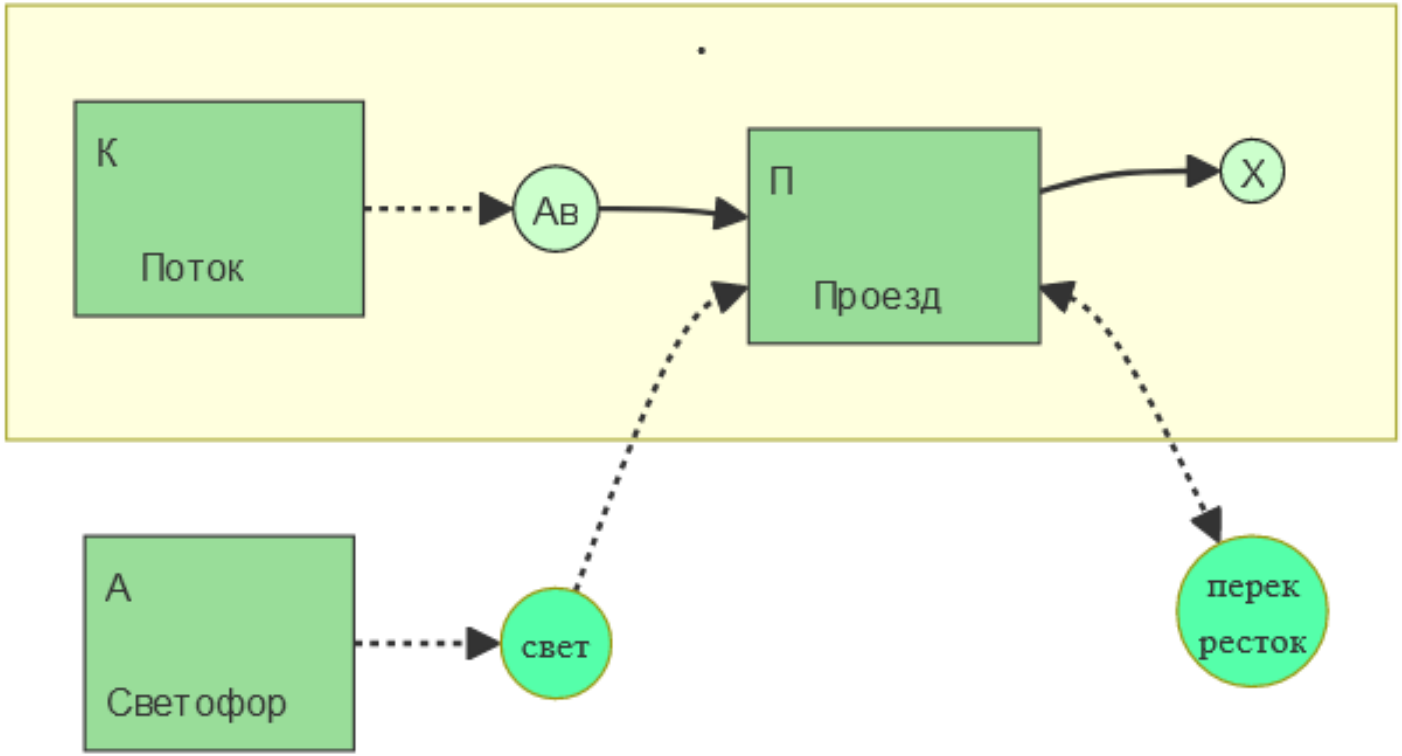
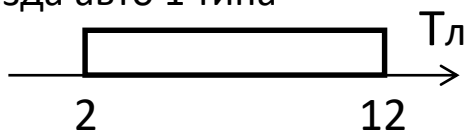
Структурная организация имитационных моделей дискретных процессов

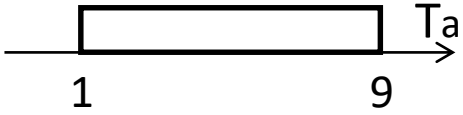
Блочная схема модели процесса проезда на перекрестке



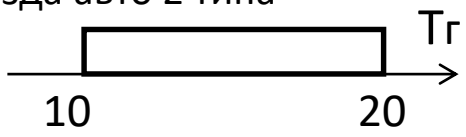
Время проезда авто 1 типа



Время появления авто



Время проезда авто 2 типа



Модель процесса проезда на перекрестке (2)

```
Блок-агрегат СВЕТОФОР
описание
 Тс, СВЕТ – скаляр; ЦИКЛ – метка;
 // Тс – длительность одного состояния
 // начальное положение ИНИЦИАТОРА на метке ЦИКЛ
всё описание;
алгоритм
 ЦИКЛ: СВЕТ := «зеленый»;
       Тс := ВРЕМЯ + 30;
       ждать ВРЕМЯ = Тс;
       СВЕТ := «жёлтый»;
       Тс := ВРЕМЯ + 10;
       ждать ВРЕМЯ = Тс;
       СВЕТ := «красный»;
       Тс := ВРЕМЯ + 30;
       ждать ВРЕМЯ = Тс;
       направить ИНИЦИАТОР на метку ЦИКЛ;
всё алгоритм;
всё блок.
```

Модель процесса проезда на перекрестке (1)

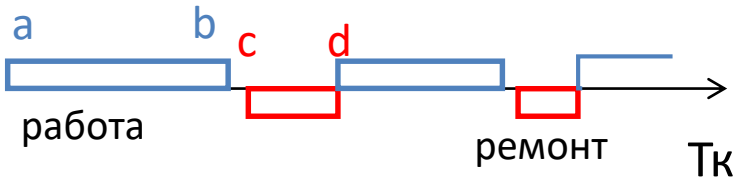
```
Блок-контроллер ПОТОК
описание
 Та – скаляр;   НАЧ – метка;
 // начальное положение ИНИЦИАТОРА на метке НАЧ
всё описание;
алгоритм
 НАЧ: создать ЛС типа вектор(1-2 – скаляры);
      ЛС(1) := ЦЕЛОЕ(RAND + 0.5) + 1; // ТИП авто :целое= 1 или 2
      ЛС(2) := ВРЕМЯ; // запомнить время появления
      создать Авто типа ссылка;
      Авто := ссылка на вектор ЛС;
      активизировать инициатор из Авто в блок ПРОЕЗД на метку ВЪЕЗД;
      Та := ВРЕМЯ + (RAND * 8 + 1);
      ждать ВРЕМЯ = Та;
      направить ИНИЦИАТОР на метку НАЧ;
всё алгоритм;
всё блок.
```

Модель процесса проезда на перекрестке (3)

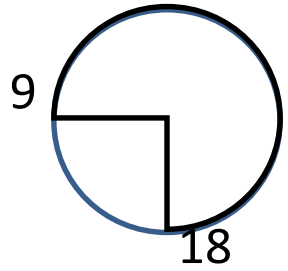
```
Блок-процессор ПРОЕЗД
описание
 ПЕРЕКРЕСТОК – скаляр; // начальное состояние «свободен»
 СВЕТ – скаляр в блоке СВЕТОФОР;
 Тпр, N, СВ – скаляры; // начальное состояние 0
 ВЪЕЗД, ПРЕ – метка;
всё описание;
алгоритм
 ВЪЕЗД: ждать (СВЕТ = «зеленый»).И.(ПЕРЕКРЕСТОК = «свободен»);
        ПЕРЕКРЕСТОК := «занят»;
        Тпр := ВРЕМЯ + (RAND * 10 + 2); //время проезда
        если ИНИЦИАТОР->вектор(1) = 1 то направить ИНИЦИАТОР
                                          на метку ДВЖ;
        Тпр := ВРЕМЯ + (RAND * 10 + 10); //перерасчет времени
  ДВЖ: ждать ВРЕМЯ = Тпр;
        ПЕРЕКРЕСТОК := «свободен»;
        N := N + 1; // счетчик проехавших автомобилей -статистика
        СВ := СВ + (ВРЕМЯ – ИНИЦИАТОР->вектор(2));
         // длительность занятости перекрёстка
        уничтожить ИНИЦИАТОР;
всё алгоритм;
всё блок.
```

Задача: процесс обслуживания компьютеров

В дата-центре работают круглосуточно 60 компьютеров, длительность их работоспособного состояния известна в интервале от **a** до **b** часов. После поломки компьютер поступает на ремонт к мастеру. На ремонт одного компьютера он тратит от **c** до **d** часов. Мастер работает с 9 до 18 часов.

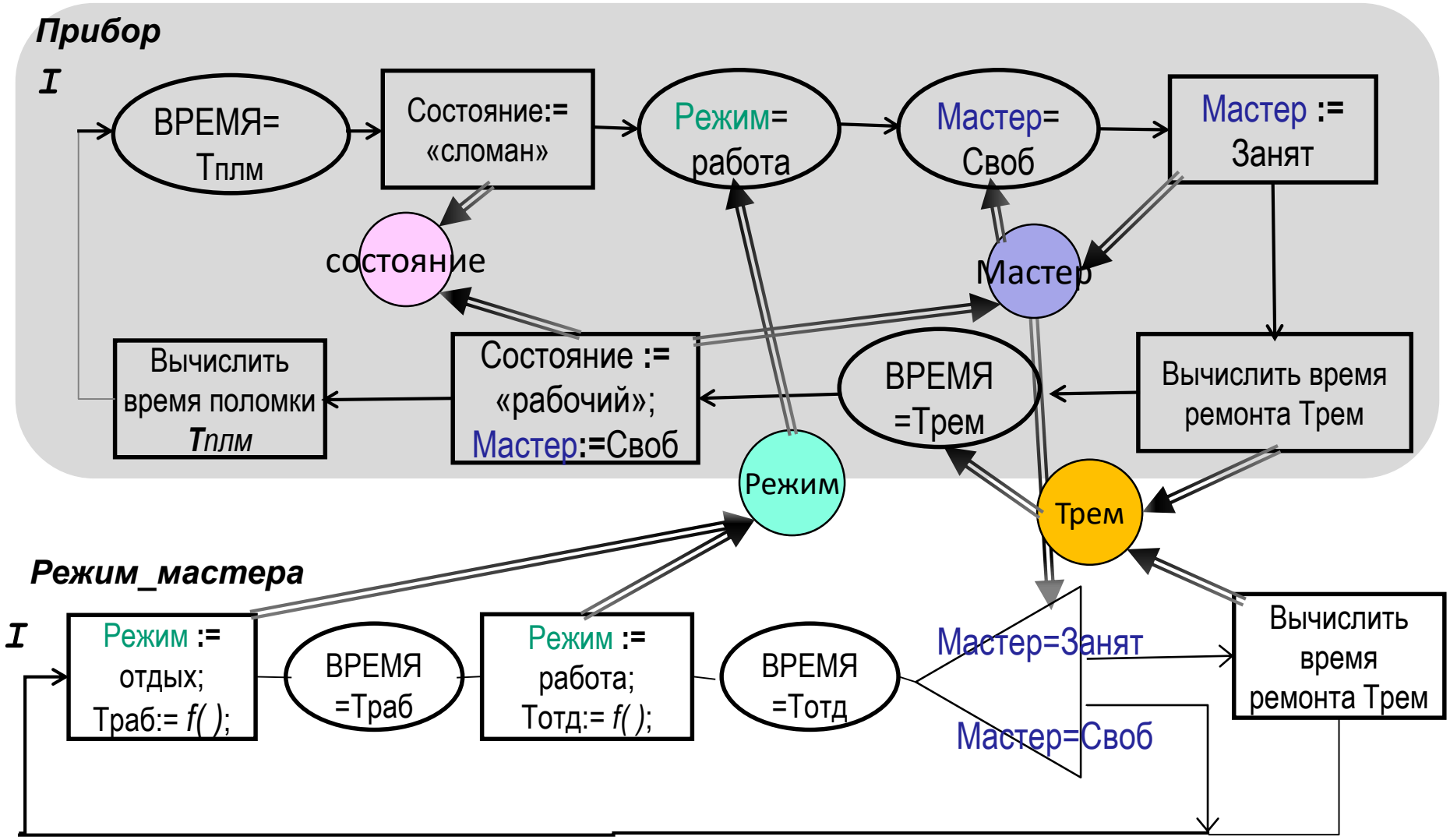


Временная диаграмма жизни компьютера

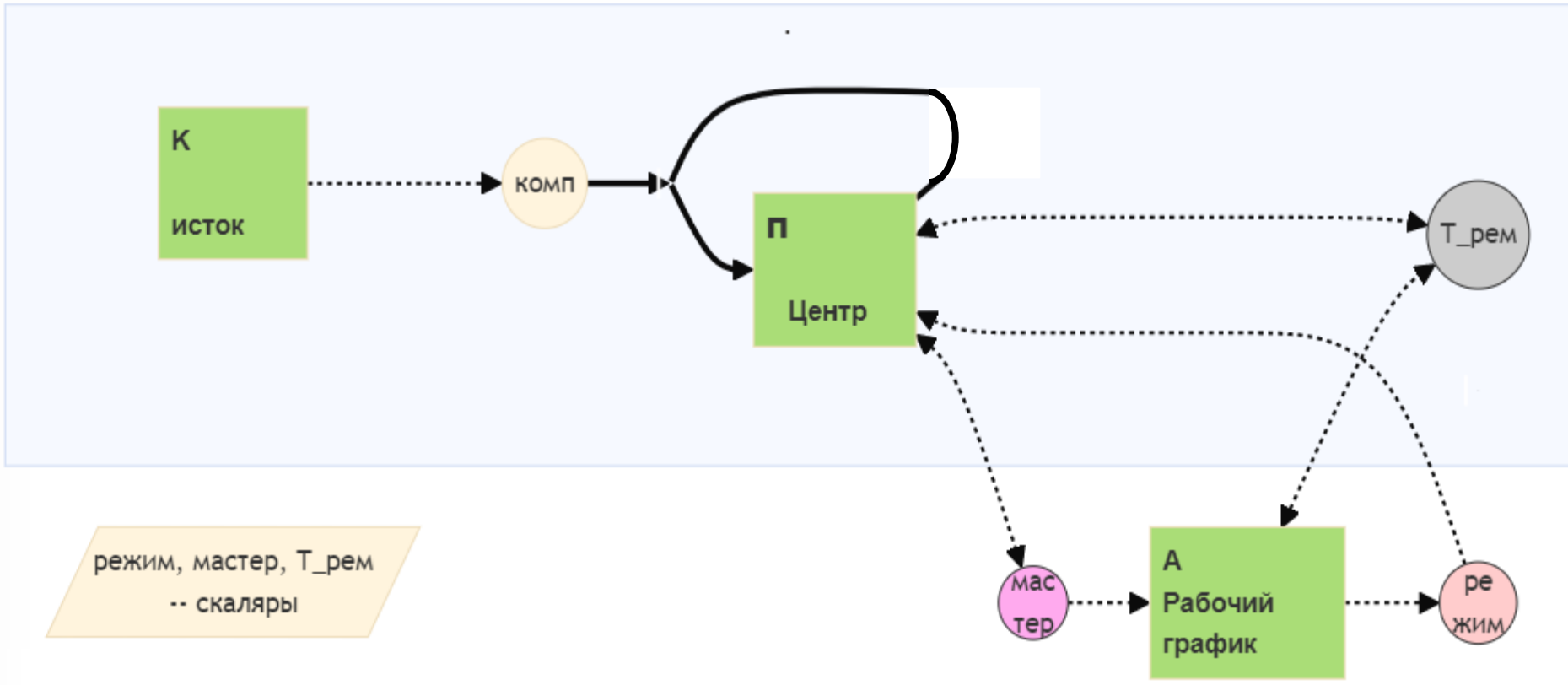


Режим работы ремонтника

Операторно-параметрическая схема (ОПС) модели процесса ремонта оборудования



Блочная схема модели процесса обслуживания компьютеров в дата-центре



Модель процесса обслуживания компьютеров в ВЦ (1)

```
Блок-контроллер Исток
описание
 N – скаляр; // начальное значение 0
 НАЧАЛО – метка;
 // начальное положение ИНИЦИАТОРА на метке НАЧАЛО
Всё описание;
алгоритм
 НАЧАЛО: создать W типа вектор(1,2 --скаляры);
   W(1) := N + 1;
   создать S типа ссылка;
   S := ссылка на вектор W;
   активизировать инициатор из S в блок ЦЕНТР на метку СТАРТ;
   N := N + 1;
   если N < 60 то направить ИНИЦИАТОР на метку НАЧАЛО;
   уничтожить ИНИЦИАТОР;
Всё алгоритм;
Всё блок.
```

Модель процесса обслуживания компьютеров в ВЦ (2)

```
Блок-агрегат Рабочий_график
описание
 Режим – скаляр;
 Мастер, Трем – скаляры в блоке ЦЕНТР;
 ЦИКЛ – метка;
 // начальное положение ИНИЦИАТОРА на метке ЦИКЛ
всё описание;
алгоритм
ЦИКЛ: РЕЖИМ := «не работает»;
    ждать ВРЕМЯ % 24 = 9; // % -оператор деления по модулю
    РЕЖИМ := «работает»;
    ждать ВРЕМЯ % 24 = 18;
    если МАСТЕР = «свободен» то направить ИНИЦИАТОР на метку ЦИКЛ;
    Трем := Трем + 15;
    направить ИНИЦИАТОР на метку ЦИКЛ;
всё алгоритм;
всё блок.
```

Модель процесса обслуживания компьютеров в ВЦ (3)

```
Блок-процессор ЦЕНТР
описание
 МАСТЕР – скаляр; // начальное состояние «свободен»
 РЕЖИМ – скаляр в блоке Рабочий_график;
 Трем, a, b, c, d – скаляры;
 СТАРТ – метка;
всё описание;
алгоритм
 СТАРТ: (ИНИЦИАТОР->вектор(2)) := ВРЕМЯ + (RAND * (b - a) + b);
        ждать ВРЕМЯ = ИНИЦИАТОР->вектор(2); //время поломки
        ждать (МАСТЕР = «свободен»).И.(РЕЖИМ = «работает»);
        МАСТЕР := «занят»;
        Трем := ВРЕМЯ + (RAND * (d – c) + c); //время ремонта
        ждать ВРЕМЯ = Трем;
        МАСТЕР := «свободен»;
        направить ИНИЦИАТОР на метку СТАРТ;
всё алгоритм;
всё блок.
```

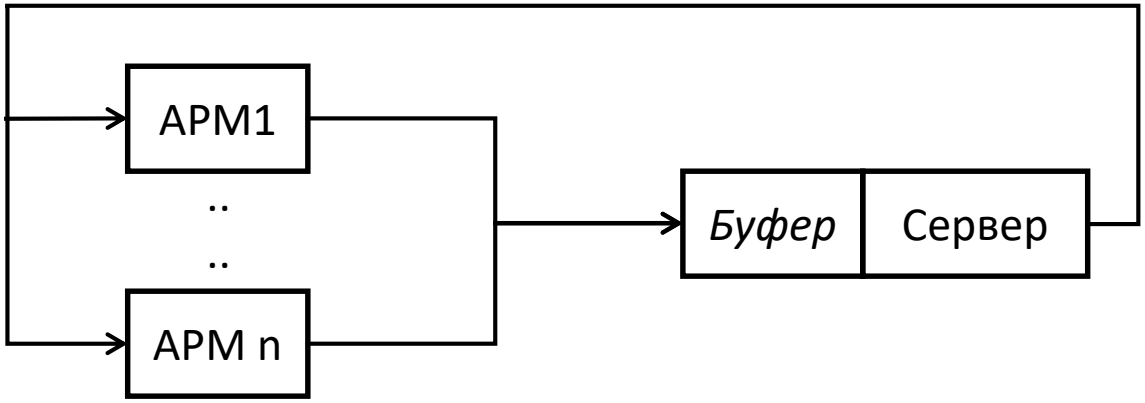
Задача: Клиент-серверная система

Сеть состоит из 20 рабочих станций и одного сервера, работающих в режиме диалога (запрос-ответ-запрос-…).

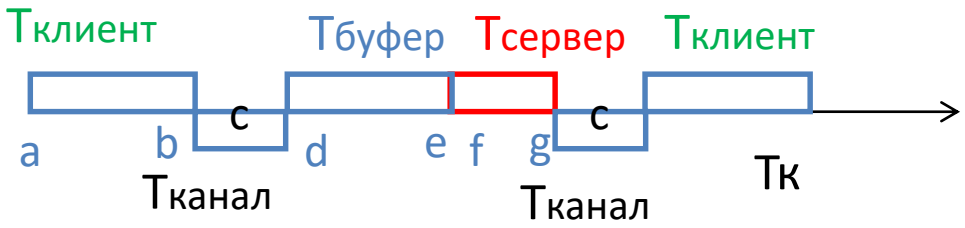
Сервер обрабатывает запрос и отсылает на рабочую станцию ответ. Время формирования ответа зависит от сложности запроса.

Время передачи запросов/ответов по сети задано.

Оценка интервала формирования запросов задана.

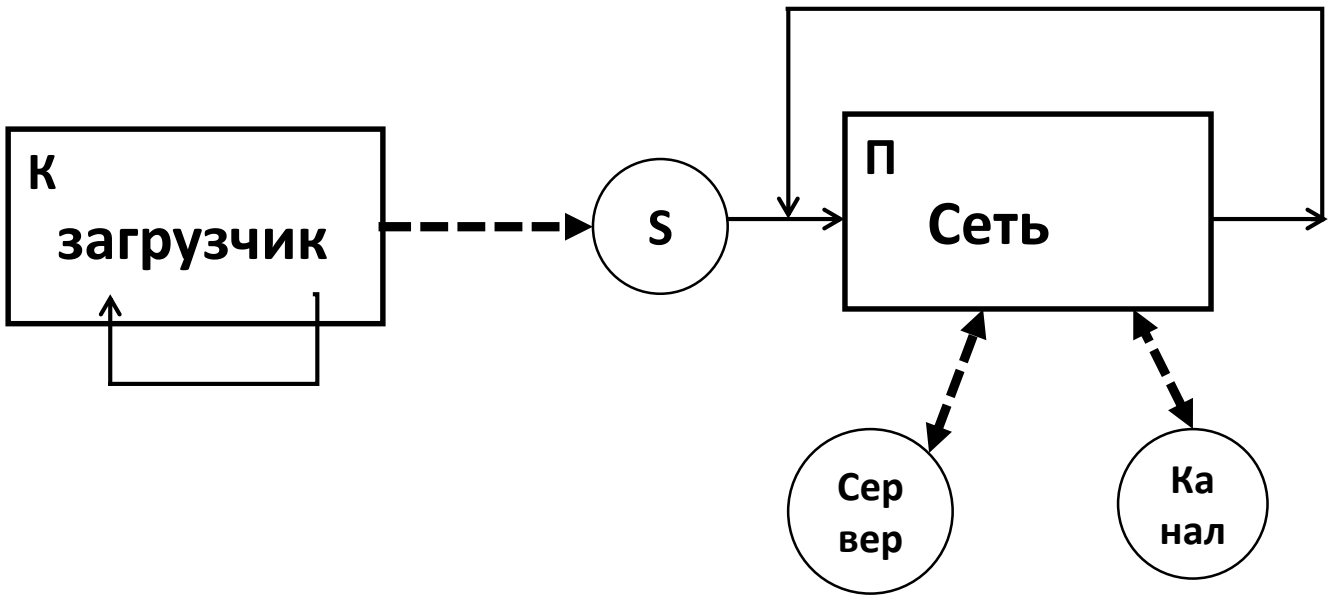


Блочная схема модели процесса обслуживания запросов «клиент-сервер»



Временая диаграмма жизни запроса

Например, в сети
20 клиентов
и 10 уровней
сложности запросов



Модель процесса обслуживания запросов «клиент-сервер» (1)

```
Блок-контроллер Загрузчик
описание
 N – скаляр; // начальное значение 0
 НАЧ – метка;
 // начальное положение ИНИЦИАТОРА на метке НАЧ
Всё описание;
алгоритм
НАЧ:  N := N + 1;
      создать W типа вектор(1,2,3 -скаляры);
      W(1) := N;
      W(2) := ЦЕЛОЕ(RAND * 9) + 1; // коэффициент сложности
      создать S типа ссылка;
      S := ссылка на вектор W;
      активизировать инициатор из S в блок СЕТЬ на метку СТАРТ;
      если N < 20 то направить ИНИЦИАТОР на метку НАЧ;
      уничтожить ИНИЦИАТОР;
Всё алгоритм;
Всё блок.
```

Модель процесса обслуживания запросов «клиент-сервер» (2)

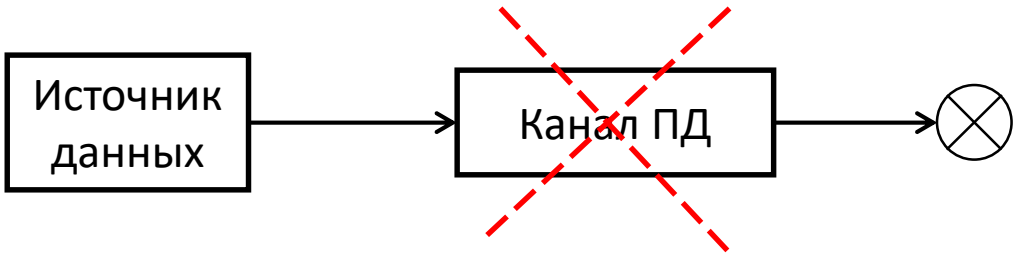
```
Блок-процессор СЕТЬ
описание
 СЕРВ, Канал, Канал1 – скаляры; // начальное состояние «свободен»
 Тс, Тк, То, a, b, g, f – скаляры; СТАРТ – метка;
Всё описание;
алгоритм
СТАРТ: (ИНИЦИАТОР->вектор(3)) := ВРЕМЯ + (RAND * (b-a) + a);
 ждать ВРЕМЯ = ИНИЦИАТОР->вектор(3); // подготовка запроса
 ждать Канал = «свободен»;
 Канал := «занят»;              Тк := ВРЕМЯ + Тс;
 ждать ВРЕМЯ = Тк;     // Тс–время передачи сообщения по каналу
 Канал := «свободен»;
 ждать СЕРВ = «свободен»;   СЕРВ := «занят»;
 Тсерв := ВРЕМЯ + (RAND*(g-f)+f) * (ИНИЦИАТОР->вектор(2));
 ждать_ВРЕМЯ = Тсерв;          СЕРВ := «свободен»;
 ждать Канал1 = «свободен»;
 Канал1 := «занят»;             То := ВРЕМЯ + Тс;
 ждать ВРЕМЯ = То;     // передача ответа клиенту
 Канал1 := «свободен»;
 направить ИНИЦИАТОР на метку СТАРТ;
Всё алгоритм;
Всё блок.
```

Задача: канал передачи данных

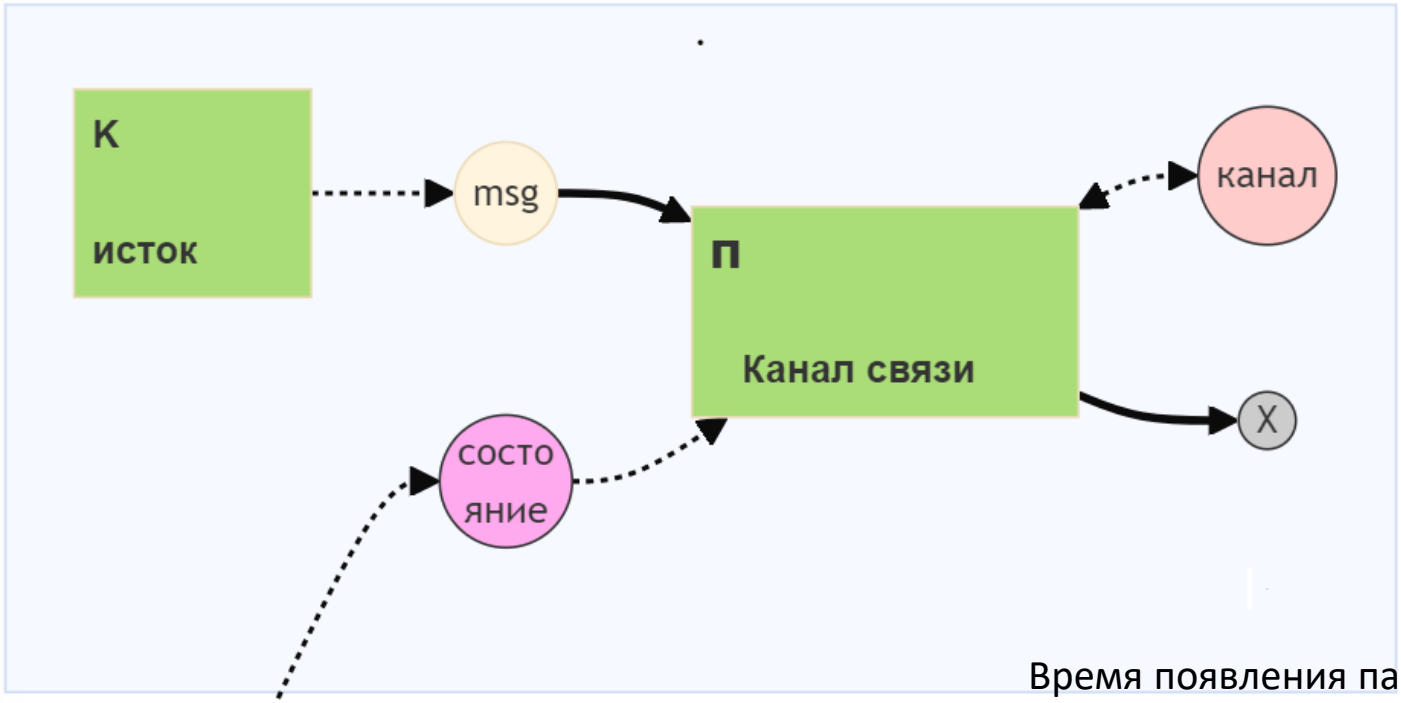
Поток сообщений разного размера поступает на канал связи.

Канал связи тратит на передачу сообщения время пропорциональное размеру данных.

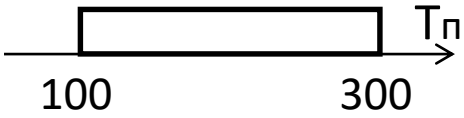
На канале периодически возникают поломки, время устранения поломки задано, но сообщения в канале не теряются.



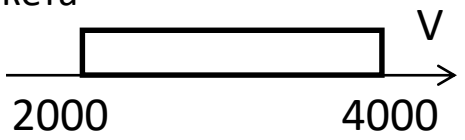
Блочная схема модели процесса передачи данных в канале



Время появления пакета

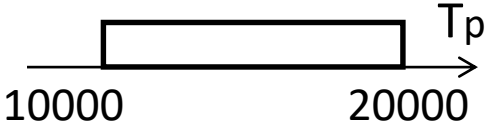


Размер пакета



W — скорость передачи

Время работоспособности



Модель процесса передачи данных в канале (1)

```
Блок-контроллер Исток
описание
 Тп – скаляр; НАЧ – метка;
 // начальное положение ИНИЦИАТОРА на метке НАЧ
Всё описание;
алгоритм
 НАЧ: создать V типа вектор(1-3 - скаляры);
   V(1) := ВРЕМЯ;
   V(2) := RAND * 2000 + 2000; // размер пакета
   создать msg типа ссылка;
   msg := ссылка на вектор V;
   активизировать инициатор из msg в блок КаналСвязи на метку Старт;
   Тп := ВРЕМЯ + (RAND * 200 + 100); // время появления данных
   ждать ВРЕМЯ = Тп;
   направить ИНИЦИАТОР на метку НАЧ;
Всё алгоритм;
Всё блок.
```

Модель процесса передачи данных в канале (2)

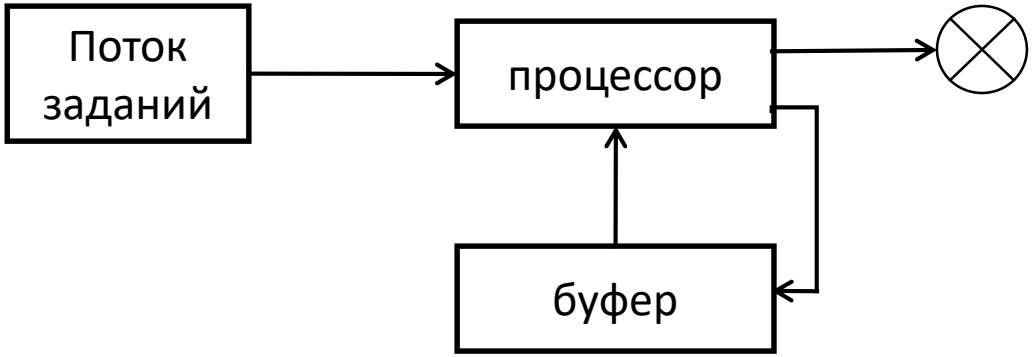
```
Блок-процессор Канал_Связи
описание
 КАНАЛ – скаляр; // начальное состояние «свободен»
 СОСТ – скаляр в блоке СОСТОЯНИЕ;
 Тк, W – скаляры; СТАРТ, МС, МК – метки;
всё описание;
алгоритм
СТАРТ: ждать (КАНАЛ = «свободный») .И. (СОСТ = «рабочий»);
  КАНАЛ := «занят»;
  Тк := ВРЕМЯ + (ИНИЦИАТОР->вектор(2)) / W ; // по условию задачи
  ждать (ВРЕМЯ = Тк) направить ИНИЦИАТОР на метку МК
        (СОСТ= «сломан») направить ИНИЦИАТОР на метку МС;
  МС: КАНАЛ := «свободен»;
  направить ИНИЦИАТОР на метку СТАРТ;
  МК: КАНАЛ := «свободен»;
  уничтожить ИНИЦИАТОР;
всё алгоритм;
всё блок.
```

Модель процесса передачи данных в канале (3)

```
Блок-агрегат  Помехи
описание
 СОСТ, Тсост – скаляры;
 ЦИКЛ – метка;
 // начальное положение ИНИЦИАТОРА на метке ЦИКЛ
Всё описание;
алгоритм
 ЦИКЛ: СОСТ := «работает»;
  Тсост := ВРЕМЯ + (RAND * 10000 + 10000); // время работы
  ждать ВРЕМЯ = Тсост;
  СОСТ := «сломан»;
  Тсост := ВРЕМЯ + (RAND * 1000 + 1000); // время ремонта
  ждать ВРЕМЯ = Тсост;
  направить ИНИЦИАТОР на метку ЦИКЛ;
Всё алгоритм;
Всё блок.
```

Задача: система обработки данных (1)

На вход системы поступает поток заданий. Система представляет собой процессор и буфер объёмом на три задания. Система работает в режиме разделения времени (задан квант времени на решение задачи, если досчиталась – покинула систему, не успела – решение прерывается и задание поступает в буфер в режиме FIFO). Одновременно в системе не может быть больше 4 (3+1) заданий. Остальные ждут во внешней очереди. Количество квантов для каждой задачи индивидуально и распределено в диапазоне от T1 до T2.

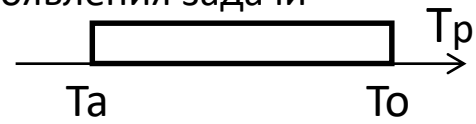


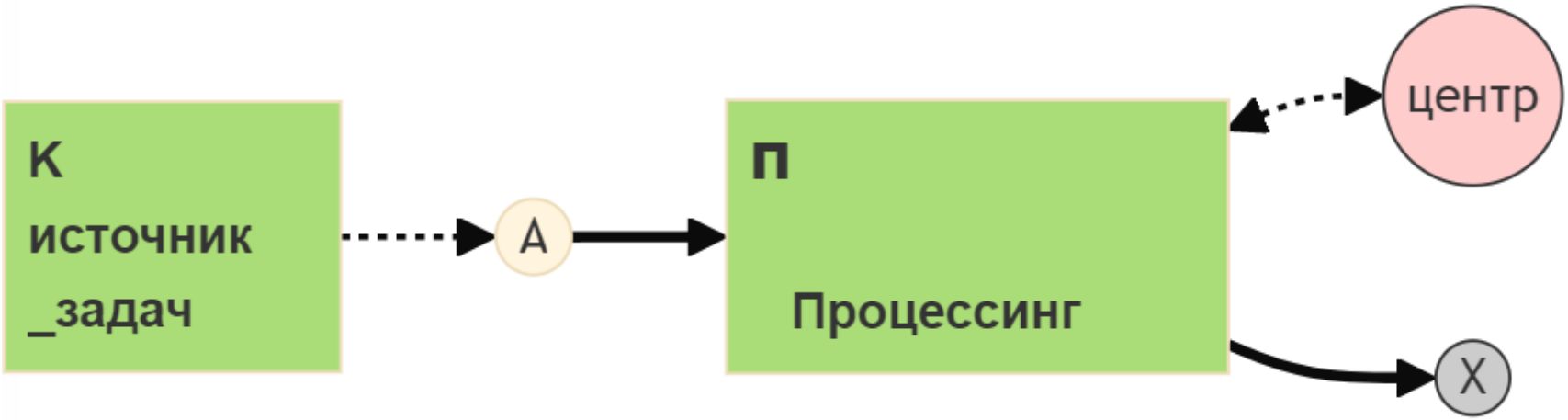
Задача: система обработки данных (2)

Время решения задачи

Размер кванта - Q

Интервал появления задачи





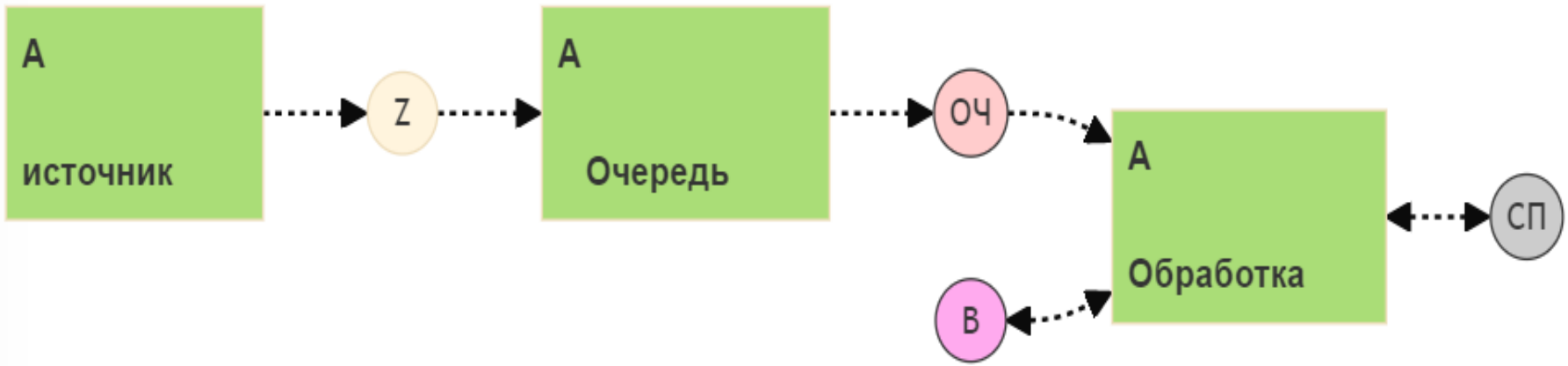
Задача: система обработки данных (3)

```
Блок-контроллер Источник_задач
описание
 Тп – скаляр;
 Т1, Т2, Та, То – скаляры; // из исходных данных
всё описание;
алгоритм
 НАЧ: создать W типа вектор(1-2 - скаляры);
   W(1) := ВРЕМЯ;
   W(2) := ЦЕЛОЕ(RAND * (Т2-Т1) + Т1);
   создать А типа ссылка;
   А := ссылка на вектор W;
   активизировать инициатор из А в блок ПРОЦЕССИНГ на метку ВХОД;
   Тп := ВРЕМЯ + (RAND * (То-Та) + Та);
   ждать ВРЕМЯ = Тп;
   направить ИНИЦИАТОР на метку НАЧ;
всё алгоритм;
всё блок.
```

Задача: система обработки данных (4)

```
Блок-процессор ПРОЦЕССИНГ
описание
 N – скаляр; // начальное значение 0
 Q – скаляр; // размер кванта – из исходных данных
 ЦЕНТР – скаляр; // начальное значение ‘свободен’
всё описание;
алгоритм
 ВХОД: ждать N < 4; // условие появления задачи в процессоре
       N := N + 1;
 РАСЧ: ждать ЦЕНТР = «свободен»;
       ЦЕНТР := «занят»;
       если ИНИЦИАТОР->вектор(2) >Q то направить ИНИЦИАТОР на метку КВТ;
       Треш := ВРЕМЯ + (ИНИЦИАТОР->вектор(2));
       направить ИНИЦИАТОР на метку СТ;
 КВТ:  Треш := ВРЕМЯ + Q;
  СТ:  ждать ВРЕМЯ = Треш;
       ЦЕНТР := «свободен»;
       ИНИЦИАТОР->вектор(2) := (ИНИЦИАТОР->вектор(2)) – Q;
       если ИНИЦИАТОР->вектор(2) ≤ 0 то направить ИНИЦИАТОР на метку ВЫХ;
       направить ИНИЦИАТОР на метку РАСЧ;
 ВЫХ:  N := N - 1;
       уничтожить ИНИЦИАТОР;
всё алгоритм;
всё блок.
```

Задача: система обработки данных (5)



Задача: система обработки данных (6)

```
Блок-агрегат Источник
описание
Z – скаляр; //нач.знач.0
всё описание;
алгоритм
 ЦИКЛ: Z := RAND * (T2-T1)+T1;
       Тген := ВРЕМЯ + (RAND *(b-a)+a);
       ждать ВРЕМЯ = Тген;
       направить ИНИЦИАТОР на метку ЦИКЛ;
всё алгоритм;
всё блок.
```

```
Блок-агрегат Очередь
описание
 ОЧ – вектор(1,2 - ссылки);
 Z – скаляр в блоке Источник;
всё описание;
алгоритм
 НАЧ: ждать Z ≠ 0;
      записать Z в список ОЧ;
      Z := 0;
      направить ИНИЦИАТОР на метку НАЧ;
всё алгоритм;
всё блок.
```

```
записать &B в список &A
алгоритм
создать &C типа вектор
   (1-скаляр, 2-ссылка);
&C(1) := &B;
если &A(1) ≠ 0 то направить
инициатор на &M1;
&A(1), &A(2) := ссылка
на вектор &C;
направить инициатор на &M2;
&M1: (&A(1)→вектор(2)):= ссылка
на вектор &C;
&A(1) := ссылка на вектор &C;
&M2:
все алгоритм.
```

```
перенести в список &J из списка &I
. . .
считать из списка &A в &B
. . .
```

Задача: система обработки данных (7)

```
Блок-агрегат Обработка
описание
 ОЧ – вектор в блоке Очередь;  Q – скаляр; // квант
 СП – вектор(1,2 - ссылки); // внутренний буфер
 В – скаляр; // рабочий параметр - задание
Всё описание;
алгоритм
 СТАРТ: ждать (ОЧ ≠ ∅).ИЛИ.(СП ≠ ∅);
        если (ОЧ ≠ ∅).И.(N < 4) то направить ИНИЦИАТОР на метку M_0;
        если (СП ≠ ∅) то направить ИНИЦИАТОР на метку M_1;
   M_0: перенести в список СП из списка ОЧ;      N := N + 1;
   M_1: считать из списка СП в В;
        если В > Q то направить ИНИЦИАТОР на метку M_2;
        Треш := ВРЕМЯ + В;
        направить ИНИЦИАТОР на метку M_3;
   M_2: Треш := ВРЕМЯ + Q;
   M_3: ждать ВРЕМЯ = Треш;           В := В – Q;
        если В ≤ 0 то направить ИНИЦИАТОР на метку M_4;
        записать В в список СП;
        направить ИНИЦИАТОР на метку СТАРТ;
   M_4: N:= N – 1;
        направить ИНИЦИАТОР на метку СТАРТ;
Всё алгоритм;
Всё блок.
```

Модель СМО: 1 очередь - 3 сервера – таймаут(1)

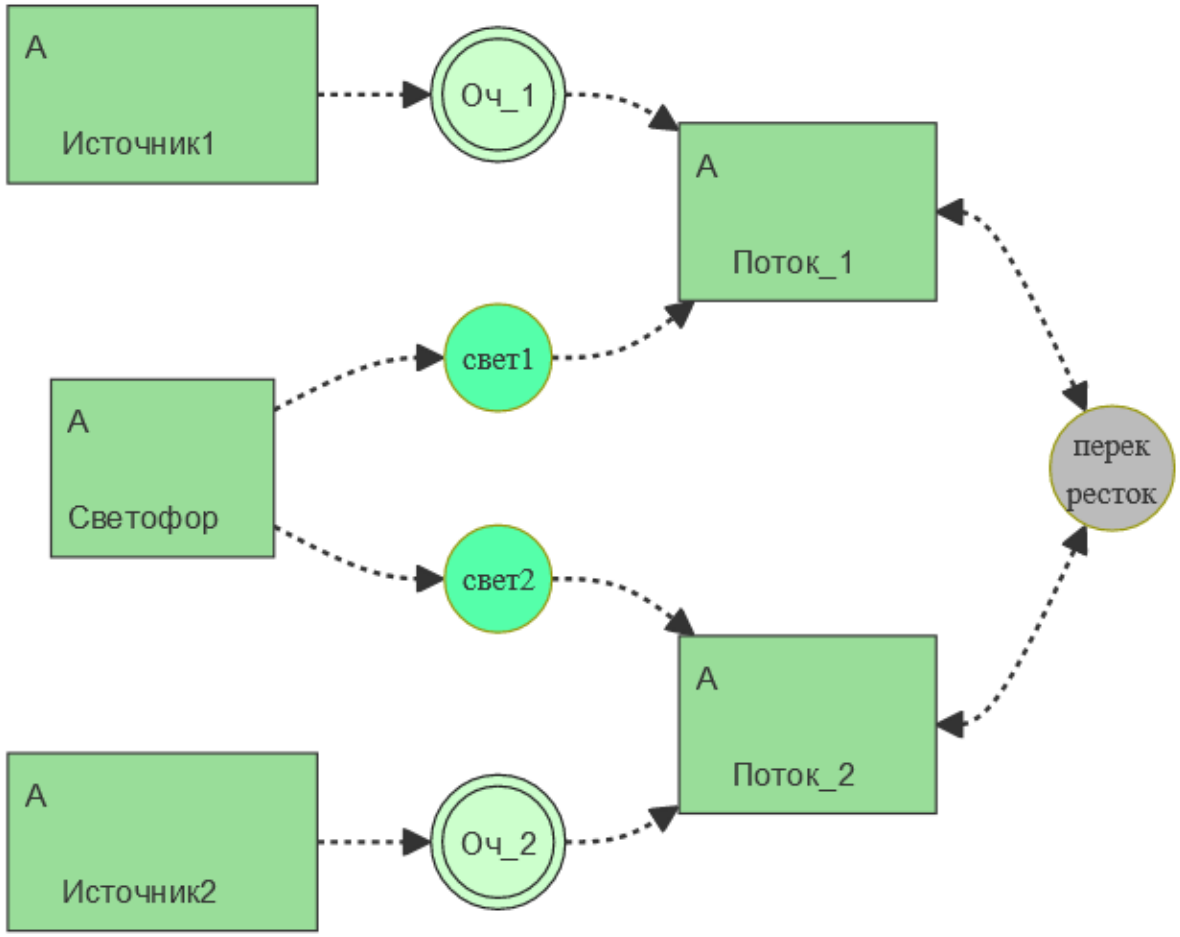
```
Блок-процессор ПРОЦЕССИНГ
алгоритм
  (ИНИЦИАТОР->вектор(1)) := ВРЕМЯ + (RAND*3+3); //T_прибытия
  ждать ВРЕМЯ = (ИНИЦИАТОР->вектор(1));
  (ИНИЦИАТОР->вектор(2)) := ВРЕМЯ + Таймаут;   // T_отказа
  Буфер +=1;
  ждать (сервер1 ="своб").ИЛИ.(сервер2 ="своб").ИЛИ.
        (сервер3 ="своб").ИЛИ.((ИНИЦИАТОР->вектор(2)) < ВРЕМЯ);
  Буфер -=1;
  если (сервер1="своб") то направить ИНИЦИАТОР на МЕТ1
       (сервер2="своб") то направить ИНИЦИАТОР на МЕТ2
       (сервер3="своб") то направить ИНИЦИАТОР на МЕТ3
  иначе  направить ИНИЦИАТОР на Выход;

Выход: потери +=1;
        уничтожить ИНИЦИАТОР;
```

Модель СМО: 1 очередь - 3 сервера – таймаут(2)

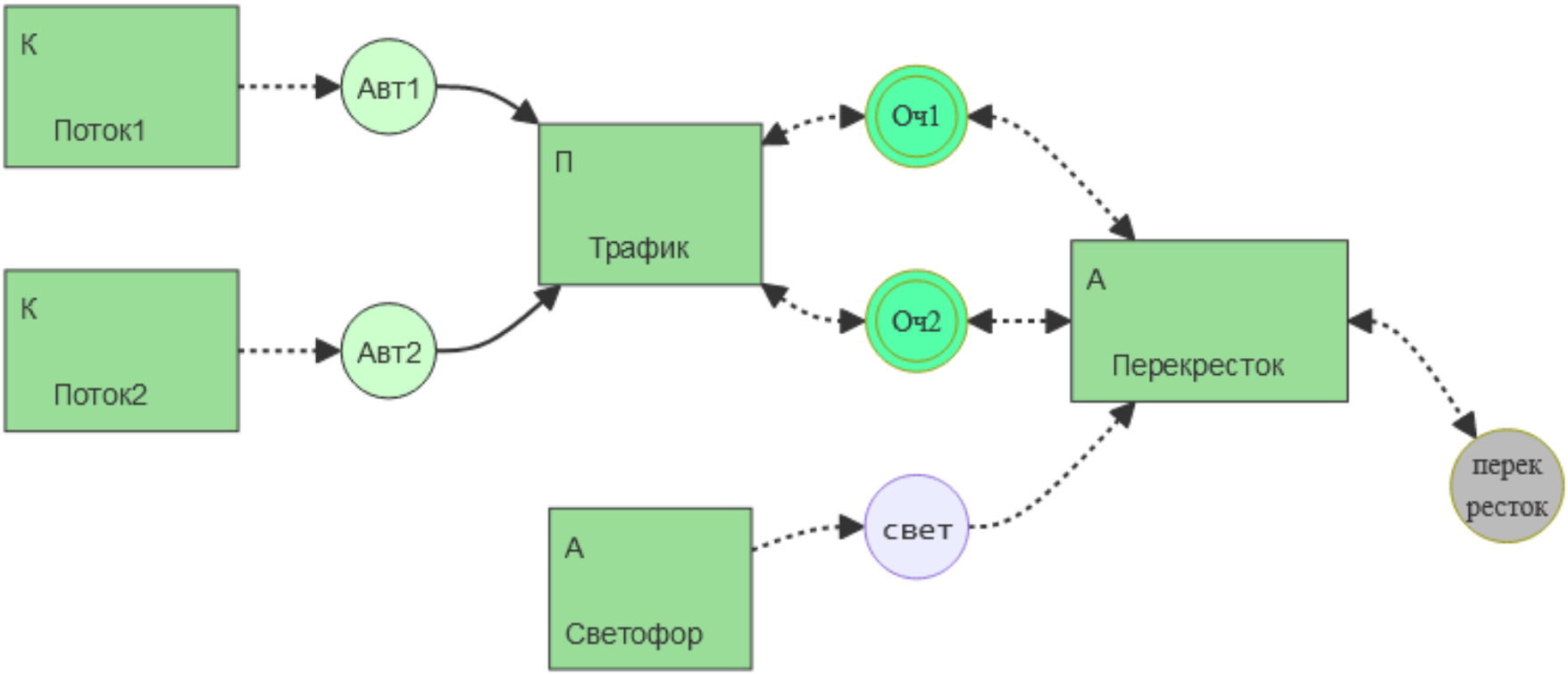
```
МЕТ1: сервер1 := "занят";
      Тсерв1 := ВРЕМЯ + (RAND*4+4);
      ждать ВРЕМЯ = Тсерв1;
      сервер1 := "своб";
      уничтожить ИНИЦИАТОР;
МЕТ2: сервер2 := "занят";
      Тсерв2 := ВРЕМЯ + (RAND*4+4);
      ждать ВРЕМЯ = Тсерв2;
      сервер2 := "своб";
      уничтожить ИНИЦИАТОР;
МЕТ3: сервер3 := "занят";
      Тсерв3 := ВРЕМЯ + (RAND*4+4);
      ждать ВРЕМЯ = Тсерв3;
      сервер3 := "своб";
      уничтожить ИНИЦИАТОР;
все алгоритм;  всё блок.
```

Агрегативная модель перекрёстка

Модель перекрестка двух автодорог, регулируемых светофором, работающим в режиме «красный» - «зеленый».
По каждой из дорог автомашины движутся в одном направлении по одной полосе.



вариант модели перекрёстка (2)



вариант модели дорожной сети (СеМО)