Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления (ИУ)»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии (ИУ7)»

ОТЧЕТ

Лабораторная работа №3

по курсу «Математические основы верификации ПО»

на тему: «Моделирование сетевого протокола»

Студент ИУ7-42М (Группа) _____ (Подпись, дата) К.Э. Ковалец (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) О.В. Кузнецова (И. О. Фамилия)

2025 г.

1 Выполнение лабораторной работы

1.1 Задание

Выбирается любой сетевой протокол и описывается упрощенная модель этого протокола. Необязательно полностью все поля, например, IP-пакетов.

В отчёте содержится:

— описание протокола и принятые допущения;

— описываемые uml-sequence при работе;

— модель протокола;

— логи SPIN, демонстрирующие отправку/получение данных: пакетов, HTML-документов и т.д.;

— выводы по работе.

1.2 Описание протокола и принятые допущения

В данной работе разработана и реализована упрощённая модель транспортного протокола TCP (Transmission Control Protocol). TCP является протоколом транспортного уровня, который обеспечивает надёжную и упорядоченную передачу данных между двумя узлами в сети. Для упрощения модели были приняты следующие допущения:

— исключены сложные механизмы управления потоком, перегрузкой и повторной передачей данных;

— передача данных осуществляется по принципу Stop-and-Wait, то есть по одному пакету за раз с обязательным подтверждением;

— используются только основные типы сообщений: `AUTH` (авторизация), `OK` (подтверждение), `REQ` (запрос данных), `DATA` (данные), `ACK` (подтверждение получения данных) и `FIN` (завершение соединения);

— канал связи моделируется с ограниченным буфером, способным хранить не более трёх сообщений одновременно;

— отсутствует обработка ошибок, таких как потеря или повреждение пакетов.

Модель демонстрирует основные этапы взаимодействия между клиентом и сервером, включая установление соединения, передачу данных и завершение соединения. Такое упрощение позволяет сосредоточиться на базовых принципах работы протокола TCP, не углубляясь в сложные аспекты его реализации.

1.3 Описываемые UML-схемы при работе

На рисунке 1.1 показана диаграмма последовательности, иллюстрирующая процесс взаимодействия между получателем и отправителем в рамках упрощённой модели протокола TCP. Диаграмма отображает основные этапы работы протокола, включая установление соединения, передачу данных и завершение соединения.

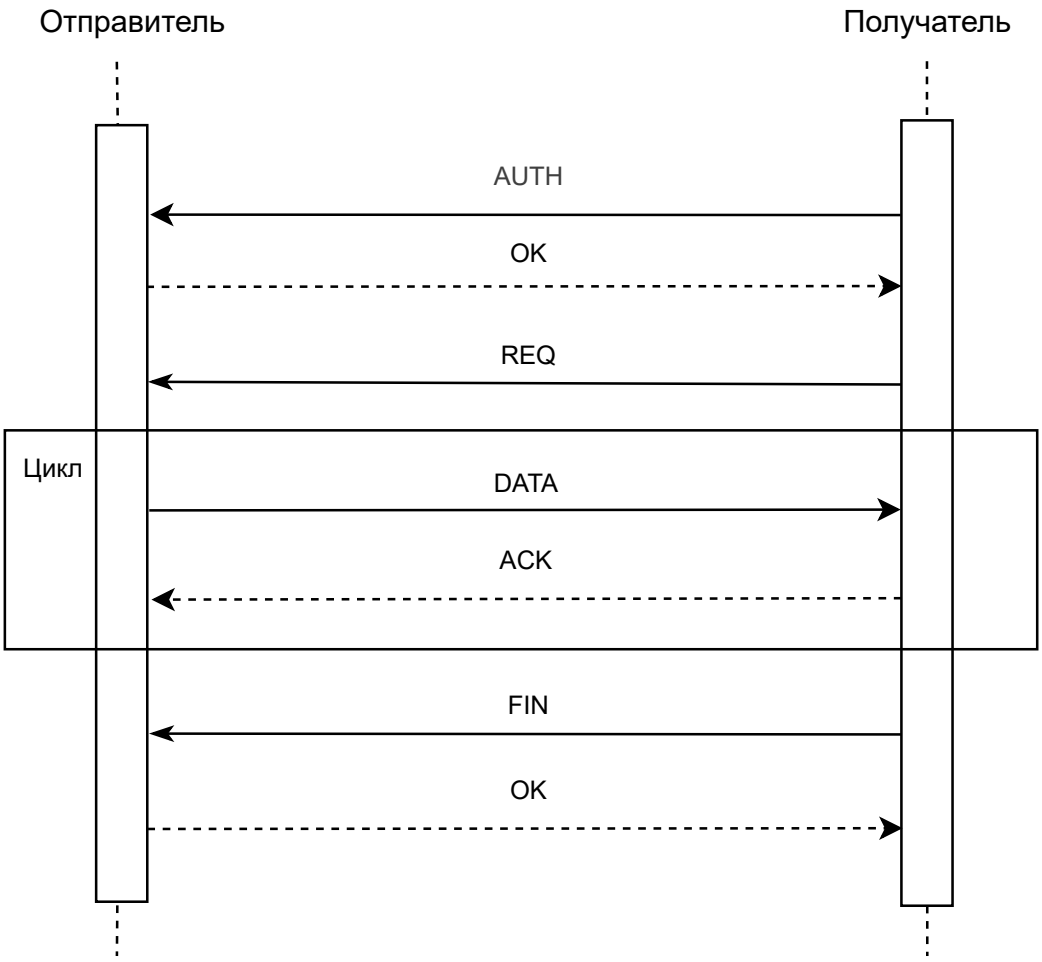

Рисунок 1.1 – Диаграмма последовательности упрощенного протокола TCP

1.4 Модель протокола

В данной работе была разработана и реализована модель упрощённого транспортного протокола TCP с использованием языка Promela. Код модели приведён в листинге 1.1.

Модель позволяет наглядно продемонстрировать основные принципы работы протокола TCP, включая обмен сообщениями и управление последовательностью взаимодействий между процессами.

Система включает два процесса: `Sender` (отправитель) и `Receiver` (получатель), которые взаимодействуют посредством канала `ch`. Для обмена данными используются заранее определённые типы сообщений: `AUTH`, `OK`, `REQ`, `DATA`, `ACK` и `FIN`.

Основные этапы взаимодействия между отправителем и получателем, которые демонстрирует модель, включают:

— Установление соединения, где получатель инициирует авторизацию (`AUTH`), а отправитель подтверждает её (`OK`).

— Запрос данных, где получатель отправляет запрос (`REQ`), а отправитель начинает передачу данных (`DATA`).

— Передачу данных, осуществляемую по принципу Stop-and-Wait: отправитель отправляет данные (`DATA`), а получатель подтверждает их получение (`ACK`).

— Завершение соединения, при котором получатель инициирует завершение (`FIN`), а отправитель подтверждает его (`OK`).

Листинг 1.1 — Код с реализацией программы для демонстрации работы упрщенного протокола TCP

```
1  mtype = { AUTH, OK, REQ, DATA, ACK, FIN };
2
3  proctype Sender(chan ch; int msg_count) {
4    ch?AUTH, 1;
5    printf("\nОтправитель <- AUTH");
6    printf("\nОтправитель -> OK");
7    ch!OK, 0;
8
9    ch?REQ, 0;
```

Продолжение листинга 1.1

```
10      printf("\nОтправитель <- REQ");
11
12      int ind = 1;
13      int number = 2025;
14      do
15      :: ind <= msg_count -> {
16        printf("\n(%d) Отправитель -> DATA = %d", ind, number);
17        ch!DATA, number;
18        ch?ACK, ind;
19        printf("\n(%d) Отправитель <- ACK", ind);
20
21        number++;
22        ind++;
23      }
24      :: else -> break;
25      od;
26
27      ch?FIN, 0;
28      printf("\nОтправитель <- FIN");
29      printf("\nОтправитель -> OK");
30      ch!OK, 0;
31  }
32
33  proctype Receiver(chan ch; int msg_count) {
34      printf("\nПолучатель  -> AUTH");
35      ch!AUTH, 1;
36      ch?OK, 0;
37      printf("\nПолучатель  <- OK");
38
39      printf("\nПолучатель  -> REQ");
40      ch!REQ, 0;
41
42      int ind = 1;
43      int receive_data;
44      do
45      :: ind <= msg_count -> {
46        ch?DATA, receive_data;
47        printf("\n(%d) Получатель  <- DATA = %d", ind, receive_data);
48        printf("\n(%d) Получатель  -> ACK", ind);
49        ch!ACK, ind;
50
51        ind++;
52      }
53      :: else -> break
54      od;
```

Продолжение листинга 1.1

```
55
56      printf("\nПолучатель  -> FIN");
57      ch!FIN, 0;
58      ch?OK, 0;
59      printf("\nПолучатель  <- OK\n");
60    }
61
62    init {
63      chan ch = [3] of { mtype, int };
64      int msg_count = 4;
65
66      run Receiver(ch, msg_count);
67      run Sender(ch, msg_count);
68    }
```

1.5 Логи SPIN, демонстрирующие отправку/получение данных

Логи SPIN демонстрируют последовательность взаимодействий между процессами `Sender` и `Receiver`. В логах видно, что процессы корректно обмениваются сообщениями в соответствии с описанным протоколом.

Логи SPIN приведены в листинге 1.2.

Листинг 1.2 — Логи SPIN, демонстрирующие отправку/получение данных

```
1
2   Получатель  -> AUTH
3   Отправитель <- AUTH
4   Отправитель -> OK
5   Получатель  <- OK
6   Получатель  -> REQ
7   Отправитель <- REQ
8   (1) Отправитель -> DATA = 2025
9   (1) Получатель  <- DATA = 2025
10  (1) Получатель  -> ACK
11  (1) Отправитель <- ACK
12  (2) Отправитель -> DATA = 2026
13  (2) Получатель  <- DATA = 2026
14  (2) Получатель  -> ACK
15  (2) Отправитель <- ACK
16  (3) Отправитель -> DATA = 2027
```

Продолжение листинга 1.2

```
(3) Получатель  <- DATA = 2027
(3) Получатель  -> ACK
(3) Отправитель <- ACK
(4) Отправитель -> DATA = 2028
(4) Получатель  <- DATA = 2028
(4) Получатель  -> ACK
(4) Отправитель <- ACK
Получатель  -> FIN
Отправитель <- FIN
Отправитель -> OK
Получатель  <- OK
3 processes created
```

1.6 Вывод

В ходе выполнения лабораторной работы была разработана и реализована упрощённая модель транспортного протокола TCP с использованием языка Promela. Модель продемонстрировала основные принципы работы протокола, включая установление соединения, передачу данных и завершение соединения. Логи SPIN демонстрируют корректное взаимодействие процессов `Sender` и `Receiver`, а также отсутствие взаимных блокировок.