|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по Лабораторной работе №3**

**по курсу**

**«Математические основы верификации ПО»**

**Тема**

**«Моделирование сетевого протокола»**

Студент \_\_\_ИУ7-41М\_\_\_\_ \_\_\_\_\_Карпухин А.С.\_\_\_\_

(Группа) (И.О.Фамилия)

Преподаватель \_\_\_\_Кузнецова О.В.\_\_\_\_

(И.О.Фамилия)

*2022 г.*

**Задание**

Необходимо выбрать любой сетевой протокол и описать его модель на языке Promela. Протокол может быть реализован не полностью, но необходимо описать принятые допущения в реализации.

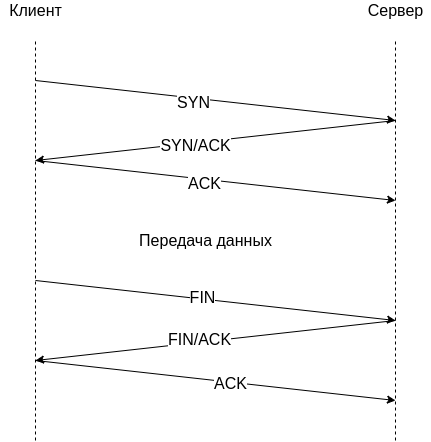
**Описание модели протокола**

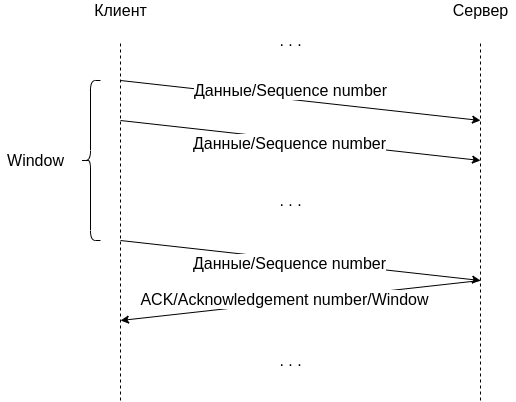
Для реализации был выбран сетевой протокол транспортного уровня — Transmission Control Protocol (TCP).

Протокол был реализован в упрощенном виде со следующими допущениями:

* размер окна получателя выбирается случайным образом
* получатель ожидает, что порядок следования пакетов при передаче будет строгим, то есть все пакеты доставляются по порядку, и порядок следования получателем не проверяется
* заголовок пакета передается в упрощенном виде и может содержать только флаги SYN, ACK, FIN, RST, размер окна и непосредственно передаваемые данные
* не высчитывается хеш-сумма заголовка, а также не формируется псевдозаголовок при передаче
* получатель ждет заполнения окна, и только после этого отправляет подтверждение, при этом если отправитель завершает передачу последнего пакета до заполнения окна, получатель отправляет ACK с нулевым номером подтверждения по истечении таймаута, после чего происходит закрытие соединения
* начальный размер окна не выбирается автоматически, а задается вручную в качестве аргумента

Диаграмма механизмов установки и закрытия соединения, реализованных в модели, приведена на рисунке 1. Диаграмма передачи данных приведена на рисунке 2.

Рисунок 1 — Установка и закрытие соединения

Рисунок 2 — Передача данных

**Листинги кода**

Код моделей отправителя (клиента) и получателя (сервера) приведен в листингах 1 и 2 соответственно.

Листинг 1 — Модель отправителя (клиента).

|  |
| --- |
| proctype Sender(chan ch; byte msgCount)  {  byte senderNumber, receiverNumber;  byte seqNumber = 0, ackNumber, window;  byte data[MAX\_DATA\_BUFER\_SIZE];  printf("SENDER: running, pid=%d\n", \_pid);  rand(senderNumber);  printf("SENDER: sending SYN\n");  ch ! SYN(senderNumber);  printf("SENDER: waiting for SYNACK\n");  if  :: ch ? SYNACK(receiverNumber, window) ->  ch ! ACK(receiverNumber);  printf("SENDER: connection established, senderNumber=%d, receiverNumber=%d, window=%d\n", senderNumber, receiverNumber, window);  do  :: seqNumber < msgCount ->  do  :: window > 0 && seqNumber < msgCount ->  rand(data[seqNumber]);  printf("SENDER: sending DATA %d with seq number %d\n", data[seqNumber], senderNumber + seqNumber);  ch ! DATA(senderNumber + seqNumber, data[seqNumber]);  seqNumber++;  window--;  :: else -> break;  od    printf("SENDER: waiting for ACK\n");  ch ? ACK(ackNumber, window);  byte nextPackage = ackNumber - receiverNumber;  if  :: nextPackage >= seqNumber && window > 0 ->  printf("SENDER: ACK received, ackNumber=%d(packageNumber=%d), window=%d\n", ackNumber, nextPackage, window);  :: nextPackage < seqNumber && window > 0 ->  printf("SENDER: ACK received, package lost, retring from packageNumber=%d\n", nextPackage);  :: else ->  printf("SENDER: ACK received, window == 0, wating for receiver is ready to receive\n");  do  :: ch ? ACK(ackNumber, window) ->  if  :: window > 0 ->  nextPackage = ackNumber - receiverNumber;  break;  :: else -> skip;  fi  od  fi  seqNumber = nextPackage;  :: else ->  printf("SENDER: end of transmition, closing connection ...\n");  ch ! FIN;  ch ? FINACK;  ch ! ACK;  printf("SENDER: connection closed\n");  break;  od  :: ch ? RST ->  printf("SENDER: RST received, connection was not established\n");  fi  } |

Листинг 2 — Модель получателя (сервера).

|  |
| --- |
| proctype Receiver(chan ch; byte window)  {  byte senderNumber, receiverNumber;  byte seqNumber, ackNumber = 0, expectedPackageNumber;  byte data, temp;  printf("RECEIVER: running, pid=%d\n", \_pid);  printf("RECEIVER: waiting for SYN\n");  ch ? SYN(senderNumber);  printf("RECEIVER: sending SYNACK\n");  rand(receiverNumber);  ch ! SYNACK(receiverNumber, window);  printf("RECEIVER: waiting for ACK\n");  ch ? ACK(temp);  if  :: temp == receiverNumber ->  printf("RECEIVER: connection established, senderNumber=%d, receiverNumber=%d, window=%d\n", senderNumber, receiverNumber, window);  byte receivedSeqNumbers[MAX\_WINDOW\_SIZE];  byte i = 0, j = 0;  do  :: ch ? DATA(seqNumber, data) ->  byte packageNumber = seqNumber - senderNumber;  if  :: packageNumber >= ackNumber && packageNumber < ackNumber + window ->  j = 0;  do  :: j < i && receivedSeqNumbers[j] != seqNumber -> j++;  :: j == i ->  printf("RECEIVER: valid DATA %d received, seqNumber=%d\n", data, seqNumber);  receivedSeqNumbers[j] = seqNumber;  i++;  break;  :: else ->  printf("RECEIVER: duplicate DATA %d received, ignoring\n", data);  break;  od  :: else ->  printf("RECEIVER: invalid DATA %d received, seqNumber=%d out of expected range\n", data, seqNumber);  fi  if  :: i == window ->  printf("RECEIVER: window is full, generating new ackNumber and window\n");  randAckNumber(ackNumber, window, receivedSeqNumbers, j);  rand(window);  printf("RECEIVER: new ackNumber=%d, new window=%d\n", ackNumber, window);  ch ! ACK(receiverNumber + ackNumber, window);  i = 0;  fi  :: timeout ->  ch ! ACK(0, 1);  :: ch ? FIN ->  printf("RECEIVER: FIN received, closing connection ...\n");  ch ! FINACK;  ch ? ACK;  printf("RECEIVER: connection closed\n");  od  :: else ->  printf("RECEIVER: error during connection establishment, exiting ...")  fi  } |

Пример работы модели приведен в листинге 3.

Листинг 3 — Пример результатов работы модели.

|  |
| --- |
| SENDER: running, pid=1  RECEIVER: running, pid=2  RECEIVER: waiting for SYN  SENDER: sending SYN  SENDER: waiting for SYNACK  RECEIVER: sending SYNACK  RECEIVER: waiting for ACK  SENDER: connection established, senderNumber=1, receiverNumber=1, window=1  SENDER: sending DATA 1 with seq number 1  RECEIVER: connection established, senderNumber=1, receiverNumber=1, window=1  SENDER: waiting for ACK  RECEIVER: valid DATA 1 received, seqNumber=1  RECEIVER: window is full, generating new ackNumber and window  RECEIVER: new ackNumber=1, new window=1  SENDER: ACK received, ackNumber=2(packageNumber=1), window=1  SENDER: sending DATA 5 with seq number 2  SENDER: waiting for ACK  RECEIVER: valid DATA 5 received, seqNumber=2  RECEIVER: window is full, generating new ackNumber and window  RECEIVER: new ackNumber=3, new window=1  SENDER: ACK received, ackNumber=4(packageNumber=3), window=1  SENDER: sending DATA 1 with seq number 4  SENDER: waiting for ACK  RECEIVER: valid DATA 1 received, seqNumber=4  RECEIVER: window is full, generating new ackNumber and window  RECEIVER: new ackNumber=5, new window=1  SENDER: ACK received, ackNumber=6(packageNumber=5), window=1  SENDER: end of transmition, closing connection ...  RECEIVER: FIN received, closing connection ...  RECEIVER: connection closed  SENDER: connection closed |