Санкт-Петербургский политехнический университет Петра Великого

Физико-механический институт

Высшая школа прикладной математики и вычислительной физики

Компьютерные сети
Отчёт по лабораторной работе №3
“Симуляция работы децентрализованного хранилища файлов”

Выполнил:

Студент: Сачук Александр
Группа: 5040102/30201

Принял:

к. ф.-м. н., доцент
Баженов Александр Николаевич

2025 г.

Содержание

1. Постановка задачи

Требуется реализовать симуляцию работы децентрализованного хранилища данных: каждый элемент (компьютер) P2P-сети содержит часть некоторого файла. По команде сборки файла для некоторого компьютера сети части файлов передаются по кратчайшим путям посредством каналов связи.

2. Теория

На сегодняшний день одним из основных методов хранения данных является централизованное хранение - хранение при помощи одного выделенного сервера. Такое хранение позволяет легко получать данные, однако имеет ряд изъянов: данные и процессы на этих компьютерах не являются прозрачными, а атака на центральный сервер может нарушить конфиденциальность.

В качестве альтернативы общественность предлагает использовать децентрализованные хранилища (такая идея входит в парадигму Web3.0). Основная идея такого метода состоит в хранении частей файлов на нескольких нодах (узлах) P2P сети. Такой подход усложняет процедуру получения файла ввиду дополнительной загрузки, однако обеспечивает защищенность: взлом одного из узлов не гарантирует утечку полного содержимого файла. В данной работе необходимо реализовать симуляцию работы децентрализованного хранилища (хранения и сборки файлов для некоторого узла сети) и продемонстрировать работу для трех различных топологий: линейной, круговой и звездочной.

В качестве симуляции работы будем представлять одноранговую сеть как некоторый неориентированный граф. Каждый элемент этого графа содержит части исходного файла. Число частей файлов обычно формируется исходя из разных желаемых факторов.

Помимо узлов в P2P-сети с фрагментами файлов существует специальный Designated router (DR), который "дирижирует"остальными роутерами: непосредственного участия в передаче самих фрагментов файла он не принимает. В функционал данного роутера входят следующие исполняемые задачи:

- Знакомство роутеров со всеми остальными роутерами сети;
- Координирование запросов от роутера ко всем остальным роутерам;
- Организация процедуры распределения частей исходного файла по узлам;

Будем считать, что каждый узел сети имеет отказоустойчивое соединение с DR, однако сами узлы могут иметь нестабильное соединение со своими соседями. Потому сами фрагменты файлов между узлами будут распространяться согласно с протоколами повторной отправки данных. В данной работе реализованы два протокола: Selective repeat (SRP) и Go-back-N (GBN) протоколы.

Фрагменты файлов передаются от узла к узлу по кратчайшему маршруту, используя протокол Open Shortest Path First (OSPF).

Таким образом общий процесс симуляции работы децентрализованного хранилища будет следующим:

1. Инициализация топологии сети, загрузка файла, разбиение его на части и распределение частей между узлами сети.
2. Построение карты сети для каждого роутера и кратчайщих путей посредством протокола OSPF.
3. Построение между узлами каналов связи, обеспечивающие повторную отправку.
4. Запуск алгоритма загрузки (сборки файла) для некоторого узла:
   (a) Получение необходимой информации о частях файлов с DR.
   (b) Независимая параллельная передача всех частей файлов с разных роутеров сети посредством передачи по каналам связи к роутеру, на котором строится файл.
   (c) Сравнение построенного файла с исходным.

Промежуточные данные такие как топология сети, результаты разбиения исходного файла на части и сохранение для каждого узла, результат сборки логируются и представляются в качестве результата работы.

3. Реализация

Симуляция системы реализована на языке программирования Python. Исходный код доступен по следующей ссылке:
https://github.com/AS2/comp_networks/tree/main/lab3

Для создания симуляции работы протокола маршрутизации реализованы следующие основные абстракции:

- LossyQueue (файл lossy_queue.py) - класс реализации очереди с потерями. Потери реализуются посредством "отказа"от добавления нового элемента с некоторой вероятностью;

- IResendBehaviour (файл resend_protocol_base.py) - базовый класс для реализации канала связи с повторной отправкой при потере между двумя узлами;

- GBN (файл GBN.py) - спецификация IResendBehaviour, реализующую протокол Go-Back-N;

- SP (файл SP.py) - спецификация IResendBehaviour, реализующую протокол SelectiveRepeat;

- Connection: пара очередей, для реализации взаимодействия, вместе с реализованной политикой IResendBehaviour.

- Topology (файл topology.py) - графовое преставление сети. На основе топологии строится итоговая P2P сеть в симуляции;

- Router (файл network.py) - базовый элемент сети, выполняющий основные элементы симуляции: инициализацию, отправка информации о соседях и её получение. Информация о соседях распространяется в рамках данной симуляции отказоустойчиво;

- Designated router (файл network.py) - выделенный маршрутизатор. Его функции описаны раннее;

- FileMetadata (файл primitives.py) - структура описания файла и процесса его разбиения;

- FilePart (файл primitives.py) - структура, содержащая часть файла. Состоит из порядкового номера, размера, а также непосредственно содержания;

Симуляция реализована на ЯП Python 3.10, где каждый элемент сети работает параллельно: в течении 3 секунд запускаются узлы, после чего запускается построение кратчайших путей и постройка каналов связи, согласно топологии, а затем сборка файлов.

4. Результаты

В данном блоке приведены примеры работы сети с протоколом повторной отправки GBN, вероятности потери 0.2, а также с окном, равному 5. Исходные данные делились на 63 фрагмента.

4.1. Топология “Линейная”

Линейная топология представляет из себя 7 узлов, соединенных в одну линию шестью ребрами. Формально выглядит так:

- Вершины: [0, 1, 2, 3, 4, 5, 6]
- Соседи: [[1], [0, 2], [1, 3], [2, 4], [3, 5], [4, 6], [5]]

После разбиения тестового файла на фрагменты и загрузки фрагментов на данные 7 узлов, можно увидеть такое распределение частей, как на 1. Как видно, сами фрагменты на каждом отдельном роутере дают мало понимания о том, что за файл хранится на роутерах. Рассмотрим результат работы при сборке фрагментов для крайнего (нулевого) и центрального (третьего) узлов.

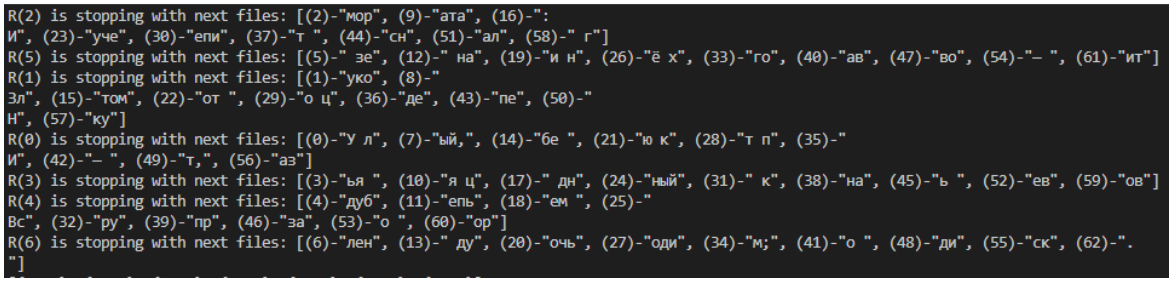

Рис. 1. Разбиение файла на фрагменты и их распределения на роутерах

Как видно из рисунков 2 и **??** - за этими фрагментами прятались вполне известные строчки вполне известного произведения. Однако видно еще и то, что время работы для центрального узла меньше почти в 2 раза: это можно попытаться обяснить тем, что суммарный путь пройденный пакетами во втором случае больше, чем для первого. А чем больше путь, тем больше пересылок по каналам связи с потерями было осуществлено. Однако, для более точных выводов, нужно производить большую серию замеров и усреднение.

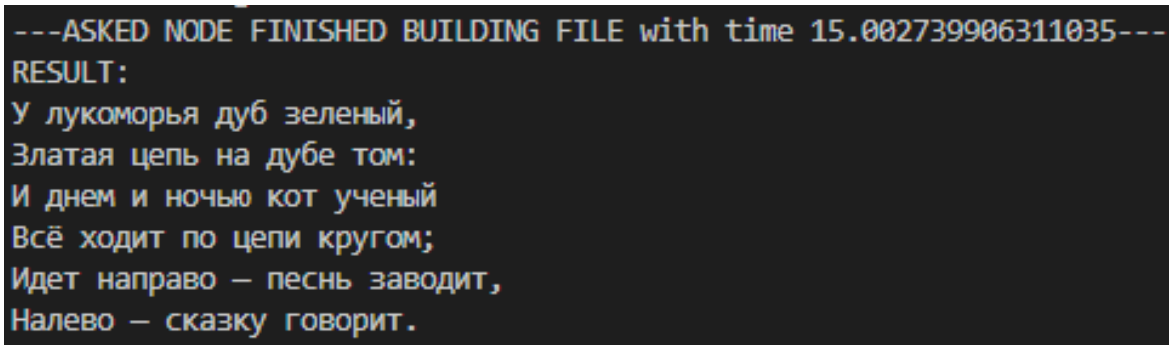

Рис. 2. Результат сборки файла для нулевого (крайнего) узла прямой

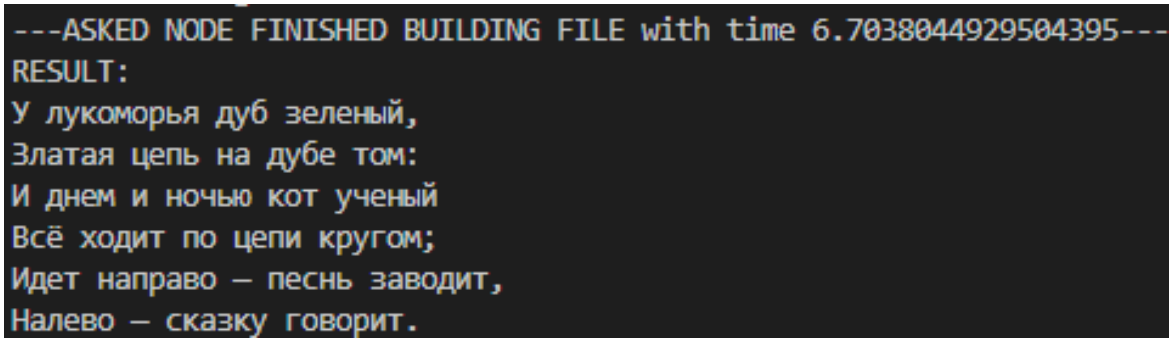

Рис. 3. Результат сборки файла для третьего (центрального) узла прямой

4.2. Топология “Звезда”

В данном случае была рассмотрена следующая топология:

- Вершины: [0, 1, 2, 3, 4]
- Соседи: [[1, 2, 3, 4], [0], [0], [0], [0]]

Как видно, нулевой узел здесь является центральным, а все остальные - “лучами” получаемой сети.

Рассмотрим содержимое роутеров после разбиения того же файла на фрагменты и их распределения по узлам. На рис. 4 видны те же фрагменты, однако для каждый роутер хранит большее число фрагментов, чем в предыдущем случае. Рассмотрим результаты.

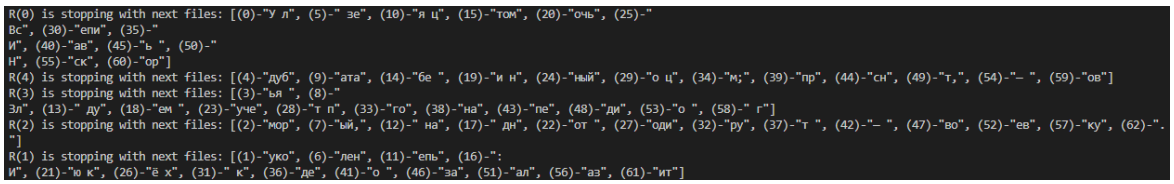

Рис. 4. Разбиение файла на фрагменты и их распределения на роутерах для звезды

Рисунки 6 и 5 отображают тот же результат, только другие времена, в сравнении с предыдущим опытом.

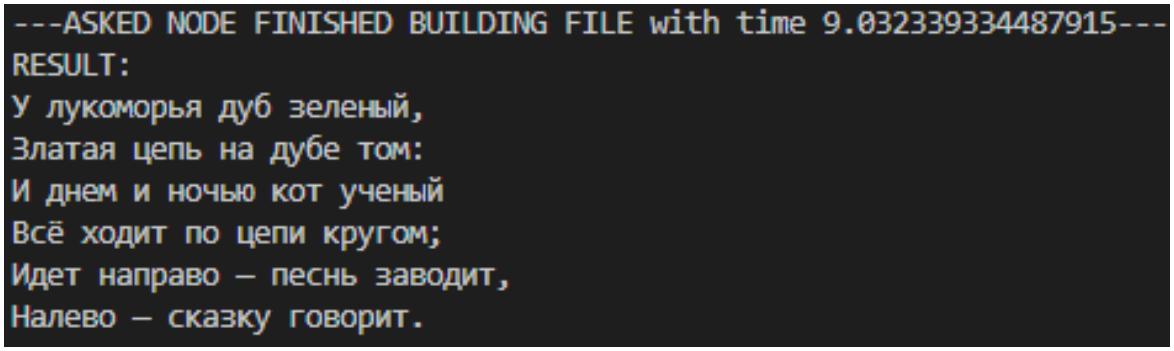

Рис. 5. Результат сборки файла для нулевого (центрального) узла звезды

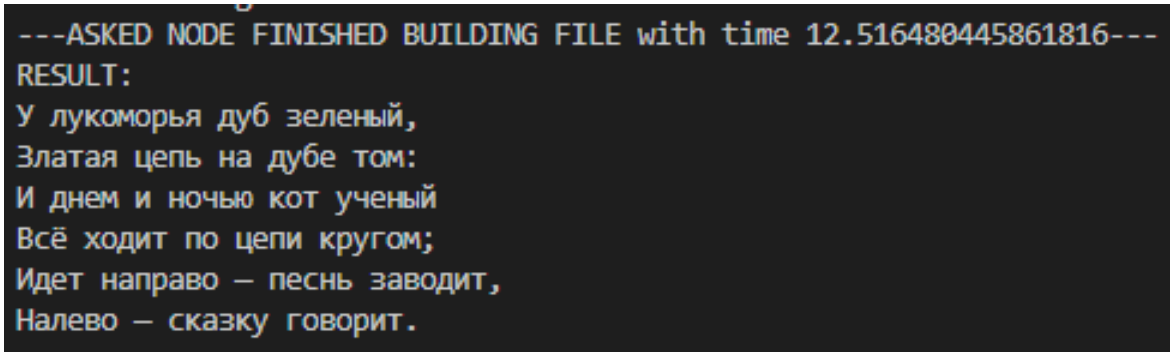

Рис. 6. Результат сборки файла для первого (лучевого) узла звезды

4.3. Топология “Кольцо”

В данном случае была рассмотрена следующая топология:

- Вершины: [0, 1, 2, 3, 4, 5, 6]
- Соседи: [[6, 1], [0, 2], [1, 3], [2, 4], [3, 5], [4, 6], [5, 0]]

Все узлы имеют два соседа с индексами, отличающихся на 1 - данная топология является кольцом. В данном случае достаточно рассмотреть 1 случай работы для произвольного узла.

Содержимое роутеров после разбиения файла на фрагменты и распределения по узлам представлена на рис. 4. Рассмотрим результаты сборки.

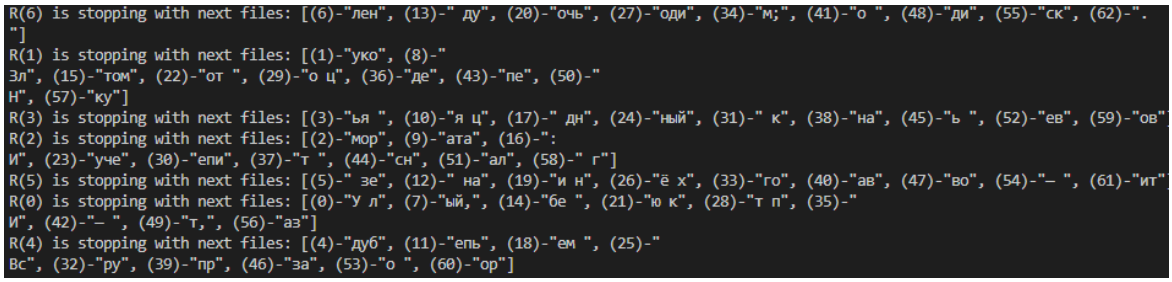

Рис. 7. Разбиение файла на фрагменты и их распределения на роутерах для кольца

Рисунок 8 отображает корректность работы симуляции.

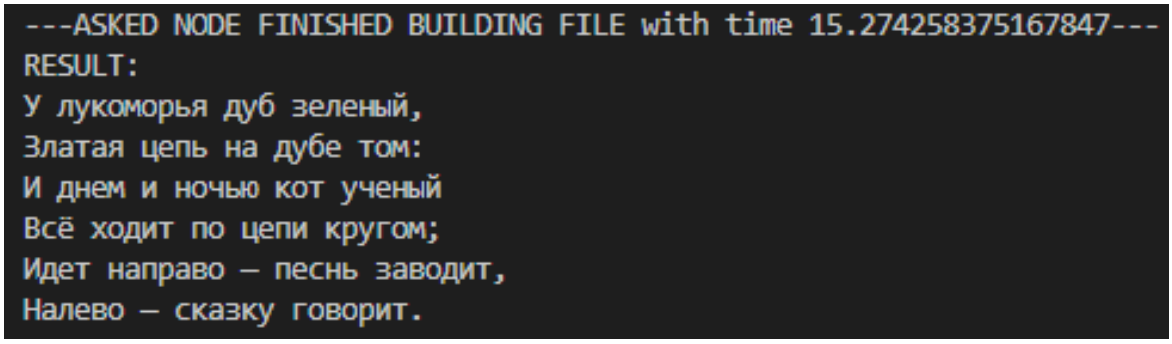

Рис. 8. Результат сборки файла для нулевого (произвольного) узла кольца

5. Обсуждение

В рамках данной работы была реализована симуляция работы децентрализованного хранилища файлов. Как было продемонстрировано в блоке результатов, работоспособность симуляции была продемонстрирована на трех различных топология: звезды, кольца и линейной. Отправка сообщений и передача файлов между роутерами была реализована при помощи одного класса (Connection), описывающих поведение каналов. Для обеспечения корректной работы параллельных алгоритмов, использовались различные примитивы синхронизации.