Санкт-Петербургский политехнический университет Петра Великого
Физико-механический институт

Отчёт по лабораторной работе

по дисциплине «Компьютерные сети»

«Реализация протокола маршрутизации OSPF»

Выполнил
студент гр. 5040102/30201 Завьялов И.В. _____

Проверил
доцент, к.ф.-м.н. Баженов А.Н. _____

Санкт-Петербург
2024

Содержание

1 Теория

OSPF (Open Shortest Path First) — это протокол динамической маршрутизации, основанный на технологии отслеживания состояния каналов связи (link-state). Он применяется для построения топологии сети и поиска кратчайшего маршрута между узлами с использованием алгоритма Дейкстры.

Основные особенности протокола OSPF:

- OSPF работает с использованием метрики на основе стоимости (*cost*), которая может быть основана на различных параметрах, таких как пропускная способность канала, задержка и загруженность.
- OSPF является внутренним протоколом маршрутизации (Interior Gateway Protocol, IGP), предназначенным для работы внутри автономной системы (AS).
- Протокол поддерживает иерархическую структуру сети, разделяя её на области (areas), что позволяет уменьшить размер таблиц маршрутизации и снизить нагрузку на процессор маршрутизатора.
- OSPF использует мультикаст-адреса (224.0.0.5 и 224.0.0.6) для передачи информации о состоянии сети между маршрутизаторами.

Описание работы протокола OSPF:

- После включения маршрутизатора OSPF обнаруживает соседей, которые подключены непосредственно, и устанавливает с ними отношения (*adjacency*). Эти отношения позволяют маршрутизаторам обмениваться информацией о сети.
- Каждый маршрутизатор передаёт свои данные о состоянии каналов (*Link State Advertisements, LSA*) всем маршрутизаторам в той же области. Это достигается с помощью механизма *flooding*.
- На основе полученной информации о состоянии каналов все маршрутизаторы строят одинаковую топологическую карту сети, хранящуюся в базе данных состояния канала (*Link State Database, LSDB*).
- С использованием алгоритма Дейкстры маршрутизаторы рассчитывают кратчайший путь (*Shortest Path First, SPF*) до всех других узлов сети, формируя таблицу маршрутизации.
- При изменении состояния сети (например, отказ или восстановление связи) маршрутизаторы обновляют свою базу данных LSDB и пересчитывают маршруты.

Преимущества OSPF:

- Быстрая сходимость — OSPF оперативно реагирует на изменения в топологии сети.
- Поддержка сложных иерархических структур, что позволяет масштабировать сеть.
- Использование метрики, которая учитывает качество канала, позволяет выбирать оптимальные маршруты.

Недостатки OSPF:

- Сложность конфигурации и управления в больших сетях.
- Значительная загрузка процессора и памяти маршрутизаторов из-за обработки LSA и расчёта маршрутов.

2 Реализация

2.1 Линейная топология

Граф сети с линейной топологией приведён на рисунке 1. При радиусе соединения $r = 1.5$.

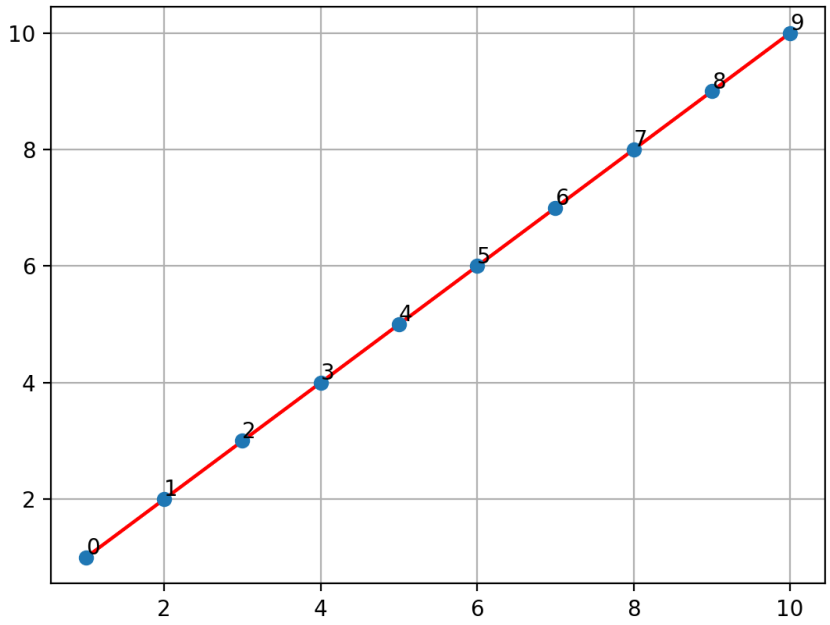

Рис. 1: Граф сети с линейной топологией

С помощью алгоритма Дейкстры найдем кратчайшие пути от каждого узла. Результаты поиска находятся в `/path/line_ospf.txt`.

```
Start node 0:
path 0 -> 0: [0]
path 0 -> 1: [0, 1]
path 0 -> 2: [0, 1, 2]
path 0 -> 3: [0, 1, 2, 3]
path 0 -> 4: [0, 1, 2, 3, 4]
path 0 -> 5: [0, 1, 2, 3, 4, 5]
path 0 -> 6: [0, 1, 2, 3, 4, 5, 6]
path 0 -> 7: [0, 1, 2, 3, 4, 5, 6, 7]
path 0 -> 8: [0, 1, 2, 3, 4, 5, 6, 7, 8]
path 0 -> 9: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
----------------------------------------
Start node 1:
path 1 -> 0: [1, 0]
path 1 -> 1: [1]
path 1 -> 2: [1, 2]
path 1 -> 3: [1, 2, 3]
path 1 -> 4: [1, 2, 3, 4]
path 1 -> 5: [1, 2, 3, 4, 5]
path 1 -> 6: [1, 2, 3, 4, 5, 6]
path 1 -> 7: [1, 2, 3, 4, 5, 6, 7]
path 1 -> 8: [1, 2, 3, 4, 5, 6, 7, 8]
path 1 -> 9: [1, 2, 3, 4, 5, 6, 7, 8, 9]
----------------------------------------
Start node 2:
path 2 -> 0: [2, 1, 0]
path 2 -> 1: [2, 1]
path 2 -> 2: [2]
path 2 -> 3: [2, 3]
path 2 -> 4: [2, 3, 4]
path 2 -> 5: [2, 3, 4, 5]
path 2 -> 6: [2, 3, 4, 5, 6]
path 2 -> 7: [2, 3, 4, 5, 6, 7]
path 2 -> 8: [2, 3, 4, 5, 6, 7, 8]
path 2 -> 9: [2, 3, 4, 5, 6, 7, 8, 9]
----------------------------------------
```

Листинг 1: Содержимое файла line_ospf.txt

Пути для всех узлов приведены в файле `/path/line_ospf.txt`.

Теперь удалим один узел из сети (например, узел 3). Тогда граф будет выглядеть следующим образом (рисунок 2).

Найдем теперь кратчайшие пути до каждого из узлов. Результаты пред-

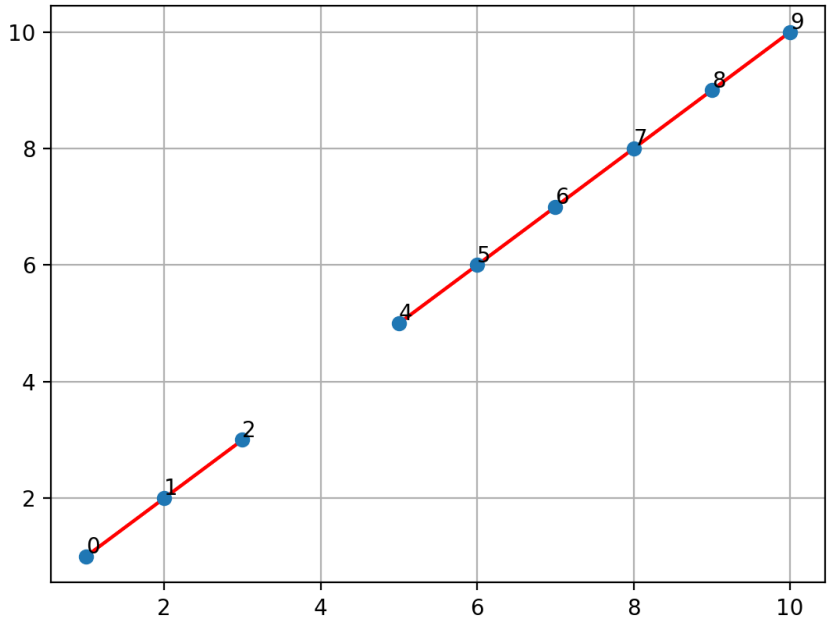

Рис. 2: Граф сети с удалённым узлом

ставлены в `/path/line_modified_ospf.txt`. В листинге 2 представлены пути для некоторых узлов.

```
Start node 0:
path 0 -> 0: [0]
path 0 -> 1: [0, 1]
path 0 -> 2: [0, 1, 2]
----------------------------------------
Start node 1:
path 1 -> 0: [1, 0]
path 1 -> 1: [1]
path 1 -> 2: [1, 2]
----------------------------------------
Start node 2:
path 2 -> 0: [2, 1, 0]
path 2 -> 1: [2, 1]
path 2 -> 2: [2]
----------------------------------------
Start node 3:
path 3 -> 3: [3]
----------------------------------------
Start node 4:
path 4 -> 4: [4]
path 4 -> 5: [4, 5]
```

```
path 4 -> 6: [4, 5, 6]
path 4 -> 7: [4, 5, 6, 7]
path 4 -> 8: [4, 5, 6, 7, 8]
path 4 -> 9: [4, 5, 6, 7, 8, 9]
----------------------------------------
Start node 5:
path 5 -> 4: [5, 4]
path 5 -> 5: [5]
path 5 -> 6: [5, 6]
path 5 -> 7: [5, 6, 7]
path 5 -> 8: [5, 6, 7, 8]
path 5 -> 9: [5, 6, 7, 8, 9]
----------------------------------------
```

Листинг 2: Содержимое файла line_modified_ospf.txt

2.2 Кольцевидная топология

Граф сети с кольцевидной топологией приведён на рисунке 3. При радиусе соединения $r = 3.0$.

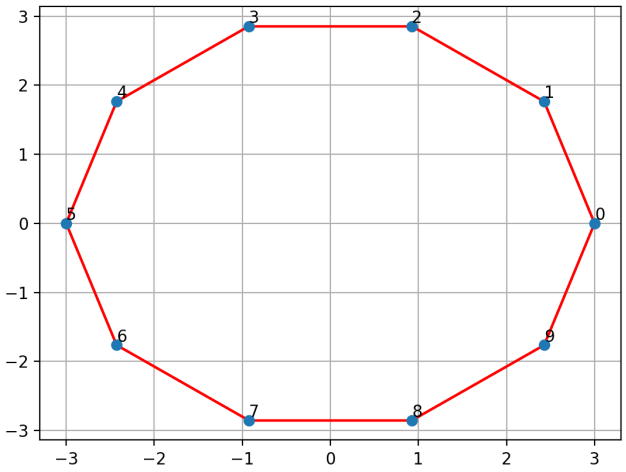

Рис. 3: Граф сети с кольцевидной топологией

С помощью алгоритма Дейкстры найдем кратчайшие пути от каждого узла. Результаты поиска находятся в `/path/ring_ospf.txt`.

```
Start node 0:
path 0 -> 0: [0]
path 0 -> 1: [0, 1]
path 0 -> 2: [0, 1, 2]
path 0 -> 3: [0, 1, 2, 3]
```

```
path 0 -> 4: [0, 1, 2, 3, 4]
path 0 -> 5: [0, 1, 2, 3, 4, 5]
path 0 -> 6: [0, 9, 8, 7, 6]
path 0 -> 7: [0, 9, 8, 7]
path 0 -> 8: [0, 9, 8]
path 0 -> 9: [0, 9]
----------------------------------------
Start node 1:
path 1 -> 0: [1, 0]
path 1 -> 1: [1]
path 1 -> 2: [1, 2]
path 1 -> 3: [1, 2, 3]
path 1 -> 4: [1, 2, 3, 4]
path 1 -> 5: [1, 2, 3, 4, 5]
path 1 -> 6: [1, 2, 3, 4, 5, 6]
path 1 -> 7: [1, 0, 9, 8, 7]
path 1 -> 8: [1, 0, 9, 8]
path 1 -> 9: [1, 0, 9]
----------------------------------------
Start node 2:
path 2 -> 0: [2, 1, 0]
path 2 -> 1: [2, 1]
path 2 -> 2: [2]
path 2 -> 3: [2, 3]
path 2 -> 4: [2, 3, 4]
path 2 -> 5: [2, 3, 4, 5]
path 2 -> 6: [2, 3, 4, 5, 6]
path 2 -> 7: [2, 3, 4, 5, 6, 7]
path 2 -> 8: [2, 1, 0, 9, 8]
path 2 -> 9: [2, 1, 0, 9]
----------------------------------------
```

Листинг 3: Содержимое файла ring_ospf.txt

Пути для всех узлов приведены в файле `/path/ring_ospf.txt`.

Теперь удалим один узел из сети (например, узел 3). Тогда граф будет выглядеть следующим образом (рисунок 4).

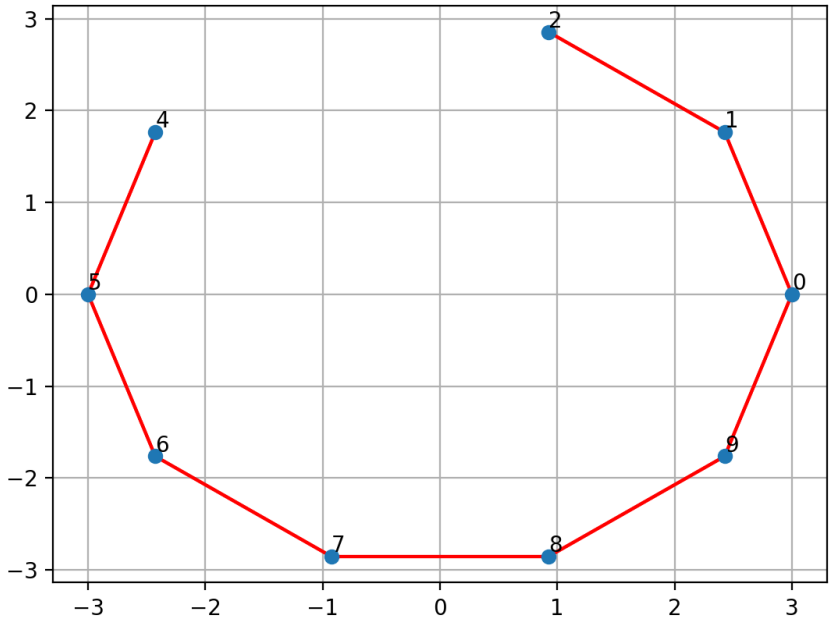

Рис. 4: Граф сети с удалённым узлом

Найдем теперь кратчайшие пути до каждого из узлов. Результаты представлены в `/path/ring_modified_ospf.txt`. В листинге 4 представлены пути для некоторых узлов.

```
Start node 0:
path 0 -> 0: [0]
path 0 -> 1: [0, 1]
path 0 -> 2: [0, 1, 2]
path 0 -> 4: [0, 9, 8, 7, 6, 5, 4]
path 0 -> 5: [0, 9, 8, 7, 6, 5]
path 0 -> 6: [0, 9, 8, 7, 6]
path 0 -> 7: [0, 9, 8, 7]
path 0 -> 8: [0, 9, 8]
path 0 -> 9: [0, 9]
----------------------------------------
Start node 1:
path 1 -> 0: [1, 0]
path 1 -> 1: [1]
path 1 -> 2: [1, 2]
path 1 -> 4: [1, 0, 9, 8, 7, 6, 5, 4]
path 1 -> 5: [1, 0, 9, 8, 7, 6, 5]
path 1 -> 6: [1, 0, 9, 8, 7, 6]
path 1 -> 7: [1, 0, 9, 8, 7]
path 1 -> 8: [1, 0, 9, 8]
```

```
path 1 -> 9: [1, 0, 9]
----------------------------------------
Start node 2:
path 2 -> 0: [2, 1, 0]
path 2 -> 1: [2, 1]
path 2 -> 2: [2]
path 2 -> 4: [2, 1, 0, 9, 8, 7, 6, 5, 4]
path 2 -> 5: [2, 1, 0, 9, 8, 7, 6, 5]
path 2 -> 6: [2, 1, 0, 9, 8, 7, 6]
path 2 -> 7: [2, 1, 0, 9, 8, 7]
path 2 -> 8: [2, 1, 0, 9, 8]
path 2 -> 9: [2, 1, 0, 9]
----------------------------------------
```

Листинг 4: Содержимое файла ring_modified_ospf.txt

2.3 Звёздная топология

Граф сети с звёздной топологией приведён на рисунке 5.

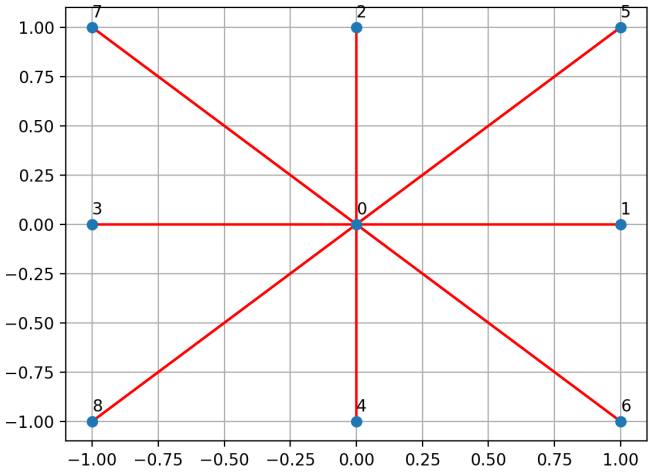

Рис. 5: Граф сети с звёздной топологией

С помощью алгоритма Дейкстры найдем кратчайшие пути от каждого узла. Результаты поиска находятся в `/path/star_ospf.txt`.

```
Start node 0:
path 0 -> 0: [0]
path 0 -> 1: [0, 1]
path 0 -> 2: [0, 2]
path 0 -> 3: [0, 3]
path 0 -> 4: [0, 4]
```

```
path 0 -> 5: [0, 5]
path 0 -> 6: [0, 6]
path 0 -> 7: [0, 7]
path 0 -> 8: [0, 8]
----------------------------------------
Start node 1:
path 1 -> 0: [1, 0]
path 1 -> 1: [1]
path 1 -> 2: [1, 0, 2]
path 1 -> 3: [1, 0, 3]
path 1 -> 4: [1, 0, 4]
path 1 -> 5: [1, 0, 5]
path 1 -> 6: [1, 0, 6]
path 1 -> 7: [1, 0, 7]
path 1 -> 8: [1, 0, 8]
----------------------------------------
Start node 2:
path 2 -> 0: [2, 0]
path 2 -> 1: [2, 0, 1]
path 2 -> 2: [2]
path 2 -> 3: [2, 0, 3]
path 2 -> 4: [2, 0, 4]
path 2 -> 5: [2, 0, 5]
path 2 -> 6: [2, 0, 6]
path 2 -> 7: [2, 0, 7]
path 2 -> 8: [2, 0, 8]
----------------------------------------
Start node 3:
path 3 -> 0: [3, 0]
path 3 -> 1: [3, 0, 1]
```

Листинг 5: Содержимое файла star_ospf.txt

Пути для всех узлов приведены в файле `/path/star_ospf.txt`.

Теперь удалим один узел из сети (например, узел 3). Тогда граф будет выглядеть следующим образом (рисунок 6).

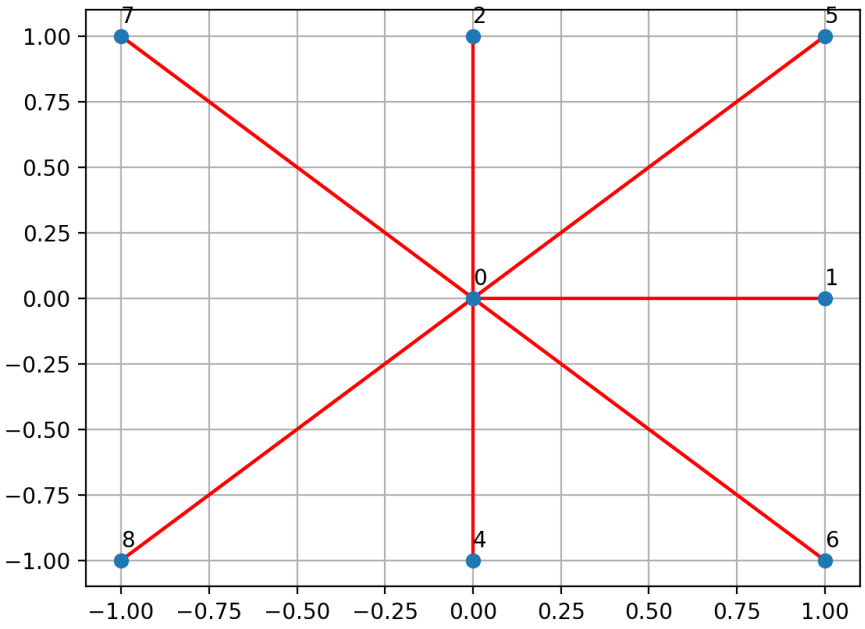

Рис. 6: Граф сети с удалённым узлом

Найдем теперь кратчайшие пути до каждого из узлов. Результаты представлены в `/path/star_modified_ospf.txt`. В листинге 6 представлены пути для некоторых узлов.

```
Start node 0:
path 0 -> 0: [0]
path 0 -> 1: [0, 1]
path 0 -> 2: [0, 2]
path 0 -> 4: [0, 4]
path 0 -> 5: [0, 5]
path 0 -> 6: [0, 6]
path 0 -> 7: [0, 7]
path 0 -> 8: [0, 8]
----------------------------------------
Start node 1:
path 1 -> 0: [1, 0]
path 1 -> 1: [1]
path 1 -> 2: [1, 0, 2]
path 1 -> 4: [1, 0, 4]
path 1 -> 5: [1, 0, 5]
path 1 -> 6: [1, 0, 6]
path 1 -> 7: [1, 0, 7]
path 1 -> 8: [1, 0, 8]
----------------------------------------
```

```
Start node 2:
path 2 -> 0: [2, 0]
path 2 -> 1: [2, 0, 1]
path 2 -> 2: [2]
path 2 -> 4: [2, 0, 4]
path 2 -> 5: [2, 0, 5]
path 2 -> 6: [2, 0, 6]
path 2 -> 7: [2, 0, 7]
path 2 -> 8: [2, 0, 8]
----------------------------------------
Start node 3:
path 3 -> 3: [3]
----------------------------------------
```

Листинг 6: Содержимое файла star_modified_ospf.txt

3 Выводы

На основе проведённого анализа можно сделать следующие выводы:

- Сеть с линейной топологией демонстрирует высокую уязвимость к сбоям. Потеря даже одного узла может привести к утрате связи между остальными участками сети, что значительно ухудшает её работоспособность.
- Кольцевая топология более устойчива к потерям узлов, так как в случае сбоя одного узла трафик может быть перенаправлен по обходному пути. Однако, если исчезает больше одного узла, кольцо разрывается, и сеть теряет свою функциональность.
- Сеть со звёздной топологией отличается хорошей стабильностью при отказах узлов, за исключением центрального узла. Потеря центрального узла приводит к полной потере связи между всеми остальными узлами, что делает сеть неработоспособной.

4 Код и ресурсы

Код проекта доступен в публичном репозитории на GitHub.
https://github.com/IlyaZawyalow/networks/tree/main/Lab2