Лабораторная работа 12

Пример моделирования простого протокола передачи данных

Извекова Мария Петровна

Содержание

Список иллюстраций

Список таблиц

# Цель работы

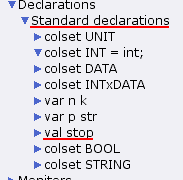
Реализовать простой протокол передачи данных в CPN Tools.

# Задание

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

# 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). Зададим декларации модели (рис. [-@fig:001]).

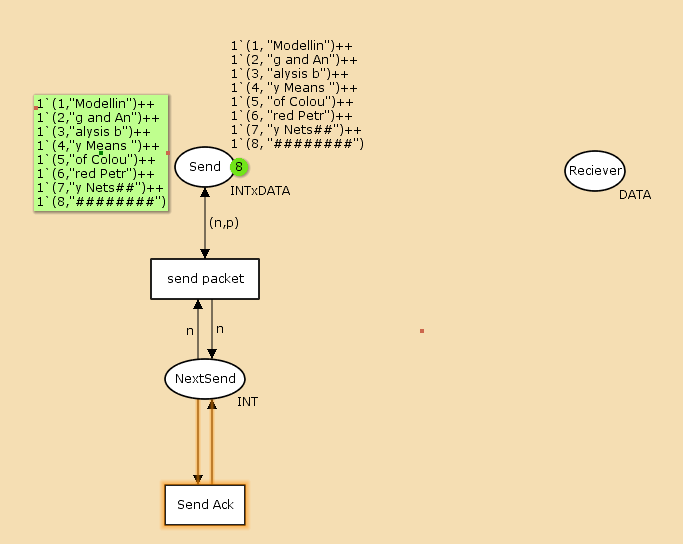


Задаем декларацию модели

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

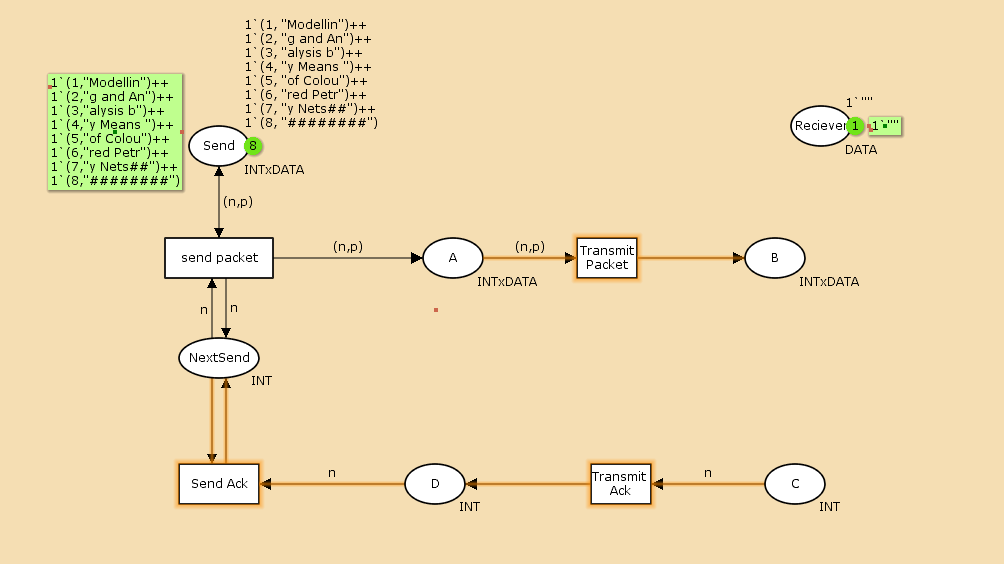
Стоповый байт (“########”) определяет, что сообщение закончилось. Состояние Receiver имеет тип DATA и начальное значение 1’“” (т.е. пустая строка, поскольку состояние собирает данные и номер пакета его не интересует). Состояние NextSend имеет тип INT и начальное значение 1’1. Поскольку пакеты представляют 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 (рис. 12.1). Также необходимо получать информацию с подтверждениями о получении данных. От перехода Send Packet к состоянию NextSend дуга с выражением n, обратно – k.

Построим начальный граф(рис. [-@fig:002]):



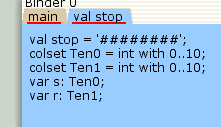
Начальный граф

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. [-@fig:003]):



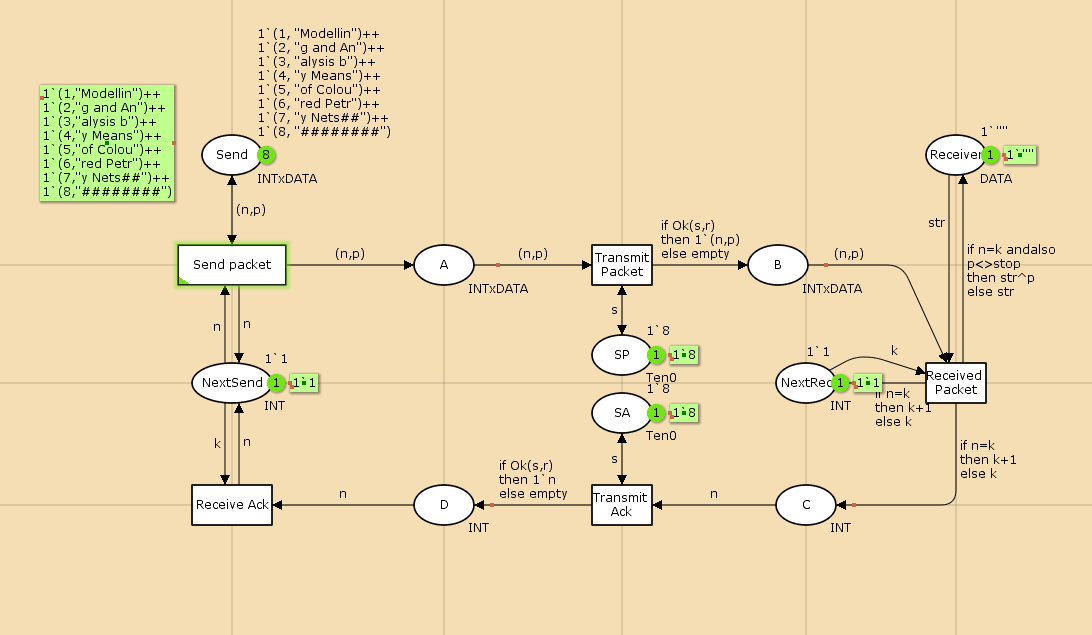
Граф со вспомогательными состояниями

В декларациях задаём(рис. [-@fig:004]):

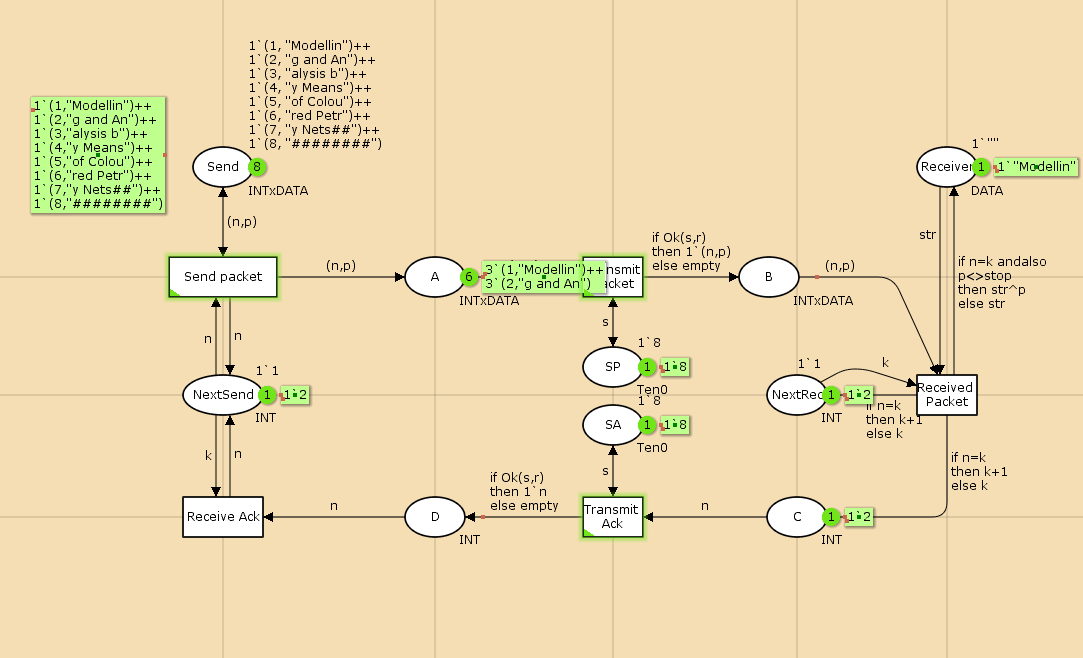


Дополнительная декларация

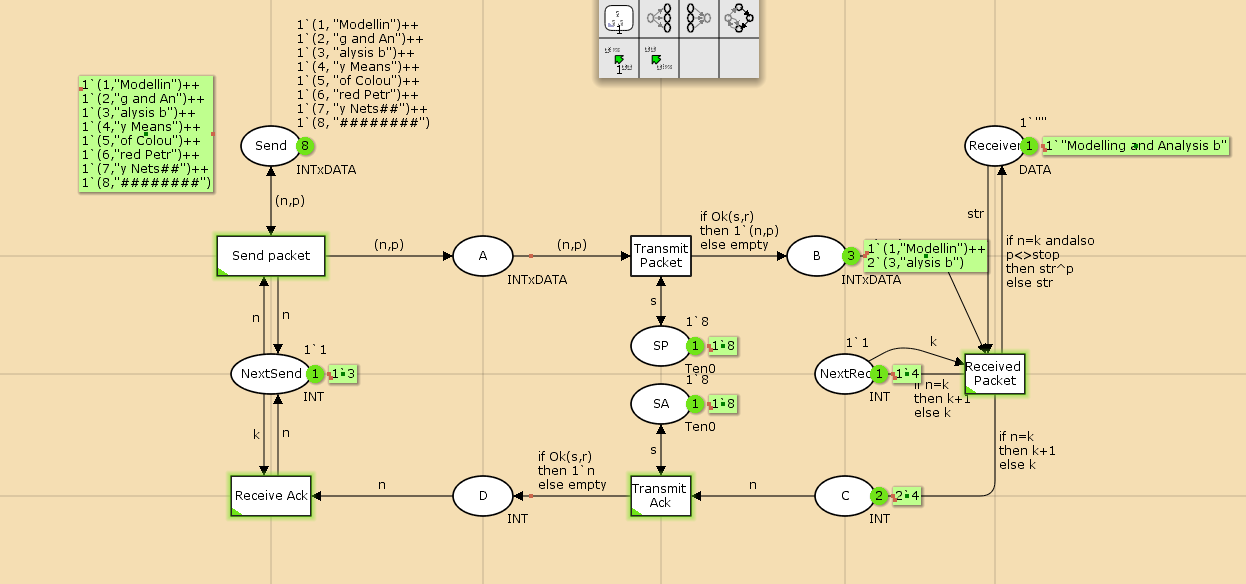
Таким образом, получим модель простого протокола передачи данных (рис. 12.3). Пакет последовательно проходит: состояние Send, переход Send Packet, состояние A, с некоторой вероятностью переход Transmit Packet, состояние B, попадает на переход Receive Packet, где проверяется номер пакета и если нет совпадения, 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. [-@fig:005]):



Готовая модель



Готовая модель в запущенном состоянии



# Упражнение

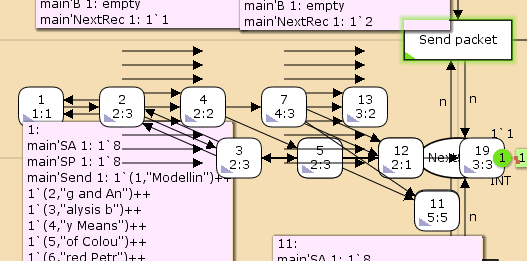
Вычислим пространство состояний. Прежде, чем пространство состояний может быть вычислено и проанализировано, необходимо сформировать код пространства состояний. Этот код создается, когда используется инструмент Войти в пространство состояний. Вход в пространство состояний занимает некоторое время. Затем,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

35452 состояний и 595954 переходов между ними. Указаны границы значений для каждого элемента: промежуточные состояния A, B, C, 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 12534 [35452,35451,35450,35449,35448,…] – это состояния, в которых нет включенных переходов.

CPN Tools state space report for:  
/home/openmodelica/new\_ptr.cpn  
Report generated: Sat Apr 26 10:49:24 2025  
  
  
 Statistics  
------------------------------------------------------------------------  
  
 State Space  
 Nodes: 35452  
 Arcs: 595954  
 Secs: 300  
 Status: Partial  
  
 Scc Graph  
 Nodes: 18609  
 Arcs: 501643  
 Secs: 5  
  
  
 Boundedness Properties  
------------------------------------------------------------------------  
  
 Best Integer Bounds  
 Upper Lower  
 main'A 1 23 0  
 main'B 1 11 0  
 main'C 1 7 0  
 main'D 1 5 0  
 main'NextRec 1 1 1  
 main'NextSend 1 1 1  
 main'Receiver 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 23`(1,"Modellin")++  
17`(2,"g and An")++  
12`(3,"alysis b")++  
7`(4,"y Means")++  
2`(5,"of Colou")  
 main'B 1 11`(1,"Modellin")++  
8`(2,"g and An")++  
6`(3,"alysis b")++  
3`(4,"y Means")++  
1`(5,"of Colou")  
 main'C 1 7`2++  
6`3++  
4`4++  
2`5  
 main'D 1 5`2++  
4`3++  
3`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++  
1`5  
 main'Receiver 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'Receiver 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  
 12534 [35452,35451,35450,35449,35448,...]  
  
 Dead Transition Instances  
 None  
  
 Live Transition Instances  
 None  
  
  
 Fairness Properties  
------------------------------------------------------------------------  
 main'Receive\_Ack 1 No Fairness  
 main'Received\_Packet 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. [-@fig:007]):



Граф состояний

# Вывод

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

# Библиография

1. Зайцев Д. А., Шмелева Т. Р. Моделирование телекоммуникационных систем в CPN Tools. — Одесса : Одесская национальная академия связи им. А.С. Попова,
2. CPN Tool. — 2014. — URL: http://cpntools.org.