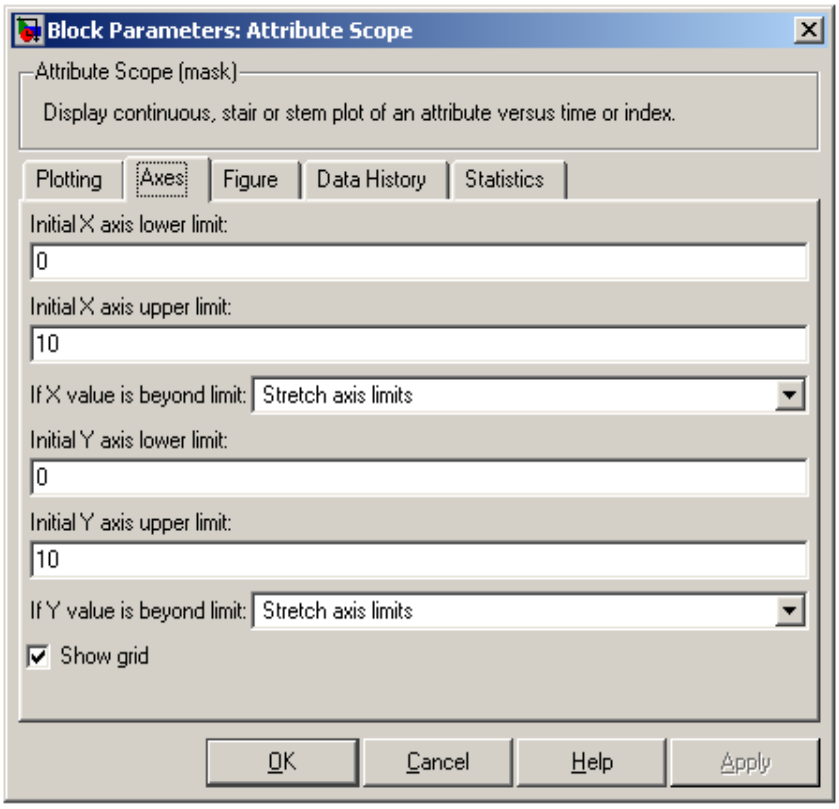


Рис. Закладка налаштування координатних вісей

- **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.)застосовується для створення підписів на графіку:

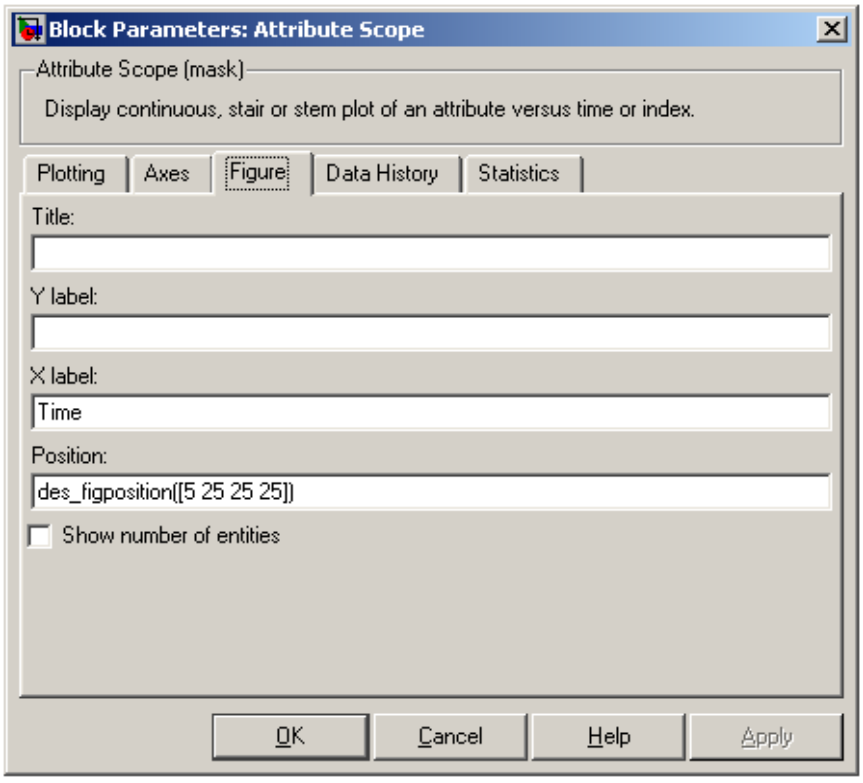


Рис. Закладка налаштування графіка

- **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.):

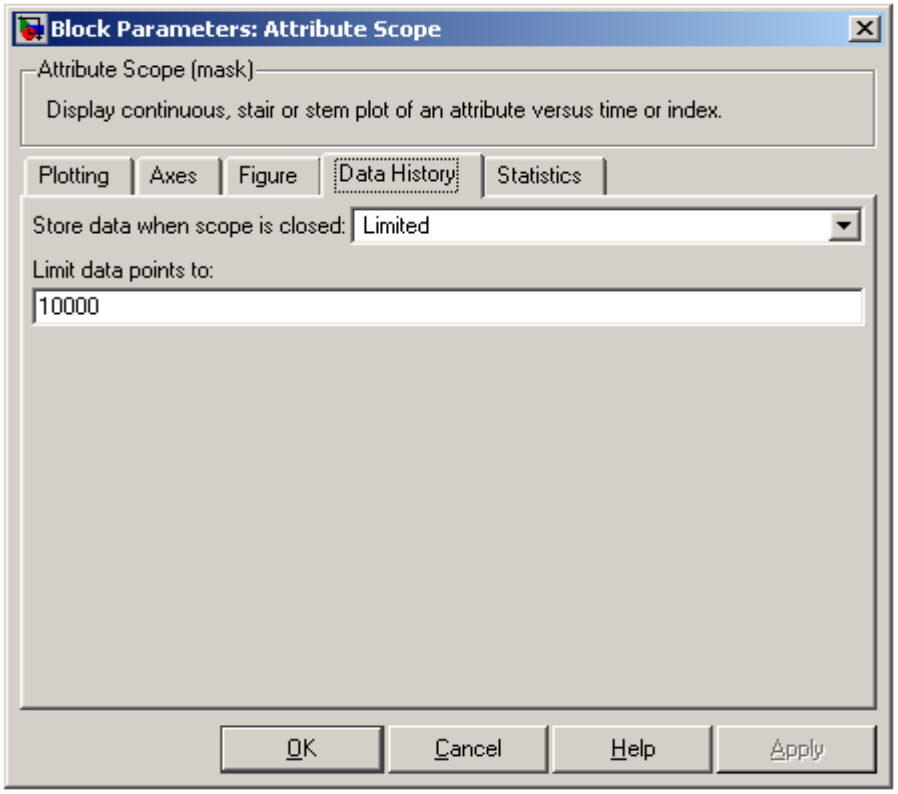


Рис. Закладка налаштування даних

- **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.

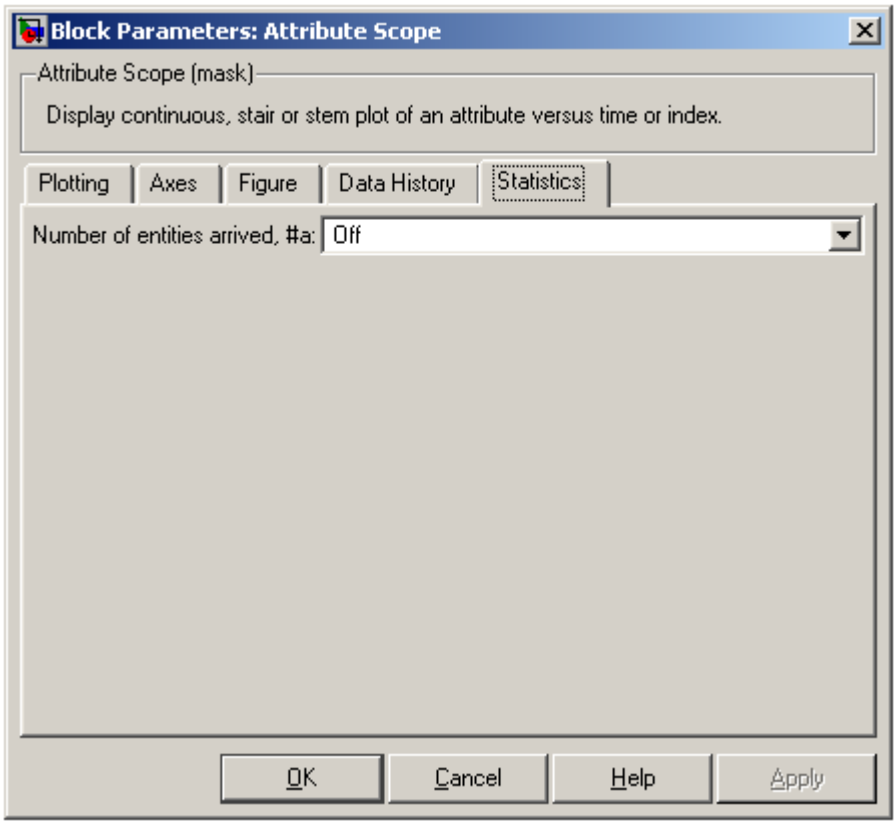


Рис. Закладка налаштуваннястатистики

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.

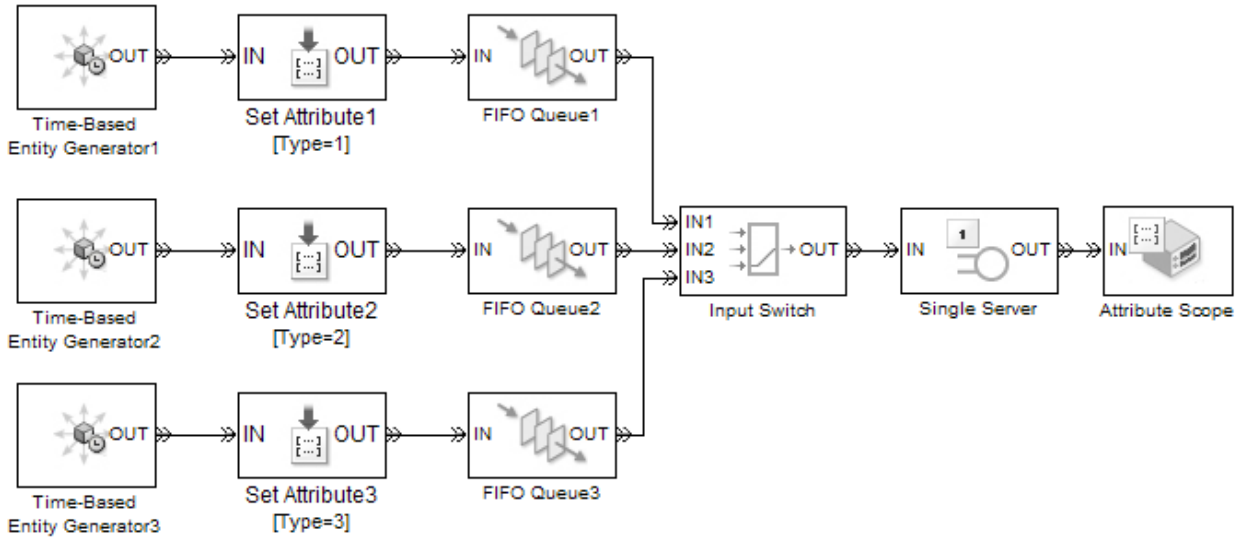


Рис. Модель системи з круговим перемикачем вхідного потоку на сервер

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 данному випадку, за часом (рис.). Кожному типу сутностей відповідає значення певної висоти.

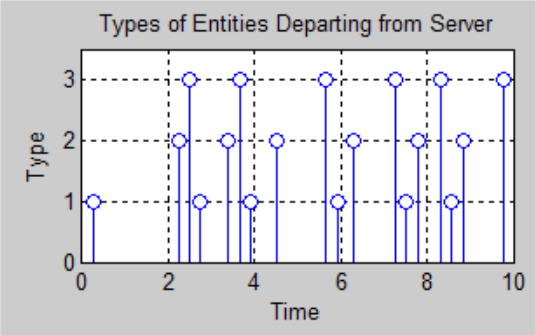


Рис. Графік Attribute Scope за часом

X-Y Attribute Scope

X-Y Attribute Scope **(**X-Y атрибути вибірки) – блок діаграми, який будує криву, використовуючи данні з двох реальних скалярних атрибутів вхідних сутностей (рис.).

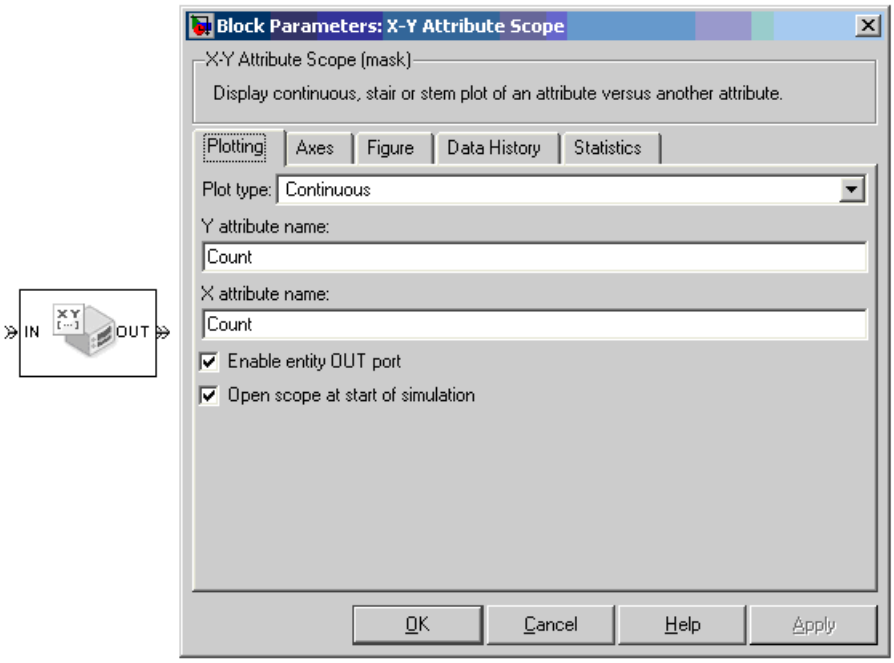


Рис. Блок ***X-Y*** Attribute Scope та його діалогове вікно

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.

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.).

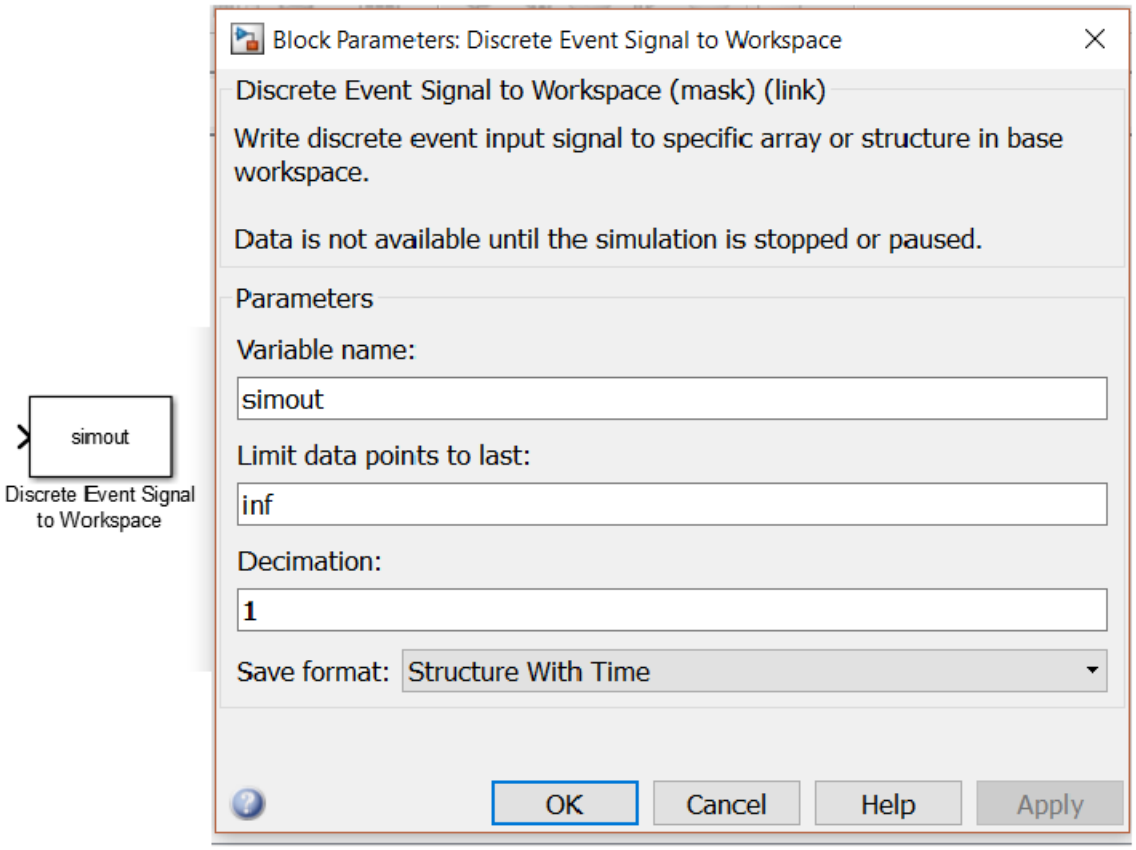


Рис. Блок Discrete Event та його діалогове вікно

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 данними, 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.

Entity Sink

Entity Sink – блок для накоичення або або блокування сутностей. Блоку доступний лише один вхідний порт **IN** – порт для надходження сутностей та вихідний порт **#a** для виведення кількості сутностей.

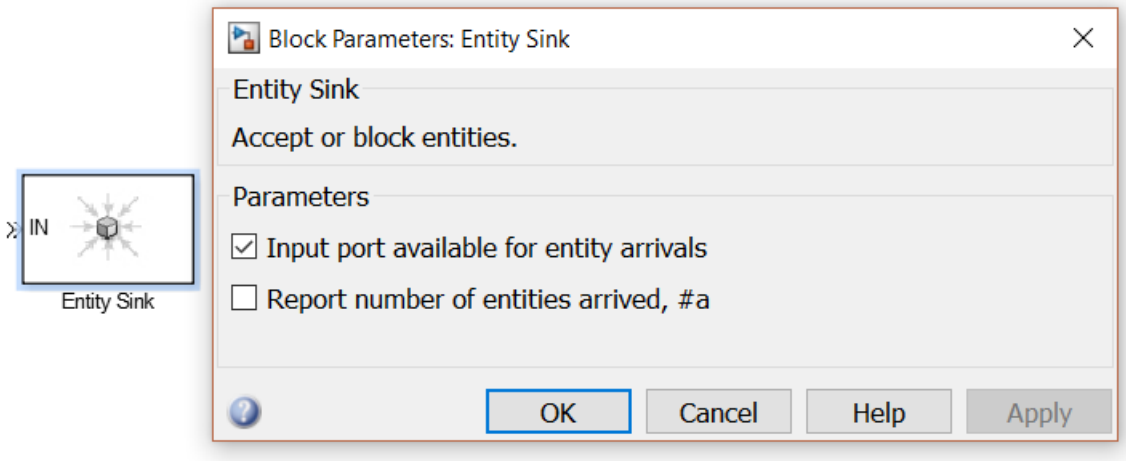


Рис. Блок Entity Sink та його діалогове вікно

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. представлений фрагмент імітаційної моделі, яка складається з трьох одиничних сервісів. Для кохного сервісу передбачений свій накопичувач вихідних сутностей.

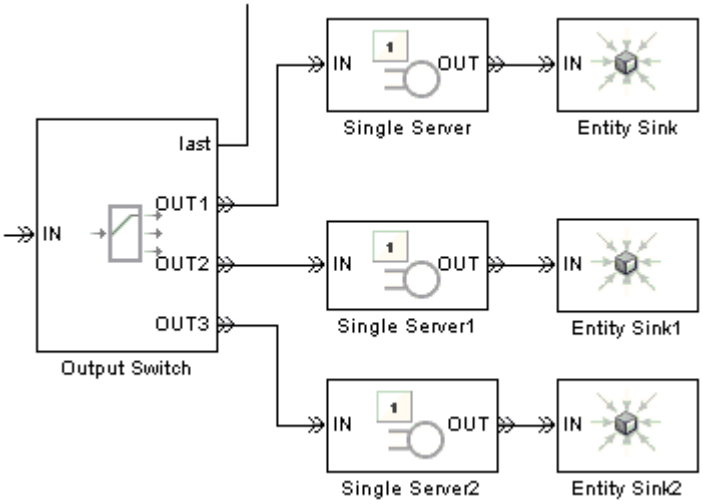


Рис. Фрагмент моделі СМО з використанням блоку Entity Sink

Приклад 2. Використання атрибуту для вибору вихідного порту (рис.) . 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.

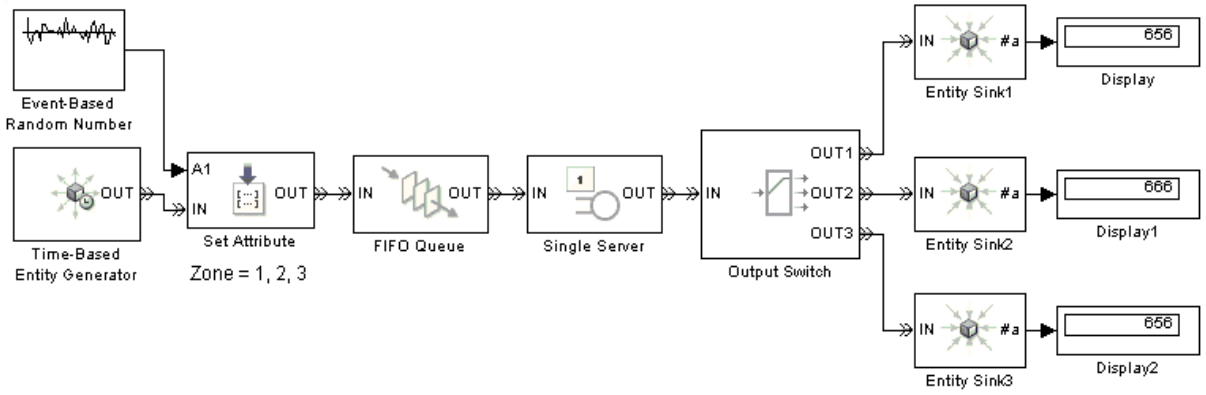


Рис. Модель до прикладу 2

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 перезапускаеться із значенням кількості сутностей рівним 1. Початкове значення вихідного сигналу дорівнює 0. Кількість сутностей накопичується протягом заданого часу, але не підсумовується протягом усього моделювання.

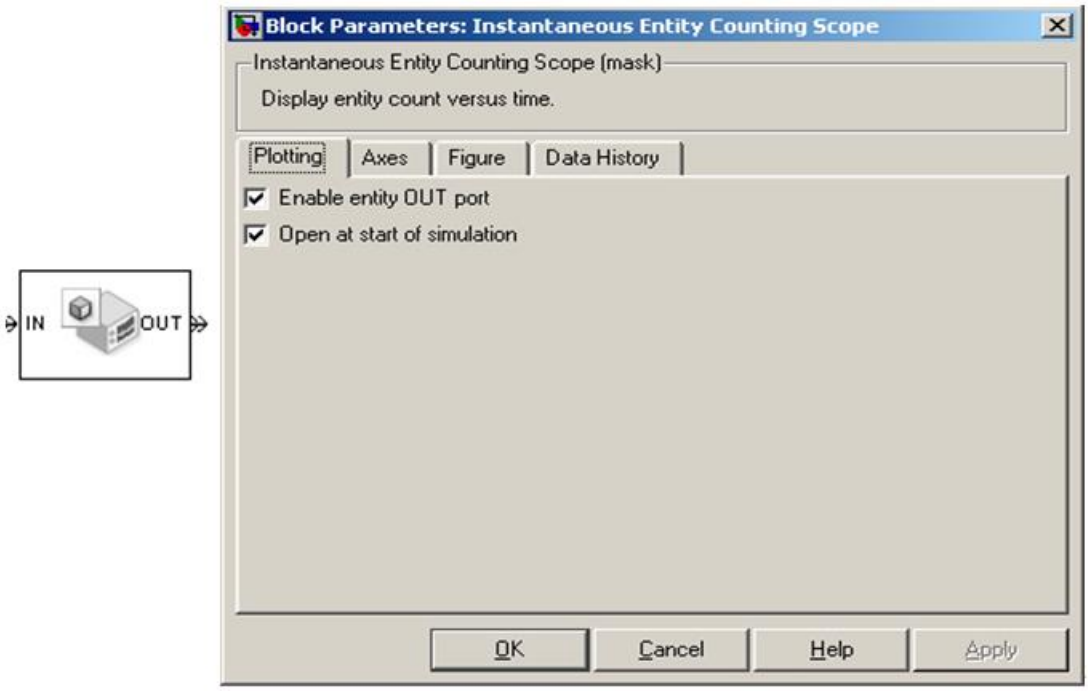


Рис. Блок Instantaneous Entity Counting Scope та його діалогове вікно

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 осі**)** представлена на рис. 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 осі 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 осі 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.) дозволяє встановлювати підписи графіка та його вісей і містить наступні поля:

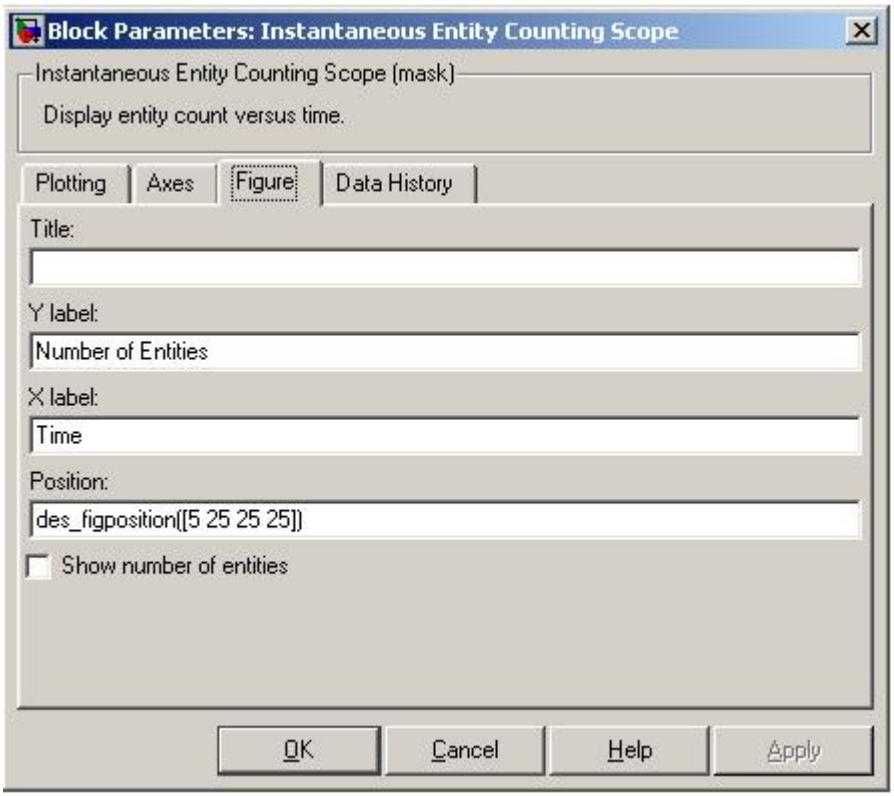


Рис. Закладка Figure блоку Instantaneous Entity Counting Scope

- **Title** (заголовок) – текст, який з'являється в якості назви діаграми, над осями;

- **Y label** (вісь Y) – текст, який з'являється зліва від вертикальної осі;

- **X label** (вісь X) – текст, який з'являється нижче горизонтальної осі;

- **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.) 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 подем **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.

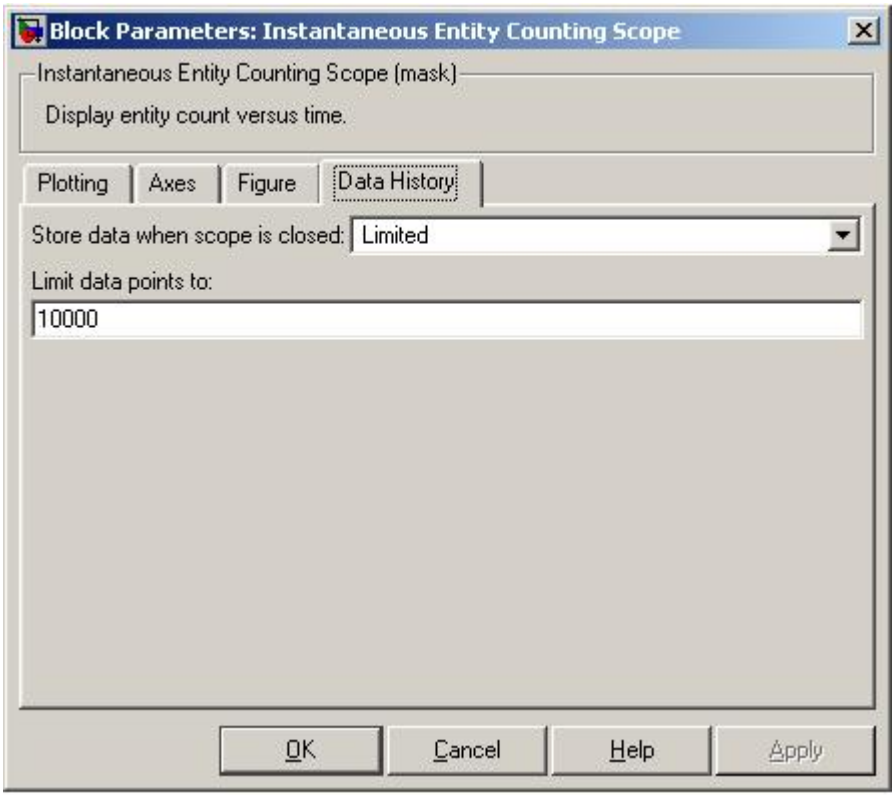


Рис. Закладка Data History блоку Instantaneous Entity Counting Scope

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.

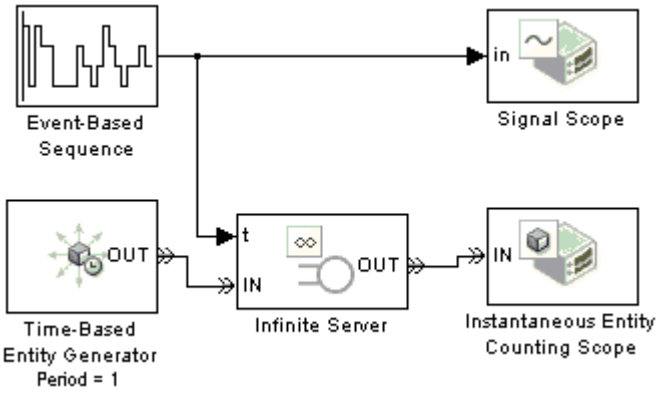
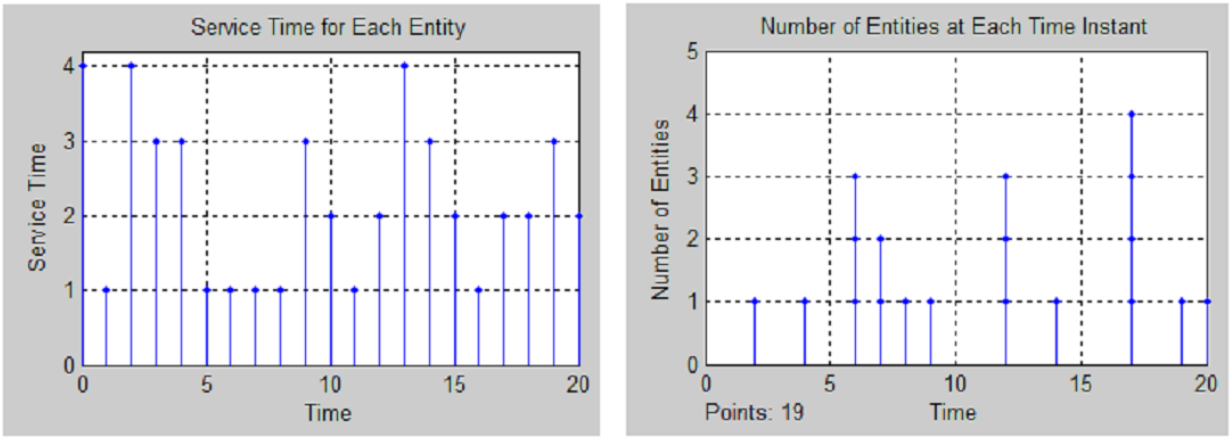


Рис. Приклад використання блоку Instantaneous Entity Counting Scope

Результати моделювання наведені на рис. , 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. Результати роботи моделі з використанням блоків а)Signal Scope

б) Instantaneous Entity Counting Scope

Instantaneous Event Counting Scope

Instantaneous Event Counting Scope (Події графіку розраховуются в залежності від часу) – блок відображає графік подій (рис.).

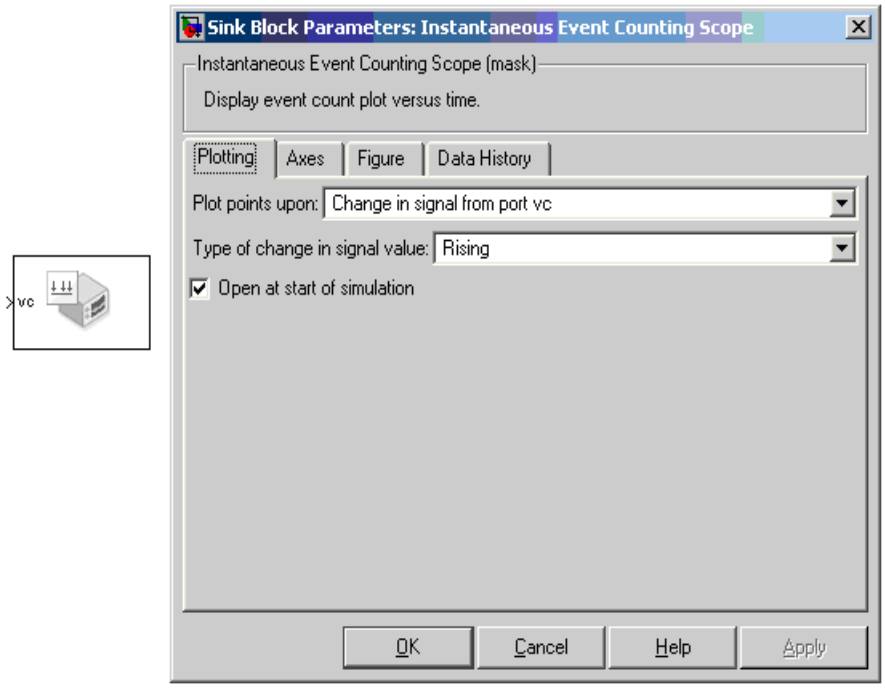


Рис.

Блок оновлює розрахунок, починаючи з 1, коли відбувається зміна у часі. Розрахунок є кумулятивним для кожної миті часу, але не кумулятивний для всієї симуляції.

Вхідні порти, доступні для блоку представлені у табл..

Таблиця 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 критерія. 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 надходження 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..):

- **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 поделювання. 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 одночасных подій, які ви хотіли би бачити дійсно 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 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 спів падання – одночасно.

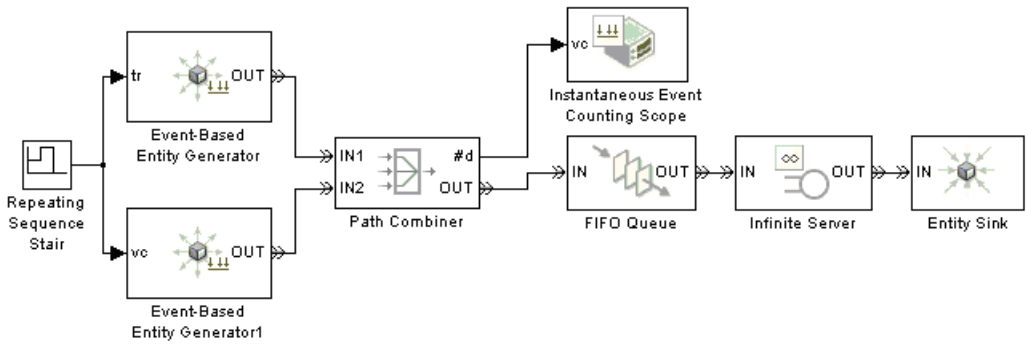


Рис. Модель з одночасними подіями

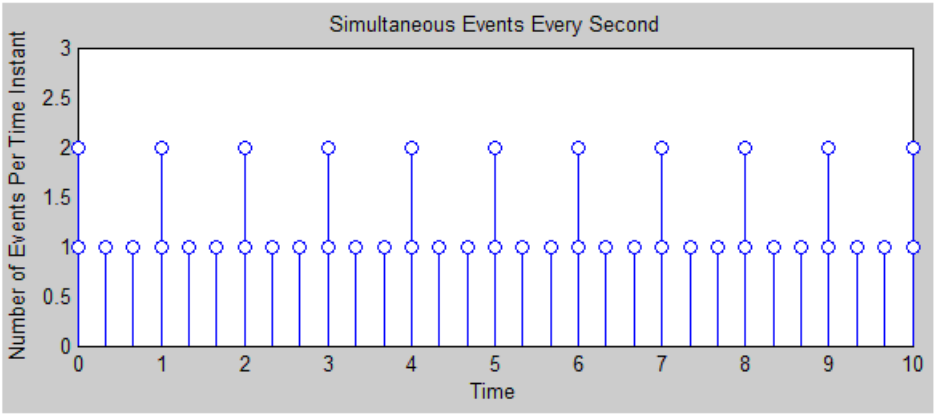


Рис. Графік надходження подій

В даному прикладі замість блоку Instantaneous Event Counting Scope можна було використати блок Instantaneous Entity Counting Scope.

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 осі виходять із сигналу, підключеного до вхідного порту **in** (рис.).

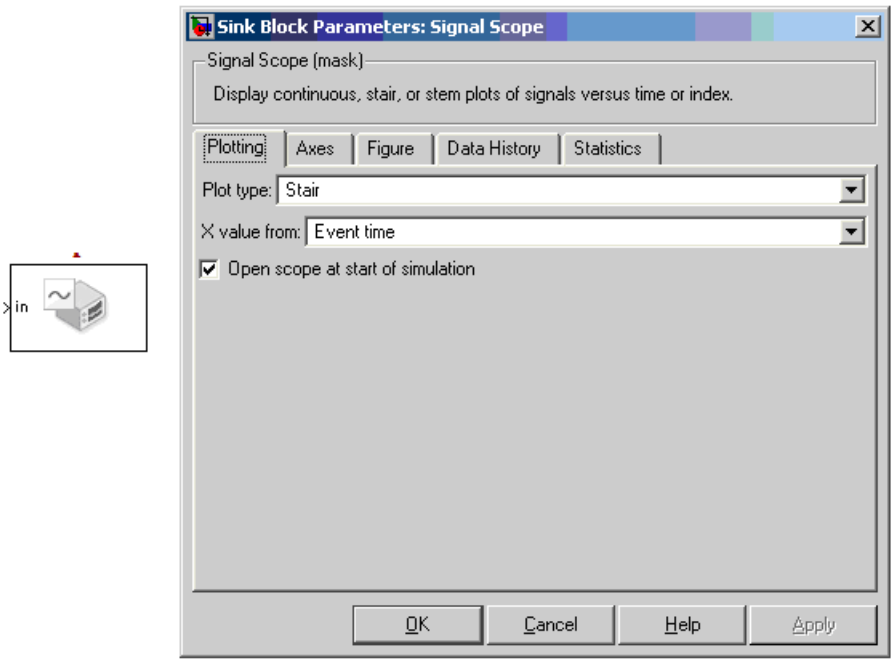


Рис.

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 осі. 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 осі. Ділянки схожі, за винятком того, що друга ділянка має рівномірний горизонтальний відстань, а не на основі часу відстань між сусідніми точками.

Наведені нижче графіки, що ілюструють використання різних джерел для горизонтальної осі X. Графіки схожі, за винятком того, що на другому графіку горизонтальні відстані рівномірні, а не на основі часу.

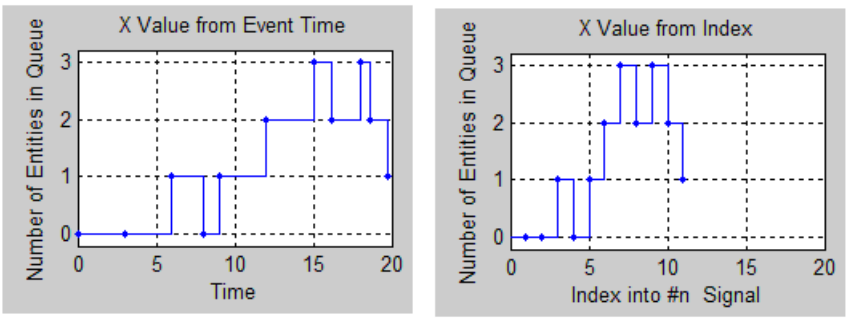


Рис.

Структура та значення полів закладок Axes, Figure? Data History така ж сама, як і для блоку Instantaneous Entity Counting Scope, описного 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.

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.).

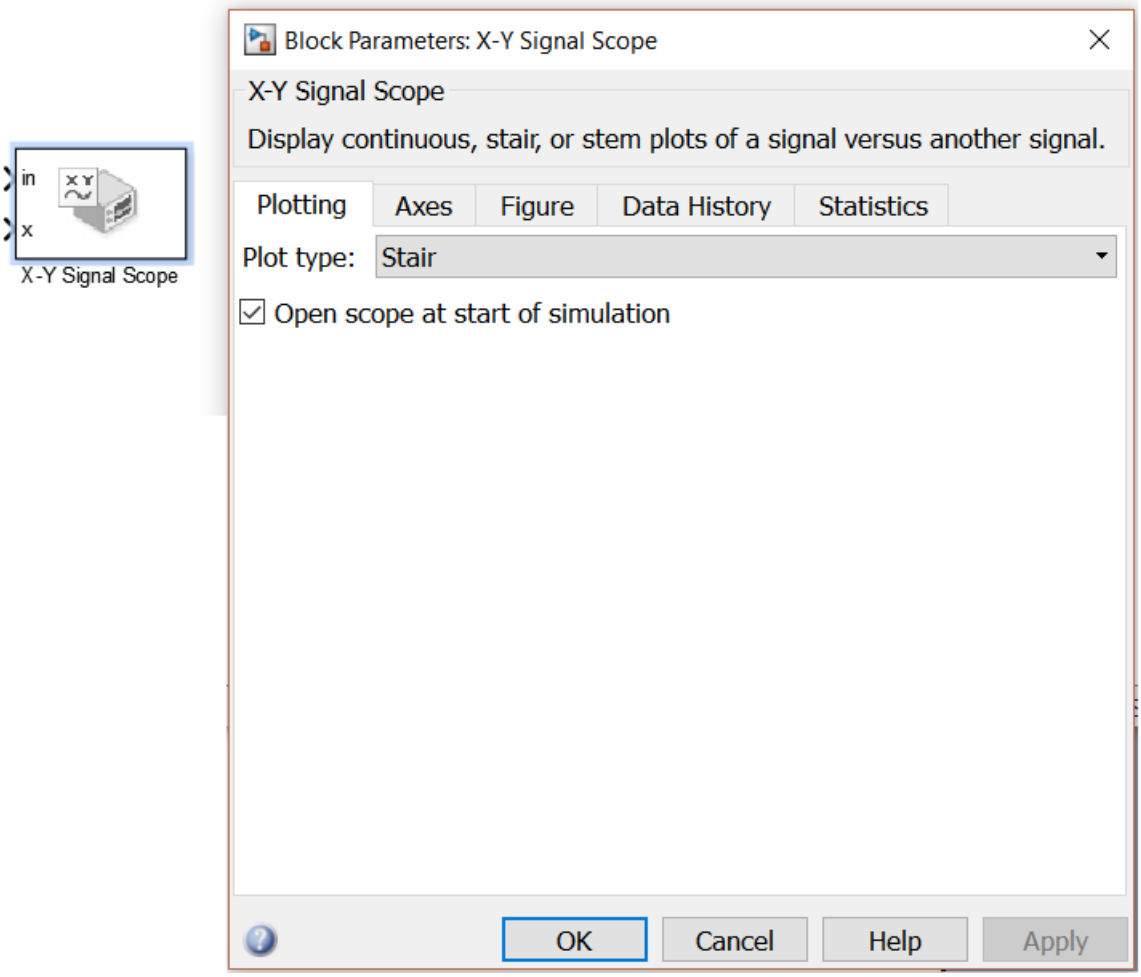


Рис.. Вигляд блоку X-Y Signal Scope та його діалогове вікно

Блок має два вхідних порти: **in** – для надходження даних для вертикальної осі; **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. представлена імітаційна модель з використанням блоку X-Y Signal Scope, де у якості значень за віссю X береться сигнал лінійний сигнал Ramp, а даними для осі Y є оброблені замовлення одиничним сервісом.

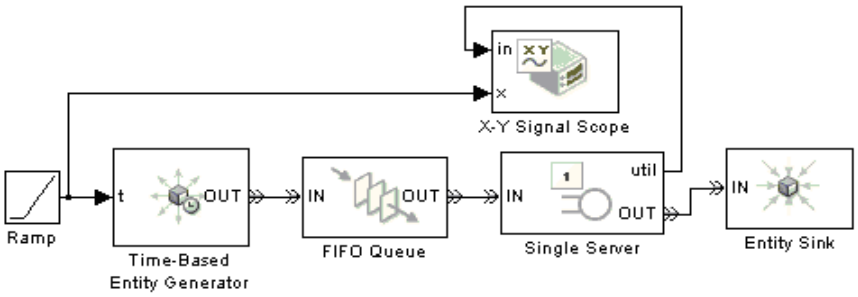


Рис. Модель з використанням блоку X-Y Signal Scope

Результати моделювання представлені графіком, сформованим блоком X-Y Signal Scope представлені на рис.

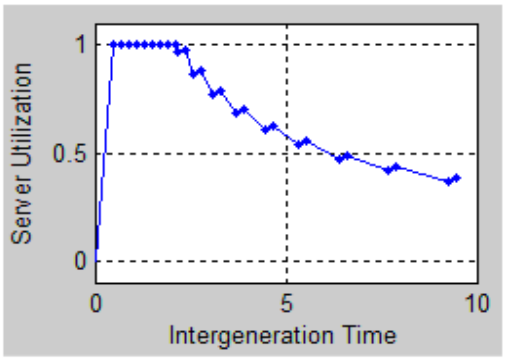


Рис.

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.

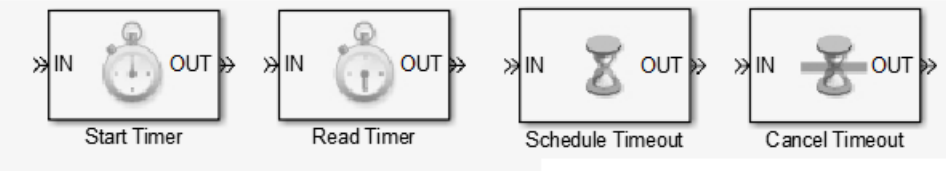


Рис. Блоки бібліотеки Timing

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.

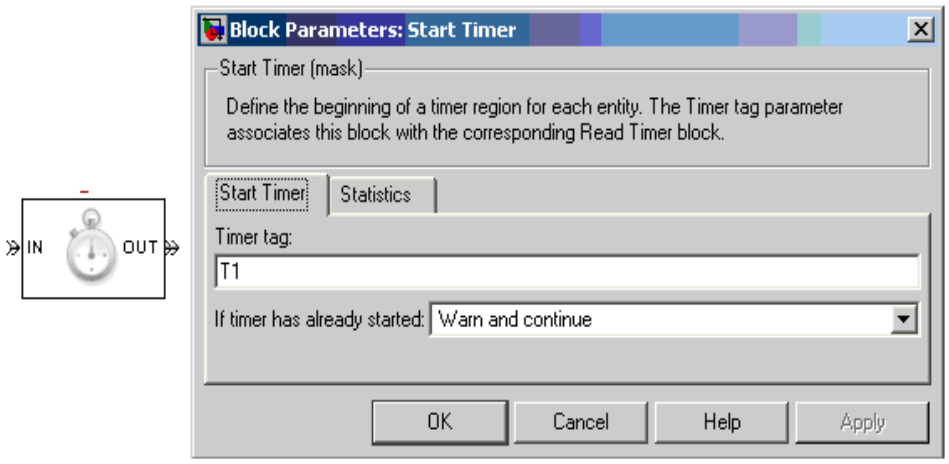


Рис.

Даний блок працює з блоком Read Timer (зчитуючий таймер). 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.).

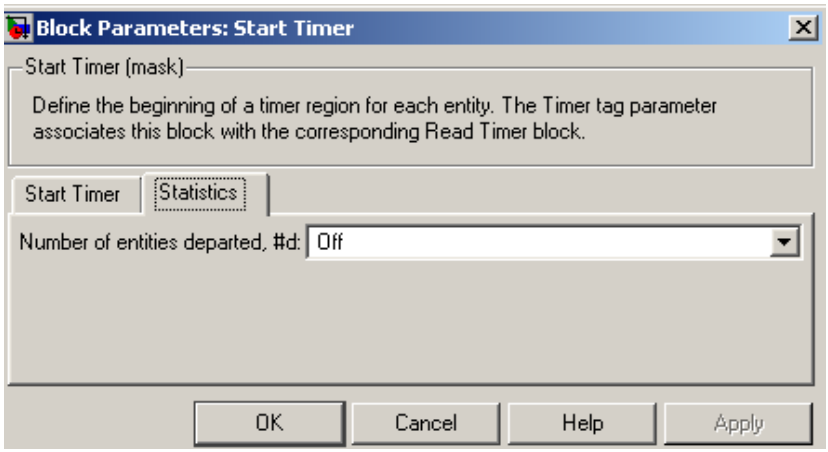


Рис.

Стандартні процедури встановлення блоку.

Для застосування блоку необхідно виконати певний порядок дій:

1. Визначте точки моделі з яких необхідно починати таймінг та задайти 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 налашувати 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 більшої інформації див. Timing Multiple Processes Independently.

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. 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.

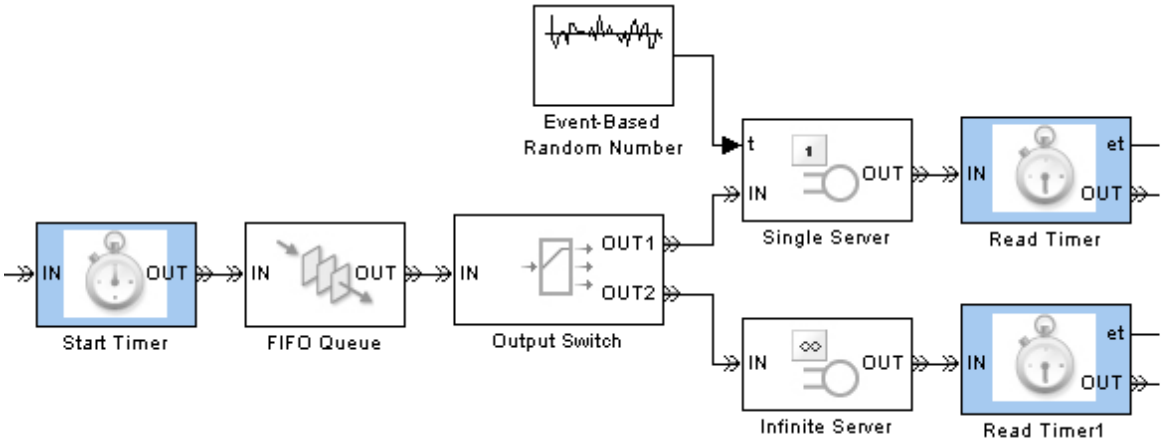


Рис.

2. На рис. нижче 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.

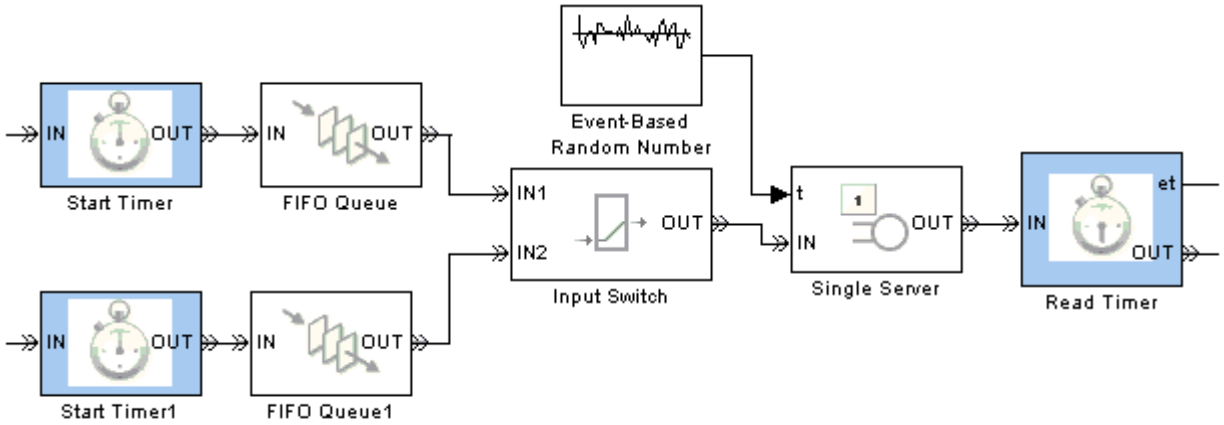


Рис.

Зчитуючий Таймер (Read Timer)

Read Timer – виведення статистичних даних про вказаний (названий) таймер, пов'язаний з надходженням сутностей (рис.). Блок зчитує значення таймера, відповідного блоку Start Timer (Запуску таймера).

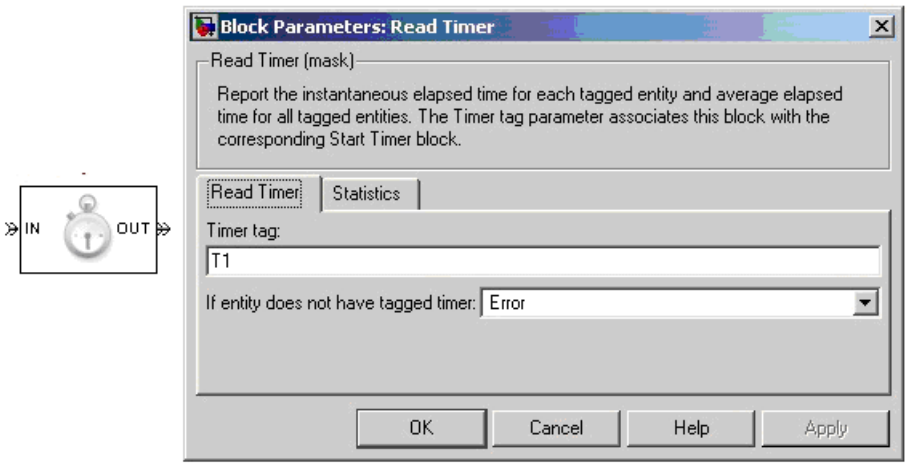


Рис. Блок Read Timer та його діалогове вікно

Використовуючи час, що минув у звіті, та відзвітувати 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..) час оновлення яких після виходу сутності.

Таблиця

Назва	Опис	Порядок поновлення
#d	Кількість сутностей, які пройшли через блок з початку моделювання.	З

#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 вкладці 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.

.

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.).

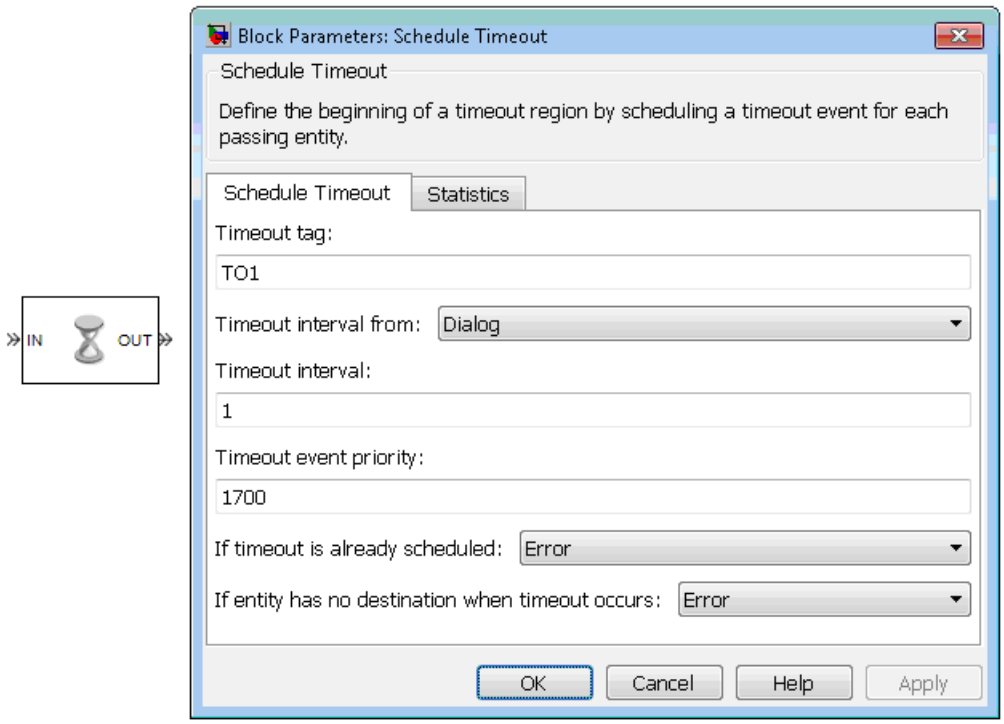


Рис. Блок Schedule Timeout та його діалогове вікно

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 значення, 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 ви 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.) має лише одне поле **Number of entities departed** для визначення кількості подій що вийшли з блоку через сигнальний порт **#d**. Початкове вихідне значення 0.

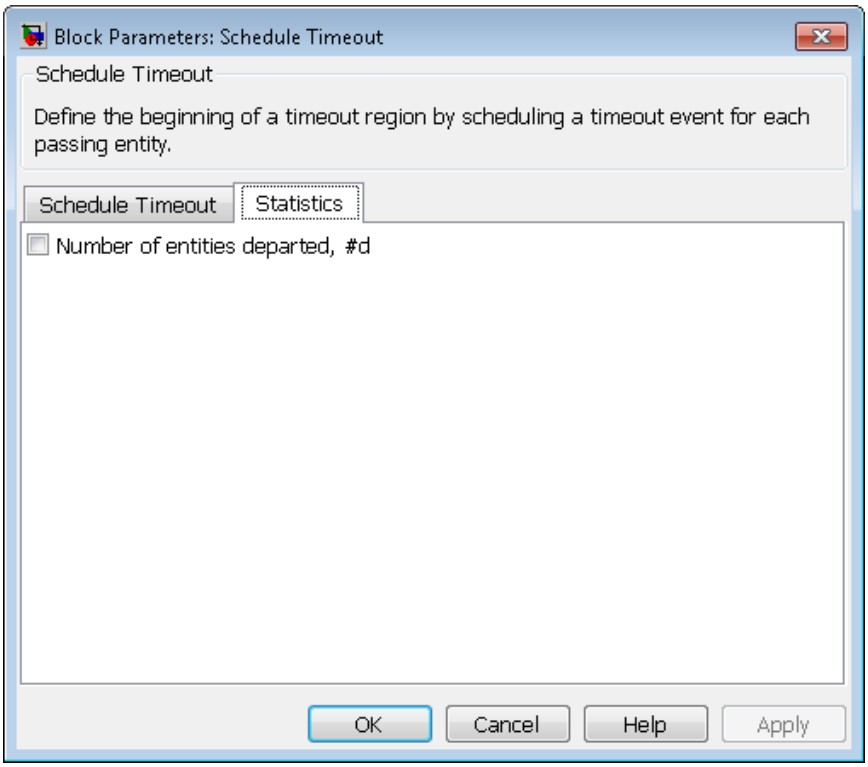


Рис. Закладка статистики блоку Schedule Timeout

Приклад використання блоку Schedule Timeout наведений нижче в описі блоку Cancel Timeout

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.). Цей блок позначає завершення часового обмеження для сутностей. Можливість скасування події Timeout перш, ніж вона відбудеться дозволяє продовжити життя сутності.

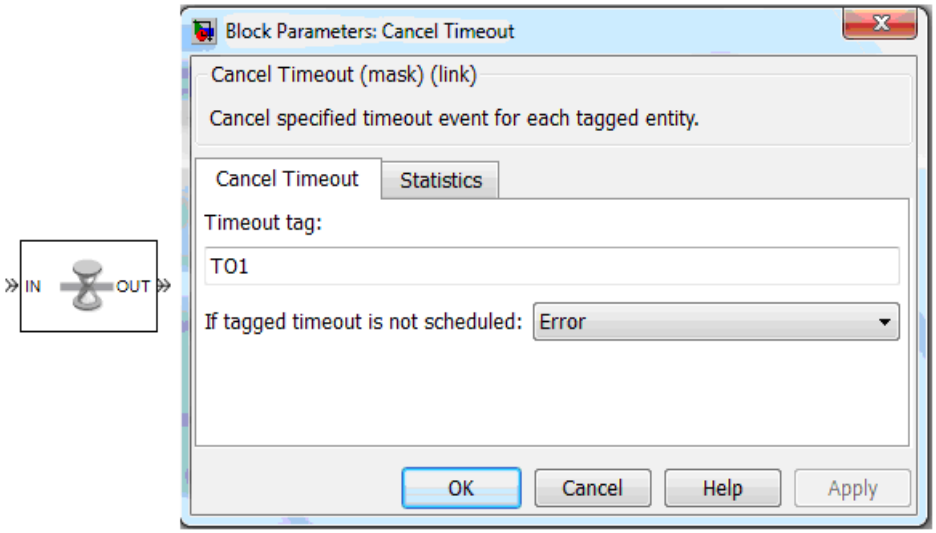


Рис. Блок Cancel Timeout та його діалогове вікно

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.).

Таблиця 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 імям події Timeout	2
rt	Час між час прибуття в цей блок і часом блоку Scheduled time пов’язаного з імям 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** (Обєкт 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** – і продовжувати.

На вкладці **Statistics** (Статистика) розташовані наступні поля:

- **Number of entities departed** – виводить кількість сутностей, що пройшли через блок і контролює поведінкою вихідного сигналу порту **#d**;
- **Number of entities departed with specified tag** – виводить кількість сутностей з режимом тайм-аут, що пройшли через блок і контроль поведінкою вихідного сигналу порту **#t**;
- **Residual time** – визначає проміжок часу між прибуттям сутності до блоку і часу 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 контроль поведінкою вихідного сигналу порту **#w.**

Для всіх сигналів початкове вихідне значення, яке діє з початку моделювання до першого оновлення блока, дорівнює 0.

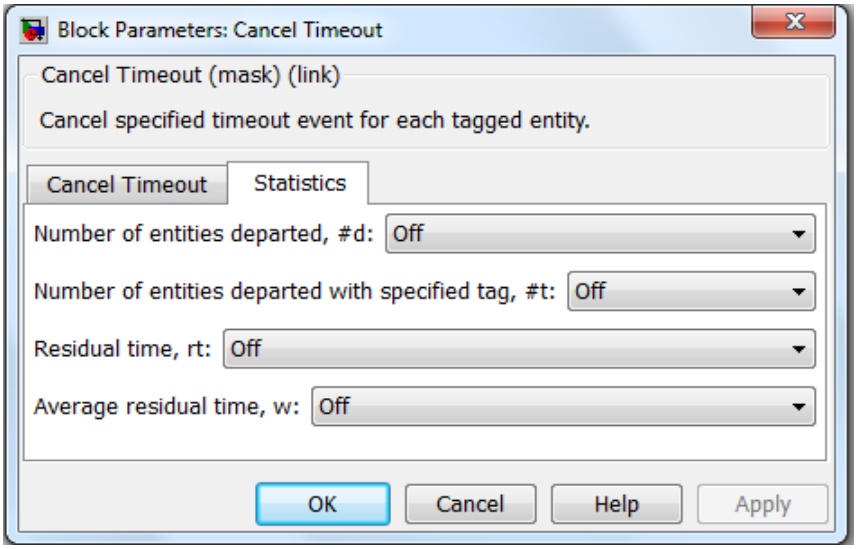


Рис. Вкладка Statistics

Приклад. Використиння тайм-аутів для обмеження часу перебування сутностей у черзі.

Модель зображена на рис. обмежує час, існування кожної сутності в черзі, але не обмежується 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.

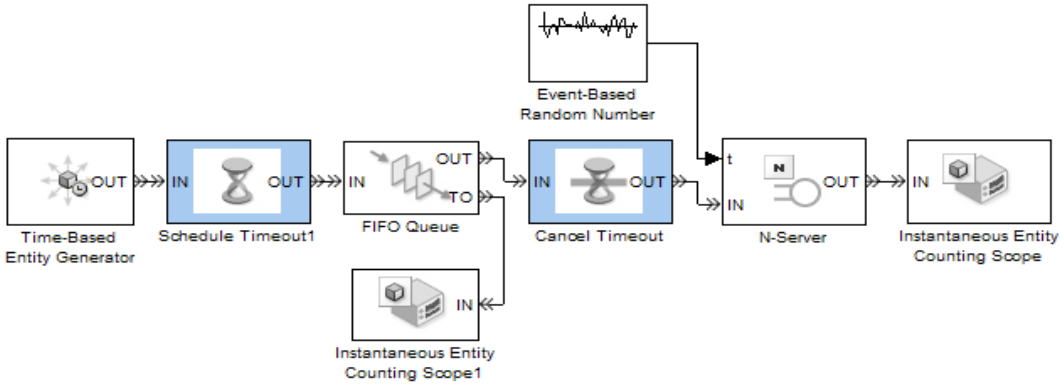


Рис.

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 загальному, якщо сутність надходить до блоку Cancel Timeout, перш ніж годинник досягає ліміту часу, з сутності знімається її часове обмеження. Сутність переходить до сервера.

Блоки перемикачів та управління потоками (Routing)

Блоки бібліотеки Routing дозволяють об’єднувати або розподіляти потоки сутностей (рис.)

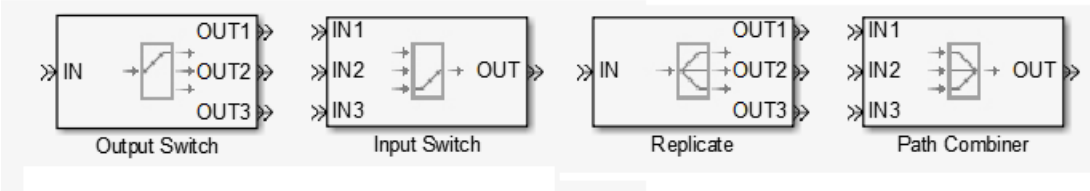


Рис. Блоки бібліотеки Routing

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.

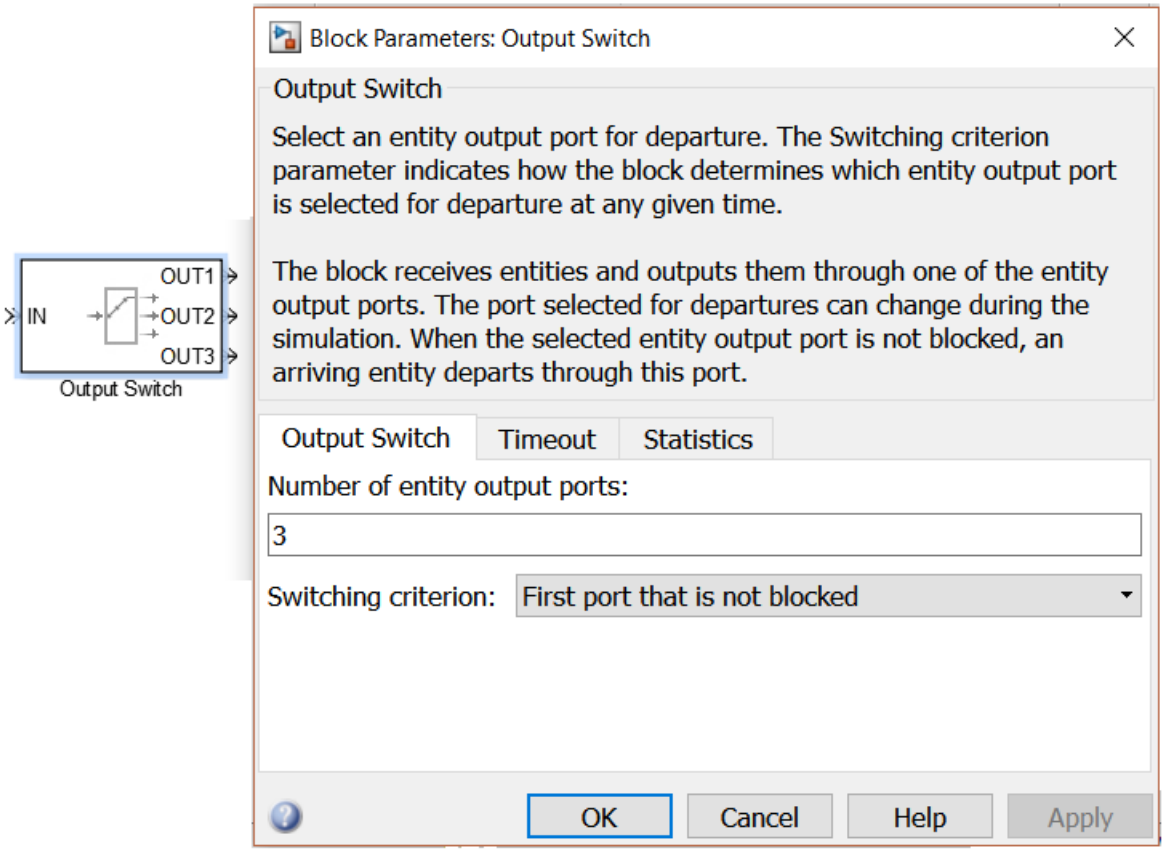


Рис. Блок Output Switch тайого діалогове вікно

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 сигнал на основі події. Цей портдоступний тільки в тому випадку, якщо критерій переключення (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..).

Таблиця Вхідні порти блоку Output Switch

Назва	Опис	Час оновлення при виборістатистика	Порядок
#d	Кількість сутностей, які вийшли з блоку, починаючи з початку моделювання	Після виходу сутностей через порт, крім TO	3
#to	Кількість сутностей з режимом тайм-аут, які вийшли з блоку, починаючи з початку моделювання	Після виходу сутностей через порт, крім TO	2
pe	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 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 маютьобмежений 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. . Для обраного правила необхідно задати відповідні параметри, які зявляються на вкладці після встановлення режиму.

Таблиця ** 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	Створюється додатковий порт виведення сигналу, позначений як p. Сигнал на цьому порту приймає цілих

Через сигнальний порт p	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 да вірність сигналу при його використанні для визначення способу обробки прибуваючого об'єкта. Необхідно бути особливо обережним, коли сигнал знаходиться в зворотному 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.). Параметр доступний, лише при встановленні критерію переключення на потік з сигналу p та вибору об'єкту Store.

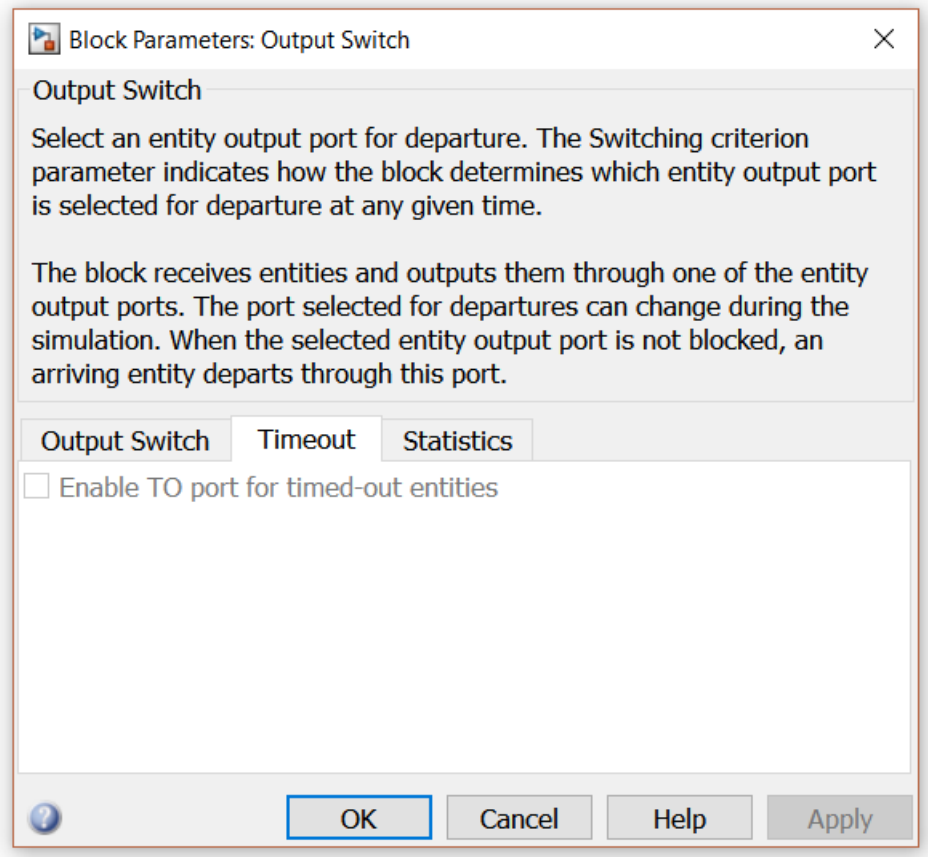


Рис. Закладка статистики блоку Output Switch

Опція стає доступною, якщо час перебування об'єкту в цьому блоці вичерпано.

На останній вкладці Statistics передбачені наступні поля (рис.):

- **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 переключення **Switching criterion** на From signal port p.;
- **Last entity departure port** (Останній вихідний порт) – дозволяє використовувати вихідний порт сигналу, позначений як останній.

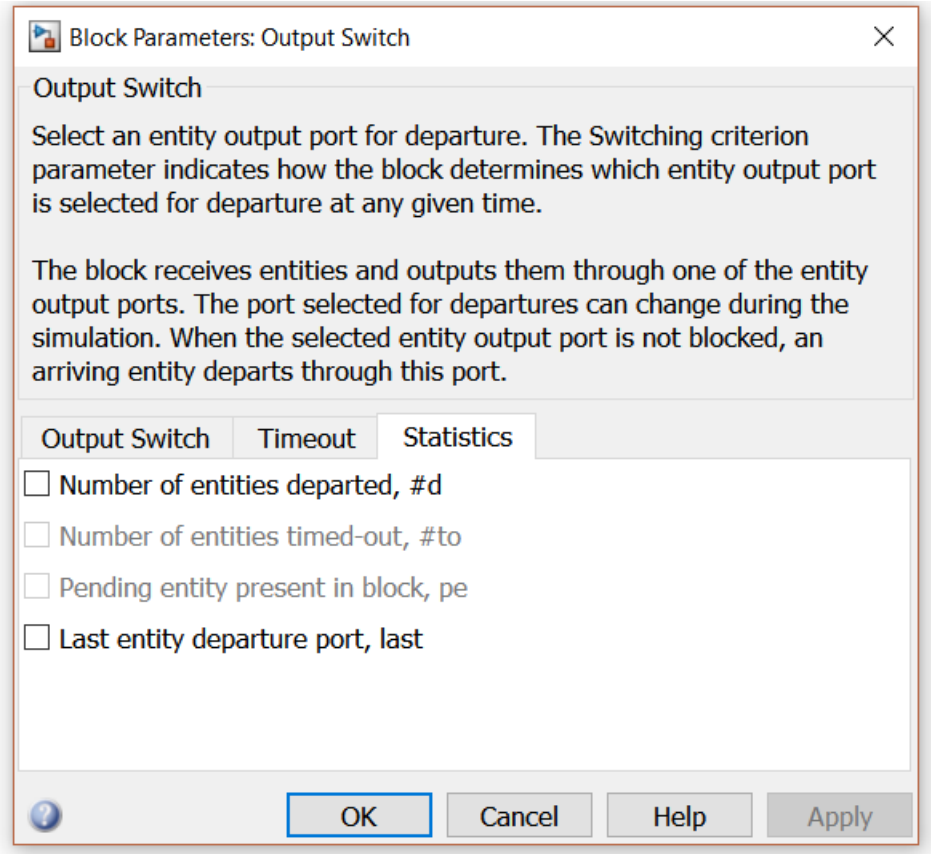


Рис. Закладка статистики діалогового вікна блоку Output Switch

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 одного 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).

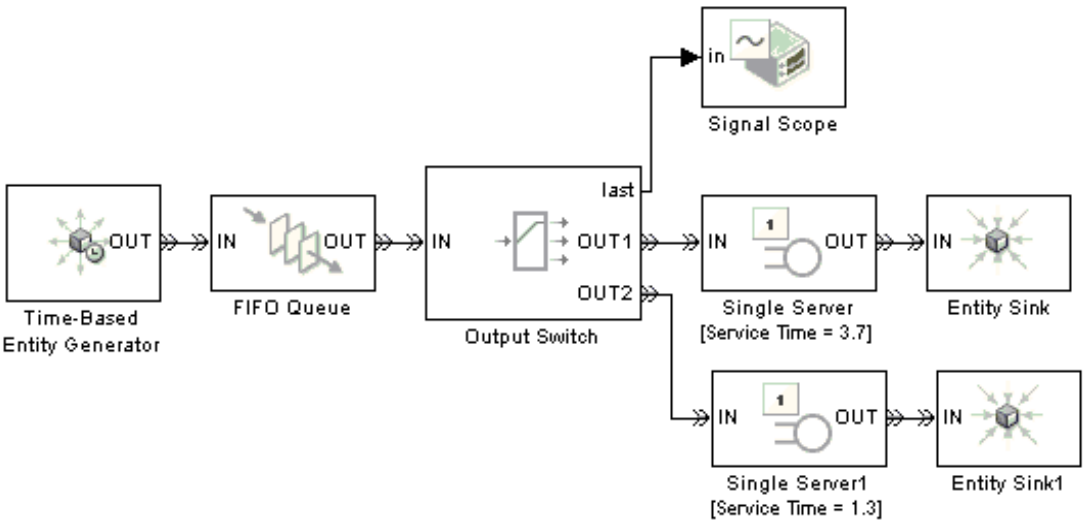


Рис. Модель з вибором першого доступного сервера

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.).

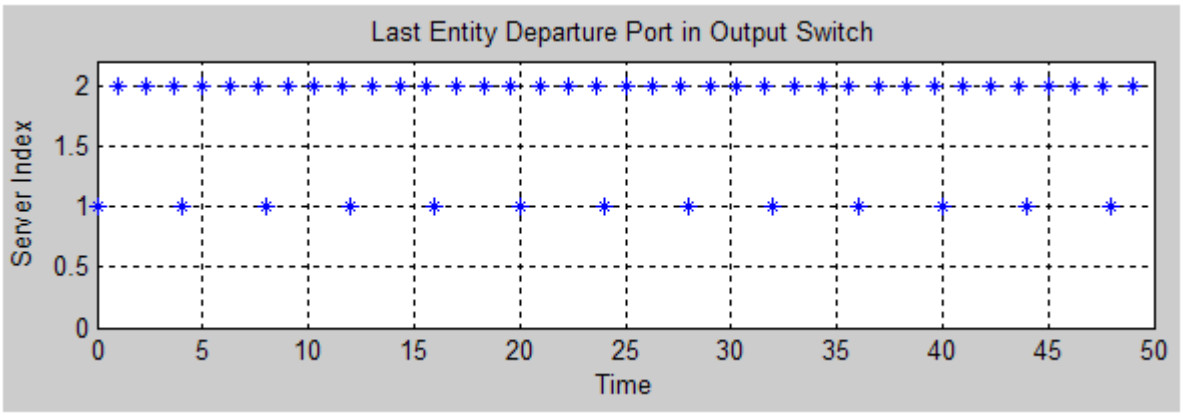


Рис. Графік індексів серверів в процесі роботи

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.

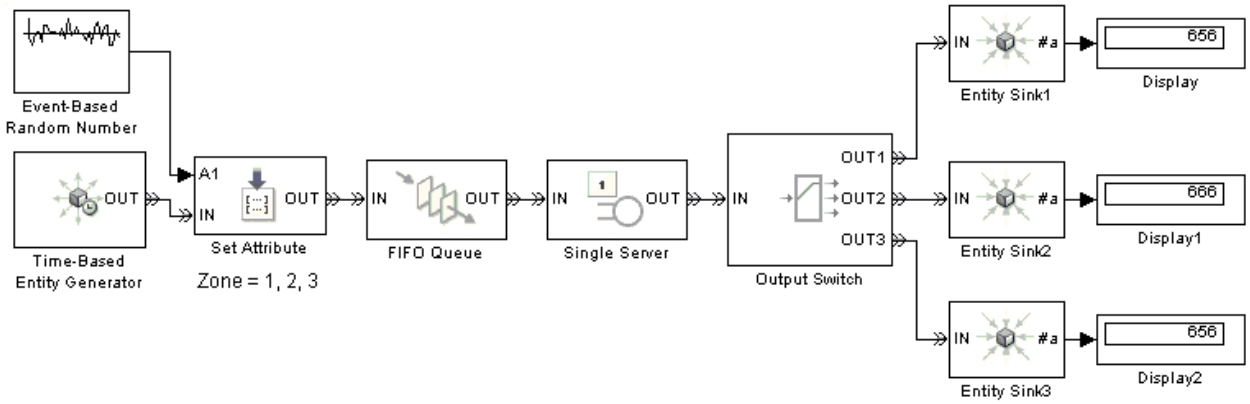


Рис. Модель сортування сутностей на три вихідні потоки

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.).

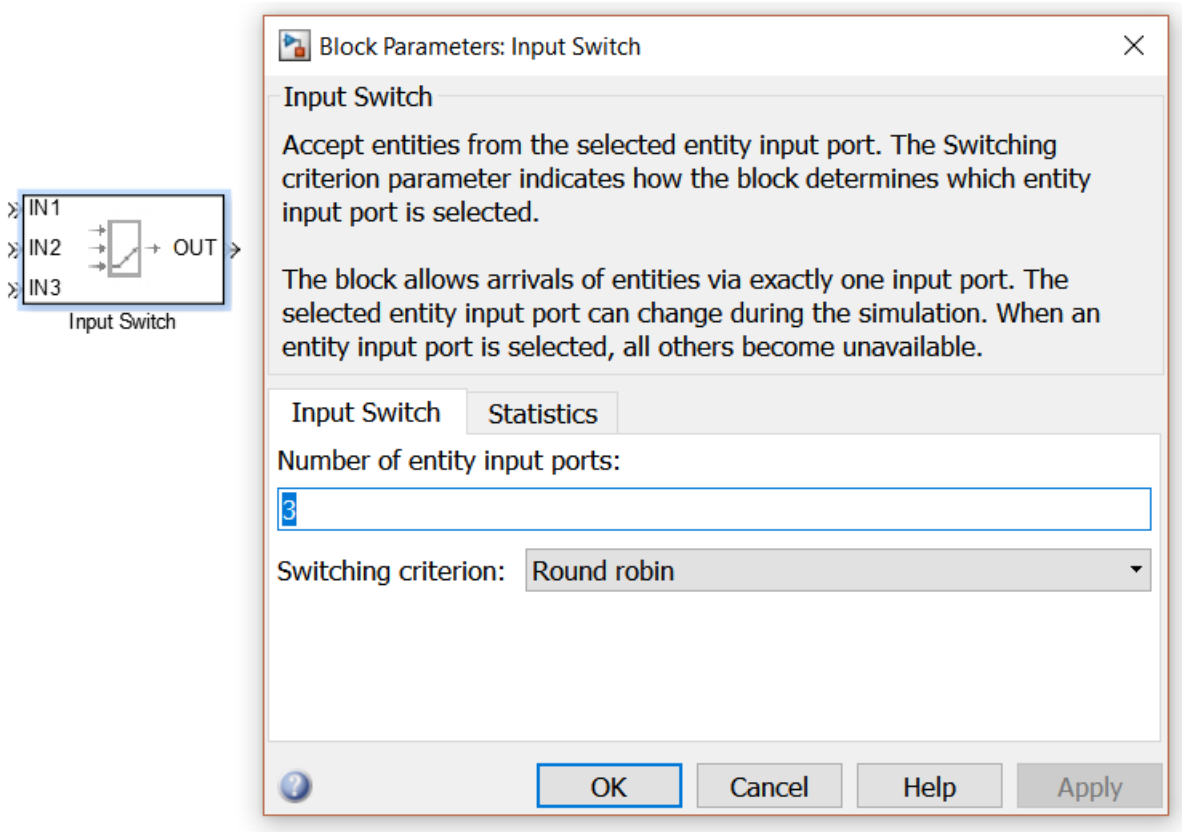


Рис. Блок Input Switch та його діалогове вікно

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 сигнал на основі події. Порт доступний тільки в тому випадку, якщо ви встановите критерій переключення **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, післявиходу 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 через який порт сутність буде виходити з блоку. Правила, які використовує блок для вибору вхідного порту об'єкта, перелічені у таблиці ** наведеній вище для блоку Output Switch.

Вкладка статистики **Statistics** дозволяє налаштовувати сигнальні вихідні порти (рис.):

- **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.

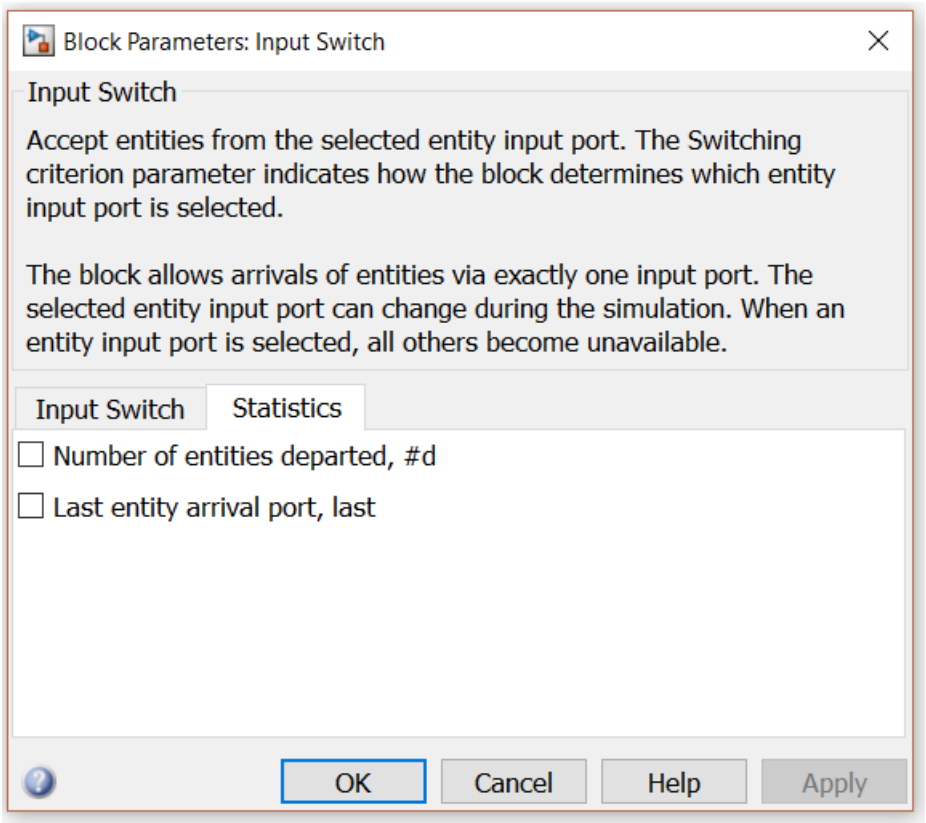


Рис. Вкладка Statistics блоку Input Switch

Приклад. Використання блоку Input Switch.

На рис. 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 щоб всі об’єкти мали таймер незалежно від їхнього маршруту перед досягненням серверу. Блок **Input Switch** управляє процесом надходження замовлень до серверу, об’єднуючи обидва потоки у єдиний потік.

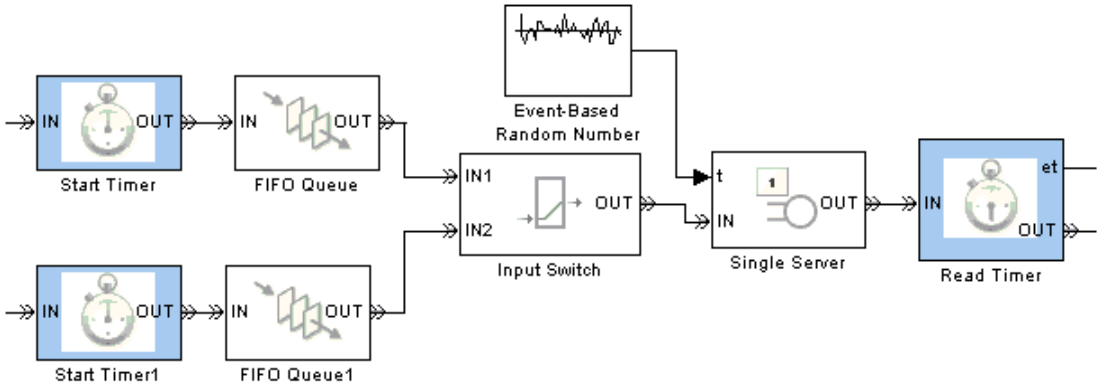


Рис. Приклад використання блоку Input Switch

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 до 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 про файлу (блок не змінює атрибути).

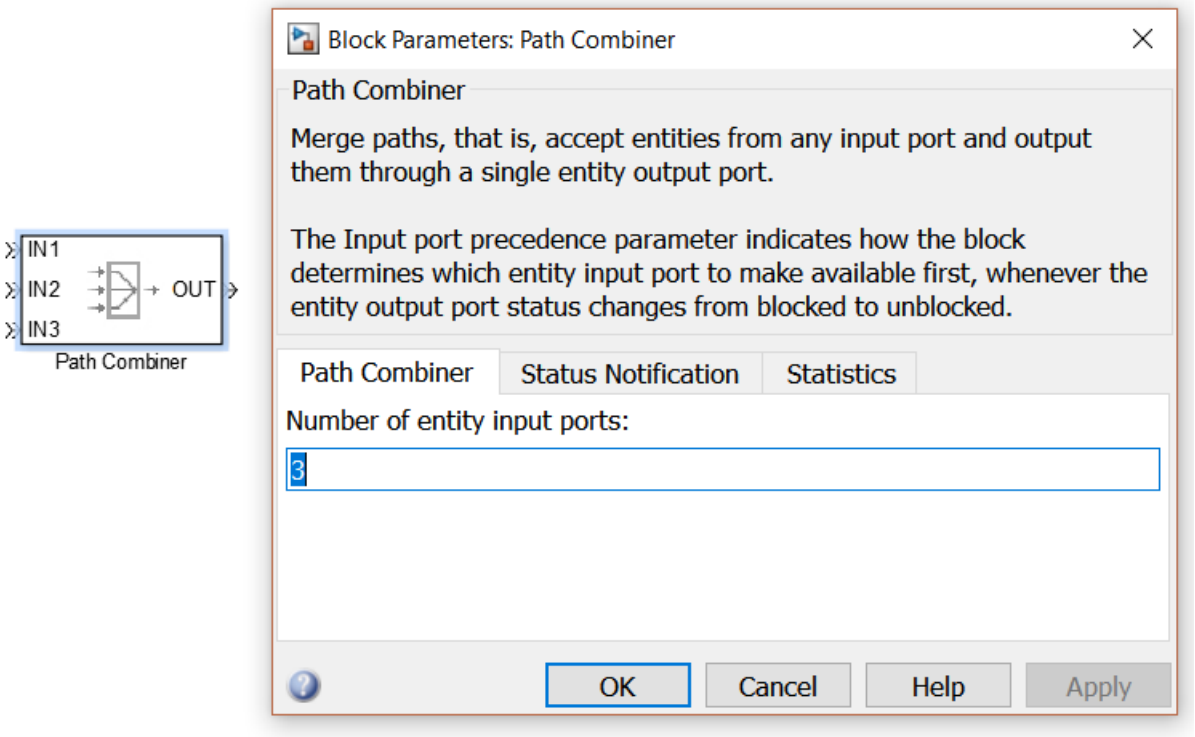


Рис. Блок Path Combiner та його діалогове вікно

На другій вкладці Status Notification також одне поле:

Input port precedence (Передача вхідного порту) – Порядок 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 атрибута, зливаються в один потік, але далі розглядаються по-різному, відповідно до значень атрибутів (рис.).

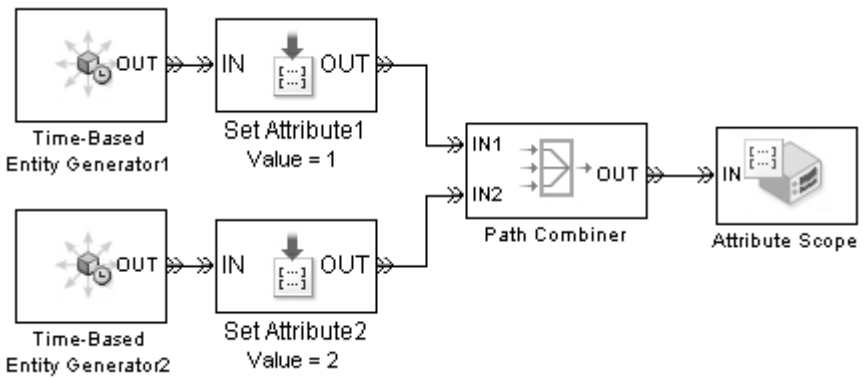


Рис. Обєднання потоків

- кілька черг об'єднуються в одну чергу.

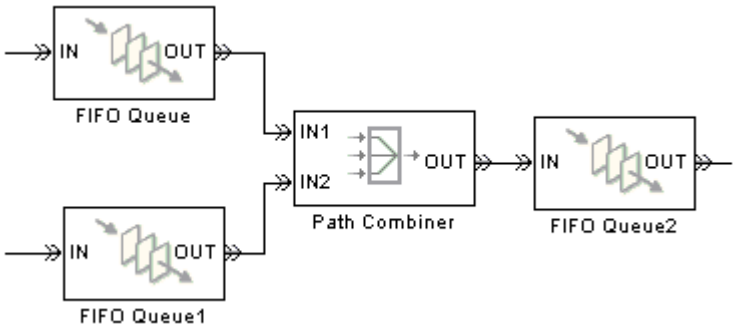


Рис. Обєднання черг

- зворотній потік входить в ту ж чергу, що й звичайний потік.

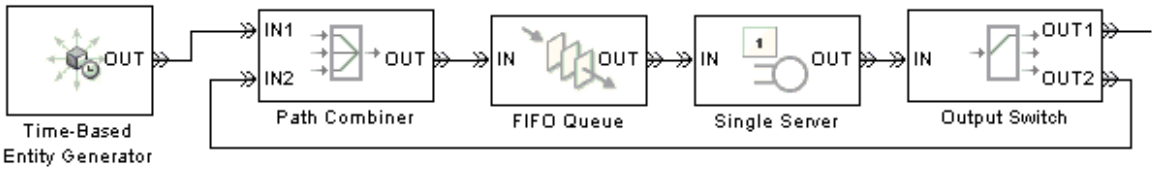


Рис. Обєднання вхідного і зворотнього потоків

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'єкту встановлені обмеження за часом (таймаут), то всі копії мають однаковий термін завершення. Логічно, блок скасовує подію тайм-ауту для сутності, що надходить прибуваючого 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 портсутностей **IN**. Кількість вихідних портів для сутностей **OUT1**, **OUT2**, **OUT3** визначається користувачем. Також для блоку є доступними два сигнальних 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.

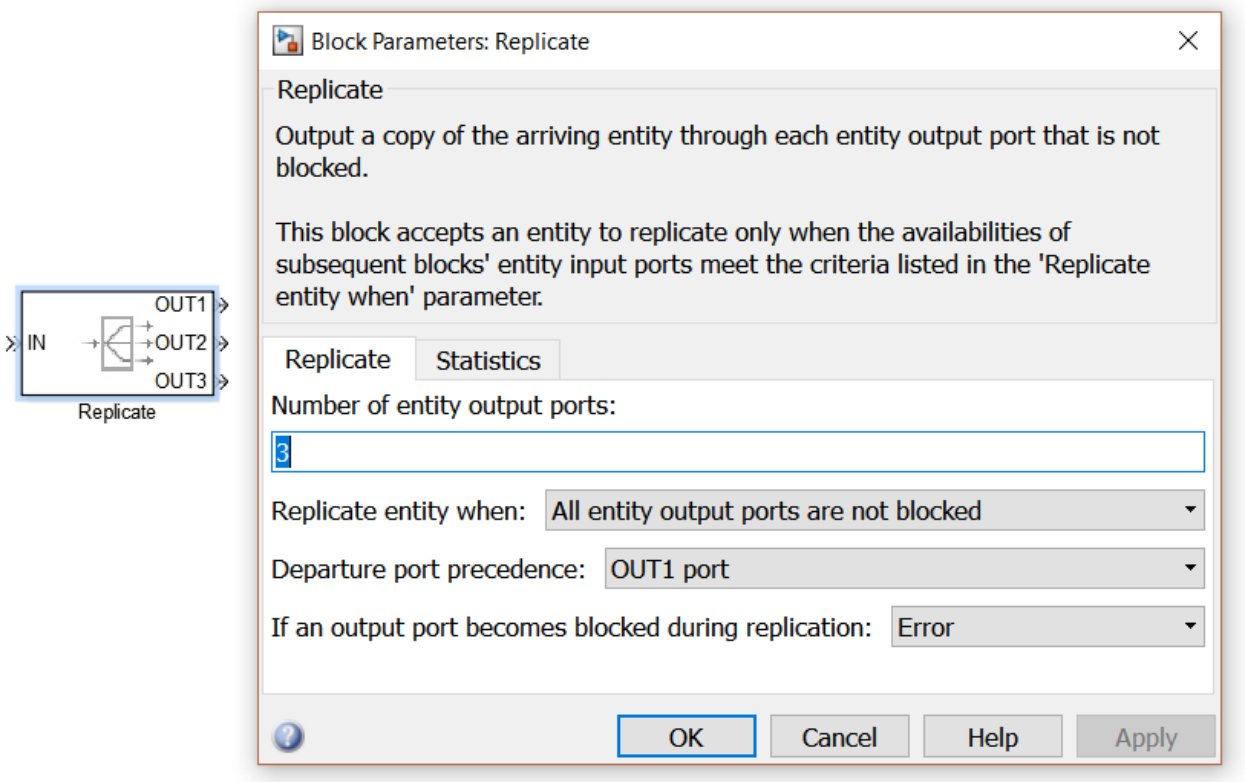


Рис. Блок Replicate та його діалогове вікно

Поля головної вкладки Replicate діалогового вікна дозволяють налаштовувати процедуру копіювання:

- **Number of entity output ports** – визначає кількість копій;
- **Replicate entity when** – визначає доступність блоку прибуваючим сутностям, коли принаймні один вихідний порт не блокується, або лише тоді, коли всі вихідні порти не заблоковано. 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..

Таблиця

Значення	Опис	ПРиклад
OUT1 port	копії відходять через вихідні порти OUT1, OUT2, OUT3, ... у зазначеній послідовності	Послідовність виходу завжди OUT1, OUT2, OUT3, ... протягом моделювання.
Round robin	перша копія відправляється через порт, що мав перевагу у 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 час аналогічний першому і т. д.
Equiprobable	перша копія відходить 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** – невідємне ціле число, яке ініціалізує генератор випадкових чисел. Поле доступне лише при встановленні значення Equiprobable.
- **If an output port becomes blocked during replication –** визначає, чи блок видає повідомлення, 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.) містить два поля, які визначають доступ до вихідних сигнальних портів (табл..)

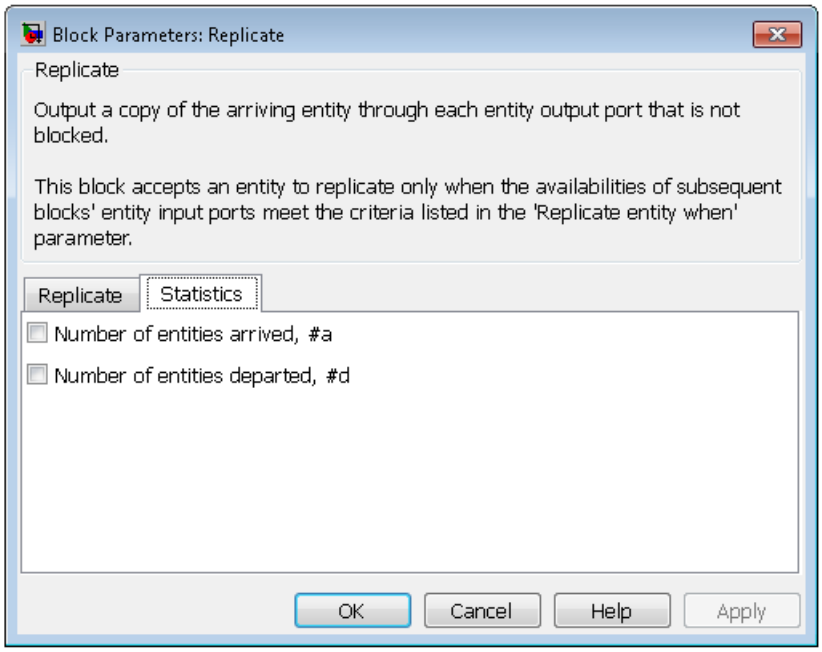


Рис Вкладка статистики блоку Replicate

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

.

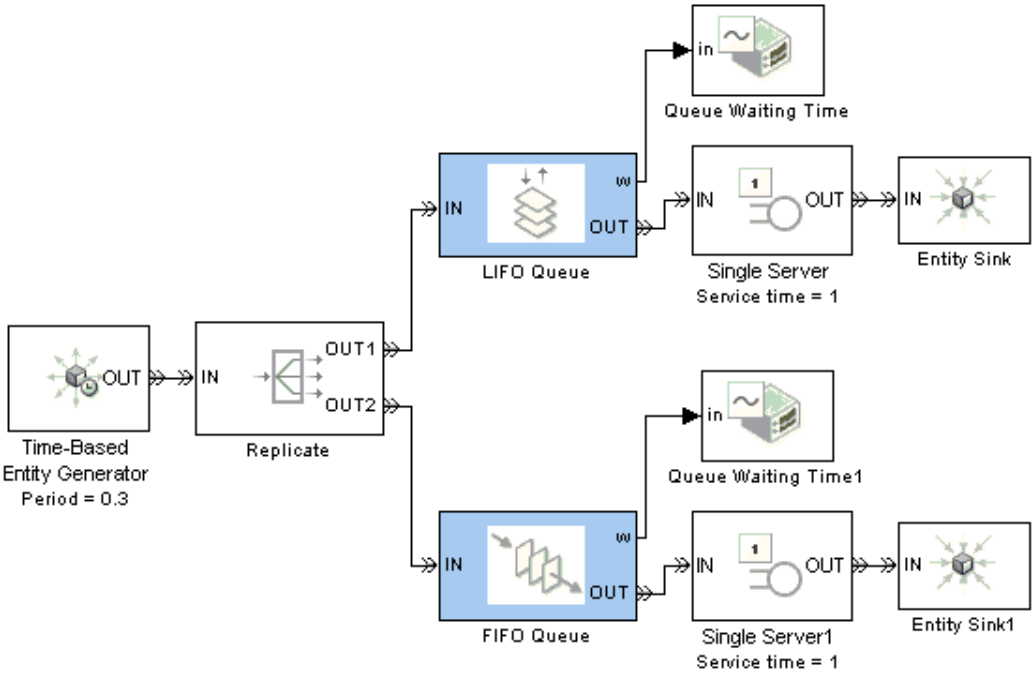


Рис. Модель із застосуванням блоку Replicate для демонстрації різниці між різними типами черги

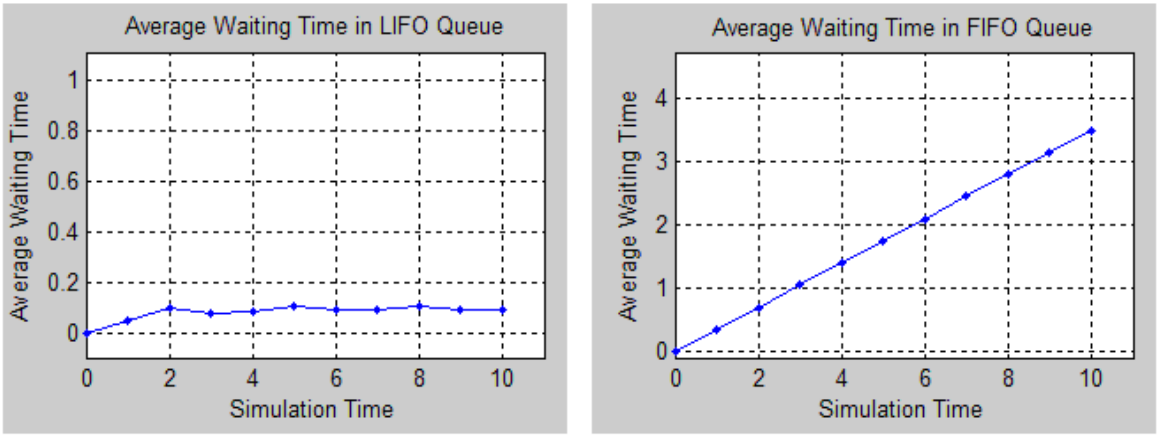


Рис. Результати виконання моделі (рис.)

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 основний 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 основний 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 настройками вихідних портів.

Для створення в моделі підсистеми можна скористатися двома способами:

1. Скопіювати блок підсистеми Atomic Subsystem з бібліотеки в модель (рис.). 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 монолітної, змінивши її параметри, або керованою, додавши керуючий елемент. Скасувати угруповання блоків в підсистему можна командою Undo.

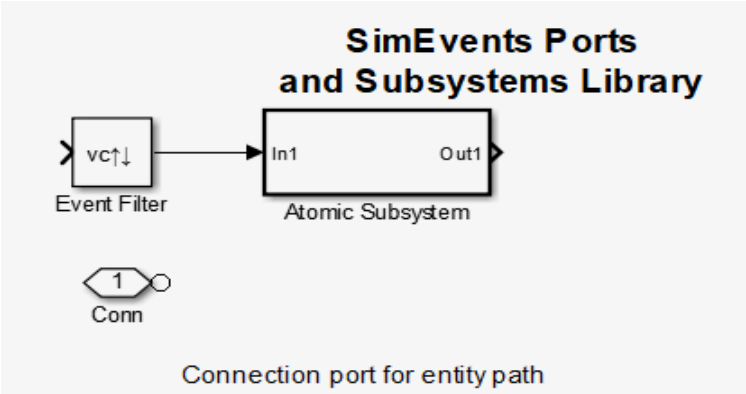


Рис. Блоки бібліотеки Ports & Subsystems

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.) 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.

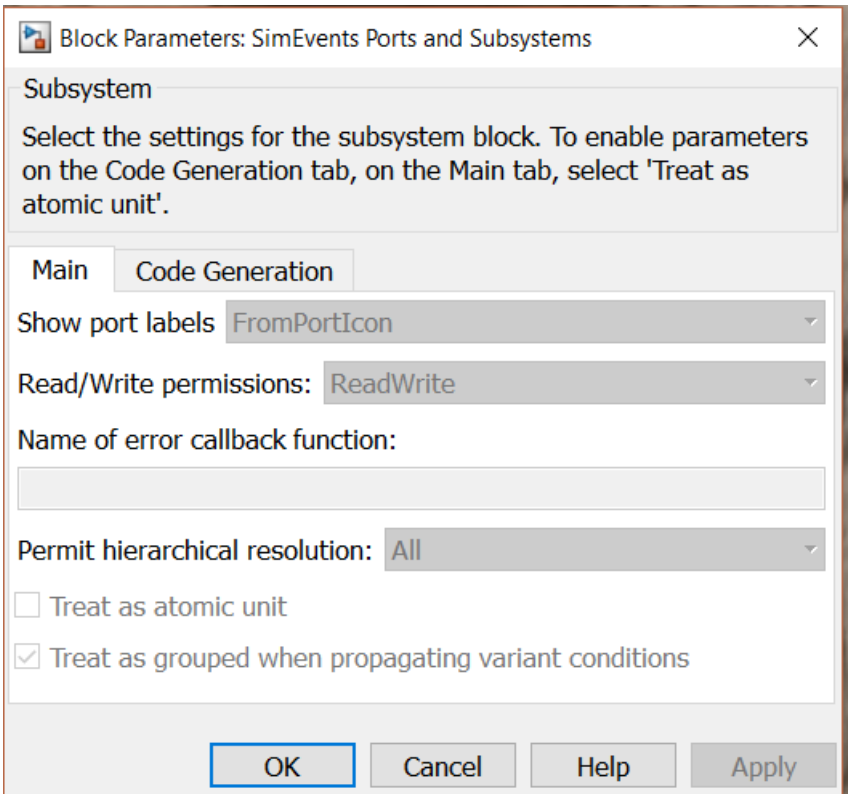


Рис. Головна закладка діалогового вікна блоку Atomic Subsystem

- **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 екземпляра 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 імя 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 приймат 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 програмноъ одиниці при визначенні порядку виконання блочних методів. Поле може бути встановлене врежим **On** або **Off**. За замовчуванням встановлюэться значення **Off.**
- **Treat as grouped when propagating variant conditions** – визначає обробку підсистеми як програмної одиниці при змінних умовах в алежності 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.):

- Function packaging – визначення формату коду, який буде згенерований для атомної (невіртуальної) підсистеми.

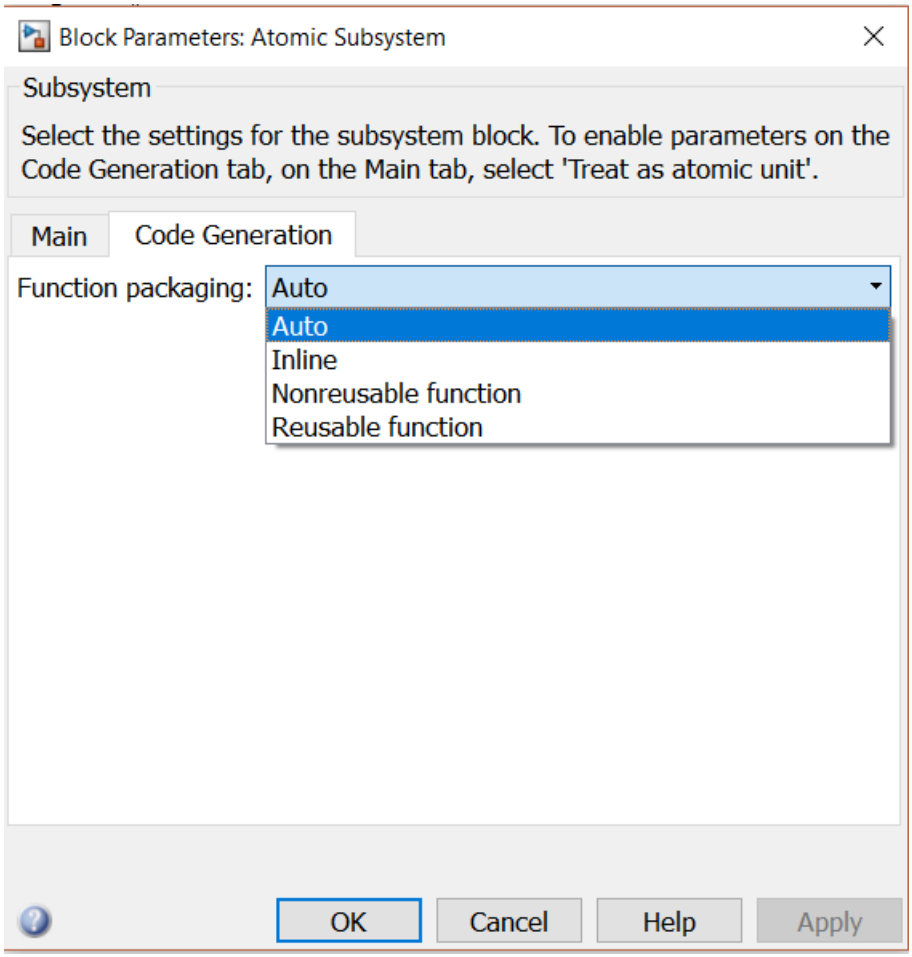


Рис. Закладка Code Generation діалогового вікна блоку Atomic Subsystem

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.

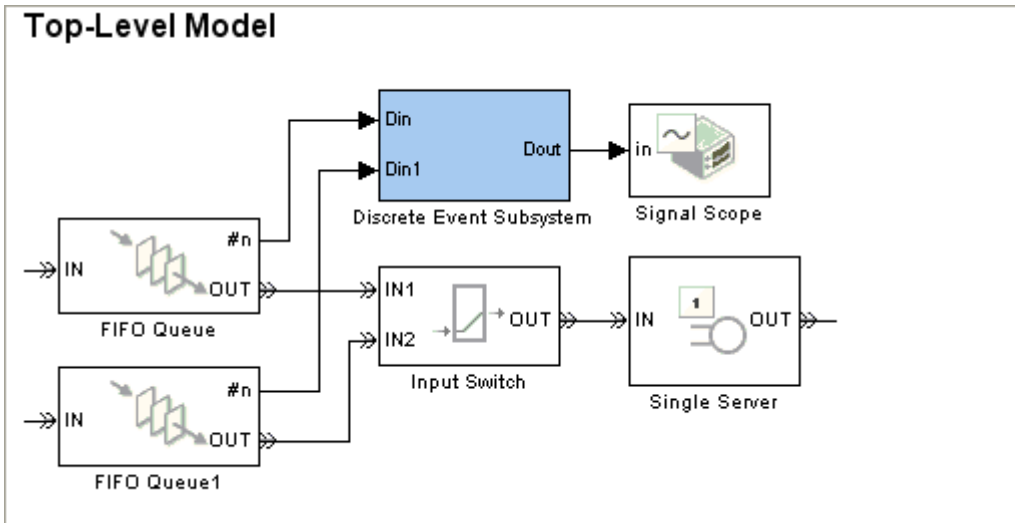


Рис. Представлення імітаційної моделі

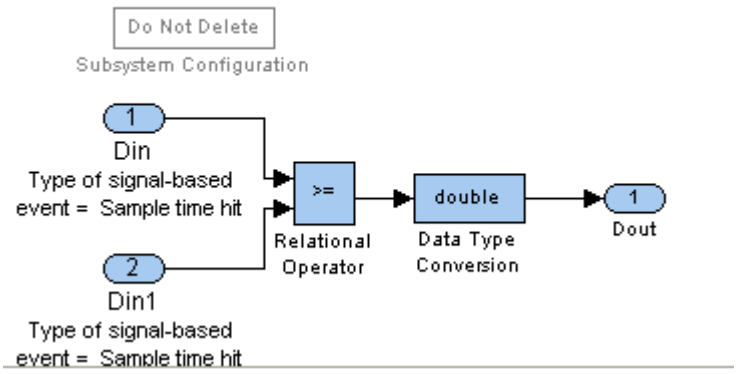


Рис. Підсистема Discrete Event Subsystem

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:

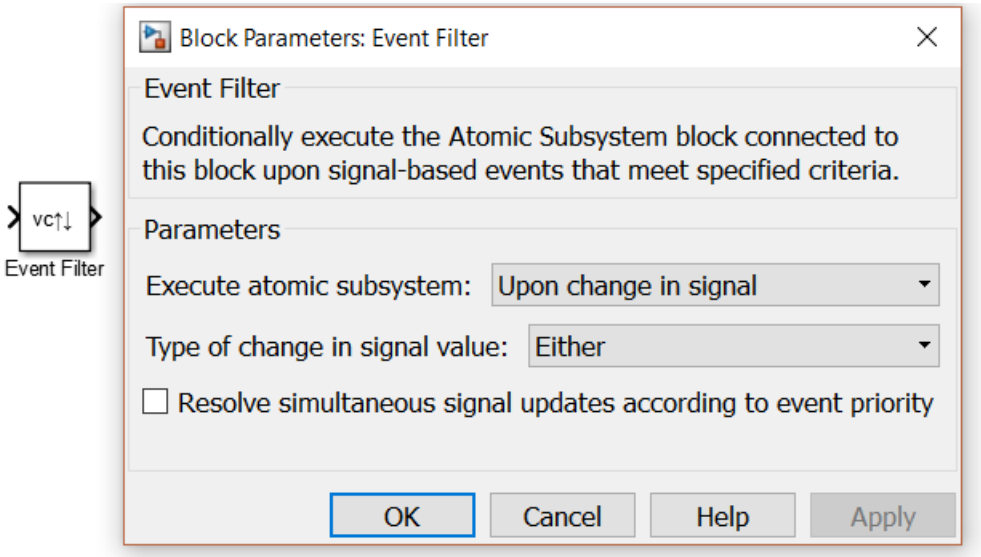


Рис. Блок Event Filter та його діалогове вікно

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. Ви доступне, лише якщо встановлено **Execute atomic subsystem** (Виконати атомну підсистему) у **Upon change in signal** (після зміни сигналу).

Приклади

Приклади побудови імітаційних моделей

Приклад 1

.1 Постановка задачі

В центрі обслуговування банку працює два оператора. 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 експоненціальним законом. 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 пакета Simulink. Ввести в модель вихідні дані і зробити необхідні настройки моделі.

Необхідні параметри системи представити у вигляді графіків.

2.1 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.

2.2 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, яка використана на зберігання сутності);
- приймач оброблених замовлень (Entity Sink);
- дисплей, що відображає кількість оброблених замовлень;
- три блоки Signal Scope для візуалізації процесу моделювання, які зєднуємо з відповідними вихідними портами блоків черги та сервера;
-. блок осцилографа Scope для виводу графіків в одному вікні;
- шину обєднання потоків для побудови графіка Scope.

Виконаємо необхідні з'єднання блоків і отримаємо схему моделі, представлену на рис. 2.1

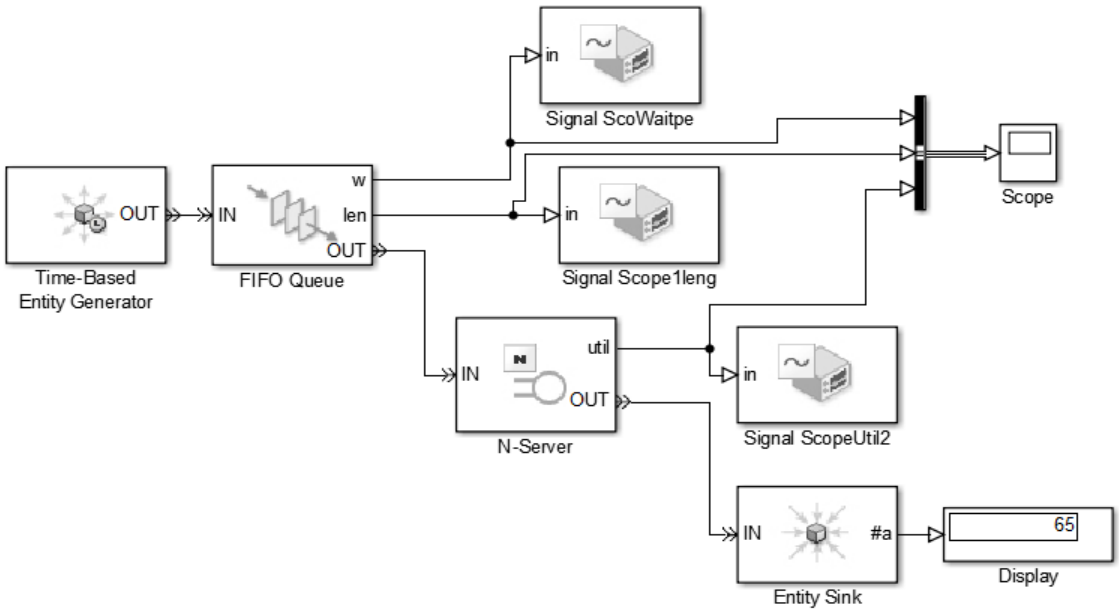
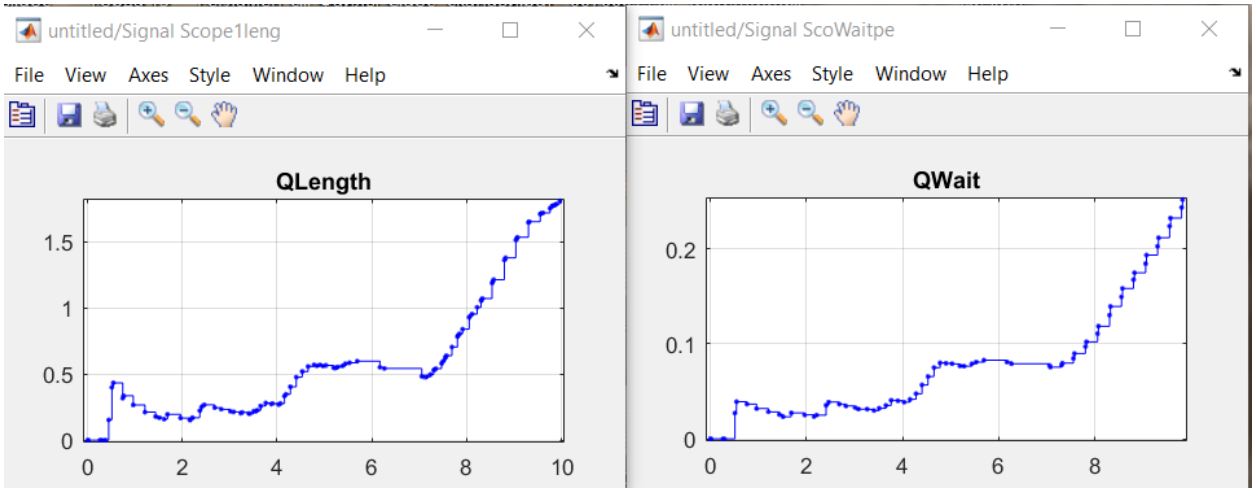
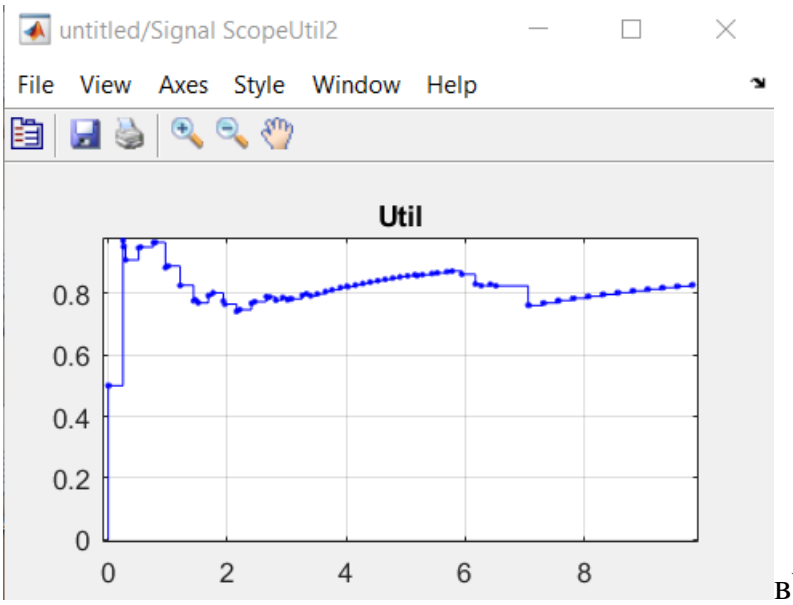


Рисунок 2.1 - Схема імітаційної моделі

Результати моделювання представленф на рис.



а) б)



в)

Рис. Графічне представлення результатів моделювання: а) середня довжина черги; б) середній час очікування; в) доля часу моделювання, яка використана на зберігання сутності

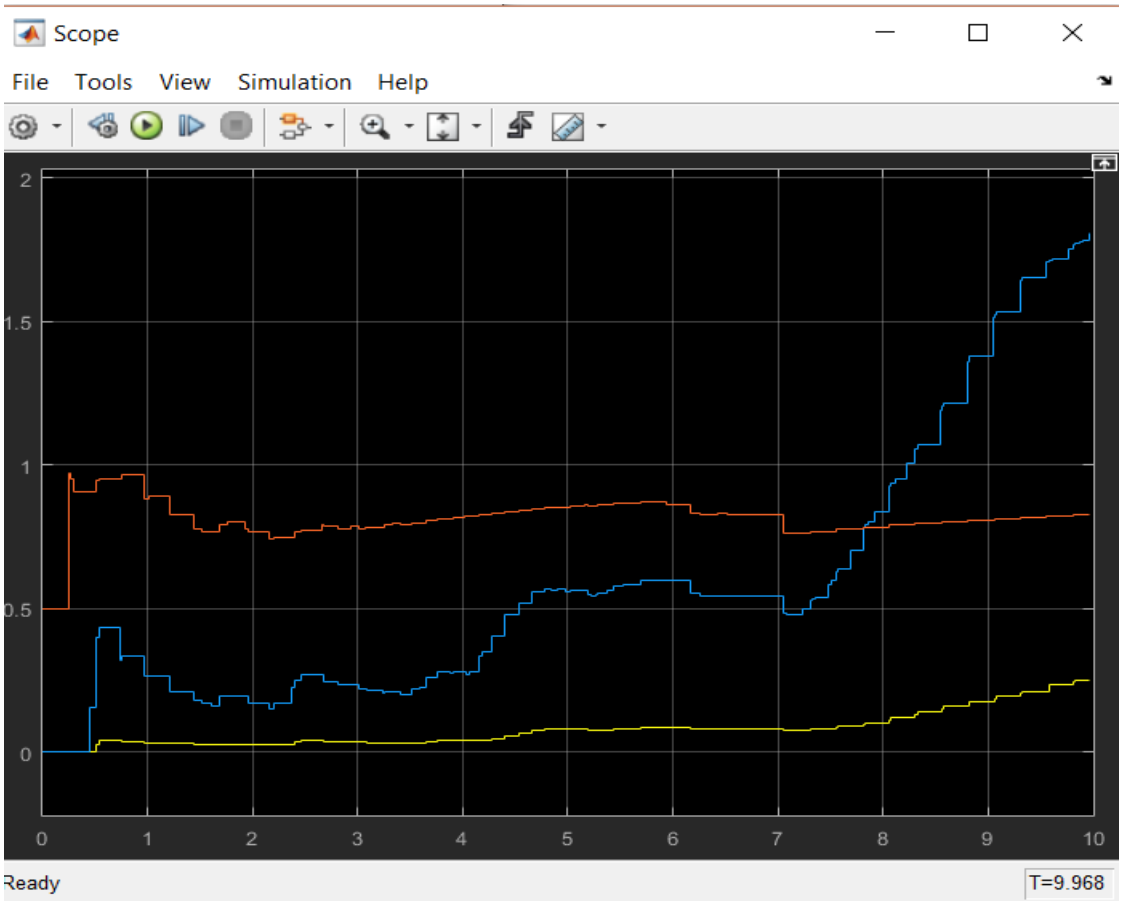


Рис. Представлення графіків у вікні Scope

Приклад 2

.1 Постановка задачі.

Створити імітаційну модель сервісного центру, де інтервали часу між надходженням клієнтів з одним майстром становлять 18±6 хв. Час обслуговування розподілений рівномірнотна 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, сереню і поточну кількість клієнтів у черзі; середній час обслуговування.

Порядок виконання роботи:

1. Розробити структуру моделі.

2. Побудувати імітаційну модель за допомогою пакета Simulink. Ввести в модель вихідні дані і зробити необхідні настройки 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.

Зєднавши всі блоки отримуємо імітаційну модель системи (рис.)

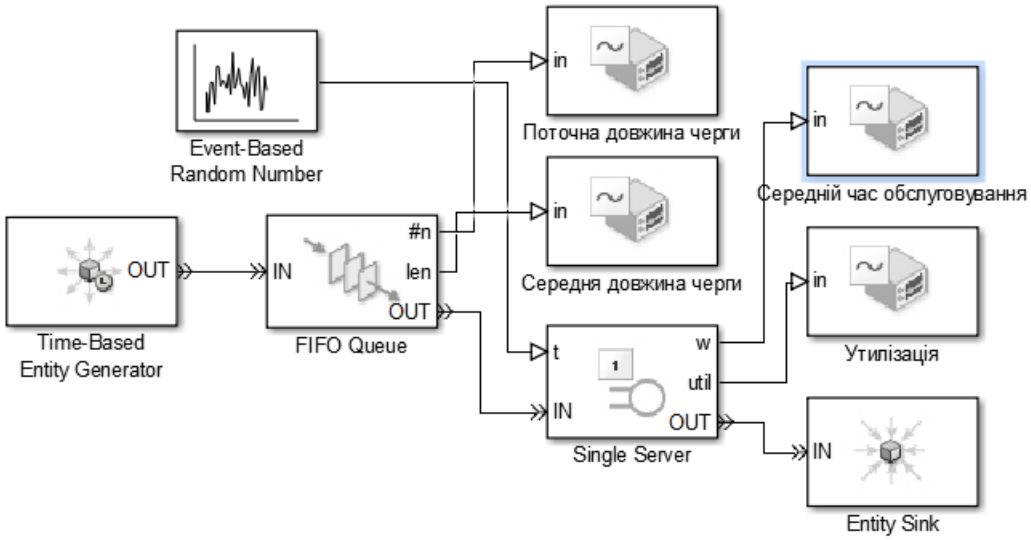
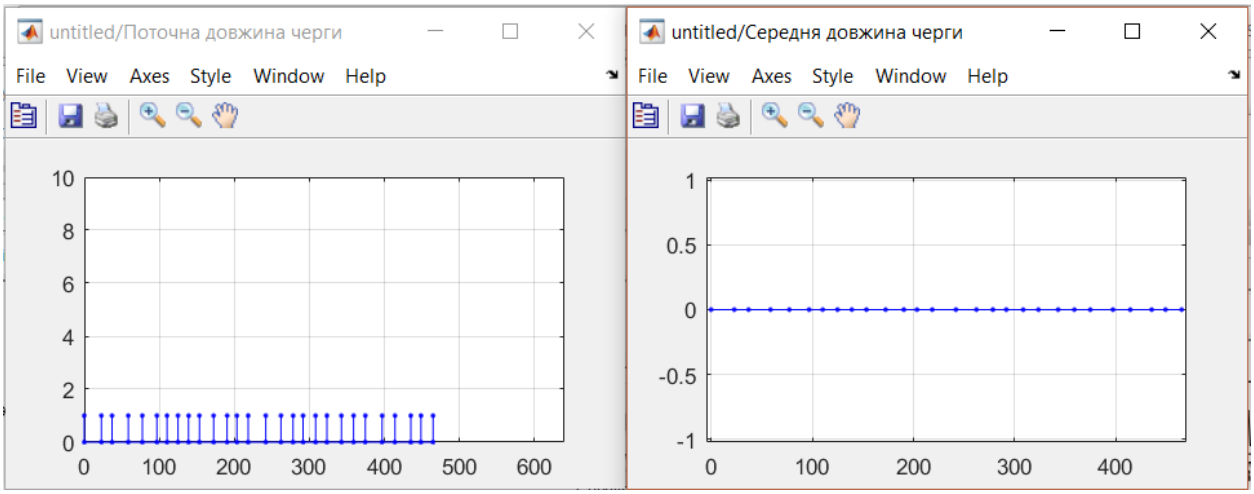


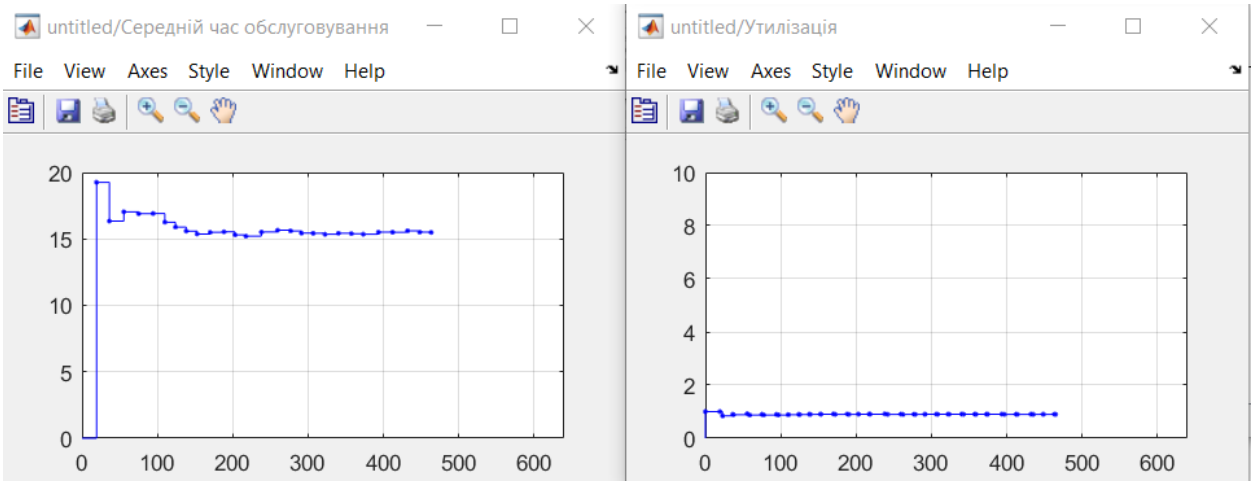
Рис. . Імітаційна модель центру обслуговування

Встановимо час моделювання 8 х 60=48о хв. І запустимо модель на виконання. Графічне представлення результатів імітації представлено на рис.



а) б)

Рис Графіки а) поточної та б) середньої довжини черги



а) б)

Рис.Графіки роботи сервісу: а) середній час обслуговування; б) зміна завантаження майстра за часом.

Задачі для самостійного виконання.

1. Автозаправочная станция имеет 4 бензоколонки. Среднее время заправки 2 мин. Входящий поток автомашин - простейший с интенсивностью 1,5 авт./мин. При всех занятых колонках требование теряется. Определите вероятность отказа и среднее число занятых колонок.

2. Покупатели магазина образуют простейший поток требований с интенсивностью 150 чел/ч. Определите наименьшее число продавцов, при которых среднее число покупателей, ожидающих обслуживания, не превысит 3.

3. В нефтеналивном порту 4 причала для заправки танкеров, которые приходят в среднем через 18 ч, а время загрузки составляет в среднем двое суток. В очереди могут стоять не более 2 танкеров. Определите пропускную способность и холостой ход порта.

4. Определите закон распределения промежутка времени между приходом двух требований в простейшем потоке требований интенсивностью L

5. Поток желающих оформить вызов врача на дом - простейший. В среднем абоненты звонят через каждые 10 с. Время приема вызова распределено по показательному закону со средним значением 12 с. Определите наименьшее число телефонов в регистратуре, при котором вызов принимается не менее чем от 90% абонентов. Считается, что в случае неудачи абонент не предпринимает больше попыток дозвониться.

7. Автоматическая мойка может принять на обслуживание одновременно 4 автомашины. В среднем машины прибывают через 2 мин, а средняя продолжительность мойки - 10 мин. В очереди могут находиться не более 6 машин. Определите вероятность того, что в системе находится хотя бы одна машина, и загруженность одной установки для мойки машин.

8. В магазине имеется 3 справочных телефона. В среднем обращаются за справками 40 чел/ч. Средняя продолжительность справочного разговора 3 мин. Издержки, связанные с работой одного телефона, - а руб./мин. Определите минимальную стоимость одной минуты разговора по телефону, при которой система неубыточна.

9. Платная стоянка для легковых машин имеет 7 мест. Найдите вероятность того, что прибывшая машина найдет свободное место, если машины в среднем прибывают через 10 мин. а занимают место на стоянке в среднем 1ч.

10. Поток деталей, сходящих с конвейера, простейший с интенсивностью 2 дет/мин. Время проверки детали контролером имеет показательный закон распределения со средним 2 мин/дет. Определите долю непроверенных деталей.

11. Город обслуживают 4 машины скорой помощи. Вызовы поступают в среднем через 4 ч. Вероятность того, что хотя бы одна машина занята, равна 0,25. Определите среднее число занятых машин и среднюю долю простоя машин.

12. В парикмахерской работают два мастера. Время обслуживания распределено по показательному закону со средним 12 мин. Ожидать обслуживание могут не более трех человек. Поток клиентов - простейший с интенсивностью 10 клиентов/ч. Найдите важнейшие операционные характеристики этой системы.

13. Система автоматической посадки самолетов одновременно может хранить данные только о шести самолетах, находящихся в воздухе. Самолеты, подлетающие к аэродрому, образуют простейший поток с интенсивностью 6 самолетов/ч. Если в момент запроса посадки система заполнена, то самолет улетает к запасному аэродрому. Аэродром имеет 3 посадочные полосы, самолет занимает полосу в среднем 20 мин. Найдите пропускную способность СМО, загруженность одной полосы, среднее число занятых полос, среднее время ожидания начала посадки после запроса.

14. Рассматривается работа автозаправочной станции (АЗС), на которой имеется 2 заправочные колонки. Предположим, что она описывается процессом размножения и гибели в стационарном режиме. Заправка каждой машины длится в среднем 3 минуты. В среднем на АЗС каждые две минуты прибывает машина, нуждающаяся в заправке. Число мест в очереди неограниченно. Все машины, вставшие на заправку, терпеливо дожидаются своей очереди. Определите: 1. Вероятность того, что на заправке находится 5 машин. 2. Вероятность того, что вновь прибывшей машине придется ждать обслуживания.

15. Закусочная на АЗС имеет один прилавок. Автомобили прибывают в соответствии с пуассоновским распределением, в среднем 2 автомобиля за 5 минут. Для выполнения заказа в среднем достаточно 1.5 минуты, хотя продолжительность обслуживания распределена по экспоненциальному закону. Найдите: а) вероятность простоя прилавка; Ь) средние характеристики; с) вероятность того, что количество прибывших автомобилей будет не менее 10.

16. Рентгеновский аппарат позволяет обследовать в среднем 7 человек в час. Интенсивность посетителей составляет 5 человек в час. Предполагая стационарный режим работы, определите средние характеристики.

17. В речном порту один причал, интенсивность входного потока - 5 судов в день. Интенсивность погрузочно-разгрузочных работ - 6 судов в день. Имея в виду стационарный режим работы, определите все средние характеристики системы.

18. Какое оптимальное число каналов обслуживания должна иметь СМО, если интенсивность потока заявок равна 3, среднее число, заявок обслуженных в единицу

времени равно 2, штраф за каждый отказ равен 5, а стоимость простоя одной линии равна 2?

19. Какое оптимальное число каналов обслуживания должна иметь СМО, если интенсивность потока заявок равна 3, среднее число, заявок обслуженных в единицу времени равно 1, штраф за каждый отказ равен 7, а стоимость простоя одной линии равна 3?

20. Какое оптимальное число каналов обслуживания должна иметь СМО, если интенсивность потока заявок равна 4, среднее число, заявок обслуженных в единицу времени равно 2, штраф за каждый отказ равен 5, а стоимость простоя одной линии равна 1 ?

21. Определите число взлетно-посадочных полос для самолетов с учетом требования, что вероятность ожидания должна быть меньше, чем 0.05. При этом интенсивность входного потока 27 самолетов в сутки, а интенсивность их обслуживания - 30 самолетов в сутки.

22. Сколько равноценных независимых конвейерных линий должен иметь цех, чтобы обеспечить ритм работы, при котором вероятность ожидания обработки изделий должна быть меньше 0.03 (каждое изделие выпускается одной линией). Известно, что интенсивность поступления заказов 30 изделий в час, а интенсивность обработки изделия одной линией - 36 изделий в час.

23. Среднее число вызовов, поступающих на АТС за одну минуту, равно 3. Считая поток пуассоновским, найдите вероятность того, что за 2 минуты поступит: а) два вызова; б) меньше двух вызовов; в) не менее двух вызовов.

24. Сколько каналов должна иметь СМО с отказами, если интенсивность потока заявок равна 2треб/час, среднее число, заявок обслуженных в единицу времени равно 1 треб/час, штраф за каждый отказ составляет 8т.руб., стоимость простоя одной линии - 2т.руб. в час?

25. Система массового обслуживания представляет собой автоматическую телефонную станцию, которая может обеспечить не более пяти переговоров одновременно. Заявка-вызов, поступившая в тот момент, когда все каналы заняты, получает отказ и покидает систему. В среднем на станцию поступает 0,8 вызовов в минуту, а средняя продолжительность одних переговоров равна 1,5 минуты. Для стационарного режима функционирования системы необходимо определите: а) вероятности состояний системы; б) вероятность отказа; в) абсолютную и относительную пропускные способности; г) среднее число занятых каналов.

27. Рабочий обслуживает три однотипных станка. Каждый станок останавливается в среднем два раза в час, а процедура наладки занимает в среднем 10 минут. В стационарном режиме функционирования системы нужно определите: а) вероятности

состояний системы; б) вероятность занятости рабочего; в) среднее количество неисправных станков; г) среднее число налаживаемых станков.

30. Специализированный пост диагностики представляет собой одноканальную СМО. Число стоянок для автомобилей, ожидающих проведения диагностики, ограниченно и равно 3. Если все стоянки заняты, то очередной автомобиль, пришедший на диагностику, в очередь на обслуживание не становится. Поток прибывающих автомобилей - пуассоновский и имеет интенсивность 0.85 автомобиля в час. Время диагностики распределено по показательному закону и в среднем составляет 1.05 час. Проведите сравнительный анализ работы СМО при S=3 и S=4.

5. Задание. Моделирование работы магазина. Требуется промоделировать работу небольшого магазина, который имеет один кассовый аппарат и одного продавца. Известны следующие параметры функционирования магазина: поток покупателей (заявок), приходящих в магазин за покупками,• равномерный; интервал времени прибытия покупателей колеблется в пределах• от 8,7 минуты до 10,3 мин. включительно, или 9,5 ± 0,8 мин; время пребывания покупателей у кассового аппарата составляет• 2,3 ± 0,7 мин. После этого покупатели подходят к продавцу для получения товара; время, потраченное на обслуживание покупателей продавцом,• составляет 10 ± 1,4 мин. Требуется определить параметры функционирования магазина: коэффициент загрузки кассира;• коэффициент загрузки продавца;• максимальное, среднее и текущее число покупателей в каждой• очереди; среднее время обслуживания в каждом канале обслуживания.•

Задача 2. Дисплейный зал имеет 5 дисплеев. Поток пользователей простейший. Среднее число пользователей, посещающих дисплейный зал за сутки, равно 140. Время обработки информации одним пользователем на одном дисплее распределено по показательному закону и составляет в среднем 40 минут. Определить, существует ли стационарный режим работы зала; вероятность того, что пользователь застанет все дисплеи занятыми; среднее число пользователей в дисплейном зале; среднее число пользователей в очереди; среднее время ожидания свободного дисплея; среднее время пребывания пользователя в дисплейном зале.

1.Контроль готовой продукции фирмы осуществляют три контролера. Если изделие поступает на контроль, когда все контролеры заняты проверкой готовых изделий, то оно остается непроверенным. Среднее число изделий, выпускаемых фирмой, составляет 20 изд./ч. Среднее время на проверку одного изделия - 7 мин.

Определить показатели эффективности отдела технического контроля. Сколько контролеров необходимо поставить, чтобы вероятность обслуживания составила не менее 97%?