

# Лабораторная работа 12

Пример моделирования простого протокола передачи данных

---

Акопян Сатеник

01 января 1940

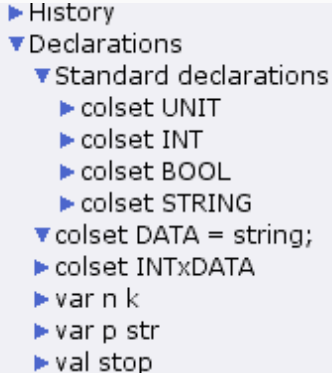
Российский университет дружбы народов, Москва, Россия

Объединённый институт ядерных исследований, Дубна, Россия

Смоделировать простой протокол передачи данных.

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.

1. Зададим декларации модели (рис. (fig:001?)).

A screenshot of a code editor with a light blue background. The code is written in a dark font and uses blue arrow icons for indentation. The code defines a model with various declarations.

```
▶ History
▼ Declarations
  ▼ Standard declarations
    ▶ colset UNIT
    ▶ colset INT
    ▶ colset BOOL
    ▶ colset STRING
  ▼ colset DATA = string;
  ▶ colset INTxDATA
  ▶ var n k
  ▶ var p str
  ▶ val stop
```

Рис. 1: декларации модели

- 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 11. 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. (fig:002?))

## Выполнение лабораторной работы

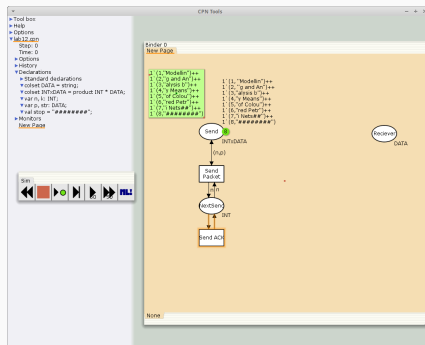


Рис. 2: Начальный граф

3. 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 NextRes 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 —  $\text{if } n=k \text{ then } k+1 \text{ else } k$ . 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:  $\text{if } n=k \text{ and also } p \neq \text{stop then } str^p \text{ else } str$



## Выполнение лабораторной работы

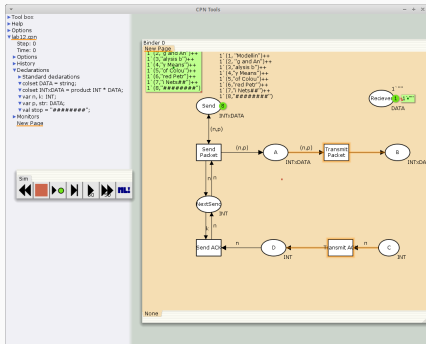


Рис. 3: Добавление промежуточных состояний

4. 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.

В декларациях задаём:

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

(рис. (fig:004?))

Таким образом, получим модель простого протокола передачи данных (рис. (fig:005?)).

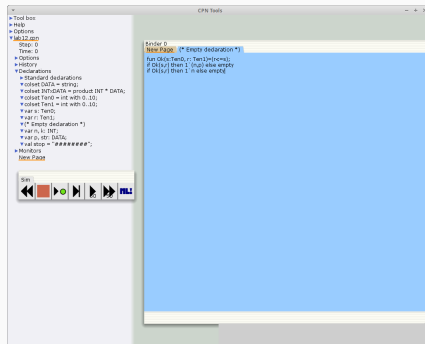


Рис. 4: Декларации модели

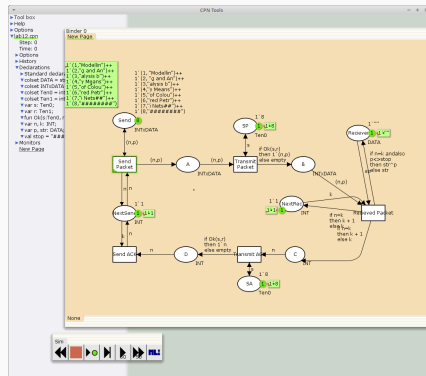


Рис. 5: Модель простого протокола передачи данных

В результате был смоделирован простой протокол передачи данных.