Мини-статья

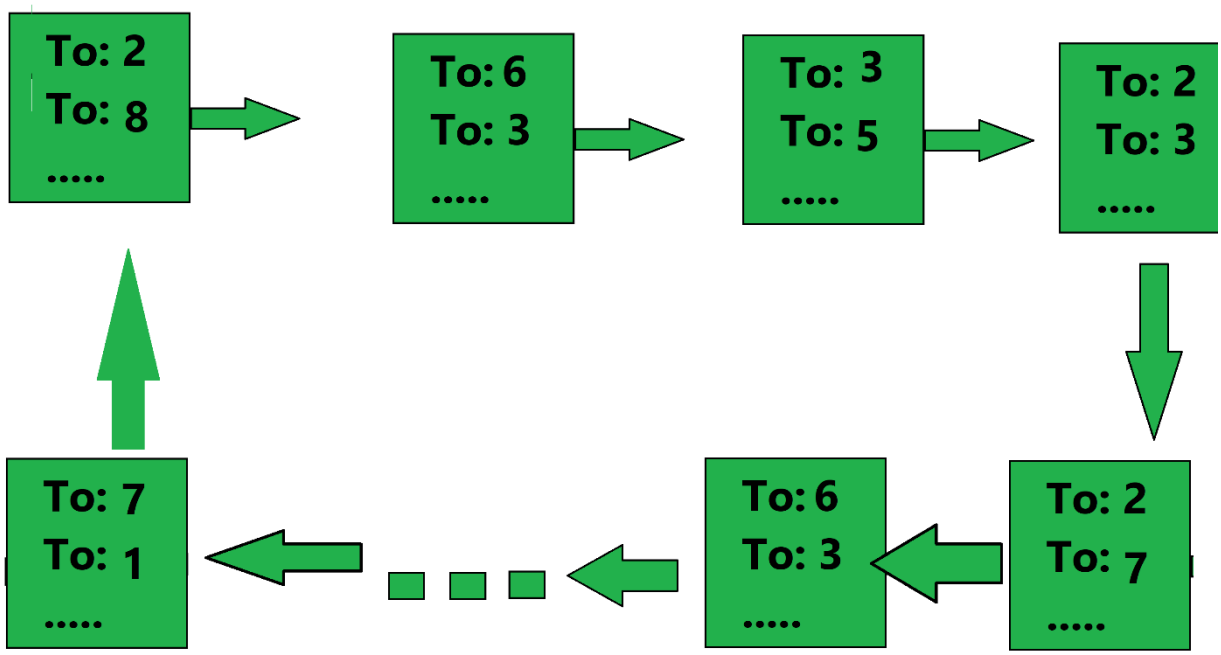
Задача

Исследовать пропускную способность сети (throughput) и характерное время задержки (latency) в зависимости от количества узлов N и количества пакетов P (1...N), находящихся в транзите одновременно.
Попытаться оптимизировать (улучшить) throughput или latency как в целом так и для отдельно взятых конкретных режимов (недогруженная сеть, перегруженная сеть) и исследовать влияние оптимизаций для одного режима на весь спектр режимов.

Throughput сети – количество пакетов, которое прошло за секунду.
Latency сети – время прохождения одного пакета

Исследование

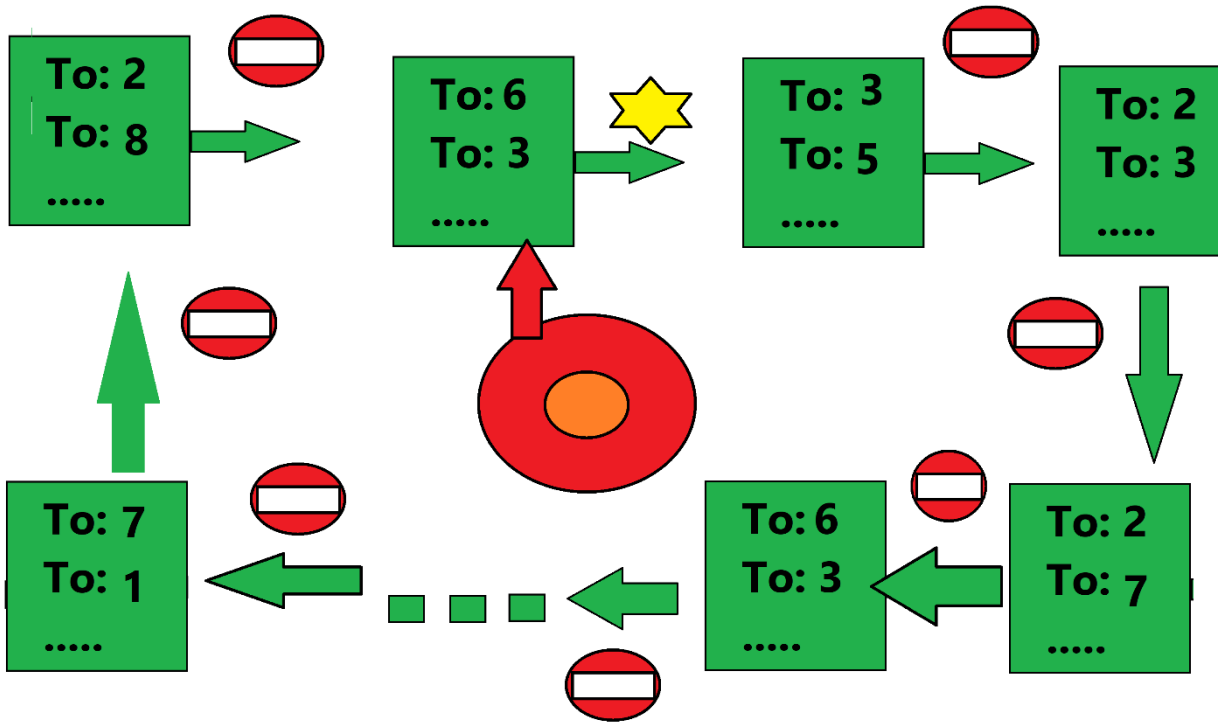
Для исследования throughput и latency для пакетов находящихся одновременно в транзите, был использован такой механизм. Каждый узел 1..N нагружаем M пакетами, выбирая с помощью равномерного распределения для каждого пакета, до какого узла его нужно переслать. После чего запускаем работу все узлы. И того у нас в транзите: N x M пакетов в транзите одновременно. Замеры проводились с помощью библиотеки JMH.



Исследование обычной реализации:

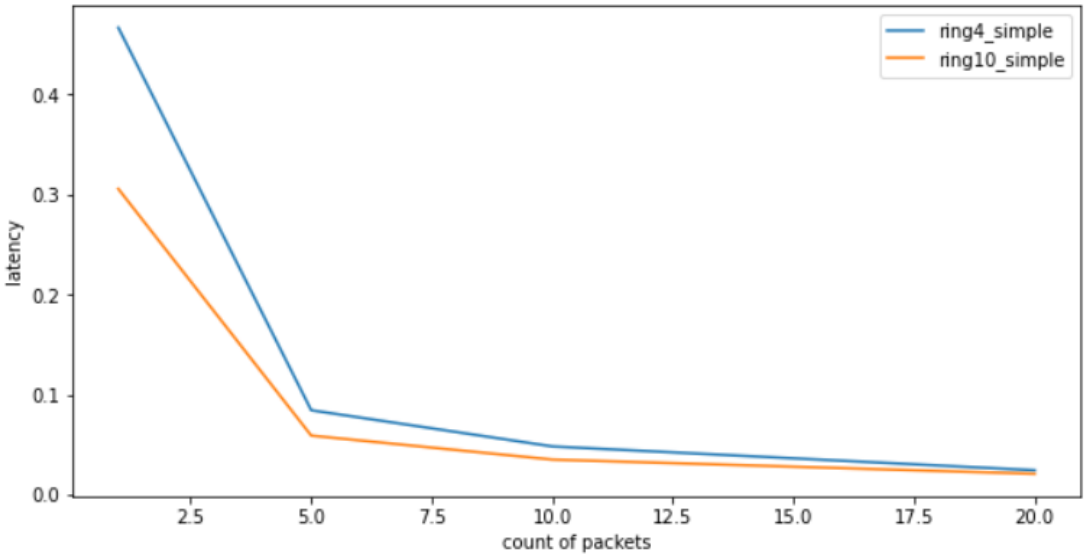
Обычная реализация предполагала, что у нас есть общий флаг, на котором мы синхронизируемся. Только один узел может перемещать информацию по сети в заданный промежуток времени.

Схема:

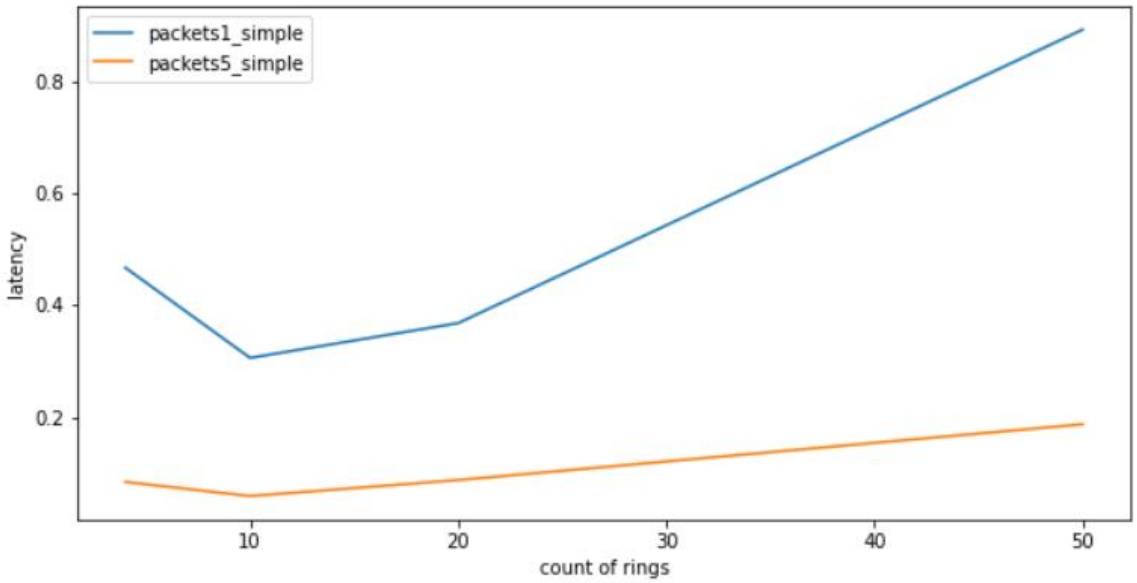


Графики:

Latency от количества пакетов, на сетях с 4 и 10 узлами

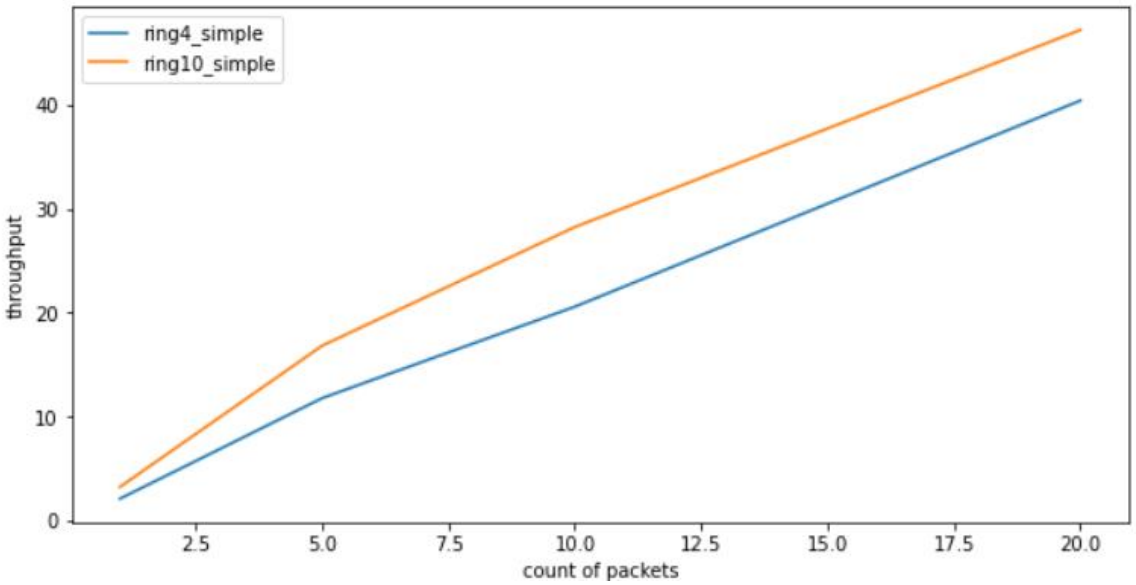


Latency от количества узлов, на сетях с 1 и с 5 пакетами на узел

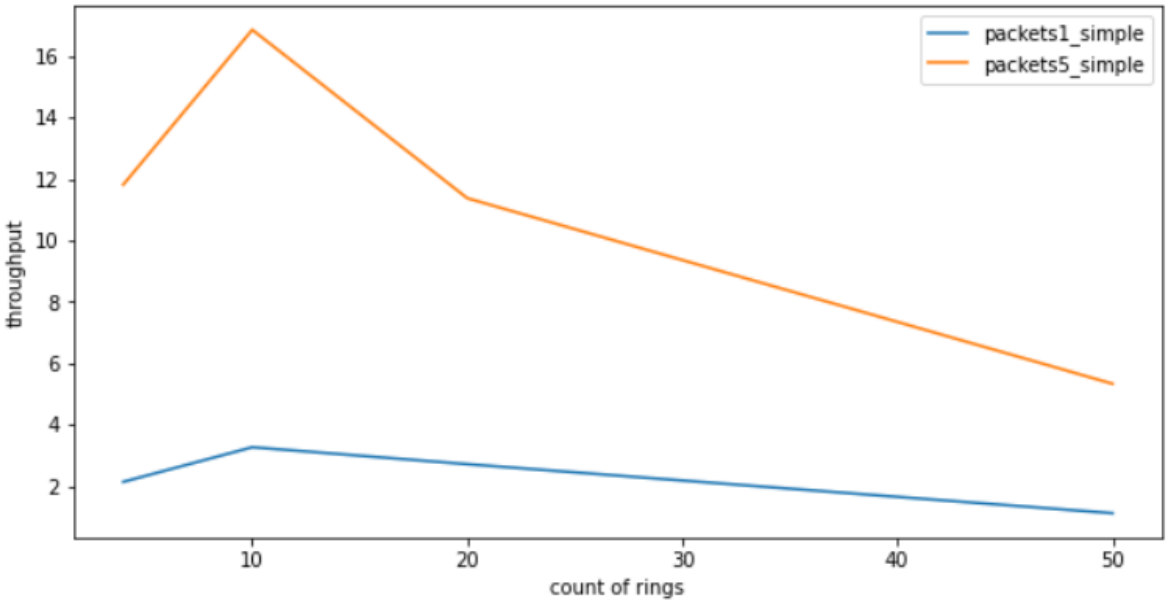


Количество узлов замедляют проход пакетов по сети.

Throughput от количества пакетов, на сетях с 4 и 10 узлами



Throughput от количества узлов, на сетях с 1 и с 5 пакетами на узел



Количество узлов уменьшают также и throughput сети.

Исследования показали, что у такой сети достаточно низкий качество latency и throughput во всех режимах: недогруженная сеть, стандартная работа, перегруженная сеть.

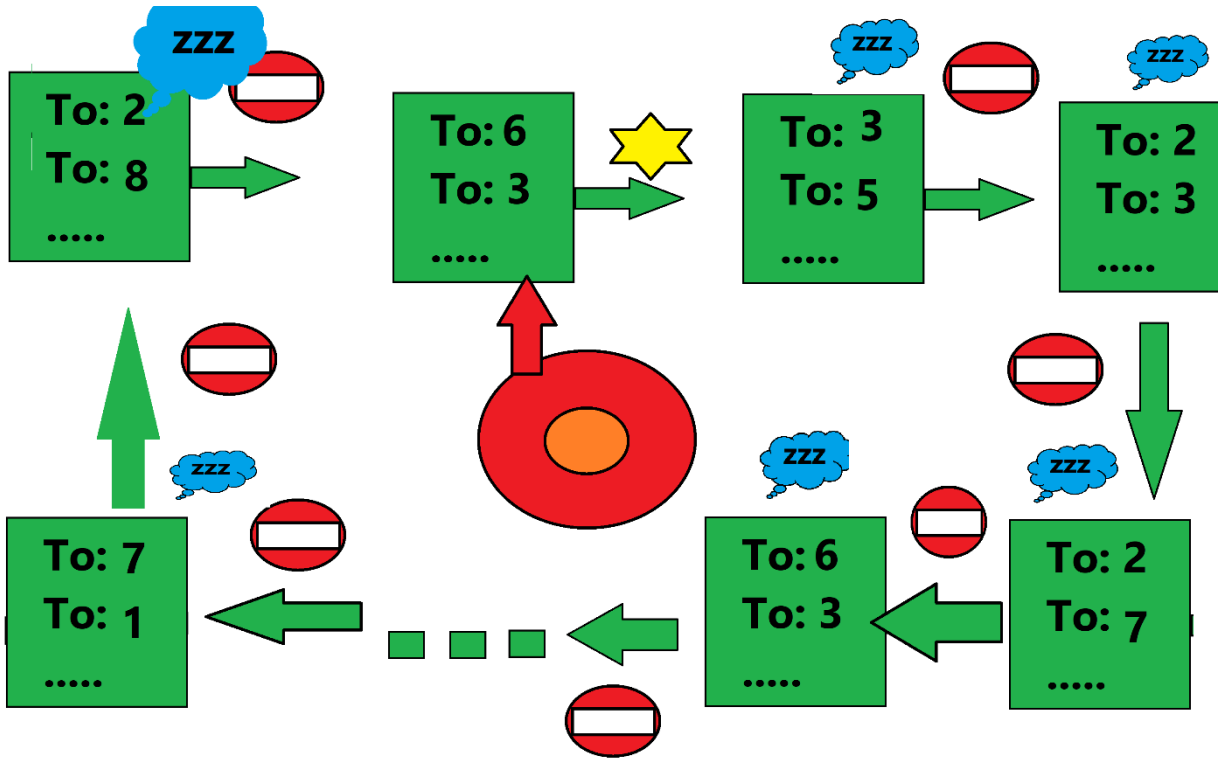
В чем проблема?

Все узлы крутятся на одной точке синхронизации, в отдельно взятый промежуток времени либо только один поток выполняет полезную работу, либо не выполняет никто. Все потоки утилизируют ЦПУ, пытаясь захватить лок.

Оптимизация 1

Пусть теперь каждый узел ждёт, пока не появится новых задач. Не будем утилизировать цпу, добавим wait. Таким образом попытаемся оптимизировать режим недогруженной сети и стандартный режим.

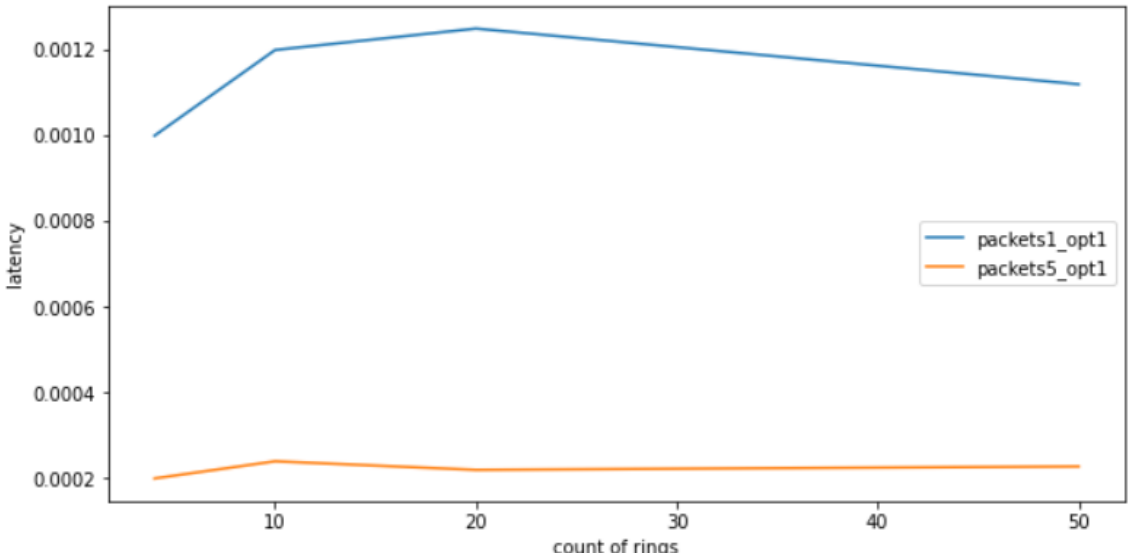
Схема:

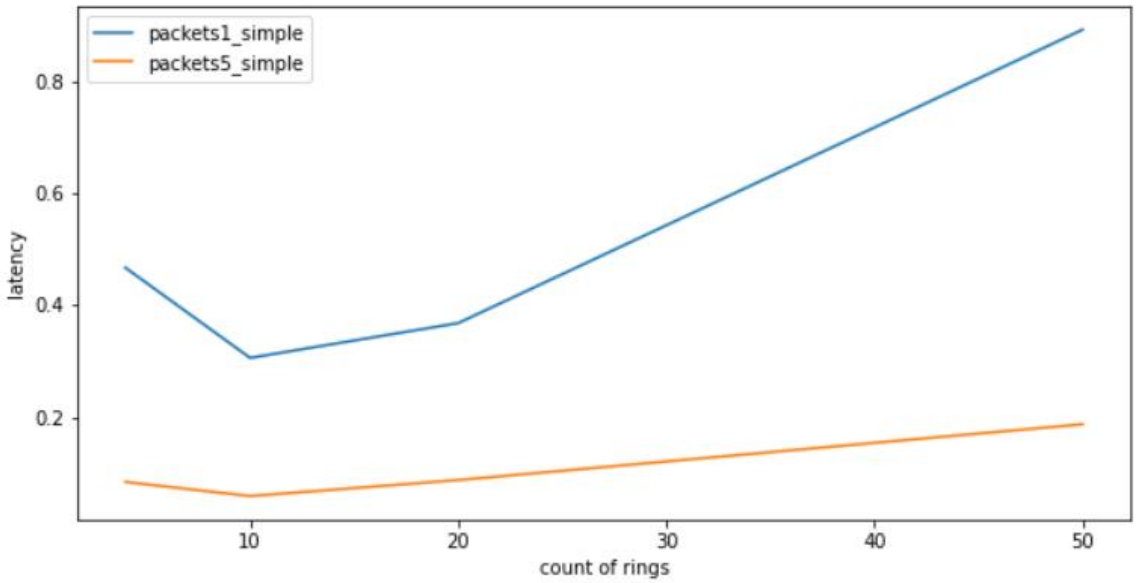


Графики:

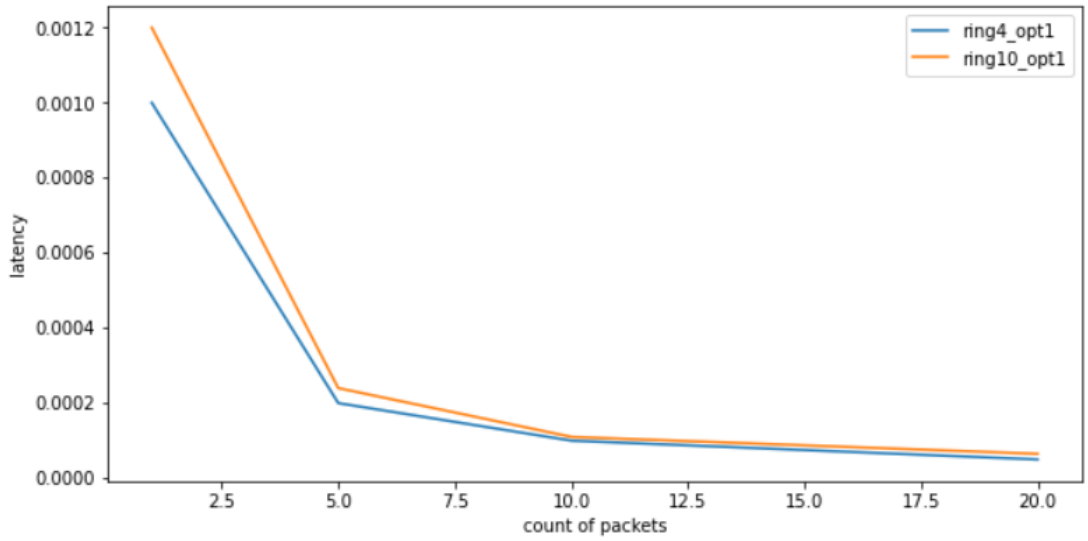
Графики сильно отличаются от предыдущих результатов

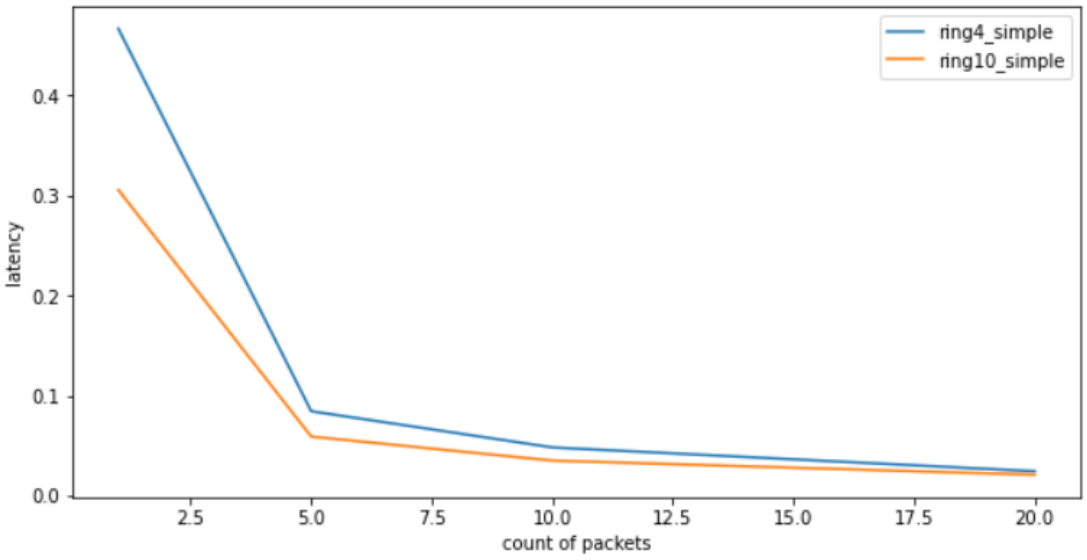
Latency от количества узлов, на сетях с 1 и с 5 пакетами на узел



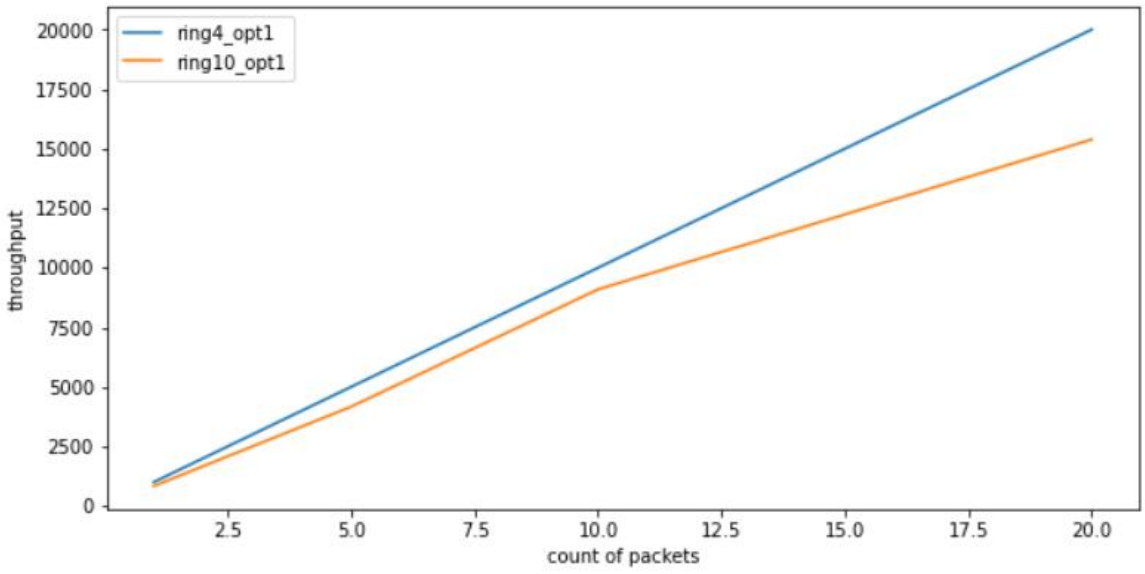


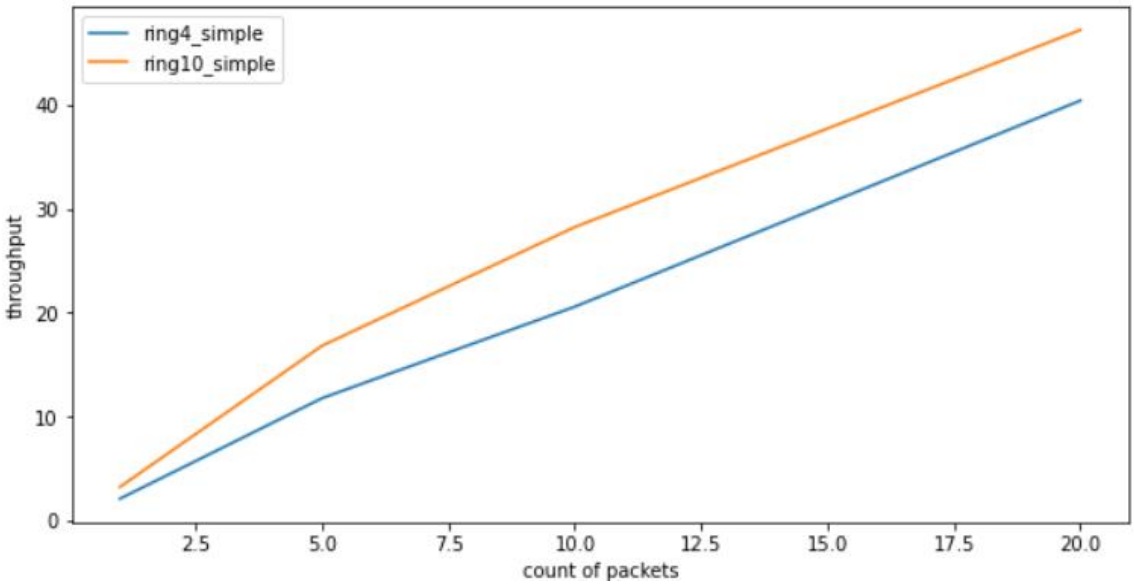
Latency от количества пакетов, на сетях с 4 и 10 узлами



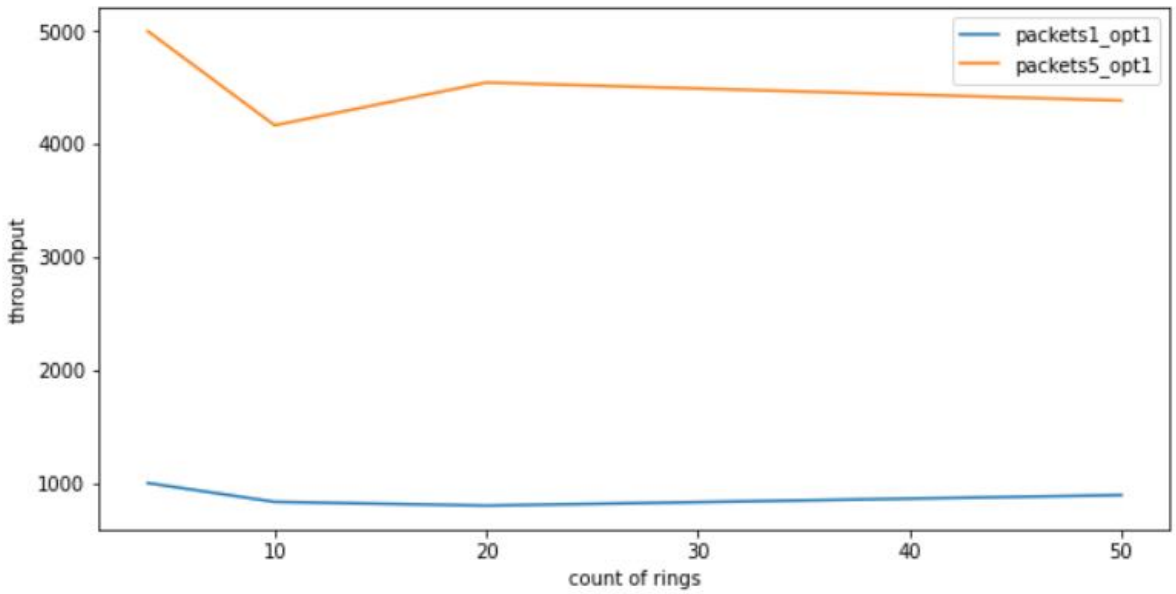


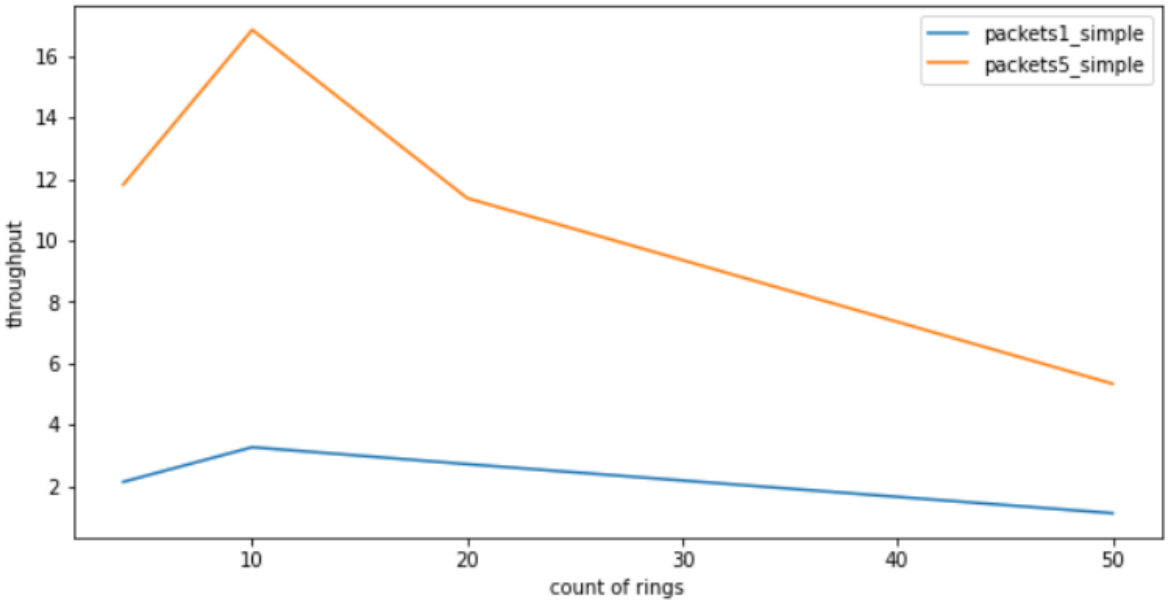
Throughput от количества пакетов, на сетях с 4 и 10 узлами





Throughput от количества узлов, на сетях с 1 и с 5 пакетами на узел





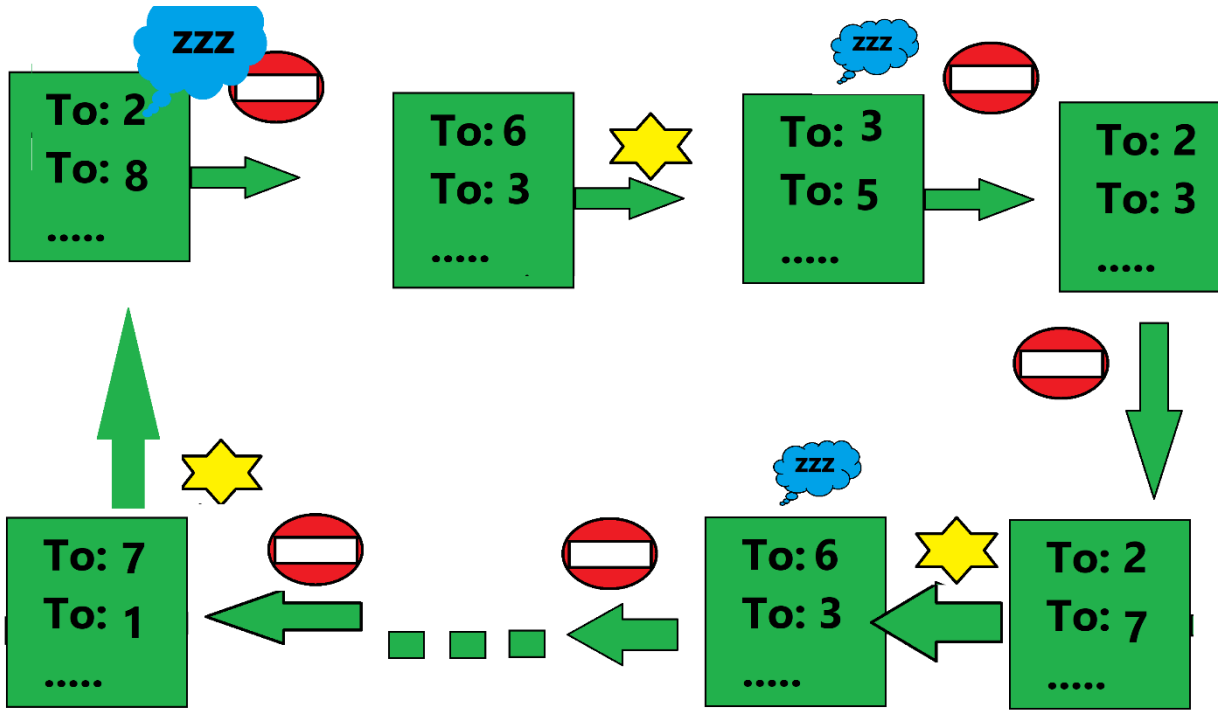
Исследования выявили ещё ряд недостатков.

Оптимизация 2

Теперь пусть каждый узел имеет и видит только 2 лока, ведь только ими он пользуется. Теперь все потоки могут работать в произвольном режиме, нет общей точки синхронизации. У нас есть свой лок, который мы захватываем, чтобы вытащить из очереди элемент или делаем wait на нем, чтобы ждать появления нового пакета. Также есть лок соседнего по часовой стрелке узла. Мы его захватываем, чтобы засунуть в него пакет, после чего делаем на нем notify.

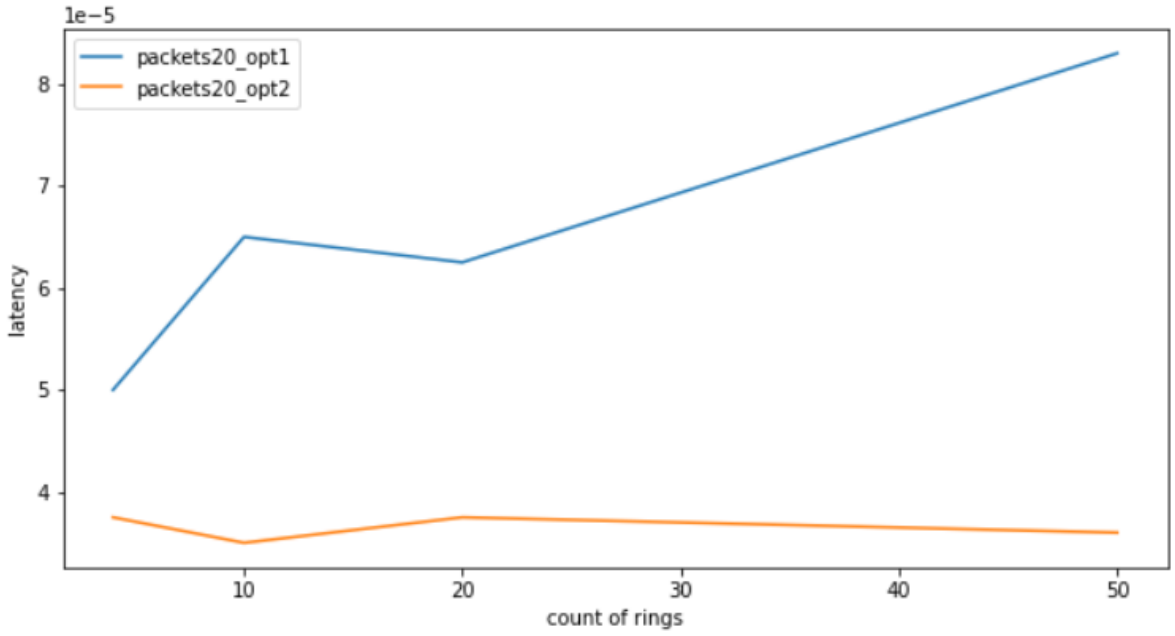
Таким образом хотелось повысить качество всех режимов.

Схема:

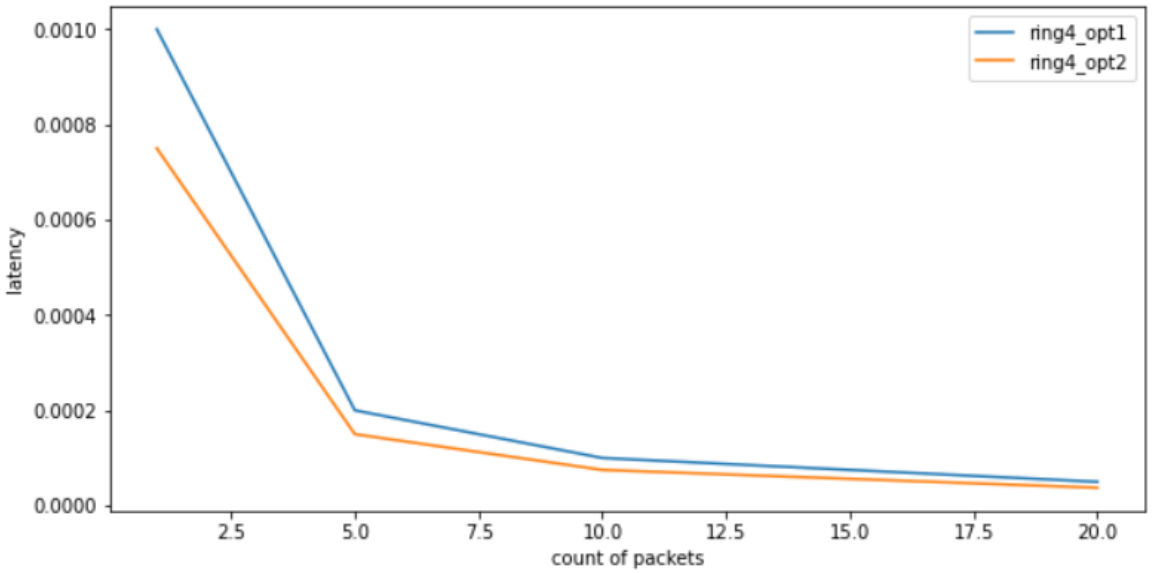


Графики:

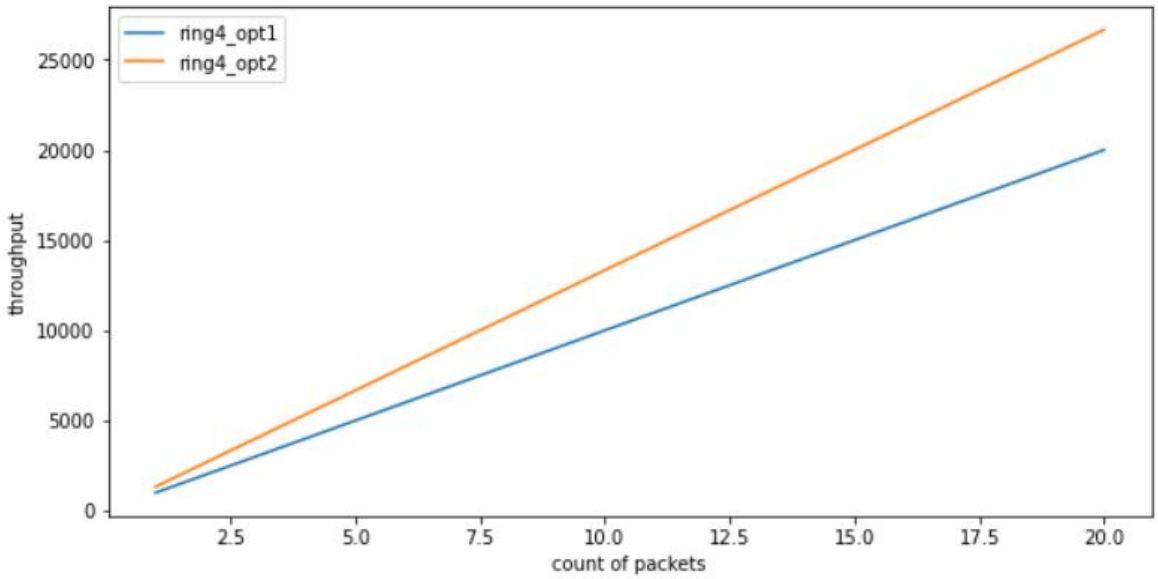
Latency от количества узлов, на сетях с 20 пакетами на узел



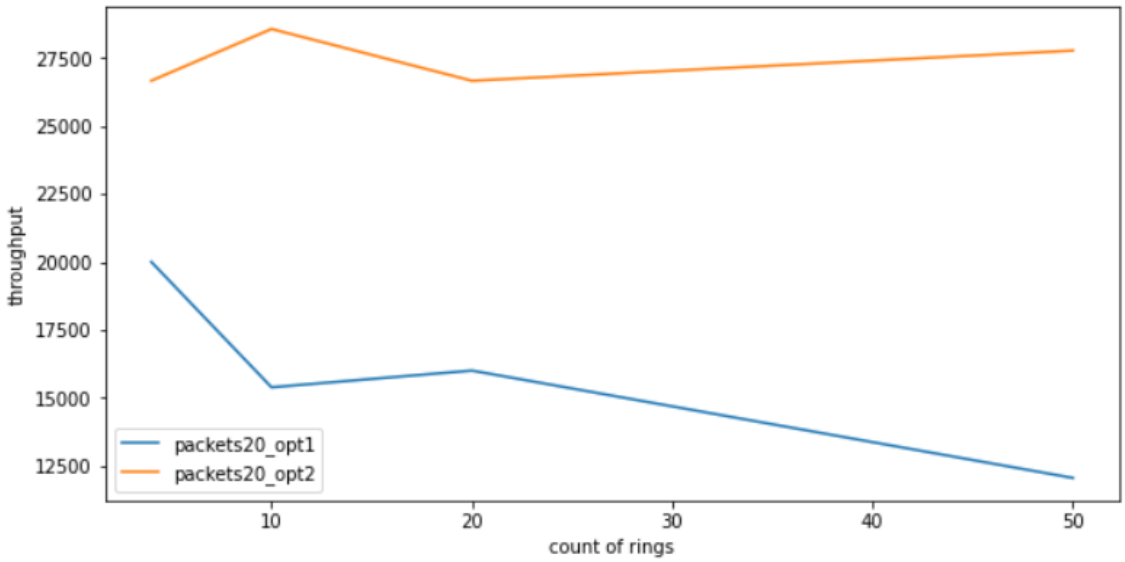
Latency от количества пакетов, на сетях с 4 узлами



Throughput от количества пакетов, на сетях с 4 узлами



Throughput от количества узлов, на сетях с 20 пакетами на узел



Графики ведут себя одинаково при разном количестве узлов. Как видно вторая оптимизация существенно повышает throughput, так как мы уже можем перемещать не синхронизируясь со всеми.

Вывод:

Были исследованы throughput и latency, также проверены влияние оптимизаций.