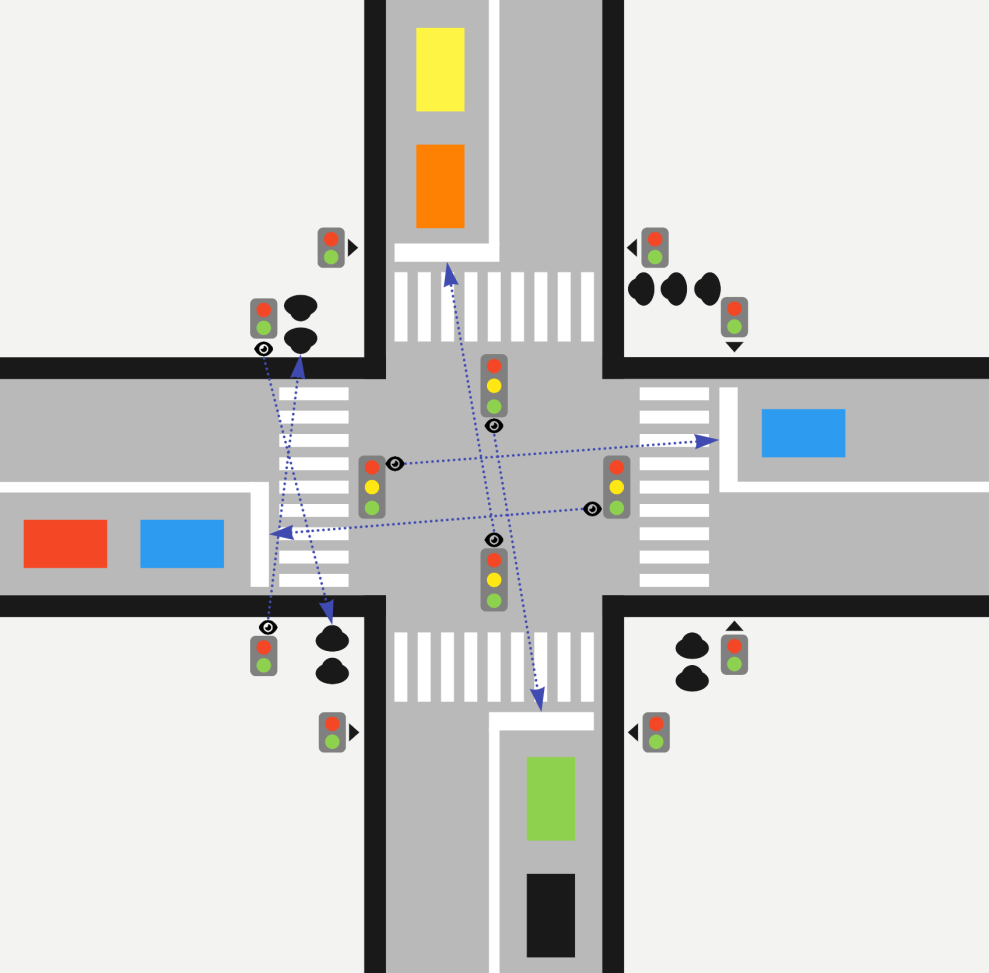
**Описание решения задачи «Организация движения на перекрестке»**

**Формулировка поставленной задачи.**

Рассмотрим перекресток на рисунке. На нем находится несколько светофоров, регулирующих движение автомобилей (4 шт), и несколько - движение пешеходов по переходам (8 шт):



У пешеходных светофоров 2 состояния, у автомобильных - 3. В каждый светофор встроена камера, которая фиксирует количество автомобилей/пешеходов в той очереди, для которых светофор установлен. Это очередь на противоположной стороне пешехода/перекрестка (см рисунок). Автомобили при проезде перекрестка едут либо прямо, либо направо. Люди и автомобили осуществляют переход или проезд перекрестка по одному, уменьшая размер соответствующей очереди на 1.

Каждый светофор имеет уникальный id. Светофоры могут общаться при помощи событий, отсылая события друг другу по id. Пересылаемые события - это некоторые контейнеры с данными (например, там может лежать количество людей/автомобилей в очереди, id отправителя, текущее состояние светофора). Светофор может взводить таймер, который через заданное время отсылают заданное событие на заданный id. Отправка события - это помещение контейнера в очередь событий для светофора, у каждого светофора очередь своя собственная. Светофоры обрабатывают события параллельно, независимо от друг от друга. При этом каждый светофор обрабатывает свои события последовательно, в том порядке, в каком они помещаются в очередь. Светофор может получить информацию о текущем состоянии любого другого светофора синхронно (не через событие).

***Задача:*** *придумать и описать адаптивный алгоритм работы светофоров для оптимизации общей пропускной способности перекрестка в зависимости от ситуации на перекрестке.*

**Краткое описание задачи**

Помимо взаимодействующих друг с другом светофоров, в их общую сеть встраивается некий сервер, обрабатывающий события, связанные с величиной очереди каждого автомобильного светофора. В зависимости от того, чья очередь длиннее, он изменяет приоритет на проезд автомобилей. Касаемо, пешеходных светофоров, им предоставляется сессионный интервал для активации, зависящий от максимального количества в любой из очередей. Пешеходные переходы активируются и деактивируются одновременно по команде сервера, лишь контролируя заполненность своей очереди.

Схема организации в данном случае выглядит так:



**Сервер - регулировщик**

Находится в одном из двух состояний:

1) пропускать автомобили;

2) пропускать пешеходов.

***Первое состояние***

В течение некоторого времени выбирает, каким светофорам предоставить приоритет и переключает их на зелёный, собирая информацию о количестве в очередях с определенной задержкой по следующему принципу (хоровода):

*первый если в нем больше, чем во втором;*

*второму если больше, чем в третьем;*

*третьему если больше, чем в четвертом;*

*четвертому если больше, чем в первом.*

Либо давать приоритет наибольшей очереди

Так же, по условию задачи, так как движение с каждого направления разрешено только либо вперёд, либо направо, противолежащие светофоры могут работать независимо друг от друга, так как маршруты проезда не пересекаются.

Переключается в режим пешеходов

***Второе состояние***

В течение некоторого времени или пока очередях не будет пусто пропускает пешеходов по всем дорогам.

Интервал работы в каждом режиме может быть рассчитан по принципу определения интервала с коэффициентом загруженности\*

**Каждый автомобильный светофор по отдельности** работает по следующему принципу.

Отслеживает, переключил ли регулировщик его состояние на зеленый. Если переключил, то пропускает *4 авто* + «*коэффициент загруженности»\** относительно других светофоров, который вычисляется по формуле

*(цел) (очередь текущего / (среднее трёх других) – 1) \* 2*

Например, у текущего светофора в очереди 15 машин, у других светофоров по 7. Соответственно светофор пропустит

4+((15 / ((7+7+7)/3)-1) \* 2=4 + 2.285714 ~ 6 (без остатка) автомобилей.

Или, у текущего светофора в очереди 7 машин, у других светофоров по 15. Соответственно светофор пропустит

4+((7 / ((15+15+15)/3)-1) \* 2=4 + (-1.0666) ~ 2 (без остатка) автомобиля.

На проезд одного автомобиля отводится одна секунда.

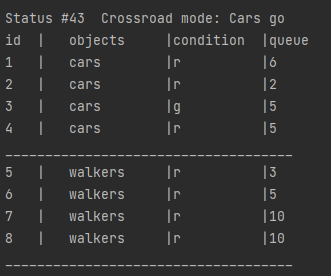
После пропуска необходимого кол-ва объектов светофор переключает сам себя в состояние жёлтого сигнала (автомобили завершают проезд), а после красного (проезд запрещен)

Уходит в сон на некоторое время.

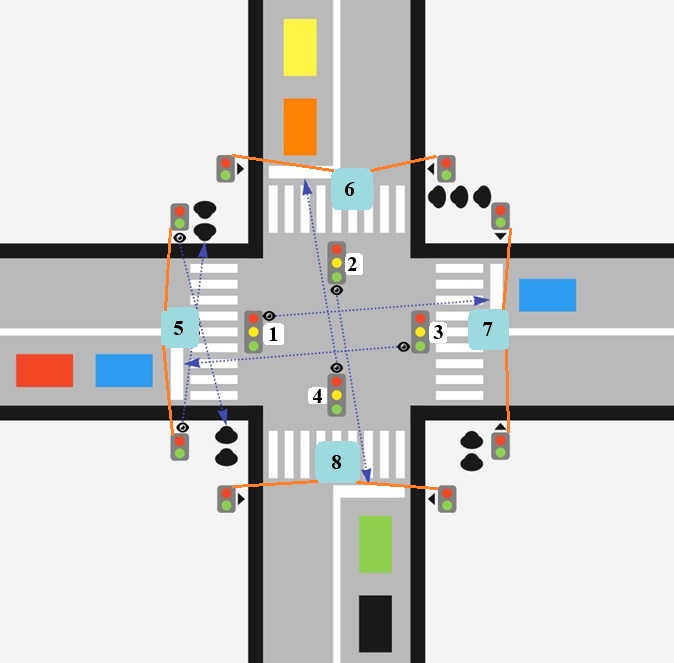
\*-нечто, отдаленно напоминает PID-регулятор (компенсатор ошибки).

**Реализация**

Пример реализация описанной модели выполнен в качестве консольного приложения на языке Java с применением библиотек Tread, Collections. В качестве вывода пользователь получает статус-таблицу о текущем состоянии перекрестка. Можно установить количество статусов и частоту обновления.



ID светофоров распределены в следующем порядке



**Генератор очереди**

Для проверки работоспособности системы в первую очередь создается генератор очереди, который добавляет случайное количество объектов в случайный момент времени в случайную очередь.

**Класс TrafficLight**

Выполняет роль контейнера для данных каждого из 8 объектов (светофора)

**Потоки**

Для дальнейшей реализации решения необходимо 6 потоков:

1. генератор очереди;
2. поток-регулировщик;
3. поток обработки событий светофора 1;
4. поток обработки событий светофора 2;
5. поток обработки событий светофора 3;
6. поток обработки событий светофора 4;

Когда поток-регулировщик переходит в режим пропуска пешеходов, он дожидается пока все автомобильные светофоры не переключатся на красный. После чего активирует пешеходные светофоры обрабатывая их события самостоятельно.

**Вывод**

Существует множество способов решения. В данном случае предложен, на мой взгляд, наиболее оптимальный и простой, естественно требующий доработки. Ссылка на репозиторий: https://github.com/2dic/crossroad