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
Сергеев. Г.К.

Преподаватель
Баженов А.Н.

Санкт-Петербург
2022 год

Оглавление

Постановка задачи 3
Протокол OSPF 3
Реализация 4
Примеры работы 4
Линейная топология 4
Топология «звезда» 7
Результаты 8
Приложение 8
Список литературы 8

Постановка задачи

Требуется разработать систему из взаимодействующих друг с другом маршрутизаторов, которые организуются в сеть и обеспечивают передачу сообщений от каждого маршрутизатора к каждому по кратчайшему пути (протокол Open Shortest Path First). [1]

Необходимо рассмотреть:

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

Протокол OSPF

OSPF – это протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала, использующий для нахождения кратчайшего пути алгоритм Дейкстры [3].
Принцип работы протокола [2]:

1. Включение маршрутизатора и присвоение ему уникального ID.
2. После включения нескольких маршрутизаторов протокол ищет непосредственно подключенных соседей (через какие-либо интерфейсы поддерживающие OSPF) и устанавливает с ними «дружеские» отношения с помощью т.н. «hello-пакетов».
3. Маршрутизаторы обмениваются друг с другом информацией о подключенных и доступных им сетях - строят топологию сети. Топология одинакова на всех маршрутизаторах.
4. На основе полученной информации запускается алгоритм SPF (Shortest Path First), который рассчитывает оптимальный маршрут к каждой сети.

Выделенный маршрутизатор (Designated Router) — управляет процессом рассылки LSA (сообщений о состоянии канала) в сети. Каждый маршрутизатор сети устанавливает отношения смежности с DR. Информация об изменениях в сети отправляется маршрутизатором, обнаружившим это изменение, на DR, который отправляет эту информацию остальным маршрутизаторам.

Реализация

Система реализована на языке программирования Python.

Роутеры связаны с помощью орграфа с единичными весами ребер. Также отдельно (вне топологии) выделен designated router, который обеспечивает маршрутизацию сообщений об изменениях в топологии.

Реализация канала связи – протокол Go-Back-N, но в данной работе внимание уделяется порядку отправки и получения сообщений.

Возможные типы сообщений:

Для Designated Router

NEIGHBORS ([neighbors]) – запрос на добавление в топологию новых соседей.
GET_TOPOLOGY () – запрос на получение от DR текущей топологии сети
OFF () – Сообщение об отключении роутера

Для Router

NEIGHBORS ($i, neighbors_i$) – сообщение от DR о необходимости добавления новых соседей для узла i
SET_TOPOLOGY (topology) – сообщение от DR с информацией о текущей топологии
OFF (i) – сообщение от DR о необходимости исключения узла i из топологии.

Примеры работы

Линейная топология

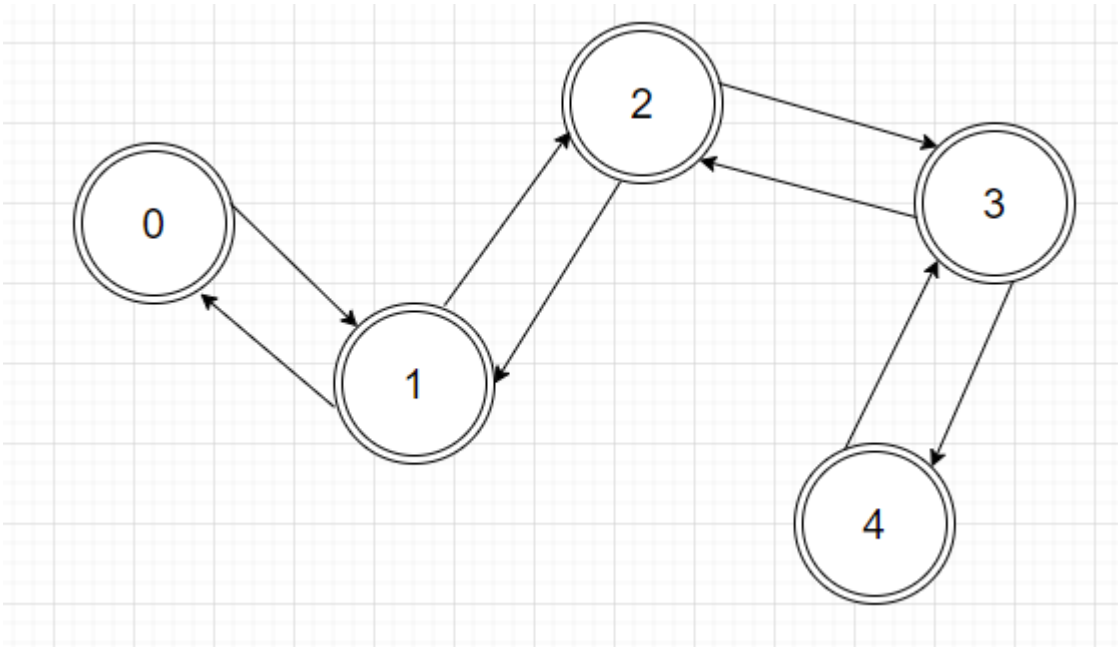

Рисунок 1. Конфигурация сети «линейная» из пяти роутеров

Конфигурация:

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

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

Подключение роутеров:

Designated Router received from no. (0) message: (NEIGHBORS: [1])
Designated Router received from no. (0) message: (GET_TOPOLOGY)
Router no. (1) received message: (NEIGHBORS: {'index': 0, 'neighbors': [1]})
Designated Router received from no. (1) message: (NEIGHBORS: [0, 2])
Designated Router received from no. (1) message: (GET_TOPOLOGY)
Router no. (0) received message: (SET_TOPOLOGY)
Router no. (2) received message: (NEIGHBORS: {'index': 1, 'neighbors': [0, 2]})
Router no. (0) received message: (NEIGHBORS: {'index': 1, 'neighbors': [0, 2]})
Designated Router received from no. (2) message: (NEIGHBORS: [1, 3])
Router no. (1) received message: (SET_TOPOLOGY)
Designated Router received from no. (1) message: (NEIGHBORS: [0])
Designated Router received from no. (2) message: (GET_TOPOLOGY)
Router no. (4) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1, 3]})
Router no. (2) received message: (NEIGHBORS: {'index': 1, 'neighbors': [0]})
Router no. (0) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1, 3]})
Router no. (0) received message: (NEIGHBORS: {'index': 1, 'neighbors': [0]})
Router no. (3) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1, 3]})
Router no. (1) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1, 3]})
Designated Router received from no. (3) message: (NEIGHBORS: [2, 4])
Router no. (2) received message: (SET_TOPOLOGY)
Router no. (4) received message: (NEIGHBORS: {'index': 1, 'neighbors': [0]})
Designated Router received from no. (4) message: (NEIGHBORS: [3])
Router no. (3) received message: (NEIGHBORS: {'index': 1, 'neighbors': [0]})
Router no. (1) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})
Designated Router received from no. (2) message: (NEIGHBORS: [1])
Router no. (3) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Router no. (1) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Designated Router received from no. (3) message: (GET_TOPOLOGY)
Router no. (2) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})
Router no. (3) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1]})
Router no. (0) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})
Designated Router received from no. (4) message: (GET_TOPOLOGY)
Router no. (2) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Router no. (3) received message: (SET_TOPOLOGY)
Designated Router received from no. (2) message: (NEIGHBORS: [1])
Designated Router received from no. (3) message: (NEIGHBORS: [2])
Designated Router received from no. (3) message: (NEIGHBORS: [4])
Router no. (1) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1]})
Router no. (1) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (3) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1]})

Router no. (0) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Router no. (2) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (2) received message: (NEIGHBORS: {'index': 3, 'neighbors': [4]})
Router no. (1) received message: (NEIGHBORS: {'index': 3, 'neighbors': [4]})
Router no. (4) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2, 4]})
Router no. (0) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1]})
Router no. (0) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (0) received message: (NEIGHBORS: {'index': 3, 'neighbors': [4]})
Router no. (1) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (4) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1]})
Router no. (4) received message: (SET_TOPOLOGY)
Router no. (4) received message: (NEIGHBORS: {'index': 2, 'neighbors': [1]})
Router no. (2) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (4) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (4) received message: (NEIGHBORS: {'index': 3, 'neighbors': [4]})
Designated Router received from no. (4) message: (NEIGHBORS: [3])
Router no. (4) received message: (NEIGHBORS: {'index': 3, 'neighbors': [2]})
Router no. (3) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Router no. (1) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Router no. (2) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})
Router no. (0) received message: (NEIGHBORS: {'index': 4, 'neighbors': [3]})

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]]

Пусть отключен третий (no.3) роутер:
Designated Router received from no. (3) message: (OFF)
Router no. (2) received message: (MessageType.OFF: 3)
Router no. (1) received message: (MessageType.OFF: 3)
Router no. (0) received message: (MessageType.OFF: 3)
Router no. (4) received message: (MessageType.OFF: 3)

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

Топология «звезда»

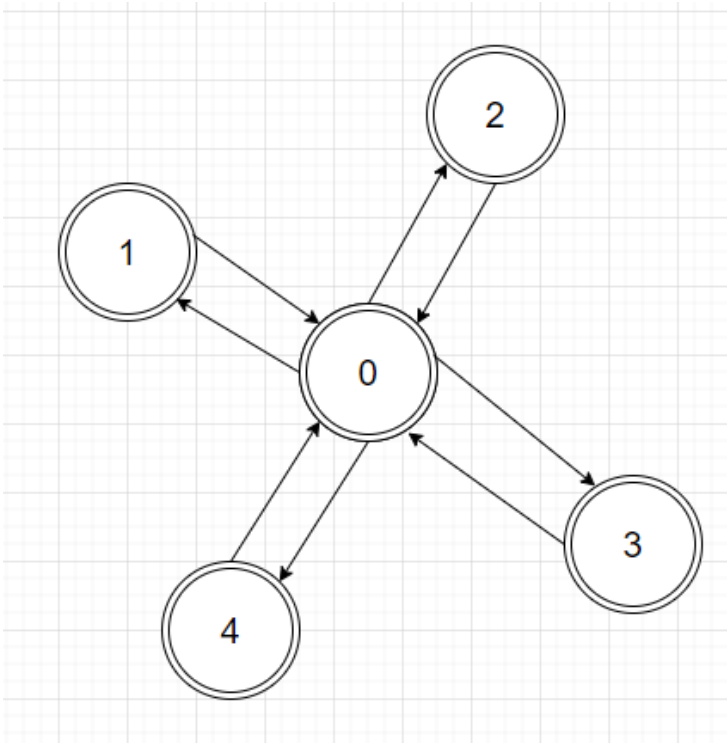

Рисунок 2. Конфигурация сети «звезда» из пяти роутеров

Исходная конфигурация:

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

Исходная топология после включения и обработки сообщений:

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

При отключении любого роутера кроме центрального путь до него пропадет из топологий других роутеров, а его топология примет вид изолированной, как в линейном примере.

Пусть отключится центральный роутер (no. 0):
0: [[0], [], [], [], []]
1: [[], [1], [], [], []]
2: [[], [], [2], [], []]
3: [[], [], [], [3], []]
4: [[], [], [], [], [4]]

Все роутеры оказались изолированы из-за конфигурации данной сети. Она устойчива к поломке «крайних» звеньев, но уязвима в «центральной» части.[4]

Результаты

Таким образом, была реализовано моделирование протокола динамической маршрутизации OSPF со стохастическими разрывами соединения. Программа тестировалась на на трёх топологиях – «линейная», «кольцо», «звезда».

Приложение

Код программы доступен на GitHub.

Список литературы

1. А.Н. Баженов, Компьютерные сети, курс лекций
2. Описание протокола OSPF – интернет-ресурс. https://en.wikipedia.org/wiki/Open_Shortest_Path_First
3. Алгоритм Дейкстры для поиска кратчайшего пути – интернет-ресурс. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
4. Описание топологии сети «звезда» – интернет-ресурс. https://ru.wikipedia.org/wiki/Звезда_(топология_компьютерной_сети)