

# Отчёт по проекту: Моделирование сетевых технологий с использованием мультиагентной системы

Студент: Ковалев С.А.

Группа: 2261-ДМ

2025

## Содержание

<b>1</b>	<b>Введение</b>	<b>2</b>
<b>2</b>	<b>Архитектура системы</b>	<b>2</b>
<b>3</b>	<b>Модель OSI и её реализация в системе</b>	<b>2</b>
<b>4</b>	<b>Реализация уровней сетевого стека</b>	<b>2</b>
4.1	Канальный уровень (L2): Коммутация и MAC-адресация . . . . .	2
4.2	Сетевой уровень (L3): IP-адресация и маршрутизация . . . . .	3
4.2.1	IP-адресация и подсети . . . . .	3
4.2.2	Маршрутизация . . . . .	3
4.2.3	TTL (Time to Live) . . . . .	4
<b>5</b>	<b>Формат пакета и обоснование передаваемой информации</b>	<b>4</b>
5.1	Структура пакета . . . . .	4
5.2	Сериализация пакета . . . . .	5
<b>6</b>	<b>Визуализация и отладка</b>	<b>5</b>
<b>7</b>	<b>Интерфейс приложения</b>	<b>5</b>
<b>8</b>	<b>Заключение</b>	<b>6</b>

# 1 Введение

Данный проект представляет собой мультиагентную систему (Multi-Agent System, MAS), реализованную на платформе JADE (Java Agent Development Framework), предназначенную для моделирования сетевых технологий. Система имитирует поведение реальных сетевых устройств: компьютеров (PC), коммутаторов (Switch) и маршрутизаторов (Router), а также передачу данных между ними.

Цель проекта — продемонстрировать работу сетевого стека на разных уровнях модели OSI, визуализировать маршрутизацию пакетов и отладить поведение устройств в условиях межсетевого взаимодействия.

## 2 Архитектура системы

Система состоит из следующих компонентов:

- **Агенты:** PCAgent, SwitchAgent, RouterAgent — моделируют поведение устройств.
- **Контейнеры JADE:** Каждая подсеть (subnet) — отдельный контейнер.
- **Конфигурация:** JSON-файл описывает топологию, IP-адреса и маршруты.
- **GUI:** Визуализация сети и анимация передачи пакетов.

## 3 Модель OSI и её реализация в системе

Ниже представлена таблица соответствия уровней модели OSI и реализованных в системе функций.

**Таблица 1:** Соответствие уровней модели OSI и функций агентов

Уровень	Название	Реализация	Компонент
7	Прикладной	Отправка/приём PING/PONG	PCAgent
6	Представления	—	—
5	Сеансовый	—	—
4	Транспортный	— (упрощён)	—
3	Сетевой	IP-адресация, маршрутизация, TTL	RouterAgent, PCAgent
2	Канальный	MAC-адресация, коммутация	SwitchAgent, DeviceAgent
1	Физический	Передача сообщений (JADE)	ACLMessage

## 4 Реализация уровней сетевого стека

### 4.1 Канальный уровень (L2): Коммутация и MAC-адресация

Коммутатор (SwitchAgent) работает на канальном уровне, обрабатывая пакеты по MAC-адресам. Каждый агент имеет уникальный MAC, генерируемый на основе имени:

```

1 private String generateMAC(String name) {
2     int hash = Math.abs(name.hashCode());
3     return String.format("%02X:%02X:%02X:%02X:%02X:%02X",
4         (hash >> 16) & 0xFF,
5         (hash >> 8) & 0xFF,
6         hash & 0xFF,
7         (hash >> 24) & 0xFF,
8         (hash >> 16) & 0xFF,
9         (hash >> 8) & 0xFF);
10 }

```

**Листинг 1:** Генерация MAC-адреса в DeviceAgent.java

Коммутатор поддерживает MAC-таблицу и пересылает пакеты:

```

1 macTable.put(sender, "Port" + (macTable.size() + 1));

```

**Листинг 2:** Обновление MAC-таблицы в SwitchAgent.java

Это соответствует принципу **обучения коммутатора** в реальных сетях: при получении кадра устройство запоминает MAC-адрес источника и порт, через который он пришёл [1].

## 4.2 Сетевой уровень (L3): IP-адресация и маршрутизация

### 4.2.1 IP-адресация и подсети

Каждое устройство имеет IP-адрес и принадлежит к подсети (например, 192.168.1.0/24). Проверка принадлежности к сети реализована через маску:

```

1 private boolean isInNetwork(String ip, String network) {
2     String[] parts = network.split("/");
3     long ipLong = ipToLong(ip);
4     long netLong = ipToLong(parts[0]);
5     long mask = (0xFFFFFFFFL << (32 - Integer.parseInt(parts[1]))) &
6     0xFFFFFFFFL;
7     return (ipLong & mask) == (netLong & mask);
8 }

```

**Листинг 3:** Проверка принадлежности IP к сети

Алгоритм реализует побитовую маску — аналогично обработке маршрутов в реальных ОС и сетевых устройствах.

### 4.2.2 Маршрутизация

Маршрутизатор (RouterAgent) использует таблицу маршрутизации, загружаемую из конфигурации:

```

1 "routing": [
2     {
3         "from": "192.168.1.0/24",
4         "to": "192.168.2.0/24",
5         "nextHop": "Router2"
6     },
7     {
8         "from": "192.168.2.0/24",
9         "to": "192.168.1.0/24",

```

```

10     "nextHop": "Router1"
11   }
12 ]

```

**Листинг 4:** network-config.json: правила маршрутизации

Это соответствует статической маршрутизации, используемой в небольших сетях.

### 4.2.3 TTL (Time to Live)

Каждый маршрутизатор уменьшает TTL. При достижении 0 пакет отбрасывается:

```

1 packet.setTTL(packet.getTTL() - 1);
2 if (packet.getTTL() <= 0) {
3     System.out.println("TTL
4         ");
5     return;
6 }

```

**Листинг 5:** Проверка TTL в RouterAgent.java

Поле TTL защищает сеть от заикливания пакетов — критический механизм, описанный в RFC 791 [2].

## 5 Формат пакета и обоснование передаваемой информации

### 5.1 Структура пакета

Класс `PacketInfo` моделирует сетевой пакет. Ниже представлена таблица с полями и их назначением.

**Таблица 2:** Поля пакета и их сетевое назначение

Поле	Назначение и обоснование
<code>packetId</code>	Уникальный идентификатор. Позволяет коррелировать PING и PONG.
<code>sourceIP, destIP</code>	Адресация на L3. Используется для маршрутизации.
<code>sourceMAC, destMAC</code>	Зарезервированы для L2. Позволяют в будущем реализовать ARP.
<code>type</code>	Тип пакета (PING/PONG). Имитирует ICMP.
<code>protocol</code>	Указывает протокол (ICMP). Добавляет семантику.
<code>ttl</code>	Ограничивает число переходов. Защита от петель.
<code>currentHop</code>	Текущее устройство. Используется для логирования.
<code>path</code>	Полный маршрут. Критично для визуализации в GUI.

## 5.2 Сериализация пакета

Пакет передаётся как строка:

```
1 public String toMessageString() {  
2     return String.join(":", packetId, type, sourceIP, destIP, ...);  
3 }
```

Листинг 6: Сериализация в PacketInfo.java

### Формат пакета PacketInfo

packetId	type	sourceIP	destIP	sourceMAC	destMAC	ttl	currentHop	path
0	1	2	3	4	5	6	7	8

Разделитель полей: :

Пример: P001:PING:192.168.1.10:192.168.2.10:...

Рис. 1: Структура сериализованного пакета

## 6 Визуализация и отладка

Графический интерфейс (NetworkGUI) позволяет:

- Выбирать источник и получателя для PING.
- Визуализировать маршрут прохождения пакета.
- Наблюдать за TTL, типом пакета и путём.

Анимация пакета реализована через таймер:

```
1 animationTimer = new Timer(500, e -> {  
2     if (animationStep < currentRoute.size() - 1) {  
3         animationStep++;  
4         repaint();  
5     }  
6 });
```

Листинг 7: Анимация пакета в NetworkPanel.java

## 7 Интерфейс приложения

Панель схемы сети:

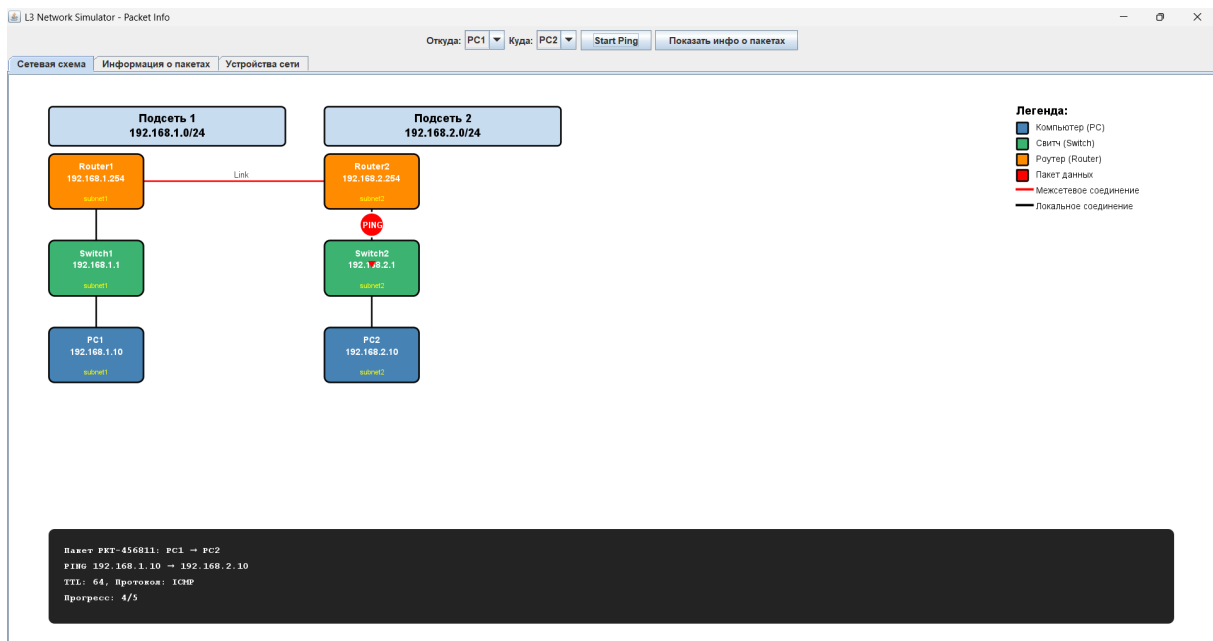


Рис. 2: Панель схемы сети

## 8 Заключение

Проект успешно моделирует ключевые принципы сетевых технологий:

- Работу L2 и L3 уровней модели OSI.
- Маршрутизацию между подсетями.
- Использование TTL, IP/MAC-адресации.
- Визуализацию сетевого трафика.

Система демонстрирует, как агентный подход может быть использован для моделирования сложных сетевых взаимодействий. В будущем возможна реализация ARP, DHCP, NAT и других протоколов.

## Список литературы

- [1] Tanenbaum, A. S., *Computer Networks*, 5th ed. Prentice Hall, 2011.
- [2] Postel, J., *RFC 791: Internet Protocol*, DARPA, 1981.
- [3] Postel, J., *RFC 792: Internet Control Message Protocol*, DARPA, 1981.
- [4] Bellifemine, F., Caire, G., Greenwood, D., *Developing Multi-Agent Systems with JADE*. Wiley, 2007.