Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Саратовский государственный технический университет имени Гагарина
Ю.А.»

Институт прикладных информационных технологий и телекоммуникаций

Кафедра «Информационно-коммуникационные системы
и программная инженерия»

Пояснительная записка
по дисциплине «Моделирование компьютерных сетей»

РАЗРАБОТКА ИМИТАЦИОННОЙ МОДЕЛИ СЕТИ ТОПОЛОГИИ «Кольцо+Шина+Звезда»

Выполнил студент группы б1-ИВЧТ41:
Кудряшов Алексей Владимирович

_____
подпись студента

Руководитель: к.ф.-м.н., доцент кафедры ИКСП
Ивженко Сергей Петрович

_____
подпись руководителя

Саратов 2022

Оглавление

Введение

Целью данной курсовой работы является проектирование компьютерных сетей.

Задачи курсовой работы:

1) изучение возможностей имитационного моделирования в среде AnyLogic;

2) построение четырех спроектированных моделей компьютерных сетей: шина, кольцо, звезда и их объединение;

3) получение результатов рабочей модели объединенной сети Кольцо+Шина+Звезда».

Данная курсовая работа ставит целью помощь будущим студентам в понимании моделирования и построения моделей компьютерных сетей в программном обеспечении AnyLogic.

Что такое AnyLogic

AnyLogic — программное обеспечение для имитационного моделирования, разработанное российской компанией The AnyLogic Company (бывшая XJ Technologies). Инструмент обладает современным графическим интерфейсом и позволяет использовать язык Java для разработки моделей.

Программа предоставляет встроенные библиотеки анимации, относящиеся к различным отраслям, позволяея охватить сложность практически любой системы на любом уровне детализации. Таким образом, модели AnyLogic позволяют аналитикам, инженерам и менеджерам получать более глубокое представление о взаимозависимых процессах внутри и вблизи организации и оптимизировать сложные системы и процессы в широком спектре отраслей.

Система AnyLogic позволяет аналитикам данных создавать имитационные модели с использованием различных методологий и языков моделирования, включая дискретно-событийное моделирование, агентную динамику, системную динамику, стохастическое моделирование, блок-схемы процессов, диаграммы состояний и диаграммы действий. Программный комплекс позволяет сотрудникам представлять визуальные модели с графическими объектами для визуализации транспортных средств, сотрудников, оборудования, зданий и других объектов в соответствии с бизнес-спецификациями. Встроенные ГИС-карты позволяют организациям искать и находить города, улицы, дороги, больницы, магазины и автобусные остановки для создания имитационных моделей. Платформа также предоставляет предварительно разработанные инструменты моделирования, такие как Монте-Карло, анализ чувствительности и эксперименты по изменению параметров.

Графическая среда моделирования AnyLogic включает в себя следующие элементы:

- Stock & Flow Diagrams (диаграмма потоков и накопителей) применяется при разработке моделей, используя метод системной динамики.

- Statecharts (карты состояний) в основном используются в агентных моделях для определения поведения агентов. Они также часто используются в дискретно-событийном моделировании, например для симуляции машинных сбоев.
- Action charts (блок-схемы) используются для построения алгоритмов. Применяется в дискретно-событийном моделировании (маршрутизация звонков) и агентном моделировании (для логики решений агента).
- Process flowcharts (процессные диаграммы) — основная конструкция, используемая для определения процессов в дискретно-событийном моделировании.

Описание блоков, используемых при создании моделей

Source

Рисунок 1. Вид блока Source

Создает агентов. Обычно используется в качестве начальной точки потока агентов.

Агенты могут быть стандартными или заданными пользователем агентами типа Agent. Вы можете сконфигурировать блок так, чтобы он создавал агентов других типов, указав конструктор нужного типа в параметре Новый агент, а также задать действие, которое должно выполняться перед тем, как новый агент покинет блок Source, в поле действия **При выходе**.

Агенты могут создаваться согласно заданной интенсивности (которая может изменяться динамически с помощью функции **set_rate()**), времени между прибытиями, изменяющейся во времени интенсивности, заданной с помощью расписания, расписанию, задающему точные времена и количество прибывающих агентов, или "вручную" путем вызова функции блока **inject()**. Например, пуассоновский поток агентов может быть промоделирован путем генерации агентов согласно заданной интенсивности, или согласно времени между прибытиями, подчиняющемуся экспоненциальному закону распределения. Может быть задано как максимально допустимое число генераций, так и число агентов, создаваемых за каждый раз.

Sink

Рисунок 2. Вид блока SInk

Уничтожает поступивших агентов. Обычно используется в качестве конечной точки потока агентов.

Для того, чтобы агенты удалялись из модели и уничтожались, нужно соединить выходной порт последнего блока процессной диаграммы с портом блока Sink или Exit.

Для успешного уничтожения агента необходимо выполнение трех условий:

Если агент находится в сети, то он должен быть удален из этой сети.

Агент не должен обладать ни одним ресурсом или сетевым ресурсом.

Если агент содержит других агентов, то они тоже должны удовлетворять вышеуказанным условиям.

Если какое-то из этих условий не выполняется, блок Sink выдает ошибку.

Delay

Рисунок 3. Вид блока Delay

Задерживает агентов на заданный период времени. Время задержки вычисляется динамически, может быть случайным, зависеть от текущего агента или от каких-то других условий.

Сразу несколько агентов (не более заданной вместимости блока **capacity**) могут быть задержаны одновременно или независимо друг от друга.

Пример задания времени задержки: пусть время обработки пакета данных (агент типа **Packet**) пропорционально размеру пакета какое-то случайное время. Тогда вы можете сделать следующее: указать Packet в качестве Типа агента блока Delay и написать **processingTimePerDataUnit * agent.size uniform(timeMin, timeMax)** в поле параметра **Время задержки**.

Если вместимость блока **Delay** меняется динамически, и количество агентов, находящихся в блоке в данный момент времени, превышает значение вместимости блока, то блок **Delay** даст каждому такому агенту завершить его время ожидания, и не будет принимать новых агентов до тех пор, пока их количество в блоке не станет меньше нового значения вместимости блока.

Вы можете выполнять любые необходимые вам действия над содержащимися в блоке агентами, например, узнавать, сколько времени еще агентам осталось находиться в блоке, и даже извлекать агентов, не дожидаясь того, когда их времена задержек истекут.

Блок **Delay** может отображать анимации находящихся в нем агентов как движущимися вдоль заданного пути, так и ожидающими в заданных точках. В том случае, если агенты отображаются движущимися, за время их задержки они должны будут пройти весь путь выбранной фигуры анимации блока **Delay**.

Вы можете динамически увеличить длительность задержки для тех агентов, которые уже находятся в состоянии задержки, используя функцию **extendDelay()**.

Queue

Рисунок 4. Вид блока Queue

Блок **Queue** моделирует очередь агентов, ожидающих приема блоками, следующими за данным в потоковой диаграмме, или же хранилище агентов общего назначения. При необходимости вы можете задать максимальное время ожидания агента в очереди. Вы также можете программно извлекать агентов из любых позиций в очереди.

Поступающие агенты помещаются в очередь в определенном порядке: либо согласно правилу **FIFO** (в порядке поступления в очередь — по умолчанию), **LIFO**, либо согласно приоритетам агентов. Приоритет может либо явно храниться в агенте, либо вычисляться согласно свойствам агента и каким-то внешним условиям. Очередь с приоритетами всегда примет нового входящего агента, вычислит его приоритет и поместит его в очередь в позицию, соответствующую его приоритету. Если очередь будет заполнена, то приход нового агента вынудит последнего хранящегося в очереди агента покинуть

блок через порт **outPreempted** (но если приоритет нового агента не будет превышать приоритет последнего агента, то тогда вместо него будет вытеснена именно этот новый агент).

В режиме таймаута агент покинет очередь через порт **outTimeout**, если проведет в очереди заданное количество времени.

Вы можете извлекать любых агентов из очереди с помощью функции **remove()**. В некоторых случаях, например, когда блок **Queue** используется для моделирования хранилища с произвольным доступом, имеет смысл оставлять порт out несоединенным и извлекать агентов из очереди с помощью функции **remove()**. Извлеченные из очереди агенты могут быть вставлены в другие процессы с помощью блоков **Enter**.

Вы можете расширять функциональность блока **Queue**, выполняя необходимые вам действия в момент помещения агента в очередь, при достижении им начала очереди, а также при уходе агента из блока через любой из его портов. Пожалуйста, обратите внимание, что на момент вызова кода параметра **onEnter** агент уже будет помещен в очередь, а на момент вызова кода параметров **onExit** или **onExitTimeout** будет удален из очереди.

Вы можете динамически изменять вместимость очереди.

Агенты в очереди могут отображаться на презентации как стоящими один за другим (тип анимации **Путь**), так и стоящими в как-то по-другому заданных местах.

SelectOutput

Рисунок 5. Вид блока SelectOutput

Блок направляет входящих агентов в один из двух выходных портов в зависимости от выполнения заданного (детерминистического или заданного с помощью вероятностей) условия. Условие может зависеть как от агента, так и

от каких-то внешних факторов. Поступивший агент покидает блок **SelectOutput** в тот же момент времени.

Может использоваться для сортировки агентов согласно заданному критерию, для случайного разделения потока агентов на части и т.д. Предположим, например, что в вашей модели моделируются клиенты (с помощью агентов типа **Customer**, у которого есть параметр **vip** типа **boolean**). Тогда если вы захотите направлять VIP клиентов в верхний порт (True), а всех остальных — в нижний (**False**), то вы должны задать условие **agent.vip** и выбрать тип **Customer** в качестве типа агента блока **SelectOutput**. Более сложный случай: вы хотите перенаправить в верхний порт блока только 80% VIP клиентов, а оставшиеся 20% (и всех остальных) — в нижний порт. Тогда условие будет выглядеть как **agent.vip && randomTrue(0.8)**.

Иногда требуется иметь более двух выходов. Мы предоставляем вам два блока для направления агентов в разные отделы диаграммы процесса: блоки **SelectOutput** и **SelectOutput5**. Блок **SelectOutput5** имеет пять выходных портов, соответственно, он может направлять агентов в пять выходов. Используя блоки **SelectOutputIn** и **SelectOutputOut**, вы можете создать один большой блок **SelectOutput** с требуемым количеством выходов.

SelectOutput5

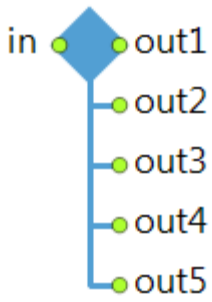

Рисунок 6. Вид блока SelectOutput5

Блок направляет входящих агентов в один из пяти выходных портов в зависимости от выполнения заданных (детерминистических или заданных с помощью вероятностей) условий.

У блока есть три режима работы:

- Условия — Пользователю предлагается задать 4 условия. Когда в блок поступит новый агент, эти условия будут поочередно вычисляться одно за другим. Если будет выполнено условие 1, то агент покинет блок через порт 1, если нет, то будет проверяться условие 2, и так далее. Если не будет выполнено ни одно из условий, то агент покинет блок через последний порт. Каждое условие может зависеть как от агента, так и от каких-то внешних факторов.
- Вероятности — Пользователю предлагается задать 5 вероятностей для пяти выходных портов (если их сумма не равна 1, то они нормализуются). Агент будет перенаправляться на тот или другой выходной порт, выбор которого будет случайно определяться в соответствии с заданными вероятностями.
- Номер выхода — Пользователь должен предоставить выражение, которое возвращает целое число в диапазоне от одного до пяти. Когда агент попадает в этот блок, выражение вычисляется, и результат означает номер выходного порта, через который агент должен покинуть этот блок. Выражение может зависеть как от агента, так и от каких-то внешних факторов.

Поступивший агент покидает блок **SelectOutput5** в тот же момент времени.

Блок может использоваться для сортировки агентов согласно заданному критерию, для случайного разделения потока агентов на части и т.д.

Иногда требуется иметь более пяти выходов. Используя блоки **SelectOutputIn** и **SelectOutputOut**, вы можете создать один большой блок **SelectOutput** с требуемым количеством выходов.

Создание моделей в программе

Создание модели Шина

Основа модели данной топологии – кабель, соединяющий конечные точки – компьютеры. Роль такого кабеля будет играть объект – SelectOutput.

Принцип работы SelectOutput в данном случае – «Сигнал либо пойдет дальше, либо попадет на конечную точку». Реализуется данный принцип установкой распределения сигналов на основе вероятности 0.8, 0.75, 0.66, 0.5 на каждый выход. Вероятность рассчитывается исходя из количества компьютеров в сети для равномерного распределения данных, хотя и можно оставить одинаковую вероятность в 0.5.

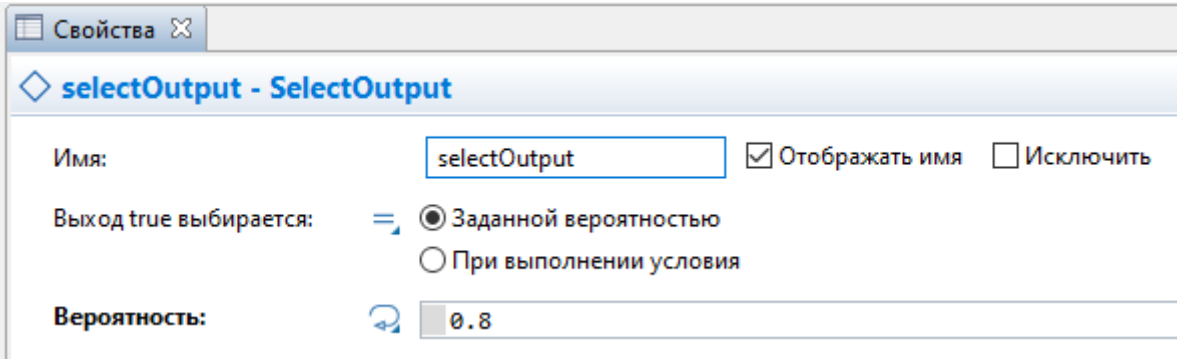

Рисунок 7. Свойства блока SelectOutput

Конечными точками в топологии «Шина» являются компьютеры, которые реализованы с помочью объектов Delay и Sink.

В свойствах объектов Delay необходимо указать:

- Тип задержки (Определенное время)
- Время задержки(triangular(0.5, 1, 1.5))
- Вместимость (1)
- Вернуть агента в исходную точку (true).

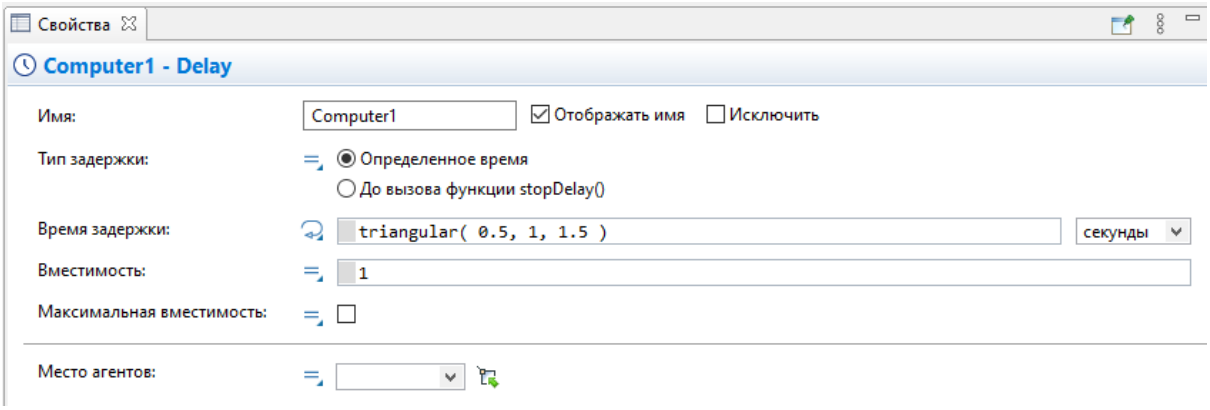

Рисунок 8. Свойства блока Delay

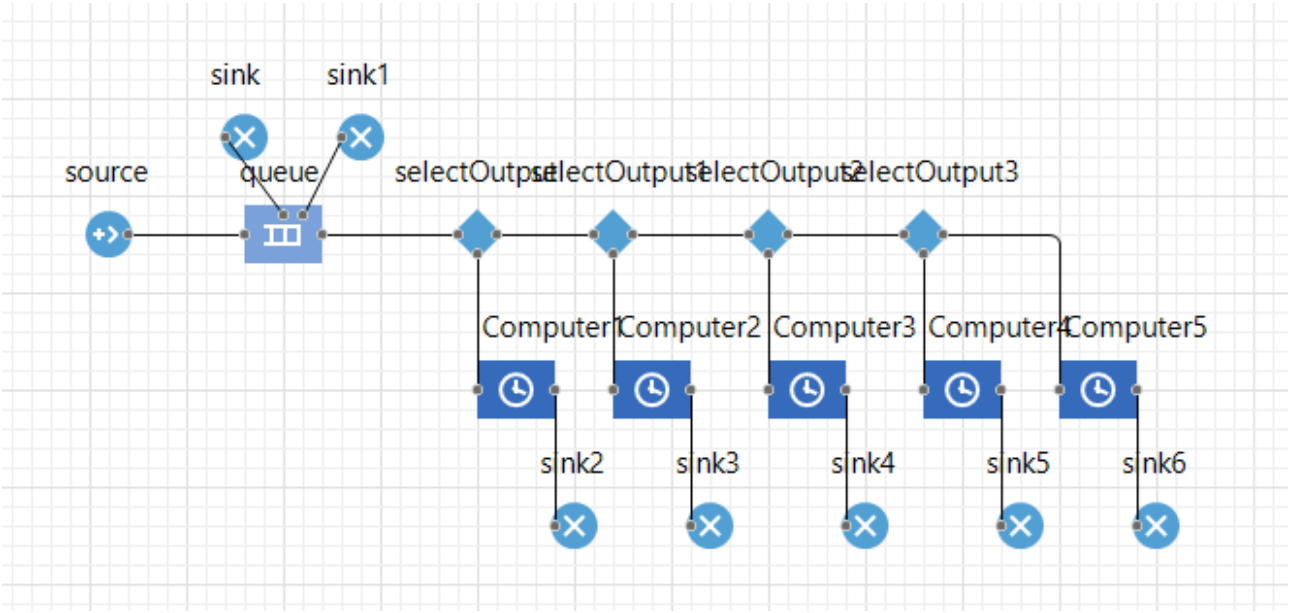

Рисунок 9. Модель сети «Шина»

Создание модели Кольцо

Модель очень похожа на топологию сети «Шина». Вероятность можно оставить такой же для равномерного распределения. Также, как и в топологии Шина в качестве компьютера у нас будут выступать объекты Delay, а в качестве выходных точек объекты Sink**.**

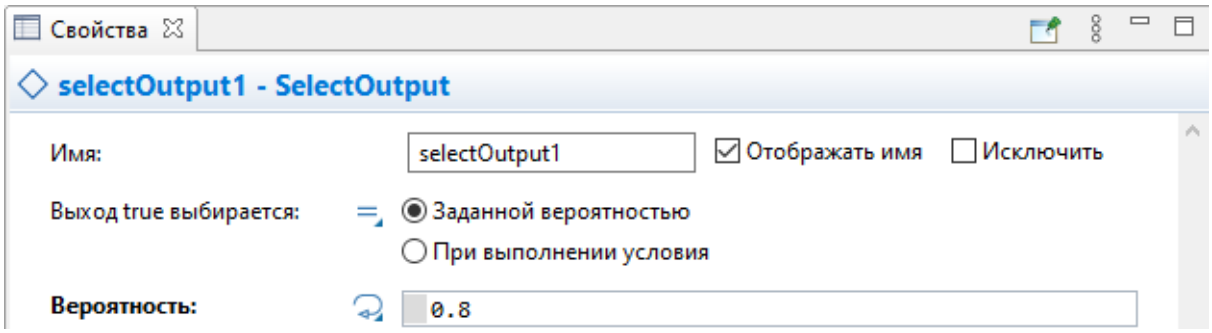

Рисунок 10. Свойства блока SelectOutput

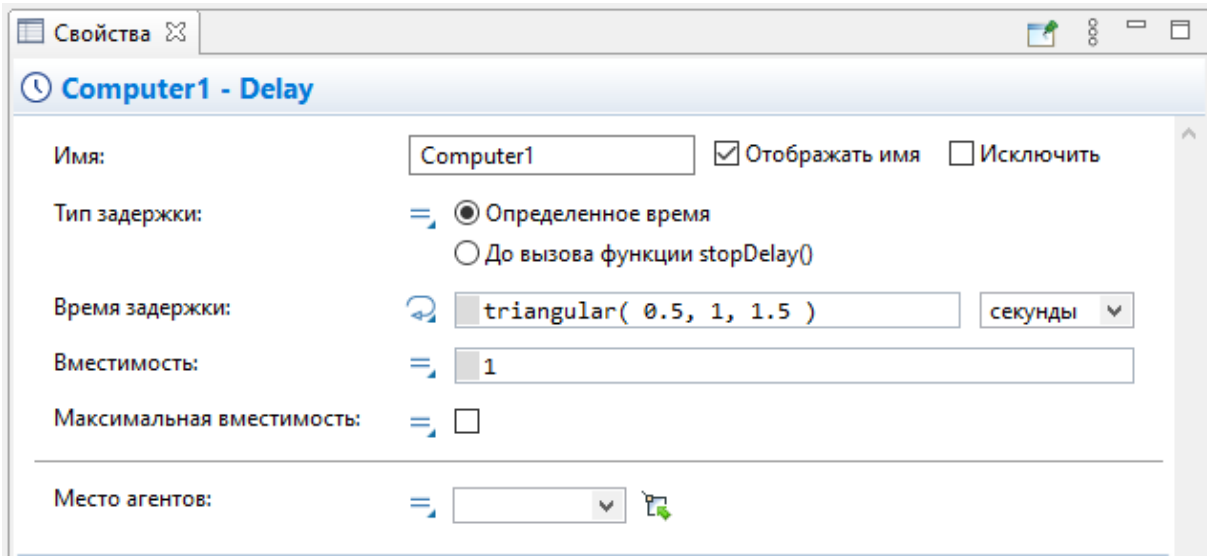

Рисунок 11. Свойста блока Delay

Для создания топологии Кольцо, необходимо создать нужное количество компьютеров (в примере компьютеров в сети равно 5, можно сделать любое другое количество) соединить их поочередно, образуя цепочку из компьютеров, последний компьютер должен замкнуть наше кольцо, подключившись к созданному первому компьютеру.

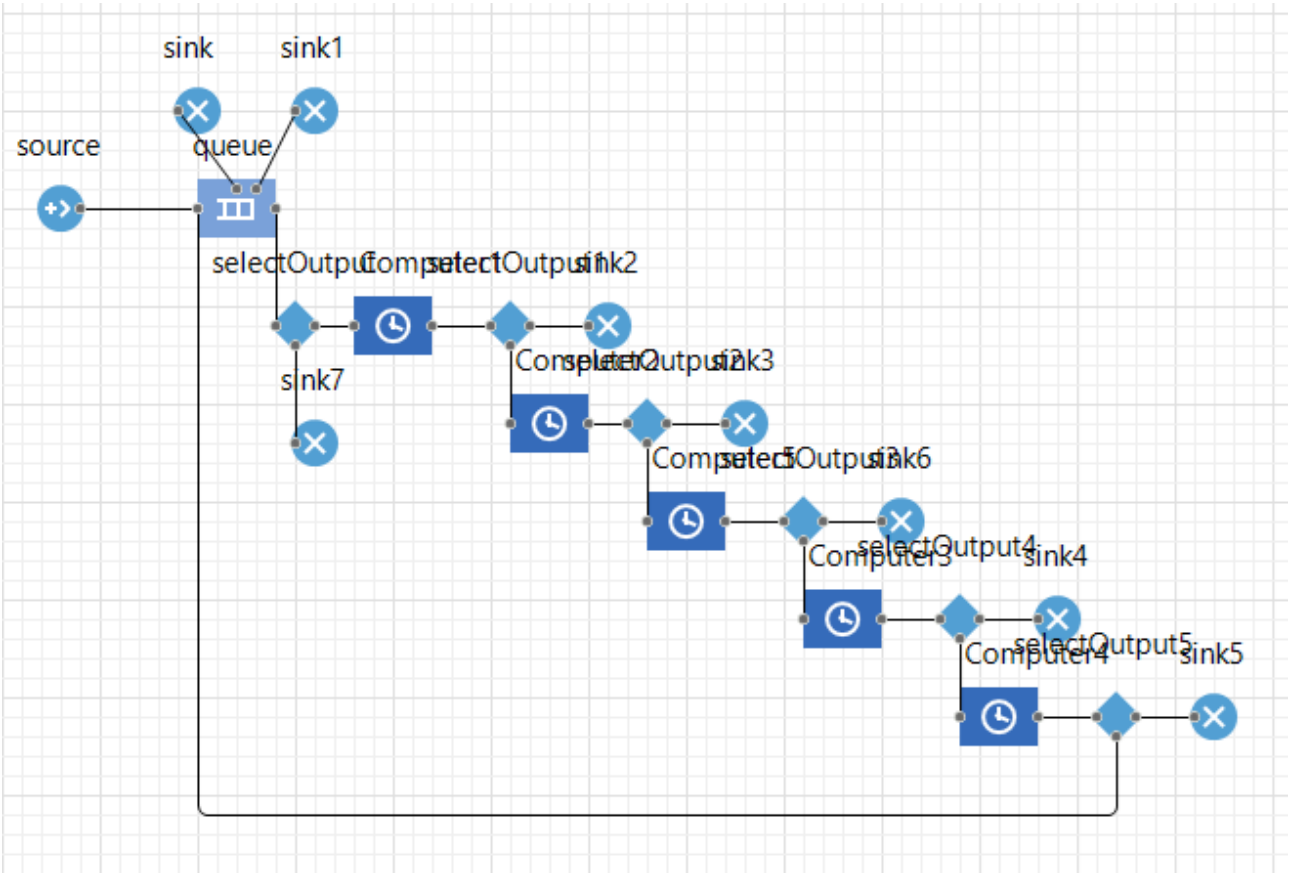

Рисунок 12. Общий вид модели Кольцо

Создание модели Звезда

Теперь можно перейти к созданию третьей модели. Данная модель будет несколько отличаться от предыдущих моделей. Центральным узлом в нашей модели будет объект SelectOutput5, так как компьютеры в нашей сети равно значимы, то в качестве параметра нашего узла необходимо использовать следующий параметр – Использовать вероятности (0,2). Вероятность рассчитывается из используемых узлов, т.е. если мы будет использовать только 3 узла, то вероятность будет (0.3)

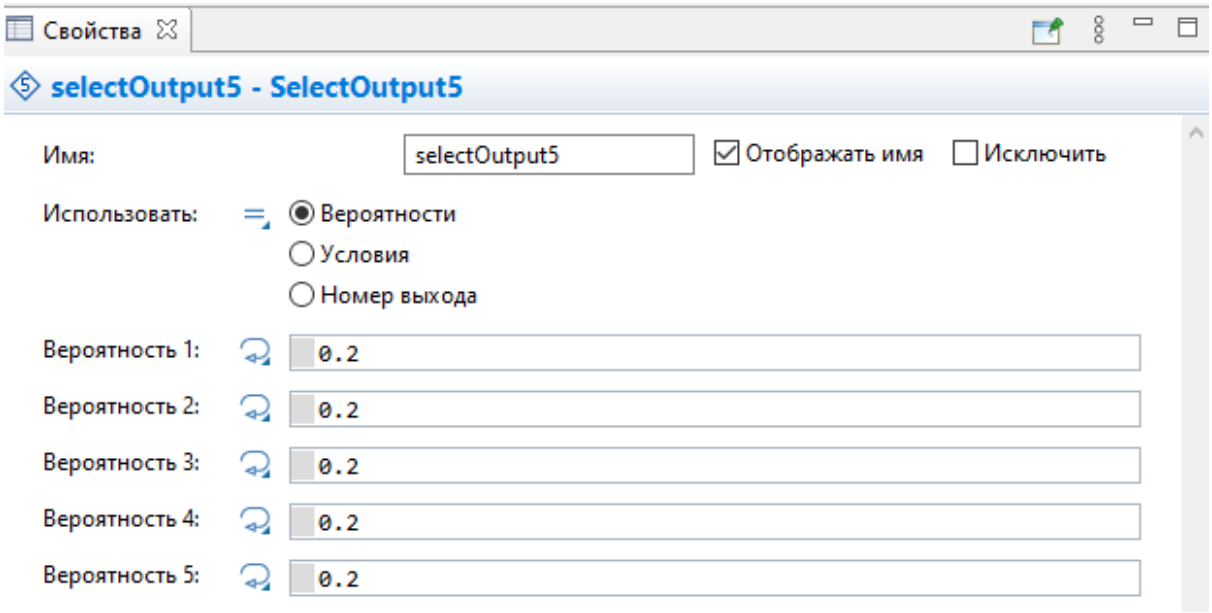

Рисунок 13. Свойста SelectOutput5

Компьютерами в разрабатываемой модели выступают объекты Delay и Sink. В свойствах для объекта Delay необходимо указать следующий параметры:

- Тип задержки (Определенное время)
- Время задержки(triangular(0.5, 1, 1.5))
- Вместимость (1)
- Вернуть агента в исходную точку (true).

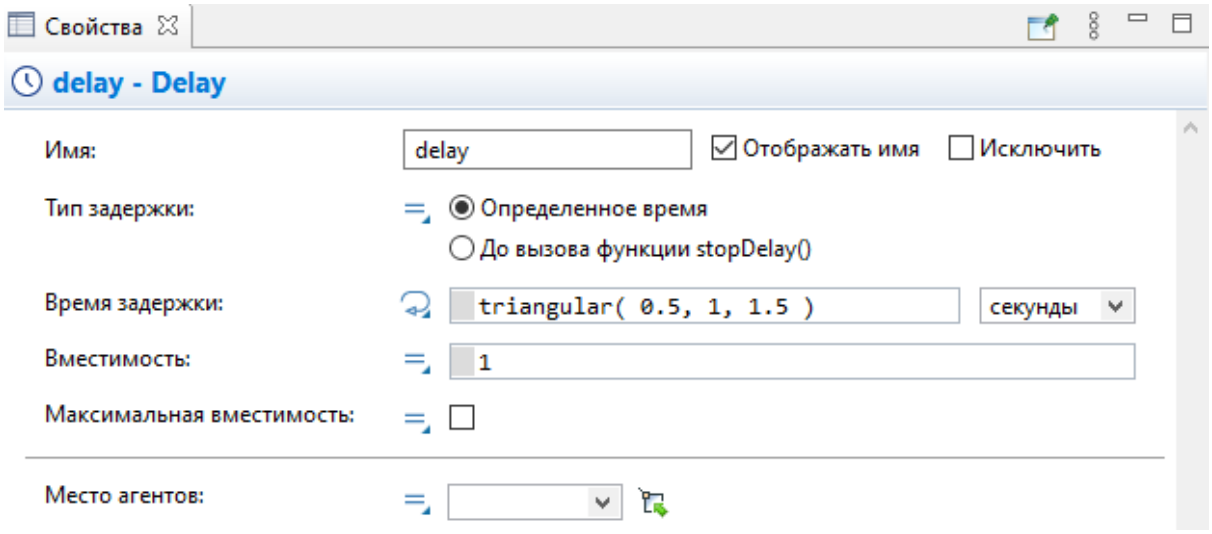

Рисунок 14. Параметры блока Delay

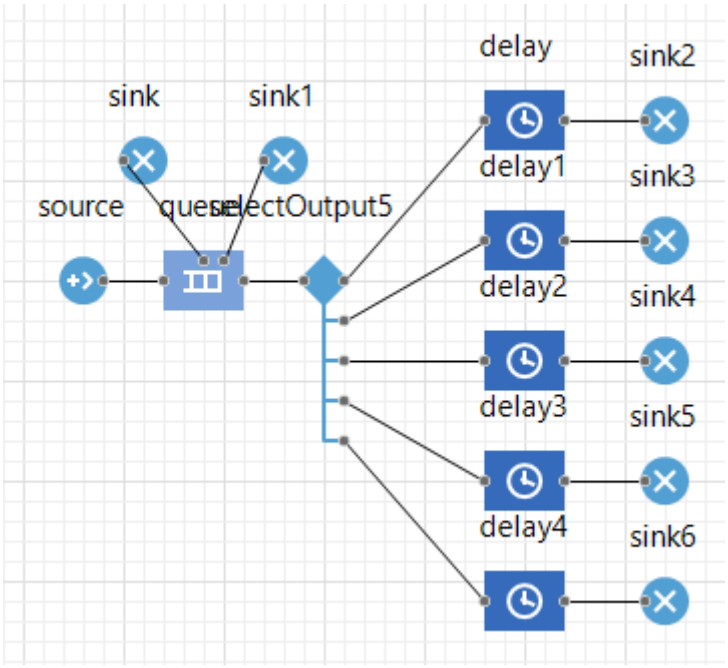

Рисунок 15. Вид модели Звезда

Создание объединенной модели «Кольцо+Шина+Звезда»

В ходе изучения курса по моделированию компьютерных сетей в AnyLogic была получена задача построить модель, которая бы объединяла функционал созданных ранее моделей сетей трех топологий: кольцо, шина и звезда.

Начальным этапом в построении модели стало создание главного сервера, из которого в процессе работы модели отправлялись все запросы. Для этого был использован объект Source.

Для функционирования сети необходимо настроить созданный объект source, задав значение параметра «Arrivals defined by» («Прибытия

согласуются») как «Interarrival time» («Время между прибытиями»), а параметр «Interarrival time» равное exponential(10/1.0)

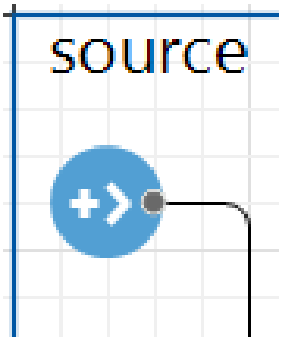

Рисунок 16. Блок Source

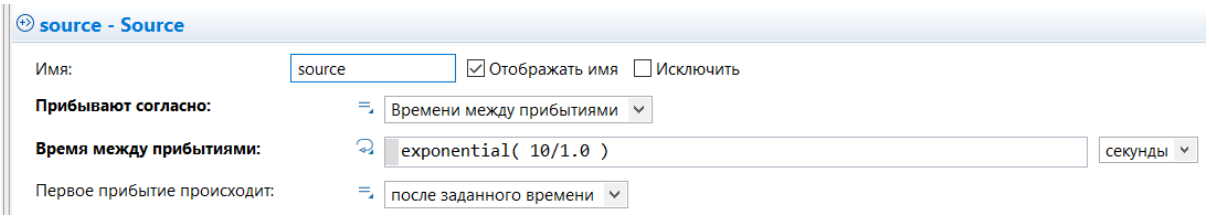

Рисунок 17. Параметры блока Source

Источник запросов (source) создан. Следующий этап - создание объекта Queue (Очередь для главного сервера). Были заданы следующие параметры объекта:

Capacity (вместимость) – 10;

Queuing (порядок обработки очереди) – FIFO («Первый зашел – первый вышел»);

Enable exit on timeout (разрешить выход по таймауту) – Включено;

Timeout – 3;

Enable preemption (разрешить вытеснение) – Включено. Это означает, что агенты помещаются в очередь в порядке приоритета и могут быть вытеснены из очереди агентами с более высоким приоритетом.

Restore agents location on exit (вернуть агента в исходное положение при выходе) – Включено;

Force statistics collection (принудительный сбор статистики) – Включен.

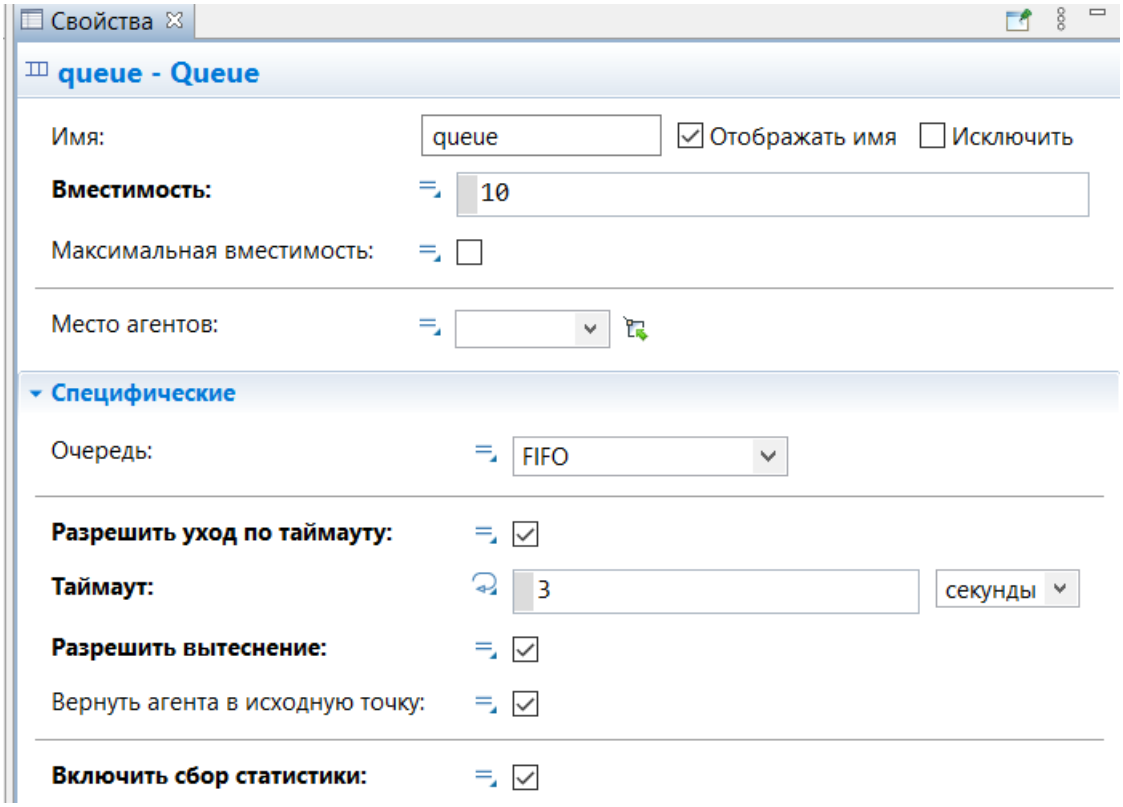

Рисунок 18. Параметры блока queue

Из того факта, что «Enable exit on timeout» и «Enable preemption» включены, следует создать 2 конечных объекта Sink, к которым очередь будет обращаться при переполнении или длительной задержке.

Для реализации трех моделей (Шина, Звезда и Кольцо) следует создать под каждую из них вспомогательный сервер. Вспомогательные сервера должны быть связаны друг с другом. Связь будет осуществляться с помощью блока SelectOutput5. Данный блок нужен для перенаправления входящего агента в один из пяти выходов (портов) при соблюдении различных условий. Как следует из его названия, максимальное число выходов – 5, но будут использоваться только 3. Распределение поступающих на вход агентов будет осуществляться на случайной основе (на основе вероятности).

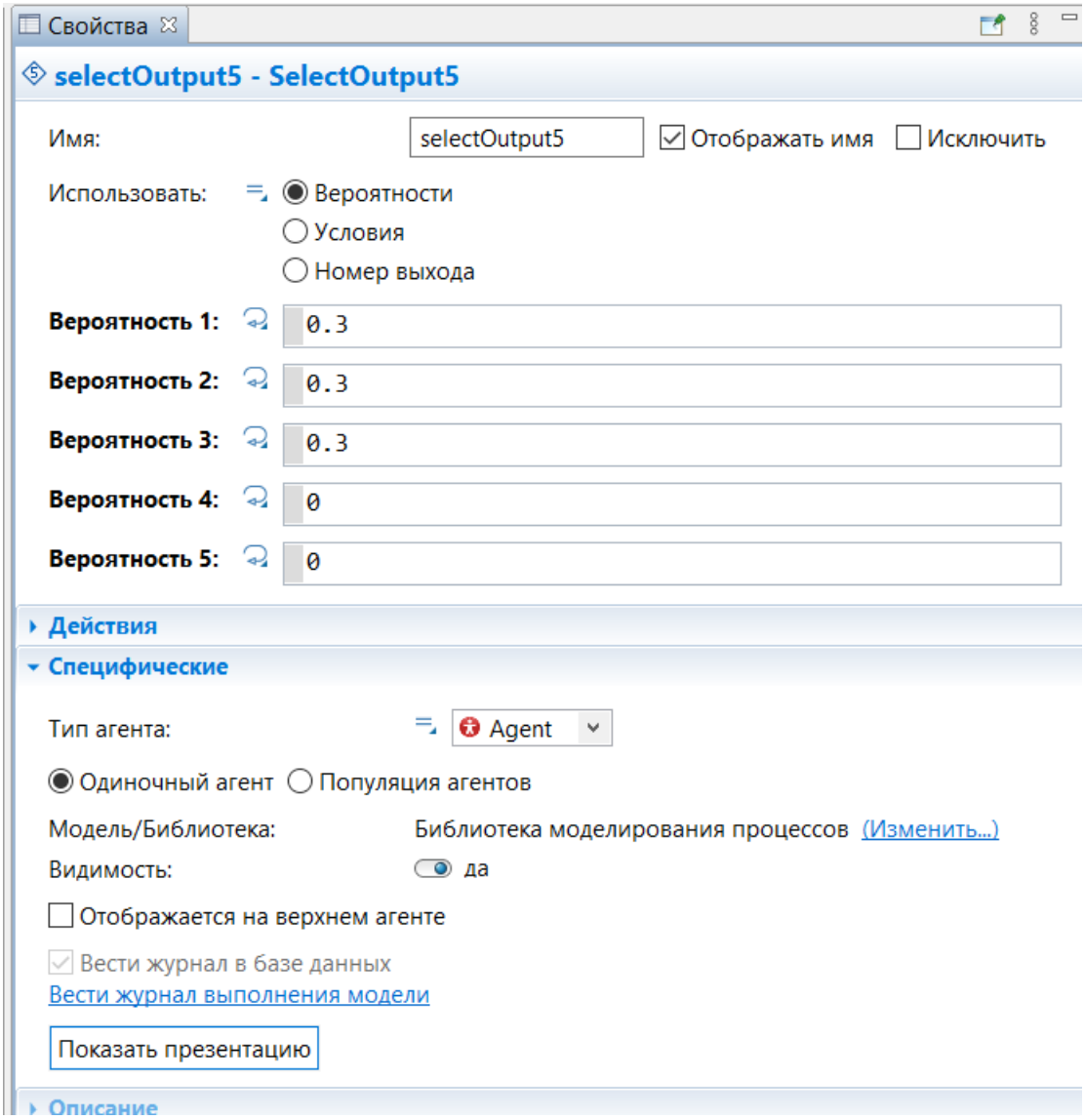

Рисунок 19. Параметры блока SelectOutput5

После добавления объекта SelectOutput5 были созданы объекты queue под каждую из моделей: BusServer, StarServer и RingServer (параметры данных объектов идентичны параметрам главного сервера). К каждому из данных серверов были присоединены по два объекта Sink.

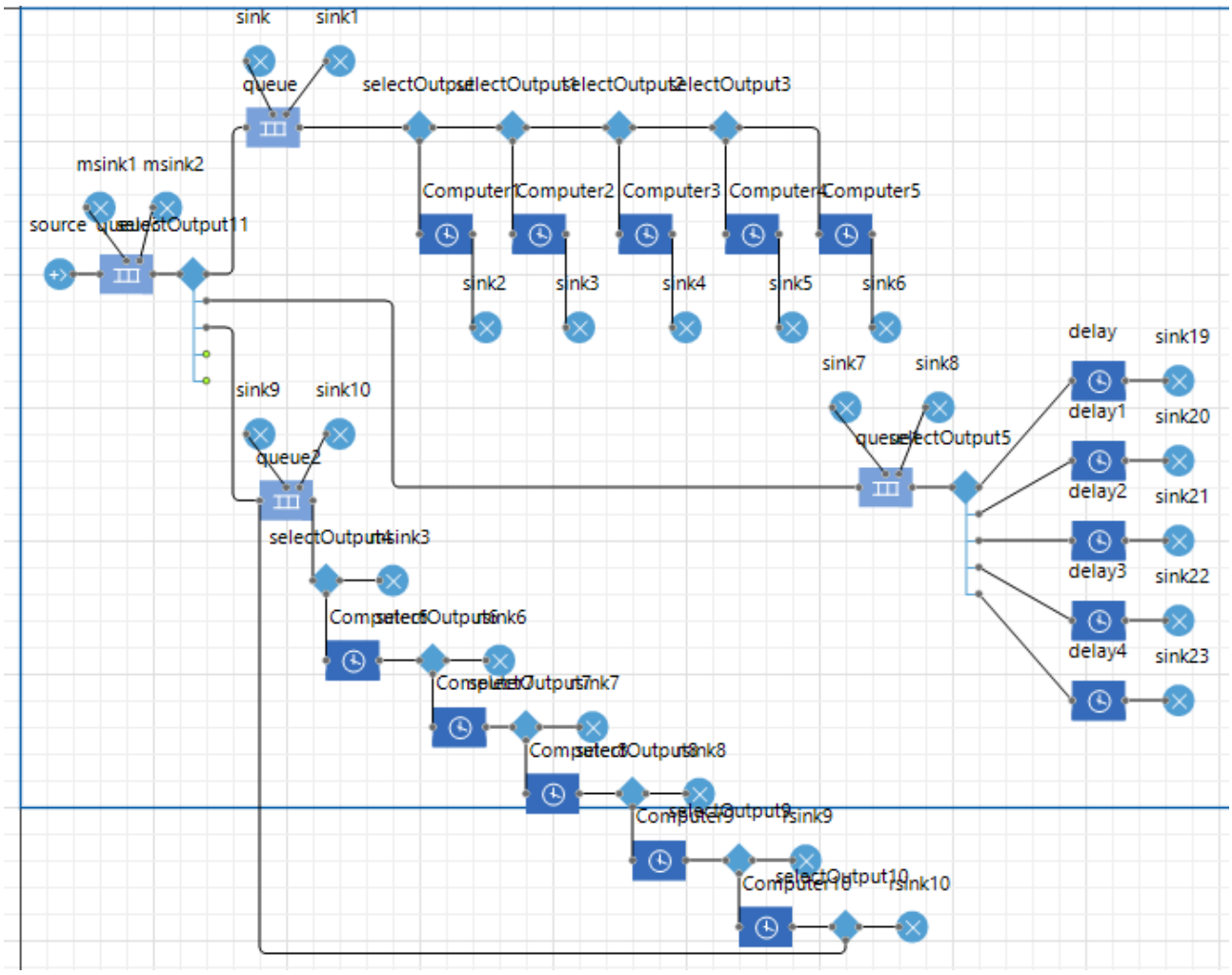

Рисунок 20. Объединенная модель

Заключение

В ходе выполнения курсовой работы были спроектированы и реализованы имитационные модели компьютерных сетей «Кольцо», «Шина», «Звезда». Также была спроектирована их объединенная модель.

Созданные модели позволяют имитировать работу серверов и компьютеров, подсоединенных к ним, осуществить передачу данных между ним, проверять нагрузку.

Выполненная курсовая работа поможет студентам в понимании моделирования и построения моделей компьютерных сетей в программном обеспечении AnyLogic.