Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра вычислительных систем

Курсовая работа по дисциплине «Моделирование»

Выполнили:

студенты гр. МГ-211 _____ Бурдуковский И.А.

Афонин А.Д.

Стояк Ю.К.

Проверил:

Профессор

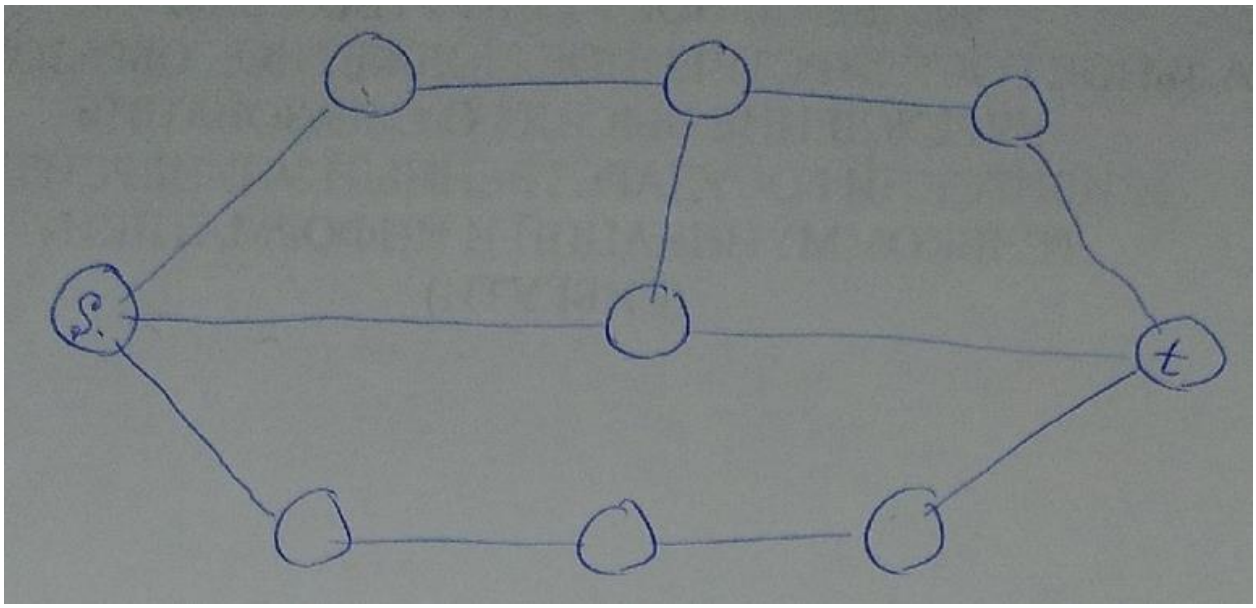
кафедры ВС _____ Родионов А.С.

Новосибирск

2023 г.

Оглавление

Формулировка задания



Необходимо составить схему из 9 связанных узлов одной сети для которой:

1. В сети присутствует интенсивность желаний передать между узлами сообщение случайной длины (в пакетах от 1 до 80).
2. Каждый узел содержит буфер на K пакетов сообщения.
3. На обработку полученного сообщения, узел тратит t времени (задержка узла).
4. В сети могут случаться потери:
    - Если сообщение достигло лимита своих переходов между узлами.
    - Если в узле не хватает места в буфере чтобы принять сообщение

Выбор направления для каждого узла при передаче сообщения - определять с помощью маршрутной матрицы.

Цели:

1. Сравнить потери между разными маршрутными матрицами.
2. Получить зависимость среднего времени доставки между узлами S и T от K.
3. Сформулировать собственную цель, провести анализ и сделать выводы.

Теоретические сведения

AnyLogic – программное обеспечение для имитационного моделирования, разработанное российской компанией The AnyLogic Company. Инструмент обладает современным графическим интерфейсом и позволяет использовать язык Java для разработки моделей. AnyLogic – первый и единственный инструмент имитационного моделирования, объединивший методы системной динамики, "процессного" дискретно-событийного и агентного моделирования в одном языке и одной среде разработки моделей. AnyLogic позволяет создавать модели с помощью трех современных подходов: дискретно-событийного, агентного и системной динамики. Эти три метода могут использоваться в любой комбинации на базе одного ПО, чтобы смоделировать бизнес-систему любой сложности. В AnyLogic есть разные визуальные языки моделирования: диаграммы процессов, диаграммы состояния, блок-схемы и диаграммы потоков и накопителей

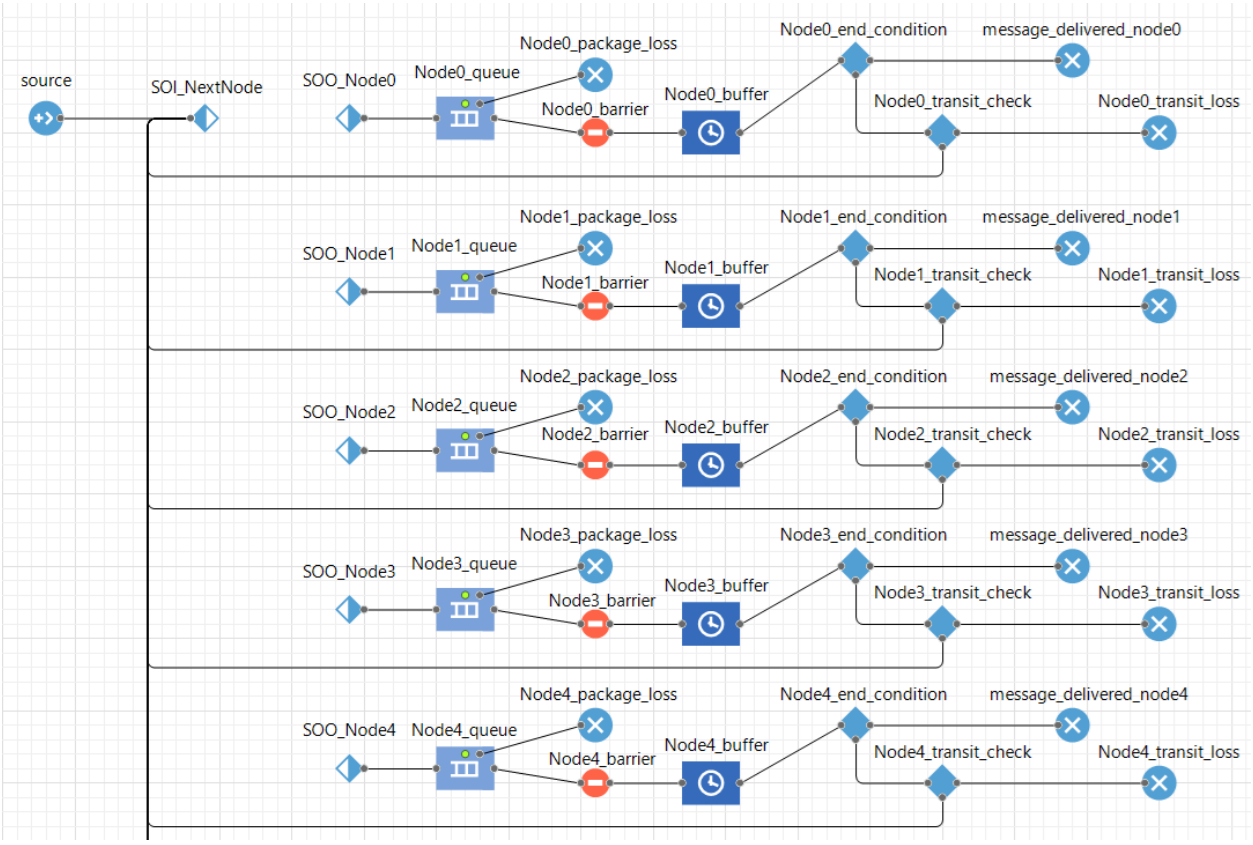
СМО - система массового обслуживания, в которой возникает массовые требования на выполнения каких-либо видов услуг, а также происходит удовлетворение этих требований – обслуживание. Главной особенностью процессов массового обслуживания является их случайность. Причины случайности заключаются в массовом характере потребностей, а также в случайности работы обслуживающей системы.

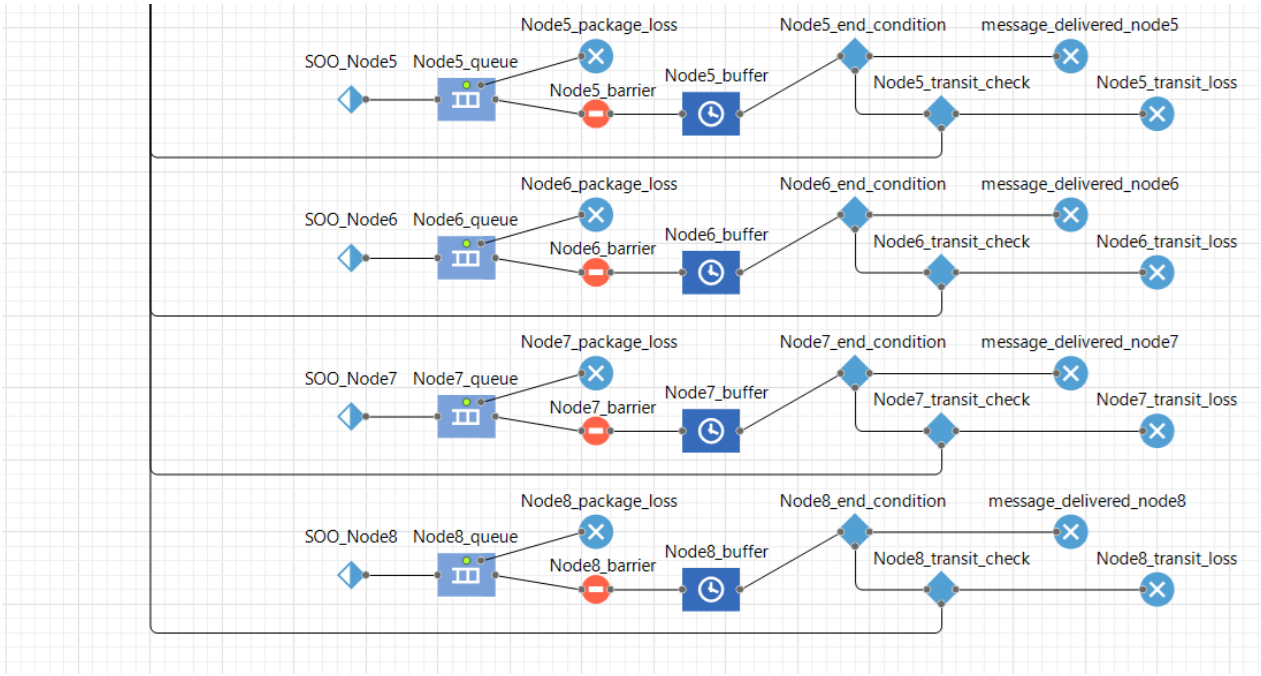
В качестве системы моделирования использовалось агентно-ориентированная система, для которой характерны следующие свойства:

- Агентный подход — это особый случай объектно ориентированного подхода;
- Поведение все системы можно рассматривать как результат поведения большого числа объектов, называемых агентами;
- Агенты являются автономными. Могут взаимодействовать друг с другом, и преследуют свои цели;

Ход работы

Согласно заданию, была реализована данная схема:



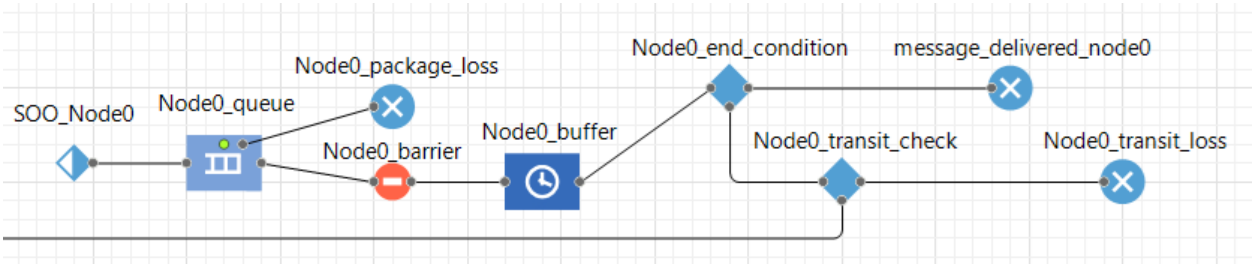


- Элемент Source используется в качестве начальной точки и создаёт агентов – сообщения со случайным количеством пакетов (от 0 до 80), случайным номером начального узла и случайным номером узла конечного назначения.

  Интенсивность прибытия - exponential(0.01)

- Элемент SelectOutputIn – SOI_NextNode используется для организации перехода сообщений по узлам.

Остальные элементы собраны в один функциональный блок, дублируемый для каждого узла в сети.



- SOO_Node – используется как переход агента к определённому узлу из элемента SOI_NextNode.
- Node_queue – очередь, обрабатывающая поток агентов на возможность попасть в узел (по вместимости буфера узла). Если агент не может сразу же покинуть очередь, то он выходит по таймауту в Node_package_loss и завершает свою работу.
- Node_barrier – блокирует выход агента из очереди, если количество его пакетов больше текущей вместимости буфера узла.
- Node_end_condition – проверяет, доставилось ли сообщение в конечный узел. Если да, то агент переходит в message_delivered_node, завершает свою работу и будет считаться завершённым. Если нет – переходит в Node_transit_check.
    - Node_transit_check – проверяет количество совершённых агентом переходов между узлами. Если сообщение достигло установленного лимита переходов, то агент завершает свою работу на выходе Node_transit_loss. Иначе агент продолжает свою работу: по маршрутной матрице выбирается следующий узел, и агент возвращается в SOI_NetxNode.

Параметры созданного класса агента Message:

- package_length – количество пакетов
- transit_count – количество совершённых агентом транзитов между узлами
- mTimeStart – время создания агента
- mTime – время жизни агента
- start_node – номер начального узла сообщения
- destination_node – номер узла-назначения сообщения
- current_node – номер текущего узла, на котором в данный момент обрабатывается сообщение
- next_node – номер следующего узла, в который будет осуществляться транзит

При запуске модели вызывается процедура fill_route_matrix, заполняющая маршрутную матрицу route_matrix.

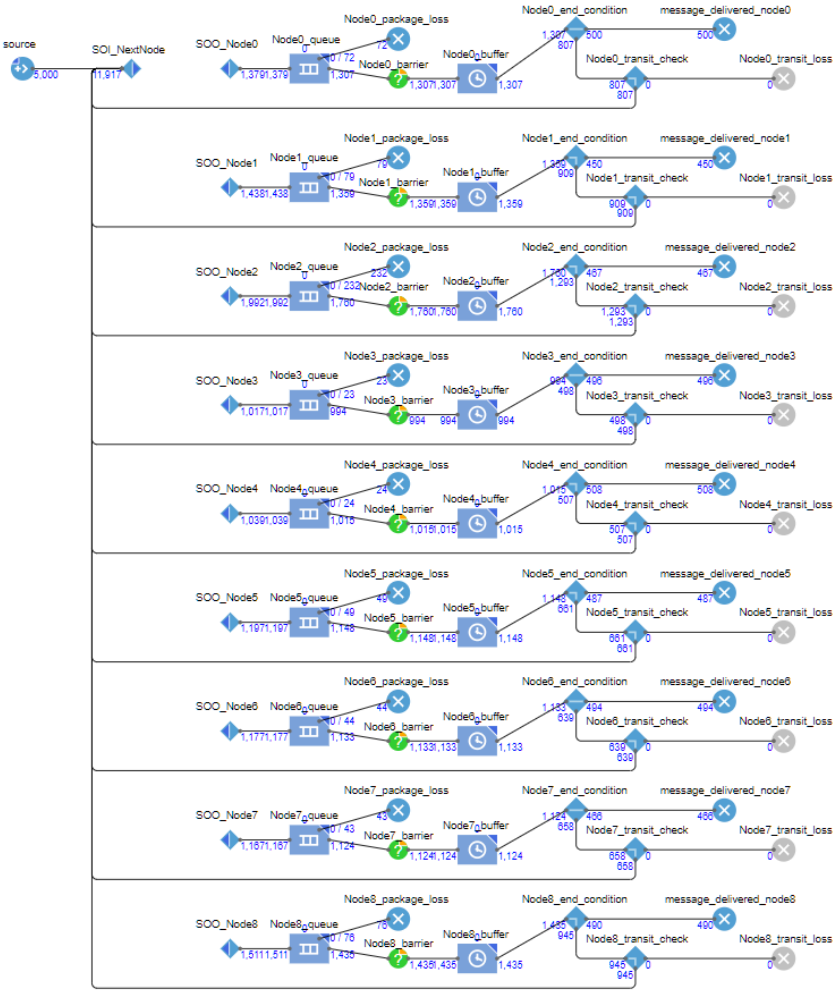
При создании агента выполняется процедура generate_message, которая заполняет параметры агента:

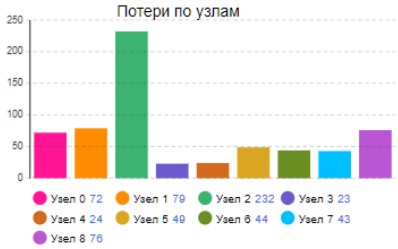
- package_length – генерируется случайное целое число от 0 до 80
- start_node – выбирается случайный номер узла от 0 до 8
- destination_node – выбирается случайный номер узла от 0 до 8, отличный от значения параметра start_node

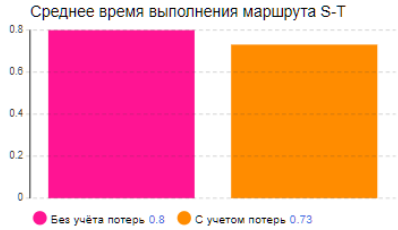
При выходе агента из источника Source в параметре mTimeStart фиксируется его время начала движения, а также выполняется процедура select_next_node, выбирающая по маршрутной матрице нужный номер следующего узла

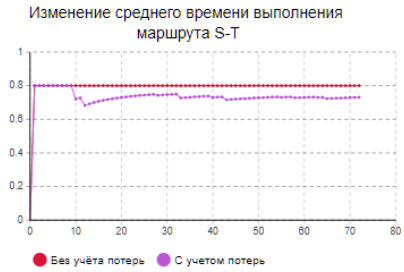
При завершении работы агента фиксируется время, из него вычитается значение mTimeStart, а результат сохраняется в параметр mTime.

Работа смоделированной схемы:









В результате запуска смоделированной схемы, сообщения успешно начали передаваться между узлами. Рассматриваемая работа модели была ограничена 5000 сообщениями.

Эксперимент 1

Сравнить потери между разными маршрутными матрицами

При работе модели было сразу выявлено, что количество потерь из-за ограничения транзитов напрямую зависит от маршрутной матрицы. При заданных связях узлов маршрутную матрицу можно организовать так, что потерь из-за ограничения числа транзитов не будет.

В рамках эксперимента были рассмотрены 3 маршрутные матрицы:

1. Матрица оптимального маршрута без транзитных потерь
2. Матрица маршрута с возможными транзитными потерями
3. Матрица маршрута с критической секцией и транзитными потерями

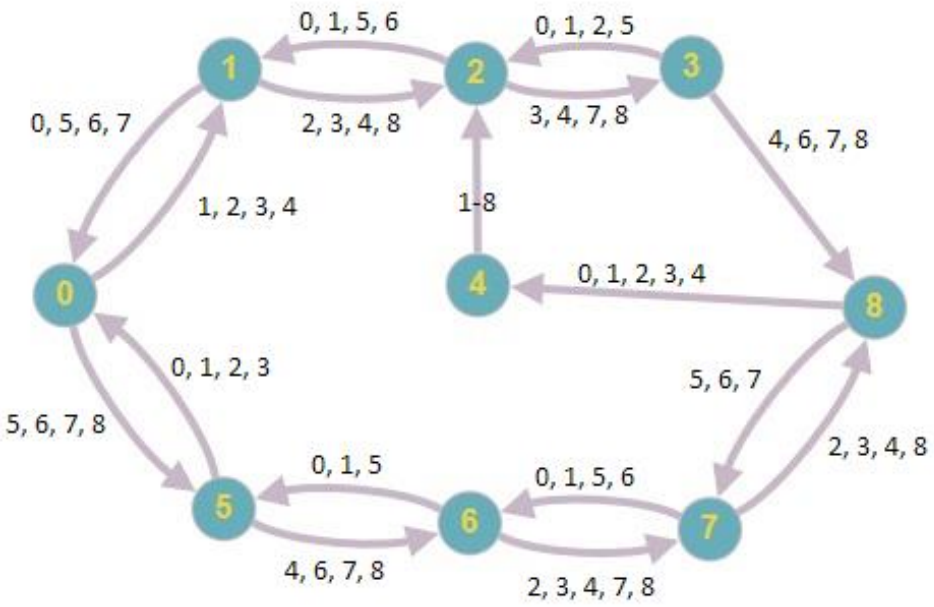


Рисунок 1.1. Маршрутная матрица без транзитных потерь

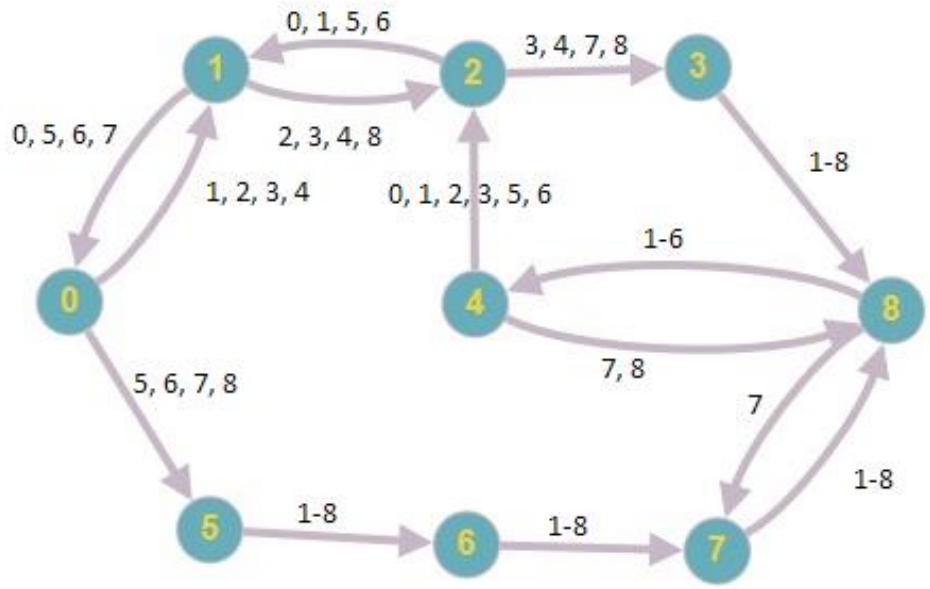


Рисунок 1.2. Маршрутная матрица с транзитными потерями

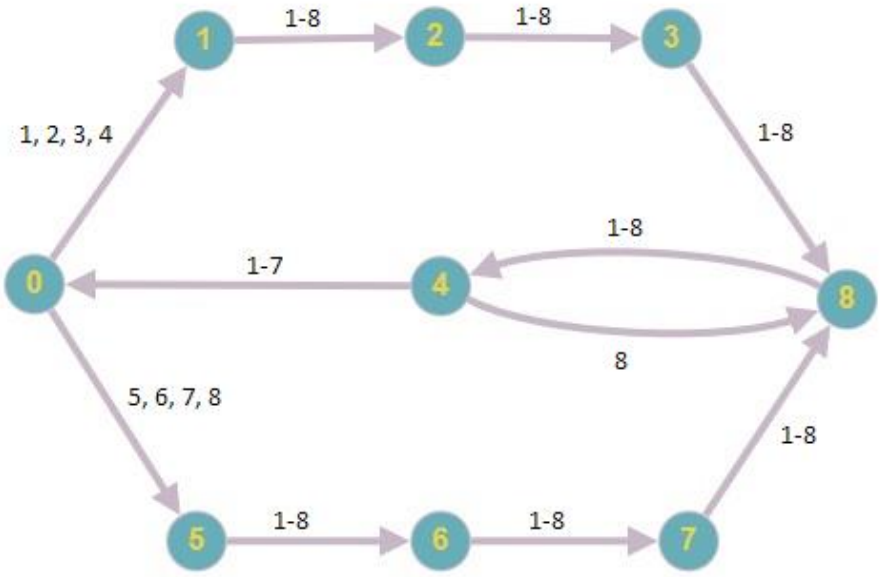


Рисунок 1.3. Маршрутная матрица с критической секцией и транзитными потерями

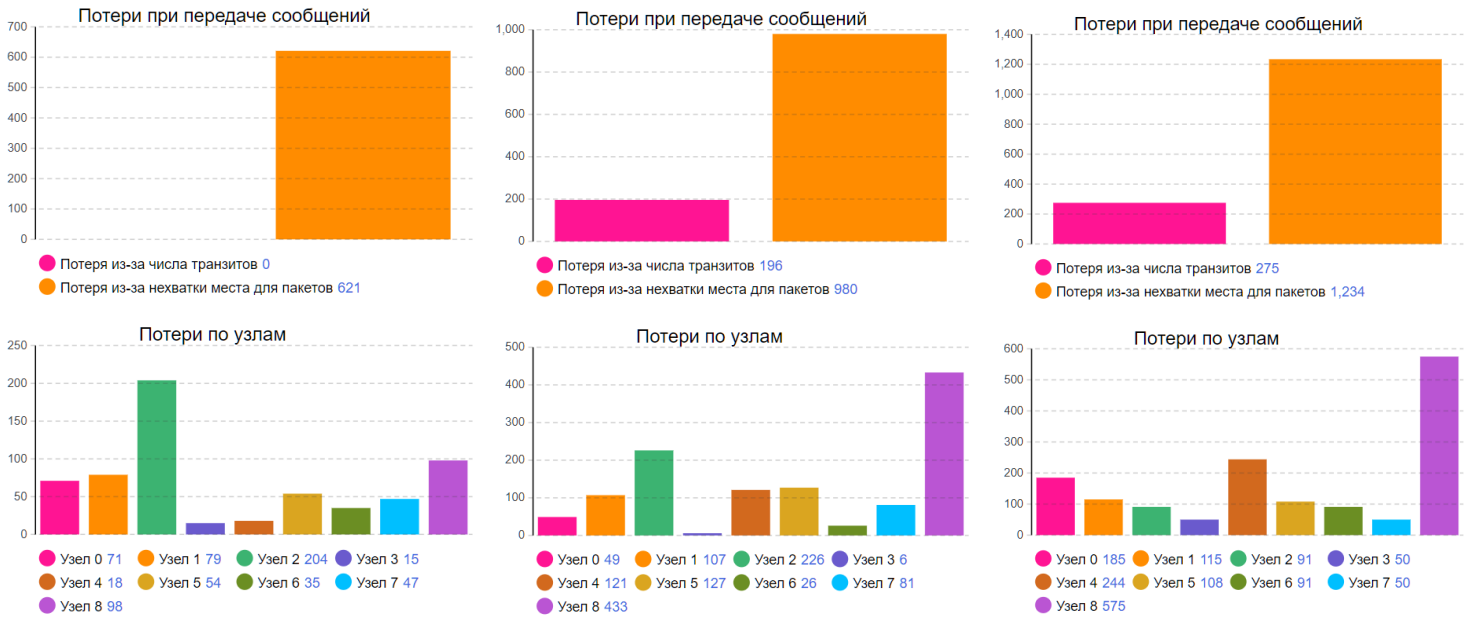


Рисунок 1.4. Полученные диаграммы потерь по результатам работы модели

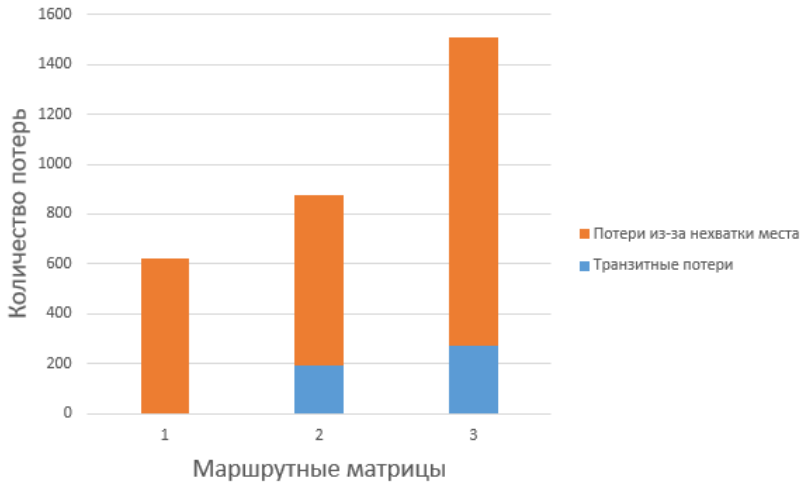


Рисунок 1.5. Количество потерь для маршрутных матриц по видам потерь

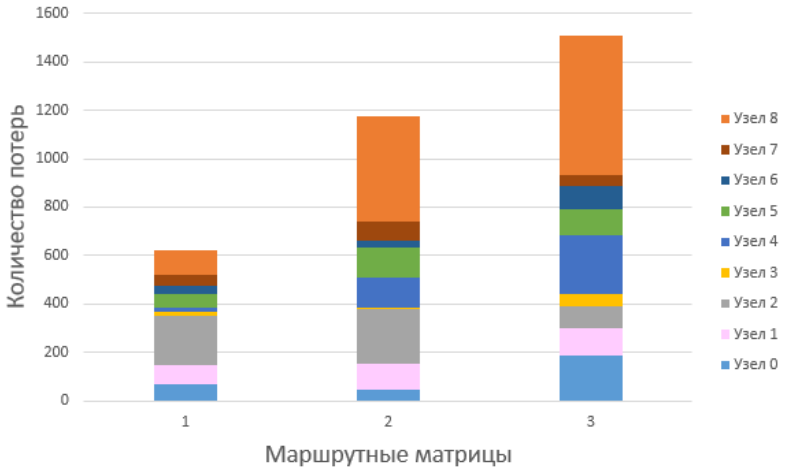


Рисунок 1.6. Количество потерь для маршрутных матриц по узлам

Данный эксперимент наглядно демонстрирует, что способ организации маршрутов между узлами напрямую влияет на количество потерь в системе. Число транзитных потерь можно охарактеризовать маршрутной матрицей. Число потерь из-за переполнения буфера узла зависит от нескольких факторов:

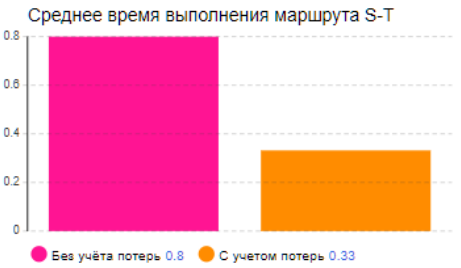
1. Распределение маршрутов между узлами
2. Размер буфера и входящих сообщений
3. Скорость обработки на узле

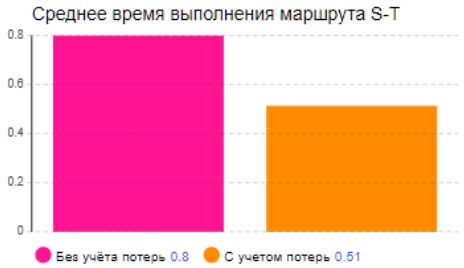
Эксперимент 2

Получить зависимость среднего времени доставки между узлами S и T от K (размера буфера).

В эксперименте использовалась маршрутная матрица №1. Так как в модели маршрутная матрица задаётся статически, то среднее время успешной доставки сообщения от узла S до узла T будет одинаковым для любого размера буфера.

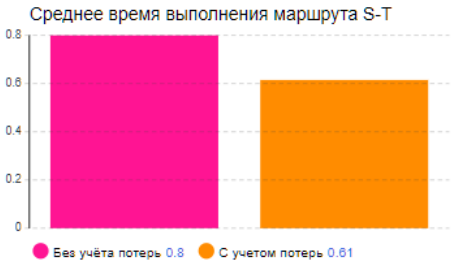
Было совершено несколько тестовых запусков, на графиках ниже отображено среднее время выполнения маршрута S-T с разными размерами буфера на узлах.

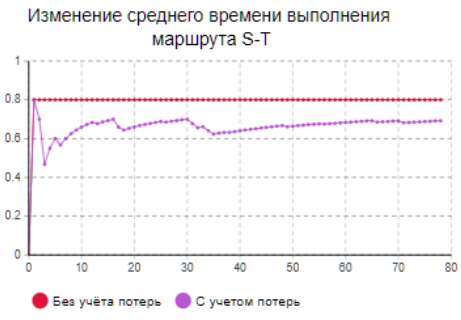




Размер буфера = 50

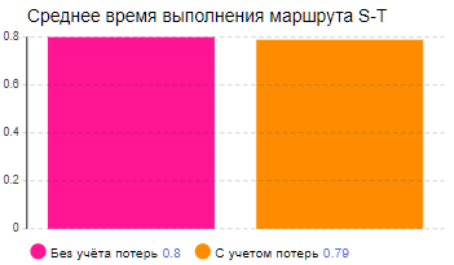
Размер буфера = 100

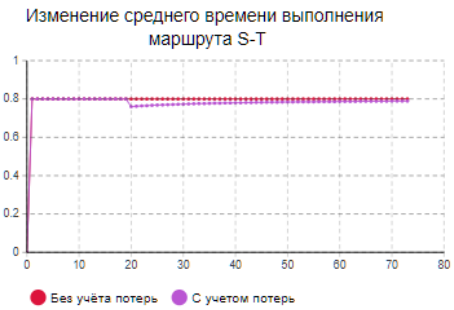




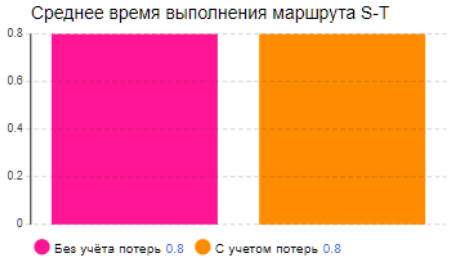
Размер буфера = 150

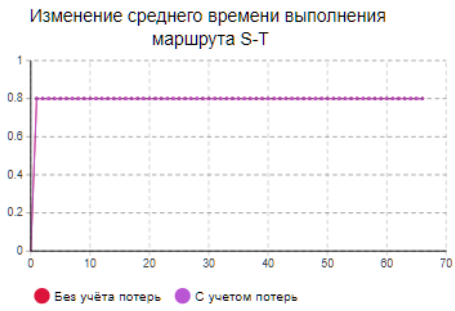
Размер буфера = 200





Размер буфера = 300





Размер буфера = 400

Эксперимент 3

Сформулировать собственную цель, провести анализ и сделать выводы.

Бизнес цель:

Определить во что эффективнее вложить ресурсы:

1. Улучшение программной реализации: разработать новый протокол сжатия, уменьшающий усреднённый размер сообщений на M%.
2. Улучшение инфраструктуры: сократить на N% время задержки на вычислительных узлах.

Для рассматриваемой модели использовать маршрутную матрицу №1.

Пример:

1. При генерации размеров сообщений перейти от равномерного распределения к нормальному.
2. Уменьшить время задержки на узлах на 10%.

Какое решение будет эффективнее?

Произведённые замеры:

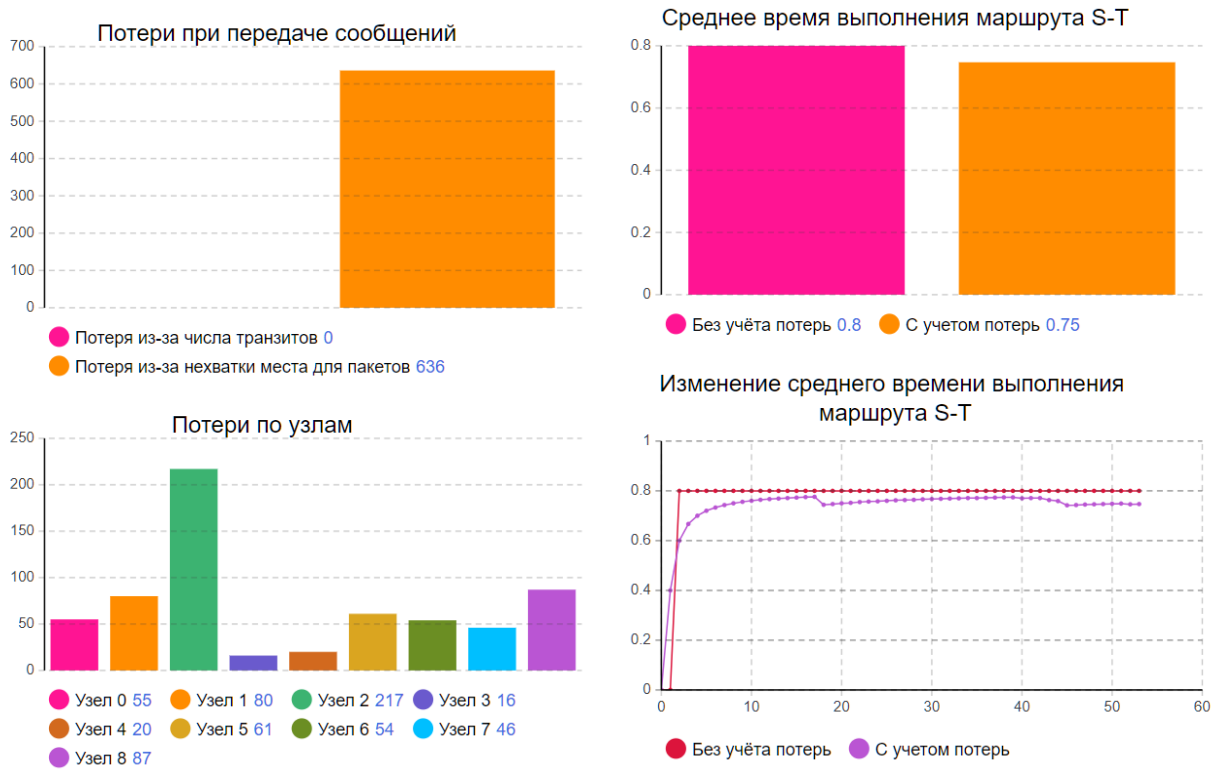


Рисунок 3.1. Обычное состояние

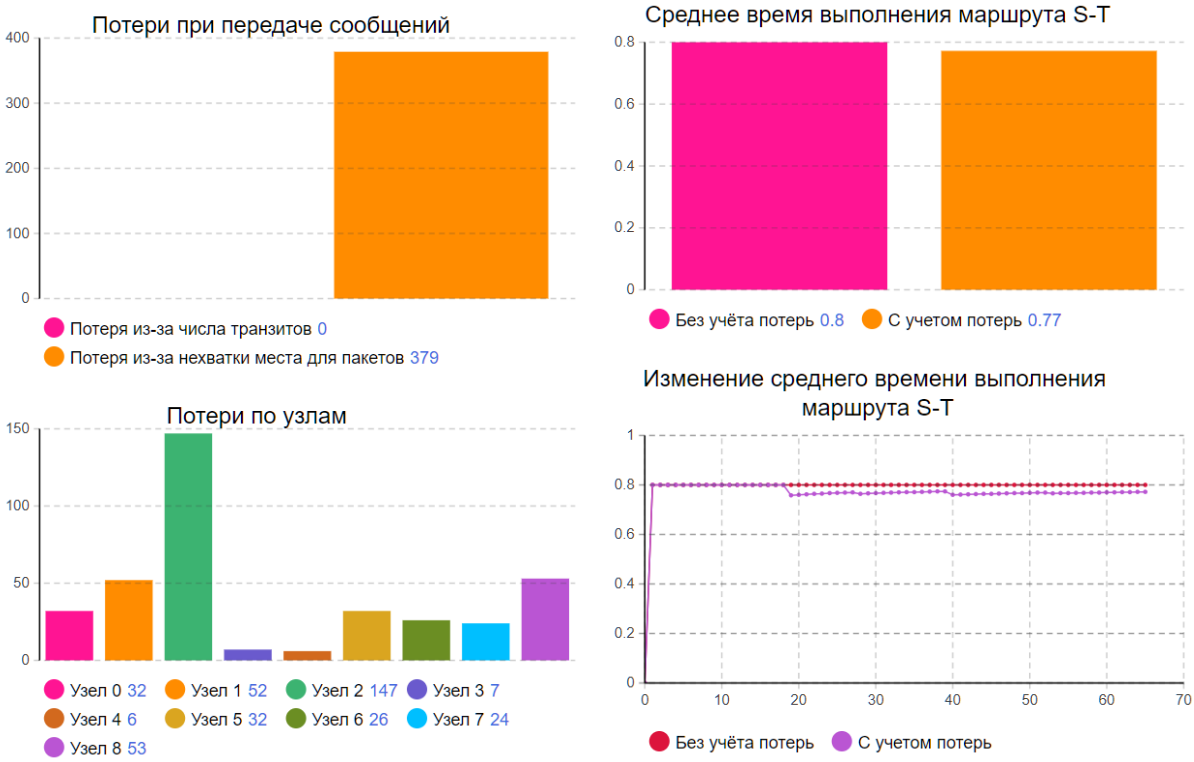


Рисунок 3.2. Нормальное распределение размера сообщения

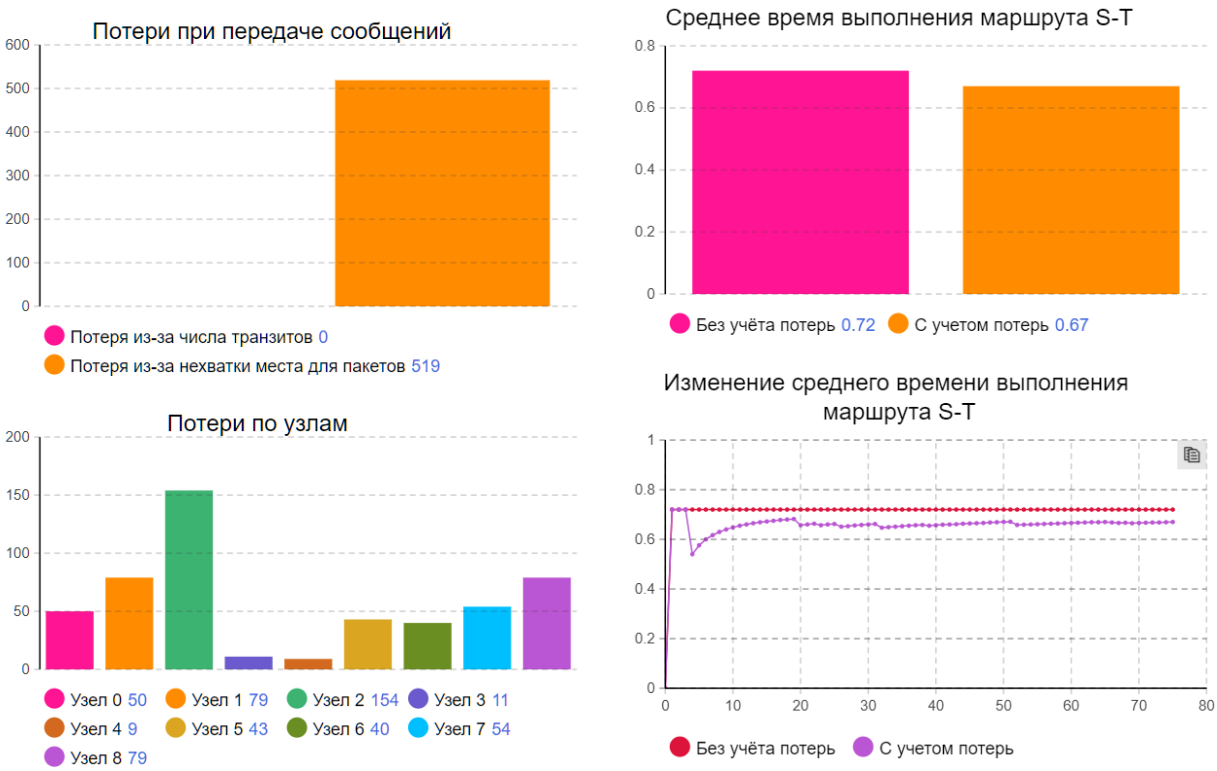


Рисунок 3.3. Сокращённое на 10% время задержки

Вывод:

Работать над протоколом, чтобы заменить равномерное распределение на нормальное - не выгодно. Мы способны уменьшим шум в системе, но не сможем выиграть в производительности. В то же время - увеличить производительность вычислительных узлов - надёжная и выгодная идея. Результат таких улучшений будет предсказуемым и эффективным.