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**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

ОТЧЕТ ПО по Лабораторной работе №3
по курсу
«Моделирование сетевого протокола»
Тема
«Моделирование гонки процессов»

Студент группы **ИУ7И-41М** **Баматраф Сохайб С.А.**

(И.О. Фамилия)

Преподаватель **Кузнецова О.В.**

(И.О. Фамилия)

***2024** г.*

Содержание

1. Введение

В этом отчете описывается процесс моделирования упрощенной версии протокола управления передачей (TCP) с помощью языка Promela и верификация модели с помощью программы проверки моделей SPIN. TCP - это фундаментальный протокол в наборе протоколов Интернета, отвечающий за надежную, упорядоченную и проверенную на ошибки доставку потока байтов между приложениями, работающими на хостах, взаимодействующих через IP-сеть.

1.1 Цель и задачи

Цель:

Основная цель этой лабораторной работы - понять фундаментальные аспекты работы сетевых протоколов с помощью моделирования и симуляции. Основное внимание уделяется фазам установления соединения, передачи данных и завершения соединения TCP.

Задачи:

Задача состоит в том, чтобы создать упрощенную модель протокола TCP, сосредоточившись на фазах трехстороннего рукопожатия и передачи данных, проверить модель с помощью программы проверки моделей SPIN и проанализировать полученные результаты.

2. Описание протокола и принятые допущения

Протокол TCP обеспечивает надежную связь между клиентскими и серверными приложениями по сети. Для упрощения были сделаны следующие предположения:

- Сеть является идеальной, без потери, переупорядочивания или повреждения пакетов.
- Моделируются только сообщения SYN, SYN-ACK, ACK и DATA.
- Последовательность операций строго соблюдается без внесения случайностей и ошибок.

3. Описываемые uml-sequence при работе

Диаграмма последовательности UML иллюстрирует процесс трехстороннего рукопожатия TCP и последующую передачу данных:

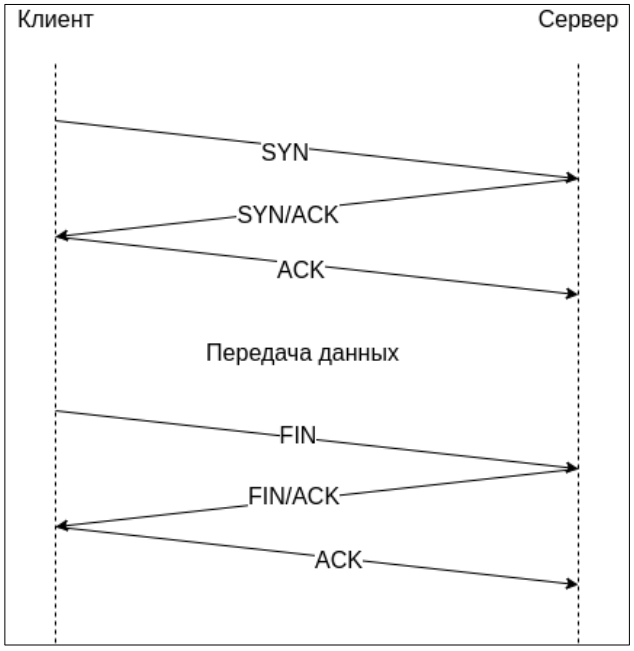

Рисунок 1.- Установление и закрытие TCP-соединения.

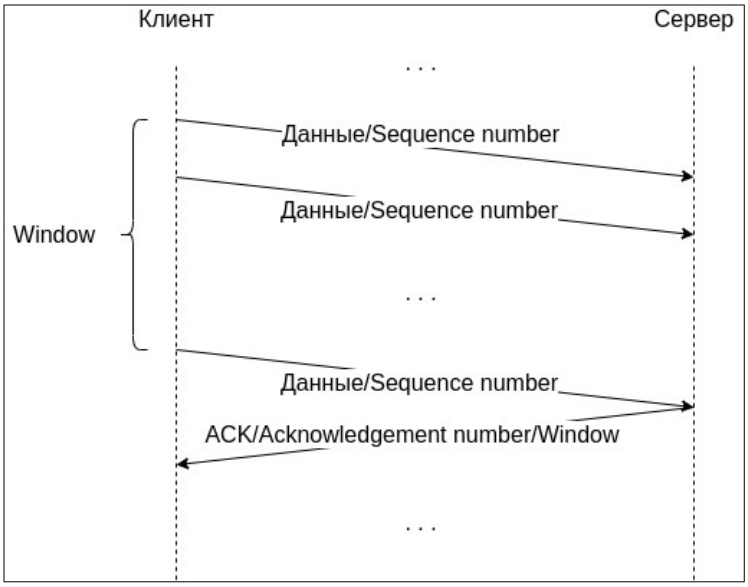

Рисунок 2.- Последовательность передачи данных TCP.

1. Клиент посылает SYN, чтобы инициировать соединение.
2. Сервер отвечает SYN-ACK, чтобы подтвердить SYN и продолжить процесс соединения.
3. Клиент посылает ACK, чтобы подтвердить SYN-ACK, устанавливая соединение полностью.
4. Передача данных: Для представления фазы обмена данными показано упрощенное событие передачи данных.

4. Модель протокола

В этом разделе отчета представлен код Promela, использованный для моделирования упрощенного протокола TCP. Код включает в себя основные элементы трехстороннего рукопожатия TCP и фазы передачи данных, обеспечивая четкое и исполняемое представление операций протокола.

Код Promela определяет два процесса: Клиент и Сервер, которые взаимодействуют через каналы сообщений. Клиент инициирует соединение, ожидает подтверждения, отправляет данные и, наконец, выдает запрос на закрытие соединения. Сервер прослушивает сообщения клиента, отвечает на них и отправляет данные обратно клиенту.

Ниже приведена версия кода Promela для Protocol.pml:

Листинг 1 — Код модели RProtocol.pml.

```
mtype = { SYN, SYN_ACK, ACK, DATA, FIN };
// Channels for communication
chan clientToServer = [2] of { mtype, byte };
chan serverToClient = [2] of { mtype, byte };
proctype Client() {
    clientToServer ! SYN, 0;     // Send SYN to server
    serverToClient ? SYN_ACK, _; // Wait for SYN-ACK from server
    clientToServer ! ACK, 0;     // Send ACK to server
    // Data transmission
```

```
    serverToClient ? DATA, _;    // Receive data
    printf("Client received data\n");
    // Connection termination
    clientToServer ! FIN, 0;     // Send FIN to close connection
}
proctype Server() {
    clientToServer ? SYN, _;     // Wait for SYN
    serverToClient ! SYN_ACK, 0; // Send SYN-ACK
    clientToServer ? ACK, _;     // Wait for ACK
    // Data transmission
    serverToClient ! DATA, 0;    // Send data
    printf("Server sent data\n");
    // Connection termination
    clientToServer ? FIN, _;     // Wait for FIN to close connection
}
init {
    run Client();
    run Server();
}
```

5. Логи SPIN, демонстрирующие отправку/получение данных

Журналы моделирования и верификации SPIN позволили получить подробное представление о взаимодействии процессов:

```
0:    proc  - (:root:) creates proc  0 (:init:)
      Starting initial system setup and creating primary processes.
1:    proc  0 (:init::1) creates proc  1 (Client)
      The initial process spawns the Client process.
2:    proc  1 (Client:1) clientToServer!SYN,0
      Client sends SYN packet to Server to initiate connection.
3:    proc  0 (:init::1) creates proc  2 (Server)
      The initial process spawns the Server process.
4:    proc  2 (Server:1) clientToServer?SYN,_
      Server receives the SYN packet from the Client.
5:    proc  2 (Server:1) serverToClient!SYN_ACK,0
      Server sends SYN_ACK packet back to the Client, acknowledging the SYN.
6:    proc  1 (Client:1) serverToClient?SYN_ACK,_
      Client receives the SYN_ACK packet from the Server.
7:    proc  1 (Client:1) clientToServer!ACK,0
      Client sends ACK packet to Server, completing the three-way handshake.
8:    proc  2 (Server:1) clientToServer?ACK,_
      Server receives the ACK packet, connection is now established.
9:    proc  2 (Server:1) serverToClient!DATA,0
      Server sends DATA packet to the Client, initiating data transfer.
10:   proc  1 (Client:1) serverToClient?DATA,_
      Client receives DATA packet from Server.
      Server sent data
      Client received data
```

Рисунок 3.- Снимок трехстороннего рукопожатия TCP и передачи данных.

Инициализация: Корневой процесс инициирует создание основной среды моделирования.

Создание процесса: Начальный процесс (:init:) создает процесс клиента.

Отправка SYN: Клиент начинает трехстороннее рукопожатие TCP, отправляя пакет SYN на сервер для запроса соединения.

Создание сервера: В ответ начальный процесс создает процесс Server.

SYN Received: Сервер получает SYN-пакет от клиента.

SYN-ACK Sent: Сервер отвечает, отправляя пакет SYN-ACK обратно клиенту, подтверждая полученный SYN.

SYN-ACK Received: Клиент получает SYN-ACK от сервера.

ACK Sent: Клиент посылает серверу пакет ACK, завершая трехстороннее рукопожатие и устанавливая соединение.

ACK Received: Сервер получает пакет ACK, подтверждая успешное установление соединения.

Data Sent: Сервер отправляет клиенту пакет DATA, представляющий собой передачу данных по установленному соединению.

Данные получены: Клиент получает пакет DATA от сервера.

Трассировка заканчивается тем, что Сервер и Клиент регистрируют успешную передачу и прием данных, иллюстрируя эффективную связь между двумя сущностями через установленное TCP-соединение.

Заключение

Результаты моделирования подтверждают корректное функционирование трехстороннего рукопожатия TCP и передачи данных в идеальной сетевой среде в соответствии с заданной моделью Promela. Все ожидаемые сообщения были отправлены и получены в правильном порядке, а соединение было изящно завершено. Журналы проверки SPIN подтвердили отсутствие ошибок, тупиков или недействительных конечных состояний, продемонстрировав устойчивость модели протокола в протестированных условиях.

Это упражнение дает ценное представление о динамике сетевых протоколов и подчеркивает полезность инструментов формальной верификации для обеспечения корректности реализации протоколов в контролируемой среде.

Литература

1. https://github.com/sohaibssb/Mathematical-basics-of-verification/tree/main/NetworkProtocolModeling-Lab3