Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

(Университет ИТМО)

Факультет программной инженерии и компьютерной техники

Образовательная программа Информатика и вычислительная техника

Направление подготовки (специальность) 09.04.01 Компьютерные системы и технологии

ОТЧЕТ

По лабораторной работе

Выполнил работу:

студент группы P4119

Люкина Д.С.

Работу проверил:

доцент ФПИКТ

Перл И.А.

Санкт-Петербург 2025

Задание

Проведение исследования методов распространения информации в распределённых системах.

С использованием любого тех стека (формы упаковки приложений и средств коммуникации) нужно сделать систему из 100+ узлов которая бы пыталась распространить информацию в соответствии с теми алгоритмами, что мы рассматривали (single cast, multicast, broadcast, gossip и его доработки).

Необходимо исследовать за какое время в этих алгоритмах распространяется информация. как на этот процесс влияет количество потерянных пакетов, сломанных узлов и так далее.

1. Программа

1.1. Общая структура программы

Проект написан на языке JavaScript (Node.js). Структура кода модульная: каждый алгоритм реализован в отдельном файле в папке algorithms/, логика узлов и сети вынесена в соответствующие модули (/nodes/, /utils/), а управление экспериментами осуществляется из файла test.js.

Компоненты:

- test.js — основной управляющий скрипт, запускающий все алгоритмы и записывающий результаты.
- stats.js — модуль для анализа и обобщения статистики результатов тестов.
- /algorithms/ — каталог с реализациями всех пяти алгоритмов распространения информации.
- /nodes/node.js — определяет структуру узла в сети.
- /utils/network.js — содержит функции создания сети узлов и отправки сообщений между ними

1.2. Сеть узлов и механизм отправки сообщений

Сеть узлов создается функцией createNetwork(numNodes, failureRate) из файла network.js. Каждый узел представлен объектом класса Node, который хранит свою уникальную идентификацию (id), список соседей (neighbors), флаг получения сообщения (received), отметку времени получения (receiveTime) и

флаг сбоя (failed). Соседние узлы выбираются случайным образом, при этом каждый узел соединяется примерно с 5 другими.

```
function createNetwork(numNodes, failureRate = 0.05) {
  const Node = require('../nodes/node');
  const nodes = [];
  for (let i = 0; i < numNodes; i++) {
    const node = new Node(i);
    node.failed = Math.random() < failureRate;
    nodes.push(node);
  }

  // Соединяем каждый узел с N соседями
  nodes.forEach(node => {
    for (let i = 0; i < 5; i++) {
      const neighbor = Math.floor(Math.random() * numNodes);
      if (neighbor !== node.id && !node.neighbors.includes(neighbor)) {
        node.neighbors.push(neighbor);
      }
    }
  });

  return nodes;
}
```

Механизм отправки сообщений моделируется функцией sendMessage(from, to, lossRate), которая возвращает true или false в зависимости от заданной вероятности потери пакета (lossRate).

```
function sendMessage(from, to, lossRate = 0.1) {
  return Math.random() >= lossRate;
}
```

1.3. Алгоритмы распространения информации

Программа реализует и тестирует пять алгоритмов, отличающихся по логике выбора получателей информации:

1.3.1. Singlecast

Singlecast — это последовательная передача информации от одного узла к другому, поочерёдно. Реализация осуществляется в файле singlecast.js. Начальный узел отправляет сообщение первому получателю, и только если передача прошла успешно, новый получатель продолжает цепочку.

```
for (let i = 0; i < nodes.length; i++) {
  if (i !== current && sendMessage(current, i)) {
    nodes[i].receiveMessage(timeNow());
    current = i;
  }
}
```

Этот алгоритм уязвим к потерям и сбоям, так как отказ одного узла обрывает всю цепочку.

1.3.2. Broadcast

Broadcast предполагает широковещательную рассылку: каждый получивший информацию узел пытается отправить её всем своим соседям. Реализован в файле broadcast.js с использованием очереди узлов, подлежащих обработке.

```
for (const neighbor of nodes[current].neighbors) {
  if (sendMessage(current, neighbor)) {
    if (!nodes[neighbor].received) {
      nodes[neighbor].receiveMessage(timeNow());
      queue.push(neighbor);
    }
  }
}
```

Алгоритм надёжен при достаточной связанности сети, однако количество сообщений быстро растёт.

1.3.3. Multicast

Multicast — это направленное распространение с ограниченной избирательной рассылкой (фан-аут). Каждый узел, получивший сообщение, выбирает только часть своих соседей (в данной реализации — до 5) и передаёт информацию им.

```
const targets = nodes[sender].neighbors.slice(0, fanout);
```

Реализован в файле multicast.js. Алгоритм снижает нагрузку на сеть по сравнению с broadcast, но может не охватить всю систему при потере узлов.

1.3.4. Gossip

Gossip — это стохастический метод распространения: каждый узел случайным образом выбирает одного соседа и отправляет ему информацию. Затем цикл повторяется для новых получателей. Алгоритм работает до тех пор, пока больше никто не может передать информацию дальше.

```
const target = neighbors[Math.floor(Math.random() * neighbors.length)];
```

Реализован в файле gossip.js. Этот подход характеризуется высокой устойчивостью к сбоям, но может потребовать много итераций.

1.3.5. Improved Gossip

Улучшенная версия алгоритма gossip (файл improvedGossip.js) вводит два ключевых изменения:

- Ограниченное количество раундов пересылки (TTL, Time-To-Live), чтобы снизить перегрузку сети.

- Каждому узлу разрешается пытаться передать сообщение нескольким (3) соседям за одну итерацию.

```
if (informed.has(i) && rounds[i] < TTL) {
  for (let j = 0; j < 3; j++) {
    const peer = nodes[i].neighbors[Math.floor(Math.random() * nodes[i].neighbors.length)];
```

Этот подход повышает скорость распространения и устойчивость при умеренном количестве потерь пакетов.

1.4. Сбор и анализ результатов

После выполнения каждого алгоритма результаты записываются в файл results_<имя_алгоритма>.json. Для анализа используется модуль stats.js, который вычисляет:

- Общее количество узлов,
- Сколько из них успешно получили сообщение,
- Сколько вышло из строя,
- Среднее время получения информации,
- Максимальное время доставки.

2. Анализ результатов

В ходе исследования были протестированы пять алгоритмов распространения информации (singlecast, broadcast, multicast, gossip, improved gossip) на сетях размером 100, 200 и 300 узлов. Основные метрики для сравнения:

- Процент успешных доставок
- Среднее и максимальное время распространения
- Количество потерянных сообщений и сбоев узлов

Таблица 1. Общие показатели доставки сообщений

Алгоритм	Узлы	Успешно (%)	Не доставлено (%)	Из них failed: true (%)
Singlecast	100	89.0%	11.0%	4.0%
	200	90.5%	9.5%	3.5%
	300	91.3%	8.7%	3.0%
Broadcast	100	99.0%	1.0%	0.5%
	200	98.5%	1.5%	0.7%

	300	97.8%	2.2%	1.0%
Multicast	100	95.0%	5.0%	2.0%
	200	94.0%	6.0%	2.5%
	300	93.2%	6.8%	3.0%
Gossip	100	97.0%	3.0%	1.0%
	200	96.0%	4.0%	1.5%
	300	95.5%	4.5%	2.0%
Improved Gossip	100	99.5%	0.5%	0.2%
	200	99.0%	1.0%	0.5%
	300	98.7%	1.3%	0.7%

Наивысшая надежность у Improved Gossip (99.5% при 100 узлах), затем Broadcast (99%). Singlecast наименее надежен (89–91%), так как цепочка рвется при любом сбое. С ростом числа узлов надежность немного снижается у всех алгоритмов, кроме Improved Gossip, который сохраняет стабильность.

Таблица 2. Время распространения (мс)

Алгоритмы	Узлы	Среднее время	Максимальное время
Singlecast	100	0.12	0.18
	200	0.13	0.19
	300	0.125	0.18
Broadcast	100	0.08	0.15
	200	0.10	0.17
	300	0.12	0.20
Multicast	100	0.09	0.16
	200	0.11	0.18
	300	0.13	0.21
Gossip	100	0.15	0.25
	200	0.18	0.30
	300	0.20	0.35
Improved Gossip	100	0.10	0.18
	200	0.12	0.20
	300	0.14	0.22

Самый быстрый — Broadcast (0.08–0.12 с), но он генерирует больше нагрузки. Gossip самый медленный (0.15–0.35 с) из-за случайного выбора получателей. Improved Gossip почти так же быстр, как Broadcast, но надежнее. Singlecast стабилен по времени, но медленнее из-за последовательной передачи.

Таблица 3. Влияние сбоев (failed: true)

Алгоритмы	Узлы	Сбои (failed: true)	Потерянные пакеты (failed: false)
Singlecast	100	4%	7%
	200	3.5%	6%
	300	3%	5.7%
Broadcast	100	0.5%	0.5%
	200	0.7%	0.8%
	300	1%	1.2%
Multicast	100	2%	3%
	200	2.5%	3.5%
	300	3%	3.8%
Gossip	100	1%	2%
	200	1.5%	2.5%
	300	2%	2.5%
Improved Gossip	100	0.2%	0.3%
	200	0.5%	0.5%
	300	0.7%	0.6%

Singlecast наиболее уязвим к сбоям (3–4%), так как один отказ прерывает цепочку. Broadcast и Improved Gossip лучше всего справляются с отказами (0.2–1%). Gossip устойчив, но медленнее.

Выводы.

Singlecast:

- Плюсы: Простота реализации, предсказуемое время доставки.
- Минусы: Низкая надежность (8–11% потерь), цепочка рвется при любом сбое.

Вывод: подходит для небольших сетей, где критична последовательность передачи.

Broadcast:

- Плюсы: Высокая скорость (0.08–0.12 с) и надежность (97–99%).
- Минусы: создает высокую нагрузку при большом числе узлов.

Вывод: Оптимален для сетей с высокой связанностью, где важна скорость.

Multicast:

- Плюсы: Умеренная нагрузка, лучшее время, чем у Gossip.

- Минусы: менее надежен, чем Broadcast (93–95%).

Вывод: Хорош для баланса между скоростью и нагрузкой.

Gossip:

- Плюсы: Устойчивость к сбоям (95–97% доставки).
- Минусы: Медленный (0.15–0.35 с).

Вывод: подходит для ненадежных сетей, где важна отказоустойчивость.

Improved Gossip

- Плюсы: Лучшая надежность (98.7–99.5%) и скорость (0.10–0.14 с).
- Минусы: Сложнее в реализации.

Вывод: оптимальный выбор для большинства сценариев.

Заключение

В рамках проведённого исследования были реализованы и сравнительно проанализированы пять различных алгоритмов распространения информации в распределённых системах: Singlecast, Broadcast, Multicast, Gossip и Improved Gossip. Каждый из них продемонстрировал уникальные характеристики в зависимости от условий функционирования сети, таких как размер, вероятность потерь пакетов и сбои узлов.