

Лабораторная работа №12

Задание для самостоятельного выполнения

Игнатенкова Варвара Николаевна

Содержание

1	Цель работы	1
2	Задание	1
3	Выполнение лабораторной работы.....	1
4	Выводы	10

1 Цель работы

Реализовать ненадёжную сеть передачи данных, состоящую из источника, получателя в CPN tools.

2 Задание

- Реализовать в CPN Tools ненадёжную сеть передачи данных, состоящую из источника, получателя.
- Вычислить пространство состояний. Сформировать отчёт о пространстве состояний и проанализировать его. Построить граф пространства состояний.

3 Выполнение лабораторной работы

Постановка задачи

Рассмотрим ненадёжную сеть передачи данных, состоящую из источника получателя. Перед отправкой очередной порции данных источник должен получить от получателя подтверждение о доставке предыдущей порции данных. Считаем, что пакет состоит из номера пакета и строковых данных. Передавать будем сообщение «Modelling and Analysis by Means of Coloured Petry Nets», разбитое по 8 символов.

Построение модели с помощью CPNTools

Зададим декларации модели:

```
Declarations
  colset INT = int;
  colset DATA = string;
  colset INTxDATA = product INT * DATA;
  var n, k: INT;
  var p, str: DATA;
  val stop = "#####";
Monitors
  12
```

Рисунок 1 Декларации модели

Зададим начальный граф:

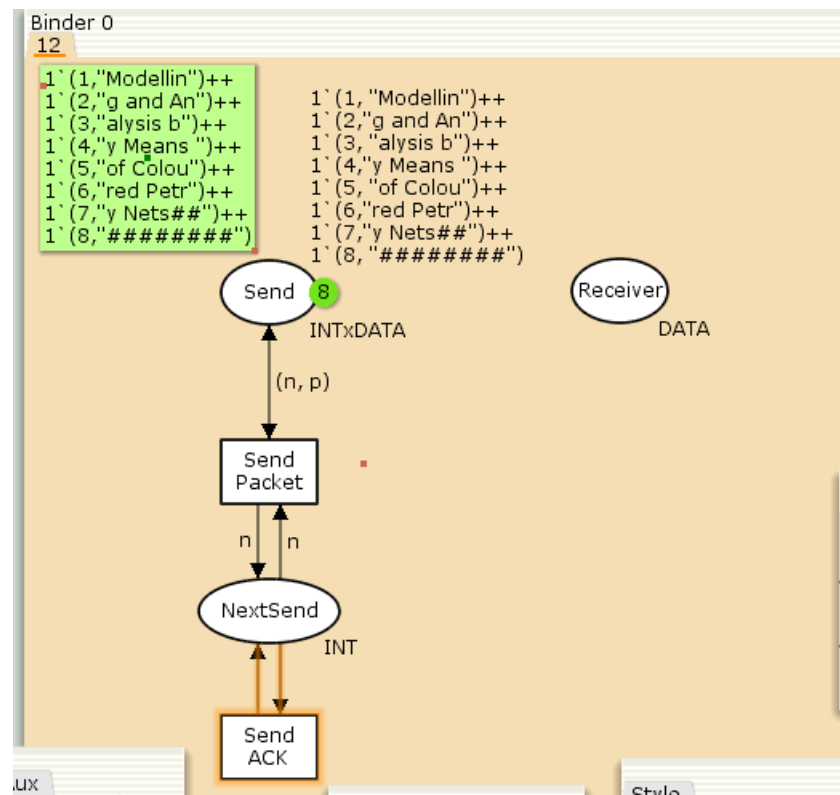


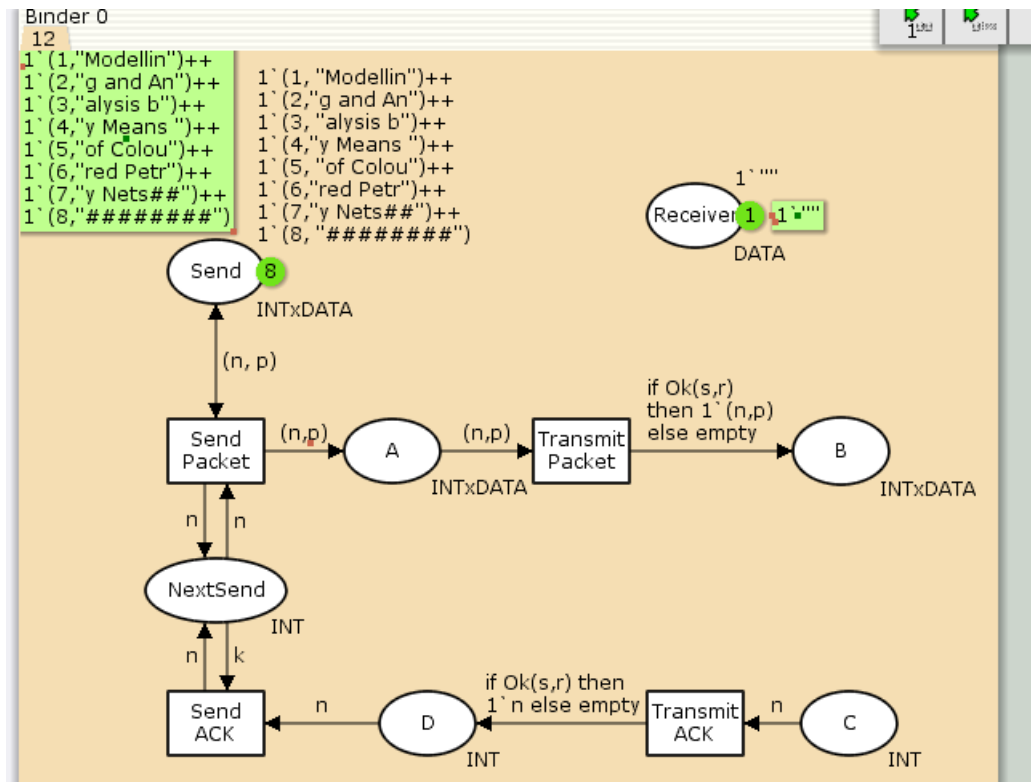
Рисунок 2 Начальный граф:

Состояние Send имеет тип INTxDATA и 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).

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 (А, В с типом INTxDATA, С, D с типом INTxDATA) для переходов: передать пакет Transmit Packet (передаём (n,p)), передать подтверждение Transmit ACK (передаём целое число k).

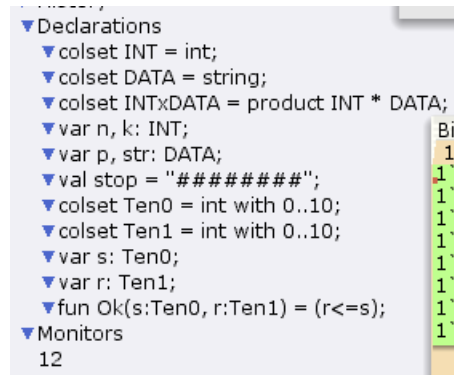
От состояния Receiver идёт дуга к переходу Receive Packet со значением той строки (str), которая находится в состоянии Receiver. Обратно: проверяем, что номер пакета новый и строка не равна стоп-биту. Если это так, то строку добавляем к полученным данным.

Связываем состояния В и С с переходом Receive Packet. От состояния В к переходу Receive Packet — выражение (n, p) , от перехода Receive Packet к состоянию С — выражение $\text{if } n=k \text{ then } k+1 \text{ else } k$.

От перехода Receive Packet к состоянию Receiver: if $n=k$ and also $p \neq \text{stop}$ then $\text{str} \leftarrow p$ 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, соединяем с соответствующими переходами.

Задаем декларации:



```

▼ Declarations
  ▼ colset INT = int;
  ▼ colset DATA = string;
  ▼ colset INTxDATA = product INT * DATA;
  ▼ var n, k: INT;
  ▼ var p, str: DATA;
  ▼ val stop = "#####";
  ▼ colset Ten0 = int with 0..10;
  ▼ colset Ten1 = int with 0..10;
  ▼ var s: Ten0;
  ▼ var r: Ten1;
  ▼ fun Ok(s:Ten0, r:Ten1) = (r<=s);
▼ Monitors
  12
  
```

Рисунок 4 Декларации

Задаём выражение от перехода Transmit Packet к состоянию B и выражение от перехода Transmit ACK к состоянию D.

Таким образом, получим модель простого протокола передачи данных.

The diagram illustrates the Stop-and-Wait protocol with the following components and transitions:

- States and Buffers:**
 - Send:** Initial state, buffer size 8.
 - Send Packet:** Buffer size 1, 1:1.
 - Transmit Packet:** Buffer size 1, 1:8.
 - NextSend:** Buffer size 1, 1:1.
 - NextRec:** Buffer size 1, 1:1.
 - Received Packet:** Buffer size 1, 1:1.
 - Transmit ACK:** Buffer size 1, 1:8.
 - SA (Sender Acknowledgment):** Buffer size 1, 1:8.
- Transitions:**
 - Send to Send Packet:** Event: (n, p) , Data: $INT \times DATA$.
 - Send Packet to A:** Event: (n, p) , Data: $INT \times DATA$.
 - A to Transmit Packet:** Event: s .
 - Transmit Packet to B:** Event: (n, p) , Data: $INT \times DATA$.
 - B to NextRec:** Event: k , Data: INT .
 - NextRec to Received Packet:** Event: $n = k$ then $k+1$ else k , Data: INT .
 - Received Packet to C:** Event: $n = k$ then $k+1$ else k , Data: INT .
 - C to Transmit ACK:** Event: n , Data: INT .
 - Transmit ACK to D:** Event: s .
 - D to Send ACK:** Event: n , Data: INT .
 - Send ACK to NextSend:** Event: n , Data: INT .
 - NextSend to Send Packet:** Event: n , Data: INT .
 - Send Packet to NextSend:** Event: n , Data: INT .
 - Transmit Packet to NextRec:** Event: $Ok(s, r)$ then $1' (n, p)$ else empty, Data: $INT \times DATA$.
 - Received Packet to Receiver:** Event: str , Data: $DATA$.
 - Receiver to NextRec:** Event: $n = k$ and also $p \neq stop$ then $str \wedge p$ else str , Data: $DATA$.

отчет, необходимо применить инструмент Сохранить отчет о пространстве состояний к листу, содержащему страницу сети и ввести имя файла отчета.

Statistics

State Space

Nodes: 13341

Arcs: 206461

Secs: 300

Status: Partial

Scc Graph

Nodes: 6975

Arcs: 170859

Secs: 14

Boundedness Properties

Best Integer Bounds

	<i>Upper</i>	<i>Lower</i>
<i>Main'A 1</i>	<i>20</i>	<i>0</i>
<i>Main'B 1</i>	<i>10</i>	<i>0</i>
<i>Main'C 1</i>	<i>6</i>	<i>0</i>
<i>Main'D 1</i>	<i>5</i>	<i>0</i>
<i>Main'NextRec 1</i>	<i>1</i>	<i>1</i>
<i>Main'NextSend 1</i>	<i>1</i>	<i>1</i>
<i>Main'Reciever 1</i>	<i>1</i>	<i>1</i>
<i>Main'SA 1</i>	<i>1</i>	<i>1</i>

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 все они не поместятся.

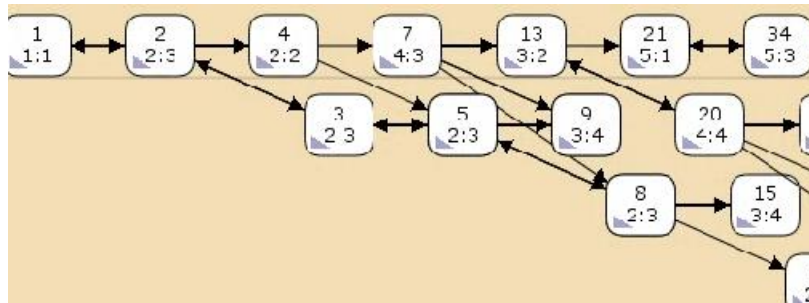


Рисунок 6 Граф пространства состояний

4 Выводы

В процессе выполнения данной лабораторной работы я реализовала модель простого протокола передачи данных.