Санкт-Петербургский политехнический университет Петра Великого
Высшая школа прикладной математики и вычислительной физики
Кафедра прикладной математики

**Отчёт по лабораторной работе №2**

по дисциплине «Компьютерные сети»

на тему

# Реализация протокола динамической маршрутизации Open Shortest Path First

Выполнил студент гр. 5040102/00201
Жуков А.К.

Преподаватель
Баженов А.Н.

Санкт-Петербург
2022 год

# Оглавление

## Постановка задачи

В лабораторной работе требуется разработать систему способную поддерживать множество взаимодействующих друг с другом маршрутизаторов. Маршрутизаторы организуют между собой сеть обмена сообщениями, таким образом что сообщение доставляется по кратчайшему пути (протокол Open Shortest Path First [2]).

Необходимо рассмотреть:

- Три вида топологии сети: линейная, кольцевая, звездчатая.
- Перестройку таблиц достижимости при стохастических разрывах связи.

Протокол Open Shortest Path First - протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала (link-state technology) и использующий для нахождения кратчайшего пути алгоритм Дейкстры [3].

Принцип работы заключается в следующем:

1. После включения маршрутизаторов протокол ищет непосредственно подключенных соседей и устанавливает с ними «дружеские» отношения.
2. Затем они обмениваются друг с другом информацией о подключенных и доступных им сетях. То есть они строят карту сети (топологию сети). Данная карта одинакова на всех маршрутизаторах.
3. На основе полученной информации запускается алгоритм SPF (Shortest Path First, «выбор наилучшего пути»), который рассчитывает оптимальный маршрут к каждой сети. Данный процесс похож на построение дерева, корнем которого является сам маршрутизатор, а ветвями — пути к доступным сетям.

## Реализация

Для реализации системы был выбран язык Python и среда разработки PyCharm.

Роутеры связаны между собой в виде орграфа с ребрами единичного веса. Для маршрутизации сообщений об изменениях в топологии выделен отдельный роутер Owner.

Все роутеры запускаются в отдельных потоках при помощи модуля threading.

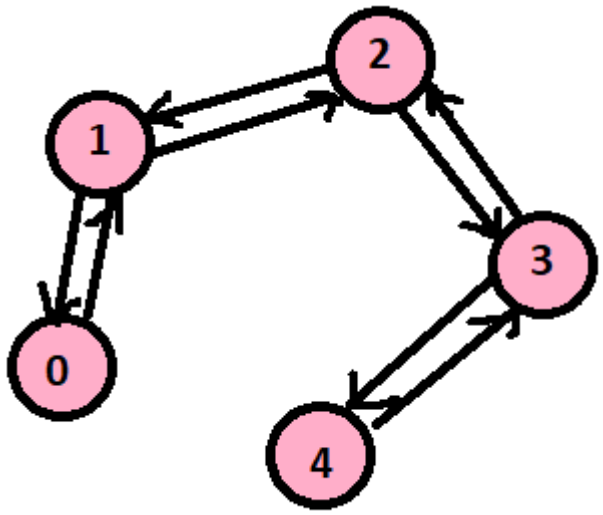
Роутеры обмениваются сообщениями следующих типов:

- NEIGHBORS – сообщение о добавлении в топологию новых соседей. Когда сообщение приходит роутеру Owner, он рассылает всем остальным роутерам это сообщение за исключением отправителя.
- GET_TOPOLOGY – запрос к Owner на получение текущей топологии. В ответ Owner посылает сообщение UPDATE_TOPOLOGY, которое содержит текущую топологию.
- UPDATE_TOPOLOGY – сообщение для роутера об обновлении топологии
- NODE_REMOVE – сообщение для Owner об отключении роутера
- PRINT_PATHS – запрос на печать кратчайших путей для текущей топологии.

Код приложения: https://github.com/akzhukov/CompNetworksLabs

## Тестирование работы программы

### Линейная топология




nodes: [0, 1, 2, 3, 4]

neighbors: [[1], [0, 2], [1, 3], [2, 4], [3]]

**Подключение роутеров к сети:**

Owner received message from router(0): (MsgType.NEIGHBORS: [1])

Owner received message from router(0): (MsgType.GET_TOPOLOGY: None)

router(0) : (MsgType.UPDATE_TOPOLOGY)

Owner received message from router(1): (MsgType.NEIGHBORS: [0, 2])

Owner received message from router(1): (MsgType.GET_TOPOLOGY: None)

router(0) : (MsgType.NEIGHBORS: {'index': 1, 'neighbors': [0, 2]})

router(1) : (MsgType.UPDATE_TOPOLOGY)

Owner received message from router(2): (MsgType.NEIGHBORS: [1, 3])

Owner received message from router(2): (MsgType.GET_TOPOLOGY: None)

router(0) : (MsgType.NEIGHBORS: {'index': 2, 'neighbors': [1, 3]})

router(1) : (MsgType.NEIGHBORS: {'index': 2, 'neighbors': [1, 3]})

router(2) : (MsgType.UPDATE_TOPOLOGY)

Owner received message from router(3): (MsgType.NEIGHBORS: [2, 4])

Owner received message from router(3): (MsgType.GET_TOPOLOGY: None)

router(1) : (MsgType.NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})

router(2) : (MsgType.NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})

router(0) : (MsgType.NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})

router(3) : (MsgType.UPDATE_TOPOLOGY)

Owner received message from router(4): (MsgType.NEIGHBORS: [3])

Owner received message from router(4): (MsgType.GET_TOPOLOGY: None)

router(0) : (MsgType.NEIGHBORS: {'index': 4, 'neighbors': [3]})

router(3) : (MsgType.NEIGHBORS: {'index': 4, 'neighbors': [3]})

router(2) : (MsgType.NEIGHBORS: {'index': 4, 'neighbors': [3]})

router(1) : (MsgType.NEIGHBORS: {'index': 4, 'neighbors': [3]})

router(4) : (MsgType.UPDATE_TOPOLOGY)

router(0) : (MsgType.PRINT_PATHS: None)

router(4) : (MsgType.PRINT_PATHS: None)

**Полученные кратчайшие пути:**

0: [[0], [0, 1], [0, 1, 2], [0, 1, 2, 3], [0, 1, 2, 3, 4]]

1: [[1, 0], [1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]

2: [[2, 1, 0], [2, 1], [2], [2, 3], [2, 3, 4]]

3: [[3, 2, 1, 0], [3, 2, 1], [3, 2], [3], [3, 4]]

4: [[4, 3, 2, 1, 0], [4, 3, 2, 1], [4, 3, 2], [4, 3], [4]]

Удалим один из роутеров, например под номером 4, и посмотрим, как перестроится таблица кратчайших путей.

Owner received message from router(4): (MsgType.NODE_REMOVE: None)

router(0) : (MsgType.NODE_REMOVE: 4)

router(2) : (MsgType.NODE_REMOVE: 4)

router(1) : (MsgType.NODE_REMOVE: 4)

router(3) : (MsgType.NODE_REMOVE: 4)

**Таблица достижимости после обновления топологии**

0: [[0], [0, 1], [0, 1, 2], [0, 1, 2, 3], []]

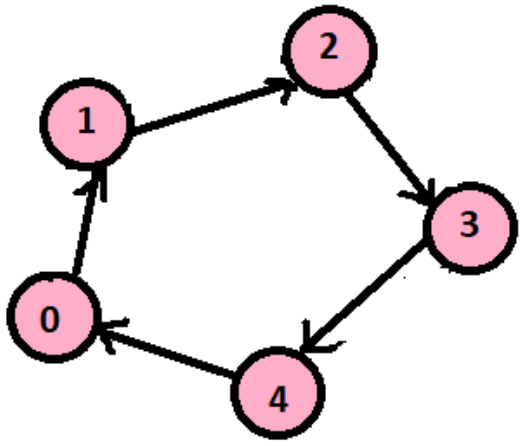
1: [[1, 0], [1], [1, 2], [1, 2, 3], []]

2: [[2, 1, 0], [2, 1], [2], [2, 3], []]

3: [[3, 2, 1, 0], [3, 2, 1], [3, 2], [3], []]

4: [[], [], [], [], [4]]

## Кольцевая топология



nodes: [0, 1, 2, 3, 4]

neighbors: [[1], [2], [3], [4], [0]]

**Кратчайшие пути:**

0: [[0], [0, 1], [0, 1, 2], [0, 1, 2, 3], [0, 1, 2, 3, 4]]

1: [[1, 2, 3, 4, 0], [1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]

2: [[2, 3, 4, 0], [2, 3, 4, 0, 1], [2], [2, 3], [2, 3, 4]]

3: [[3, 4, 0], [3, 4, 0, 1], [3, 4, 0, 1, 2], [3], [3, 4]]

4: [[4, 0], [4, 0, 1], [4, 0, 1, 2], [4, 0, 1, 2, 3], [4]]

**Кратчайшие пути после отключение второго и третьего роутера:**

0: [[0], [0, 1], [], [], []]

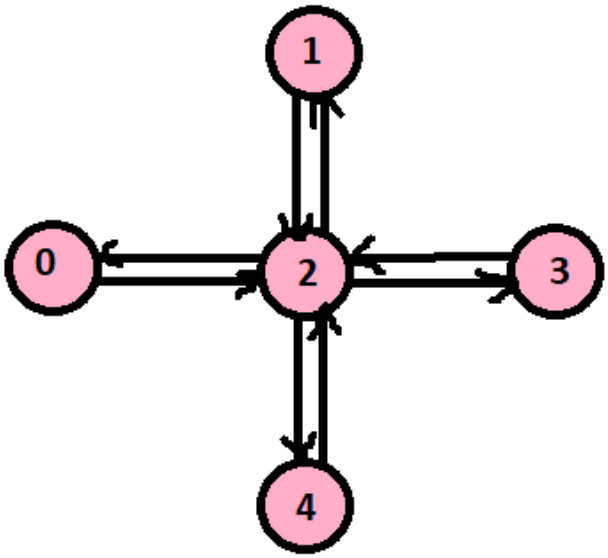
1: [[], [1], [], [], []]

2: [[], [], [2], [], []]

3: [[], [], [], [3], []]

4: [[4, 0], [4, 0, 1], [], [], [4]]

## Звездчатая топология




nodes: [0, 1, 2, 3, 4]

neighbors: [[2], [2], [0, 1, 3, 4], [2], [2]]

**Кратчайшие пути:**

0: [[0], [0, 2, 1], [0, 2], [0, 2, 3], [0, 2, 4]]

1: [[1, 2, 0], [1], [1, 2], [1, 2, 3], [1, 2, 4]]

2: [[2, 0], [2, 1], [2], [2, 3], [2, 4]]

3: [[3, 2, 0], [3, 2, 1], [3, 2], [3], [3, 2, 4]]

4: [[4, 2, 0], [4, 2, 1], [4, 2], [4, 2, 3], [4]]

**Кратчайшие пути после удаление второго роутера:**

0: [[0], [], [], [], []]

1: [[], [1], [], [], []]

2: [[], [], [2], [], []]

3: [[], [], [], [3], []]

4: [[], [], [], [], [4]]

Видим, что после удаления второго роутера, все роутеры стали недостижимы из других роутеров.

## Результаты

По результатам данной лабораторной работы была реализована программа, моделирующая работу алгоритма *Open Shortest Path First* со стохастическими разрывами соединения. Программа была протестирована на трех топологиях: линейной, кольцевой и звездчатой.

## Использованная литература


1. А.Н. Баженов, Компьютерные сети, курс лекций
2. https://ru.wikipedia.org/wiki/OSPF
   Статья Википедии «*Open Shortest Path First*»
3. https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Дейкстры
   Статья «Алгоритм Дейкстры»