|  |  |
| --- | --- |
|  | **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, искусственный интеллект и системы управления» |
|  |  |
| КАФЕДРА | «Программное обеспечение ЭВМ и информационные технологии» |

ОТЧЁТ

по лабораторной работе № 3  
по курсу «Математические основы верификации ПО»  
«Моделирование сетевого протокола»

**Студент:** Керимов А. Ш.

**Группа:** ИУ7-42М

**Преподаватель:** Кузнецова О. В.

Москва.

2024 г.

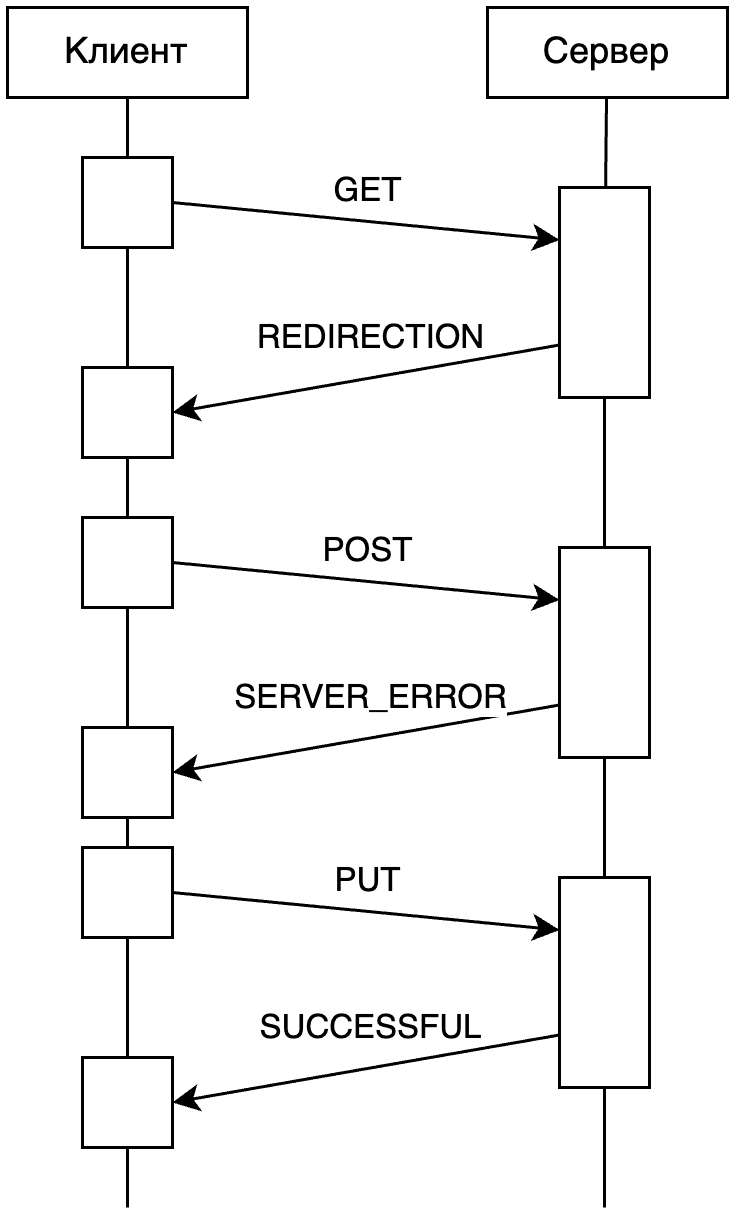
# Задание

# Выбирается любой сетевой протокол и описывается упрощенная модель этого протокола.

# Описание протокола и принятые допущения

В качестве протокола был выбран протокол HTTP, для упрощения взаимодействия рассматриваются только типы запросов GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH.

# Описываемые UML Sequence при работе



# Модель протокола

mtype = {

GET,

HEAD,

POST,

PUT,

DELETE,

TRACE,

OPTIONS,

CONNECT,

PATCH,

INFORMATIONAL,

SUCCESSFUL,

REDIRECTION,

CLIENT\_ERROR,

SERVER\_ERROR

}

chan http = [0] of { mtype };

active proctype client() {

start:

if :: http ! GET -> printf("client: send GET\n")

:: http ! HEAD -> printf("client: send HEAD\n")

:: http ! POST -> printf("client: send POST\n")

:: http ! PUT -> printf("client: send PUT\n")

:: http ! DELETE -> printf("client: send DELETE\n")

:: http ! TRACE -> printf("client: send TRACE\n")

:: http ! OPTIONS -> printf("client: send OPTIONS\n")

:: http ! CONNECT -> printf("client: send CONNECT\n")

:: http ! PATCH -> printf("client: send PATCH\n")

fi

if :: http ? INFORMATIONAL -> printf("client: receive INFORMATIONAL\n")

:: http ? SUCCESSFUL -> printf("client: receive SUCCESSFUL\n")

:: http ? REDIRECTION -> printf("client: receive REDIRECTION\n")

:: http ? CLIENT\_ERROR -> printf("client: receive CLIENT\_ERROR\n")

:: http ? SERVER\_ERROR -> printf("client: receive SERVER\_ERROR\n")

fi

goto start

}

active proctype server() {

start:

if :: http ? GET -> printf("server: receive GET\n")

:: http ? HEAD -> printf("server: receive HEAD\n")

:: http ? POST -> printf("server: receive POST\n")

:: http ? PUT -> printf("server: receive PUT\n")

:: http ? DELETE -> printf("server: receive DELETE\n")

:: http ? TRACE -> printf("server: receive TRACE\n")

:: http ? OPTIONS -> printf("server: receive OPTIONS\n")

:: http ? CONNECT -> printf("server: receive CONNECT\n")

:: http ? PATCH -> printf("server: receive PATCH\n")

fi

if :: http ! INFORMATIONAL -> printf("server: send INFORMATIONAL\n")

:: http ! SUCCESSFUL -> printf("server: send SUCCESSFUL\n")

:: http ! REDIRECTION -> printf("server: send REDIRECTION\n")

:: http ! CLIENT\_ERROR -> printf("server: send CLIENT\_ERROR\n")

:: http ! SERVER\_ERROR -> printf("server: send SERVER\_ERROR\n")

fi

goto start

}

# Логи SPIN

client: send GET

server: receive GET

server: send CLIENT\_ERROR

client: receive CLIENT\_ERROR

client: send CONNECT

server: receive CONNECT

server: send SUCCESSFUL

client: receive SUCCESSFUL

client: send GET

server: receive GET

server: send CLIENT\_ERROR

client: receive CLIENT\_ERROR

client: send PUT

server: receive PUT

server: send CLIENT\_ERROR

client: receive CLIENT\_ERROR

client: send PATCH

server: receive PATCH

server: send SERVER\_ERROR

client: receive SERVER\_ERROR

client: send PATCH

server: receive PATCH

server: send SERVER\_ERROR

client: receive SERVER\_ERROR

client: send GET

server: receive GET

server: send INFORMATIONAL

client: receive INFORMATIONAL

client: send POST

server: receive POST

server: send INFORMATIONAL

client: receive INFORMATIONAL

...

# Выводы

В результате выполнения лабораторной работы № 3 была описана упрощённая модель протокола HTTP, описана UML Sequence при работе и приведена модель протокола на языке Promela.