Предлагаю решение на Python 3, представляющее из себя консольное приложение для моделирования ситуации на перекрёстке и проверки разработанного алгоритма.

**О ПРОГРАММЕ:**

В корне проекта в \_\_main\_\_.py содержится программа для создания и настройки 4+8 объектов светофоров, задания ситуации на перекрёстке, моделирования движения очередей.

В корне проекта в папке sources содержаться python файлы:

* constants.py – содержит константы настройки.
* communication.py – содержит класс Communication для передачи сообщений между объектами. В реальных условиях для общения 12 устройств заменяется на кодирование, передачу p2p по протоколу транспортного уровня и декодирование.
* traffic\_lights.py – содержит общий класс предок и 2 класса PedestrianTrafficLight и CarTrafficLight.

Классы CarTrafficLight и PedestrianTrafficLight – сгруппированные методы, реализующие логику управления светофорами, их общения с помощью пакетов, независимого хранения, получения и обработки информации.

4+8 объектов этих классов общаются между собой только через пакеты данных, хранят и собирают информацию об обстановке независимо, что позволяет с минимальными изменениями применять этот код в 12 независимых реальных устройствах. Нужно изменить только способ передачи пакетов и сделать физический вывод сигналов.

Хотя реализовать управление на клиент-серверной архитектуре, как реализуется в действительности, с одним дополнительным устройством-сервером и более простыми в устройстве светофорами-исполнителями было бы быстрее, дешевле и проще, выбран вариант создания распределённой сети, где 4 автомобильных светофора – равнозначные ведущие, а 8 пешеходных светофоров – равнозначные ведомые. В программировании есть принцип KISS, но тут конкуренция, а я вон что 3 дня придумывал.

Пакеты данных представляют из себя словари с типизированными ключами и значениями. Такая форма наиболее удобна, если потребуется использовать пакеты в формате json или xml. В каждом пакете передаётся id отправителя, описание действия, о котором сообщается, и дополнительные данные. Пакеты широковещательные.

Все возможные виды пакетов:

* message = {'id': self.id, 'do': 'change\_monitored\_queue', 'change': value} – отправляется всеми светофорами ко всем светофорам, для отслеживания получения и работы каждого
* message = {'id': self.id, 'leader': leader, 'do': 'change\_state', 'state': self.state} – отправляется всеми светофорами к ведущим светофорам, для отслеживания очередей
* message = {'id': self.id, 'do': 'leader\_taken'} – отправляется лидером к ведущим светофорам
* message = {'id': self.id, 'do': 'leader\_released'} – отправляется лидером к ведущим светофорам

**АЛГОРИТМ:**

4 равнозначных объекта-«устройства» автомобильных светофоров постоянно знают и отслеживают, за счёт сообщений, о размере суммарной очереди: у себя и у двух пешеходных светофоров слева, о размере других трёх суммарных очередях.

В конкретный момент только один из 4х автомобильных светофоров может быть лидером, для этого у каждого есть флаги is\_leader и leader\_exists, при смене лидера передается сообщение 'leader\_released'. Затем каждый ведущий светофор проверяет, является ли его групповая очередь наибольшей или второй по размеру, следующей за очередью предыдущего лидера, и, если да, берёт лидерство пакетом 'leader\_taken'.

Став лидером, автомобильный светофор запускает процесс своего открытия timer\_yellow в отдельном потоке, о смене состояния сообщает пакетами 'change\_state', где 'id' = 'leader'.

Сначала включается жёлтый, ожидается установленное время жёлтого сигнала, затем включается зелёный.

Все остальные светофоры определяют по 'id' == 'leader', что это управляющий пакет и переключают своё состояние в зависимости от состояния лидера.

Если светофор ведомый в группе лидера, то он зажжёт зелёный сразу после лидера, если лидер другого цвета, то ведомый будет каждый раз проверять и, если надо, устанавливать красный.

Если светофор ведомый НЕ в группе лидера – он красный.

Если светофор ведущий НЕ лидер – он красный.

Лидерство отдаётся, когда групповая очередь светофора станет равной нулю, отправляется 'leader\_released', светофоры выбирают нового лидера, цикл повторяется, первым шагом закрывая открытые светофоры.

Изменение очереди симулируется из \_\_main\_\_ функцией simulation\_of\_queue\_movement с помощью метода светофора monitored\_queue\_size\_add, который отправляет ведущим пакет с информацией об изменении очереди каждого светофора.

Стартом работы цикла каждого светофора является получение им любого нового сообщения.

В консоль для демонстрации выводятся не все сообщения. В общем их примерно х8, в основном о подтверждении реакции на получение цветов. Выводятся только информативные (командные, или которые что-то поменяли)

**ЧТО Я УЧЁЛ В АЛГОРИТМЕ:**

* Полная независимость «устройств», код для которых находится в классах CarTrafficLight и PedestrianTrafficLight.
* Распределённое управление с отсутствием централизованного сервера.
* Удобство переноса кода на реальное устройство
* Для оптимизации пропускной способности решено выбирать наибольшую суммарную очередь и пропускать её полностью, чтобы не тратить время на разгон-торможение и реакцию участников очереди.
* Ситуация с двумя одинаковыми очередями разрешается тем, что об одной из очередей доложено в любом случае быстрее, и лидер этой группы уже взял лидерство.
* Одновременное взятие лидерства двумя светофорами невозможно из-за флагов.
* Для предотвращения постоянного пропуска одной бесконечной очереди, время принятия лидерства засекается в \_\_take\_the\_lead\_time, и, если оно превысит MAX\_LEADER\_TIME, лидерство передаётся следующей по длине очереди группе светофоров.

Прошу в любом случае какой-то фитбек, код-ревью от более опытных.

<https://t.me/spirinis>

На инженера-программиста я учился 6 лет, но у меня есть довольно много знаний в компьютерном зрении и нейросетях. Но не достаточно, чтобы пройти отбор на Junior 1 к 100. Планирую развиваться и развиваюсь в этом направлении. А у вас и вакансии такие есть. Хочу к вам, вижу будущее.