Лабораторная работа 12. Пример моделирования простого протокола передачи данных

12.1. 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.

12.2. Построение модели с помощью CPNTools

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:

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

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

```
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,"########")
```

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`.

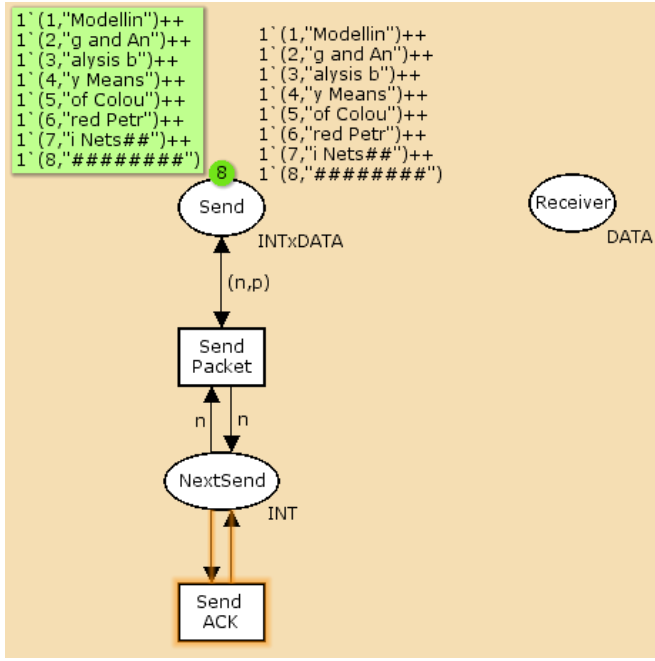

Рис. 12.1. Начальный граф

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 толко 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

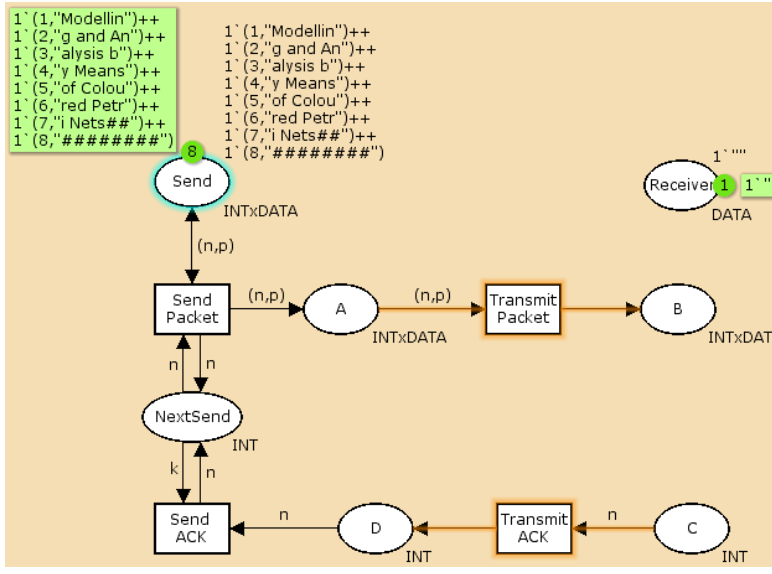

Рис. 12.2. Добавление промежуточных состояний

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.

В декларациях задаём:

```
colset  Ten0 = int with 0..10;
colset  Ten1 = int with 0..10;
var s: Ten0;
var r: Ten1;
```

и определяем функцию (если нет превышения порога, то истина, если нет — ложь):

```
fun Ok(s:Ten0, r:Ten1)=(r<=s);
```

Задаём выражение от перехода `Transmit Packet` к состоянию `B`:

```
if Ok(s,r) then 1`(n,p) else empty
```

Задаём выражение от перехода `Transmit ACK` к состоянию `D`:

```
if Ok(s,r) then 1`n else empty
```

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.

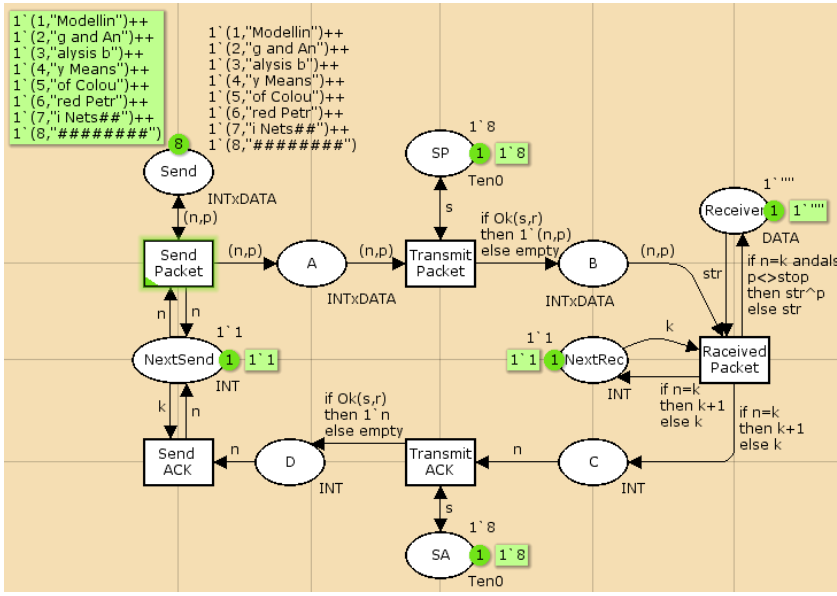

Рис. 12.3. Модель простого протокола передачи данных

Упражнение. Вычислите пространство состояний. Сформируйте отчёт о пространстве состояний и проанализируйте его. Постройте граф пространства состояний.