Построение сети управления

для SDN сети

1. Постановка задачи

Пусть задана топология сети графом G(V,E) в формате GraphML или gml, где каждый узел сети задается географическими координатами (Рисунок 1). Примеры реальных сетей и их описаний в этих форматах можно найти в библиотеке Topology Zoo (http://www.topology-zoo.org/dataset.html).

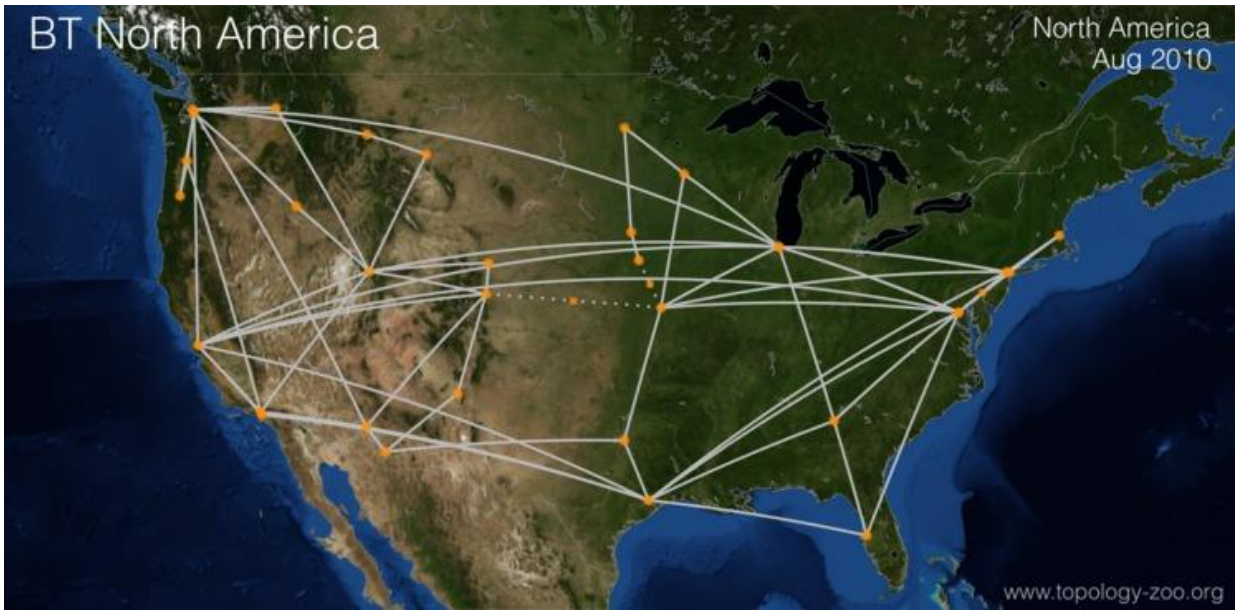

Рисунок 1. Пример топологии сети компании AT&T из библиотеки Topology Zoo (AttMpls.gml, AttMpls.GraphML)

Предварительно необходимо выполнить обработку исходных данных:

1. Вычислить расстояния между узлами сети (в км.), зная их географические координаты.
2. Для каждого канала связи вычислить задержку, предполагая, что используется оптоволокно, в котором задержка на 1 км — 4,8 мкс или 0,0048 мс. Задержка будет являться весом ребра в графе сети.

Предположим, что данную сеть решили модернизировать и преобразовать ее в SDN сеть. В связи с чем, возникает проблема выбора узла сети, к которому непосредственно будет подключен SDN контроллер. В SDN сети между каждым коммутатором и контроллером устанавливается специальное защищенное логическое соединение для обеспечения управления коммутаторами (Рисунок 2). Все множество соединенных защищенных соединений образуют сеть управления коммутаторами и является остовным деревом графа.

Под ***остовным деревом*** графа понимается дерево (подграф данного графа), с тем же числом вершин, что и у исходного графа. Неформально говоря, остовное дерево получается из исходного графа удалением максимального числа рёбер, входящих в циклы, но без нарушения связности графа. Остовное дерево включает в себя все n вершин исходного графа и содержит (n-1) ребро.

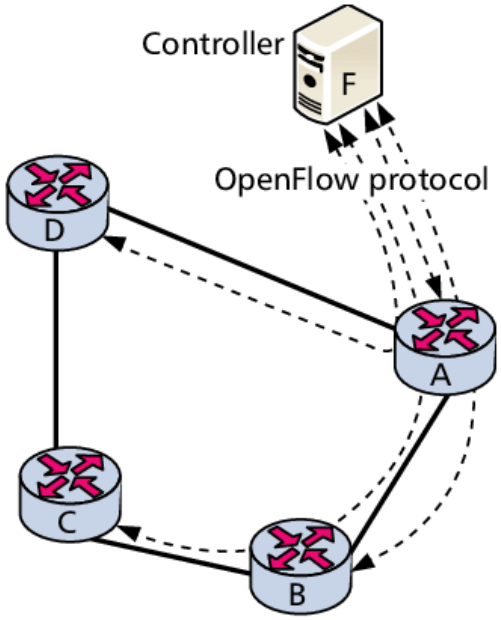

Рисунок 2. Выбранный узел сети для подключения контроллера и сеть управления (остовное дерево)

В сети из N узлов соответственно существует N вариантов выбора размещения контроллера в сети и множество вариантов построения остовного дерева в сети управления, поэтому желательно выбрать оптимальное размещение контроллера и остовное дерево для сети управления.

Выбор узла подключения контроллера и остовного дерева (для сети управления) из этого выбранного узла-контроллера можно выполнять по одному из следующих критериев:

K1: Минимизировать максимальную задержку маршрутов от узла-контроллера до любого из коммутаторов.

K2+K1: Минимизировать сумму ребер остовного дерева (то есть построить минимальное остовное дерево в графе – Рисунок 3). Затем выбрать место размещения контроллера в рамках данного остовного дерева по критерию K1.

Для выбора узла-контроллера и остовного дерева по критерию K1 можно использовать алгоритм Дейкстры [1,2], который позволяет найти кратчайшие пути от одной из вершин графа до всех остальных (и построить остовное дерево соответственно для сети управления). Для этого необходимо для N узлов сети, N раз запустить алгоритм Дейкстры, для каждого запуска определить задержку на самом длинном пути (определить максимальную задержку). И в итоге выбрать тот узел для контроллера и остовное дерево, у которого максимальная задержка будет наименьшей.

Для выбора узла-контроллера и остовного дерева по критерию K2+K1 сначала необходимо выбрать минимальное остовное дерево для данной сети. Минимальным остовным деревом называется остовное дерево этого графа, имеющее минимальный возможный вес (задержку), где под весом дерева понимается сумма весов (задержек) входящих в него рёбер (каналов связи). Пример остовного дерева приведен на рисунке.

Для построения минимального остовного дерева можно использовать любой понравившийся/понятный Вам алгоритм из [2] (алгоритм Прима, алгоритм Краскала (или алгоритм Крускала), алгоритм Борувки или алгоритм обратного удаления). В комментариях в коде указать, какой алгоритм используете.

Затем в рамках минимального остовного дерева (используя только его каналы связи) по алгоритму Дейкстры (критерий K1) уже выбрать место размещения контроллера (узел подключения контроллера).

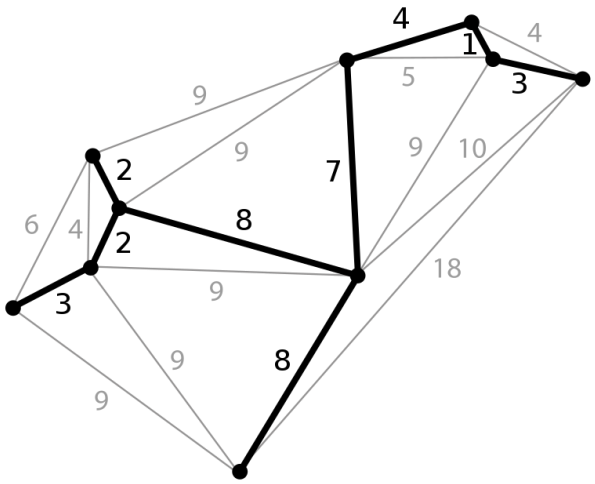

Рисунок 3. Пример построения минимального остовного дерева в графе

2. Этапы выполнения задания

1. Выполнить парсинг топологии из GrahpML (или gml, на Ваш выбор) файла.
2. Вычислить расстояние между координатами точек, заданными долготой и широтой (в комментариях описать, по какому принципу осуществлялось преобразование или ссылки).
3. Предполагая, что вес каналов связи пропорционален расстоянию и в качестве среды передачи данных используются оптоволоконные кабели. Оптоволоконные кабели имеют задержку на 1 км — 4,8 мкс или 0,0048 мс. Рассчитать веса для каждого канала связи.
4. Сформировать первый CSV файл с описанием каналов связи сети в порядке возрастания id узлов.
5. Реализовать решение задачи выбора размещения контроллера в сети и построения остовного дерева по критерию K1.

6. Реализовать решение задачи выбора размещения контроллера в сети и построения остовного дерева по критерию K2+K1.
7. Сформировать второй CSV файл с описанием остовного дерева.

3. Формат входных данных

Предусмотреть следующие опции запуска программы

- **-t <имя файла топологии>** – задание топологии сети из библиотеки TopologyZoo
- **-k** <номе критерия> – опция выбора критерия для выбора места размещения контроллера и построения остовного дерева для сети управления (например, -k 1 или –k 2)

Примеры запуска программы:

./yourApp –t **AttMpls.GraphML –k 1**

Команда означает выбор размещения контроллера в сети по критерию K1 для сети с топологией **AttMpls.GraphML.**

./yourApp –t **AttMpls.GraphML –k 2**

Команда означает выбор размещения контроллера в сети по критерию K2+K1 для сети с топологией **AttMpls.GraphML.**

.4. Формат выходных данных

1. Первый CSV файл с описанием топологии (узлов и каналов связи) сети в порядке возрастания id узлов.

формат названия файла**: <имя топологии>_topo.csv**

Пример:

Node 1 (id)	**Node 1 (label)**	**Node 1 (longitude)**	**Node 1 (latitude)**	**Node 2 (id)**	**Node 2 (label)**	**Node 2 (longitude)**	**Node 2 (latitude)**	**Distance (km)**	**Delay (mks)**
1	Moscow	56.345	23.456	2	London	10.345	15.456	1000	4830
1				3					
1				5					
2	London	10.345	15.456	1					
2				8					

2. Второй CSV файл, содержащий описание остовного дерева.

формат названия файла**: <имя топологии>_routes.csv**

Пример (с контроллером в узле 1):

Node 1 (id)	**Node 2 (id)**	Path type	Path	Delay (mks)	
1	2	main	[1, 3, 5, 2]	2000	
1	2	reserv	[1, 4, 6, 7, 2]	2400	
1	3	main	[1,3]	200	
1	3	reserve	no		
…	…				

5. Задания и система оценки*

№	Критерий	Количество баллов
1	**Базовая часть (обязательная):** − Корректный парсинг GraphML(или gml) файла − Перевод расстояний в км и в задержку − Поддержка опции выбора размещения контроллера и построения остовного дерева для сети управления по критерию K1. − Корректный CSV файл с топологией − Корректный CSV файл с описание остовного дерева сети управления.	4 баллов
2	**Дополнительное задание 1** − Поддержка опции построения минимального остовного дерева и выбора размещения контроллера для сети управления по критерию K2+K1.	3 балла
3	**Дополнительное задание 2** − Визуализация графа с возможностью отображения узла с контроллером (другим цветом) и остовного дерева (сети управления)	2 балла

** За несамостоятельное выполнение работы или копирование кода из открытых источников оценка 0.*

6. Требования к программной реализации

1. Программа должна быть реализована на одном из следующих языков программирования: Python, C++ или Go.
2. Программа должна работать в среде Linux (ОС Ubuntu версии 20.04.4 LTS и выше).
3. Текст программы должен быть хорошо читаем, структурирован и оформлен в соответствии с правилами оформления программ на соответствующем языке программирования, а также содержать комментарии.
4. Можно использовать библиотеки, реализующие парсинг GRAPHML или GML форматов.
5. Можно использовать любые библиотеки поддержки визуализации.

6. Программа должна корректно обрабатывать все возможные ситуации ошибок и некорректного пользовательского ввода.
7. Не выкладывать программу в github или другие публичные репозитории.
8. Приветствуется использование графовых баз данных для хранения и работы с графами (список можно посмотреть, например, здесь: "Википедия: графовая база данных"), но не является обязательным.

7. Отправка работ

В письмо нужно приложить архив (формат .tar.bz2) с реализацией. В архиве должен присутствовать README.md файл содержащий:

1. ФИО, номер группы.
2. Список библиотек, которые используете в реализации.
3. Описанием, как собирать (с списком зависимостей, которые нужно предварительно установить).
4. Описание, как запускать вашу программу, как ей пользоваться.
5. В папку tests положить результаты работы вашей программы на одной или нескольких топологиях (т.е. GRAPHML или GML файл топологии и соответствующие CSV файлы).
6. Тестирование будет проводиться в ОС Ubuntu версии 20.04.4 LTS и выше.

Тема письма:

[номер группы] – SPT – Фамилия Имя

Пример: **[217] – SPT – Иванов Иван**

Адрес для отправки работ: asvk-nabor@asvk.cs.msu.ru

Список литературы

1. Алгоритм Дейкстры https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B

2. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. **Глава 23. Минимальные остовные деревья** // Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд. — М.: Вильямс, 2005. — 1296 с