# Сети Петри — Простой протокол

**Отчёт по лабораторной работе №12**

Ибатулина Дарья Эдуардовна

# Содержание

# Список иллюстраций

# 1 Введение

## 1.1 Цели и задачи

**Цель работы**

Реализовать простой протокол передачи данных в CPN Tools.

**Задание**

- Реализовать простой протокол передачи данных в CPN Tools.
- Вычислить пространство состояний, сформировать отчет о нем и построить граф [1].

# 2 Теоретическое введение

Протоколы передачи данных - это наборы правил, определяющих порядок и способы обмена информацией между участниками вычислительных или телекоммуникационных систем. Они регулируют формат сообщений, последовательность их передачи, временные интервалы, а также методы обнаружения и исправления ошибок. В современной практике протоколы используются на всех уровнях модели OSI, начиная с физического и заканчивая прикладным [2].

Реальные протоколы передачи данных часто имеют сложную структуру, что затрудняет их анализ и отладку. Для исследования их свойств и выявления возможных ошибок широко применяется моделирование, в частности, с помощью сетей Петри и их расширений - раскрашенных сетей Петри (Coloured Petri Nets, CPN). Этот подход позволяет наглядно представить процессы передачи и подтверждения данных, а также учесть такие важные аспекты, как потеря пакетов и подтверждений, задержки и порядок доставки [3].

В данной лабораторной работе рассматривается простейший протокол передачи данных в ненадёжной сети. Передача осуществляется по следующему принципу: отправитель разбивает исходное сообщение на части (пакеты) и отправляет их получателю, ожидая подтверждения (ACK) о доставке каждого пакета перед отправкой следующего. Если подтверждение не получено (например, из-за потери пакета или ACK), пакет передаётся повторно. Такой механизм обеспечивает надёжность передачи даже в условиях, когда сеть может терять отдельные сообщения.

Модель протокола строится в среде CPN Tools с использованием раскрашен-

ных сетей Петри, что позволяет формально описать все возможные состояния системы и переходы между ними, а также смоделировать случайные потери пакетов и подтверждений. Анализ полученной модели даёт возможность оценить корректность работы протокола, выявить возможные тупиковые состояния и оптимизировать алгоритмы передачи данных [4].

# 3 Выполнение лабораторной работы

Основные состояния: источник (Send), получатель (Receiver). Действия (переходы): отправить пакет (Send Packet), отправить подтверждение (Send ACK). Промежуточное состояние: следующий посылаемый пакет (NextSend) [1]. Зададим декларации модели (рис. 3.1).

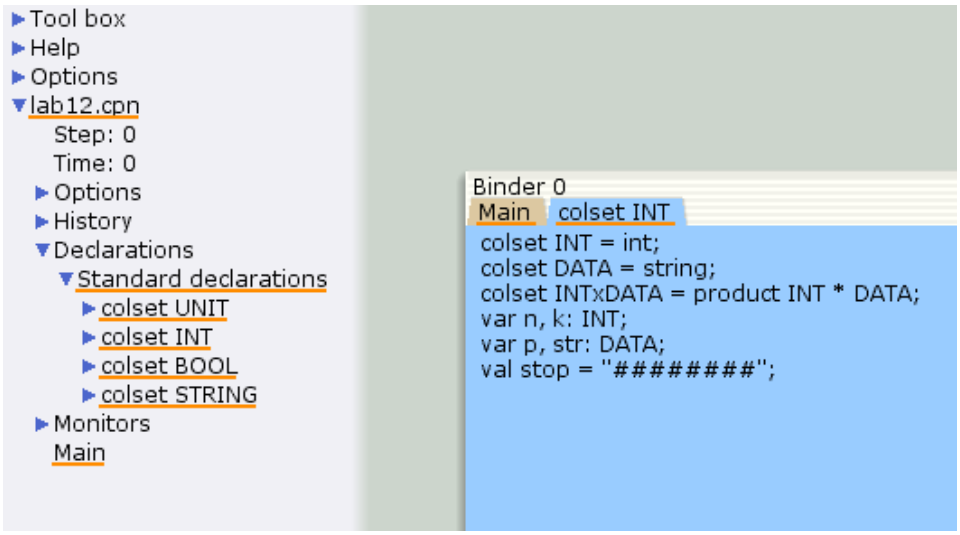


Рис. 3.1: Задание деклараций

Состояние Send имеет тип INTxDATA и следующую начальную маркировку (в соответствии с передаваемой фразой).

Стоповый байт (“########”) определяет, что сообщение закончилось. Состояние Receiver имеет тип DATA и начальное значение 1’“” (т.е. пустая строка, поскольку состояние собирает данные и номер пакета его не интересует). Состояние NextSend имеет тип INT и начальное значение 1’1. Поскольку пакеты

представляют собой кортеж, состоящий из номера пакета и строки, то выражение у двусторонней дуги будет иметь значение (n,p). Кроме того, необходимо взаимодействовать с состоянием, которое будет сообщать номер следующего посылаемого пакета данных. Поэтому переход Send Packet соединяем с состоянием NextSend двумя дугами с выражениями n. Также необходимо получать информацию с подтверждениями о получении данных. От перехода Send Packet к состоянию NextSend дуга с выражением n, обратно – k.

Зададим промежуточные состояния (A, B с типом INTxDATA, C, D с типом INTxDATA) для переходов (рис. 12.2): передать пакет Transmit Packet (передаём (n,p)), передать подтверждение Transmit ACK (передаём целое число k). Добавляем переход получения пакета (Receive Packet). От состояния Receiver идёт дуга к переходу Receive Packet со значением той строки (str), которая находится в состоянии Receiver. Обратно: проверяем, что номер пакета новый и строка не равна стоп-биту. Если это так, то строку добавляем к полученным данным. Кроме того, необходимо знать, каким будет номер следующего пакета. Для этого добавляем состояние NextRec с типом INT и начальным значением 1’1 (один пакет), связываем его дугами с переходом Receive Packet. Причём к переходу идёт дуга с выражением k, от перехода — if n=k then k+1 else k. Связываем состояния B и C с переходом Receive Packet. От состояния B к переходу Receive Packet — выражение (n,p), от перехода Receive Packet к состоянию C — выражение if n=k then k+1 else k. От перехода Receive Packet к состоянию Receiver: if n=k andalso p<>stop then str^p else str. (если n=k и мы не получили стоп-байт, то направляем в состояние строку и к ней прикрепляем p, в противном случае посылаем только строку). На переходах Transmit Packet и Transmit ACK зададим потерю пакетов. Для этого на интервале от 0 до 10 зададим пороговое значение и, если передаваемое значение превысит этот порог, то считаем, что произошла потеря пакета, если нет, то передаём пакет дальше. Для этого задаём вспомогательные состояния SP и SA с типом Ten0 и начальным значением 1‘8, соединяем с соответствующими переходами (рис. 3.2):

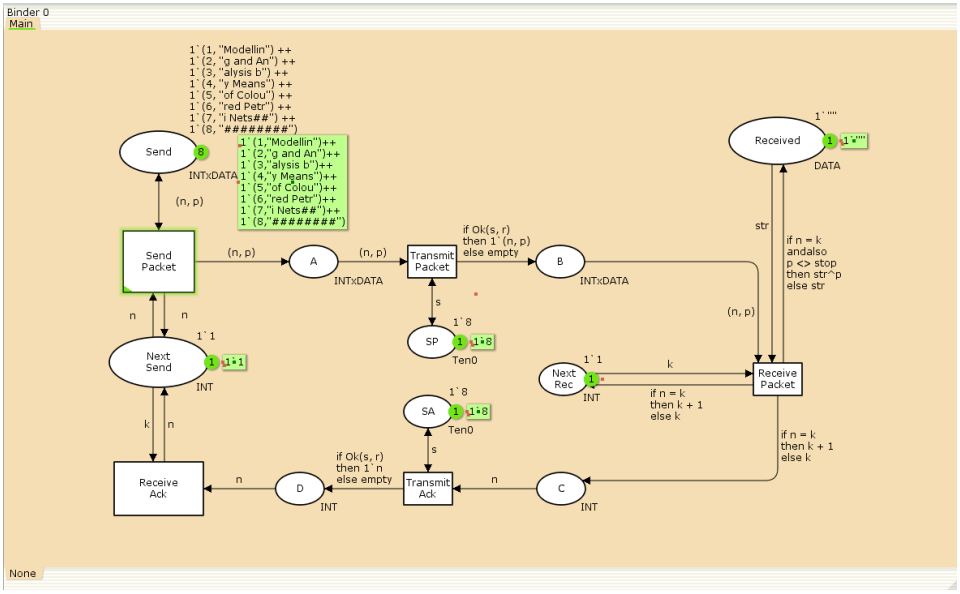


Рис. 3.2: Полученная модель

В декларациях задаём значения переменных (рис. 3.3), а также функцию, отвечающую за выбор: доставлен ли пакет или нет (рис. 3.4):

```
val stop = "########";
colset Ten0 = int with 0..10
colset Ten1 = int with 0..10
var s: Ten0;
var r: Ten1;
```

Рис. 3.3: Задание деклараций

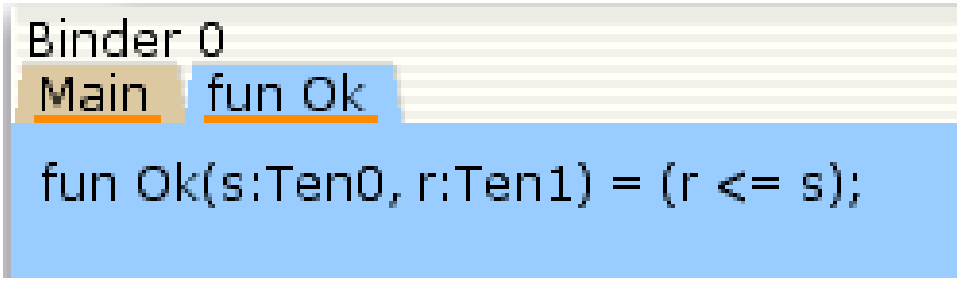


Рис. 3.4: Задание функции Ok

Таким образом, получим модель простого протокола передачи данных. Пакет последовательно проходит: состояние Send, переход Send Packet, состояние A, с некоторой вероятностью переход Transmit Packet, состояние B, попадает на переход Receive Packet, где проверяется номер пакета и если нет совпадения, то пакет направляется в состояние Received, а номер пакета передаётся последовательно в состояние C, с некоторой вероятностью в переход Transmit ACK, далее в состояние D, переход Receive ACK, состояние NextSend (увеличивая на 1 номер следующего пакета), переход Send Packet. Так продолжается до тех пор, пока не будут переданы все части сообщения. Последней будет передана стоп-последовательность (рис. 3.5, 3.6, 3.7, 3.8):

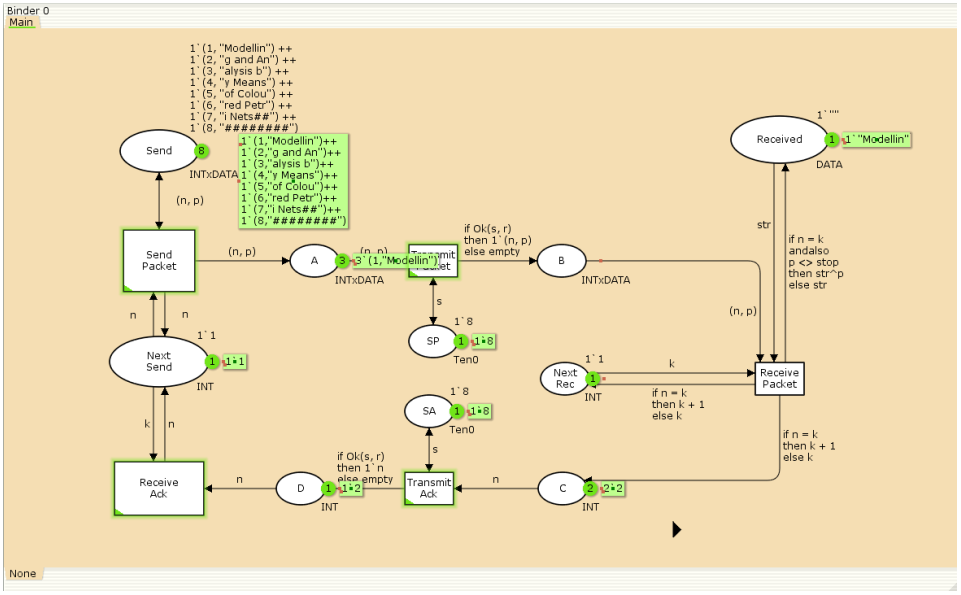


Рис. 3.5: Запуск модели простого протокола передачи данных (1)

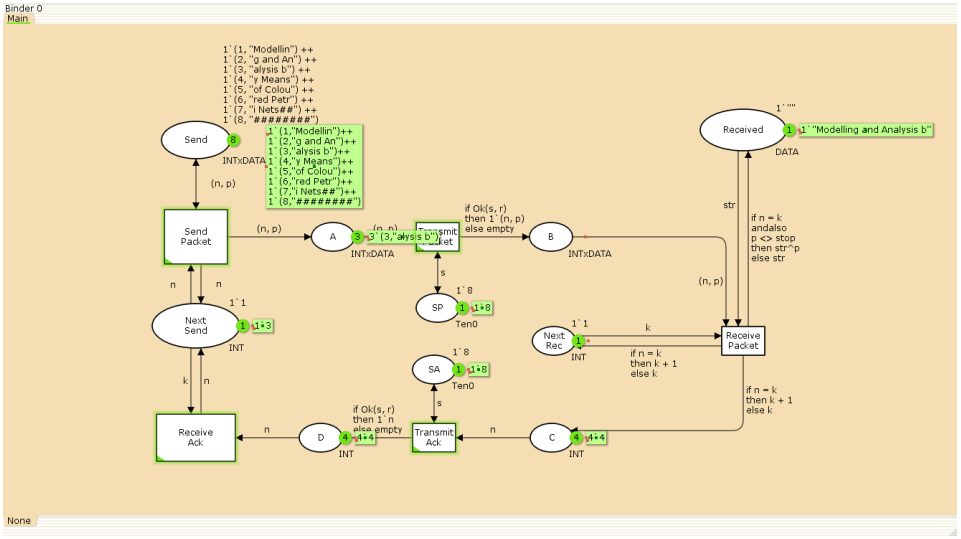


Рис. 3.6: Запуск модели простого протокола передачи данных (2)

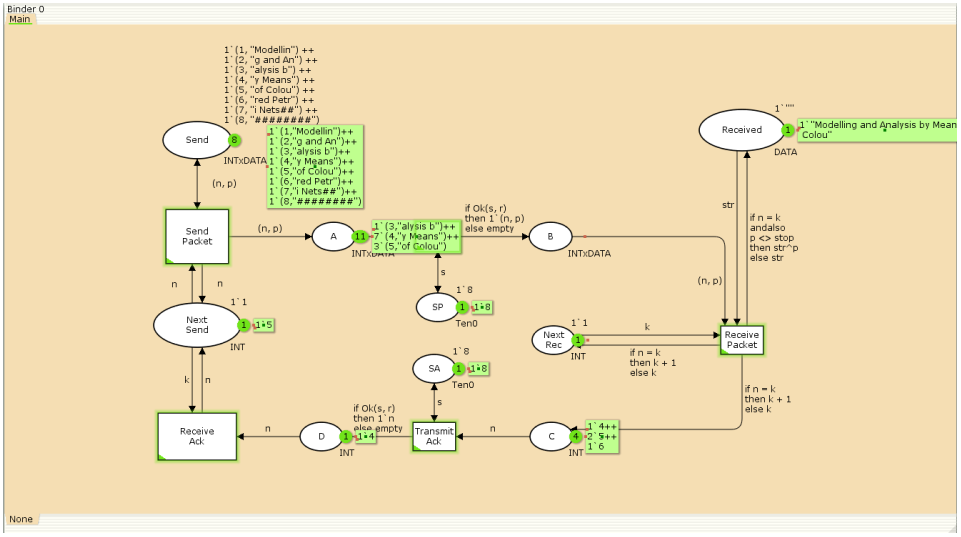


Рис. 3.7: Запуск модели простого протокола передачи данных (3)

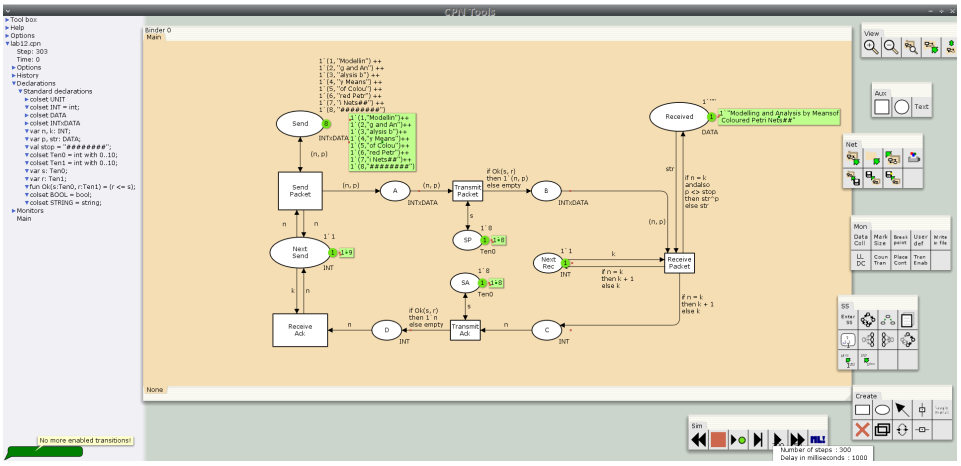

Рис. 3.8: Завершение моделирования простого протокола передачи данных

## 3.1 Упражнение

Вычислим пространство состояний. Прежде, чем пространство состояний может быть вычислено и проанализировано, необходимо сформировать код пространства состояний. Этот код создается, когда используется инструмент Войти в пространство состояний. Вход в пространство состояний занимает некоторое

время. Затем, если ожидается, что пространство состояний будет небольшим, можно просто применить инструмент Вычислить пространство состояний к листу, содержащему страницу сети. Сформируем отчёт о пространстве состояний и проанализируем его. Чтобы сохранить отчет, необходимо применить инструмент Сохранить отчет о пространстве состояний к листу, содержащему страницу сети и ввести имя файла отчета.

Из него можно увидеть:

- 13341 состояний и 206461 переходов между ними.
- Указаны границы значений для каждого элемента: промежуточные состояния A, B, C(наибольшая верхняя граница у A, так как после него пакеты отбрасываются. Так как мы установили максимум 10, то у следующего состояния B верхняя граница – 10), вспомогательные состояния SP, SA, NextRec, NextSend, Receiver(в них может находиться только один пакет) и состояние Send(в нем хранится только 8 элементов, так как мы задали их в начале и с ними никаких изменений не происходит).
- Указаны границы в виде мультимножеств.
- Маркировка home для всех состояний (в любую позицию можно попасть из любой другой маркировки).
- Маркировка dead равная 4675 [9999,9998,9997,9996,9995,...] – это состояния, в которых нет включенных переходов.

```
CPN Tools state space report for:
/home/openmodelica/protocol.cpn
Report generated: Sat May 25 21:02:31 2024


 Statistics
------------------------------------------------------------------------
```

State Space

Nodes: 13341

Arcs: 206461

Secs: 300

Status: Partial

Scc Graph

Nodes: 6975

Arcs: 170859

Secs: 14

Boundedness Properties

---

Best Integer Bounds

| | Upper | Lower |
|---|---|---|
| Main'A 1 | 20 | 0 |
| Main'B 1 | 10 | 0 |
| Main'C 1 | 6 | 0 |
| Main'D 1 | 5 | 0 |
| Main'NextRec 1 | 1 | 1 |
| Main'NextSend 1 | 1 | 1 |
| Main'Reciever 1 | 1 | 1 |
| Main'SA 1 | 1 | 1 |
| Main'SP 1 | 1 | 1 |
| Main'Send 1 | 8 | 8 |

Best Upper Multi-set Bounds

```
     Main'A 1            20`(1,"Modellin")++
15`(2,"g and An")++
9`(3,"alysis b")++
4`(4,"y Means ")
     Main'B 1            10`(1,"Modellin")++
7`(2,"g and An")++
4`(3,"alysis b")++
2`(4,"y Means ")
     Main'C 1            6`2++
5`3++
3`4++
1`5
     Main'D 1            5`2++
3`3++
2`4++
1`5
     Main'NextRec 1      1`1++
1`2++
1`3++
1`4++
1`5
     Main'NextSend 1     1`1++
1`2++
1`3++
1`4
     Main'Reciever 1     1`""++
1`"Modellin"++
1`"Modelling and An"++
1`"Modelling and Analysis b"++
```

```
1`"Modelling and Analysis by Means "
     Main'SA 1           1`8
     Main'SP 1           1`8
     Main'Send 1         1`(1,"Modellin")++
1`(2,"g and An")++
1`(3,"alysis b")++
1`(4,"y Means ")++
1`(5,"of Colou")++
1`(6,"red Petr")++
1`(7,"y Nets##")++
1`(8,"########")

  Best Lower Multi-set Bounds
     Main'A 1            empty
     Main'B 1            empty
     Main'C 1            empty
     Main'D 1            empty
     Main'NextRec 1      empty
     Main'NextSend 1     empty
     Main'Reciever 1     empty
     Main'SA 1           1`8
     Main'SP 1           1`8
     Main'Send 1         1`(1,"Modellin")++
1`(2,"g and An")++
1`(3,"alysis b")++
1`(4,"y Means ")++
1`(5,"of Colou")++
1`(6,"red Petr")++
1`(7,"y Nets##")++
```

```
1`(8,"########")


 Home Properties
--------------------------------------------------------------------------------

  Home Markings
     None


 Liveness Properties
--------------------------------------------------------------------------------

  Dead Markings
     4675 [9999,9998,9997,9996,9995,...]

  Dead Transition Instances
     None

  Live Transition Instances
     None


 Fairness Properties
--------------------------------------------------------------------------------
       Main'Recieved_Packet 1 No Fairness
       Main'Send_ACK 1        No Fairness
       Main'Send_Packet 1     Impartial
       Main'Transmit_ACK 1    No Fairness
```

```
Main'Transmit_Packet 1 Impartial
```

Сформируем начало графа пространства состояний, так как их много(рис. 3.9):

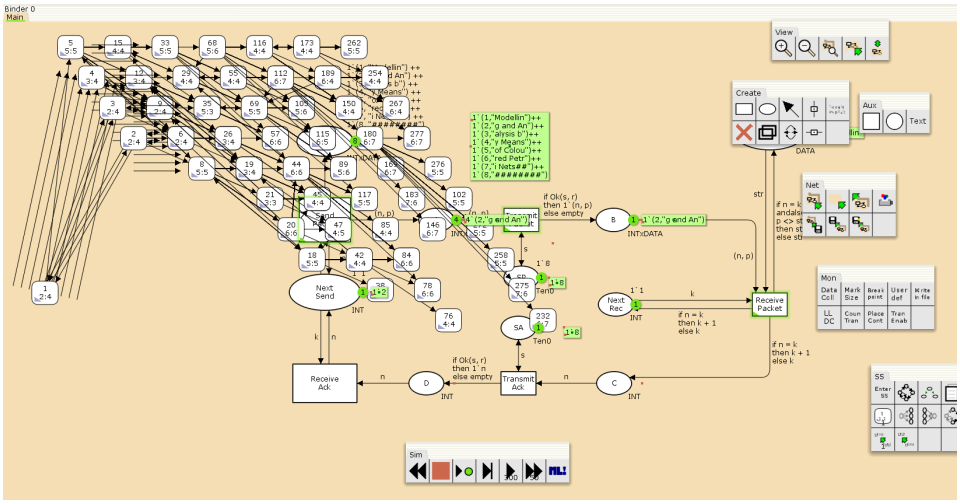


Рис. 3.9: Пространство состояний для модели простого протокола передачи данных

# 4 Выводы

В процессе выполнения данной лабораторной работы я реализовала простой протокол передачи данных в CPN Tools и провела анализ его пространства состояний.

# Список литературы


1. Королькова А.В., Кулябов Д.С. Моделирование информационных процессов: Лабораторная работа №12. РУДН, 2025.
2. Jensen K., Kristensen L.M., и др. CPN Tools: A tool for editing, simulating, and analysing Coloured Petri Nets. Aarhus University, Denmark, 2018.
3. Aly S., Mustafa K. Protocol Verification And Analysis Using Colored Petri Nets. DePaul University, 2003.
4. Westergaard M., Kristensen L.M. Two Interfaces to the CPN Tools Simulator // Proceedings of the 10th International Conference on Application of Concurrency to System Design (ACSD). 2011.