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
Чепулис М.А.

Преподаватель
Баженов А.Н.

Санкт-Петербург
2021 год

Постановка задачи

Требуется разработать систему из неограниченного количества взаимодействующих друг с другом маршрутизаторов, которые организуются в сеть и обеспечивают передачу сообщений от каждого маршрутизатора к каждому по кратчайшему пути.

Необходимо рассмотреть:

Три вида топологии сети: линейная, кольцо, звезда.

Перестройку таблиц достижимости при стохастических разрывах связи.

Реализация

Система реализована на языке программирования Python.

Топология связей роутеров представляется в виде орграфа. Веса всех рёбер графа равны единице. Также выделен отдельный роутер (DR – designated router), который находится вне топологии и обеспечивает маршрутизацию сообщений об изменениях в топологии.

В данной работе мы абстрагируемся от типа реализации канала связи. Важно, чтобы сообщения приходили в том же порядке, в каком и отправляются (например, достаточно использовать протокол связи Go-Back-N).

Для подключения нового роутера к сети:

1) Роутер устанавливает связь с DR
2) Роутер отправляет DR сообщение с информацией о соседях
3) Роутер запрашивает у DR текущую топологию сети.

Designated Router (DR) – связан со всеми узлами одновременно.

Когда DR получает сообщение о подключении / отключении узла, то он обновляет свою топологию, после чего отправляет всем узлам (кроме подключаемого / отключаемого) сообщения об изменении топологии.

Так как ключевая задача этой работы – протокол маршрутизации, то мы рассмотрим только сообщения имеющие отношения к топологии.

Возможные типы сообщений:

Для Designated Router

NEIGHBORS ([neighbors]) – запрос на добавление в топологию новых соседей. Номер узла (чью соседи) определяется номером отправителя

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.

PRINT_WAYS – запрос о выводе на печать текущих кратчайших путей до всех узлов. Это сообщение не влияет на топологию.

Так как топология представлена в виде орграфа (для моделирования схемы – кольцо ребра должны быть направленными), то Роутер, при получении сообщения о добавлении нового узла. Проверяет, является ли новый узел его соседом, если да, то посылает DR сообщение о добавлении нового соседа (если только, данного соседства ещё нет в топологии)

Все роутеры, в том числе и DR запускаются в отдельных потоках выполнения.

Пример работы программы

Рассмотрим пример работы для линейной топологии на примере сети с пятью узлами.

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

```
dr(0): (NEIGHBORS, {neighbors(0)})
dr(0): (GET_TOPOLOGY)
dr(1): (NEIGHBORS, {neighbors(1)}))
r(0)  :(SET_TOPOLOGY)
dr(1): (GET_TOPOLOGY)
r(0) : (NEIGHBORS: {1, neighbors(1)})
r(3) : (NEIGHBORS: {1, neighbors(1)}))
dr(2): (NEIGHBORS:{neighbors(2)}))
r(2) : (NEIGHBORS: {1, neighbors(1)}))
dr(3): (NEIGHBORS: {neighbors(3)}))
r(1) : (SET_TOPOLOGY)
r(3) : (NEIGHBORS: {2, neighbors(2)})
r(0) : (NEIGHBORS: {2, neighbors(2)})
r(4) : (NEIGHBORS: {2, neighbors(2)})
r(1) : (NEIGHBORS: {2, neighbors(2)})
dr(2): (GET_TOPOLOGY)
r(0) : (NEIGHBORS: {3, neighbors(3)})
r(4) : (NEIGHBORS: {3, neighbors(3)})
```

dr(3): (GET_TOPOLOGY: None)
r(2) : (NEIGHBORS: {3, neighbors(3)})
r(1) : (NEIGHBORS: {3, neighbors(3)})
dr(4): (NEIGHBORS)
r(3) : (SET_TOPOLOGY)
dr(4): (GET_TOPOLOGY)
r(0) : (NEIGHBORS: {4, neighbors(4)})
r(3) : (NEIGHBORS: {4, neighbors(4)})
r(2) : (SET_TOPOLOGY)
r(1) : (NEIGHBORS: {4, neighbors(4)})
r(4) : (SET_TOPOLOGY)
r(2) : (NEIGHBORS: {4, neighbors(4)})

Все 5 узлов подключились к сети. Посмотрим на 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]]

Предположим, что отключился нулевой узел:

dr(0): (OFF: None)
r(2) : (OFF: 0)
r(1) : (OFF: 0)
r(4) : (OFF: 0)
r(3) : (OFF: 0)

Тогда новые кратчайшие пути:

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

Видим, что нулевой узел ни с кем не связан. Пусть нулевой узел снова восстановил связь:

dr(0): (NEIGHBORS: {neighbors(0)})
dr(0): (GET_TOPOLOGY)
r(1) : (NEIGHBORS: {0, neighbors(0)})
r(4) : (NEIGHBORS: {0, neighbors(0)})
r(0) : (SET_TOPOLOGY)
r(2) : (NEIGHBORS: {0, neighbors(0)})
r(3) : (NEIGHBORS: {0, neighbors(0)})

Далее восстанавливается связь $1 \rightarrow 0$

dr(1): (NEIGHBORS: [0])
r(2) : (NEIGHBORS: {1, [0]})
r(3) : (NEIGHBORS: {1, [0]})
r(4) : (NEIGHBORS: {1, [0]})
r(0) : (NEIGHBORS: {1, [0]})

А кратчайшие пути вернулись в состояние до отключения:

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]]

Притом, заметим, что в начале, нулевой узел подключился самом первым. Других узлов в сети ещё не существовали. Потому информация о соседях нулевого узла была отправлена только DR. При повторном подключении, пришлось разослать её всем роутерам.

Аналогично можно построить и другие топологии. Отличие будет только в определении соседей для каждого узла:

Кольцо:

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

Минимальные пути:

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

После отключения третьего узла:

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

Звезда с центром во втором узле:

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

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]]
```

После отключения второго (центрального) узла не остались ни каких связей:

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

Вместо центрального (второго) отключился третий узел:

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

Результаты

Была реализована программа для моделирования протокола динамической маршрутизации OSPF для неограниченного количества взаимодействующих друг с другом маршрутизаторов и стохастическими разрывами соединения.

Данная программа была проверена на трёх топологиях, из чего был сделал вывод о её корректной работе на топологиях: линейная, кольцо звезда

Приложение:

Ссылка на проект с кодом реализации:
https://github.com/MChepulis/CompNetworks-Labs/tree/main/Lab_2

Использованная литература

1. А.Н. Баженов, Компьютерные сети, курс лекций
2. Мануилов Г. Реализация протокола динамической маршрутизации Open Shortest Path First.