Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет инфокоммуникаций

Кафедра инфокоммуникационных технологий

С. М. Лапшин, В. Ю. Цветков

МОДЕЛИРОВАНИЕ УСТРОЙСТВ И СИСТЕМ ТЕЛЕКОММУНИКАЦИЙ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для направлений специальности 1-45 01 01-02 «Инфокоммуникационные технологии (сети инфокоммуникаций)» и 1-45 01 01-05 «Инфокоммуникационные технологии (системы распределения мультимедийной информации)»

Минск БГУИР 2018

УДК [004.94+621.391](076)
ББК 32.971.35я73+32.811я73
Л24

Р е ц е н з е н т ы:

кафедра последипломного образования учреждения образования
«Белорусская государственная академия связи»
(протокол №2 от 24.11.2016);

кафедра информационных систем и технологий
Белорусского национального технического университета
(протокол №8 от 15.04.2017)

Лапшин, С. М.

Л24 Моделирование устройств и систем телекоммуникаций : учеб.-метод. пособие / С. М. Лапшин, В. Ю. Цветков. – Минск : БГУИР, 2018. – 162 с. : ил.

ISBN 978-985-543-386-7.

Рассматриваются вопросы имитационного моделирования телекоммуникационных систем и сетей. Приводятся сведения о структуре и синтаксисе специализированного языка имитационного моделирования систем массового обслуживания GPSS, рассмотрено применение специализированного программного средства OPNET MODELER для проектирования и исследования характеристик телекоммуникационных сетей. Приводятся примеры решения типовых задач моделирования.

Для студентов, изучающих дисциплину «Системы коммутации каналов и пакетов». Может быть использовано при выполнении лабораторных работ, а также на этапах курсового и дипломного проектирования.

УДК [004.94+621.391](076)
ББК 32.971.35я73+32.811я73

ISBN 978-985-543-386-7

СОДЕРЖАНИЕ

1 ОСНОВНЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

Методы моделирования получили свое развитие благодаря эволюции вычислительной техники. Исторически первыми были разработаны аналитические методы моделирования и сложился аналитический подход к исследованию систем.

Аналитические методы моделирования. Аналитические методы позволяют получить характеристики системы как некоторые функции параметров ее функционирования. Таким образом, аналитическая модель представляет собой систему уравнений, при решении которой получают параметры, необходимые для оценки системы (время ответа, пропускную способность и т. д.). Использование аналитических методов дает достаточно точную оценку, которая зачастую хорошо соответствует действительности. Смена состояний реальной системы происходит под воздействием множества как внешних, так и внутренних факторов, подавляющее большинство из которых носит стохастический характер. Вследствие этого, а также большой сложности большинства реальных систем основным недостатком аналитических методов является то, что при выводе формул, на которых они основываются и которые используются для расчета интересующих параметров, необходимо принять определенные допущения. Тем не менее нередко оказывается, что эти допущения вполне оправданы.

Численные методы моделирования – преобразование модели к уравнениям, решение которых возможно методами вычислительной математики. Класс задач у них значительно шире. Однако численные методы не дают точных решений, но позволяют задать точность решения.

Имитационные методы моделирования (ИМ). С развитием вычислительной техники широкое применение получили имитационные методы моделирования для анализа систем, в которых преобладают стохастические воздействия.

Суть ИМ заключается в имитации процесса функционирования системы во времени, соблюдении таких же соотношений длительности операций, как в системе оригинале. При этом имитируются элементарные явления, составляющие процесс, сохраняется их логическая структура, последовательность протекания во времени. Результатом ИМ является получение оценок характеристик системы.

2 КЛАССИФИКАЦИЯ ВИДОВ МОДЕЛИРОВАНИЯ

В основу классификации моделирования можно положить различные признаки.

В зависимости от характера изучаемых процессов в системе все виды моделирования могут быть разделены на *детерминированные* и *стохастические*, *статические* и *динамические, дискретные* и *непрерывные.*

Детерминированное моделирование применяется для исследования систем, поведение которых можно абсолютно точно предвидеть. Например, путь,

пройденный автомобилем при равноускоренном движении в идеальных условиях; устройство, возводящее в квадрат число на входе. Соответственно имеем детерминированный процесс и детерминированную модель.

Стохастическое (теоретико-вероятностное) моделирование применяется для исследования систем, состояние которых зависит не только от контролируемых, но и от неконтролируемых воздействий или если в ней самой есть источник случайности. Например, человек и все системы, которые включают человека. Примерами стохастических систем являются заводы, аэропорты, сети и системы ЭВМ, телекоммуникационные сети, магазины, предприятия бытового обслуживания и т. п.

Статическое моделирование служит для описания систем в какой-либо момент времени.

Динамическое моделирование отражает изменение системы во времени (выходные характеристики системы в данный момент времени определяются характером входных воздействий в прошлом и настоящем). Например, устройство, возводящее x(t) в квадрат. Примерами динамических систем являются биологические, экономические, социальные системы, а также искусственные системы, такие как завод, предприятие, поточная линия и т. п.

Дискретное моделирование применяют для исследования систем, в которых входные и выходные характеристики измеряются или изменяются во времени дискретно, через dt (например часы), в противном случае применяют непрерывное моделирование. Например: ЭВМ, электронные часы, электросчетчик – дискретные системы; песочные часы, солнечные часы, нагревательные приборы – *непрерывные* системы.

В зависимости от формы представления объекта (системы) можно выделить *мысленное* и *реальное* моделирование.

При *реальном* (натурном) моделировании (РМ) исследование характеристик системы проводится на реальном объекте либо на его части. РМ является наиболее адекватным, но его возможности с учетом особенностей реальных объектов ограничены. Например, проведение реального моделирования с АСУ предприятия требует, во-первых, создания АСУ, а во-вторых, проведения экспериментов с предприятиями, что невозможно.

К реальному моделированию относят производственный эксперимент и комплексные испытания, которые обладают высокой степенью достоверности.

Другой вид реального моделирования – физическое. При физическом моделировании исследование проводится на установках, которые сохраняют природу явления и обладают физическим подобием.

Мысленное моделирование применяется для моделирования систем, которые практически нереализуемы на заданном интервале времени. В основе мысленного моделирования лежит создание идеальной модели, основанной на идеальной мыслительной аналогии. Различают два вида мысленного моделирования: образное (наглядное) и знаковое.

При образном моделировании на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и

процессы, протекающие в объекте (например, модели частиц газов в кинетической теории газов в виде упругих шаров, воздействующих друг на друга во время столкновения).

При знаковом моделировании описывают моделируемую систему с помощью условных знаков, символов, в частности, в виде математических, физических и химических формул.

Наиболее мощный и развитый класс знаковых моделей представляют математические модели.

Математическая модель – это искусственно созданный объект в виде математических, знаковых формул, который отображает и воспроизводит структуру, свойства, взаимосвязи и отношения между элементами исследуемого объекта. Именно этот класс моделей мы будем рассматривать и далее говорить только о математическом моделировании.

Математическое моделирование – метод исследования, основанный на замене исследуемого объекта-оригинала его математической моделью и на работе с ней (вместо объекта).

Математическое моделирование можно разделить на аналитическое (АМ), имитационное (ИМ), комбинированное (КМ).

При АМ создается аналитическая модель объекта в виде алгебраических, дифференциальных, конечно-разностных уравнений. Аналитическая модель исследуется либо аналитическими методами, либо численными методами.

При ИМ создается имитационная модель и используется метод статического моделирования для реализации имитационной модели на ЭВМ.

Комбинированное моделирование – декомпозиция процесса функционирования системы на подпроцессы. Для тех из них, где это возможно, используют аналитические методы, в противном случае – имитационные.

3 ЭЛЕМЕНТЫ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ, ПРИМЕНЯЕМЫЕ ПРИ МОДЕЛИРОВАНИИ СИСТЕМ: ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И ПОНЯТИЯ, СТРУКТУРА И КЛАССИФИКАЦИЯ

Под системой массового обслуживания (СМО) понимают динамическую систему, предназначенную для эффективного обслуживания потока заявок (требований на обслуживание) при ограничениях на ресурсы системы.

Первые задачи теории систем массового обслуживания (ТСМО) были рассмотрены сотрудниками Копенгагенской телефонной компании, датским ученым А. К. Эрлангом (1878–1929) в период между 1908 и 1922 гг. Эти задачи были вызваны к жизни стремлением упорядочить работу телефонной сети и разработать методы, позволяющие заранее повысить качество обслуживания потребителей в зависимости от числа используемых устройств. Оказалось, что ситуации, возникающие на телефонных станциях, являются типичными не только для те-

лефонной связи. Работа аэродромов, морских и речных портов, магазинов, терминальных классов, радиолокационных комплексов, радиолокационных станций может быть описана в рамках ТСМО.

Приведем примеры систем массового обслуживания.

Пример 1. Телефонная связь времен А. К. Эрланга представляла собой телефонную станцию, связанную с большим числом абонентов. Телефонистки станции по мере поступления вызовов соединяли телефонные номера между собой. Таким образом, важной задачей являлось определение количества телефонисток (при условии их полной занятости), которое должно работать на станциях для того, чтобы потери требований качества соединений номеров были минимальны.

Пример 2. Система скорой помощи некоего городского района представляет собой пункт, который содержит некоторое количество автомашин скорой помощи и несколько врачебных бригад для выполнения своих функций. Здесь важно определить количество врачей, вспомогательного персонала и автомашин для того, чтобы время ожидания вызова было для больных оптимальным при условии минимизации затрат на эксплуатацию системы и максимизации качества обслуживания.

Пример 3. Важной задачей является организация морских и речных перевозок грузов. При этом особое значение имеет оптимальное использование судов и портовых сооружений. Поэтому необходимо обеспечить определенный объем перевозок при минимальных расходах, при этом сократить простои судов при погрузочно-разгрузочных работах.

Пример 4. Система обработки информации содержит мультиплексный канал и несколько компьютеров. Сигналы от датчиков поступают на мультиплексный канал, где буферизуются и предварительно обрабатываются. Затем поступают в тот компьютер, где очередь минимальна. Задачей в данном случае является обеспечение ускорения обработки сигналов при заданной суммарной длине очереди.

Нетрудно привести множество других примеров из самых различных областей деятельности. Характерным чертами для них являются:

1) условие двойной случайности: случайный момент времени поступления заказа на обслуживание (на телефонную станцию, пункт скорой помощи, на вход процессора, момент времени прибытия морского судна на погрузку и т. д.), случайная длительность времени обслуживания;

2) наличие очередей (судов перед шлюзами, машин перед туннелями, покупателей перед прилавками, задач на входе процессора вычислительного комплекса и т. д.).

Реальные системы, с которыми приходится иметь дело на практике, как правило, очень сложны и включают в себя ряд этапов (стадий) обслуживания. Причем на каждом этапе может существовать вероятность отказа в выполнении или существует ситуация приоритетного обслуживания по отношению к другим требованиям. При этом отдельные звенья обслуживания могут прекратить свою

работу (для ремонта, подналадки и т. д.) или могут быть подключены дополнительные средства. Могут быть такие обстоятельства, когда требования, получившие отказ, вновь возвращаются в систему (подобное может происходить в информационных системах).

Структура СМО. Все СМО имеют вполне *определенную структуру*, изображенную на рисунке 3.1.

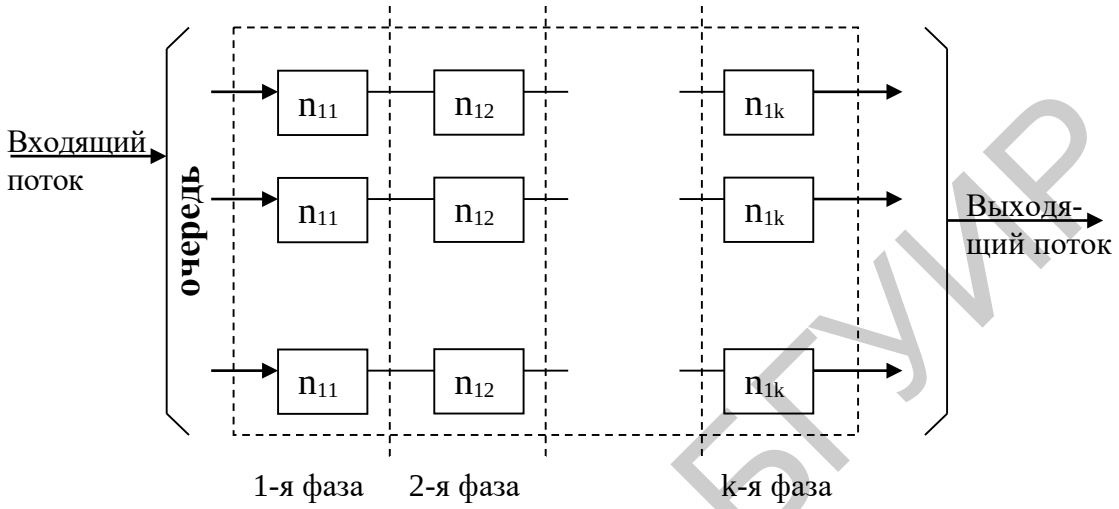


Рисунок 3.1 – Структура системы массового обслуживания

Основные определения СМО:

1 Потоком называют последовательность событий. Поток, состоящий из требований на обслуживание, назовем потоком требований.

2 Поток требований, поступающих в обслуживающую систему, назовем входным потоком.

3 Поток требований, которые обслужены, называется выходным потоком.

4 Совокупность очередей и приборов (каналов) обслуживания называется системой обслуживания.

5 Каждое требование поступает на свой канал, где подвергается операции обслуживания.

6 Каждая СМО имеет правила формирования очереди и правила или дисциплину обслуживания.

Можно выделить различные признаки классификации СМО.

СМО делятся на системы с отказами и системы без отказов.

В системе с отказами (с потерями, с конечной длиной очереди) заявка, пришедшая в момент, когда все каналы обслуживания заняты или заняты все места в очереди, получает отказ и покидает систему.

В системе без отказов (без потерь, с бесконечной длиной очереди) такая заявка не покидает систему, а становится в очередь и ждет, пока не освободится какой-нибудь канал. Время ожидания в общем случае не ограничено. Неограниченным может быть и количество требований, поступающих в систему.

СМО делятся на замкнутые и разомкнутые.

В замкнутых СМО в системе циркулирует определенное конечное число заявок (конечное число требований).

В разомкнутых СМО количество поступающих заявок бесконечно.

СМО делятся на многоканальные и одноканальные системы в зависимости от количества обслуживающих каналов.

В n-канальной СМО одновременно может обслуживаться n заявок. Каналы обслуживания иногда называют обслуживающими аппаратами (ОА).

В простейшем случае каждый ОА характеризуется своей производительностью (интенсивностью обслуживания заявок). Если в СМО поступают заявки нескольких типов, то для каждого типа заявок может быть задана соответствующая интенсивность обслуживания.

Длительность обслуживания заявки в ОА в общем виде – это случайная величина с законом распределения $B(\tau)$ и математическим ожиданием M (средним значением). Например, длительность обслуживания заявки процессором определяется временем выполнения соответствующей программы.

В случае малой разветвленности программы, когда число выполняемых операций практически постоянно, длительность обслуживания может считаться постоянной и равной M. В общем случае прикладные программы реализуют сложные алгоритмы с большим числом разветвлений. Количество операций, выполняемых в процессе обслуживания заявок одного типа, зависит от того, по какой ветви идет реализация алгоритма. В свою очередь путь реализации алгоритма определяется состоянием управляемого объекта, т. е. данными, поступающими в управляющую систему. В этом случае время выполнения программы рассматривается как случайная величина с математическим ожиданием M и дисперсией D.

Однако, если известно, что время обслуживания – случайная величина с известным значением ее математического ожидания, а сведения о законе распределения отсутствуют, то время выполнения программы целесообразно аппроксимировать экспоненциальным распределением

$$B(\tau) = 1 - e^{-\mu\tau} \quad b(\tau) = \mu e^{-\mu\tau},$$

где μ – интенсивность обслуживания (количество заявок, которое может быть обслужено в единицу времени, $\mu = \frac{1}{M}$).

Достоинства такой аппроксимации были перечислены ранее. Если в одноканальную СМО с интенсивностью обслуживания μ поступает входной поток заявок с интенсивностью λ, то величина $a = \frac{\lambda}{\mu}$ называется загрузкой СМО, или, по-другому, вероятностью того, что в произвольный момент времени ОА работает (не простаивает). Так как $\mu = \frac{1}{M}$, то $\rho = \lambda M$.

По приоритету заявок СМО делятся на следующие виды:

– СМО с заявками, имеющими разный приоритет (абсолютный, относительный);

– СМО с заявками, имеющими одинаковый приоритет.

При поступлении в СМО нескольких типов заявок могут быть организованы отдельные очереди для заявок каждого типа. Кроме размера, для каждой такой очереди обычно указывается приоритет находящихся в ней заявок. Приоритеты обычно кодирует целыми числами (0, 1, 2, 3, ...), причем чем меньше число, тем меньше приоритет соответствующих заявок. При наличии приоритетной организации в СМО на обслуживание в первую очередь выбираются заявки с высшими приоритетами. Различают относительный и абсолютный приоритет.

Если заявка с абсолютным приоритетом поступила в СМО в тот момент, когда на обслуживании находится заявка с меньшим приоритетом, то поступившая заявка сразу начинает обслуживаться, прерывая на время своего обслуживания находящуюся там заявку. Вытесненная таким образом заявка возвращается в начало своей очереди и ожидает продолжение обслуживания (дообслуживания). Для заявок с относительным приоритетом их приоритет вступает в действие не в момент их поступления в СМО, а в момент выбора следующей заявки из очереди (из очередей) на обслуживание. Прерываний в этом случае нет.

Приоритеты заявок бывают статические (постоянные) и динамические (изменяющиеся во времени). В самом общем виде задают дисциплину обслуживания заявок, представляющую собой комбинацию дисциплин со статической и динамической фиксацией приоритетов. Для этого функцию приоритетности задают в виде

$$q_i(t) = a_i + b_i(t - t_i),$$

где a_i – статическая составляющая приоритета заявки с номером i в СМО;

b_i – коэффициент динамической составляющей приоритета;

t – текущий момент времени, рассматриваемый на интервале между моментами входа в СМО и выхода после окончания обслуживания i-й заявки;

t_i – момент поступления в СМО заявки с порядковым номером i.

Основными задачами теории систем массового обслуживания являются:

1) расчет характеристик СМО (например, вероятность отказа в обслуживании, среднее число заявок в системе, среднее число занятых каналов и т. д.);

2) оценка эффективности СМО на основе рассчитанных характеристик;

3) оптимизация параметров СМО.

4 ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МОДЕЛИРОВАНИЯ

4.1 Классы инструментальных средств

Универсальным инструментальным средством создания моделей являются языки программирования общего пользования (Pascal, C/C++ и др.). На основе этих языков в настоящее время бурное развитие получили средства визуального проектирования программ (Delphi, Visual C++), облегчающие выполнение некоторых трудоемких операций, например, создание интерфейса программы. Наряду с этим существует множество специализированных средств моделирования, позволяющих быстрее и с меньшими затратами (по сравнению с универсальными языками программирования) создавать и исследовать модели. В развитии специализированных средств можно выделить два направления:

1. Средства моделирования для анализа достаточно широкого класса систем. К ним относятся языки имитационного моделирования (GPSS, SIMSCRIPT и др.), пакеты прикладных программ, использующих для моделирования аналитические методы, такие как MathCad, MathLab, SAS, Statistica и др. Основным недостатком этих средств является то, что их применение требует от исследователя специальной подготовки.

2. Программные комплексы, специализирующиеся на моделировании узкого круга систем одной конкретной предметной области. Недостаток, заключающийся в ограниченности применения таких программ одной предметной областью, с лихвой покрывается такими преимуществами, как легкость их освоения специалистами в данной предметной области и эффективность применения, являющаяся следствием узкой специализации.

Рассмотрим подробнее специфику инструментальных средств имитационного моделирования. Как было ранее отмечено, существует два направления их развития. Первое из них представляют языки имитационного моделирования. Эти языки по сравнению с универсальными языками программирования снижают трудоемкость написания моделирующих программ, включают специализированные процедуры, которые могут применяться в любой имитационной модели и отличаются точностью выражения понятий, характеризующих имитируемые процессы, и автоматическим формированием определенных типов данных, необходимых в процессе имитационного моделирования.

Второе направление – специализированные программные средства имитационного моделирования, позволяющие строить модели различных систем, используя встроенные модели элементов и связей, существенно облегчающие разработку систем и не требующие глубоких знаний в области программирования. К таким средствам относятся программы Anylogic, Cisco Packet Tracer, OpNet, OMNET и др.

4.2 Технология разработки имитационной модели

Процесс последовательной разработки имитационной модели начинается с создания простой модели, которая затем постепенно усложняется в соответствии с предъявляемыми решаемой проблемой требованиями. В каждом цикле создания программной модели можно выделить следующие этапы:

1) формулирование проблемы: описание исследуемой проблемы, установление границ и ограничений моделируемой системы, определение целей исследования;

2) разработка модели: переход от реальной системы к некоторой логической схеме (абстрагирование);

3) подготовка данных: отбор данных, необходимых для построения модели, и представление их в соответствующей форме;

4) трансляция модели: описание модели на языке имитационного моделирования;

5) оценка адекватности: повышение до приемлемого уровня степени уверенности, с которой можно судить относительно корректности выводов о реальной системе, полученных на основании обращения к модели;

6) планирование: определение условий проведения машинного эксперимента с имитационной моделью;

7) экспериментирование: многократный прогон имитационной модели на компьютере для получения требуемой информации;

8) анализ результатов: изучение результатов имитационного эксперимента для подготовки выводов и рекомендаций по решению проблемы;

9) реализация и документирование: реализация рекомендаций, полученных на основе имитации, и составление документации по модели и ее использованию.

5 МОДЕЛИРОВАНИЕ В GPSS WORLD

5.1 Основы построения и принципы функционирования языка имитационного моделирования

Модель разрабатывается на языке GPSS и состоит из операторов, а объект «Модель» создается при помощи встроенного текстового редактора. Объект «Процесс моделирования» – это результат трансляции модели. Далее процесс моделирования запускается с помощью команд GPSS. По завершении моделирования, как правило, автоматически создается объект «Отчет».

Текстовый объект (текстовый файл GPSS World) предназначен для упрощения разработки больших моделей и создания библиотеки исходных текстов. То есть модель может быть разделена на наборы операторов, представляющие собой отдельные текстовые файлы, а затем объектом «Процесс моделирования» собрана из них. Объект «Процесс моделирования» может также создавать новые

текстовые файлы с фрагментами модели, результатами моделирования, а также считывать и записывать данные в текстовые файлы.

GPSS World предназначена для имитационного моделирования систем с дискретными и непрерывными процессами. Языком моделирования в ней является язык GPSS, улучшенный встроенным языком программирования низкого уровня PLUS. Язык GPSS построен в предположении, что модель сложной системы можно представить совокупностью элементов и логических правил их взаимодействия в процессе функционирования моделируемой системы. Набор абстрактных элементов, называемых объектами, небольшой. Также набор логических правил ограничен и может быть описан стандартными операциями. Комплекс программ, описывающих функционирование объектов и выполняющих логические операции, является основой для создания программной модели.

Кроме этого комплекса в составе GPSS World имеется программа-планировщик, выполняющая следующие функции:

– обеспечение продвижения по заданным разработчиком маршрутам динамических объектов, называемых транзактами;

– планирование событий, происходящих в модели, путем регистрации времени наступления каждого события и выполнения их в нарастающей временной последовательности;

– регистрация статистической информации о функционировании модели;

– продвижение модельного времени в процессе моделирования системы.

Чтобы обеспечить правильную последовательность обработки событий во времени, имеются системные часы, хранящие значения абсолютного модельного времени.

Объекты в моделируемой системе предназначены для различных целей. Совершенно не обязательно, чтобы в одной модели участвовали все типы объектов. Необходимо лишь наличие блоков и транзактов, иначе модель работать не будет.

Объекты подразделяются на семь категорий и пятнадцать типов, которые представлены в таблице 5.1.

Таблица 5.1 – Объекты GPSS

Категории	**Типы объектов**
Динамическая	Транзакты
Операционная	Блоки
Аппаратная	Одноканальные устройства памяти (многоканальные устройства), логические ключи-переключатели
Вычислительная	Переменные, функции, генераторы случайных чисел
Статистическая	Очереди, таблицы
Запоминающая	Ячейки, матрицы ячеек
Группирующая	Числовые группы, группы транзактов, списки

Рассмотрим назначение объектов GPSS.

Динамическими объектами являются транзакты, которые создаются в определенных точках модели, продвигаются планировщиком через блоки, а затем уничтожаются. Транзакты являются аналогами единиц – потоков в реальной системе. Они могут представлять собой различные элементы даже в одной модели. С каждым транзактом связаны параметры, которые используются для конкретных данных. Каждый транзакт может иметь любое число параметров. Параметры нумеруются или им присваиваются имена. Номера параметров и имена используются для ссылок на значения, присвоенные параметрам. Транзактам может присваиваться приоритет. Приоритет определяет предпочтение, которое получает транзакт, когда он и другие транзакты претендуют на один и тот же ресурс.

Объекты аппаратной категории – это абстрактные элементы, на которые может быть декомпозирована реальная система. Воздействуя на эти объекты, транзакты могут изменять их состояние и влиять на движение других транзактов. К объектам этого типа относятся одноканальные устройства памяти (многоканальные устройства) и логические ключи.

Одноканальные устройства (ОКУ) представляют собой оборудование, которое в любой момент времени может быть занято только одним транзактом. Например, один канал передачи данных, одноканальный ремонтный орган, один станок изготовления деталей, одно транспортное средство.

Многоканальные устройства (МКУ) предназначены для имитации оборудования, осуществляющего параллельную обработку. Они могут быть использованы одновременно несколькими транзактами. МКУ можно использовать в качестве аналога, например, многоканального ремонтного органа, нескольких каналов связи.

Для моделирования такого оборудования, как переключатели, имеющие только два состояния, в GPSS используются логические ключи.

Операционные объекты, т. е. блоки, задают логику функционирования модели системы и определяют пути движения транзактов между объектами аппаратной категории. В блоках могут происходить события четырех основных типов:

1) создание или уничтожение транзактов;
2) изменение числового атрибута объекта;
3) задержка транзакта на определенный период времени;
4) изменение маршрута движения транзакта в модели.

Версия GPSS, реализованная в системе GPSS World, содержит 53 типа блоков.

В зависимости от назначения блоки подразделяются на несколько групп.

Блоки, осуществляющие модификацию атрибутов транзактов:

– генерирование и уничтожение транзактов (**GENERATE**, **SPLIT**, **TERMINATE**, **ASSEMBLE**);

– временные задержки (**ADVANCE**);

– синхронизацию движения двух (**MATCH**) и нескольких (**GATHER**) транзактов;

– изменение приоритета транзакта (**PRIORITY**);

– изменение параметров транзактов (**ASSIGN**, **INDEX**, **MARK**, **PLUS**).

Блоки, изменяющие последовательность движения транзактов (блоки передачи управления): **DISPLACE**, **TRANSFER**, **LOOP**, **TEST**, **GATE**.

Блоки, связанные с группирующей категорией: **ADOPT**, **ALTER**, **EXAMINE**, **JOIN**, **REMOVE**, **SCAN**.

Блоки, описывающие объекты аппаратной категории:

– одноканальные устройства (технические средства): **SEIZE**, **RELEASE**, **PREEMPT**, **RETURN**, **FUNAVAIL**, **FAVAIL**;

– многоканальные устройства (памяти): **ENTER**, **LEAVE**, **SAVAIL**, **SUNAVAIL**;

– ключи (логические переключатели): **LOGIC**.

Блоки, сохраняющие необходимые значения для дальнейшего использования: **SAVEVALUE**, **MSAVEVALUE**.

Блоки для получения статистических результатов:

– очереди: **QUEUE**, **DEPART**;

– таблицы: **TABULATE**.

Блоки для организации списка пользователя: **LINK**, **UNLINK**.

Блоки для организации ввода-вывода:

– открытие/закрытие файла: **OPEN/CLOSE**;

– считывание/запись в файл: **READ/WRITE**;

– установка позиции текущей строки: **SEEK**.

Специальные блоки: **BUFFER**, **COUNT**, **EXECUTE**, **INTEGRATION**, **SELECT**, **TRACE**, **UNTRACE**.

Вычислительная категория служит для описания таких ситуаций в процессе моделирования, когда связи между компонентами моделируемой системы наиболее просто и компактно выражаются в виде математических (аналитических и логических) соотношений. Для этих целей в качестве объектов вычислительной категории введены арифметические и булевы переменные и функции.

Переменные представляют собой сложные выражения, которые включают константы, системные числовые атрибуты (СЧА), библиотечные арифметические функции, арифметические и логические операции.

Выражения могут применяться в переменных и операторах GPSS. При применении в переменных выражения определяются командами GPSS. При применении в операторах GPSS выражения определяются как часть языка PLUS.

Каждому объекту соответствуют атрибуты, описывающие его состояние в данный момент времени. Они доступны для использования в течение всего процесса моделирования и называются системными числовыми атрибутами. Например, объект вычислительной категории – генератор случайных чисел имеет СЧА RNn – число, вычисляемое генератором равномерно распределенных случайных чисел номер n; объект динамической категории – транзакт СЧА PR – приоритет обрабатываемого в данный момент транзакта; Pi – значение i-го параметра активного транзакта и др. Всего в GPSS World имеется свыше 50 СЧА.

Булевы переменные позволяют пользователю проверять в одном блоке GPSS одновременно несколько условий, исходя из состояния или значения этих условий и их атрибутов.

С помощью функций пользователь может производить вычисления непрерывных или дискретных функциональных зависимостей между аргументом функции (независимая величина) и зависимым значением функции.

Кроме библиотечных арифметических функций GPSS World имеет 24 встроенных генератора случайных чисел.

Объекты запоминающей категории обеспечивают обращения к сохраняемым значениям. Ячейки и матрицы ячеек сохраняемых величин используются для сохранения некоторой числовой информации. Любой активный транзакт может произвести запись информации в эти объекты. Впоследствии записанную в эти объекты информацию может считать любой транзакт. Матрицы могут иметь до шести измерений.

К статистическим объектам относятся очереди и таблицы. В любой системе движение потока транзактов может быть задержано из-за недоступности устройств. В этом случае задержанные транзакты ставятся в очередь – еще один тип объектов GPSS. Учет этих очередей составляет одну из основных функций планировщика. Планировщик автоматически накапливает определенную статистику относительно устройств и очередей. Кроме этого, пользователь может собирать дополнительную статистическую информацию, указав специальные точки в модели.

Для облегчения табулирования статистической информации в GPSS предусмотрен специальный объект – таблица. Таблицы используются для получения выборочных распределений некоторых случайных величин. Таблица состоит из частотных классов (диапазонов значений), куда заносится число попаданий конкретного числового атрибута в каждый, тот или иной, частотный класс. Для каждой таблицы вычисляется также математическое ожидание и среднеквадратичное отклонение.

К группирующей категории относятся три типа объектов: числовая группа, группа транзактов и списки.

При моделировании транзакты хранятся в списках. Существует пять видов списков, только в одном из которых в любой момент времени может находиться транзакт:

– текущих событий;

– будущих событий;

– задержки ОКУ или МКУ;

– отложенных прерываний ОКУ;

– пользователя.

Одноканальное устройство имеет следующие списки:

– отложенных прерываний – список транзактов, ожидающих занятия ОКУ по приоритету;

– прерываний – список транзактов, обслуживание которых данным ОКУ было прервано;

– задержки – список транзактов, ожидающих занятия ОКУ в порядке приоритета;

– повторных попыток – список транзактов, ожидающих изменения состояния ОКУ.

Многоканальное устройство имеет следующие списки:

– задержки – список транзактов в порядке приоритета, ожидающих возможность занять освободившиеся каналы МКУ;

– повторных попыток – список транзактов, ожидающих изменения состояния МКУ.

Список пользователя содержит транзакты, удаленные пользователем из списка текущих событий и помещенные в список пользователя как временно неактивные. Списки пользователя используются для организации очередей с дисциплинами, отличными от дисциплины «первым пришел – первым обслужен».

5.2 Построение моделей с устройствами

Для представления собственно обслуживания используются определенные элементы. Такими элементами могут быть либо люди, либо какие-то устройства. Независимо от этого подобные элементы в GPSS называют объектами аппаратной категории, к которой относят ОКУ, МКУ и логические ключи.

Рассмотрение методов построения моделей с устройствами начнем с имитации функционирования ОКУ.

При моделировании возможны следующие режимы организации функционирования ОКУ:

– занятие ОКУ и его освобождение;

– прерывание обслуживания ОКУ;

– недоступность ОКУ и восстановление доступности.

Прежде чем сразу рассматривать блоки, моделирующие ОКУ, вспомним, что потоки, существующие в реальных системах, в моделях имитируют транзакты. Поэтому сначала узнаем, как вводятся транзакты в модель и как выводятся из нее. А так как построение самых простейших моделей невозможно без некоторых блоков GPSS, такие блоки будут также рассмотрены.

Организация поступления транзактов в модель и удаления транзактов из нее

Поступление транзактов в модель

GENERATE – это блок, через который транзакты входят в модель. Блок **GENERATE** имеет следующий формат записи:

GENERATE [A],[B],[C],[D],[E]

Скобки [ ] означают, что данный операнд является необязательным. Не существует ограничений на число различных блоков **GENERATE** в одной модели.

Интервалы времени между последовательными появлениями транзактов блока **GENERATE** называют интервалом поступления. Все разработчики

должны задавать спецификацию распределения интервалов времени поступления в блоке **GENERATE**. Информация, необходимая для этого, задается операндами A и B. Все возможные виды распределения интервалов времени поступления в GPSS делят на равномерно распределенные и все остальные виды распределения.

Операнд A – средний интервал времени между последовательными поступлениями транзактов в модель.

Операнд B задает модификатор, который изменяет значения интервала генерации транзактов по сравнению с интервалом, указанным операндом A. Есть два типа модификаторов: модификатор-интервал и модификатор-функция.

С помощью модификатора-интервала задается равномерный закон распределения времени между генерацией транзактов.

Операнды A и B могут быть именем, положительным числом, выражением в скобках или непосредственно СЧА.

При вычислении разности значений (A – B), заданных операндами A и B, получается нижняя граница интервала, а при вычислении суммы (A + B) – верхняя граница. После генерации очередного транзакта выбирается число из полученного интервала, это и будет значение времени, через которое следующий транзакт выйдет из блока **GENERATE**.

Когда операнды A и B задают в виде констант (B – модификатор-интервал), они должны быть неотрицательными числами, т. е. интервал времени может быть выражен числами, например, 4.1, …, 12.7. Предположим, что транзакт входит в модель – блок **GENERATE** в момент модельного времени 25.6. После того как этот транзакт попадет в следующий блок модели, планировщик GPSS разыграет случайное значение из распределения интервалов времени, равного 8.4 ± 4.3. Пусть разыгранным значением будет число 9.7. Тогда планировщик планирует приход следующего транзакта в блок **GENERATE** в момент времени 25.6 + 9.7 = 35.3.

Можно выбрать для розыгрыша генератор равномерно распределенных случайных чисел. Это устанавливается на странице «Random Numbers» («Случайные числа») в журнале настроек модели. Нужно выбрать Edit/Settings (Правка/Настройка) и страницу «Random Numbers» («Случайные числа»), на которой в поле ввода «**GENERATE**» ввести номер генератора – любое положительное целое число. По умолчанию используется генератор равномерно распределенных случайных чисел номер 1.

Операнды A и B необязательно должны быть заданы. Когда один или оба операнда не указаны, по умолчанию предполагается их нулевое значение. Например, A = 16.4, B = 0. Поскольку операнд B = 0, то интервалы времени распределены равномерно (16.4 ± 0), т. е. интервал времени поступления равен 16.4. Это пример того, как может быть задано детерминированное значение интервалов времени.

Более сложные интервалы времени поступления транзактов (не по равномерному закону) могут быть заданы с использованием модификатора-функции или встроенных генераторов случайных чисел. Под действием модификатора-

функции значение операнда A умножается на значение функции, заданной операндом B.

При любом способе вычисления интервала времени значение операнда B не должно превышать значение операнда A, в противном случае в блоке **GENERATE** может быть получен отрицательный интервал времени, который вызовет остановку по ошибке «Отрицательное время задержки».

Рассмотрим три дополнительных операнда: смещение интервалов C, ограничитель D, уровень приоритета E.

Смещение интервалов (первоначальная задержка) C – это момент времени, в который в блоке **GENERATE** должен появиться первый транзакт. После этого первого прихода все остальные приходы транзактов возникают в соответствии с распределением интервалов времени, задаваемых операндами A и B. Операнд C можно использовать как для ускорения, так и для замедления прихода первого транзакта или для указания прихода в нужный момент времени. Начальная задержка может быть меньше, равна или больше среднего времени, заданного операндом A. Когда операнд C не используется, интервалы генерирования определяются операндами A и B (они не оказывают влияния на задержку). Операнд C может быть таким же, как и операнды A и B.

Операнд D задает граничное значение общего числа транзактов, которые могут войти в модель через данный блок **GENERATE** в течение времени моделирования. Когда это число достигнуто, данный блок **GENERATE** перестает быть активным. Если не определено граничное значение (операнд D не используется), блок **GENERATE** остается активным в течение всего времени моделирования, т. е. по умолчанию ограничения на количество создаваемых транзактов нет.

Операнд E устанавливает класс приоритета каждого из транзактов, входящих в модель через данный блок **GENERATE**. Для задания приоритетов с целью повышения эффективности работы GPSS World рекомендуется использовать последовательность целых чисел 0, 1, 2, 3. Чем выше число, тем выше приоритет. Если операнд E не используется, по умолчанию приоритет генерируемых данным блоком **GENERATE** транзактов равен нулю.

Операнды D и E могут задаваться так же, как и операнды A, B и C, но при этом принимать значения только целых положительных и целых чисел соответственно.

В любом блоке **GENERATE** должен быть обязательно задан либо операнд A, либо операнд D. Нельзя использовать в качестве операнда параметры транзактов. Необходимо также помнить, что транзакт не должен входить в блок **GENERATE**. Если транзакт пытается это делать, возникает ошибка выполнения. Приведем примеры записи блоков **GENERATE**:

с операндом A

```
GENERATE    38.6
GENERATE    X$IntPostTran
GENERATE    MX$VrPost(3,6)
GENERATE    V$Prom
```

GENERATE (Exponential(11,0,X$Mat))
GENERATE IntPostTran

с операндами A и B

GENERATE 73.25,X$Otk
GENERATE X$Sredne,FN2
GENERATE Sredne,FN4
GENERATE (V$Post+7.1),FN$Mod

с операндами A и C

GENERATE 7.3,,4.1
GENERATE 7.3,,X$VrSm
GENERATE V$IntP,,MX2(X$Stroka,X$Stolbez)
GENERATE (Normal(3,XSre,XSreOtk)),,Sme

с операндами A, B, E

GENERATE 13.3,2.8,,,1
GENERATE (Normal(8,XSr,XSrOtk)),Post,,1
GENERATE V$IntPostTran,(V1–12.3),,12

Приведенные примеры демонстрируют различные способы задания операндов блока **GENERATE**. Однако при этом нужно помнить следующее. В начальный момент времени в каждом блоке **GENERATE** производится подготовка к выходу одного транзакта. На этой стадии модель еще полностью не инициализирована для выполнения, т. е. не все переменные получили значения. Но переменные, описанные в блоке **GENERATE**, должны быть уже определены, т. е. инициализированы. Поэтому в модели блоку **GENERATE** должны предшествовать команды определения **EQU**, **INITIAL**, **FUNCTION**, **VARIABLE**, **FVARIABLE**. Это делается для того, чтобы СЧА в блоке GENERATE, который ссылается на них, давали нужные для ввода транзактов в модель результаты.

Например:

```
SrIntPost    EQU     47.2
StanOtkl     EQU     28.6
   INITIAL   X$KolTrans,43
. . .
   GENERATE SrIntPost,StOtk,,X$KolTrans
```

Как видно из примера, блоку **GENERATE** предшествуют присвоения командой **EQU** именам числовых значений, а командой **INITIAL** – начального значения сохраняемой ячейке с именем **KolTrans**.

Удаление транзактов из модели и завершение моделирования

Транзакты удаляются из модели, попадая в блок **TERMINATE** (завершить).

Блоки **TERMINATE** всегда позволяют войти всем транзактам, которые пытаются это сделать. В модели может быть любое число блоков **TERMINATE**. Блок имеет следующий формат записи:

TERMINATE [A]

Значением операнда A является число единиц, на которое блок **TERMINATE** уменьшает содержимое счетчика завершения, определяющего

момент окончания моделирования. Операнд А может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА*<параметр>. По умолчанию значение операнда А равно нулю. В этом случае транзакт уничтожается, а значение счетчика завершения не меняется.

Счетчик завершения представляет собой ячейку памяти с именем TG1, которая хранит положительное целое число. Это число записывается в ячейку TG1 командой **START** в начале процесса моделирования.

В процессе моделирования транзакты попадают в блок **TERMINATE** и в соответствии со значением операнда А вычитают определенное число из счетчика завершения. При достижении содержимым счетчика нуля моделирование завершается. В модели может быть много блоков **TERMINATE**, но счетчик завершения только один.

Когда пользователь подготавливает модель, он задает время моделирования, указывая в операторе **START** значение счетчика завершения. Поскольку пути прохождения транзактов в модели имеют различный физический смысл, каждый блок **TERMINATE** может либо уменьшать, либо не уменьшать содержимое счетчика завершения.

Рассмотрим пример, в котором блок **TERMINATE** и команда **START** используются для управления временем моделирования. Предположим, что разработчик выбрал в качестве единицы времени 1 мин. Он хочет промоделировать поведение системы в течение 10 ч, затем моделирование должно быть закончено. За единицу модельного времени возьмем 1 мин, тогда время моделирования равно 10 × 60 = 600 ед.

Любая модель на GPSS состоит из одного или нескольких сегментов. Для управления временем моделирования разработчик (рисунок 5.1) включает в модель сегмент из блоков **GENERATE** и **TERMINATE**; в блоке **TERMINATE** в качестве операнда А использует 1; во всех прочих блоках **TERMINATE** модели использует операнд А по умолчанию (однако возможны и другие варианты, т. е. и в других блоках **TERMINATE** операнд А может быть 1).

В процессе моделирования транзакты, которые двигаются в других сегментах модели, время от времени выводятся из модели в других блоках **TERMINATE**, но они не оказывают воздействия на счетчик завершения. В момент модельного времени 600 транзакт в приведенном выше сегменте попадает в блок **GENERATE** и сразу же переходит в следующий блок **TERMINATE**.

Поскольку операнд А блока содержит 1, то из счетчика завершения вычитается 1. Предположим, что в команде **START** было указано число 10, т. е. десять прогонов модели, и в счетчик завершений записано число 10: TG1 = 10. После первого вычитания содержимое ячейки TG1 = 9, т. е. не равно нулю. Поэтому моделирование продолжается. После десяти прогонов, т. е. вычитания десяти единиц, TG1 = 0. Планировщик прекращает моделирование.

Команда **START** используется для запуска процесса моделирования. Она имеет следующий формат записи:

START A,[B],C,[D]

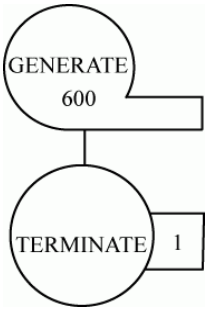


Рисунок 5.1 – Блок-диаграмма процесса **GENERATE** – **TERMINATE**

Операнд A задает значение счетчика завершения, определяющего момент окончания прогона модели. Он может быть только целым положительным числом. Операнд B – операнд вывода статистики. Он может быть NP («нет вывода данных») либо опущен. При задании NP стандартный отчет не выводится. По умолчанию выводится стандартный отчет. Операнд C не используется и сохранен для совместимости с описаниями ранних версий GPSS. Операнд D определяет необходимость вывода содержимого списков событий. Если операнд D указать любым положительным целым числом, например, 1, то списки текущих и будущих событий включаются в стандартный отчет и выводятся. Если операнд D опущен, то по умолчанию списки в стандартном отчете не выводятся.

Команду **START** можно сразу указывать в конце программы модели при ее подготовке и в таком виде записывать на магнитный носитель. Тогда после трансляции модели, т. е. создания объекта «процесс моделирования», сразу начинается моделирование. Этот же оператор можно вводить в программу модели в интерактивном режиме.

Однако может возникнуть необходимость завершить моделирование не по истечении какого-то времени, а после обработки определенного количества транзактов, имитирующих, например, изготовленные детали, переданные по каналу связи сообщения, и т. д. В этом случае сегмент задания времени моделирования не нужен. Для организации такого способа завершения моделирования необходимо сделать следующее. В блоках **TERMINATE**, которые выводят из модели транзакты, имеющие смысл тех же изготовленных деталей или переданных сообщений, указать число, на которое уменьшается счетчик завершения моделирования. В команде **START** также указать число, деление которого на указанное в блоке **TERMINATE** число даст требуемое количество переданных сообщений. Например, требуется завершить моделирование после передачи 100 сообщений. В модели это может быть так:

```
. . .
TERMINATE    1
. . .
TERMINATE    1
. . .
TERMINATE
```

START 100

Блоков **TERMINATE**, которые выводят из модели транзакты, соответствующие переданным сообщениям, может быть несколько. Все эти блоки должны иметь 1 в качестве операнда А. У остальных блоков **TERMINATE**, если они есть в модели, операнд А должен быть опущен.

Итак, транзакты введены в модель. Но аналоги транзактов – элементы потоков – в реальных системах имеют различные характеристики. Рассмотрим, как эти же характеристики присваиваются транзактам.

Изменение значений параметров транзактов

Каждый транзакт может иметь любое число параметров. Интерпретация смысла параметров произвольная. В момент генерации транзактов все его параметры нулевые (только те, которые используются в модели). Блок **ASSIGN** (рисунок 5.2) является основным средством для задания значений параметров транзактов.

Формат записи будет следующим:

ASSIGN A,B,[C]

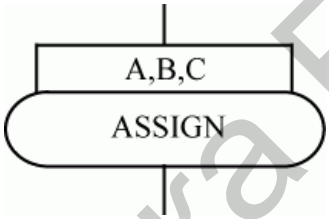


Рисунок 5.2 – Блок-диаграмма блока **ASSIGN**

Операндом А задается номер параметра, которому присваивается значение. Операнд А может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА<параметр> и следующими за ними знаками: + (если нужно увеличить), – (если нужно уменьшить).

Операнд В определяет значение, которое следует добавить, вычесть или которым следует заменить значение в параметре, заданном операндом А. Если такой параметр не существует, то он создается со значением, равным нулю. Операнд В может быть таким же, как и операнд А, кроме того, числом и строкой.

Операнд С задает номер модификатора-функции. При использовании операнда С значение операнда В умножается на значение модификатора-функции. Полученное произведение становится значением, которое изменяет значение параметра, заданного операндом А. Следует отметить, что операнд С определяет номер функции или ее имя (не нужно использовать СЧА FN или FN$ перед ним). Если используется СЧА FN, например, FN3, вычисляется функция номер 3 GPSS. Полученный результат используется для определения второй функции GPSS, которая также вычисляется и ее значение умножается на значение операнда В. Приведем примеры записи блока **ASSIGN**:

ASSIGN 1,754.3
ASSIGN 4+,Q5

ASSIGN 3–,5.85,7
ASSIGN Name,"Plan"
ASSIGN Tr1–,(Normal(32,Sred,SrOtkl),Expdis

В первом примере параметру 1 присваивается 754.3. Во втором примере к значению параметра 4 прибавляется значение длины текущей очереди номер 5. В третьем примере из значения параметра 3 вычитается произведение 5.85 на вычисленное значение функции номер 7. В четвертом примере параметру с именем Name присваивается строка Plan. В пятом примере вычисляются выражение в скобках и функция с именем Expdis, они перемножаются, а полученное произведение вычитается из значения параметра с именем Tr1.

Занятие и освобождение одноканального устройства

В GPSS элементами, которые требуют обслуживания, являются транзакты. Они перемещаются в модели от блока к блоку. Если в какой-то момент активности транзакт занимает ОКУ, то для этого он входит (или пытается войти) в соответствующий блок, описывающий это ОКУ. Блок должен обладать следующими свойствами:

– если ОКУ уже используют, транзакт не может войти в блок и должен ждать в очереди;

– если ОКУ не используют, транзакт может войти в блок, статус ОКУ изменяется на «занято».

Блок, обладающий этими свойствами, является блоком **SEIZE** («Занято»). Вход транзакта в блок **SEIZE** моделирует занятие ОКУ.

После обслуживания вход того же транзакта в другой блок моделирует освобождение ОКУ. Назначением этого блока является изменение состояния ранее занятого ОКУ с «занято» в «незанято». Этим блоком является блок **RELEASE** («Освободить»).

Форматы блоков:

SEIZE A
RELEASE A

В обоих блоках операнд А – это имя занимаемого (освобождаемого) ОКУ. Может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА<параметр>. Планировщик автоматически обеспечивает возникновение транзактов и ОКУ, когда этого требует логика модели. В то время как транзакты находятся в модели временно, ОКУ, используемые в модели, существуют постоянно в течение всего процесса моделирования. Прежде чем освободить ОКУ, транзакт может пройти через неограниченное число блоков. Например:

SEIZE Can
. . .
RELEASE Can

Если ОКУ с именем **Can** не занято, активный транзакт занимает его. Если ОКУ занято, транзакт помещается в список задержки данного ОКУ позади транзактов с таким же приоритетом. Этот транзакт не входит в блок **SEIZE**. Транзакту также отказывается во входе в блок **SEIZE**, если ОКУ с именем **Can** находится в недоступном состоянии.

ОКУ, как уже отмечалось, может иметь имя или номер. В данном случае разрешается записывать вместо операнда А номер непосредственно без предварительного присвоения его имени командой **EQU**. Например:

SEIZE 5

. . .

RELEASE 5

Блоки **SEIZE** и **RELEASE** при необходимости создания других версий модели могут переопределяться. Для этого они должны иметь метки (не путать с операндами А этих блоков).

Имитация обслуживания посредством задержки во времени

Обычно транзакт занимает ОКУ для того, чтобы немедленно начать на нем обслуживание, которое длится некоторый промежуток модельного времени. В течение этого времени транзакт должен прекратить двигаться по модели. Только по истечении времени обслуживания он должен попасть в блок **RELEASE** для освобождения ОКУ.

Для задержки транзакта в течение некоторого интервала модельного времени используется блок **ADVANCE**. Чаще всего этот интервал задается случайной переменной. Как и при использовании блока **GENERATE**, информация, необходимая для описания соответствующего времени обслуживания и его распределения, задается операндами А и В. Формат записи блока **ADVANCE** следующий:

ADVANCE A,[B]

Здесь А – среднее время обслуживания, а В – способ модификации операнда А. Каждый из операндов А и В может быть именем, числом, выражением в скобках, СЧА или СЧА<параметр>.

Как и в блоке **GENERATE**, модификаторы могут быть двух типов: модификатор-интервал и модификатор-функция, т. е. блок **ADVANCE** вычисляет время задержки (приращения модельного времени) такими же способами.

Если задан только операнд А, он вычисляется и используется в качестве времени задержки. Например:

ADVANCE (Normal(12,X$Sredn,X$SreOtk))

Время задержки распределено по нормальному закону со средним значением и среднеквадратичным отклонением, предварительно записанными командой **INITIAL** в сохраняемые ячейки с именами **Sredn** и **SreOtk** соответственно. Для генератора нормального распределения источником случайных чисел, равномерно распределенных в интервале [0, 1], является генератор номер 12.

При задании операндов А и В, В не определяет функцию, оба операнда вычисляются (если они не константы) и в качестве времени задержки выбирается случайное число, равномерно распределенное в интервале (А – В, А + В). Для розыгрыша может быть выбран любой генератор равномерно распределенных случайных чисел. Делается это так же, как и при выборе генератора для блока **GENERATE**. Только номер генератора на странице «**Random Numbers**» («Случайные числа») в журнале настроек модели нужно указать в поле ввода

ADVANCE. По умолчанию используется генератор случайных чисел номер 1. Например:

ADVANCE 56.7,23.2

В данном примере входящий транзакт задерживается на время, равномерно распределенное в интервале от 33.5 до 79.9.

Так же как и в блоке **GENERATE**, при любых способах вычисления времени задержки значение операнда B не должно превышать значение операнда A. Если в приведенном выше примере операнд B взять равным 56.8, в процессе моделирования произойдет остановка по ошибке «Отрицательное приращение времени».

Для задания времени задержки по другому закону, отличному от равномерного, в операнде B записывается модификатор-функция. При обращении к функции определяется некоторое число – значение функции. Оно умножается на значение операнда A. Результат используется как время задержки. Например:

ADVANCE Ring,FN$Exper

В примере вычисляется значение функции с именем **Exper** и умножается на значение переменной пользователя **Ring**, которой предварительно должно быть присвоено числовое значение командой **EQU**.

Как отмечалось ранее, в блоке **GENERATE** можно использовать функции и арифметические переменные, предварительно определенные командами **FUNCTION** и **VARIABLE** или **FVARIABLE**. Но в этих командах не будет ссылки на параметры транзактов, т. к. транзактов еще нет. В операндах блока ADVANCE ссылки на параметры транзактов возможны. Естественно, что этим параметрам пользователем должны быть предварительно присвоены соответствующие значения. Например:

ADVANCE P1
ADVANCE (Exponental(7,0,MX$Sred(P2,P$Stolb)))
ADVANCE SrIntPost,FN*2

В первом примере транзакт задерживается на время, равное значению параметра 1. Во втором примере время задержки определяется по экспоненциальному закону. При этом среднее значение выбирается из элемента матрицы, номера строки и столбца которого содержатся в параметре 2 и параметре с именем **Stolb** соответственно. Для генератора экспоненциального распределения источником равномерно распределенных случайных чисел в интервале [0, 1] является генератор номер 7 (RN7).

В третьем примере время задержки находится как произведение значения переменной пользователя SrIntPost и вычисленного значения функции, номер которой содержится в параметре 2 активного транзакта.

Блок **ADVANCE** никогда не препятствует входу транзакта. Любое число транзактов может находиться в этом блоке одновременно.

Приведем пример использования блоков **SEIZE**, **ADVANCE** и **RELEASE** (рисунок 5.3).

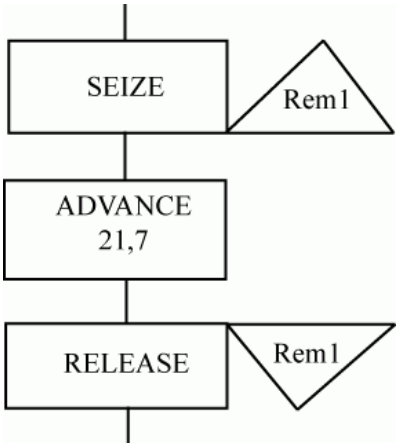


Рисунок 5.3 – Блок-диаграмма процедуры **SEIZE** – **RELEASE**

Транзакт, войдя в блок **SEIZE**, займет ОКУ с символическим именем **Rem1**, задержится там благодаря блоку **ADVANCE** на (21 ± 7) ед. времени и затем покинет его. После того как транзакт войдет в блок **RELEASE**, планировщик попытается продвинуть транзакт в следующий блок модели, и следующий транзакт попытается использовать ОКУ, называемое **Rem1**.

Блок **ADVANCE** можно располагать в любых местах модели, а не только между блоками **SEIZE** и **RELEASE**.

Время задержки может быть также равным нулю. Если время равно нулю, транзакт в блоке **ADVANCE** не задерживается и переходит в следующий блок.

Замечание. Только блоки **GENERATE** и **ADVANCE** позволяют поместить транзакты в список будущих событий. С помощью этих блоков моделируется продолжительность какого-либо события или промежуток времени между наступлением каких-либо событий.

Проверка состояния одноканального устройства

Представим себе, что в какой-либо системе, модель которой вы собираетесь разработать, имеются, например, два канала передачи данных – два ОКУ. Естественно, что если один канал занят, то выясняется свободен ли второй канал. Если он не занят, надо попытаться передать по этому каналу. Но как проверить состояние каналов в модели?

В GPSS World блок **GATE** используется для определения состояния устройств без изменения их состояния (рисунок 5.4).

Формат записи:

GATE X A,[B]

Блок **GATE** работает в двух режимах:

– отказа во входе;

– разрешении во входе и альтернативном выходе.

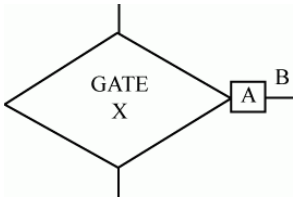


Рисунок 5.4 – Блок-диаграмма процедуры **GATE**

При работе в режиме отказа блок **GATE** не пропускает транзакты, если соответствующий объект не находится в требуемом состоянии. Если же поставленное в блоке **GATE** условие удовлетворяется, то активный транзакт входит в него и затем переходит к следующему по порядку блоку. Операнд А определяет имя или номер ОКУ.

Условие задается одним из следующих условных операторов X, связанных с ОКУ:

– **NU** – ОКУ, заданное операндом А, свободно;

– **U** – ОКУ, заданное операндом А, занято.

Как видно, тип проверяемого объекта неявно задается условным оператором (позже мы рассмотрим применение блока **GATE** для проверки и других объектов аппаратной категории).

Операнд В содержит номер следующего блока для входящего транзакта, когда условный оператор имеет значение «ложь». Операнды А и В могут быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА<параметр>. Если операнд В не используется, то проверка проводится в режиме отказа. Если результат этой проверки не будет иметь значения «истина», то транзакт помещается в список повторных попыток проверяемого объекта. Когда состояние любого из объектов меняется, заблокированный транзакт снова активизируется, повторяется проверка заданного блоком **GATE** условия. Если это условие выполняется, транзакту разрешается войти в блок **GATE** и далее перейти к следующему по порядку блоку. Например:

GATE U Zrk
GATE NU (FN$Expert+4)
GATE NU Bat,Oper

В первом примере блок **GATE** не пропустит транзакт при условии незанятости ОКУ с именем **Zrk**, во втором случае – когда занято ОКУ, номер которого определяется как результат вычисления и последующего округления выражения в скобках **FN$Expert+4**. В третьем примере в случае занятости ОКУ с именем **Bat** транзакт будет направлен к блоку с именем **Oper**.

Следует отметить, что несмотря на удобство применения блока **GATE**, в некоторых случаях при отсутствии операнда В это может привести к увеличению машинного времени за счет проведения проверок для заблокированных транзактов. Уменьшение числа проверок и исключение из проверяемых транзактов, для которых вряд ли когда-нибудь результат проверки будет истинным, может быть

организовано с помощью блоков **LINK** и **UNLINK**. Методы применения этих блоков будут рассмотрены позже.

Для проверки состояния различных объектов, в том числе и ОКУ, можно использовать также булевы переменные и блок **TEST**.

5.3 Методы сбора статистики в имитационной модели

Регистратор очереди

В GPSS объекты типа «очередь» вводятся для сбора статистических данных. Эта статистика должна дать ответы на следующие вопросы:

1. Сколько раз транзакты приходили в очередь?
2. Сколько пришедших транзактов фактически присоединились к очереди (задержались) и сколько их сразу без задержки заняло ОКУ?
3. Каково было максимальное значение длины очереди?
4. Какое среднее число ожидающих транзактов в очереди?
5. Каково было среднее время ожидания тех транзактов, которым пришлось ждать?

GPSS обеспечивает возможность сбора такой статистики с помощью средства, называемого регистратором очереди. При использовании регистратора очереди в тех точках модели, где ресурсы ограничены, планировщик начинает автоматически собирать статистику, описывающую ожидание (если оно есть), возникающее в этих точках. Регистраторы очередей различают заданием имен. Условия задания имен те же, что и у устройств. Разработчик вносит регистратор очереди в модель с помощью пары взаимодополняющих блоков **QUEUE** («Встать в очередь») и **DEPART** («Покинуть очередь»).

Формат записи блока **QUEUE** следующий:

QUEUE A,[B]

Блок **QUEUE** увеличивает длину очереди. Операнд А задает номер или имя очереди, к длине которой добавляются единицы.

Операнд В определяет число единиц, на которое увеличивается текущая длина очереди. Если операнд В не используется, то прибавляется единица.

Рассмотрим примеры записи блока **QUEUE**.

QUEUE RemQ

Увеличивает длину очереди **RemQ** на единицу при входе каждого транзакта.

QUEUE P14,P1

Увеличивает длину очереди, номер или имя которой задан в параметре **P14** транзакта, на число единиц, заданных в параметре **P1**.

Формат записи блока **DEPART** имеет вид

DEPART A,[B]

Блок **DEPART** служит для уменьшения длины очереди. Операнд А задает номер или имя очереди, длину которой надо уменьшить. Операнд В задает число единиц, на которое уменьшается длина очереди. Это число не должно превышать текущую длину очереди. Если операнд В не используется, то по умолчанию длина очереди уменьшается на единицу. Операнды А и В в блоках **QUEUE** и

DEPART могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА<параметр>.

Приведем примеры записи блока **DEPART**.

DEPART RemQ

Уменьшает длину очереди **RemQ** на единицу.

DEPART V3,(V4+3.7)

Операнды A и B заданы арифметической переменной **V3** и выражением в скобках, которое также содержит арифметическую переменную **V4**. При входе транзакта в блок **DEPART** переменная и выражение в скобках вычисляются и округляются. После этого длина очереди, номер которой есть значение переменной **V3**, уменьшается на значение выражения в скобках **(V4+3.7)**.

Очереди, как и ОКУ, вместо операнда A разрешается записывать номер без предварительного присвоения его имени командой **EQU**. Например:

QUEUE 5

…

DEPART 5

Рассмотрим использование блоков **QUEUE** и **DEPART** на примере представленной блок-диаграммы (рисунок 5.5). Ожидание может возникнуть ввиду занятости ОКУ с именем **Rem1**. Для сбора статистики об ожидании введем регистратор очереди и дадим ему имя **RemQ**.

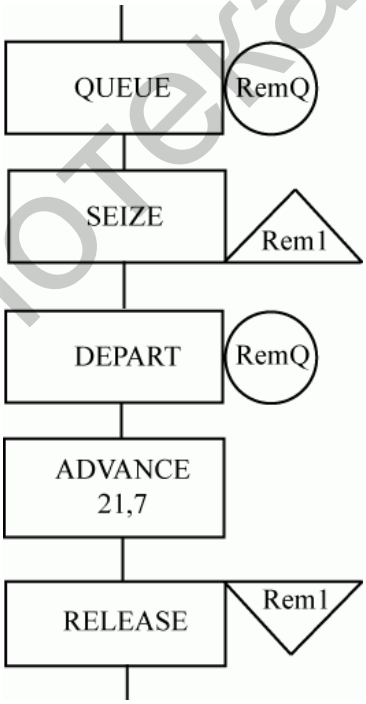


Рисунок 5.5 – Блок-диаграмма процедуры **QUEUE – RELEASE**

Если транзакт вошел в сегмент в момент, когда ОКУ **Rem1** не занято, транзакт входит в блок **QUEUE**. Далее транзакт пытается войти в блок **SEIZE** и, поскольку **Rem1** свободно, эта попытка оказывается успешной. Состояние **Rem1**

меняется на «занято», и далее транзакт сразу попадает в блок **DEPART**. Выполняется соответствующая подпрограмма и транзакт попадает в блок **ADVANCE**, где задерживается на некоторое время, вычисленное в соответствии с распределением (21 ± 7).

Предположим теперь, что ОКУ **Rem1** находится в занятом состоянии и следующий транзакт входит в сегмент модели. Он проходит в блок **QUEUE** и получает далее отказ, поскольку **Rem1** находится в занятом состоянии. Транзакт перестает двигаться, оставаясь в блоке **QUEUE**.

Позднее, когда транзакт, находящийся на обслуживании в устройстве, покидает его, ожидающий транзакт опять попытается войти в блок **SEIZE**. На этот раз попытка окажется успешной. Двигаясь дальше, транзакт войдет в блок **DEPART**, уменьшая значение счетчика содержимого очереди на единицу (по умолчанию, т. к. операнд B не используется).

Длина очереди не может быть отрицательной. Если такое случится, произойдет остановка по ошибке «Запрещенная попытка сделать содержимое очереди отрицательным».

Остановка по ошибке «Недопустимое отрицательное число в операторе GPSS. Operand B» происходит и тогда, когда при создании очереди делается попытка уменьшить ее длину.

Например:

QUEUE Server,3

Статистические таблицы

Для получения плотности распределения, ее интегральных относительных частот, среднего значения и стандартного отклонения некоторых аргументов, которыми могут быть СЧА (например, времени нахождения транзакта в модели или задержки в ее отдельных частях, длин очередей, содержимого МКУ, коэффициентов использования устройств и т. д.), используются статистические таблицы **TABLE** и **QTABLE**.

Команда описания таблицы **TABLE** имеет следующий формат:

Name TABLE A,B,C,D

Команда определяет аргумент, а также число и ширину частотных интервалов (классов). Метка **Name** определяет имя таблицы.

Операндом A задается аргумент таблицы – элемент данных, чье частотное распределение будет табулироваться. Операнд может быть именем, выражением в скобках или СЧА.

Операндом B задается верхний предел первого частотного интервала.

Операнд C задает ширину частотного интервала – разницу между верхней и нижней границей каждого частотного класса.

Операнды B и C могут быть числами или string.

Операндом D задается число частотных интервалов (положительное целое число).

Для сбора данных транзакт должен войти в блок **TABULATE** с тем же именем таблицы, которое определено в блоке **TABLE**. Блок помещается в ту точку модели, которая соответствует исследуемому объекту.

Блок **TABULATE** имеет следующий формат:

TABULATE A,[B]

Операндом A задается имя таблицы, в которую табулируется значение аргумента.

Операндом B (весовой коэффициент) задается число единиц, которые должны быть занесены в тот частотный интервал, в который попало значение аргумента. Если операнд B отсутствует, то по умолчанию эта величина равна единице.

Операнды A и B могут быть именем, выражением в скобках, СЧА или СЧА<параметр>. Кроме того, операнд A может быть только положительным целым числом, а операнд B – положительным числом. Например:

VrRem TABLE P$ReaLvs,8.2,5.5,10

. . .

TABULATE VrRem

Оператором **TABLE** описывается таблица с именем **VrRem**. Аргументом таблицы является СЧА **P$ReaLvs** с верхним пределом первого интервала 8.2, шириной 5.5 и числом интервалов 10. Каждое значение табулируемого аргумента **P$ReaLvs**, меньшее или равное 8.2, увеличивает частоту первого частотного класса таблицы на единицу. Если аргумент таблицы не попадает в первый частотный класс, класс определяется делением значения аргумента на операнд C оператора **TABLE**. Например, значение **P$ReaLvs** равно 30.25. Тогда 30.25 / 5.5 = 5.5 и, следовательно, частота шестого класса будет увеличена на единицу. Если аргумент A таблицы больше B + C # D = 8.2 + 5.5 # 10 = 63.2, изменен будет на единицу последний (десятый) класс. Одновременно корректируются текущие значения СЧА таблицы: счетчик входов в таблицу **TC**, среднее время ожидания **TB** и стандартное отклонение времени ожидания **TD**. Собранная в таблице статистика выводится в стандартный отчет.

При использовании в операнде B блока **TABULATE** весового коэффициента, например **TABULATE VrRem,3**, последний будет добавляться к значению частоты частотного класса. Весовой коэффициент применяется также для среднего и стандартного отклонения, что равносильно нескольким входам в блок **TABULATE**.

Таким образом, команда **TABLE** вместе с блоком **TABULATE** служат для табулирования любого СЧА.

Кроме таблиц **TABLE** могут использоваться **Q**-таблицы, являющиеся средством получения распределения только времени пребывания транзакта в очереди. Формат команды описания **Q**-таблицы такой же, как и **TABLE**. Отличие состоит в том, что операндом A задается имя очереди. Назначение операндов B, C, D такое же, что и в команде описания **TABLE**. Операнд B может быть нулем или положительным числом. Для создания в модели такой таблицы ее нужно предварительно определить с помощью команды **QTABLE** следующего формата:

Name QTABLE A,B,C,D

Например:

VTime QTABLE Dlina,18.2,4.3,6

Dlina – имя очереди к устройству, а не СЧА, как в случае использования **TABLE**.

При прохождении транзакта через блоки **QUEUE** и **DEPART** его время ожидания фиксируется, и к счетчику частотного интервала таблицы, в который попало это время, добавляется единица. Одновременно в таблице накапливается информация для вычисления среднего значения и среднеквадратичного отклонения времени ожидания. Следует обратить внимание, что при использовании **QTABLE** информация в таблицу заносится автоматически при входе транзакта в блоки **QUEUE** и **DEPART** и никаких специальных мер, т. е. блока **TABULATE** при этом не требуется. По окончании моделирования собранная в таблице информация также выводится в стандартном отчете GPSS.

Методы изменения маршрутов движения транзактов в модели

Для изменения маршрутов движения транзактов в модели применяются блоки **DISPLACE**, **LOOP**, **GATE**, **TEST** и **TRANSFER**. Здесь мы рассмотрим методы применения блоков **TRANSFER** и **DISPLACE**.

Блок TRANSFER

Блок **TRANSFER** («Передать») предназначен для передачи входящего в него транзакта в любой другой блок модели. Он имеет следующий формат:

TRANSFER [A],[B],[C],[D]

Все режимы блока **TRANSFER**, кроме безусловного, выборочные, т. е. отличаются друг от друга способом выбора очередного блока, к которому должен быть направлен активный транзакт. Операнд А задает этот режим выбора. Существуют следующие девять режимов работы блока **TRANSFER**:

1) по умолчанию – безусловный;
2) статистический – выбор случайным образом одного из двух блоков;
3) **BOTH** – последовательный выбор одного из двух блоков;
4) **ALL** – последовательный выбор одного из нескольких блоков;
5) **PICK** – выбор случайным образом одного из нескольких блоков;
6) **FN** – функциональный;
7) **P** – параметрический;
8) **SBR** – подпрограммный;
9) **SIM** – одновременный.

Операнд А может принимать указанные выше значения, а также может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА*<параметр>.

Операнды В и С задают возможные значения номеров следующих блоков или их положение. Они могут быть такими же, как и операнд А. Использование этих значений будет описано далее при рассмотрении указанных выше режимов работы. Если операнд В опущен, то планировщик записывает вместо него номер блока, следующего за блоком **TRANSFER**.

Рассмотрим режимы безусловной передачи и статистической передачи блока **TRANSFER**, как наиболее часто используемые в моделях.

Режим безусловной передачи

В режиме безусловной передачи операнд А не используется. Операнд В указывает имя блока, в который транзакт должен попытаться войти. Блок **TRANSFER** не может отказать транзакту во входе.

Например:

TRANSFER ,Oper

После входа транзакт сразу же пытается войти в блок с меткой **Oper**. Если этот блок отказывает во входе, транзакт остается в блоке **TRANSFER**.

Рассмотрим еще один пример использования безусловного режима блока **TRANSFER** (рисунок 5.6). Пусть требуется поток обслуженных транзактов перед удалением из модели разделить на четыре составляющие. Первый параметр каждого транзакта имеет одно из четырех присвоенных ранее значений: 1, 2, 3 или 4. Вначале для разделения потока используем блок **TEST**.

```
. . .
      TEST E      P1,1,Met2
Met2  TERMINATE
      TEST E      P1,2,Met3
      TERMINATE
Met3  TEST E      P1,3,Met4
      TERMINATE
Met4  TERMINATE
```

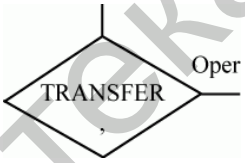


Рисунок 5.6 – Обозначение блока **TRANSFER**

Теперь этот же фрагмент модели перепишем с использованием блока **TRANSFER**.

```
. . .
Met1  TRANSFER    ,(Met1+P1)
      TERMINATE
      TERMINATE
      TERMINATE
      TERMINATE
```

В блоке **TRANSFER** в качестве операнда В указано выражение в скобках. При входе активного транзакта выражение вычисляется, т. е. к номеру, который присвоен планировщиком блоку с меткой **Met1**, прибавляется значение первого параметра. В итоге получается номер блока, к которому и направляется транзакт.

Режим статистической передачи

Когда операнд А используется и не является зарезервированным словом, блок **TRANSFER** работает в режиме передачи транзакта в один из двух блоков случайным образом.

Значение операнда A, записываемого после точки, рассматривается как трехзначное число, показывающее (в долях от тысячи), какая доля входящих в блок транзактов должна быть направлена в блок C. Остальные транзакты направляются в блок B или к следующему по номеру блоку, если операнд B опущен.

Числовое значение операнда A может быть задано любым СЧА. При этом возможны следующие случаи:

– значение операнда A меньше или равно нулю;

– значение операнда A равно или больше 1000;

– значение операнда A больше нуля, но меньше 1000.

Если вычисленное значение операнда A меньше или равно нулю, то будет производиться безусловная передача транзакта к блоку B. Если значение операнда A больше или равно 1000, то будет осуществляться безусловная передача транзакта к блоку C. В третьем случае блок **TRANSFER** работает в обычном режиме.

Например:

TRANSFER .P5,,Rrw

Трехзначное число, записанное в параметре 5 транзакта, входящего в блок **TRANSFER**, интерпретируется как вероятность (в долях от тысячи) того, что транзакт будет передаваться блоку **Rrw**, а в остальных случаях – следующему блоку, т. к. операнд B не используется.

Режим статистической передачи удобно использовать, например, в таких случаях. При моделировании работы цеха по производству деталей известно, что 12,5 % изготовленных деталей бракуется. В модели это можно реализовать так:

. . .

TRANSFER .125,Sam,Wzw

. . .

Транзакты, имитирующие изготовленные в цехе детали, в 12,5 % случаев будут направлены к блоку с меткой **Wzw**, а в остальных 87,5 % случаях – к блоку с меткой **Sam**.

Можно указать генератор – источник случайных чисел. Для этого нужно выбрать **Edit/Settings** (Правка/Настройки). Затем выбрать страницу **Random Numbers** (Случайные числа) и ввести номер генератора в поле ввода, отмеченное **TRANSFER**. После инсталляции по умолчанию используется генератор номер 1.

Блок DISPLACE

Блок **DISPLACE** предназначен для нахождения любого транзакта и перемещения его к новому блоку. Блок **DISPLACE** имеет следующий формат:

DISPLACE A,B,[C],[D]

Операнд A – номер транзакта, который нужно переместить.

Операнд B – метка блока, к которому перемещается транзакт, указанный операндом A.

Операнд C – номер параметра перемещаемого транзакта, в который записывается оставшееся до конца его обслуживания время, если он находился в списке будущих событий.

Операнд D – метка альтернативного блока для транзакта.

Операнды A, B, C и D могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА<параметр>. Например:

DISPLACE (P2+32),Term3,Ostatok,Met2

Операнд A указан выражением в скобках. Это выражение вычисляется и округляется до целого. Полученный результат является номером транзакта, который следует переместить. Далее блок **DISPLACE** отыскивает этот транзакт. При этом возможны следующие случаи:

1) транзакт есть в модели и не находится в списке будущих событий;

2) транзакт есть в модели и находится в списке будущих событий;

3) транзакта с нужным номером нет в модели.

В первом случае транзакт перемещается к блоку с меткой **Term3**. Во втором случае определяется время, оставшееся до его повторного ввода в процесс моделирования, и записывается в параметр с именем **Ostatok**. Если параметра с таким именем нет, он создается. Транзакт также перемещается к блоку с меткой **Term3**. В третьем случае, т. е. когда в модели нет транзакта с нужным номером, активный транзакт, вошедший в блок **DISPLACE**, направляется к блоку с меткой **Met2**. Если операнда D нет, активный транзакт переходит к следующему блоку.

Когда транзакт перемещается к новому блоку, он исключается из следующих списков:

1) будущих событий;

2) отложенных прерываний (для прерывающих транзактов);

3) задержки (в порядке приоритета);

4) пользователя;

5) повторных попыток.

Но транзакт не исключается из списков текущих событий, прерываний (для прерванных транзактов), групп.

При перемещении прерванные выполнения в устройствах не сбрасываются. Это означает, что перемещаемый транзакт продолжает занимать устройство. Следовательно, если необходимо, нужно освободить устройство.

Прерывание функционирования одноканального устройства

Если на входе ОКУ образуется очередь, выбор транзакта для занятия ОКУ после его освобождения происходит в порядке поступления (**FIFO**) (для транзактов с равными приоритетами), а также с учетом приоритета, указанного операндом E блока **GENERATE**. При этом очередной транзакт с большим приоритетом ждет окончания обслуживания предыдущего транзакта независимо от его приоритета. Приоритет учитывается только в образующейся очереди. В ней транзакты выстраиваются в приоритетном порядке.

Такой режим функционирования ОКУ организуется блоками **SEIZE** и **RELEASE**.

Однако может возникнуть необходимость смоделировать ситуацию, когда очередной транзакт должен занять ОКУ, прервав обслуживание предыдущего транзакта. Такое прерывание называется «захватом» ОКУ и моделируется блоком **PREEMPT** («Захватить»). Формат блока следующий:

PREEMPT A,[B],[C],[D],[E]

Операнд A – имя или номер захватываемого ОКУ.

Когда ОКУ свободно (рисунок 5.7), блок **PREEMPT** работает так же, как и блок **SEIZE**.

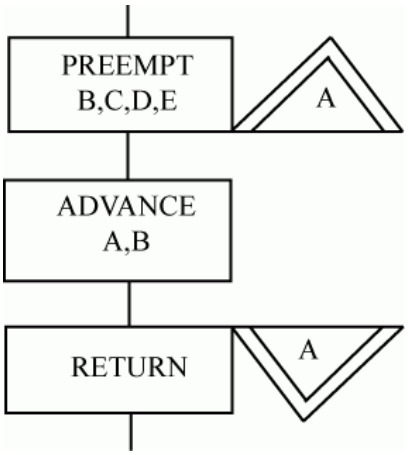


Рисунок 5.7 – Блок-диаграмма процедуры прерывания

При занятом ОКУ блок **PREEMPT** функционирует либо в приоритетном режиме, либо в режиме прерывания. Режимы определяются операндом B:

1) **PR** – приоритетный;

2) по умолчанию (B не используется) – режим прерывания.

Прерывание в приоритетном режиме

В приоритетном режиме прервать обслуживание предыдущего (обслуживаемого) транзакта, т. е. «захватить» ОКУ, может только транзакт с большим приоритетом. Если приоритет претендующего на занятие ОКУ транзакта равен или ниже приоритета обслуживаемого транзакта, он помещается в список задержки ОКУ последним в своем приоритете.

Что делать с транзактом, обслуживание которого прерывается, определяют операнды C, D и E.

Операнд C – имя или номер блока, куда должен быть направлен прерванный транзакт.

Операнд E при значении **RE** определяет режим удаления прерванного транзакта.

Операнд D – номер параметра прерванного транзакта, в который записывается оставшееся до завершения обслуживания время.

Операнды A, C, D и E могут быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА<параметр>.

Транзакт, захвативший ОКУ, освобождает его от захвата вхождением в блок **RETURN**. Формат блока следующий:

RETURN A

Операнд A – имя или номер освобождаемого ОКУ. Например: **RETURN Rem1** (освободить от захвата ОКУ **Rem1**).

Прерывание в режиме «захвата»

В режиме «захвата», если ОКУ уже используется, активный транзакт помещается в список отложенных прерываний или «захватывает» ОКУ. Прерывание обслуживания сразу, а не помещение транзакта в список, происходит тогда, когда список отложенных прерываний пуст и обслуживаемый транзакт сам не является «захватчиком».

Транзактам из списка отложенных прерываний предоставляется право занять ОКУ ранее, чем прерванным транзактам или транзактам из списка задержки ОКУ.

Проверка состояния одноканального устройства, функционирующего в приоритетном режиме

Проверка состояния ОКУ в приоритетном режиме может проводиться блоком **GATE**, а также с использованием булевой переменной и блока **TEST**.

Рассмотрим проверку состояния ОКУ блоком **GATE**.

Условие проверки задается одним из следующих условных операторов X:

1) I – ОКУ, заданное операндом A, прервано;

2) NI – ОКУ, заданное операндом A, не прервано.

Например:

```
GATE  I   Stan
GATE NI   (V$Rasp–3)
GATE I    Print,Udal
GATE I    *1
```

В первом примере блок **GATE** пропустит транзакт, когда ОКУ **Stan** будет прервано. Во втором примере транзакт пройдет к следующему блоку, когда не прервано ОКУ, номер которого определяется как результат вычисления и последующего округления до целого выражения в скобках (**V$Rasp–3**). В третьем примере в случае прерывания ОКУ **Print** транзакт будет направлен к блоку с меткой **Udal**.

В первом и втором примерах блок **GATE** работает в режиме отказа во входе в случае невыполнения условия. Здесь отсутствие операнда B может привести к увеличению машинного времени моделирования. Однако в некоторых случаях такой режим, наверное, можно использовать.

Недоступность одноканального устройства

Для моделирования неисправностей ОКУ и других ситуаций в GPSS World предусмотрены блоки, реализующие недоступность и доступность ОКУ. При использовании этих блоков статистика ОКУ не искажается. Здесь имеется в виду следующее. Для моделирования, например, неисправностей можно использовать и режим прерывания (**PREEMPT**). Однако при этом транзакты, вызывающие прерывания (имитирующие отказы ОКУ), учитываются в статистике, как и транзакты, обслуженные при исправном функционировании ОКУ. А это неправильно, вследствие чего и искажается статистика ОКУ.

Перевод в недоступное состояние и восстановление доступности

Недоступность ОКУ моделируется блоком **FUNAVAIL** (рисунок 5.8), (символ **F** означает ОКУ, **UNAVAIL** – недоступный). При использовании этого блока статистика ОКУ не искажается.

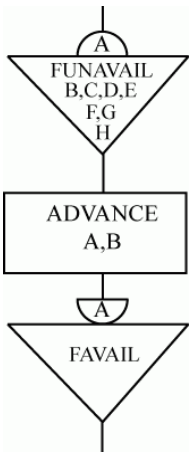


Рисунок 5.8 – Обработка транзактов блоком **FUNAVAIL**

Формат блока имеет следующий вид:

FUNAVAIL A,[B],[C],[D],[E],[F],[G],[H]

Блок делает недоступным ОКУ с именем или номером, указываемым операндом.

Все транзакты, обрабатываемые блоком **FUNAVAIL**, разделяются на три класса, которые и определяют назначение операндов:

1) транзакт, занимающий ОКУ (по **SEIZE** или **PREEMPT**) в момент перевода его в недоступное состояние (операнды B, C, D);

2) ранее прерванные транзакты, находящиеся в списке прерываний (операнды E, F);

3) транзакты, находящиеся в списке отложенных прерываний и в списке задержки ОКУ (операнды G, H).

Операндом B задаются режимы обработки транзакта, занимающего ОКУ в момент перевода его в недоступность:

1) **CO** – режим продолжения: продолжить обработку занимающего ОКУ транзакта во время недоступности;

2) **RE** – режим удаления: удалить и направить занимающий ОКУ транзакт к блоку, метка которого должна быть указана операндом C;

3) по умолчанию – прервать обработку и поместить в список прерываний ОКУ; после восстановления доступности этот транзакт может занять ОКУ и «дообслужиться».

Операнд C – метка блока, в который будет направлен в режиме удаления транзакт, занимавший ОКУ в момент перевода его в недоступное состояние.

Операнд D – номер или имя параметра транзакта, занимавшего ОКУ в момент перевода его в недоступное состояние; если он будет удален (режим **RE**), т. е. исключен, в этот параметр будет записано время, оставшееся удаленному транзакту до конца обслуживания.

Операндом E задаются режимы обработки транзактов, находящихся к моменту перевода ОКУ в недоступное состояние в списке прерываний, т. е. тех транзактов, обслуживание которых на данном ОКУ было ранее прервано:

1) **CO** – режим продолжения: продолжить работу ОКУ во время недоступности, т. е. обслуживать транзакты из списка прерываний;

2) **RE** – режим удаления: удалить и направить транзакты из списка прерываний к новому блоку, метка которого должна быть указана операндом F;

3) по умолчанию – оставить ранее прерванные транзакты в списке прерываний ОКУ и запретить им занимать его во время недоступности.

Операнд F указывает метку блока, к которому будут направлены транзакты из списка прерываний ОКУ, из-за чего они не могут находиться в СБС, поэтому для них нет возможности занесения в их параметры времени, оставшегося до конца обслуживания.

Операнд F может использоваться и тогда, когда отсутствует операнд E (по умолчанию). В этом случае для перемещенных к новому блоку транзактов прерывание обслуживания сохраняется.

Операндом G задаются режимы обработки транзактов, находящихся к моменту перевода ОКУ в недоступное состояние в списке отложенных прерываний, т. е. ожидающих выполнения с прерыванием, и в списке задержки:

1) **CO** – режим продолжения: продолжить работу ОКУ во время недоступности, т. е. обслуживать транзакты из списка отложенных прерываний и списка задержки;

2) **RE** – режим удаления: удалить и направить транзакты из списка отложенных прерываний и списка задержки к новому блоку, метка которого должна быть указана операндом H;

3) по умолчанию – оставить транзакты в списке отложенных прерываний и списке задержки ОКУ и запретить им занимать его во время недоступности.

Операндом H указывается метка нового блока, к которому в режиме удаления (**RE**) направляются транзакты из списка отложенных прерываний и списка задержки. Когда операнд G не используется, нельзя использовать и операнд H.

Недоступность ОКУ сохраняется до тех пор, пока транзакт, вызвавший переход в недоступное состояние, не войдет в блок **FAVAIL A**.

Блок **FAVAIL** изменяет состояние ОКУ на доступное, т. е. восстанавливает обычный режим вхождения транзактов в ОКУ. Все транзакты, ожидающие доступного состояния ОКУ, указанного операндом A, активизируются и могут попытаться занять его.

Замечание 1. Операнды B–H относятся только к транзактам указанных ранее трех классов. Другие транзакты, которые пытаются прервать ОКУ, уже находящееся в состоянии, в эти классы не входят и операнды B–H не имеют к ним никакого отношения.

Замечание 2. Перевод ОКУ в недоступное состояние и разрешение продолжать обработку транзактов из указанных трех классов дает возможность имитировать не только отказы, но и различные дисциплины обслуживания.

Проверка состояний недоступности и доступности одноканального устройства

Проверка состояния ОКУ в режиме недоступности проводится блоком **GATE**. Условие проверки задается одним из следующих условных операторов X:

1) **FNV** – ОКУ, заданное операндом А, недоступно;
2) **FV** – ОКУ, заданное операндом А, доступно.

Например:

GATE FNV Stan
GATE FV (FN$Rasp–X$Col)
GATE FNV Print,Udal

В первом примере блок **GATE** пропустит транзакт, когда ОКУ **Stan** будет недоступно.

Во втором примере транзакт пройдет к следующему блоку, когда доступно ОКУ, номер которого определяется как результат вычисления и последующего округления до целого выражения в скобках (**FN$Rasp–X$Col**).

В третьем примере в случае доступности ОКУ **Print**, т. е. не выполнения заданного в блоке **GATE** условия, транзакт будет направлен к блоку с меткой **Udal**.

В первом и втором примерах блок **GATE** работает в режиме отказа, если условия не выполняются. Здесь отсутствие операнда В может привести к увеличению времени моделирования.

Сокращение машинного времени и изменение дисциплин обслуживания методом применения списков пользователя

При движении по модели транзакты могут быть заблокированы, например, как отмечалось ранее, при проверке состояния ОКУ блоками **GATE** и **TEST**. Если заблокированные транзакты находятся в списке текущих событий (СТС), то при большом их количестве планировщик расходует много времени на просмотр СТС с целью выбора очередного транзакта для продвижения.

Для экономии машинного времени заблокированные транзакты целесообразно помещать в списки пользователя и оставлять их там до тех пор, пока не будут выполнены условия, позволяющие дальнейшее продвижение этих транзактов. Кроме того, этим предоставляется возможность организовать различные дисциплины очередей, отличающиеся от дисциплины **FIFO**, реализованной в списке текущих событий.

Список пользователя представляет собой некоторый буфер, в который могут временно помещаться транзакты, выведенные из СТС. В отличие от списков текущих и будущих событий транзакты вводятся в список пользователя и выводятся из него не автоматически, а по решению пользователя в соответствии с логикой модели при помощи специальных блоков.

Для ввода транзактов в список пользователя служит блок **LINK** («Ввести в список»), который может быть использован в условном и безусловном режимах.

Ввод транзактов в список пользователя в безусловном режиме

В безусловном режиме блок **LINK** (рисунок 5.9) имеет следующий формат записи:

[имя] LINK A,B

Операндом А задается имя или номер списка пользователя, в который безусловно помещается транзакт, вошедший в блок **LINK**.

Операнд B определяет, в какое место списка пользователя следует поместить вошедший транзакт, допустимые значения которого следующие:

1) **FIFO** – транзакт помещается в конец списка;

2) **LIFO** – транзакт помещается в начало списка;

3) **PR** – транзакты помещаются по убыванию приоритета;

4) **P** – транзакты помещаются позади тех транзактов, значения соответствующего параметра которых меньше (в порядке возрастания значения параметра);

5) **M1** – транзакты помещаются в порядке возрастания относительного времени их пребывания в модели.

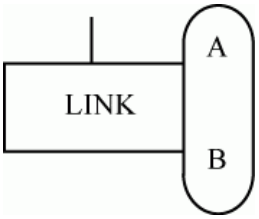


Рисунок 5.9 – Обозначение блока **LINK**

В качестве операнда B могут использоваться и другие СЧА, кроме указанных ранее СЧА транзактов: арифметическая переменная, функция, а также выражение в скобках. В этом случае выполняются вычисления указанного операндом B для активного транзакта и для всех остальных транзактов, уже находящихся в списке пользователя, начиная с начала очереди. После этого производится упорядочивание транзактов в списке пользователя по убыванию вычисленного значения. Например, блок **LINK 3,FIFO** помещает транзакты в конец списка пользователя с номером 3 в порядке их поступления в блок. Блок **LINK Otst,P$Pol** помещает транзакты в список пользователя с именем **Otst**, упорядочивая их по возрастанию значения параметра с именем **Pol**.

Условия, при которых транзакт помещается в список пользователя, в безусловном режиме проверяются средствами, предусмотренными разработчиком модели. Например, направить транзакт в список пользователя в случае занятости ОКУ можно так, как показано ниже.

```
. . .
      GATE NU Rem1,Wait
      SEIZE   Rem1
      . . .
Wait  LINK    Otst,FIFO
      . . .
```

Если ОКУ **Rem1** занято, блок **GATE** не впускает транзакт в блок **SEIZE**, а направляет его в блок **LINK** с именем **Wait**, и транзакт вводится в конец списка пользователя с именем **Otst**.

В том же фрагменте модели список пользователя можно разместить и иначе:

```
. . .
        GATE U   Rem1,Met1
        LINK     Otst,FIFO
Met1    SEIZE    Rem1
        . . .
```

Здесь ОКУ **Rem1** проверяется на занятость. Если ОКУ занято, транзакт проходит к следующему блоку **LINK** и помещается в список пользователя с именем **Otst**. В случае незанятости ОКУ транзакт направляется к блоку **SEIZE** с меткой **Met1** и занимает свободное ОКУ.

В только что рассмотренных примерах предполагается, что список пользователя неограничен, т. е. в него может помещаться любое количество транзактов. При моделировании реальных систем список пользователя может использоваться для имитации, например, входного накопителя, емкость которого, как правило, ограничена. Это ограничение можно реализовать следующим образом.

```
Emk     EQU         10
        . . .
        GATE NU     Rem1,Wait
        SEIZE       Rem1
        . . .
Wait    TEST L      CH$Otst,Emk,Term1
        LINK        Otst,LIFO
```

Если ОКУ **Rem1** занято, то блок **GATE** не впускает транзакт в блок **SEIZE**, а направляет его в блок **TEST** с меткой **Wait**, находящийся перед блоком **LINK**. Если текущее содержимое списка пользователя с именем **Otst** меньше заданной емкости **Emk**, транзакт проходит в список пользователя, в противном случае направляется к блоку с меткой **Term1**.

Приведем еще возможный вариант этого же фрагмента модели.

```
Emk EQU         10
. . .
        GATE U   Rem1,Met1
        TEST L   CH$Otst,Emk,Term1
        LINK     Otst,LIFO
Met1    SEIZE    Rem1
```

Если ОКУ **Rem1** занято, блок **GATE** пропускает транзакт к блоку **TEST**. Если текущее содержимое списка пользователя с именем **Otst** меньше заданной емкости **Emk**, транзакт проходит в список пользователя, в противном случае направляется к блоку с меткой **Term1**. Если ОКУ не занято, транзакт направляется к блоку **SEIZE** с меткой **Met1** и занимает свободное ОКУ **Rem1**.

Вывод транзактов из списка пользователя в условном режиме

Для вывода одного или нескольких транзактов из списка пользователя и помещения их обратно в список текущих событий служит блок **UNLINK** («Вывести из списка») (рисунок 5.10), имеющий следующий формат:

```
[имя]       UNLINK X            A,B,C,[D],[E],[F]
```

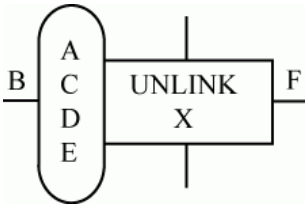


Рисунок 5.10 – Обозначение блока **UNLINK**

Операндом A указывается имя или номер списка пользователя.

Операнд B – метка блока, в который переходят выведенные из списка пользователя транзакты.

Операндом C указывается число выводимых транзактов или ключевое слово ALL для вывода всех находящихся в списке транзактов. По умолчанию, т. е. когда не используется операнд C, берется слово.

Операнды D и E вместе с условным оператором X определяют способ и условия вывода транзактов из списка пользователя. Значения оператора X те же, что и в блоке C. В случае когда условный оператор X должен использоваться, но не указан, по умолчанию он принимает значение E (равенство). Если операнды D и E не используются, не указывается и условный оператор X. В этом случае транзакты выводятся с начала списка, а количество выводимых транзактов определяется обязательным операндом C.

Операнд D может быть булевой переменной, номером параметра транзакта, ключевым словом **BACK**.

Если операнд D является булевой переменной, операнд E и оператор X не используются. Булева переменная вычисляется относительно транзакта, находящегося в списке пользователя. Если результат не нуль, т. е. условие вывода выполняется, транзакт выводится. Количество выводимых транзактов определяется операндом C. Однако выведено может быть и меньше, чем указано операндом C: по числу ненулевых результатов вычисления булевой переменной. Кроме того, и транзактов в списке пользователя может быть меньше, чем указано операндом C.

Если операндом D указано ключевое слово **BACK**, то операнд E и условный оператор X не используются, а транзакты выводятся с конца списка в количестве, определяемом обязательным операндом C.

Если операнд D не булева переменная и не ключевое слово **BACK**, должны быть указаны операнд E и условный оператор X. Операнд D вычисляется относительно транзакта, находящегося в списке пользователя, и используется в качестве номера параметра, значение которого сравнивается с результатом вычисления операнда E.

Если операнд D задает параметр, а операнд E не используется, значение параметра транзакта из списка пользователя сравнивается со значением такого же параметра выводящего транзакта. Если они равны, транзакт выводится из списка пользователя. И в этом случае количество выводимых транзактов определяется операндом C.

Операндом F указывается имя блока, куда переходит транзакт, выходящий из блока **UNLINK**, если из списка пользователя не выведен ни один транзакт. Если операнд F не используется, выходящий транзакт переходит в следующий блок независимо от количества выведенных транзактов. Например, блок **UNLINK 4, Apd,1** выводит из списка пользователя с номером 4 один транзакт с начала списка и направляет его в блок с меткой **Apd**. Блок **UNLINKOtst,Mars,1,BACK** выводит из списка пользователя с именем **Otst** один транзакт с конца списка и направляет его в блок с меткой **Mars**. Блок **UNLINK E P$Wiw,App1,ALL,App2,P$App2,App3** выводит из списка пользователя, номер которого записан в параметре **Wiw** выводящего транзакта, и направляет в блок с меткой **App1** все транзакты, содержимое параметра **App2** которых равно содержимому одноименного параметра выводящего транзакта. Если таких параметров в списке не окажется, выводящий транзакт будет направлен в блок с меткой **App3**, в противном случае – к следующему блоку.

Замечание. Отметим следующие особенности выполнения блока **UNLINK**. Во-первых, если операнды D и E содержат ссылки на СЧА транзактов, операнд D вычисляется относительно транзактов в списке пользователя, а операнд E – относительно активного транзакта. Во-вторых, после вывода транзактов из списка пользователя планировщик продолжает или начинает продвижение транзакта с наивысшим приоритетом, а при равенстве приоритетов отдает предпочтение транзакту – инициатору вывода, т. е. первым выводит этот транзакт.

5.4 Построение моделей систем с многоканальными устройствами и переключателями

Два или более обслуживающих устройств могут быть промоделированы на GPSS двумя или более ОКУ, располагаемыми рядом, т. е. параллельно. Так нужно поступать, когда отдельные устройства являются разнородными, т. е. характеризуются различными свойствами, например, различной интенсивностью обслуживания.

Однако часто различные параллельно работающие устройства являются однородными. GPSS представляет для моделирования однородных параллельных устройств специальное средство, именуемое многоканальным устройством. МКУ может быть использовано несколькими транзактами одновременно. Ограничений на число МКУ в модели нет. Для различия им дают имена.

МКУ определяется до его использования командой **STORAGE**. Формат команды следующий:

Name STORAGE A

Name – имя МКУ. Символическому имени может быть поставлен в соответствие номер командой **EQU**. Это необходимо, если требуется обращаться к нескольким МКУ в блоках **SELECT** и **COUNT** или в других случаях. Операнд A может быть только целым положительным числом. Иные способы задания емкости вызывают ошибку.

В модели можно организовать функционирование МКУ в двух режимах:

1) занятие и освобождение МКУ;

2) недоступность МКУ.

Занятие многоканального устройства и его освобождение

Занятие и освобождение МКУ имитируется блоками **ENTER** («Войти») и **LEAVE** («Выйти»). Форматы блоков следующие:

ENTER A,[B]

LEAVE A,[B]

Операнд А в обеих блоках используется для указания имени, соответствующего МКУ. Операнд В задает число устройств (элементов памяти), которое должно быть занято в блоке **ENTER** или освобождено в блоке **LEAVE**. По умолчанию операнд В = 1. При В = 0 блок считается неработоспособным.

Когда транзакт входит в блок **ENTER**, операнд А используется для нахождения МКУ с указанным именем. Если такого МКУ нет, происходит остановка по ошибке «Обращение к несуществующей памяти». Если МКУ существует и задан операнд В, он вычисляется, округляется до целого и полученный результат используется для оценки свободной емкости. Транзакт может войти в блок **ENTER**, если МКУ находится в доступном состоянии и достаточно емкости для выполнения запроса. В противном случае транзакт помещается в список задержки устройства в соответствии с приоритетом.

Когда транзакт входит в блок **ENTER** (рисунок 5.11), планировщик выполняет следующие действия:

1) увеличивает на единицу счетчик входов МКУ;

2) увеличивает на значение операнда В (по умолчанию на единицу) текущее содержимое МКУ;

3) уменьшает на значение операнда В (по умолчанию на единицу) доступную емкость МКУ.

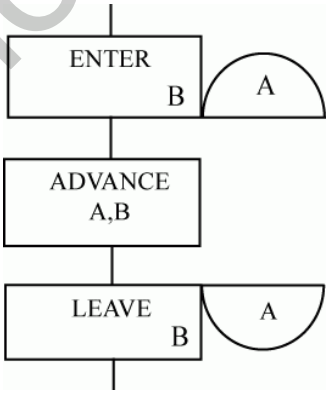


Рисунок 5.11 – Блок-диаграмма процедуры **ENTER – LEAVE**

Если транзакт при входе в блок **ENTER** запрашивает больше устройств (элементов памяти), чем определено командой **STORAGE**, т. е. ее операнд А меньше операнда В блока **ENTER**, возникает ошибка «Запрос элементов памяти превышает ее общую емкость».

МКУ никогда не может быть удалено из текущей модели, даже если команда **STORAGE** удаляется из рабочей программы. МКУ можно переопределить, т. е. изменить емкость другой командой **STORAGE** с тем же самым именем. Например:

```
Batr STORAGE        18
```

Повторное описание:

```
Batr STORAGE        24
```

Имитация обслуживания транзакта в течение какого-то промежутка времени также осуществляется блоком **ADVANCE**.

Пример 1

```
Nak  STORAGE        20
     . . .
      ENTER         Nak,2
      ADVANCE       120,40
      LEAVE         Nak,2
. . .
```

Командой **STORAGE** определяется МКУ с именем **Nak** емкостью 20 единиц. При входе транзакта в блок **ENTER** занимается 2 единицы и столько же освобождается в блоке **LEAVE** при выходе из МКУ.

Пример 2

```
Nak  STORAGE        20
      ENTER         Nak,21
      ADVANCE       120,40
      LEAVE         Nak,2
```

При входе транзакта в блок **ENTER** произойдет остановка по ошибке, т. к. транзакт будет пытаться занять больше каналов (21), чем определено (20) командой **STORAGE**. То же самое произойдет, если при выходе из блока **LEAVE** транзакт будет пытаться освободить каналов больше, чем определено командой **STORAGE**.

Пример 3

```
Pun1 EQU    1
Pun2 EQU    2
Pun3 EQU    3
Pun1 STORAGE        6
Pun2 STORAGE        5
Pun3 STORAGE        3
. . .
ENTER       *1
ADVANCE             MX$NorVr(P2,P3)
LEAVE       *1
. . .
```

В данном примере определены три МКУ с именами **Pun1**, **Pun2**, **Pun3** и емкостями 6, 5 и 3 соответственно. Именам командами **EQU** поставлены в соответствие номера 1, 2 и 3.

Предполагается, что при входе транзакта в блок **ENTER** в его первом параметре (ссылка *1) содержится какой-либо один из трех номеров. Согласно этому номеру и занимается МКУ, а затем освобождается. Операнд B в блоках **ENTER** и **LEAVE** не используется, поэтому транзактом занимается и освобождается одна единица емкости МКУ.

Ранее для МКУ был употреблен термин «память». Это связано с тем, что блоки **ENTER** и **LEAVE** могут быть использованы для имитации функционирования запоминающих устройств, например, оперативного или внешних устройств памяти. В этом случае в качестве операнда B этих блоков может быть использован один из параметров транзакта, содержащий количество занимаемых (освобождаемых) ячеек памяти, например, имитирующего сообщение.

Перевод многоканального устройства в недоступное состояние и восстановление его доступности

Недоступность МКУ моделируется блоком **SUNAVAIL** (символ **S** означает МКУ, **UNAVAIL** – недоступный). Формат блока имеет следующий вид:

SUNAVAIL A

Операнд A – имя или номер МКУ может быть именем, положительным целым числом, выражением в скобках, СЧА, СЧА<параметр>.

Например:

SUNAVAIL Batr

Когда транзакт входит в этот блок, МКУ **Batr** становится недоступным. Если при переводе в недоступное состояние в МКУ находились транзакты, т. е. текущее содержимое МКУ не равнялось нулю, то обслуживание этих транзактов продолжается, пока текущее содержимое не станет равным нулю.

Транзакты, которые пытаются занять МКУ во время нахождения его в недоступном состоянии, не входят в блок **ENTER** и помещаются в список задержки МКУ.

Нахождение в недоступном состоянии продолжается до тех пор, пока транзакт не войдет в блок **SAVAIL**.

Формат блока имеет следующий вид (рисунок 5.12).

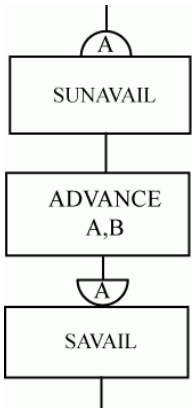


Рисунок 5.12 – Блок-диаграмма блока **SUNAVAIL**

Операнд A – имя или номер МКУ. Может быть таким же, как операнд A в блоке **SUNAVAIL**.

Если в момент перевода МКУ в доступное состояние в его списке задержки были транзакты, им предоставляется возможность занять МКУ в соответствии с дисциплиной «**first–fit–with–skip**» (первый подходящий с пропусками). Транзакты, которым будет отказано в занятии МКУ, остаются в его списке задержки.

Замечание. Имитация выхода МКУ из строя, при котором все транзакты, находившиеся в МКУ на обслуживании, теряются, блоками **SUNAVAIL** и **SAVAIL** невозможна. Такая имитация возможна с использованием блока **DISPLACE**.

Проверка состояния многоканального устройства

Состояние МКУ, как и состояние ОКУ, проверяется блоком **GATE** такого же формата:

GATE X A,[B]

Отличие состоит в значениях условного оператора X, которые могут быть следующими:

1) **SE –** МКУ, заданное операндом A, пусто;
2) **SF –** МКУ, заданное операндом A, заполнено;
3) **SNE –** МКУ, заданное операндом A, не пусто;
4) **SNF –** МКУ, заданное операндом A, не заполнено;
5) **SNV –** МКУ, заданное операндом A, не доступно;
6) **SV –** МКУ, заданное операндом A, доступно.

Блок **GATE** также работает в двух режимах:

1) отказа во входе;
2) разрешении во входе и альтернативном выходе.

Например:

GATE SNF Can,Met5

Если МКУ с именем **Can** не заполнено, т. е. имеются свободные каналы (элементы памяти), заданное в блоке **GATE** условие выполняется, и транзакт будет направлен к следующему блоку. Если МКУ заполнено, транзакт будет направлен к блоку с меткой **Met5**.

Блок **GATE** позволяет только определить состояние незаполненности МКУ, т. е. наличие свободных каналов, но достаточно ли их для удовлетворения запроса, он не определяет.

Для проверки состояния МКУ могут также использоваться булева переменная и блок **TEST**. Их использование позволяет расширить возможности по осуществлению проверок состояния, а также сократить машинное время, т. к. в одном блоке с помощью булевой переменной может проверяться сразу несколько условий.

Моделирование переключателей

Для моделирования такого оборудования, как переключатели, имеющие только два состояния, в GPSS используются логические ключи. Логический ключ может находиться в одном из двух состояний:

1) включен (**ON**, или 1);

2) выключен (**OFF**, или 0).

В зависимости от состояния ключа изменяется направление движения транзактов.

Логический ключ моделируется блоком **LOGIC** (рисунок 5.13). Формат блока имеет следующий вид:

LOGIC X A

Операнд А – имя или номер логического ключа. Может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА<параметр>.

Логический оператор X – состояние логического ключа, устанавливается в зависимости от следующих его значений:

1) **S** – ключ, заданный операндом А, включается;

2) **R** – ключ, заданный операндом А, выключается;

3) **I** – логический ключ инвертируется, т. е. состояние его меняется на противоположное, например, если был включен, будет выключен.

Состояние логического ключа проверяется также блоком **GATE**. Блок **GATE** имеет такой же формат, как и при проверке состояний ОКУ и МКУ, и два режима работы:

GATE X A,[B]

Операнд А – имя или номер проверяемого ключа. Может быть именем, положительным целым числом, выражением в скобках, СЧА или СЧА<параметр>.

Операнд В – метка блока, к которому будет направлен транзакт при невыполнении условия, заданного оператором X.

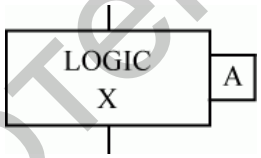


Рисунок 5.13 – Блок **LOGIC**

Условный оператор может принимать следующие значения:

1) **LS** – равен единице, если логический ключ, заданный операндом А, включен; нулю – если выключен;

2) **LR** – равен единице, если логический ключ, заданный операндом А, выключен; нулю – если включен.

Решение прямой и обратной задач в системе моделирования

При имитационном моделировании с использованием специальных инструментальных средств, например, GPSS World, в общем случае решаются две задачи. Назовем их прямой и обратной.

Прямая задача заключается в нахождении оценки математического ожидания какого-либо параметра моделируемой системы при заданном времени ее функционирования.

Обратная задача состоит в определении оценки математического ожидания времени функционирования моделируемой системы, за которое какой-либо ее показатель достигает заданного значения.

Решение этих задач, особенно обратной задачи, имеет свои особенности. Рассмотрим эти особенности на примере.

Постановка прямой и обратной задач

Рассмотрим следующий пример. Сервер обрабатывает запросы, поступающие с автоматизированных рабочих мест (АРМ) с интервалами, распределенными по показательному закону со средним значением T1 = 2 мин. Вычислительная сложность запросов распределена по нормальному закону с математическим ожиданием $S1 = 6 \cdot 10^7$ операций и среднеквадратичным отклонением $S2 = 2 \cdot 10^5$. Производительность сервера $Q = 6 \cdot 10^5$ операций/с. В случае занятости сервера поступающий запрос теряется.

Сервер представляет собой однофазную систему массового обслуживания разомкнутого типа с отказами.

Прямая задача. Построить имитационную модель для определения оценки математического ожидания количества запросов (далее – количества запросов), обработанных сервером за время функционирования T = 1 ч, и оценки математического ожидания вероятности обработки запросов (далее – вероятности обработки запросов).

Обратная задача. Построить имитационную модель для определения оценки математического ожидания времени (далее – времени обработки), за которое будет обработано сервером N запросов, и оценки математического ожидания вероятности обработки запросов.

Решение прямой задачи

Рассчитаем количество прогонов, которые нужно выполнить в каждом наблюдении, т. е. проведем так называемое тактическое планирование эксперимента. Пусть результаты моделирования (вероятность обработки запросов) нужно получить с доверительной вероятностью $\alpha = 0{,}95$ и точностью $\varepsilon = 0{,}01$. Расчет проведем для худшего случая, т. е. при вероятности $p = 0{,}5$, т. к. до эксперимента p неизвестно.

В модели для имитации источника запросов следует использовать блок **GENERATE**, для имитации сервера как одноканального устройства – блоки **SEIZE** и **RELEASE**, для имитации обработки запросов – блок **ADVANCE**.

В модели должны быть следующие элементы:

1) задание исходных данных;

2) описание арифметических выражений;

3) сегмент имитации поступления и обработки запросов;

4) сегмент задания времени моделирования и расчета результатов моделирования.

Серверу дадим имя **Server**. Для вывода из модели транзактов, имитирующих обработанные и потерянные запросы, используем блоки **TERMINATE** с

метками **ObrZap** и **PotZap** соответственно. Для счета количества всех запросов используем метку **KolZap**.

Блок-диаграмма модели

Построим блок-диаграмму модели для решения прямой задачи, т. е. сегмент имитации поступления и обработки запросов и сегмент задания времени моделирования и расчета результатов моделирования (рисунок 5.14).

Блок-диаграмма представляет собой набор стандартных блоков. Она строится следующим образом. Из множества блоков выбирают нужные и далее выстраивают их в диаграмму, для того чтобы в процессе функционирования модели они как бы взаимодействовали друг с другом. Диаграмма сопровождается необходимыми комментариями. Использование блоков при построении моделей зависит от логических схем работы реальных систем, моделируемых на ЭВМ.

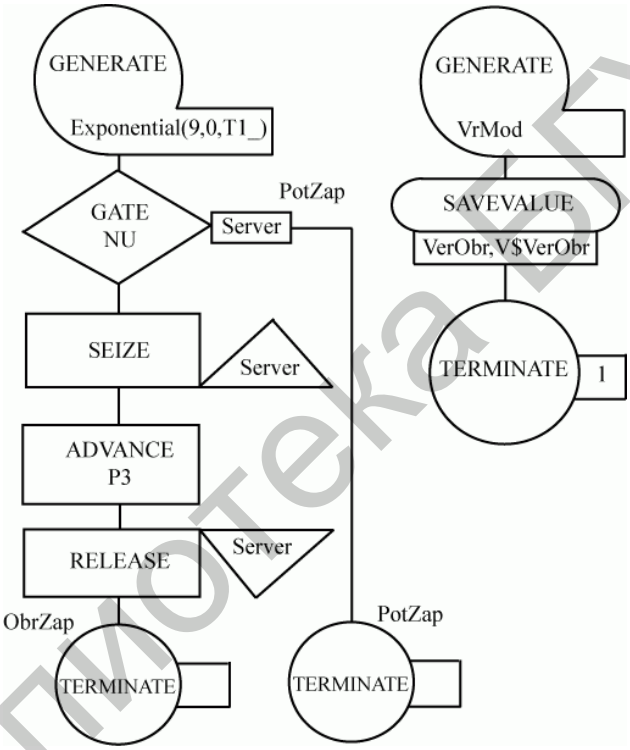


Рисунок 5.14 – Блок-диаграмма модели

Теперь приступим к написанию программы модели.

Программа модели

Для задания исходных данных используем переменные пользователя. Они задаются с помощью команды **EQU**. Переменным пользователя даны такие же имена, как и в постановке задачи, но добавлен знак подчеркивания. Например, **T1_**, **S1_** и т. д. Время моделирования зададим переменной пользователя **VrMod**.

Арифметическая переменная для расчета времени обработки **VrObr** запроса на сервере следующая:

VrObr VARIABLE (Normal(2,S1_#Koef,S2_#Koef))/Q_

Переменная пользователя **Koef** введена для удобства изменения (пропорционального изменения) характеристик нормального закона распределения, которому подчиняется вычислительная сложность запросов. Особенно целесообразно использование этой переменной при проведении экспериментов.

Вероятность обработки **VerObr** запросов на сервере будем определять как отношение количества обработанных **N$ObrZap** запросов к количеству всего поступивших **N$KolZap** запросов:

VerObr VARIABLE N$ObrZap/N$KolZap

В арифметическом выражении **VerObr**, например, **N$ObrZap** – системный числовой атрибут – количество транзактов, вошедших в блок с меткой **ObrZap**, а **N$KolZap** – количество транзактов, вошедших в блок с меткой **KolZap**.

Все необходимое для написания программы модели имеется. Напишем программу решения прямой задачи.

```
; Обработка запросов сервером. Прямая задача
; Задание исходных данных
T1_   EQU      120; Средний интервал поступления запросов, с
S1_   EQU      60000000; Среднее значение вычислительной сложности запросов, количество операций
S2_   EQU      200000; Стандартное отклонение вычислительной сложности запросов, количество операций
Q_    EQU      600000; Среднее значение производительности сервера, количество операций/с
Koef  EQU      1; Коэффициент изменения характеристик нормального распределения
VrObr VARIABLE (Normal(2,(S1_#Koef),(S2_#Koef))/Q_
VerObr VARIABLE N$ObrZap/N$KolZap
VrMod  EQU 3600; Время моделирования, 1 ед. мод. времени = 1 с
; Сегмент имитации обработки запросов
GENERATE   (Exponential(1,0,T1_)); Источник запросов
KolZap  GATE NU Server,PotZap; Свободен ли сервер? Если да, то
SEIZE   Server; занять сервер
ADVANCE    V$VrObr; Имитация обработки запроса
RELEASEServer; Освободить сервер ObrZap
TERMINATE; Обработанные запросы
PotZap  TERMINATE; Потерянные запросы
; Сегмент задания времени моделирования и расчета результатов
GENERATE   VrMod
TEST L  X$Prog,TG1,Met1; Если X$Prog < TG1,
SAVEVALUE  Prog,TG1; то X$Prog = TG1
Met1TEST E TG1,1,Met2; Если TG1 = 1, то
SAVEVALUE  VerObr, V$VerObr; расчет и сохранение в ячейке VerObr вероятности обработки запросов
```

SAVEVALUE Res,(INT(N$ObrZap/X$Prog)); *расчет и сохранение в ячейке Res количества обработанных запросов*

Met2 TERMINATE 1

START 1000,NP; *Прогоны до установившегося режима*

RESET; *Сброс накопленной статистики*

START 9604; *Количество прогонов модели*

При расчете количества обработанных запросов **SAVEVALUE Res,(INT(N$ObrZap/X$Prog))** в арифметическом выражении **N$ObrZap/X$Prog** используется число прогонов. В арифметическом выражении указано не явное число прогонов, а в виде содержимого ячейки **X$Prog**. Число прогонов заносится предварительно в эту ячейку по завершении первого прогона модели, но до того момента, когда из счетчика завершений TG1 будет вычтена первая единица. В этом случае арифметическое выражение не зависит от числа прогонов, которое может меняться на различных этапах создания и эксплуатации модели, в том числе и в зависимости от исходных данных, а также от точности и достоверности результатов моделирования. Поскольку количество обработанных запросов не может быть дробным числом, то для получения целого числа, записываемого в ячейку **Res**, используется процедура **INT** из встроенной библиотеки.

Для уменьшения машинного времени расчет искомых показателей производится не после каждого прогона, а после завершения последнего прогона, т. е. когда содержимое счетчика завершений будет равно единице (TG1 = 1).

Ввод текста программы модели, исправление ошибок и проведение моделирования

Запустите GPSS World.

Закройте окно напоминания о необходимости обновления «Заметок». Откроется главное меню.

Для ввода текста GPSS World имеет текстовый редактор. Откройте окно текстового редактора. Для этого выберите в меню **File/New** и в появившемся меню выберите **Model**. Нажмите **Ok**.

Наберите текст программы модели, т. е. создайте объект «Модель». При вводе текста следует использовать клавишу [Tab]. Например, после набора **T1_** нужно нажать клавишу [Tab]. Интервалы табуляции установлены по умолчанию.

После ввода текста программы модели создайте объект «Процесс моделирования», представляющий собой оттранслированный объект «Модель». Для трансляции объекта «Модель» выберите **Command/Create Simulation**. По этой команде транслятор GPSS проверяет текст программы модели на наличие синтаксических ошибок.

При наличии синтаксических ошибок транслятор в окне **JOURNAL** выдаст список сообщений об ошибках трансляции. При отсутствии ошибок в окне **JOURNAL** появится сообщение «Model Translation Begun. Ready».

Исправьте ошибки. Для поиска ошибок в тексте программы модели и их исправления используйте команду **Search/Next Error**. При первом выполнении

этой команды курсор мыши помещается в строке текста модели с ошибкой. После исправления первой ошибки вновь используйте команду **Search/Next Error** столько раз, сколько ошибок в тексте программы.

Сохраните модель. Для этого выберите в главном меню **File/Save As**. Дайте модели имя «Модель процессов изготовления изделий» и нажмите **Ok**.

Откликом в данной модели является вероятность обработки запросов сервером. Рассчитаем количество прогонов модели при среднеквадратичном отклонении для худшего случая при точности $\varepsilon = 0{,}01$ и доверительной вероятности $\alpha = 0{,}95$.

Запустите модель. Для этого в главном меню выберите **Command/Start** и в диалоговом окне вместо 1 наберите 9604. Нажмите **Ok**.

При наличии логических ошибок для их поиска в тексте программы модели и исправления используйте команду **Search/Goto Line** столько раз, сколько ошибок указано в окне **JOURNAL**. Там же (в окне **JOURNAL**) указаны номера строк с ошибками.

При отсутствии логических ошибок в модели по окончании ее работы система GPSS World автоматически создает стандартный отчет, который появится в окне **Report**. В окне будут содержаться следующие данные:

1) об именах объектов модели;
2) блоках модели;
3) ОКУ и МКУ;
4) очередях;
5) сохраняемых величинах.

В результате решения прямой задачи получим, что за один час сервером будет обработано 16 запросов, а вероятность обработки составит 0,546. Если не использовать процедуру **INT** – выделение целого числа с отбрасыванием дробной части – будет обработано 16,345 запроса.

Решение обратной задачи

Для решения обратной задачи возьмем количество запросов, ожидаемое время обработки которых нужно определить, N = 16 – результат решения прямой задачи.

Программа модели приведена ниже.

```
; Обработка запросов сервером. Обратная задача
; Задание исходных данных
T1_   EQU 120; Средний интервал поступления запросов, с
S1_   EQU 60000000; Среднее значение вычислительной сложности запросов, количество операций
S2_   EQU 200000; Стандартное отклонение вычислительной сложности запросов, количество операций
Q_    EQU 600000; Среднее значение производительности сервера, операций/с
Koef EQU 1; Коэффициент изменения характеристик нормального распределения
Koef1      EQU 1; Коэффициент учета дробной части
N_    EQU 16; Количество запросов
```

; Сегмент имитации обработки запросов

GENERATE (Exponential(1,0,T1_)); *Источник запросов*

KolZap GATE NU Server,PotZap; *Свободен ли сервер? Если да, то*

SEIZE Server; *занять сервер*

ADVANCE ((Normal(2,(S1_#Koef),(S2_#Koef)))/Q_); *Имитация обработки запроса*

RELEASE Server; *Освободить сервер*

TRANSFER ,ObrZap; *Запрос отправляется в сегмент завершения моделирования*

PotZap TERMINATE; *Потерянные запросы*

; Сегмент организации завершения моделирования и расчета результатов

**ObrZap TEST L X$Prog,TG1,Met1;** *Если X$Prog < TG1,*

SAVEVALUE Prog,TG1; *то X$Prog = TG1*

SAVEVALUE NZap,0; *Обнуление счетчика обработанных запросов*

Met1 SAVEVALUE NZap+,1; *Счет количества обработанных запросов*

TEST E X$NZap,N_,Ter1; *Если X$NZap = N_, то*

TEST E TG1,1,Met2; если *TG1 = 1, то*

SAVEVALUE VerObr,(N$ObrZap/N$KolZap); *расчет и сохранение в ячейке VerObr вероятности обработки запросов*

SAVEVALUE TimeNZap,((AC1–X$AC2)/(X$Prog#Koef1)); *сохранение в ячейке TimeNZap времени обработки запросов* **SAVEVALUE AC2,AC1;** *Запомнить абсолютное модельное время в ячейке AC2*

Met2 SAVEVALUE NZap,0; *Обнуление счетчика обработанных запросов*

TERMINATE 1

Ter1 TERMINATE

START 1000,NP; *Прогоны до установившегося режима*

RESET; *Сброс накопленной статистики*

START 9604; *Количество прогонов модели*

При решении обратной задачи один прогон определяется заданным количеством запросов N_, а не временем моделирования. Для этого организован счетчик обработанных запросов в виде сохраняемой ячейки **NZap**. Как только содержимое **X$NZap = N_**, из счетчика завершений вычитается единица.

Для расчета времени обработки заданного количества запросов используется арифметическое выражение **(AC1–X$AC2)/X$Prog**. В состав этого выражения входят абсолютное модельное время AC1 и количество прогонов. Запоминается количество прогонов, так же как и при решении прямой задачи.

Кроме этого, в арифметическом выражении есть сохраняемая ячейка **X$AC2**. Дело в том, что команда **RESET** не влияет на абсолютное модельное время AC1. Время же выполнения 1000 прогонов до установившегося режима не должно участвовать в расчете. Поэтому оно запоминается, а затем вычитается из абсолютного модельного времени выполнения 1000 + 9604 = 10 604 прогонов. Количество прогонов до установившегося режима может быть и другим.

В результате моделирования получим время обработки 16 запросов, равное 3523,658 с.

Фрагмент из отчета моделирования приведен ниже.

SAVEVALUE	RETRY	VALUE
PROG	0	9604.000
NZAP	0	0
VEROBR	0	0.546
TIMENZAP	0	3523.658

А почему не 3600 с? Ведь это же время моделирования было задано при решении прямой задачи. Потому что мы отбросили дробную часть, т. е. взяли 16, а не 16,345. Как же поступить, чтобы учесть и отброшенную дробную часть, ведь в счетчике фиксируются обработанные запросы только целыми числами, а не дробными?

Для учета десятых долей дробной части зададим N_ = 163, т. е. увеличим в десять раз. Это нужно учесть и в следующем арифметическом выражении:

((AC1–X$AC2)/(X$Prog#Koef1))

Переменной пользователя **Koef1** задается значение 10. По завершении моделирования получим 3586,504. Этот результат уже ближе к 3600.

Для учета сотых долей дробной части зададим N_ = 1634, а Koef1 = 100. Получим 3595,399 с.

Вероятность обработки запросов во всех случаях практически одна и та же, т. е. 0,546. Однако время моделирования существенно возрастает: 4 с, 39 с и 6 мин 29 с, т. е. в 10 и 100 раз соответственно.

В примере решения обратной задачи также показано, что арифметические выражения можно не описывать отдельно до блоковой части программы вместе с заданием исходных данных (как в модели прямой задачи), а сразу записывать в соответствующих блоках, заключив в скобки (скобки можно и не ставить, но лучше это делать).

Например:

ADVANCE ((Normal(2,(S1_#Koef),(S2_#Koef)))/Q_) ;
Розыгрыш времени обработки запроса

SAVEVALUE VerObr,(N$ObrZap/N$KolZap)); *Расчет вероятности обработки запросов*

SAVEVALUE TimeNZap,((AC1–X$AC2)/(X$Prog#Koef1)); *Расчет среднего времени обработки запросов*

5.5 Моделирование работы вычислительной системы в среде GPSS World

Рассмотрим следующую задачу и решение к ней. К компьютеру на обработку поступают задания. Из предварительного обследования получена информация, что интервал времени между двумя последовательными поступлениями заданий к компьютеру подчиняется равномерному закону распределения в интервале (1–11 мин). Перед компьютером допустима очередь заданий, длина которой не ограничена. Время выполнения задания также равномерно распределено в интервале (1–19 мин). Необходимо смоделировать обработку 100 заданий.

В среде GPSS программа, моделирующая работу вычислительной системы, выглядит следующим образом:

```
GENERATE 360,300
SEIZE B
ADVANCE 600,540
RELEASE B
TERMINATE 1
START 100
```

Единица модельного времени – 1 с.

Так как среднее время обработки задания больше, чем среднее время поступления задания, в вычислительной системе будет накапливаться очередь с течением времени. Для сбора статистики об очереди используются операторы **QUEUE**, **DEPART**. В этом случае программа выглядит следующим образом:

```
GENERATE 360,300
QUEUE BR
SEIZE B
DEPART BR
ADVANCE 600,540
RELEASE B
TERMINATE 1
START 100
```

Наберите эту программу в среде GPSS World.

Студенческая версия GPSS World не требует установки. Для запуска программы достаточно запустить на выполнение файл GPSSW.exe. После этого откроется среда моделирования GPSS World. Далее необходимо выбрать пункт меню **File/Open** и в открывшемся диалоговом окне «Новый документ» создать **Model**. В результате будет открыто окно **Untitled Model1**, в котором необходимо набрать текст программы.

Файл с программой можно сохранить в файле с расширением .gps (пункты меню **File/Save**, **File/Save As**).

Для запуска программы на выполнение необходимо выбрать пункт меню **Command/Create Simulation**.

Приведем основные операторы, которые используются в программе.

1. Для создания транзактов (заявок), входящих в модель, служит блок **GENERATE** («Генерировать»), имеющий следующий формат:

```
GENERATE  A,B,C,D,E
```

В поле A задается среднее значение интервала времени между моментами поступления в модель двух последовательных транзактов. Если этот интервал постоянен, то поле B не используется. Если же интервал поступления является случайной величиной, то в поле B указывается модификатор среднего значения, который может быть задан в виде модификатора-интервала или модификатора-функции.

Модификатор-интервал используется, когда интервал поступления транзактов является случайной величиной с равномерным законом распределения вероятностей. В этом случае в поле B может быть задан любой СЧА, кроме ссылки на функцию, а диапазон изменения интервала поступления имеет границы A – B, A + B.

Например, блок **GENERATE 100,40** создает транзакты через случайные интервалы времени, равномерно распределенные на отрезке [60;140].

Модификатор-функция используется, если закон распределения интервала поступления отличен от равномерного. В этом случае в поле B должна быть записана ссылка на функцию (ее СЧА), описывающую этот закон, и случайный интервал поступления определяется, как целая часть произведения поля A (среднего значения) на вычисленное значение функции.

В поле C задается момент поступления в модель первого транзакта. Если это поле пусто или равно нулю, то момент появления первого транзакта определяется операндами A и B.

Поле D задает общее число транзактов, которое должно быть создано блоком **GENERATE**. Если это поле пусто, то блок генерирует неограниченное число транзактов до завершения моделирования.

В поле E задается приоритет, присваиваемый генерируемым транзактам. Число уровней приоритетов не ограничено, причем самый низкий приоритет – нулевой. Если поле E пусто, то генерируемые транзакты имеют нулевой приоритет.

Транзакты имеют ряд стандартных числовых атрибутов. Например, СЧА с названием **PR** позволяет ссылаться на приоритет транзакта. СЧА с названием **M1** содержит так называемое резидентное время транзакта, т. е. время, прошедшее с момента входа транзакта в модель через блок **GENERATE**. СЧА с названием **XN1** содержит внутренний номер транзакта, который является уникальным и позволяет всегда отличить один транзакт от другого. В отличие от СЧА других объектов СЧА транзактов не содержат ссылки на имя или номер транзакта. Ссылка на СЧА транзакта всегда относится к активному транзакту, т. е. транзакту, обрабатываемому в данный момент симулятором.

2. Для удаления транзактов из модели служит блок **TERMINATE** («Завершить»), имеющий следующий формат:

TERMINATE A

Значение поля A указывает, на сколько единиц уменьшается содержимое так называемого счетчика завершений при входе транзакта в данный блок **TERMINATE**. Если поле A не определено, то оно считается равным нулю, и транзакты, проходящие через такой блок, не уменьшают содержимого счетчика завершений.

Начальное значение счетчика завершений устанавливается управляющим оператором **START** («Начать»), предназначенным для запуска прогона модели. Поле A этого оператора содержит начальное значение счетчика завершений. Прогон модели заканчивается, когда содержимое счетчика завершений обращается в нуль. Таким образом, в модели должен быть хотя бы один блок

TERMINATE с непустым полем A, иначе процесс моделирования никогда не завершится.

Текущее значение счетчика завершений доступно программисту через СЧА TG1.

Участок блок-схемы модели, связанный с парой блоков **GENERATE – TERMINATE**, называется сегментом. Простые модели могут состоять из одного сегмента, в сложных моделях может быть несколько сегментов.

Например, простейший сегмент модели, состоящий всего из двух блоков **GENERATE** и **TERMINATE** и в совокупности с управляющим оператором **START**, моделирует процесс создания случайного потока транзактов, поступающих в модель со средним интервалом в 100 единиц модельного времени, и уничтожения этих транзактов. Начальное значение счетчика завершений равно 1000. Каждый транзакт, проходящий через блок **TERMINATE**, вычитает из счетчика единицу, и таким образом моделирование завершится, когда тысячный по счету транзакт войдет в блок **TERMINATE**. При этом точное значение таймера в момент завершения прогона непредсказуемо. Следовательно, в приведенном примере продолжительность прогона устанавливается не по модельному времени, а по количеству транзактов, прошедших через модель:

```
GENERATE  100,40
TERMINATE  1
START   1000
```

Если необходимо управлять продолжительностью прогона по модельному времени, то в модели используется специальный сегмент, называемый сегментом таймера:

```
GENERATE  100,40
TERMINATE
GENERATE  100000
TERMINATE  1
START   1
```

Например, в модели из двух сегментов, первый (основной) сегмент выполняет те же функции, что и в предыдущем примере. Заметим, однако, что поле A блока **TERMINATE** в первом сегменте пусто, т. е. уничтожаемые транзакты не уменьшают содержимого счетчика завершений. Во втором сегменте блок **GENERATE** создаст первый транзакт в момент модельного времени, равный 100 000. Но этот транзакт окажется и последним в данном сегменте, т. к. войдя в блок **TERMINATE**, он обратит в нуль содержимое счетчика завершений, установленное оператором **START**, равным единице. Таким образом, в этой модели гарантируется завершение прогона в определенный момент модельного времени, а точное количество транзактов, прошедших через модель, непредсказуемо.

В приведенных примерах транзакты, входящие в модель через блок **GENERATE**, в тот же момент модельного времени уничтожались в блоке **TERMINATE**.

Замечание. Не путайте ограничитель транзактов в блоке **GENERATE** и счетчик завершения. Ограничитель задает число транзактов, которые войдут в модель, а счетчик – число транзактов, которые выйдут из модели. По окончании моделирования транзакты могут оставаться в модели.

5.6 Моделирование одноканальных устройств

Устройства используются при моделировании систем для имитации работы оборудования единичной емкости, например, процессора, канала передачи данных, человека, компьютера. Устройство в любой момент времени может обрабатывать только одно сообщение (транзакт). Если в процесс обслуживания появляется новый транзакт, то он должен выполнить одно из следующих действий:

– подождать своей очереди;

– направиться в другое место;

– прервать обслуживание текущего сообщения.

Для использования одноканального устройства транзакту необходимо выполнить следующие шаги:

1) ждать очереди, если необходимо;

2) занять устройство, когда подходит очередь;

3) находиться в состоянии занятости, пока не закончится обслуживание, т. к. для обслуживания необходим некоторый интервал времени;

4) освободить устройство, когда обслуживание закончится.

Второй и четвертый шаги реализуются блоками **SEIZE** и **RELEASE**.

Блок **SEIZE** имеет следующий формат:

SEIZE A

Свободный блок **SEIZE** позволяет вошедшему в него сообщению занять указанное устройство. Блок **SEIZE** задерживает сообщение, если устройство занято или находится в состоянии недоступности.

В поле А задается номер (имя) занимаемого устройства.

Сообщение, занявшее устройство, затем пытается перейти к следующему по номеру блоку. Устройство остается занятым до тех пор, пока занимающее его сообщение не войдет в соответствующий блок **RELEASE**. Прежде чем освободить устройство, сообщение может пройти через неограниченное число блоков.

Блок **RELEASE** имеет следующий формат:

RELEASE A

Блок **RELEASE** предназначен для освобождения устройства тем сообщением, которым в поле А задается номер (имя) освобождаемого устройства.

Транзакты обслуживаются устройствами в течение некоторого промежутка времени. Для моделирования такого обслуживания, т. е. для задержки транзактов на определенный отрезок модельного времени (реализация шага 3), служит блок **ADVANCE** («Задержать»), имеющий следующий формат:

ADVANCE A,B

Операнды в полях A и B имеют тот же смысл, что и в соответствующих полях блока **GENERATE**. Следует отметить, что транзакты, входящие в блок **ADVANCE**, переводятся из списка текущих событий в список будущих событий, а по истечении вычисленного времени задержки возвращаются назад, в список текущих событий, и их продвижение по блок-схеме продолжается. Если вычисленное время задержки равно нулю, то транзакт в тот же момент модельного времени переходит в следующий блок, оставаясь в списке текущих событий. Например, транзакты, поступающие в модель из блока **GENERATE** через случайные интервалы времени и имеющие равномерное распределение на отрезке [60;140], попадают в блок **SEIZE** и занимают устройство с номером 1. Далее в блоке **ADVANCE** определяется случайное время задержки транзакта, имеющее равномерное распределение на отрезке [30;130], и транзакт переводится в список будущих событий. По истечении времени задержки транзакт возвращается в список текущих событий и входит в блок **RELEASE** и освобождает устройство 1. Заметим, что в списке будущих событий, а значит, и в блоке **ADVANCE** может одновременно находиться произвольное количество транзактов.

```
GENERATE  100,40
SEIZE 1
ADVANCE   80,50
RELEASE 1
```

В рассмотренных выше примерах случайные интервалы времени подчинялись равномерному закону распределения вероятностей. Для получения случайных величин с другими распределениями в GPSS используются вычислительные объекты: переменные и функции.

Очереди. Блоки QUEUE и DEPART

В GPSS объекты типа «очередь» вводятся для сбора статистических данных.

Статистика об очередях собирается в моменты входа транзакта в блок **QUEUE** («Вход в очередь») или в блок **DEPART** («Выход из очереди»).

Формат записи блока **QUEUE** имеет следующий вид:

QUEUE A,[B]

Блок **QUEUE** увеличивает длину очереди.

В поле A задается номер или имя очереди, к длине которой добавляются единицы. Операнд может быть именем, положительным целым, СЧА.

Поле B определяет число единиц, на которое увеличивается текущая длина очереди. Если поле B пусто, то прибавляется единица.

Когда сообщение входит в блок **QUEUE**, то ищется очередь с именем, определенным операндом A. Если необходимо, очередь создается.

Поскольку к очереди добавляются единицы, а не сами сообщения, список членов очереди не составляется. Сообщения в этот же момент условного времени пытаются перейти к следующему блоку.

Так как очередь обычно используется для измерения времени ожидания, за блоком **QUEUE** обычно следует такой блок, как **SEIZE**, который может задержать сообщение.

Одно и то же сообщение может одновременно увеличить длину нескольких очередей, т. е. сообщение может войти в несколько блоков **QUEUE** перед тем, как войти в соответствующие блоки **DEPART**.

Значение текущей длины очереди хранится в СЧА **Q$<имя очереди>**.

Блок **DEPART** имеет следующий формат:

DEPART A,[B]

Блок **DEPART** служит для уменьшения длины очереди.

В поле А задается номер или имя очереди, длину которой нужно уменьшить. В поле В задается число единиц, на которое уменьшается длина очереди. Это число не должно превышать текущую длину очереди. Если поле В пусто, длина очереди уменьшается на единицу.

В результате выполнения программы моделирования работы вычислительной системы GPSS выдаст следующий отчет:

```
GPSS World Simulation Report – proba31.2.1
Wednesday, January 19, 2014 20:42:57
```

START TIME	END TIME	BLOCKS	FACILITIES	STOREGES
0.000	60243.977	7	1	0

NAME	VALUE
B	10001.000
BR	10000.000

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT	RETRY
1		GENERATE	166	0	0
2		QUEUE	166	65	0
3		SEIZE	101	1	0
4		DEPART	100	0	0
5		ADVANCE	100	0	0
6		RELEASE	100	0	0
7		TERMINATE	100	0	0

FACILITY	ENTRIES	UTIL.	AVE. TIME	AVAIL	OWNER	PEND	INTER	RETRY	DELAY
B	101	0.991	590	1	101	0	0	0	65

QUEUE	MAX	CONT	ENTRY	ENTRY (0)	AVE. CONT	AVE. TIME	AVE.(-0)	RETRY
BR	67	66	166	1	31.107	11289.054	11357.472	0

CEC XN	PRI	M1	ASSEM	CURRENT	NEXT PARAMETER VALUE
101	0	38246.575	101	3	4

FEC XN	PRI	BDT	ASSEM	CURRENT	NEXT PARAMETER VALUE
167	0	60773.872	167	0	1

Основные обозначения:

– **START TIME** – время начала моделирования;

– **END TIME** – время окончания моделирования;

– **BLOCKS** – количество блоков, используемых в программе;

– **FACILITIES** – количество устройств;

– **STORAGES** – количество многоканальных устройств, для которых определяется емкость накопителя.

Далее приводится информация о блоках:

– **LOC** – номер блока, назначенный системой;

– **BLOCK TYPE** – название блока;

– **ENTRY COUNT** – количество транзактов, прошедших через блок за время моделирования;

– **CURRENT COUNT** – количество транзактов, задержанных в блоке на момент конца моделирования;

– **RETRY** – количество транзактов, ожидающих специальных условий для прохождения через данный блок.

Отчет о работе устройства:

– **FACILITY** – название устройства;

– **ENTRIES** – количество транзактов, прошедших через устройство;

– **UTIL** – вероятность загрузки устройства (часть периода моделирования, когда устройство было свободно);

– **AVE. TIME** – среднее время обработки одного транзакта устройством;

– **AVAIL** – состояние готовности устройства на момент конца моделирования (1 – готово к обслуживанию очередной заявки, 0 – не готово);

– **OWNER** – номер последнего транзакта, занимающего устройство (если не занималось, то значение 0);

– **PEND** – количество транзактов, ожидающих устройство и находящихся в режиме прерывания;

– **INTER** – количество транзактов, прерывающих устройство в данный момент;

– **RETRY** – количество транзактов, ожидающих специальных условий и зависящих от состояния объекта типа «устройство»;

– **DELAY** – определяет количество транзактов, ожидающих занятия или освобождения устройства.

Статистика об очередях:

– **QUEUE** – имя очереди;

– **MAX** – максимальная длина очереди;

– **CONT** – текущая длина очереди;

– **ENTRY** – общее количество входов;

– **ENTRY(0)** – количество «нулевых» входов;

– **AVE.CONT** – средняя длина очереди;

– **AVE.TIME** – среднее время пребывания транзактов в очереди;

– **AVE.(–0)** – среднее время пребывания в очереди без учета «нулевых» входов;

– **RETRY** – количество транзактов, ожидающих специальных условий.

Информация о списке текущих событий CEC (Current Events Chain):

– **XN** – номер транзакта;

– **PRI** – приоритет транзакта (по умолчанию равен нулю);

– **M1** – время пребывания транзакта в системе с момента начала моделирования;

– **ASSEM** – номер семейства транзактов;

– **CURRENT** – номер блока, в котором находится транзакт;

– **NEXT** – номер блока, в который далее перейдет транзакт;

– **PARAMETER** – номер или имя параметра транзакта;

– **VALUE** – значение параметра.

Информация о списке будущих событий FEC (Future Events Chain):

– **XN** – номер транзакта;

– **PRI** – приоритет транзакта;

– **BDT** – таблица модельных событий – абсолютное модельное время выхода транзакта из списка будущих событий (и перехода транзакта в список текущих событий);

– **ASSEM** – номер семейства транзактов;

– **CURRENT** – номер блока, в котором находится транзакт (0 – если транзакт не вошел в модель);

– **NEXT** – номер блока, в который далее перейдет транзакт;

– **PARAMETER** – номер или имя параметра транзакта;

– **VALUE** – значение параметра.

Изменим условие задачи. Пусть в вычислительной системе два компьютера (интенсивность обработки заданий одинаковая), все остальные условия остаются без изменений.

В среде GPSS программа, моделирующая работу вычислительной системы, выглядит следующим образом:

```
NAK STORAGE 2
GENERATE 360,300
QUEUE BR
ENTER NAK
DEPART BR
ADVANCE 600,540
LEAVE NAK
TERMINATE 1
START 100
```

Обратите внимание, в программе появилась дополнительная строка **NAK STORAGE 2**, и блоки **SEIZE – RELEASE** заменены соответственно на блоки **ENTER – LEAVE**, моделирующие работу с многоканальным устройством.

Рассмотрим подробнее работу этих блоков.

5.7 Моделирование многоканальных устройств

Устройство в GPSS используют для моделирования одиночного устройства обслуживания. Два или более обслуживающих устройств, работающих параллельно, могут моделироваться двумя или более одинаковыми устройствами. Это необходимо, когда устройства являются разнородными.

Если параллельно работающие устройства являются одинаковыми, то для их моделирования может использоваться объект многоканального устройства (МКУ).

Количество устройств, которое моделирует МКУ, задает пользователь с помощью оператора **STORAGE**.

Формат оператора имеет следующий вид:

Метка **STORAGE A**,

где Метка – имя МКУ;

A – емкость МКУ (количество однотипных устройств, входящих в МКУ).

Блок **ENTER** имеет следующий формат записи:

ENTER A,[B]

Блок **ENTER** позволяет вошедшему сообщению (транзакту) использовать многоканальное устройство. Сообщение может быть задержано на входе в блок, если многоканальное устройство заполнено или имеющейся емкости недостаточно, или устройство в данный момент недоступно.

В поле A указывается номер или имя многоканального устройства, куда входит сообщение.

В поле B содержится число занимаемых единиц многоканального устройства. Если поле B пусто, то предполагается, что занимается одна единица. Если это значение равно нулю, то сообщение никогда не задерживается на входе, а блок рассматривается как нерабочий.

Активное сообщение не может войти в блок **ENTER**, если запрос на многоканальное устройство не может быть удовлетворен.

Активное сообщение не может войти в блок **ENTER**, если многоканальное устройство находится в недоступном состоянии.

Когда сообщение входит в блок **ENTER**, то операнд A используется для нахождения многоканального устройства с указанным именем. Если такое многоканальное устройство не существует, то возникает ошибка выполнения. В противном случае используется операнд B для оценки емкости многоканального устройства.

Одно и то же сообщение может входить в неограниченное число многоканальных устройств, а впоследствии освобождать их (или часть из них).

Блок **LEAVE** имеет следующий формат:

LEAVE A,[B]

Блок **LEAVE** освобождает определенное число единиц многоканального устройства. Занятый объем многоканального устройства уменьшается на число освобождаемых единиц. Оставшаяся емкость многоканального устройства увеличивается на ту же величину. Счетчик числа входов не изменяется.

Поле A блока **LEAVE** определяет номер или имя многоканального устройства, поле B – число освобождаемых единиц многоканального устройства. Если это поле пусто, освобождается одна единица. Число освобождаемых единиц не должно превышать текущее содержимое многоканального устройства.

В результате выполнения программы моделирования работы вычислительной системы с двумя компьютерами GPSS выдаст отчет с информацией об использовании МКУ:

– **STORAGE** – имя МКУ;

– **CAP** – емкость МКУ, заданная оператором **STORAGE**;

– **REM** – количество единиц свободной емкости в конце периода моделирования;

– **MIN.** – минимальное количество емкости за используемый период;

– **MAX.** – максимальное количество емкости за используемый период;

– **ENTRIES** – количество входов в МКУ за период моделирования;

– **AVL.** – состояние готовности МКУ в конце периода моделирования (1 – МКУ готов, 0 – нет);

– **AVE.C.** – среднее значение занятой емкости за период моделирования;

– **UTIL.** – средний коэффициент использования всех устройств МКУ;

– **RETRY** – количество транзактов, ожидающих специальных условий, зависящих от состояния МКУ;

– **DELAY** – определяет количество транзактов, ожидающих занятия или освобождения устройства МКУ.

5.8 Моделирование значений случайной величины с заданным законом распределения и обработка результатов моделирования средствами GPSS World

Моделирование последовательности значений случайных величин (СВ) с заданным законом распределения

Моделирование последовательности значений случайных величин (СВ) с заданным законом распределения реализуется на основе использования случайных величин, имеющих равномерное распределение в интервале [0;1].

Для получения случайной величины Y с равномерным распределением на интервале [0;1] в GPSS имеются встроенные генераторы случайных чисел. Для получения случайного числа путем обращения к такому генератору достаточно записать **RN** с номером генератора, например **RN1**. Правда, встроенные генераторы случайных чисел GPSS World дают числа не на интервале [0;1], а целые случайные числа, равномерно распределенные от 0 до 999, но их нетрудно привести к указанному отрезку делением на 1000.

При каждом запуске системы генераторы выдают одну и ту же последовательность чисел. Команда **PMULT** позволяет изменить эту последовательность путем изменения начальных множителей.

Формат команды имеет следующий вид:

PMULT A,B,C,D,E,F,G

Операнды A, B, C, D, E, F, G задают соответственно начальные множители для 1–7 генераторов случайных чисел.

Например, **PMULT 890,5** устанавливает начальные состояния множителей генераторов случайных чисел 1 и 4.

Вычисления в GPSS выполняются с использованием переменных. Они могут быть целыми и действительными (с плавающей точкой). Действительные переменные определяются перед началом моделирования с помощью оператора определения **FVARIABLE** («Переменная»), имеющего следующий формат:

Имя **FVARIABLE** выражение

Здесь имя – имя переменной, используемое для ссылок на нее, а выражение – арифметическое выражение, определяющее переменную.

Арифметическое выражение представляет собой комбинацию операндов, в качестве которых могут выступать константы, функции, знаки арифметических операций и круглых скобок. Следует заметить, что знаком операции умножения в GPSS является символ «#».

Переменные могут быть использованы для получения значений случайной величины с заданным законом распределения. Пусть необходимо сгенерировать 100 значений равномерно распределенной СВ на интервале [5;15].

```
TARR      FVARIABLE    10#(RN1/1000)+5
GENERATE  V$TARR
TERMINATE 1
START 100
```

Для получения значений случайной величины Y с показательным (экспоненциальным) законом необходимо воспользоваться соотношением $y = -\frac{1}{\lambda}\ln x_i$, полученным на основе метода обратной функции. Для $\lambda = 1$ имеем:

```
TARR1     FVARIABLE    (–1)#LOG(RN1/1000)
GENERATE  V$TARR1
TERMINATE 1
START 100
```

Такой способ является достаточно трудоемким, т. к. требует обращения к математическим функциям, вычисление которых требует десятков машинных операций. Другим возможным способом является использование вычислительных объектов GPSS типа функция.

Функции используются для вычисления величин, заданных табличными зависимостями. Каждая функция определяется перед началом моделирования с помощью оператора определения **FUNCTION** («Функция»), имеющего следующий формат:

Имя FUNCTION A,B

Здесь имя – имя функции, используемое для ссылок на нее; A – стандартный числовой атрибут, являющийся аргументом функции; B – тип функции и число точек таблицы, определяющей функцию.

Так, например, тип непрерывных числовых функций кодируется буквой С. При этом в определении непрерывной числовой функции, таблица которой содержит 24 точки, поле В должно иметь значение **С24**.

При использовании непрерывной функции для генерирования случайных чисел ее аргументом должен быть один из генераторов случайных чисел **RNj**. Так, оператор для определения функции показательного распределения может иметь следующий вид:

EXP FUNCTION RN1,C24

Особенностью использования встроенных генераторов случайных чисел **RNj** в качестве аргументов функций является то, что их значения в этом контексте интерпретируются как дробные числа от 0 до 0,999999.

Таблица с координатами точек функции располагается в строках, следующих непосредственно за оператором **FUNCTION**. Эти строки не должны иметь поля нумерации. Каждая точка таблицы задается парой **Xi** (значение аргумента) и **Yi** (значение функции), отделяемых друг от друга запятой. Пары координат отделяются друг от друга символом «/» и располагаются на произвольном количестве строк. Последовательность значений аргумента **Xi** должна быть строго возрастающей.

Как уже отмечалось, при использовании функции в поле В блоков **GENERATE** и **ADVANCE** вычисление интервала поступления или времени задержки производится путем умножения операнда А на вычисленное значение функции. Отсюда следует, что функция, используемая для генерирования случайных чисел с показательным (экспоненциальным) распределением $\lambda = 1$, должна описывать зависимость $y = -\ln(1 - x)$, представленную в табличном виде.

Оператор **FUNCTION** с такой таблицей, содержащей 24 точки для обеспечения достаточной точности аппроксимации, имеет следующий вид:

EXP FUNCTION RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8

Вычисление непрерывной функции производится следующим образом.

Сначала определяется интервал (Xi;Xi+1), на котором находится текущее значение СЧА-аргумента (в нашем примере – сгенерированное значение **RN1**). Затем на этом интервале выполняется линейная интерполяция с использованием соответствующих значений Yi и Yi+1. Результат интерполяции используется в качестве значения функции. Если функция служит операндом В блоков **GENERATE** или **ADVANCE**, то результат умножается на значение операнда А.

Использование функций для получения случайных чисел с заданным распределением дает хотя и менее точный результат за счет погрешностей аппроксимации, но зато с меньшими вычислительными затратами (несколько машин-

ных операций на выполнение линейной интерполяции). По сути этот вариант реализации (получения последовательности значений СВ) соответствует методу кусочной аппроксимации функции плотности распределения вероятностей.

Функции всех типов имеют единственный системный числовой атрибут с названием **FN**, значением которого является вычисленное значение функции. Вычисление функции производится при входе транзакта в блок, содержащий ссылку на СЧА **FN** с именем функции.

При большом количестве транзактов, пропускаемых через модель (десятки и сотни тысяч), разница в скорости вычислений по данному способу и способу, описанному выше, должна стать заметной.

```
EXP1    FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
GENERATE    100,FN$EXP1
TERMINATE 1
START 100
```

В примере среднее значение СВ равно 100.

В примере табличного задания нормального распределения СВ используется 25 точек для обеспечения достаточной точности аппроксимации.

```
NOR1 FUNCTION RN1, C25
0,–5/.00003,–4/.00135,–3/.00621,–2.5/.02275,–2
.06681,–1.5/.11507,–1.2/.15866,–1/.21186,–.8/.27425,–.6
.34458,–.4/.42074,–.2/.5,0/.57926,.2/.65542,.4
.72575,.6/.78814,.8/.84134,1/.88493,1.2/.93319,1.5
.97725,2/.99379,2.5/.99865,3/.99997,4/1,5
```

Данная таблица задает СВ Z с математическим ожиданием, равным нулю, и среднеквадратичным отклонением (СКО), равным единице. Для моделирования нормальной СВ X с другими значениями математического ожидания и СКО необходимо произвести вычисления по формуле

$$X = m_x + \sigma_x Z.$$

Если математическое ожидание $m_X = 5$, среднеквадратичное отклонение $\sigma = 2$, то

```
GENERATE  (5+2#FN$NOR1)
TERMINATE 1
START 100
```

Обработка результатов моделирования

Для получения оценок математического ожидания (среднего значения) и дисперсии последовательности значений СВ, полученных в GPSS, необходимо использовать блоки сбора статистики **TABLE** и **TABULATE**.

Оператор описания таблицы **TABLE** имеет следующий формат:

<NAME> TABLE <A>,<B>,<C>,<D>

Оператор определяет аргумент, а также число и ширину частотных интервалов.

Метка **NAME** определяет имя таблицы.

В поле A задается аргумент таблицы – элемент данных, чье частотное распределение будет табулироваться. Операнд может быть именем, целым, СЧА или СЧА<параметр>.

В поле B задается верхний предел первого интервала. Операнд может быть целым или именем.

В поле C задается ширина частотного интервала – разница между верхней и нижней границей каждого частотного класса. Операнд может быть положительным целым.

В поле D задается число частотных интервалов. Это число не может превышать 8191. Операнд может быть положительным целым.

Для сбора элементов данных сообщение должно войти в блок **TABULATE** с тем же именем таблицы, что определено в блоке **TABLE**. Когда сообщение входит в блок **TABULATE**, оценивается аргумент таблицы (операнд A в операторе **TABLE**). Если он меньше или равен операнду B в операторе **TABLE**, то выбирается первый частотный класс таблицы. Если аргумент таблицы не подходит для этого класса, то класс выбирается путем деления значения аргумента на операнд C оператора **TABLE**. Нижняя граница частотного класса включается в предыдущий класс. Если таблицы недостаточно для размещения этого значения, то выбирается последний частотный интервал. Затем выбирается целое из частотного класса и счетчик увеличивается на величину, определяемую операндом B оператора **TABULATE**. По умолчанию увеличение происходит на единицу. В конце работы оператора **TABULATE** изменяются значения среднего и стандартного отклонения аргумента таблицы.

Таблица может быть переопределена или переинициализирована другим оператором **TABLE**, с той же самой меткой, что и первая.

Стандартные числовые атрибуты, связанные с описываемым оператором, следующие:

– **TB** – среднее значение аргумента;

– **TC** – число входов в таблицу;

– **TD** – стандартное отклонение.

Блок **TABULATE** имеет следующий формат:

TABULATE <A>,[<B>]

Блок **TABULATE** табулирует текущее значение заданного аргумента. Способ табуляции зависит от режима работы таблицы, который определяется оператором описания таблицы.

В поле задается номер или имя таблицы, в которую табулируется значение аргумента. Таблица должна быть определена оператором описания.

В поле задается число единиц, которые должны быть занесены в тот частотный интервал, куда попало значение аргумента. Если поле B пусто, эта величина полагается равной единице.

Когда сообщение входит в блок **TABULATE**, то для нахождения таблицы используется операнд A. Если такой таблицы нет, то возникает ошибка выполнения. Таблица должна быть определена оператором **TABLE**. Таблица изменяется в соответствии с операндами оператора **TABLE**.

Рассмотрим пример использования блоков **TABLE** и **TABULATE**.

```
TT TABLE M1,40,50,8
EXP1   FUNCTION   RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915
.7,1.2/.75,1.38/.8,1.6/.84,1.85/.88,2.12/.9,2.3
.92,2.52/.94,2.81/.95,2.99/.96,3.2/.97,3.5/.98,3.9
.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8
GENERATE   100,FN$EXP1
ADVANCE 100,FN$EXP1
TABULATE TT
TERMINATE 1
START 100
```

Здесь **M1** – константа (стандартный числовой атрибут), которая связана с каждым транзактом и хранит время пребывания транзакта в модели. Время пребывания транзакта в модели определяется блоком **ADVANCE** и распределено по показательному закону с $\lambda = 1/100$. Строится частотное распределение, вычисляются оценки математического ожидания и среднеквадратичного отклонения для **M1**, т. е. для времени пребывания транзакта в модели. Граница первого интервала задана 40; ширина интервала группирования – 50; число интервалов группирования – 8. Все эти параметры задаются опытным путем.

Ниже приведен фрагмент отчета, выдаваемый GPSS по результатам работы программы.

TABLE	MEAN	STD.DEV.	RANGE	RETRY	FREQUENCY	CUM.%
TT	101.032	101.889	0	40.000	38	38.00
			40.000	90.000	23	61.00
			90.000	140.000	9	70.0
			140.000	190.000	8	78.00
			190.000	240.000	10	8.00
			240.000	290.000	6	94.00
			290.000	340.000	5	99.00
			340.000		1	100.0

Основные обозначения, использованные в отчете:

- **Mean** – это среднее значение или оценка математического ожидания;
- **STD.DEV** – это оценка среднеквадратичного отклонения;
- **Range** – интервалы группирования;
- **FREQUENCY** – количество наблюдений, попавших в каждый интервал.

Таким образом, погрешность в оценке математического ожидания составила:

$$\varepsilon^{m} = M(Y) - \hat{M}(Y) = 100 - 101.032 = -1.032.$$

Погрешность в оценке среднеквадратичного отклонения составила:

$$\varepsilon^{\sigma} = \sigma(Y) - \hat{\sigma}(Y) = 100 - 101.889 = -1.889.$$

Таким образом, точность имитационного моделирования значений СВ по методу кусочной аппроксимации функции плотности распределения вероятностей в среде GPSS достаточно высокая.

Для построения гистограммы необходимо выбрать после выполнения программы пункт меню **Window/Simulation Window/Table Window**. Далее в открывшемся диалоговом окне задать имя таблицы (в данном примере TT).

Визуальный анализ гистограммы позволяет сделать вывод о согласии выборочных значений СВ Y с моделью экспоненциального распределения.

5.9 Имитационная модель функционирования узла коммутации

Рассмотрим имитационную модель функционирования АТС. Исследуем зависимость вероятности разговоров абонентами TA1 от интервалов времени T1, T2, времени t1, t2 разговоров и вероятностей P5 и P10 ответа на звонки абонентов TA2 и TA1 соответственно.

Результаты моделирования необходимо получить с точностью ε = 0,01 и доверительной вероятностью (достоверностью) α = 0,99.

Модель функционирования АТС должна иметь:

– задание исходных данных;

– сегмент имитации телефонных разговоров с TA1;

– сегмент имитации телефонных разговоров с TA2;

– задание времени моделирования и расчета результатов моделирования.

Телефоны и внешние выходы в модели нужно представить ОКУ, а заявки на разговоры – транзактами. Казалось бы, что в целях сокращения программы модели нужно использовать МКУ, однако МКУ не дает возможности идентифицировать каждый свой канал. Что касается сокращения, то этого можно достичь и с ОКУ.

Присвоим, используя переменные пользователя, номера ОКУ, предварительно не указывая их имена:

– TA1 – 1…N1;

– TA2 – (N1_+1)…(N1_+N2_);

– внешние выходы – (N1_+N2_+1)…(N1_+N2_+N3_).

Нумерацию будем использовать в блок-диаграмме и программе модели. За счет этого при увеличении числа TA1, TA2 и внешних выходов программа модели не потребует внесения каких-либо изменений.

Конечно, можно также определить арифметические выражения для вычисления (N1_+N2_), (N1_+N2_+N3_) и затем ссылаться на них в программе модели, например:

```
Num1        VARIABLE   N1_+N2_
Num2        VARIABLE   N1_+N2_+N3_
```

В больших моделях этот вариант предпочтителен. Для лучшего понимания построения модели ее сегменты разделены на части, реализующие самостоятельные функции. Сегмент имитации телефонных разговоров с ТА1 содержит следующие задачи:

1) определение номера звонящего ТА1;
2) поиск свободного внешнего выхода;
3) поиск внешнего выхода, занятого ТА2;
4) прерывание разговора с ТА2;
5) имитация ведения разговора с ТА1 без прерывания разговора с ТА2;
6) определение номера ТА2, на который звонят с ТА1;
7) имитация разговора между абонентами ТА1 и ТА2.

Сегмент имитации телефонных разговоров абонентами ТА2 содержит следующие задачи:

1) определение номера звонящего ТА2;
2) поиск свободного внешнего выхода;
3) имитация ведения разговора с ТА2;
4) определение номера ТА1, на который звонят с ТА2;
5) имитация разговора абонентов ТА2 и ТА1.

В программе демонстрируется возможность GPSS описания и использования в одной и той же модели ОКУ, функционирующего в различных режимах:

– занятия и освобождения устройства (режим FIFO);
– прерывания работы устройства.

Для имитации прерывания телефонных разговоров абонентов ТА2 звонками абонентов с ТА1 используются блоки **PREEMPT** и **RETURN**.

6 МОДЕЛИРОВАНИЕ РАБОТЫ АТС

6.1 Блок-диаграмма модели

Блок-диаграмма модели представлена ниже (рисунок 6.1).

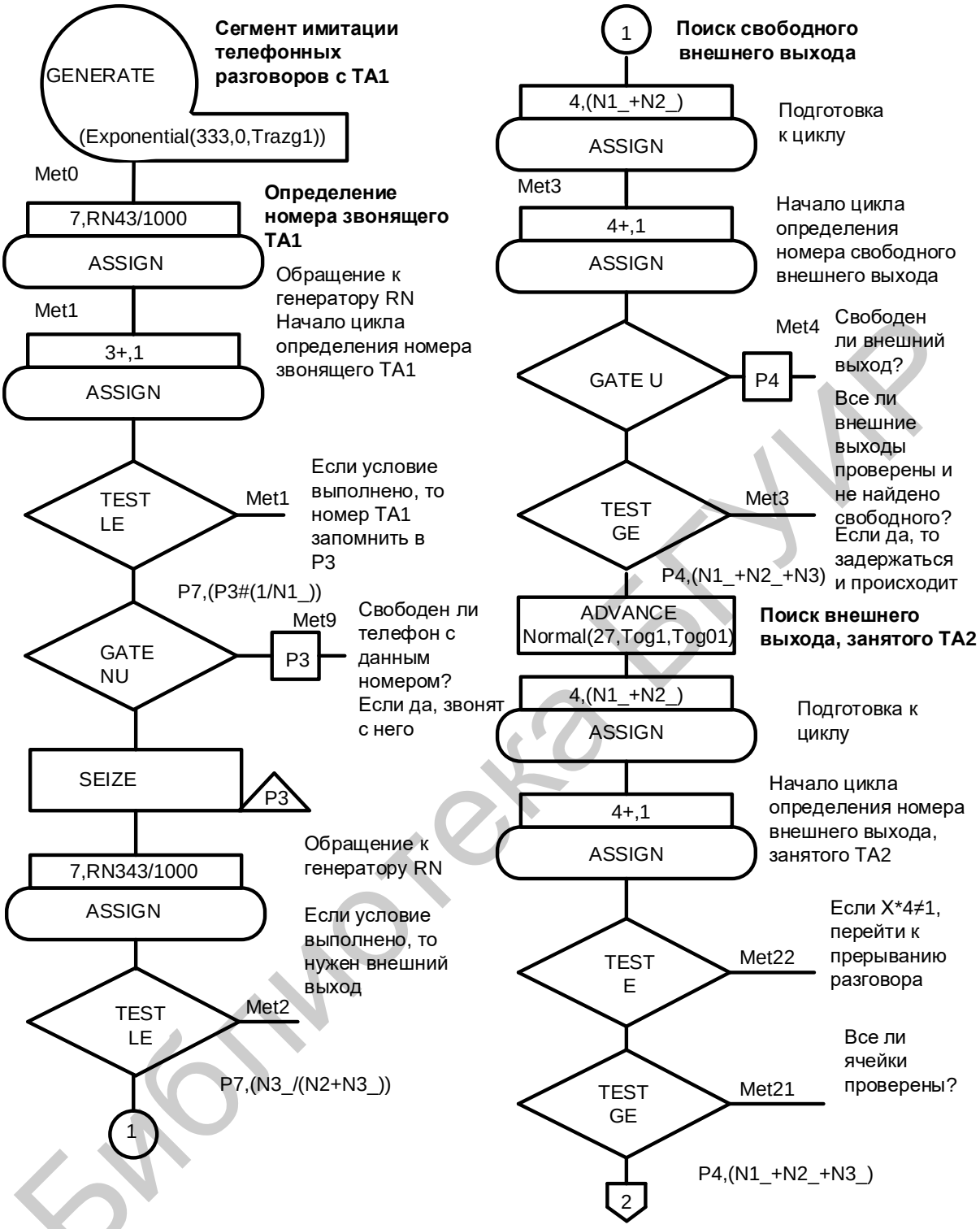


Рисунок 6.1 – Блок-диаграмма модели АТС (1-й фрагмент; продолжение и окончание см. на с. 76, 77, 78 и 79)

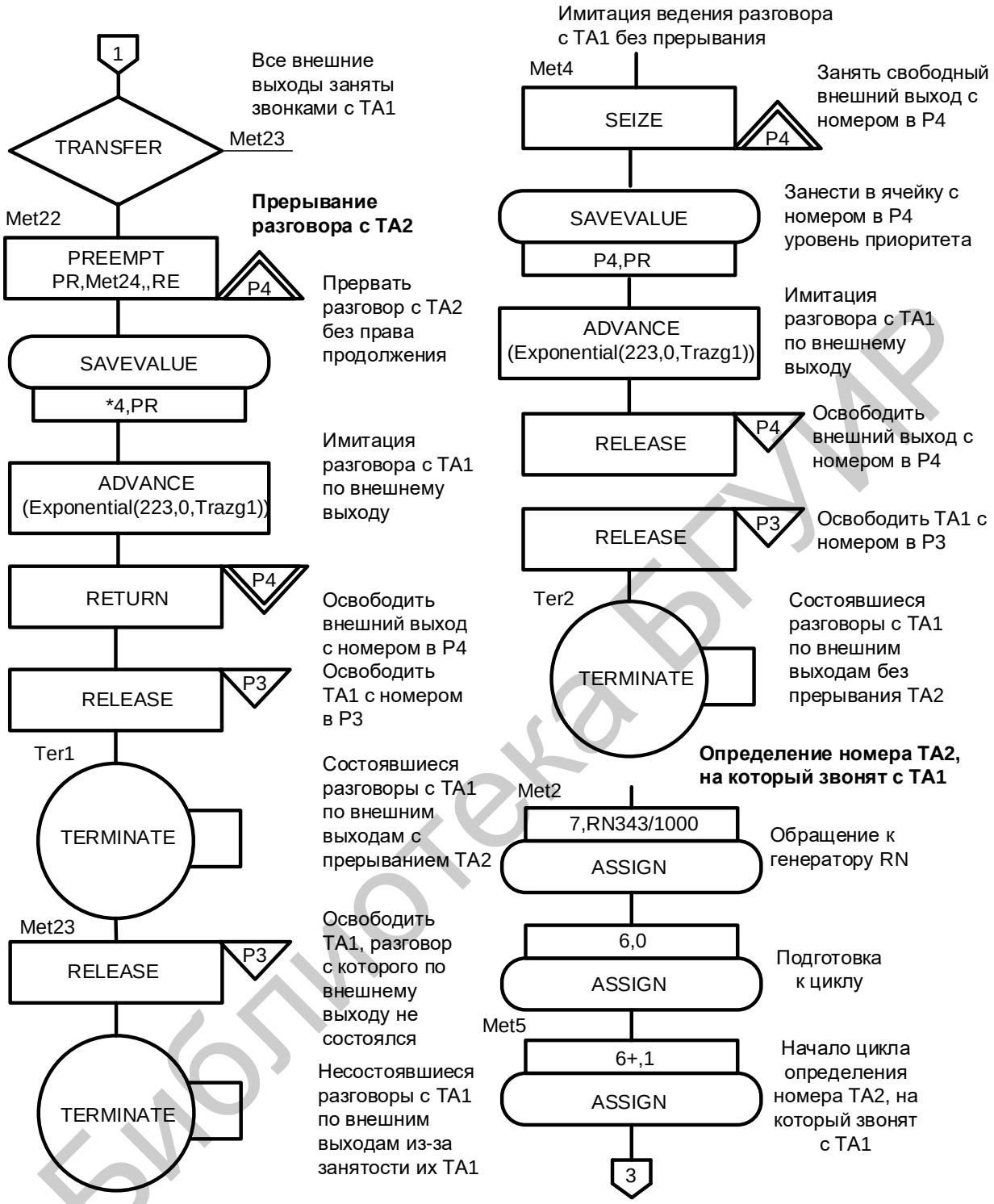


Рисунок 6.1 – Продолжение (начало см. на с. 75, продолжение и окончание – на с. 77, 78 и 79)

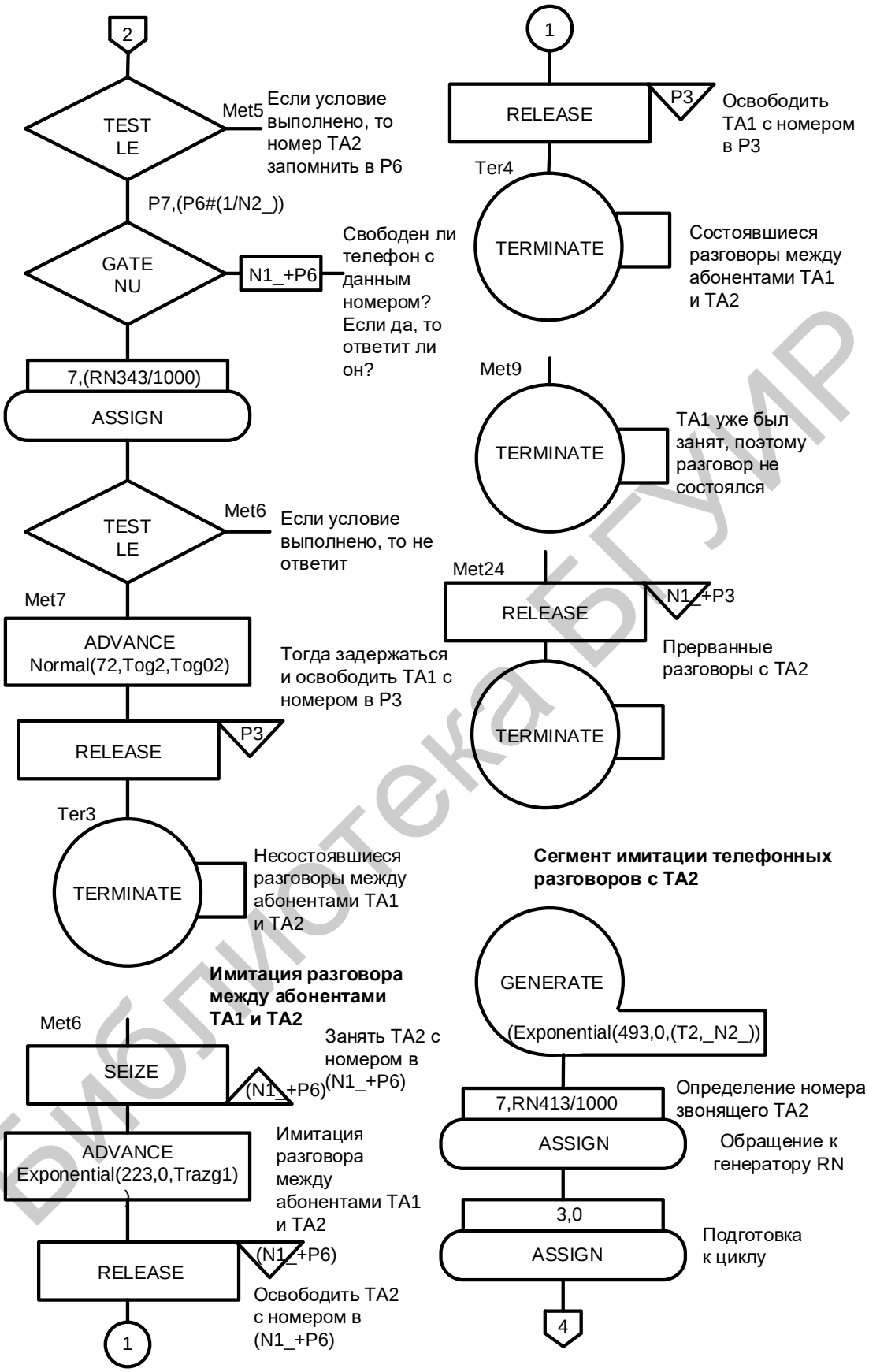


Рисунок 6.1 – Продолжение (начало см. на с. 75, продолжение и окончание – на с. 76, 78, 79)

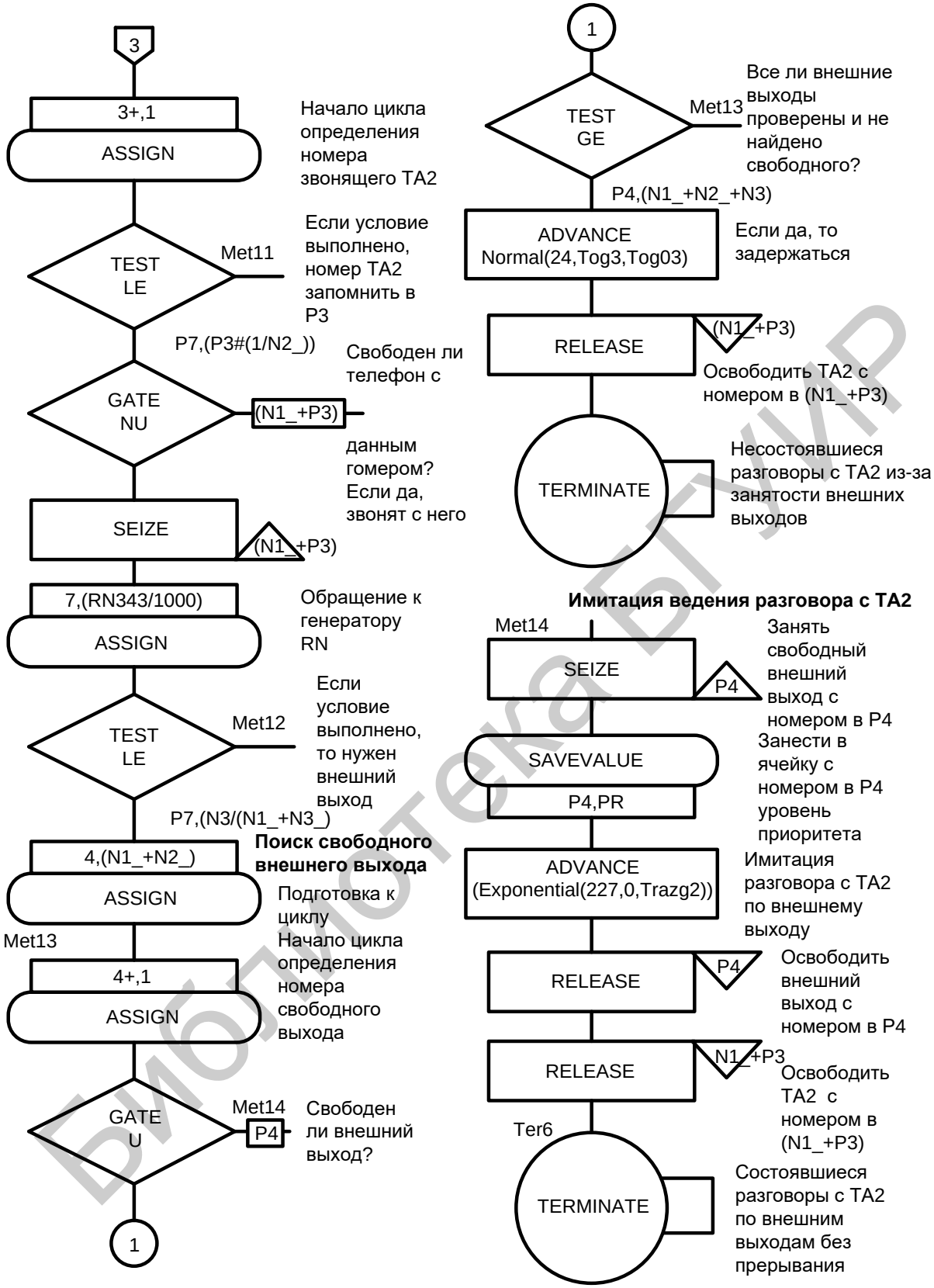


Рисунок 6.1 – Продолжение (начало см. на с. 75, продолжение и окончание – на с. 76, 77, 79)

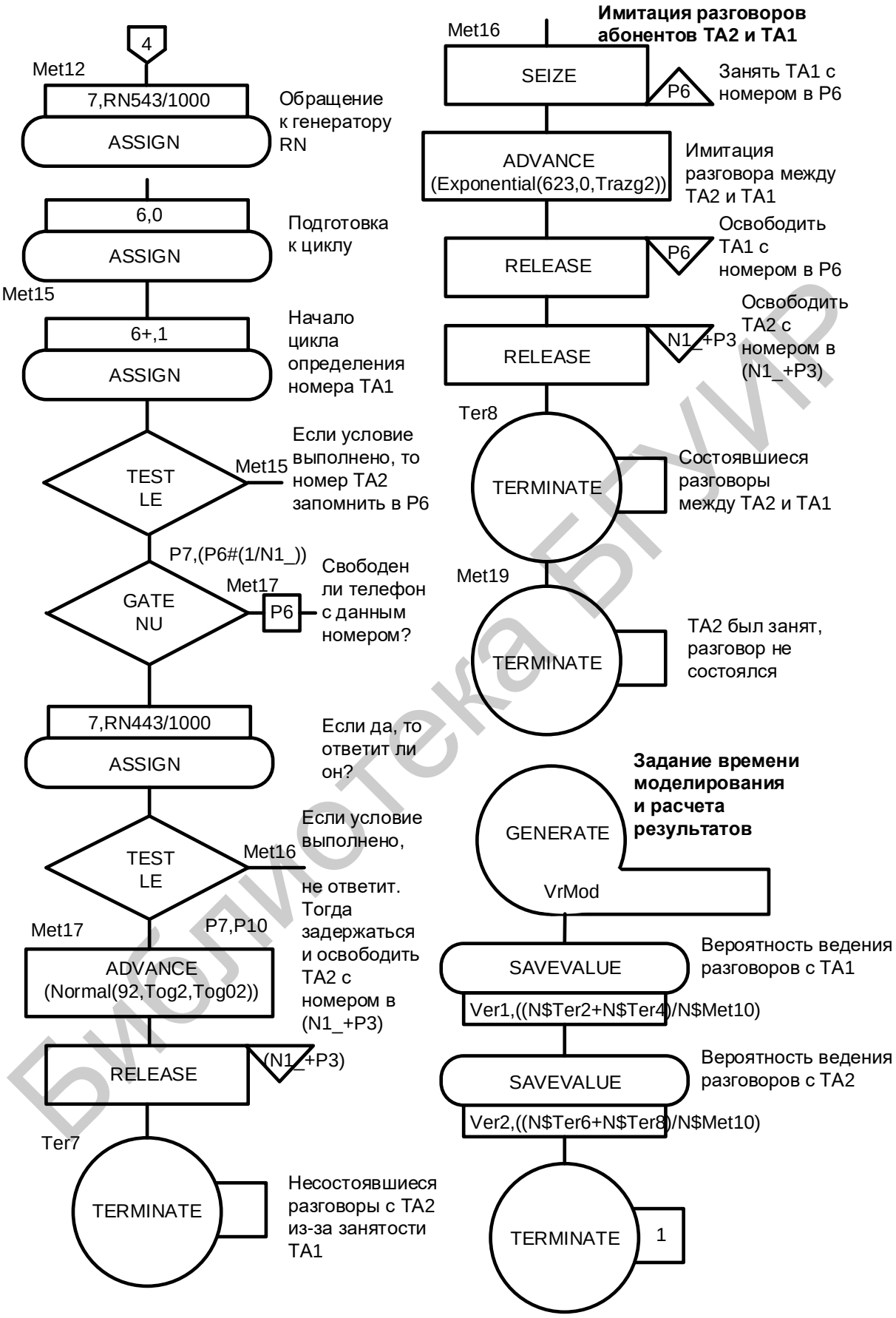


Рисунок 6.1 – Окончание (начало см. на с. 75–78)

6.2 Программа модели

В программе модели для задания исходных данных используются переменные пользователя. Значения им присваиваются командой EQU. Это позволяет проводить эксперименты с использованием встроенных генераторов, т. к. факторы в них должны быть только переменными пользователя.

Рассмотрим работу сегмента имитации телефонных разговоров с ТА1.

Генератор этого сегмента вводит в модель транзакты со средним интервалом времени (**T1_/N1_**). Поскольку звонки с ТА1 имеют абсолютный приоритет, то блоком **GENERATE** транзактам присваивается приоритет 1. Генератор сегмента имитации телефонных разговоров с ТА2 вводит в модель транзакты с нулевым приоритетом.

После ввода транзакта в модель, имитирующего заявку на звонок, производится определение номера звонящего ТА1. Группа событий – звонки с ТА1 – рассматривается как полная группа несовместных событий. В модели реализован алгоритм розыгрыша в цикле, что делает его независимым при изменении количества телефонных аппаратов.

Определенный в результате розыгрыша номер ТА1 заносится в параметр **P3** транзакта. Однако возможно, что с этого телефона уже звонят. Поэтому блоком **GATE** производится проверка: свободен ли ТА1 с данным номером. Если нет, заявка теряется. Если телефон свободен, то далее определяется, куда идет звонок: на ТА2 или требуется внешний выход. Ответ на этот вопрос находится розыгрышем единичного события.

Пусть требуется внешний выход. Поиск свободного внешнего выхода производится в цикле от номера (**N1_ + N2_ + 1**) до номера (**N1_ + N2_ + N3_**). Предположим, что найден свободный внешний выход, номер которого занесен в параметр **P4**. Тогда последовательностью блоков, начиная с метки **Мет4**, имитируется ведение разговора с ТА1 без прерывания разговора по внешнему выходу с ТА2. После занятия свободного выхода в сохраняемую ячейку с тем же номером, что и номер занятого внешнего выхода, заносится уровень приоритета, в данном случае это единица. Разговор с ТА1 по внешнему выходу состоялся. Транзакт выводится из модели блоком **TERMINATE** с меткой **Ter2**.

Вернемся к поиску свободного внешнего выхода. Пусть все внешние выходы заняты. Далее отыскивается внешний выход, занятый разговором с ТА2. Поиск производится также в цикле от номера (**N1_ + N2_ + 1**) до номера (**N1_ + N2_ + N3_**). Однако при этом проверяется содержимое сохраняемых ячеек с такими же номерами, как и номера внешних выходов. Если содержимое проверяемой ячейки равно единице, то внешний выход занят ТА1. Если все внешние выходы заняты ТА1, т. е. во всех проверяемых сохраняемых ячейках единица, заявка на звонок теряется и транзакт отправляется на метку **Мет23**.

Пусть найден внешний выход, занятый ТА2, т. е. в соответствующей сохраняемой ячейке нуль. Транзакт переходит к последовательности блоков, начи-

нающихся с метки **Мет22**. Эта последовательность имитирует прерывание разговора с ТА2 и ведение разговора с ТА1 по внешнему выходу. По окончании разговора транзакт выводится из модели блоком **TERMINATE** с меткой **Ter1**.

Возвратимся к последовательности блоков, разыгрывающей номер ТА1, с которого поступает заявка на звонок. Пусть теперь требуется не внешний выход, а нужно позвонить одному из абонентов ТА2. В этом случае транзакт направляется к последовательности блоков, начинающейся с метки **Мет2**. Эта последовательность имитирует определение номера ТА2, на который звонят с ТА1. Номер ТА2 разыгрывается также, как и ранее рассмотренный розыгрыш номера ТА1. После определения номера ТА2 проверяется, свободен ли ТА2 с данным номером. Если не свободен, фиксируется несостоявшийся разговор выводом транзакта из модели блоком **TERMINATE** с меткой **Ter3**. Если же ТА2 свободен, транзакт направляется к последовательности блоков, начинающейся с метки **Мет6**. Эти блоки имитируют ведение разговора абонентами ТА1 и ТА2. По окончании разговора транзакт выводится из модели блоком **TERMINATE** с меткой **Ter4**.

Сегмент имитации телефонных разговоров с ТА2 отличается от только что рассмотренного тем, что в нем отсутствует последовательность блоков, имитирующая прерывание разговоров с ТА2 заявками на разговор с ТА1 при отсутствии свободных внешних выходов.

Модель функционирования автоматической телефонной станции

Задание исходных данных

VrMod EQU 3600 ; *Время моделирования,* 1 ед. мод. вр. = 1 с
N1_ EQU 10 ; *Количество ТА1*
N2_ EQU 20 ; *Количество ТА2*
N3_ EQU 3 ; *Количество внешних выходов*
T1_ EQU 40 ; *Время для расчета интервалов между звонками с ТА1*
T2_ EQU 50 ; *Время для расчета интервалов между звонками с ТА2*
Tog1 EQU 3.5 ; *Среднее время ожидания при занятости внешних линий*
Tog01 EQU 0.5 ; *Среднеквадратичное отклонение времени ожидания*
Tog2 EQU 3 ; *Среднее время ожидания при звонке с ТА1 на ТА2*
Tog02 EQU 0.5 ; *Среднеквадратичное отклонение времени ожидания*
Tog3 EQU 2.5 ; *Среднее время ожидания при занятости внешних линий*
Tog03 EQU 0.4 ; *Среднеквадратичное отклонение времени ожидания*
Tog4 EQU 2 ; *Среднее время ожидания при звонке с ТА2 на ТА1*
Tog04 EQU 0.3 ; *Среднеквадратичное отклонение времени ожидания*
Trazg1 EQU 3 ; *Среднее время разговора с ТА1*
Trazg2 EQU 5 ; *Среднее время разговора с ТА2*
P5_ EQU 0.7 ; *Вероятность того, что ТА2 не ответит*
P10_ EQU 0.3 ; *Вероятность того, что ТА1 не ответит*

Сегмент имитации телефонных разговоров с ТА1

GENERATE (Exponential(333,0,(T1_/N1_))),,,,1 ; *Определение номера звонящего ТА1*
Met0 ASSIGN 7,(RN43/1000) ; *Обращение к генератору RN*
Met1 ASSIGN 3+,1 ; *Начало цикла определения номера звонящего ТА1*

TEST LE P7,(P3#(1/N1_)),Met1 ; *Если условие выполнено, то номер ТА1 запомнить в Р3*

GATE NU P3,Met9 ; *Свободен ли телефон с данным номером? Если да,*

SEIZE P3 ; *то звонят с него*

ASSIGN 7,(RN343/1000) ; *Обращение к генератору RN*

TEST LE P7,(N3_/(N2_+N3_)),Met2 ; *Если условие выполнено, то нужен внешний выход*

Поиск свободного внешнего выхода

ASSIGN 4,(N1_+N2_) ; *Подготовка к циклу*

Met3 ASSIGN 4+,1 ; *Начало цикла определения номера свободного внешнего выхода*

GATE U P4,Met4 ; *Свободен ли внешний выход?*

TEST GE P4,(N1_+N2_+N3_),Met3 ; *Все ли внешние выходы проверены и не найдено свободного?*

ADVANCE Tog1,Tog01 ; *Если да, то задержаться и начать поиск внешнего выхода, занятого ТА2*

ASSIGN 4,(N1_+N2_) ; *Подготовка к циклу*

Met21 ASSIGN 4+,1 ; *Начало цикла определения номера внешнего выхода, занятого ТА2*

TEST E X*4,1,Met22 ; *Равно ли значение сохраняемой ячейки единице? Если нет, то перейти к прерыванию разговора*

TEST GE P4,(N1_+N2_+N3_),Met21 ; *Все ли сохраняемые ячейки проверены? Не найдено ни одной, значение которой равно нулю? Если да, то*

TRANSFER ,Met23 ; *все внешние выходы заняты звонками с ТА1*

Прерывание разговора с ТА2

Met22 PREEMPT P4,PR,Met24,,RE ; *Прервать разговор с ТА2 по внешнему выходу без права продолжения*

SAVEVALUE *4,PR ; *Сохранить PR в ячейке с номером в Р4*

ADVANCE (Exponential(222,0,Trazg1)) ; *Имитация разговора с ТА1 по внешнему выходу*

RETURN P4 ; *Освободить внешний выход с номером в Р4*

RELEASE P3 ; *Освободить ТА1 с номером в Р3*

Ter1 TERMINATE ; *Состоявшиеся разговоры с ТА1 по внешним выходам с прерыванием разговоров с ТА2*

Met23 RELEASE P3 ; *Освободить ТА1, разговор с которого по внешнему выходу не состоялся*

TERMINATE ; *Несостоявшиеся разговоры с ТА1 по внешним выходам из-за занятости их ТА1*

Имитация ведения разговоров с ТА1 без прерывания

Met4 SEIZE P4 ; *Занять свободный внешний выход с номером в Р4*

SAVEVALUE P4,PR ; *Занести в ячейку с номером в Р4 уровень приоритета*

ADVANCE (Exponential(222,0,Trazg1)) ; *Имитация разговора с ТА1*

RELEASE P4 ; *Освободить внешний выход с номером в P4*
RELEASE P3 ; *Освободить TA1 с номером в P3*
Ter2 TERMINATE ; *Состоявшиеся разговоры с TA1 по внешним выходам*
Определение номера TA2, на который звонят с TA1
Met2 ASSIGN 7,(RN343/1000) ; *Обращение к генератору RN*
ASSIGN 6,0 ; *Подготовка к циклу*
Met5 ASSIGN 6+,1 ; *Начало цикла определения номера TA2, на который звонят с TA1*
TEST LE P7,(P6#(1/N2_)),Met5 ; *Если условие выполнено, то номер TA2 запомнить в P6*
GATE NU (N1_+P6),Met7 ; *Свободен ли телефон с данным номером? Если да,*
ASSIGN 7,(RN343/1000) ; *то ответит ли он?*
TEST LE P7,P5_,Met6 ; *Если условие выполнено, то не ответит. Тогда*
Met7 ADVANCE Tog2,Tog02 ; *задержаться и*
RELEASE P3 ; *освободить TA1 с номером в P3*
Ter3 TERMINATE ; *Несостоявшиеся разговоры между абонентами TA1 и TA2*
Имитация разговоров абонентов TA1 и TA2
Met6 SEIZE (N1_+P6) ; *Занять TA2 с номером в (N1_+P6)*
ADVANCE (Exponential(222,0,Trazg1)) ; *Имитация разговора между абонентами TA1 и TA2*
RELEASE (N1_+P6) ; *Освободить TA2 с номером в (N1_+P6)*
RELEASE P3 ; *Освободить TA1 с номером в P3*
TERMINATE ; *Состоявшиеся разговоры между абонентами TA1 и TA2*
Met9 TERMINATE ; *TA1 уже был занят, поэтому разговор не состоялся*
Met24 RELEASE (N1_+P3) ; *Прерванные разговоры с TA2*
TERMINATE
Сегмент имитации телефонных разговоров с TA2
GENERATE (Exponential(493,0,(T2_/N2_))) ; *Определение номера звонящего TA1*
Met10 ASSIGN 7,(RN413/1000) ; *Обращение к генератору RN*
ASSIGN 3,0 ; *Подготовка к циклу*
Met11 ASSIGN 3+,1 ; *Начало цикла определения номера звонящего TA2*
TEST LE P7,(P3#(1/N2_)),Met11 ; *Если условие выполнено, то номер TA2 запомнить в P3*
GATE NU (N1_+P3),Met19 ; *Свободен ли телефон с данным номером? Если да,*
SEIZE (N1_+P3) ; *то звонят с него*
Поиск свободного внешнего выхода
ASSIGN 7,(RN343/1000) ; *Обращение к генератору RN*
TEST LE P7,(N3_/(N1_+N3_)),Met12 ; *Если условие выполнено, то нужен внешний выход*
ASSIGN 4,(N1_+N2_) ; *Подготовка к циклу*

Met13 ASSIGN 4+,1 ; *Начало цикла определения номера свободного внешнего выхода*

GATE U P4,Met14 ; *Свободен ли внешний выход?*

TEST GE P4,(N1_+N2_+N3_),Met13 ; *Все ли внешние выходы проверены и не найдено свободного выхода?*

ADVANCE Tog3,Tog03 ; *Если да, то задержаться и*

RELEASE (N1_+P3) ; *освободить телефон с номером (N1_+P3*)

Ter5 TERMINATE ; *Несостоявшиеся разговоры с ТА2 из-за занятости внешних выходов*

Met14 SEIZE P4 ; *Занять свободный внешний выход*

SAVEVALUE P4,PR ; *Запомнить приоритет абонента, ведущего разговор*

ADVANCE (Exponential(222,0,Trazg2)) ; *Имитация разговора между абонентами ТА2 по внешним выходам*

RELEASE P4 ; *Освободить внешний выход*

RELEASE (N1_+P3) ; *Освободить ТА2*

Ter6 TERMINATE ; *Состоявшиеся разговоры с ТА2 по внешним выходам*

Определение номера ТА1, на который звонят с ТА2

Met12 ASSIGN 7,(RN343/1000) ; *Обращение к генератору RN*

ASSIGN 6,0 ; *Подготовка к циклу*

Met15 ASSIGN 6+,1 ; *Начало цикла определения номера ТА1, на который звонят с ТА2*

TEST LE P7,(P6#(1/N1_)),Met15 ; *Если условие выполнено, то номер ТА1 запомнить в Р6*

GATE NU P6,Met17 ; *Свободен ли телефон с данным номером? Если да,*

ASSIGN 7,(RN343/1000) ; *то ответит ли он?*

TEST LE P7,P10_,Met16 ; *Если условие выполнено, то не ответит. Тогда*

Met17 ADVANCE (Normal(211,Tog2,Tog02)) ; *задержаться и*

RELEASE (N1_+P3) ; *освободить телефон с номером N1_+P3*

Ter7 TERMINATE ; *Несостоявшиеся разговоры с ТА2 из-за занятости ТА1*

Имитация разговоров абонентов ТА2 и ТА1

Met16 SEIZE P6 ; *Занять ТА1 с номером в Р6*

ADVANCE (Exponential(222,0,Trazg2)) ; *Имитация разговора между абонентами ТА2 и ТА1*

RELEASE P6 ; *Освободить ТА1 с номером в Р6*

RELEASE (N1_+P3) ; *Освободить ТА2 с номером N1_+P3*

Ter8 TERMINATE ; *Состоявшиеся разговоры между абонентами ТА2 и ТА1*

Met19 TERMINATE

Сегмент задания времени моделирования и расчета результатов моделирования

GENERATE VrMod ; *Задание времени моделирования*

TEST E TG1,1,Met20 ; *Если содержимое счетчика завершений равно единице, то рассчитать*

SAVEVALUE Ver1,((N$Ter2+N$Ter4)/N$Met0) ; *Вероятность ведения разговоров с ТА1*

SAVEVALUE Ver2,((N$Ter6+N$Ter8)/N$Met10) ; *Вероятность ведения разговоров с ТА2*

Met20 TERMINATE 1 ; *Фиксация очередного прогона*

Приведем результаты моделирования, полученные после 1000 прогонов.

SAVEVALUE	**RETRY**	**VALUE**
VER1	**0**	**0.614**
VER2	**0**	**0.396**

Вероятность ведения разговоров с ТА2 Ver2 равна 0.396, что меньше, чем вероятность Ver1, равная 0.614. Одним из факторов, влияющим на это, является вероятность того, что абонент ТА1 на звонок абонента ТА2 не ответит (P10_ = 0,7).

Оставим без изменений все исходные данные, установим лишь равную вероятность отсутствия ответа на звонки абонентами обеих категорий: P10_ = P5_ = 0,3.

По окончании моделирования получим:

SAVEVALUE	**RETRY**	**VALUE**
VER1	**0**	**0.584**
VER2	**0**	**0.625**

7 МОДЕЛИРОВАНИЕ ТЕЛЕКОММУНИКАЦИОННЫХ СЕТЕЙ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММЫ OpNet

Сведения о программе OpNet. Программа OpNet представляет собой комплекс средств для создания, моделирования и изучения сетей связи. Позволяет анализировать воздействия приложений типа клиент – сервер и новых технологий на работу сети; моделировать иерархические сети, многопротокольные локальные и глобальные сети с учетом алгоритмов маршрутизации; осуществлять оценку и анализ производительности смоделированных сетей. Также с помощью пакета можно осуществить проверку протокола связи, анализ взаимодействий протокола, оптимизацию и планирование сети.

Программа OpNet содержит исчерпывающую библиотеку протоколов и объектов. Есть несколько сред редактора – по одной для каждого типа объекта. Организация объектов – иерархическая, сетевые объекты (модели) связаны набором узлов и объектов связи, в то время как объекты узла связаны набором объектов, таких, как модули очередности, модули процессора, передатчиков и приемников.

Программный пакет предлагает пользователям графическую среду для создания, выполнения и анализа событийного моделирования сетей связи. Это удобное программное обеспечение может быть использовано для большого ряда задач: создания и проверки протокола связи, анализа взаимодействия протоколов, оптимизации и планирования сети. Также с помощью пакета можно осуществить проверку правильности аналитических моделей и описание протоколов. В

рамках «редактора проекта» могут быть созданы разнообразные сетевые объекты, которым пользователь может присвоить различные формы соединения узлов вплоть до имеющих вид «головоломки». Автоматически создаются такие сетевые топологии, как кольца, звезды, случайные сети. Случайный трафик может быть автоматически сгенерирован из алгоритмов, указанных пользователем, а также импортирован из входящих в стандартную комплектацию пакета форматов реальных трафиков линий. Результаты моделирования могут быть проанализированы, а графы и анимация трафика будут сгенерированы автоматически. Одним из плюсов создания модели сети с помощью программного обеспечения является то, что уровень гибкости, обеспечиваемый ядром моделирования, тот же, что и для моделей, написанных «с нуля», но объектное построение среды дает возможность пользователю намного быстрее осуществлять разработку, усовершенствовать и создавать модели для многократного использования.

Версия программного обеспечения для моделирования радиоканала содержит модели антенны радиопередатчика, антенны приемника, перемещающихся объектов узла (включая спутники).

Логику поведения процессора и модулей очередности определяет модель процесса, которую пользователь может создавать и изменять в пределах редактора процесса. В редакторе процесса пользователь может определить модель процесса через комбинацию алгоритма работы конечного автомата (**FSM** – **Finite-State Machine**) и операторов языка программирования C/C++.

Вызов события модели процесса в течение моделирования управляется прерыванием, а каждое прерывание соответствует событию, которое должно быть обработано моделью процесса.

Основа связи между процессами – структура данных, называемая пакетом. Могут быть заданы форматы пакета, т. е. они определяют, какие поля могут содержать такие стандартные типы данных, как целые числа, числа с плавающей запятой и указатели на пакеты (такая особенность позволяет инкапсулировать моделирование пакета). Структура данных, вызывающая информацию по контролю за интерфейсом (**ICI** – **Interface Control Information**), может быть разделена между двумя событиями моделей процесса – это еще один механизм для межпроцессорной связи, что очень удобно для команд моделирования и соответствует архитектуре многоуровневого протокола. Процесс также может динамически порождать дочерние процессы, которые упростят функциональное описание таких систем, как серверы.

В базовую комплектацию пакета входят несколько из наиболее употребимых моделей процесса, например, **BGP** (**Border Gateway Protocol**), **TCP/IP**, **Frame Relay**, **Ethernet**, **ATM** (**Asynchronous Transfer Mode**) и **WFQ** (**Weighted Fair Queuing**). Базовые модели полезны для быстрого развития сложных сетевых структур широкого назначения, а также для точного функционального описания протоколов при обучении студентов. Существует возможность сопровождения моделей сети, узла или процесса комментариями и графикой (с поддержкой гипертекста).

7.1 Моделирование корпоративной сети

Краткая теоретическая справка. Основу информационной системы корпоративной сети составляет вычислительная система, включающая такие компоненты, как кабельная сеть и активное сетевое оборудование, компьютерное и периферийное оборудование, оборудование хранения данных (библиотеки), системное программное обеспечение (операционные системы, системы управления базами данных), специальное ПО (системы мониторинга и управления сетями) и в некоторых случаях прикладное ПО. Как правило, в корпоративной сети используется централизованная система управления сетью.

Ядро сети представляет собой маршрутизатор и серверы различного назначения. На маршрутизаторе хранится и рассчитывается таблица маршрутизации. По принципам формирования таблицы маршрутизации бывают статические и динамические.

При моделировании корпоративной сети рекомендуется использовать статические таблицы маршрутизации, т. к. количество маршрутизаторов в среднем не превышает трех, при построении более крупных сетей со сложной конфигурацией оборудования целесообразно использовать динамическую маршрутизацию. Доступ пользователей к ядру сети осуществляется через коммутаторы, организующие локальные сети. Основной технологией для построения корпоративной сети в настоящее время является **Ethernet**.

Моделирование сети в OpNet IT Guru Academic Edition. При создании модели сети сначала необходимо дать названия проекту и сценарию в нем: **File-New** – название проекта (рисунок 7.1).

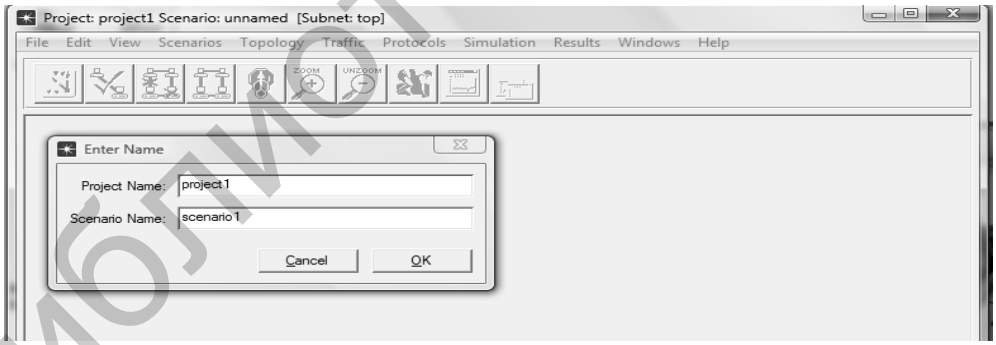


Рисунок 7.1 – Название проекта

Далее выбираем размер сети (в первой части работы строим корпоративную сеть, выбираем **campus** или **office**), выделяем необходимое и нажимаем **Next** (рисунок 7.2).

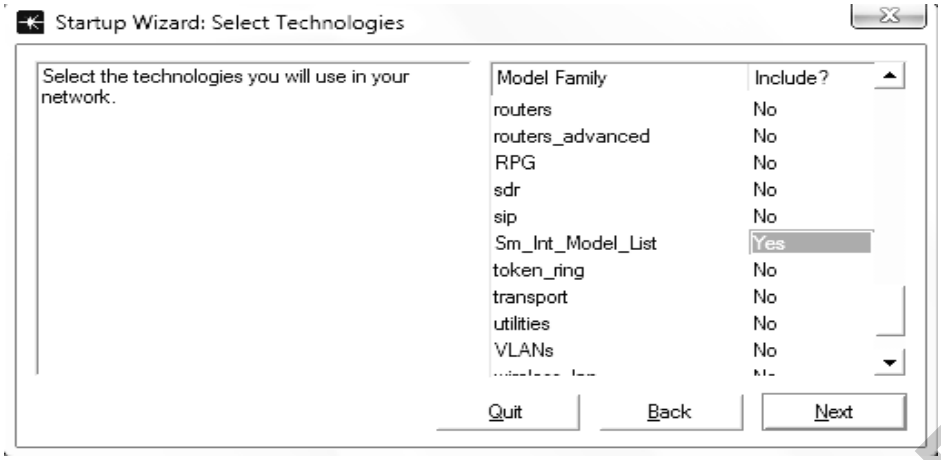


Рисунок 7.2 – Выбор размера сети

На следующем этапе необходимо выбрать конкретные размеры местности в метрах, на которой будет располагаться сеть (механизм выбора размеров местности аналогичен выбору размера сети). Далее необходимо определиться с выбором оборудования и технологий, которые будут представлены в проекте. Для этого в списке выделяем необходимый элемент, при этом в поле **Include** появится надпись **Yes**, означающая, что этот пакет включен в проект (рисунок 7.3).

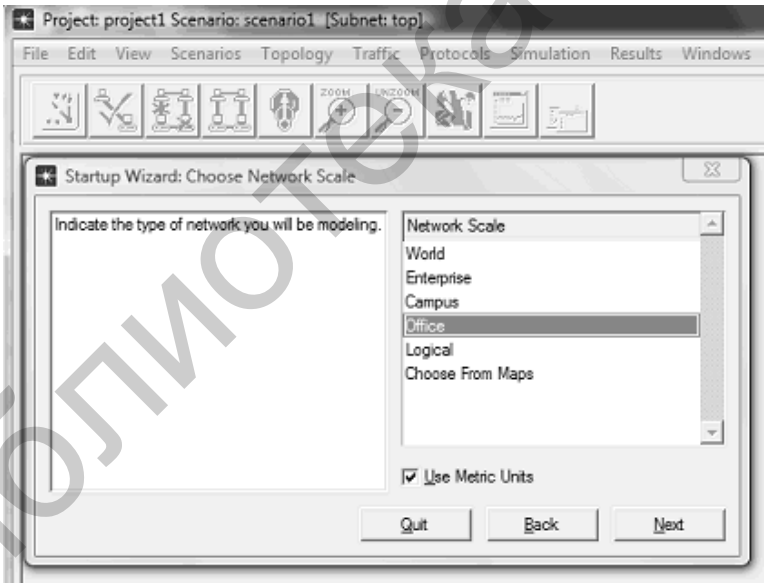


Рисунок 7.3 – Выбор оборудования и технологий

После нажатия кнопки **Next** программа предлагает убедиться в правильности введенных данных. После проверки параметров создаваемого проекта и нажатия кнопки **ОК** появляется рабочая область, где будет создаваться сеть, и палитра, где отображаются элементы, которые можно использовать в проекте (рисунок 7.4).

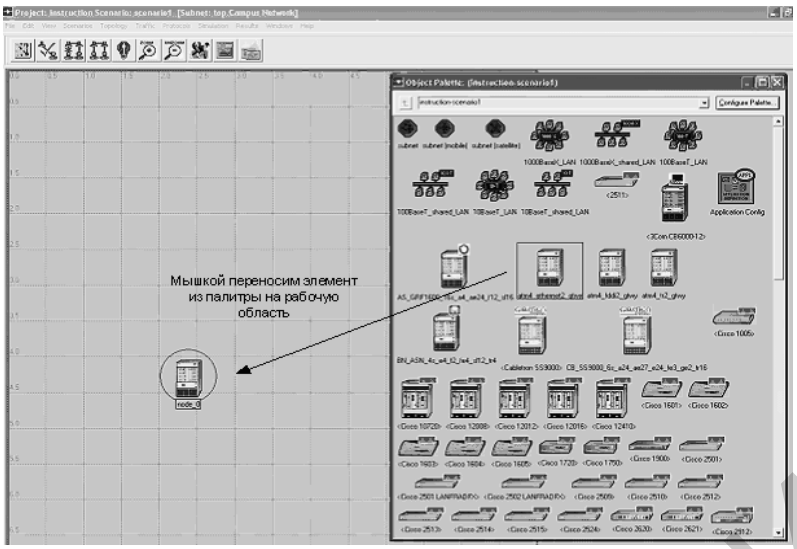


Рисунок 7.4 – Палитра проекта и рабочая область

Набор элементов можно изменить путем нажатия кнопки **Configure Palette**. Элементы на рабочую область можно переносить из палитры. Для этого выделяем элемент в палитре щелчком левой кнопки мыши, вторым аналогичным щелчком, но уже в рабочей области, добавляем элемент в рабочую область.

Кроме того, в программе есть возможность создания комбинированных элементов из шаблонов. Для этого выбираем на панели инструментов **Topology/Rapid Configuration** и, следуя пунктам, конфигурируем шаблон. Пример построения офисной сети представлен на рисунке 7.5. Сеть расположена на двух этажах, на каждом этаже пользователи подключены к коммутаторам, которые в свою очередь объединены маршрутизатором. Один коммутатор соединен с сервером. Рядом с сервером добавлены элементы, которые определяют тип трафика в сети и род приложений, работающих в этой сети.

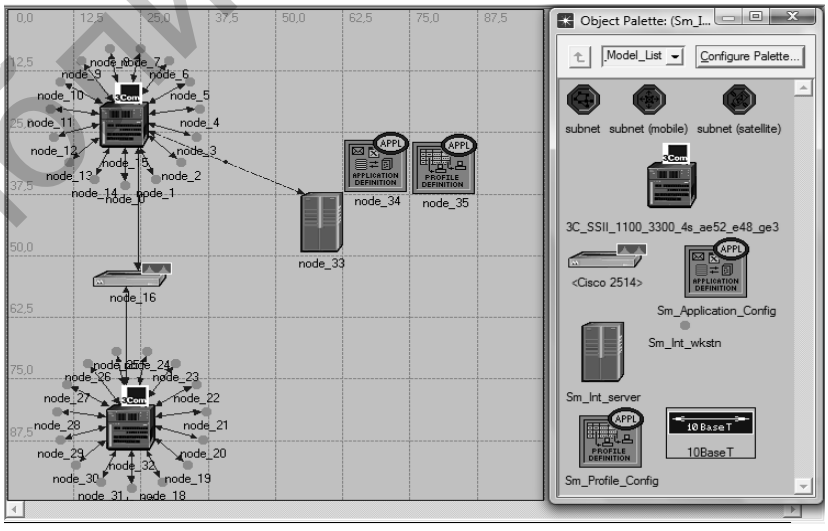


Рисунок 7.5 – Пример построения сети

После построения модели сети ее необходимо верифицировать, т. е. проверить топологию сети. Для этого на панели инструментов нажимаем кнопку с красной галочкой. Если сеть построена неправильно, **OpNet** красными крестиками обозначит некорректные соединения, которые необходимо исправить. В случае правильного построения в рабочей области не будет никаких изменений.

Задание профиля трафика, настройка оборудования, выбор типа собираемой статистики. Для настройки оборудования необходимо щелкнуть правой кнопкой мыши на этом оборудовании и выбрать в меню пункт **Edit Attributes** (рисунок 7.6). В зависимости от выбранного оборудования в рабочей области появится окно с различным набором настраиваемых параметров.

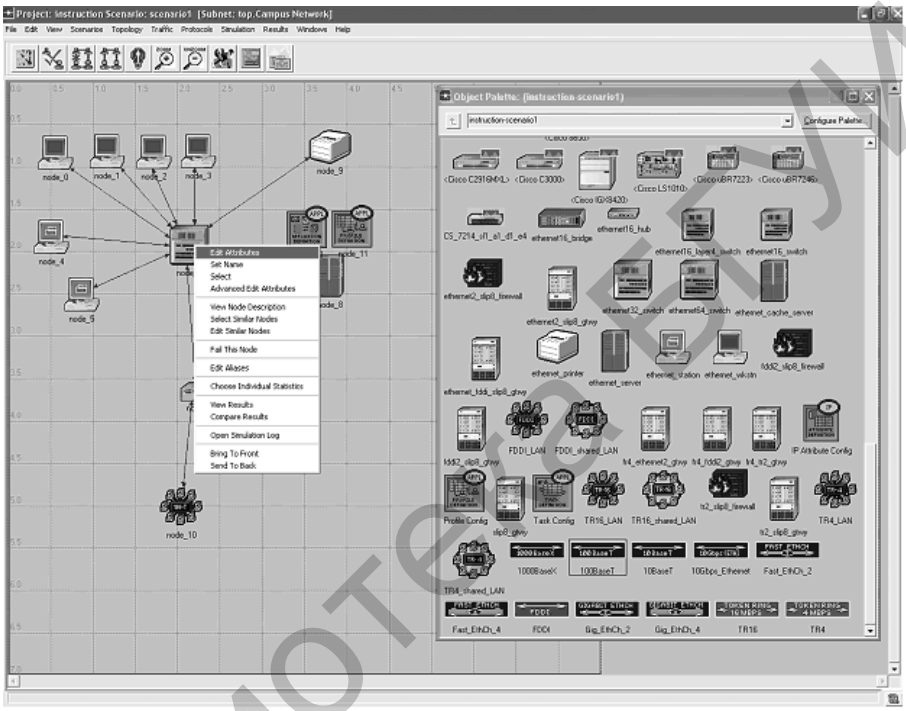


Рисунок 7.6 – Пункт ***Edit Attributes***

Настройка маршрутизатора. При настройке оборудования необходимо построить статическую таблицу маршрутизации, задать интерфейсы, выбрать тип собираемой статистики. Поскольку моделируется небольшая корпоративная сеть, содержащая не более трех маршрутизаторов, в которой редко происходят изменения (подключение нового узла, добавление нового маршрута и т. п.), целесообразно строить статическую таблицу маршрутизации.

При построении крупной сети, содержащей три и более маршрутизаторов, используется динамическая таблица маршрутизации. Для настройки параметров маршрутизатора нужно щелкнуть правой кнопкой мыши на его названии и выбрать графу **Edit Attributes** (рисунок 7.7).

Для построения статической таблицы маршрутизации необходимо вручную прописать IP-адреса и маски, для этого в поле **Attribute** выбираем пункт **IP Routing Parameters/Static Routing Table**.

В строке **rows** указываем количество активных интерфейсов маршрутизатора. Для каждого интерфейса (строки **row**) прописываем IP-адрес и маску.

Чтобы задать параметры интерфейсов, в поле **Attribute** выбираем пункт **Interface Information**, в котором номер строки (**row**) соответствует номеру интерфейса. Для каждого интерфейса, соответствующего одному или нескольким активным портам, задаем IP-адрес и маску подключенного оборудования. Аналогично настраиваем **Loopback Interface**.

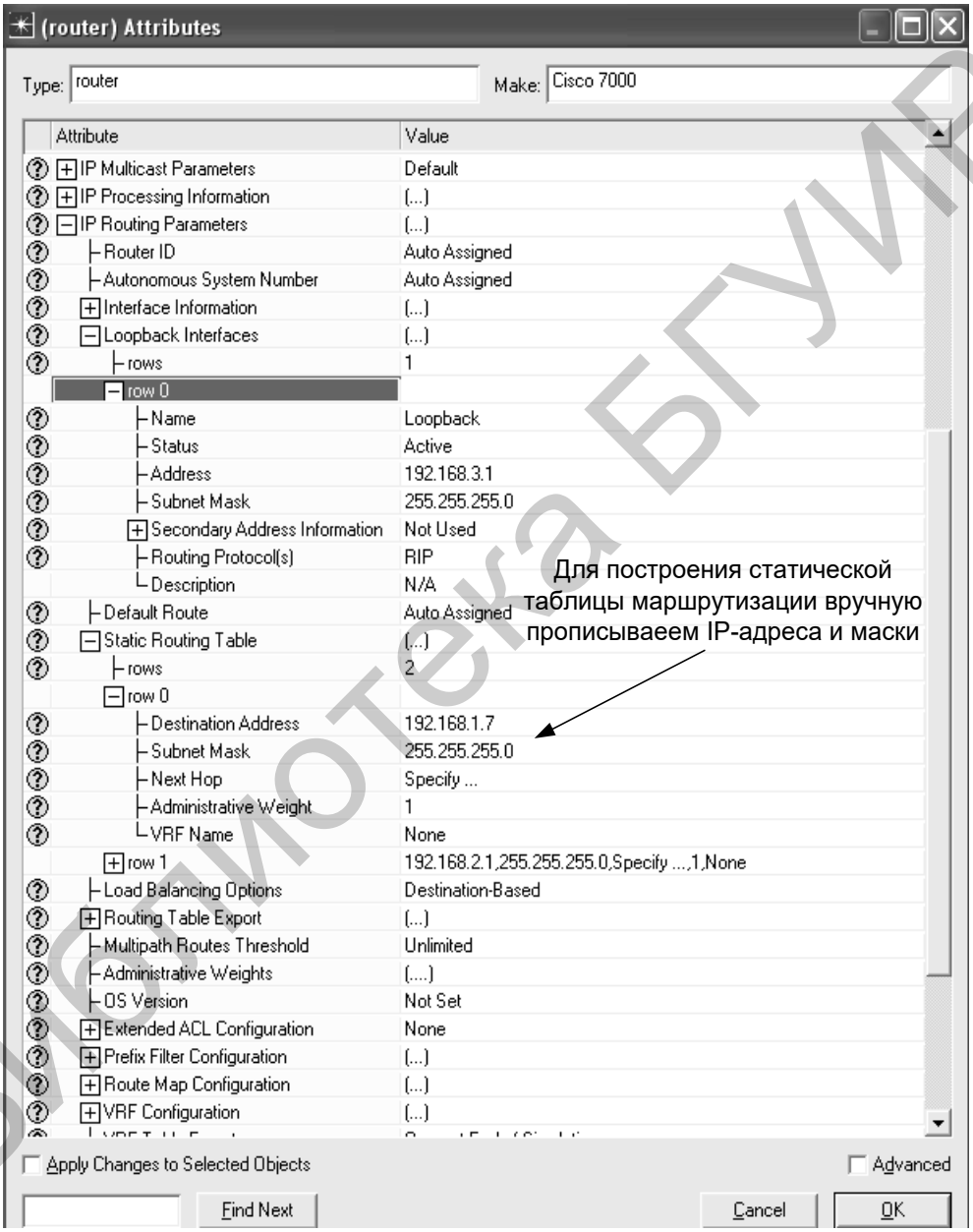


Рисунок 7.7 – Пример настройки маршрутизатора

Loopback Interface – это IP-интерфейс с адресом в сети 127.0.0.1, используется для адресации узлом самого себя (loopback, интерфейс обратной связи).

Обращение по адресу **Loopback Interface** означает связь с самим собой (без выхода пакетов данных на уровень доступа к сети). Для протоколов на транспортном уровне и выше такое соединение неотличимо от соединения, проходящего через сеть, что удобно использовать, например, для тестирования сетевого ПО.

Для того чтобы после процесса симуляции можно было посмотреть таблицу маршрутизации, необходимо в поле **Attribute** выбирать пункт **Routing Table Export** и присвоить полю **Status** значение **Enabled**.

После того как настройка оборудования завершена, необходимо указать тип собираемой статистики. Для этого на исследуемом оборудовании или соединительной линии щелкнуть правой кнопкой мыши и выбрать графу **Choose Individual Statistics**. Далее для каждого сетевого элемента предлагаются на выбор варианты сбора результатов моделирования (рисунок 7.8).

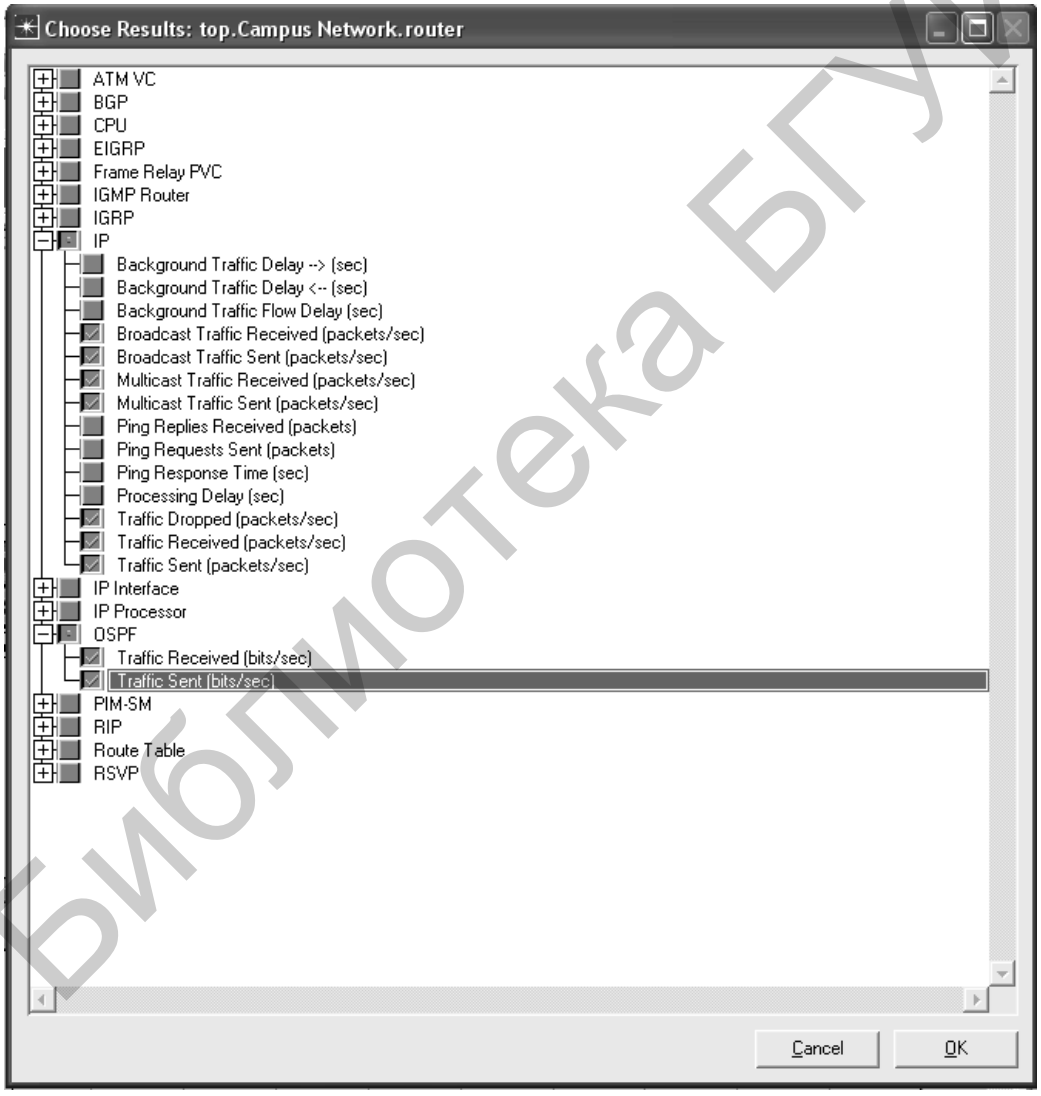


Рисунок 7.8 – Выбор типа собираемой статистики для маршрутизатора

Для маршрутизатора обязательно собрать следующие статистические показатели: загрузка процессора, объем трафика переданного, полученного, отброшенного по протоколу IP.

Настройка коммутатора. Как правило, дополнительная настройка коммутатора не требуется, если нет необходимости реконфигурации портов или настройки **VLAN**. Для коммутатора можно указать тип собираемой статистики – щелкнуть правой кнопкой мыши на объекте и выбрать графу **Choose Individual Statistic**. Далее галочками выбрать интересуемые параметры. Для коммутатора обязательно собрать следующие статистические показатели: объем трафика, переданного, полученного, отброшенного.

Настройка сервера. При настройке сервера нужно прописать тип трафика, генерируемого пользователями. Самые распространенные типы трафика: данные, речь, видео. Каждый из них предъявляет различные требования к передаче, обеспечению необходимого качества обслуживания, выделению достаточной пропускной способности. В зависимости от направления деятельности предприятия по сети будет передаваться трафик различного рода. Соответственно при проектировании важно правильно рассчитать загрузку каналов и оборудования. Проверить расчеты позволяет моделирование планируемой нагрузки. Так, например, речь – это трафик, чувствительный к задержкам, но не требующий большой пропускной способности канала, поэтому особое внимание при проектировании сети стоит уделить обеспечению необходимого качества обслуживания при передаче речи, в частности, можно настроить протокол RSVP или выбрать низкоскоростной канал.

Речевой трафик чувствителен как к задержкам, так и к потерям и требует высокой пропускной способности, а также настройки протокола **PIM** и **IGMP**.

Тип трафика задается с помощью элемента палитры **Application Definition**. Элемент **Application Definition** необходимо перенести из палитры в рабочую область и разместить рядом с сервером.

Application Definition содержит характеристики приложений, создаваемых в виде потоков и имеющих собственные параметры трафика. Для создания потоков на **Application Definition** нужно щелкнуть правой кнопкой мыши и выбрать графу **Edit Attributes** (рисунок 7.9), где созданы 16 стандартных потоков для таких случаев. Сюда входят доступ к базам данных, обработка электронной почты, передача файлов, работа в Интернете и т. д.

После создания потоков приложений необходимо сконфигурировать профили пользователей, работающих в спроектированной сети. Эту функцию выполняет элемент палитры **Profile Definition** (рисунок 7.10).

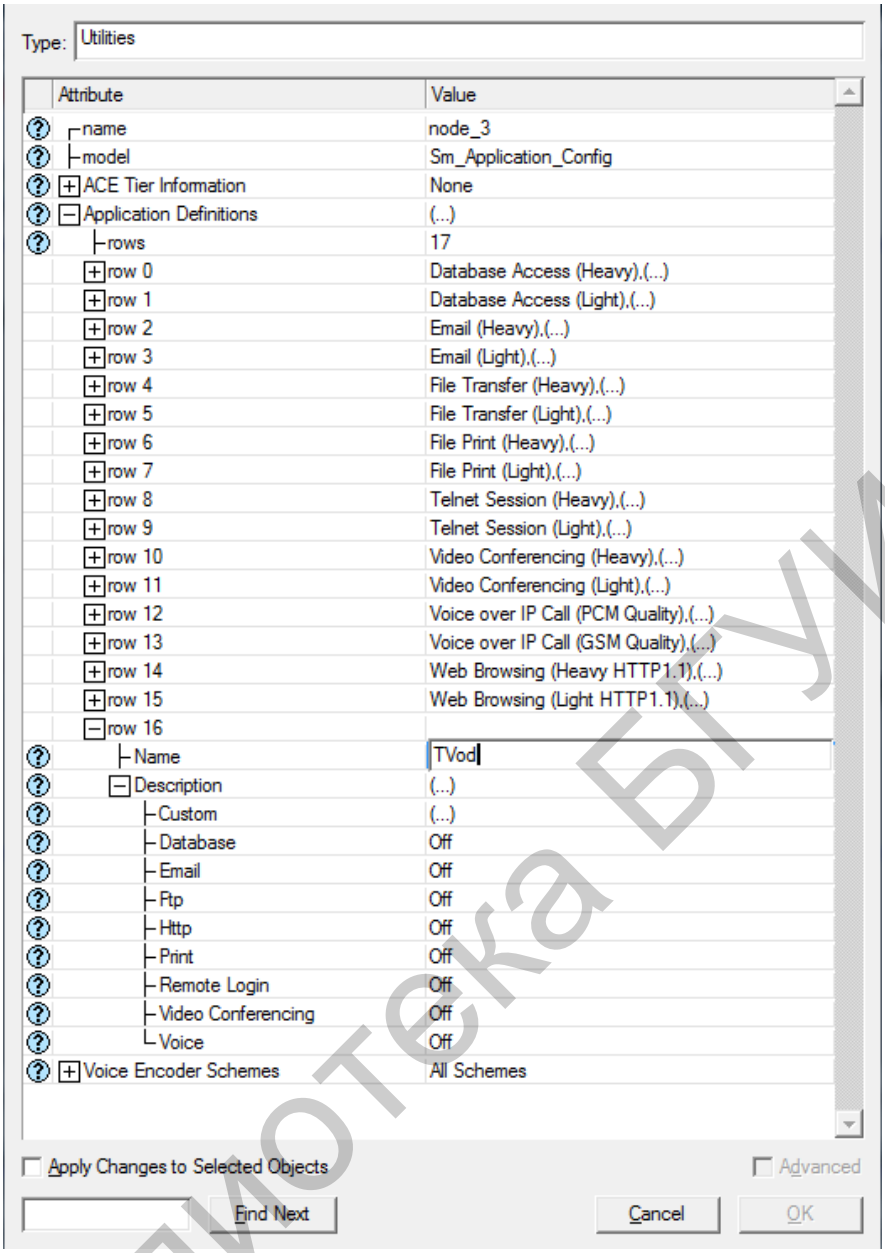


Рисунок 7.9 – Настройка приложений на сервере

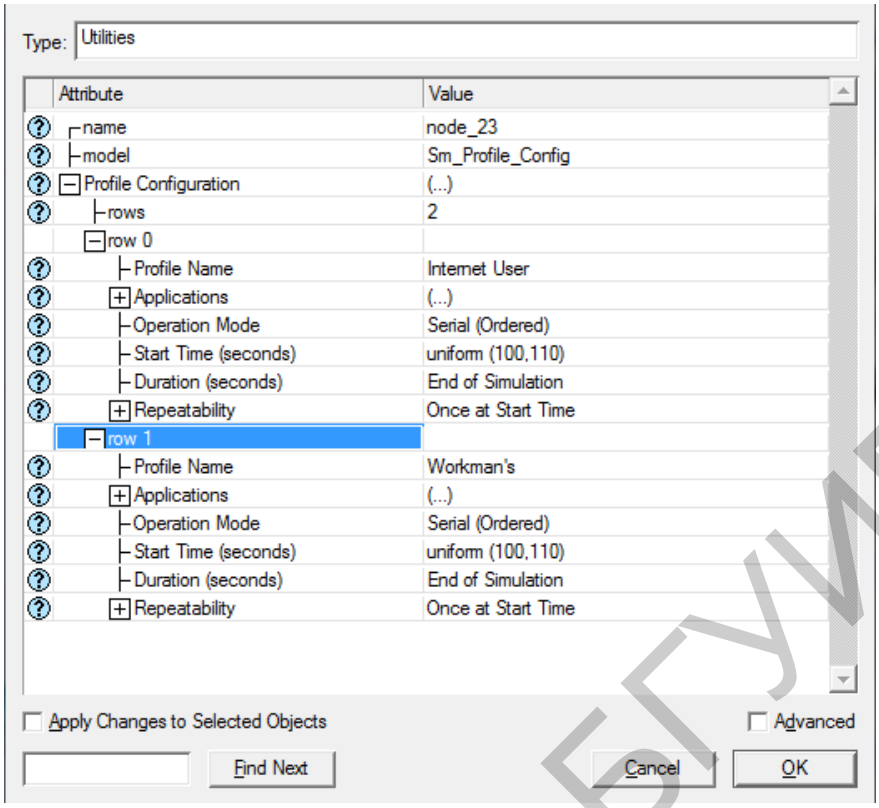


Рисунок 7.10 – Настройка профилей пользователей

Каждому профилю присваивается название и описывается ряд пользовательских характеристик: время начала работы; продолжительность; окончание; интенсивность его пребывания и работы в сети и какими из предложенных (созданных) приложений он пользуется. Указанные параметры задаются аналогично **Application Definition** через пункт меню **Edit Attributes**.

Тип собираемой статистики указывается также через пункт меню **Choose Individual Statistics**. Для сервера необходимо собрать следующие типы статистики: загрузка процессора, обращение к приложениям сервера – полученный и отправленный трафик (**Application Demand** – **Traffic Sent**, **Traffic Received**), загрузка сервера **FTP**, **HTTP**, **E-mail**, **DB**, в разделе **Requesting Server Custom Application** выбрать **Application Response**, **Total Request/Response Size**, **Traffic Received/Sent**, **Application/Group Response Time**, отметить загрузку в **Responding Server Custom Application**.

Настройка оконечных пользователей. При настройке оборудования для оконечных пользователей нужно задать IP-адреса и маски, указать название и тип пользователя. Оконечный пользователь может быть представлен двумя способами: элементом LAN, который имитирует некую сеть абонентов, или рабочей станцией.

В случае когда элемент LAN подключен к маршрутизатору, т. е. является группой оконечных пользователей, настройка параметров данного элемента происходит следующим образом. Выбираем пункт меню **Edit Attributes**. Заносим

число пользователей в графу **Number of Workstations**. Исходя из этого параметра заполняется графа **Application: Supported Profiles**, указывающая, сколько и какого рода пользователи будут присутствовать в этой подсети. Здесь аналогично серверу создаются потоки, учитывающие пользователей сети, на основе профилей, созданных в элементе рабочей области **Profile Definition** (рисунок 7.11).

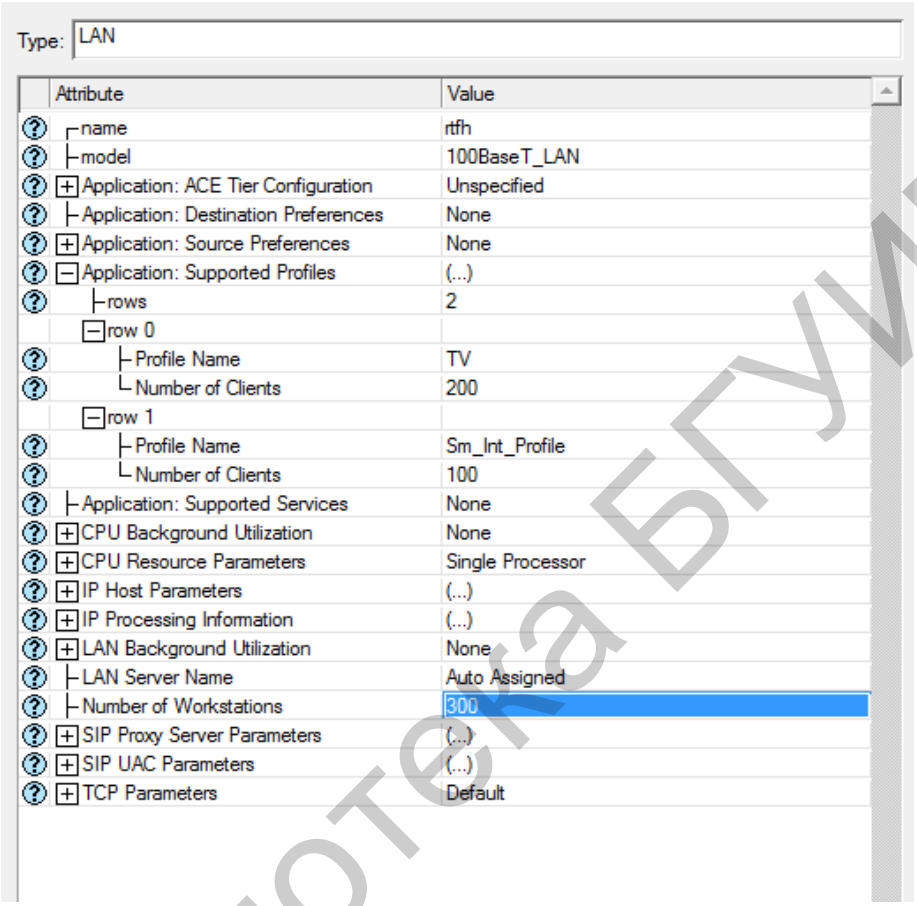


Рисунок 7.11 – Настройка LAN

Необходимо указать количество профилей пользователей в графе **rows**. Профили пользователей образуют потоки трафика определенного типа. Далее для каждого созданного таким образом потока указывается название профиля и число пользователей.

Число пользователей всех типов в итоге должно быть равно числу пользователей всей подсети. При настройке LAN можно прописать статистическую таблицу маршрутизации так же, как на маршрутизаторе.

В случае когда оконечным пользователем является рабочая станция, настройка параметров принципиально не меняется. Только количество пользователей всегда будет равно единице. Тип собираемой статистики указывается через пункт меню **Choose Individual Statistics**.

Для оконечных пользователей снять следующую статистику:

– задержку; вариацию задержки; объем полученного и отправленного трафика для следующих типов приложений: **Video Called Party**, **Video Calling Party**, **Video Conferencing**, **Voice Application**;

– загрузку процессора;

– количество загруженных объектов/страниц (**Downloaded Objects/Pages**) для **Client http**;

– размеры загруженных файлов (**Downloaded File Size**) и **Downloaded Response Time** для **Client Ftp**;

– объемы полученного/переданного трафика (**Traffic Received/Sent**) для **Client E-mail и Client DB**;

– загрузку, задержку, объем полученного/переданного трафика (**Traffic Received/Sent**) в разделе **Ethernet**;

– **Utilization** в разделе **EtherChannel**.

Теперь можно приступать к симуляции, предварительно сохранив проект, нажав в меню проекта кнопку **Save**. Проект будет сохранен в **C:\Documents and Settings\Guest\op_models**. При повторном запуске программы OpNet для открытия существующего проекта необходимо в меню **File** выбрать **Open** и название своего проекта.

Процесс симуляции. Оценка полученных результатов. Перед началом процесса симуляции необходимо настроить некоторые ее параметры. Для этого на панели инструментов нужно нажать кнопку **Configure/Run simulation** и войти в режим симуляции (рисунок 7.12).

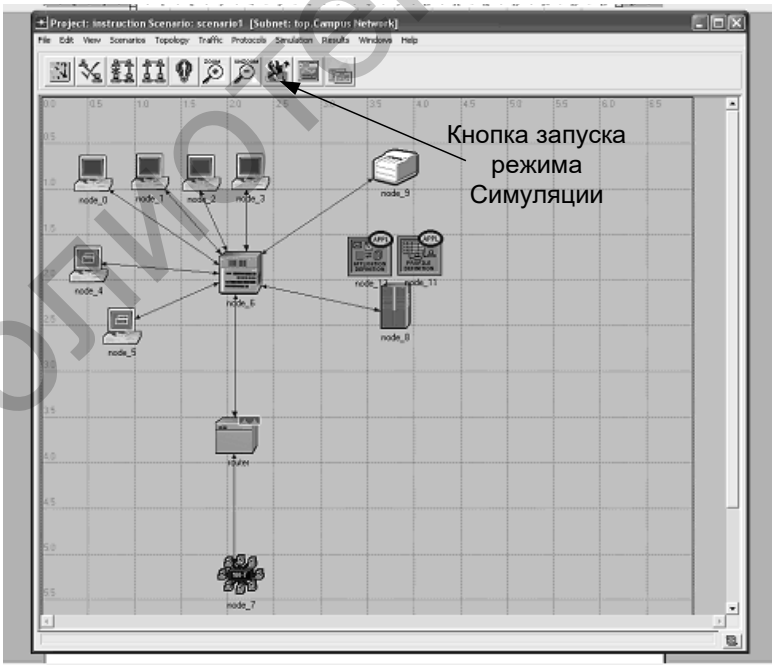


Рисунок 7.12 – Кнопка запуска режима симуляции

Пакет OpNet предлагает указать продолжительность работы сети (в данном случае – 2 ч). В следующих закладках имеется возможность настройки глобальных

параметров сети, параметров моделирования для каждого элемента, вывода отчетов, анимации во время моделирования и др. Для просмотра таблицы маршрутизации необходимо перейти на закладку **Global Attributes**, в поле **Attribute** выбрать **IP Routing Table Export/Import** и установить значение **Import**. Для данной сети таблица маршрутизации прописывалась вручную (статистическая маршрутизация), поэтому необходимо выключить протоколы динамической маршрутизации **RIP**, **OSPF**, **BGP**. Для этого в поле **Attributes** выбираем **RIP Sim Efficiency** и устанавливаем значение **Disabled** (для **OSPF** и **BGP** произвести аналогичные операции). Теперь можно запускать процесс моделирования. Чтобы запустить симуляцию, нужно нажать кнопку **Run** (рисунок 7.13).

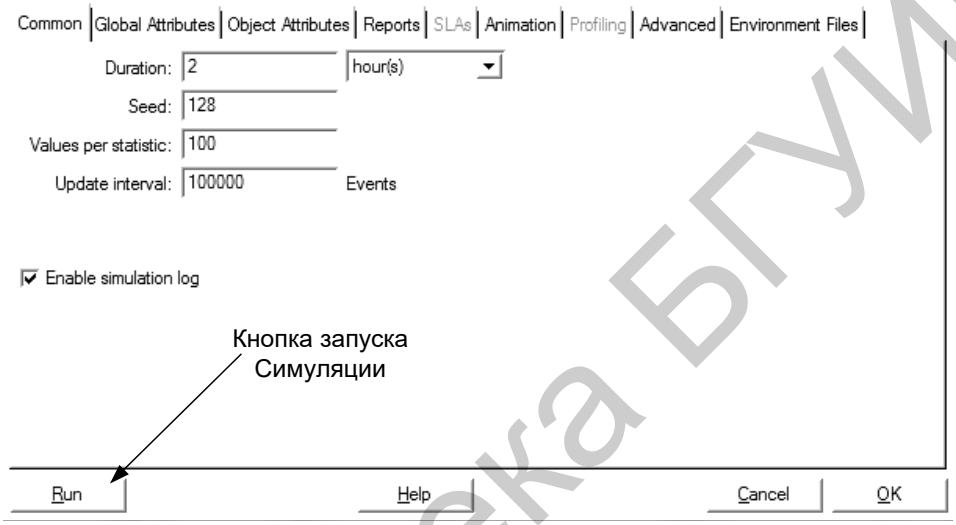


Рисунок 7.13 – Настройка параметров моделирования

После завершения процесса симуляции нагрузку на сеть можно увидеть, перейдя на закладку **Simulation Speed** (рисунок 7.14).

Во время моделирования процессов, происходящих в построенной сети, в реальном времени на экране строится график активности, состоящий из двух кривых: синяя отображает ситуацию в каждый момент времени (время откладывается по оси абсцисс), красная показывает среднее значение.

Для просмотра результатов моделирования работы сети «под нагрузкой» выбираем пункт **View Result**, для этого на рабочей области необходимо щелкнуть правой кнопкой мыши и в меню выбрать соответствующий пункт либо нажать на панели инструментов кнопку  .

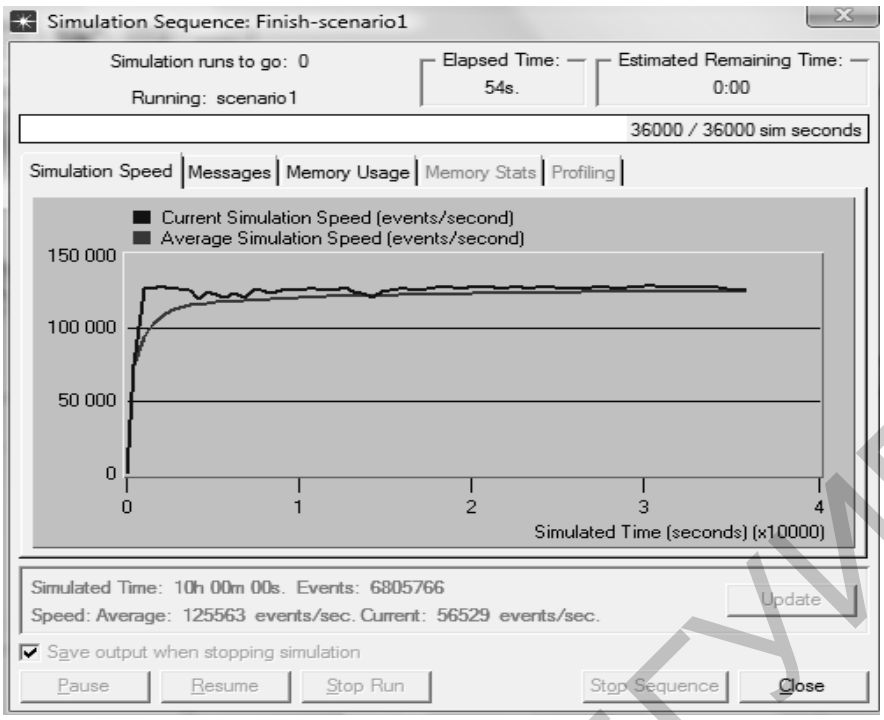


Рисунок 7.14 – Результаты моделирования

В окне **View Result** можно выбрать интересующий тип статистики на различном оборудовании, например, на рисунке 7.15 показан график количества отброшенных на маршрутизаторе пакетов за единицу времени (**Traffic dropped**).

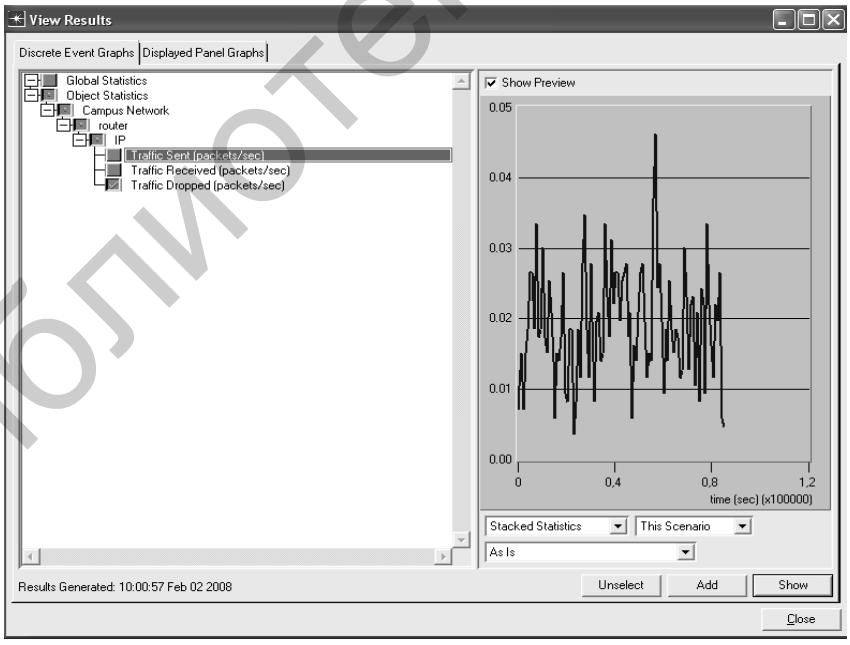


Рисунок 7.15 – Количество отброшенных пакетов за единицу времени

На рисунке 7.16 представлен график нагрузки на канал, по которому можно оценить пропускную способность канала, т. е. видно, что за 2 ч работы сети по каналу передавалось в среднем 65–70 кбит/с.

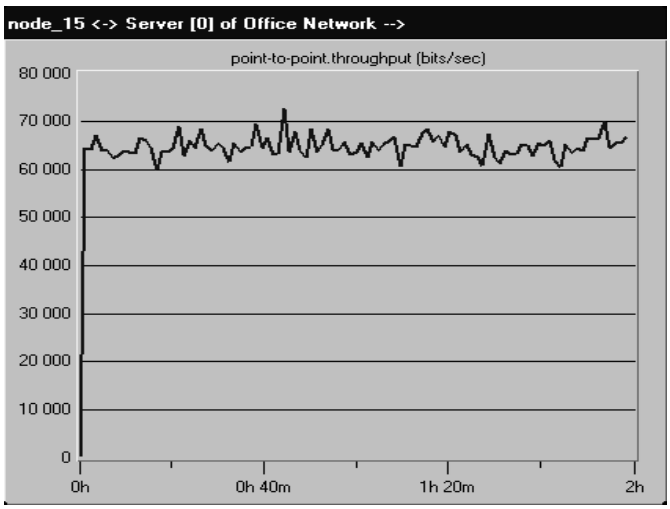


Рисунок 7.16 – График нагрузки на канал

Для просмотра таблицы маршрутизации нужно щелкнуть правой кнопкой мыши на маршрутизаторе и выбрать графу **Open Simulation Log**, в открывшемся окне нажать **Common Route Table**.

7.2 Моделирование сети Frame Relay

Процесс моделирования сети Frame Relay предполагает выполнение следующих задач:

– конструирование сети в программном продукте OpNet;
– настройка оборудования;
– симуляция работы сети;
– анализ полученных характеристик сети.

Для начала работы выбираем вкладку **File/Новый**. Изменяем имя проекта на **AO_WAN_Frame_Relay**. Устанавливаем имя сценария **CIR64** и щелкаем кнопкой мыши на **OK** (рисунок 7.17).

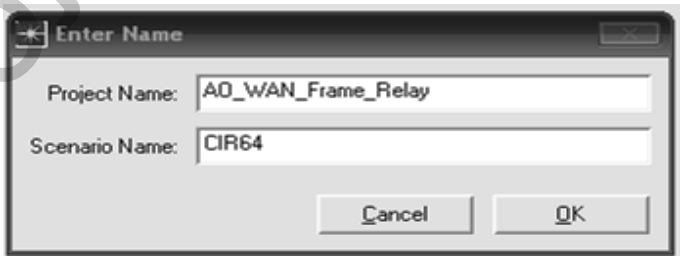


Рисунок 7.17 – Ввод имени проекта и сценария

В начальном окне топологии выбираем **Create Empty Scenario** и нажимаем кнопку **Next**.

В окне выбора сетевого масштаба, которое показано на рисунке 7.18, выбираем **World** и нажимаем кнопку **Next**.

В окне выбора технологий, используя полосу прокрутки, выбираем и включаем модель **frame_relay_advanced**, при этом в поле **Include** появится надпись **Yes**, которая означает, что этот пакет включен в проект (рисунок 7.19).

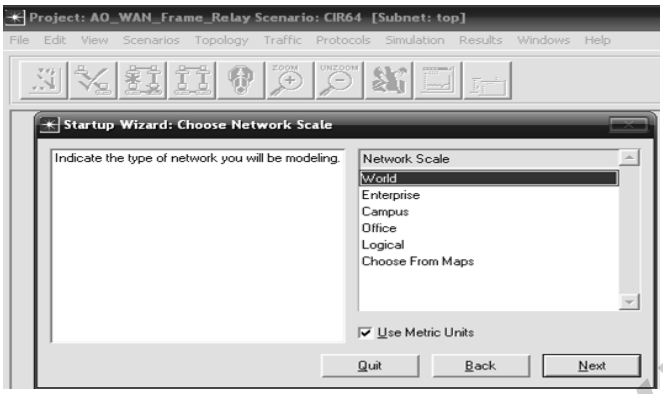


Рисунок 7.18 – Выбор масштаба сети

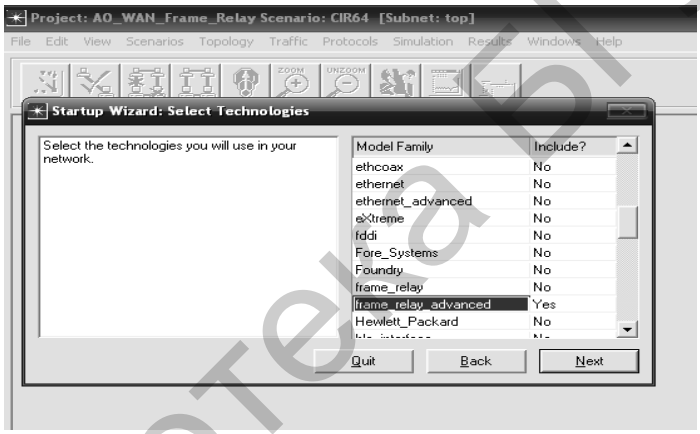


Рисунок 7.19 – Окно выбора технологии

В окне обзора нажимаем **OK**, после чего появляется **Object Palette** – палитра объектов, или палитра выбора оборудования, которое можно использовать в проекте.

Палитра выбора оборудования показана на рисунке 7.20.

Рисунок 7.20 – Палитра выбора оборудования

Для начала необходимо сконфигурировать профили пользователей, работающих в спроектированной сети. Для этого мы выбираем на палитре выбора оборудования и размещаем в рабочем пространстве следующие объекты: **Application Config** и **Profile Config**. Каждому профилю присваивается название и описывается ряд пользовательских характеристик: время начала работы; продолжительность; окончание; интенсивность его пребывания и работы в сети и использование предложенных (созданных) приложений. Указанные параметры задаются через пункт меню **Edit Attribute** при открытии вкладки **Application Definitions.**

Для объекта **Application Config** (приложение конфигурации), который, используя протокол передачи (**FTP**), обеспечит генерацию трафика для сети **Frame Relay**, установим следующие параметры (рисунок 7.21):

– имя приложения – **FTP_BURSTY**;

– на вкладке **Ftp** параметру **Inter-Requrst Time** (время между запросами (с)) присвоим значение **exponential (0.1)**;

– размеру файла (в байтах) будет соответствовать значение **constant (1000)**.

Таким образом, данное приложение в среднем будет передавать десять файлов размером 1 Кбайт каждую секунду.

Рисунок 7.21 – Настройка профиля **Application Config**

Для объекта **Profile Config** установим следующие параметры:

– имя **Bursty Profile**;

– для параметра **Repeatability** (повторяемость) значение **Once at Start Time**.

Настройка объекта **Profile Config** показана на рисунке 7.22.

Выбираем объект **fr32_cloud** из палитры объектов и размещаем его в проектное рабочее пространство. Щелчком правой кнопки мыши на облаке зададим название оборудования. Для этого выбираем **Set Name** и вводим имя элемента: **Frame_Relay_Cloud**.

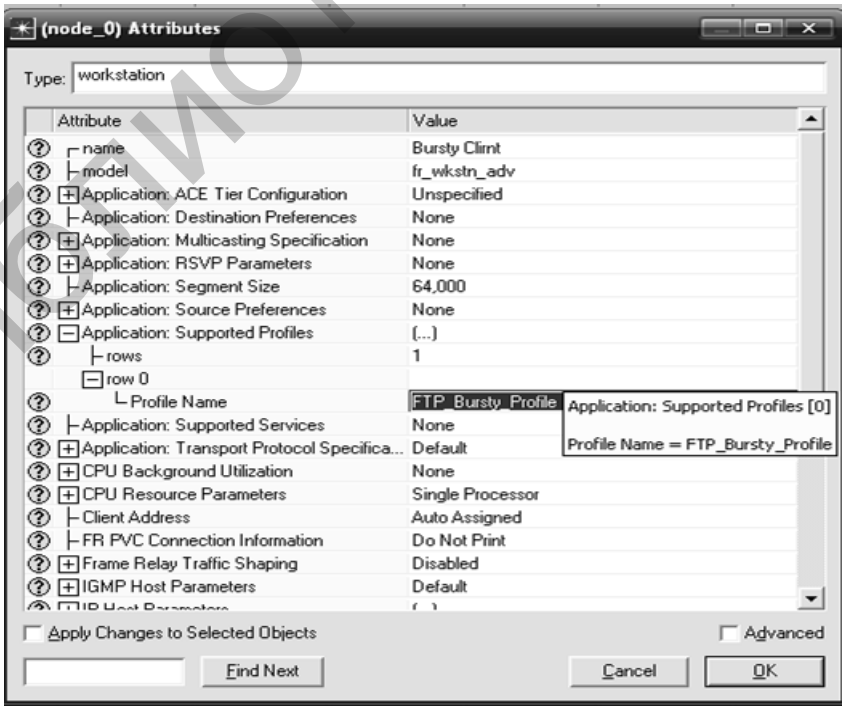


Рисунок 7.22 – Настройка профиля конфигурации

Выбираем устройство **fr_wkstn_adv** (рабочая станция) из палитры объектов и размещаем его в проектное рабочее пространство. Следует обратить внимание, что станция поддерживает протокол ретрансляции кадров. Щелчком правой кнопкой мыши на станции выбираем **Edit Attributes**. Изменяем имя устройства на **Bursty Client**. Редактируем вкладку **Application: Supported Profiles** (признак поддержки профилей). Вставляем строку и устанавливаем имя профиля **FTP_Bursty_Profile**. Настройка рабочей станции показана на рисунке 7.23.

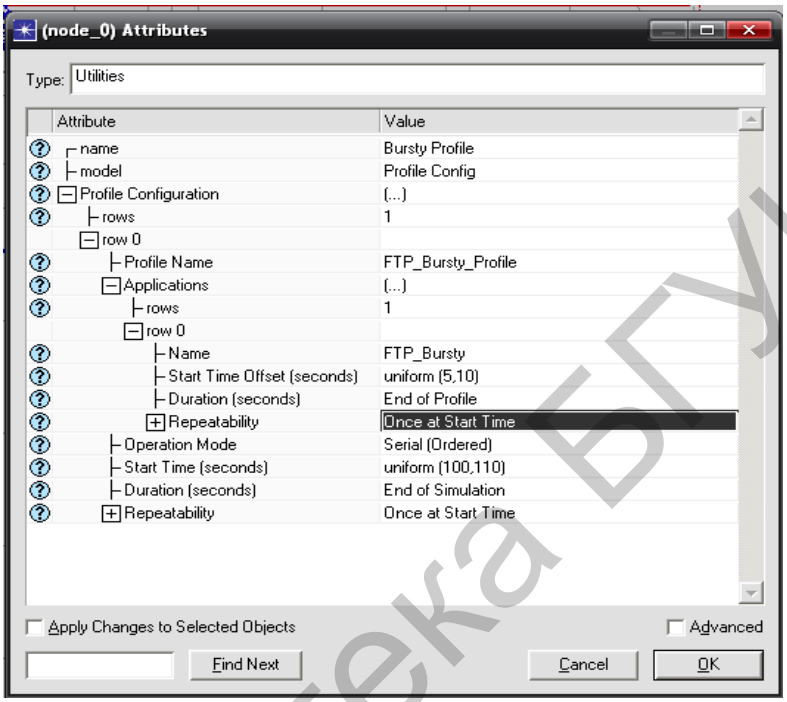


Рисунок 7.23 – Настройка рабочей станции

Выбираем устройство **fr_server_adv** из палитры объектов и размещаем его в проектное рабочее пространство. Следует обратить внимание, что сервер также поддерживает протокол ретрансляции кадров.

Теперь мы установим сервер, чтобы поддержать **Bursty**-приложение **FTP**, которое мы определили.

Щелчком правой кнопки мыши на устройстве выбираем **Edit Attributes**. Изменяем признак имени сервера на **Bursty Server**. Редактируем вкладку **Application: Supported Services** (признак поддержки услуг). Вставляем строку и устанавливаем имя **FTP_BURSTY**. Окно настройки сервера показано на рисунке 7.24.

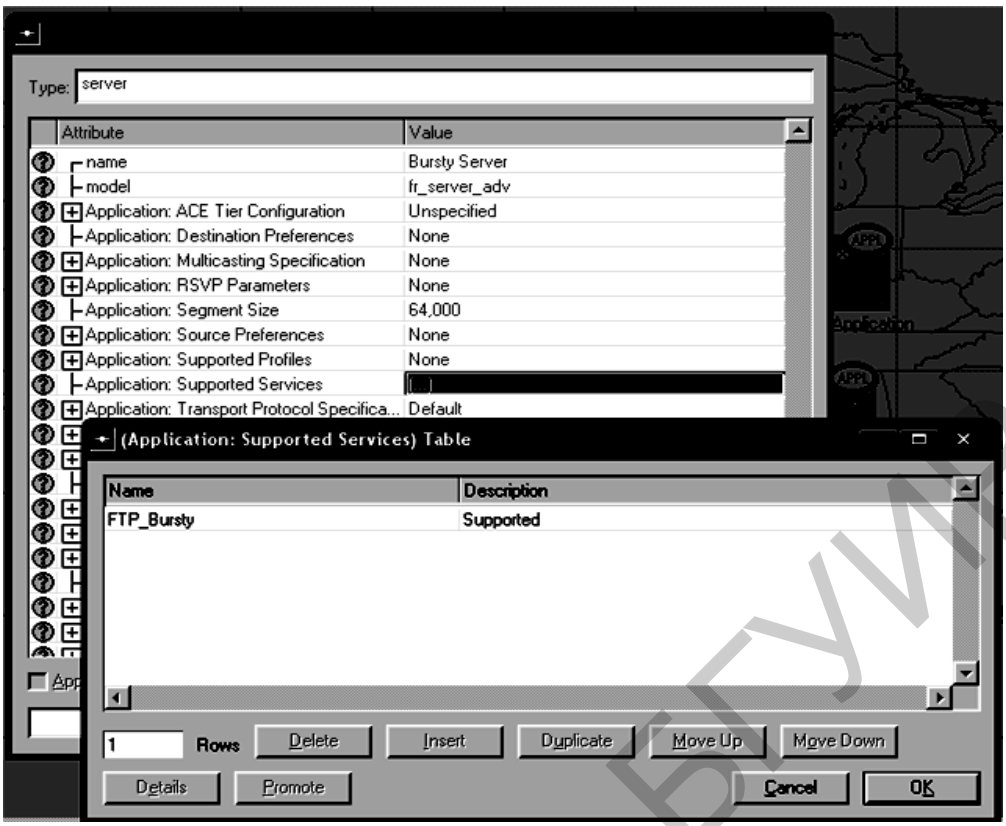


Рисунок 7.24 – Настройка сервера

Выбираем два коммутатора **fr8_switch_adv** на палитре объектов. Размещаем один рядом с клиентом, второй рядом с сервером. Необходимо обратить внимание, что коммутатор включается в работу, когда появляются перегрузки. Тогда он откажется не только от кадров, отмеченных меткой **DE=1**, но даже от нормальных кадров.

Щелчком правой кнопки мыши на коммутаторе клиента выбираем **Set Name**. Устанавливаем имя **Client Switch**.

Таким же образом устанавливаем имя для коммутатора сервера: **Server Switch**.

Выбираем три линии связи **FR_T1_int** на палитре объектов и используем их, чтобы подключить рабочую станцию клиента к одному его коммутатору, сервер к другому коммутатору и оба коммутатора к облаку Frame Relay. При этом необходимо помнить, что скорость T1 =1,5 Мбит/с.

На рисунке 7.25 показана организация связи между элементами сети.

Рисунок 7.25 – Организация линий связи между элементами сети

Наконец, мы можем организовать постоянный виртуальный канал (**PVC**) между рабочей станцией клиента и сервером так, чтобы они могли связываться. Для этого выбираем **FR PVC** – объект из палитры объектов и размещаем его в проектное рабочее пространство.

Щелчком правой кнопки мыши на этом объекте выбираем **Set Name** и устанавливаем имя **PVC Config**. Следует помнить, что объект **FR PVC Config** должен быть помещен в рабочее пространство прежде, чем будет установлен любой **PVC**.

Далее выбираем объект **fr_pvc** на палитре объектов и используем его, чтобы соединить клиента с сервером.

Щелчком правой кнопки мыши на **PVC** раскрываем вкладку **Contract Parameters** (параметры договора). Эти параметры определяют взаимодействие между оконечными элементами системы (клиентом и сервером) и провайдером услуг Frame Relay (рисунок 7.26).

Устанавливаем выходную скорость **CIR**, равную 64 000 бит/с.

Устанавливаем согласованный объем пульсации **Bc** на выходе равным 64 000 бит.

Устанавливаем дополнительный объем пульсации **Be** на выходе равным 32 000 бит.

Согласованная информационная скорость (**CIR**) – скорость, с которой сеть будет передавать данные пользователя.

Согласованный объем пульсации (**Bc**) указывает на максимальное число байтов, которое может быть передано за данный интервал. В данном случае интервал равен 1 с, так **CIR** = **Bc**.

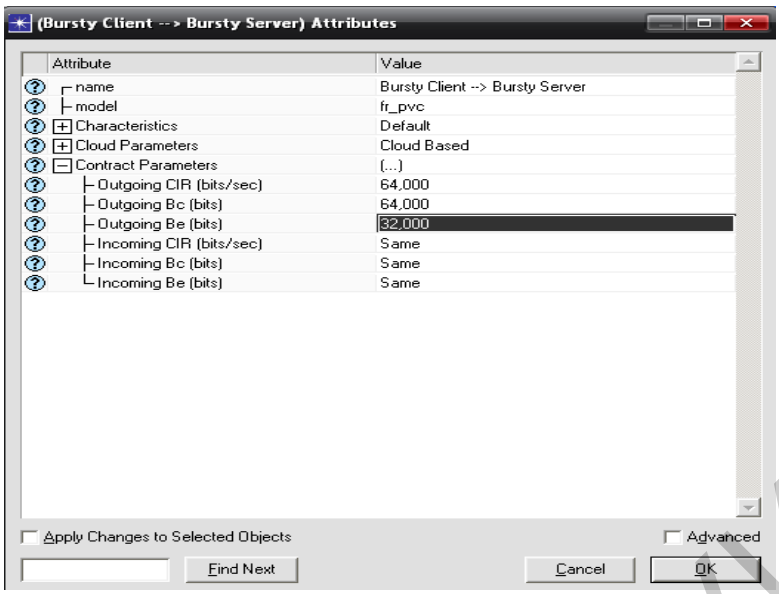


Рисунок 7.26 – Настройка параметров постоянного виртуального канала

Дополнительный объем пульсации (**Be**) – максимальное количество байт, которое сеть будет пытаться передать сверх установленного значения **Bc** за интервал времени **T**, если будут позволять сетевые условия. Дополнительные данные могут быть отклонены, если в сети возникает перегрузка.

В результате мы получаем сеть, которая изображена на рисунке 7.27.

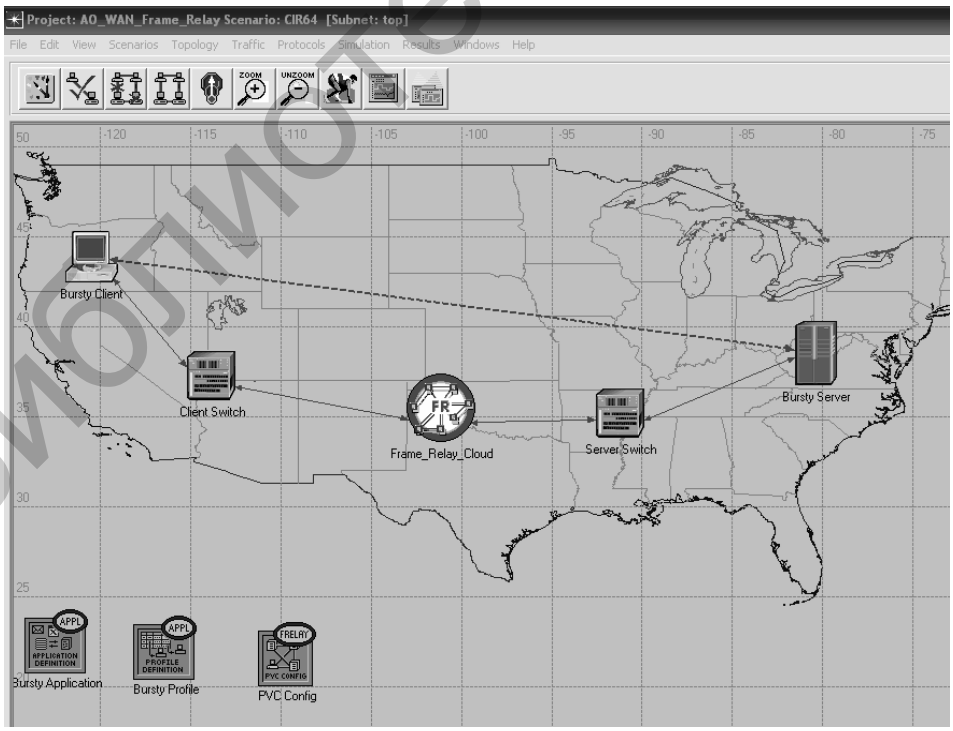


Рисунок 7.27 – Сеть Frame Relay

Перед запуском процесса моделирования необходимо определить, какие статистические данные работы сети и ее узлов будут получены в процессе моделирования.

Для этого выбираем пункт меню **Simulation**, затем вкладку **Choose Individual Statistics** (индивидуальная статистика).

Раскрыв вкладку **Global Statistics** (глобальная статистика), в пункте **Frame Relay** выбираем **Delay** – задержку (с), **Delay Variance** – дисперсию задержки и **Residual Error Rate** – остаточную частоту ошибок.

Раскрыв **FTP**-элемент, выбираем **Download Response Time** (с) и **Traffic Received** (байт/с). Окно выбора вышеперечисленных параметров показано на рисунке 7.28.

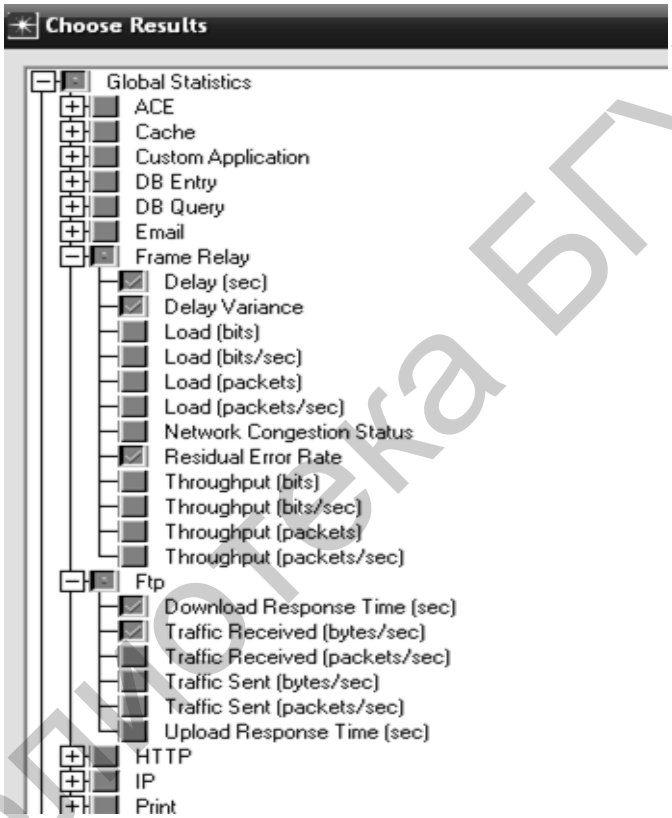


Рисунок 7.28 – Окно выбора данных статистики

В пункте **Node Statistics** (статистика узла) раскрываем элемент **Frame Relay PVC** и выбираем статистику **BECN** состояния, **DE** состояния и **FECN** состояния (рисунок 7.29).

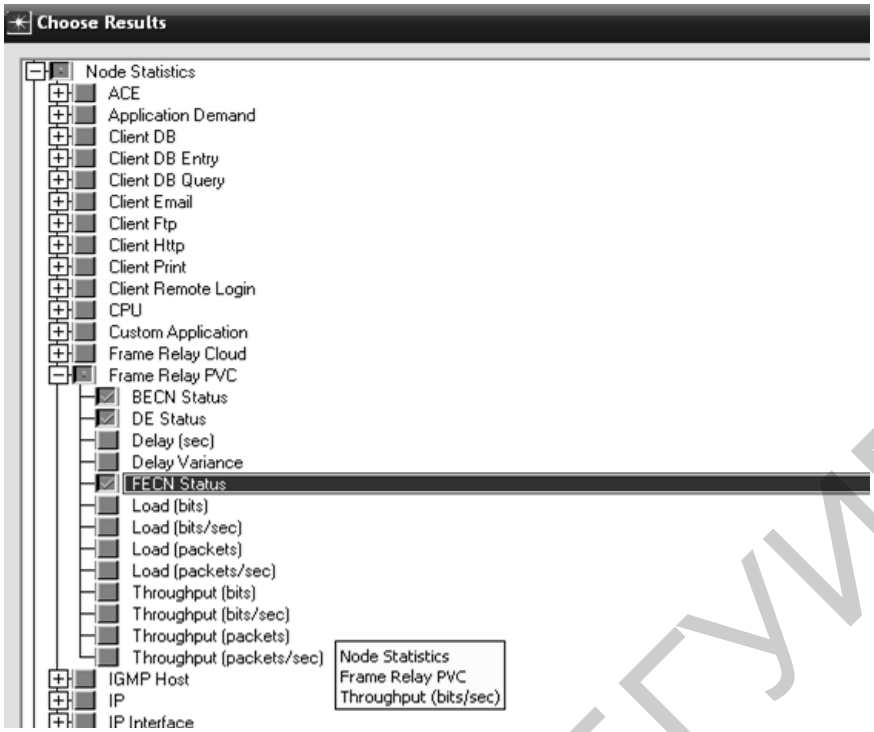


Рисунок 7.29 – Выбор данных статистики **Frame Relay PVC**

Теперь необходимо дублировать сценарий и изменить параметры постоянного виртуального канала PVC так, чтобы он поддерживал более высокую скорость передачи данных. Это позволит нам сравнивать эксплуатационные показатели сети при использовании двух различных **PVC**.

Для этого выбираем **Сценарии/Двойной Сценарий** и вводим имя нового сценария – **CIR128**.

Далее устанавливаем выходную скорость **CIR**, равную 128 000 бит/с. Устанавливаем согласованный объем пульсации **Bc** на выходе равным 128 000 бит. Дополнительный объем пульсации на выходе оставляем неизменным – равным 32 000 бит.

Во время моделирования процессов, происходящих в построенной сети, в реальном времени на экране строится график активности (рисунок 7.30), состоящий из двух кривых: синяя отображает ситуацию в каждый момент времени (время откладывается по оси абсцисс), красная показывает среднее значение.

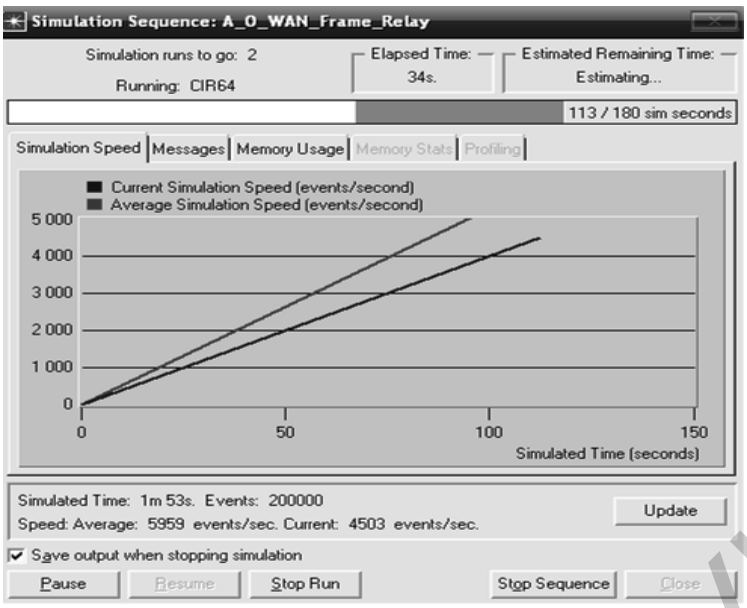


Рисунок 7.30 – График активности моделирования процессов

Для просмотра результатов моделирования работы сети выбираем пункт **View Result**. Для этого на рабочей области необходимо щелкнуть правой кнопкой мыши и в меню выбрать соответствующий пункт.

Характеристика времени загрузки (**Download Response Time**) показывает, как долго длилась каждая загрузка, и каждая из точек на графике соответствует загруженному файлу. Как мы видим из рисунка 7.31, загрузки занимали намного меньше времени в случае использования сети, построенной по сценарию **CIR128**, и колебания по времени загрузки в данном случае были намного меньше.

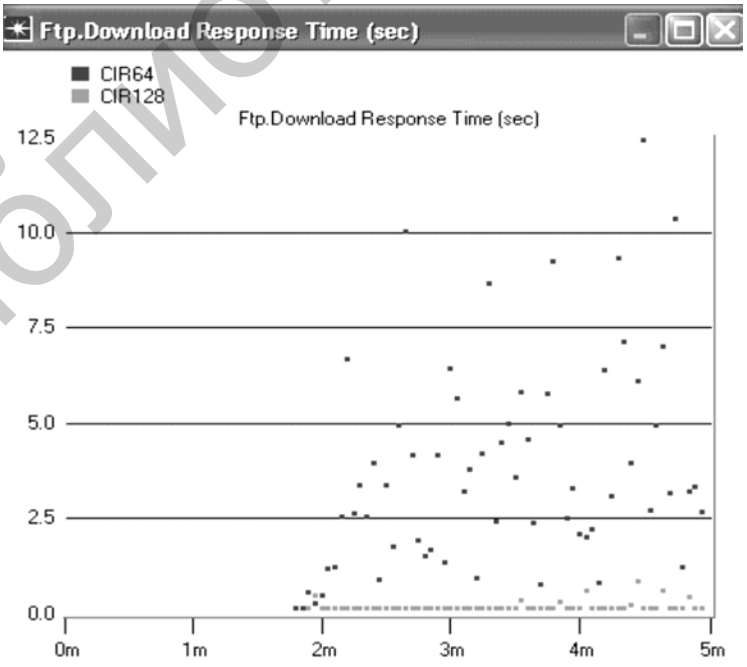


Рисунок 7.31 – Характеристика времени загрузки

На рисунке 7.32 показана статистика, определяющая общую сумму FTP трафика, полученного и сервером, и клиентом. Видно, что большее количество трафика получено в случае использования **CIR128**, а также имеется большее разнообразие трафика, чем в случае использования сценария **CIR64**.

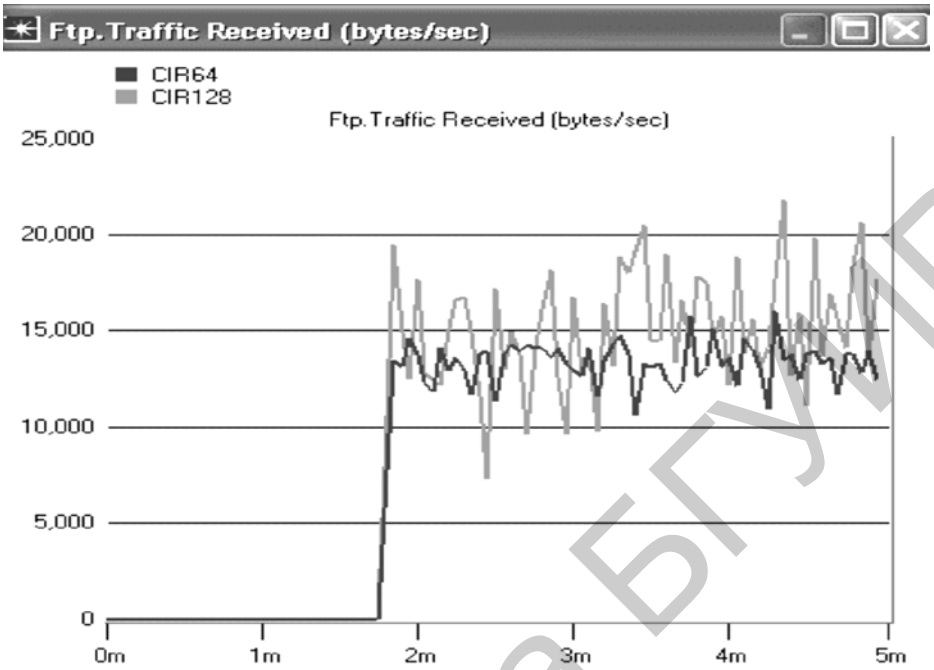


Рисунок 7.32 – **FTP**-трафик, полученный клиентом и сервером

7.3 Протокол RIP

Краткие теоретические сведения. Протокол маршрутной информации (**Routing Information Protocol, RIP**) – внутренний протокол маршрутизации дистанционно-векторного типа. Это один из наиболее ранних протоколов обмена маршрутной информацией, до сих пор широко распространенный ввиду простоты реализации.

Наиболее существенными характеристиками RIP являются:

– дистанционно-векторный тип принятия решения об оптимальном маршруте распространения информации в сети;

– использование количества переходов в качестве метрики при выборе маршрута;

– максимальное количество переходов, при котором возможна передача пакета, равно 15;

– стандартные обновления маршрутизации (routing updates) рассылаются широковещательным способом каждые 30 с.

Сегодня протокол **RIP** представлен тремя версиями:

1) **RIP v.1**;
2) **RIP v.2**;
3) **RIPng** (был разработан для маршрутизации в среде **IPv6**).

Особенностью **RIP v.1** является то, что он не распознает выделенные подсети – маска подсети не передается по каждому маршруту. Протокол считает, что все адреса принадлежат к стандартным классам A, B или C.

В 1998 г. в документе **RFC 2453** был первоначально определен протокол маршрутной информации версии 2 как расширение **RIP v.1**. Усовершенствования **RIP v.2** включают в себя:

– способность переносить дополнительную информацию о маршрутизации пакетов;

– механизм аутентификации для обеспечения безопасного обновления таблиц маршрутизации;

– способность поддерживать маски подсетей.

Однако данный протокол не учитывает качества каналов связи – с его точки зрения все каналы идентичны.

RIP представляет собой протокол маршрутизации на основе алгоритма Беллмана – Форда (или вектора расстояния – **Distance Vector** (**DV**)). В отличие от алгоритма, основанного на состоянии линий и использующего глобальную информацию, дистанционно-векторный алгоритм является распределенным, итерационным и асинхронным. Он является распределенным, т. к. каждый узел получает порцию информации от одного или нескольких напрямую соединенных с ним соседей, выполняет вычисления, а затем может разослать результаты своих вычислений своим соседям. Он является итерационным, т. к. этот процесс продолжается до тех пор, пока соседние узлы не перестанут обмениваться информацией. Алгоритм является асинхронным, т. к. он не требует, чтобы все узлы работали в жесткой взаимосвязи друг с другом. Дистанционно-векторный алгоритм **RIP v.2** требует обмена сообщениями только между напрямую соединенными узлами на каждой итерации.

Целью работы **RIP v.2** является создание таблиц маршрутизации, которые хранят данные об оптимальных маршрутах обмена информацией. В качестве расстояния до сети стандарты протокола **RIP v.2** разрешают использовать различные виды метрик: число транзитных узлов, пропускную способность, вносимые задержки и надежность сетей, а также любые их комбинации. Метрика должна обладать свойством аддитивности – метрика составного пути должна быть равна сумме метрик составляющих этого пути. В большинстве реализаций RIP применяется простейшая метрика – количество транзитных узлов, т. е. промежуточных маршрутизаторов, которые пакету нужно преодолеть для достижения сети назначения.

Конфигурирование сети. Для построения сети, конфигурация которой представлена на рисунке 7.33, необходимы 9 маршрутизаторов **ethernet4_slip8_gtwy**. Данный маршрутизатор имеет 4 **Ethernet** интерфейса и 8 **SLIP** интерфейсов.

Из палитры объектов (**Object Palette**) перенесите в проект 9 маршрутизаторов **ethernet4_slip8_gtwy**. Убедитесь, что расположение и имена маршрутиза-

торов соответствуют рисунку 7.33. Для того чтобы поменять имя объекта, щелкните на нем правой кнопкой мыши, в выпадающем меню нажмите **Edit Attributes/name**.

Соедините маршрутизаторы в сеть с помощью PPP_DS1 линий связи с палитры объектов. PPP_DS1 линии связи обеспечивают скорость передачи данных 1,544 Мбит/с.

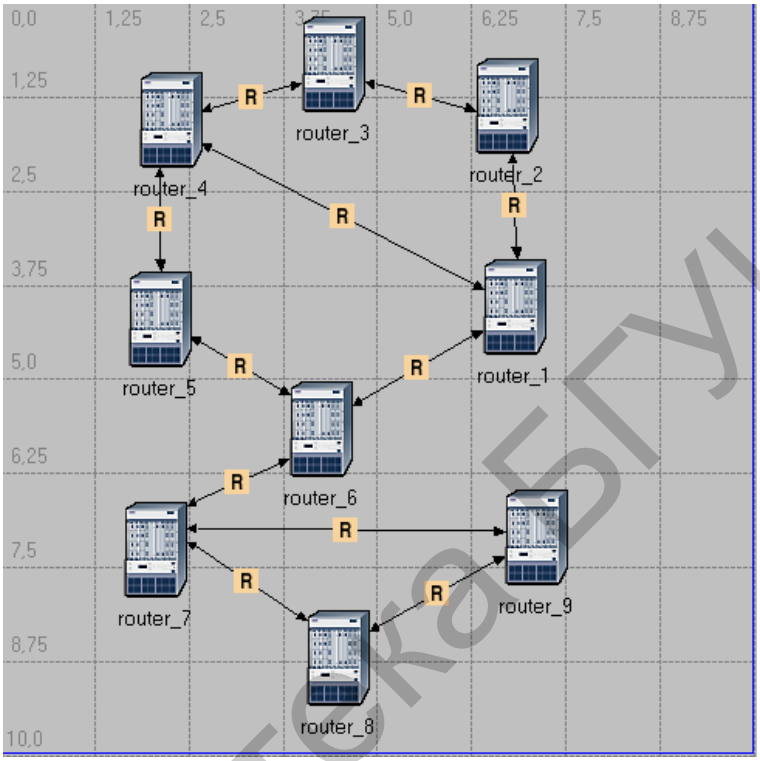


Рисунок 7.33 – Конфигурация сети

Сценарий 1: настройка сети

Настройка маршрутизаторов (рисунок 7.34)

Для установки параметров одновременно на несколько одинаковых объектов необходимо выделить один из них правой кнопкой мыши, выбрать **Select Similar Nodes/Edit Attributes**, отметить поле **Apply Changes to Selected Objects** и настроить параметры (рисунок 7.35):

1) **IP Routing Parameters/Routing Table Export/Enable**, убедитесь, что в поле **Export Time(s) Specification** установлено **Once at End of Simulation**. Данные настройки обеспечат просмотр таблиц маршрутизации после имитационного моделирования;

2) **RIP Parameters/Version**, отметьте версию протокола: **Version 2**;

3) в **RIP Parameters** установите значение **Stop Time** равным 1000 seconds. Этого времени будет достаточно для конвергенции сети;

4) в **RIP Parameters/Timers** обратите внимание на время между обновлениями (**Update Interval**), которое равно 30 с. Это значит, что каждые 30 с происходит обмен маршрутной информацией.

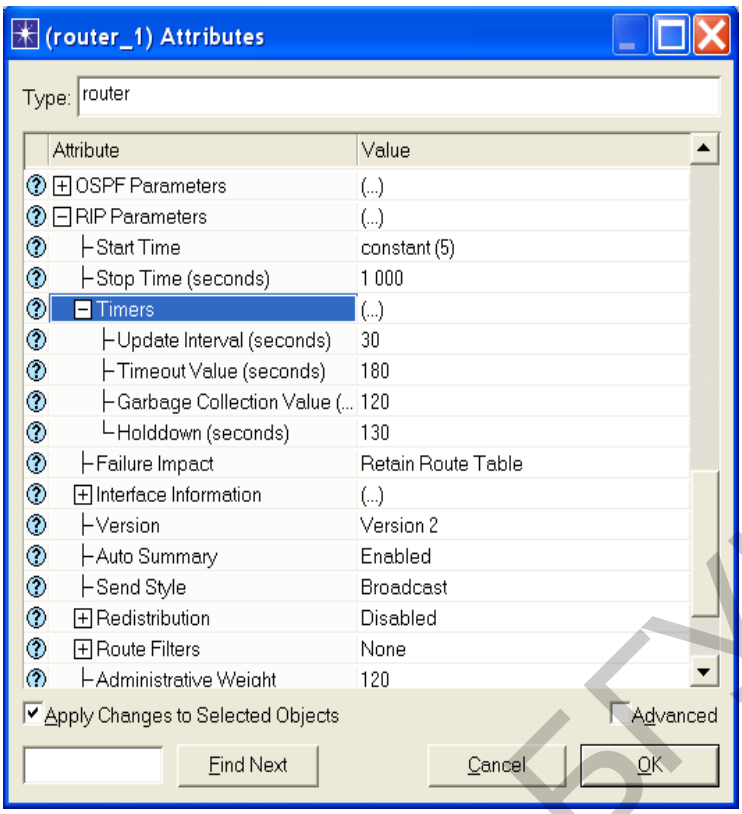


Рисунок 7.34 – Настройка маршрутизаторов

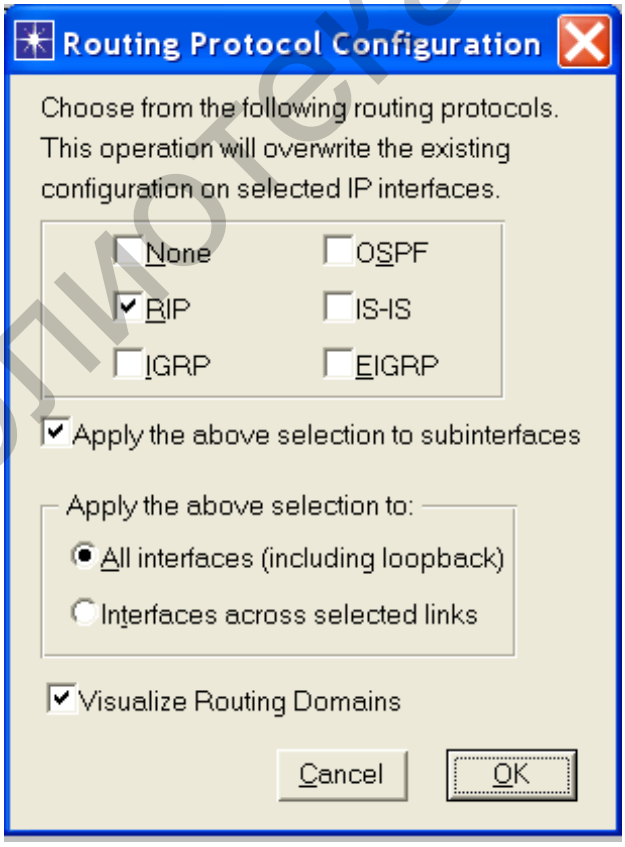


Рисунок 7.35 – Настройка протоколов

Конфигурация протоколов

Protocols/IP/Routing/Configure Routing Protocols, выберите настройки согласно рисунку 7.35.

Сохраните проект **File/Save**.

Сценарий 2: настройка сети

Создайте сценарий 2, в котором линия связи между маршрутизаторами 1 и 6 будет обрываться через 300 с после начала моделирования.

Скопируйте сценарий 1: **Scenario/Duplicate Scenario/scenario2/OK**. Быстрый переход между сценариями осуществляется с помощью сочетания клавиш **Ctrl +** «номер сценария».

Выберите **Object Palette/utilities**, поместите объект **Failure Recovery** в проект (рисунок 7.36).

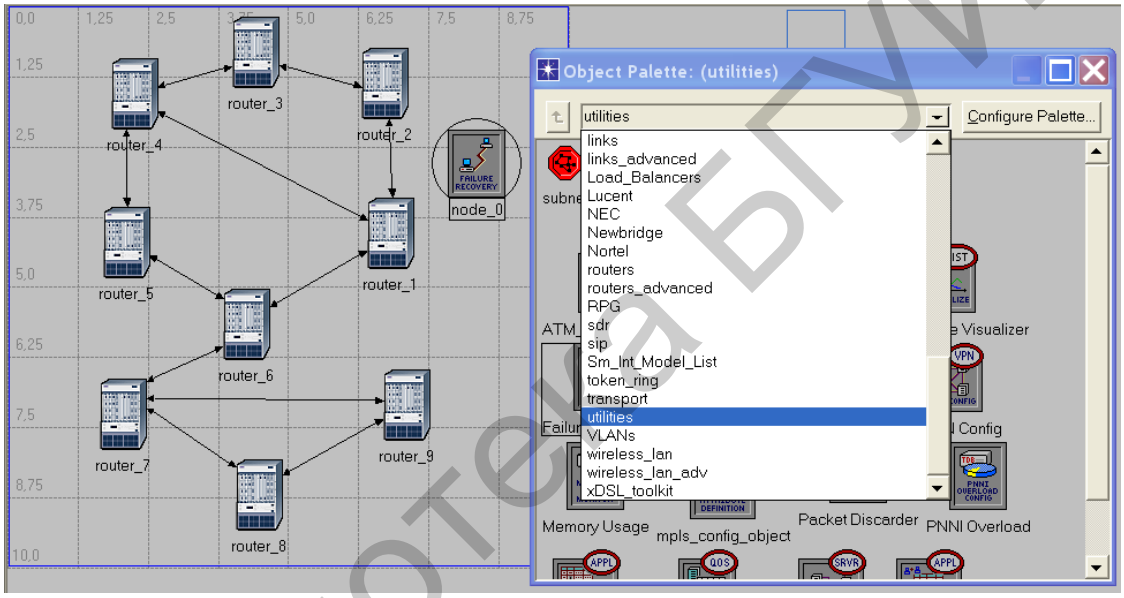


Рисунок 7.36 – Вкладка **utilities** на палитре объектов

Настройте параметры объекта **Failure Recovery**. Для этого необходимо правой кнопкой мыши выделить **Failure Recovery/Edit Attributes/Link Failure/Recovery Specification**, значение поля **row** выбрать равным **1**, в **Row 0** из выпадающего меню **Name** выбрать **CampusNetwork_router_6 <-> router_1**, установить 300 в поле **Time** (рисунок 7.37).

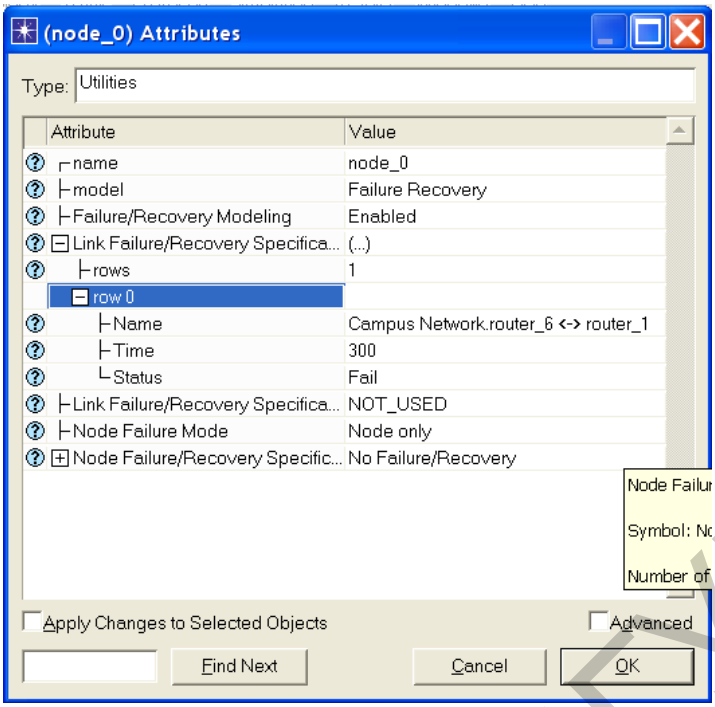


Рисунок 7.37 – Вкладка **Utilities**

Сценарий 3: настройка сети

Настройте сконфигурированную сеть таким образом, чтобы для обмена маршрутной информацией она использовала **RIP** версии 1 (рисунок 7.38).

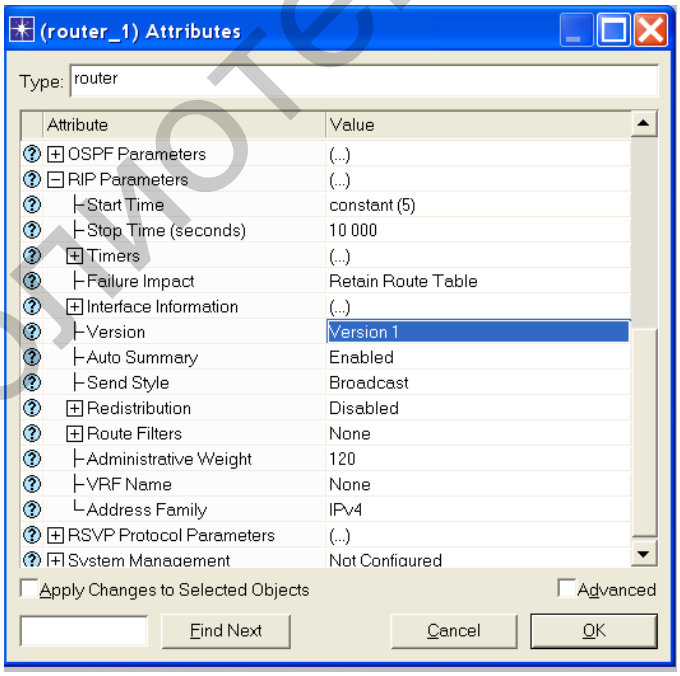


Рисунок 7.38 – Настройка первой версии **RIP**

Сценарий 4: настройка сети

Выполните пункт 4 из сценария 1 для случая обрыва двух линий связи: **router_6 <-> router_1** и **router_5 <-> router_6**. Время обрыва взять из пункта 4 сценария 1.

Настройка параметров имитационного моделирования

Для настройки параметров моделирования на главной панели нажмите кнопку **Configure/Run Simulation**. В появившемся окне на вкладке **Common** задайте длительность процесса моделирования равным **10 minute(s)** (рисунок 7.39), а на вкладке **Global Attributes** установите следующие атрибуты (рисунок 7.40):

– **IP Dynamic Routing Protocol/RIP**;
– **IP Routing Interface Addressing Mode/Auto Addressed/Export**;
– **IP Routing Table Export/Import/Export**;
– **RIP Sim Efficiency/Disabled**;
– **RIP Stop Time/1000**.

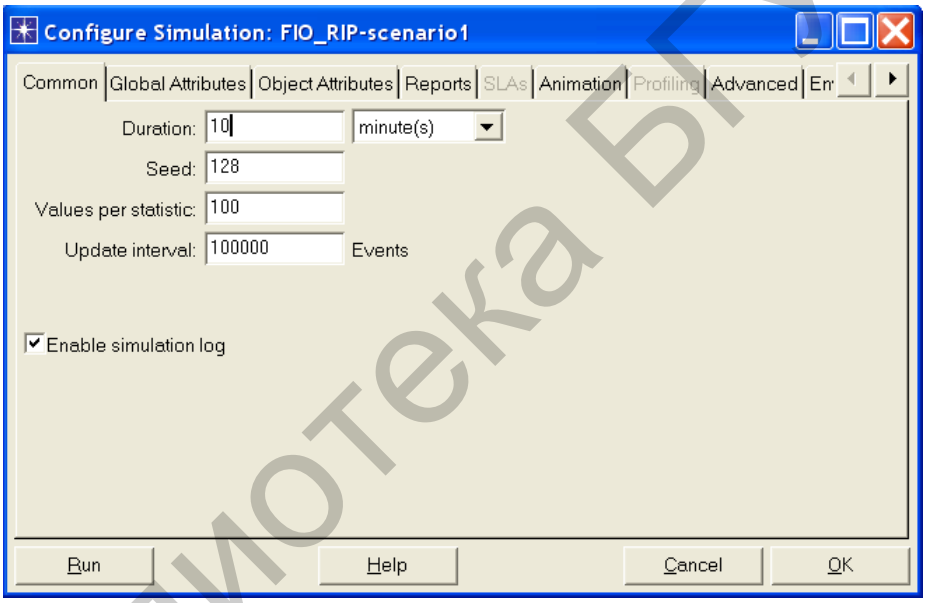


Рисунок 7.39 – Настройка длительности процесса моделирования

Согласно данным параметрам после имитационного моделирования адресные таблицы и таблицы маршрутизации всех узлов сети будут записаны в отдельные файлы. Блокировка режима **RIP Simulation Efficiency** (производительность **RIP**-моделирования) указывает на то, что RIP-пакеты будут передаваться на протяжении всего времени моделирования. Установленного в поле **RIP Stop Time** времени будет достаточно для конвергенции сети (см. рисунок 7.40).

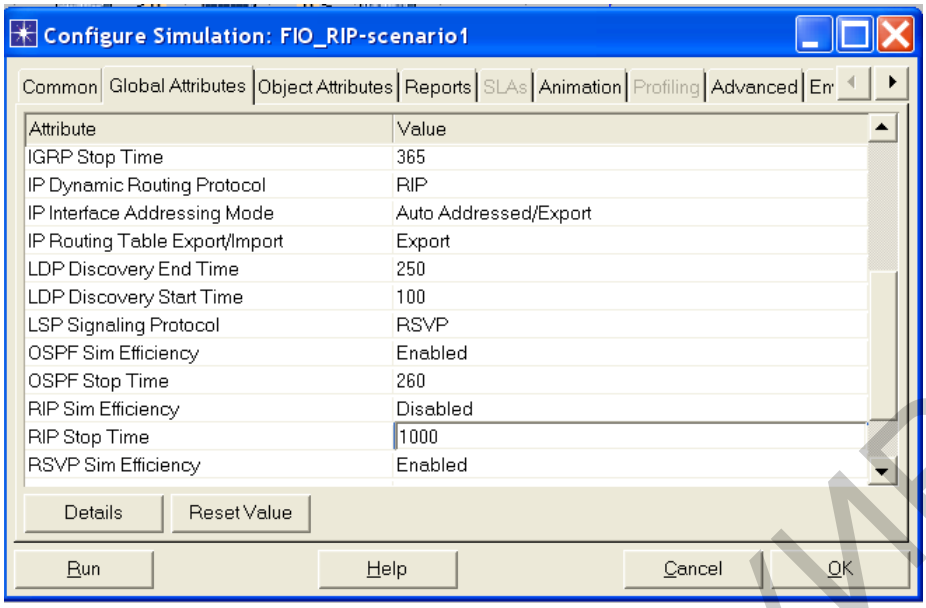


Рисунок 7.40 – Настройка общих параметров моделирования

Нажмите **OK** для сохранения параметров.

Настройте процесс моделирования для всех сценариев.

Имитационное моделирование и анализ полученных результатов. Рассмотрим влияние выхода из строя элемента сети на объем трафика, число обновлений таблиц маршрутизации и время их конвергенции.

Перед началом процесса моделирования в сценарии 1 и сценарии 2 укажите тип собираемой статистики, необходимой для изучения особенностей работы **RIP v.2**: вкладка **Simulation/Choose Individual Statistics**, отметьте следующие поля:

1) число обновлений таблиц маршрутизации каждого из узлов в отдельности: **Nodes Statistics/Route Table/Total Number of Updates**.

2) исходящий трафик всех узлов сети, использующих **RIP**: **Global Statistics/RIP/Traffic Sent (bits/sec)**.

Для получения подробного графика выделите название статистики правой кнопкой мыши, нажмите вкладку **Change Collection Mode** (изменить метод накопления данных), убедитесь, что поле **Advanced** отмечено галочкой, а поле **Capture Mode** установлено в режим **all values** (все значения);

3) исходящий трафик каждого из узлов в отдельности: **Nodes Statistics/RIP/Traffic Sent (bits/sec)**. Измените метод накопления данных;

4) объем трафика, переданного по каждому из каналов связи: **Link Statistics/point-to-point/throughput (bits/sec)** (рисунок 7.41).

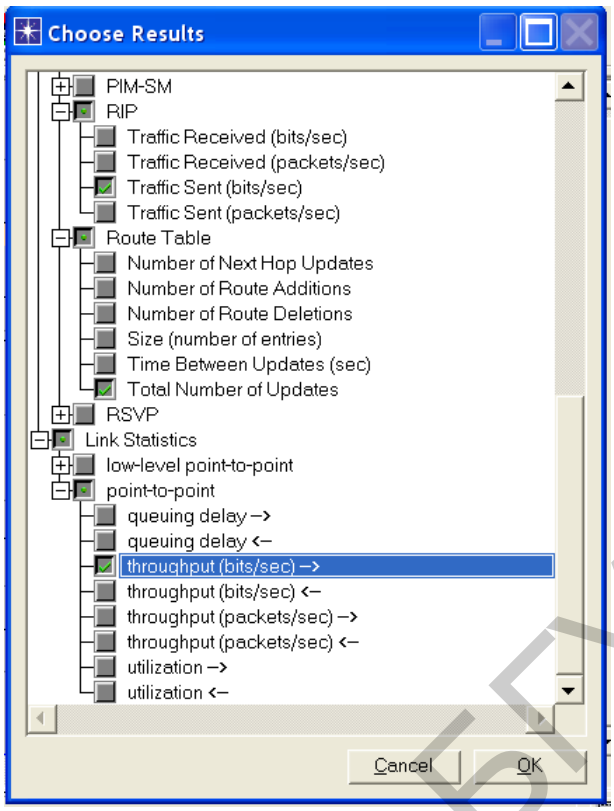


Рисунок 7.41 – Выбор статистики моделирования

Сохраните проект **File/Save**.

Запустите одновременно первый и второй сценарии: **Scenarios/Manage Scenarios**, в графе **Results** напротив сценариев 1 и 2 установите значение **<collect>** (или **<recollect>)/OK**, в окне **Simulation Sequence: FIO_RIP** после окончания процесса моделирования нажмите **Close** и сохраните проект (рисунки 7.42 и 7.43).

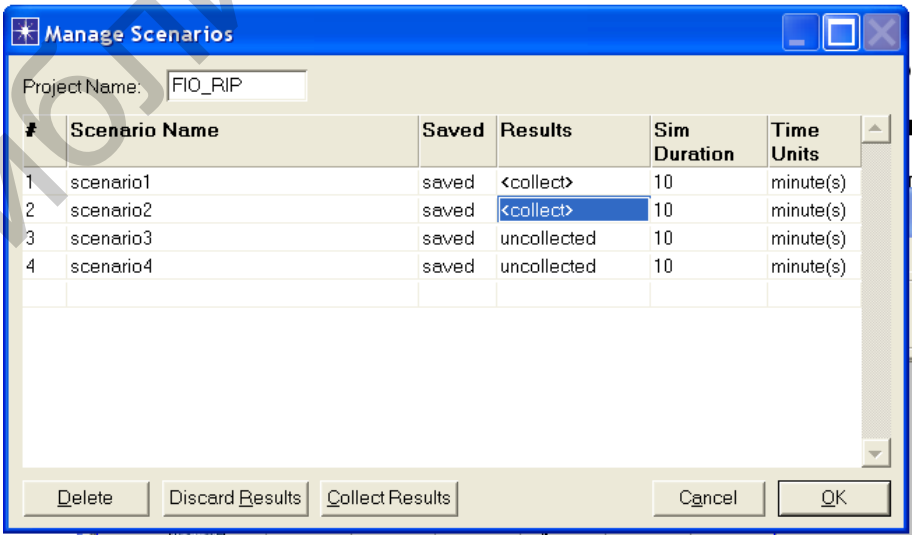


Рисунок 7.42 – Одновременный запуск сценариев

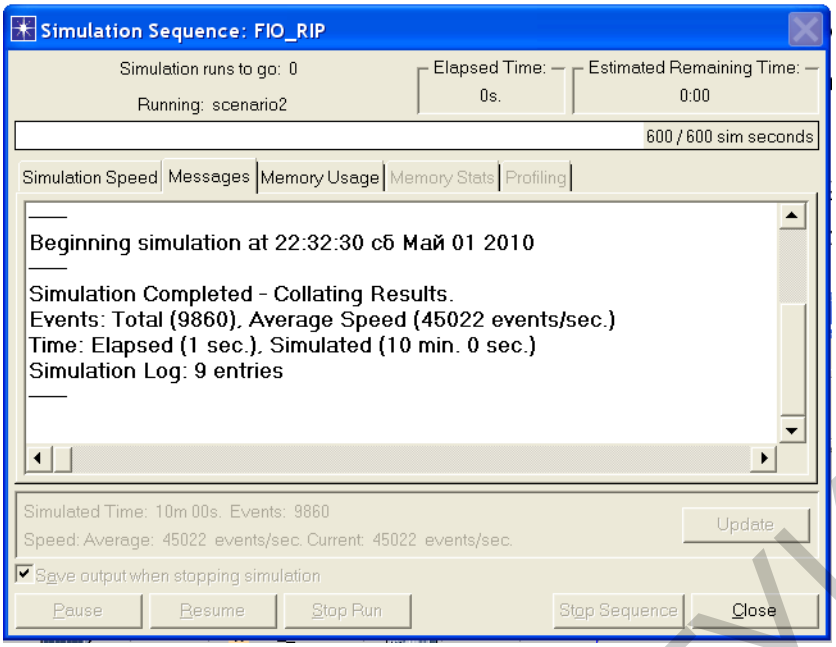


Рисунок 7.43 – Окно процесса моделирования

Для просмотра результатов имитационного моделирования зайдите на вкладку **Results/Compare Results**. В выпадающем меню правой нижней части окна **Compare Results** установите метод представления данных **Stacked Statistics**. При изучении графиков в качестве стиля представления информации рекомендуется выбирать гистограмму. Для этого правой кнопкой мыши щелкните на графике, выберите меню **Draw Style/Bar Chart** (рисунок 7.44).

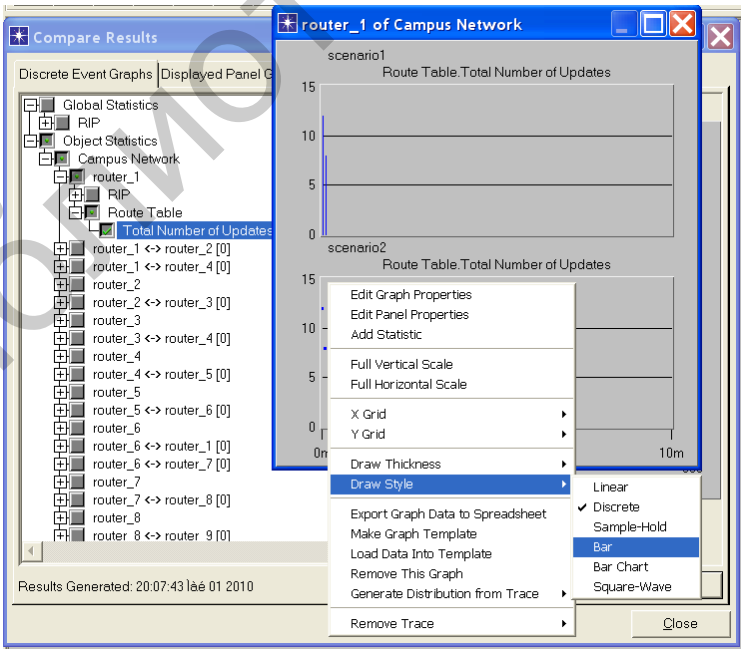


Рисунок 7.44 – Изменение вида графика

Далее необходимо проанализировать полученную в ходе имитации работы сети статистику:

– о числе обновлений таблиц маршрутизации маршрутизатора **router_1: Object Statistics/Campus Network/router_1/Route Table**, поставьте галочку около поля **Total Number of Updates**. Для просмотра графика нажмите **Show**;

– об объеме исходящего трафика маршрутизатора **router_1: Object Statistics/Campus Network/router_1RIP/Traffic Sent/Show**;

– об объеме трафика, проходящего по линии связи **router_1 <-> router_6: Object Statistics/Campus Network/router_6 <-> router_1/throughput/Show**;

– об объеме исходящего трафика всех узлов сети: **Global Statistics/RIP/Traffic Sent/Show** (рисунок 7.45).

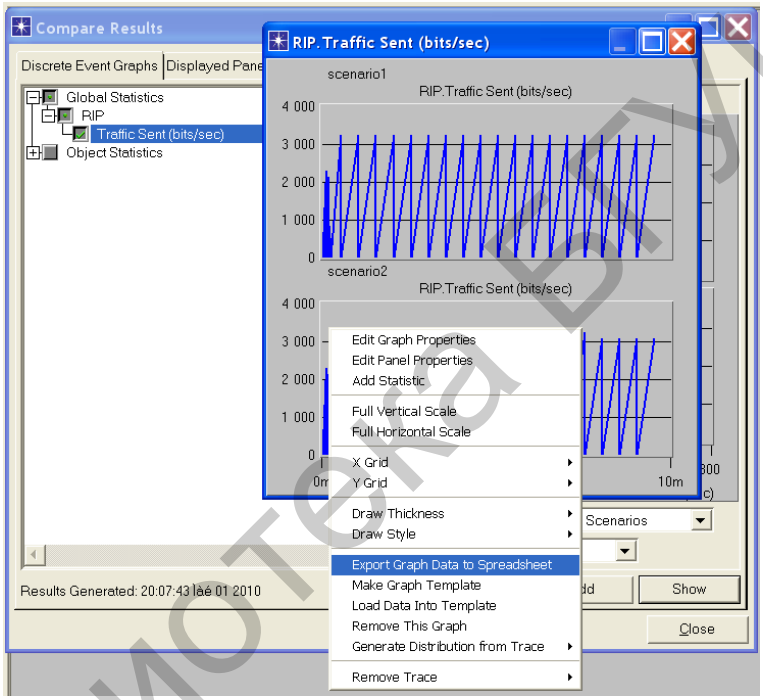


Рисунок 7.45 – Запись результатов моделирования в таблицу Microsoft Excel

Можно определить время конвергенции таблиц маршрутизации в обоих сценариях, используя статистику об объеме общего исходящего трафика в сети. Для получения точных данных результаты моделирования можно представить в виде таблицы Microsoft Excel: **Results/Compare Results/Global Statistics/RIP/Traffic Sent/Show**, правой кнопкой мыши откройте меню, а затем **Export Graph Data to SpreadSheet**.

Перед просмотром таблиц, содержащих адресную и маршрутную информацию, запустите сценарий **Configure/Run Simulation/Run**, после процесса имитационного моделирования зайдите в главное меню проекта **File/Model Files/Refresh Model Directories**. Далее через вкладку **File** откройте диалоговое окно **Open**, в выпадающем меню выделите поле **Generic Data File**, в основном меню

найдите необходимый файл, например, **FIO_RIP-scenario1-ip_addresses** (для адресной таблицы) или **FIO_RIP-scenario1-ip_routes** (для таблицы маршрутизации) (рисунок 7.46).

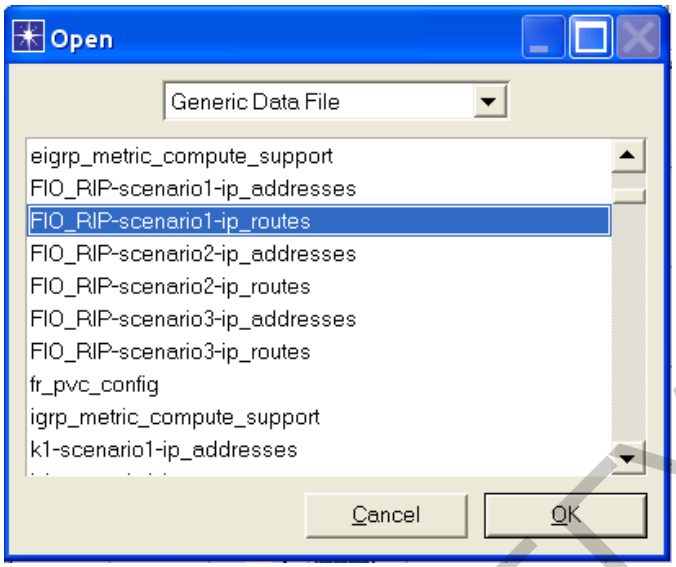


Рисунок 7.46 – Доступ к таблице маршрутизации

С помощью таблиц маршрутизации и адресных таблиц, полученных при моделировании сценария 1, определите путь между двумя маршрутизаторами. Номера маршрутизаторов указаны на рисунке 7.47, интерфейсы можно выбрать самостоятельно. Обратите внимание, что каждый маршрутизатор имеет **Loopback Interface** – IP-интерфейс, который используется для адресации узлом самого себя (loopback, интерфейс обратной связи). Обращение по адресу **Loopback Interface** означает связь с самим собой (без выхода пакетов данных на уровень доступа к сети), что удобно использовать, например, для тестирования сетевого ПО (рисунки 7.48, 7.49).

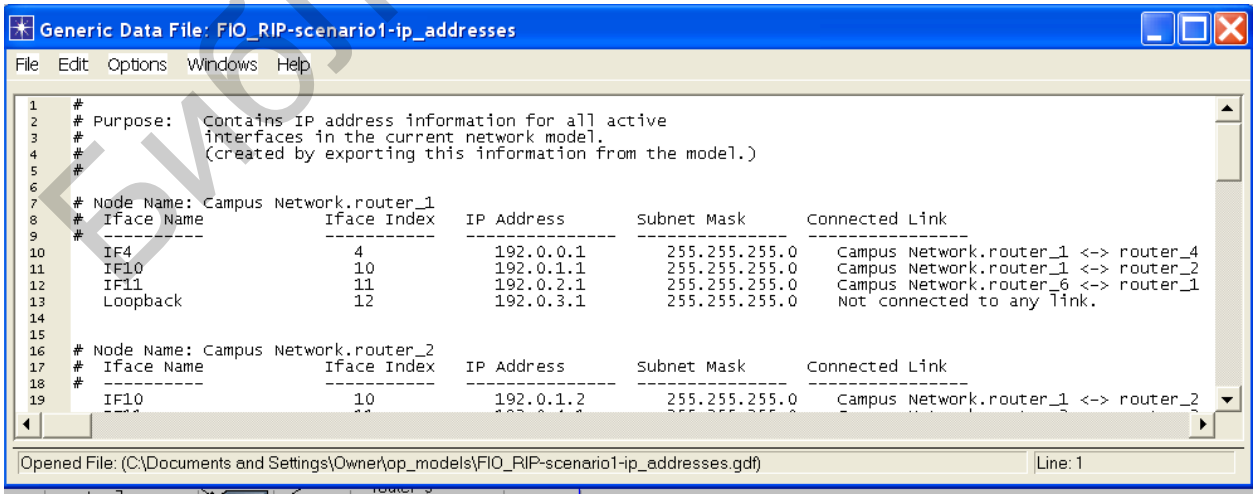


Рисунок 7.47 – Адресная таблица

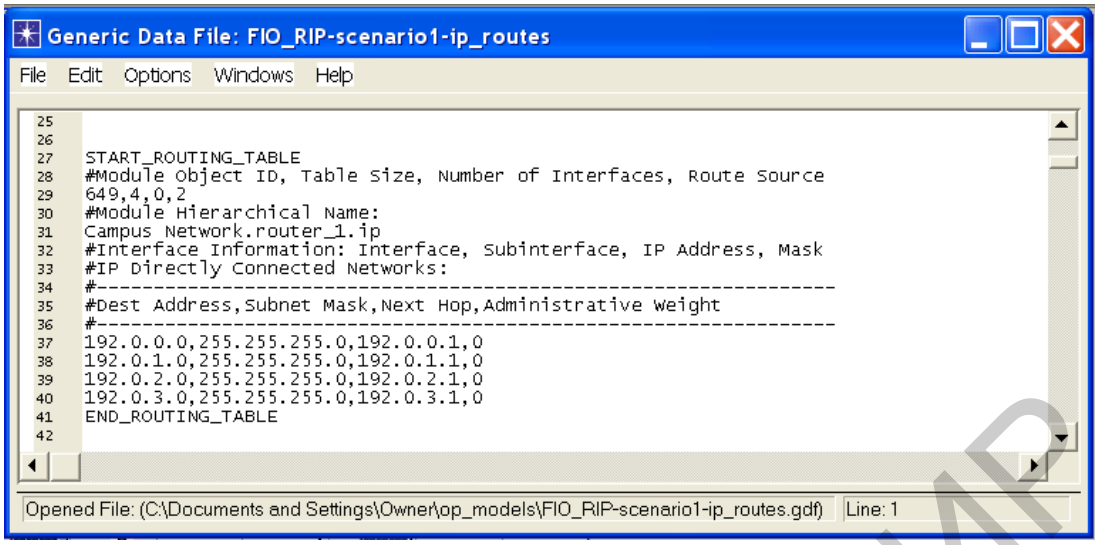


Рисунок 7.48 – Таблица маршрутизации: часть 1

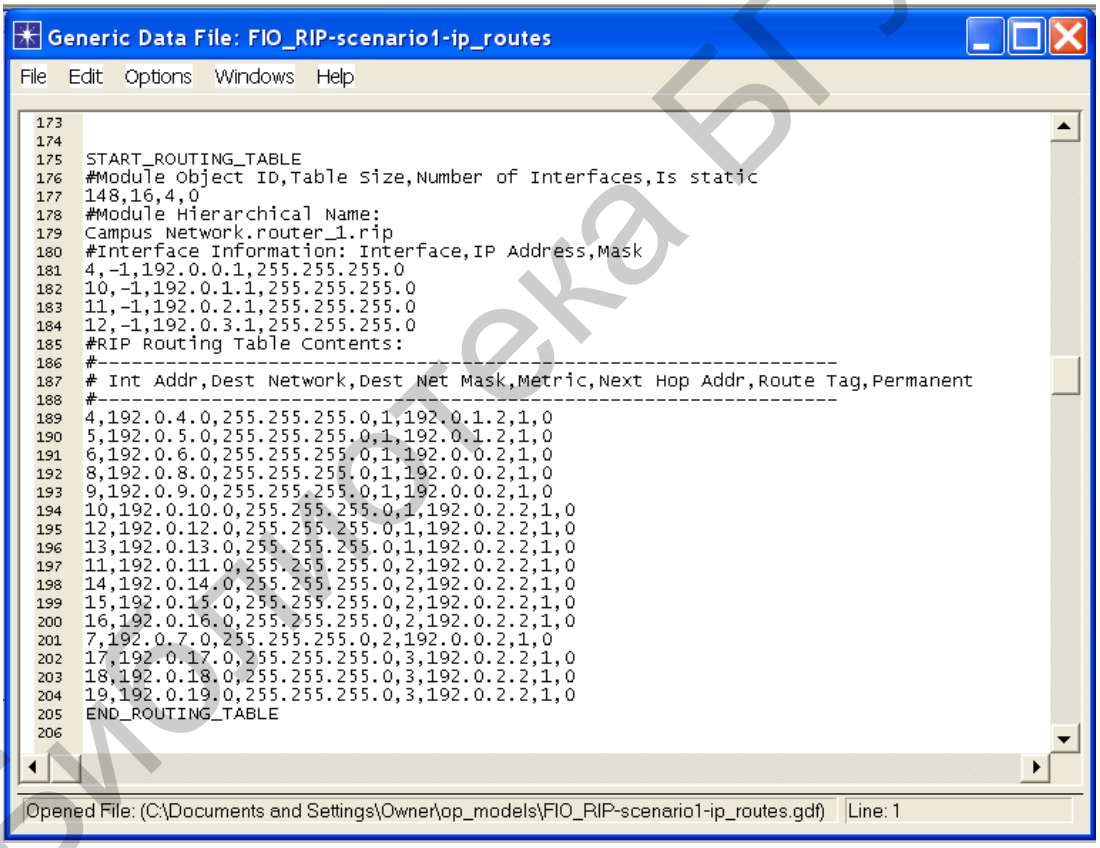


Рисунок 7.49 – Таблица маршрутизации: часть 2

Запустите сценарий 3. В таблицах маршрутизации найдите графу **Destination Network Mask**. Обратите внимание на ее значение.

По таблицам адресной и маршрутной информации сценария 4 определите путь, который проходит пакет от маршрутизатора **router_3** к **router_7**.

7.4 Влияние размера пакета на пропускную способность и задержку в локальной сети

Смоделируем работу распределенной локальной сети, построенной по технологии 10 Мбит/с **Ethernet**.

Локальные вычислительные сети (ЛВС) на основе **Ethernet** используют технологию **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection – множественный доступ с контролем несущей и обнаружением коллизий). Данная технология используется для множественного доступа к общей передающей среде в локальной компьютерной сети с контролем коллизий. **CSMA/CD** относится к децентрализованным случайным (точнее, квазислучайным) методам. Он используется как в обычных сетях типа **Ethernet**, так и в высокоскоростных сетях (**Fast Ethernet**, **Gigabit Ethernet**).

Так же называют сетевой протокол, в котором используется схема **CSMA/CD**. Протокол **CSMA/CD** работает на канальном уровне в модели **OSI**.

Характеристики и области применения этих популярных на практике сетей связаны именно с особенностями используемого метода доступа. **CSMA/CD** является модификацией «чистого» (**CSMA**).

Использование технологии **CSMA/CD** дает возможность компьютеру обнаружить, свободна ли линия перед отправкой пакета. Коллизии (столкновения) могут произойти в данной среде передачи в том случае, если больше чем один хост (компьютер) определит, что линия свободна, и затем попытается отправить пакет. Коллизии определенно будут негативно сказываться на пропускной способности канала передачи данных, что приведет к низкой производительности всей сетевой системы.

Размер пакета также влияет на пропускную способность канала передачи данных. Пакеты малого размера, как правило, имеют меньшие задержки в передаче, но могут с легкостью нагрузить канал передачи данных. Это связано с тем, что для минимального размера кадра в 64 байта всего 46 байт отводится на «полезную» информацию, остальные 20 байт выделены на служебную информацию в виде заголовков и поля **CRC**.

С другой стороны, большие пакеты данных, несмотря на более длительные задержки в передаче, могут быть более эффективными в плане «переноса» данных, т. к. отношение полезной информации к служебной будет в разы больше.

Попробуем найти так называемый «оптимальный баланс» между размером пакета и задержками при передаче данных. Таким образом, мы исследуем связь между размерами пакета, пропускной способностью и задержками в среде передачи **10BaseT Ethernet LAN** (10 Мбит/с).

Присвойте название будущему проекту, например **ваша_фамилияEthPkt**. Также необходимо дать название вашему сценарию, лучше укажите **Pkt512** (это поможет в будущем лучше ориентироваться в результатах моделирования, но оно может быть и произвольным). После чего нажмите **ОК**. Затем вы должны увидеть диалоговое окно **Startup Wizard Initial Topology**.

Проверьте, что вами выбран пункт **Create Empty Scenario** «Создать пустой сценарий» в окне **Initial Topology**. Нажмите кнопку **Next**. После чего вы увидите окно **Network Scale**.

В нем необходимо выбрать **Office** из списка. Затем нужно задать параметры будущего офиса. В поле **Size** укажите **Meters** («Метры»), задайте значения **200** для **X span** и **Y span** соответственно. Затем нажмите **Next**. После чего появится окно **Select Technologies** («Окно выбора технологий»).

В окне выбора технологий найдите поле **ethernet** в колонке **Model Family**. Напротив этого поля, но уже в колонке **Include?** выберите **Yes**. Тем самым вы добавите данную технологию в свой будущий проект. После чего нажмите кнопку **Next** (рисунок 7.50)

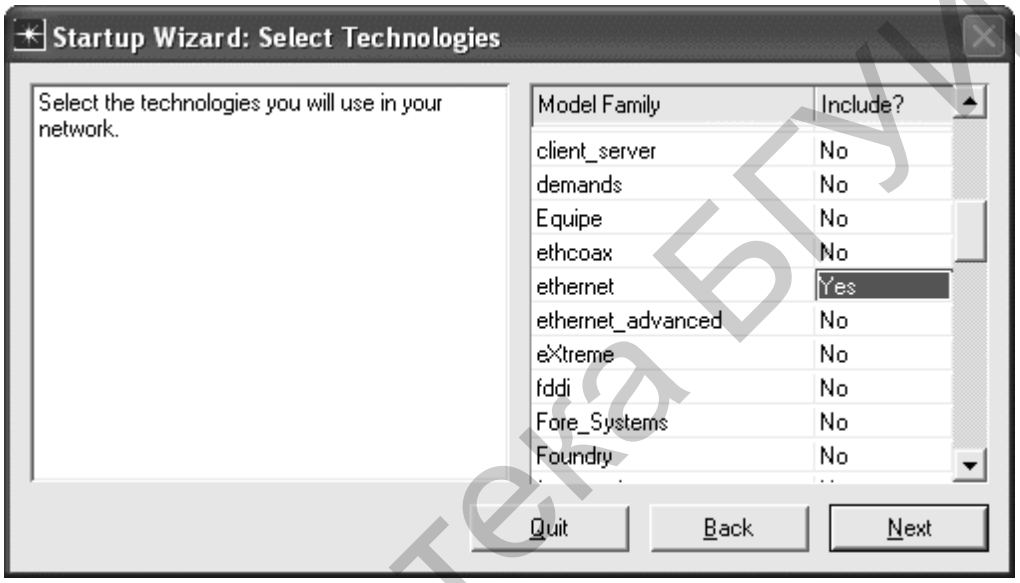


Рисунок 7.50 – Выбор технологии **Ethernet**

В окне **Startup Wizard Review** появится вся суммарная информация, которую мы выбирали и настраивали для нашего проекта. Убедимся, что в графе **Scale** у нас стоит значение **Office**, в поле **Size** стоят размеры **200 x 200 м**, в окне **Model Family** стоит значение **Ethernet**. Если какие-то параметры были настроены неверно, так, как на рисунке 7.50, то нажмите кнопку **Back** необходимое количество раз и исправьте несоответствия.

Если все верно, нажимаем кнопку **ОК**. После чего появится наша рабочая область и окно **Object Palette**.

Создание сети на основе технологии Ethernet. Теперь приступим к созданию самой сети. Основу нашей сети будет составлять hub (10 Мбит/с). Конечными устройствами будут выступать хосты (компьютеры), их нам потребуется 16.

Для того чтобы все параметры добавить на рабочую поверхность, необходимо из главного меню выбрать **Topology/Rapid Configuration**. В появившемся окне необходимо выбрать топологию сети из выпадающего списка, нам понадобится топология типа **Star** («Звезда»). Нажимаем **ОК** (рисунок 7.51).

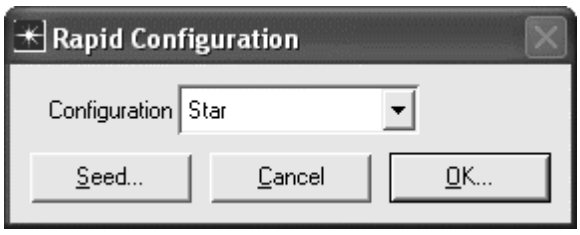


Рисунок 7.51 – Выбор конфигурации сети

В следующем окне мы сконфигурируем наши хосты и концентратор. В поле **Center Node Model** выбираем **ethernet16_hub** (из выпадающего списка). В следующем поле **Periphery Node Model** выбираем **ethernet_station**. Так как нам необходимо 16 хостов, меняем поле **Numbers** на **16**. Меняем поле **Link Model**, выбирая значение **10BaseT**.

После чего остался последний шаг – расположить в центре рабочей области нашу будущую сеть. Для этого выставляем значения **X** и **Y** – **100**, а значение **Radius** – **50**. Конфигурация закончена, нажимаем **ОК** (рисунок 7.52).

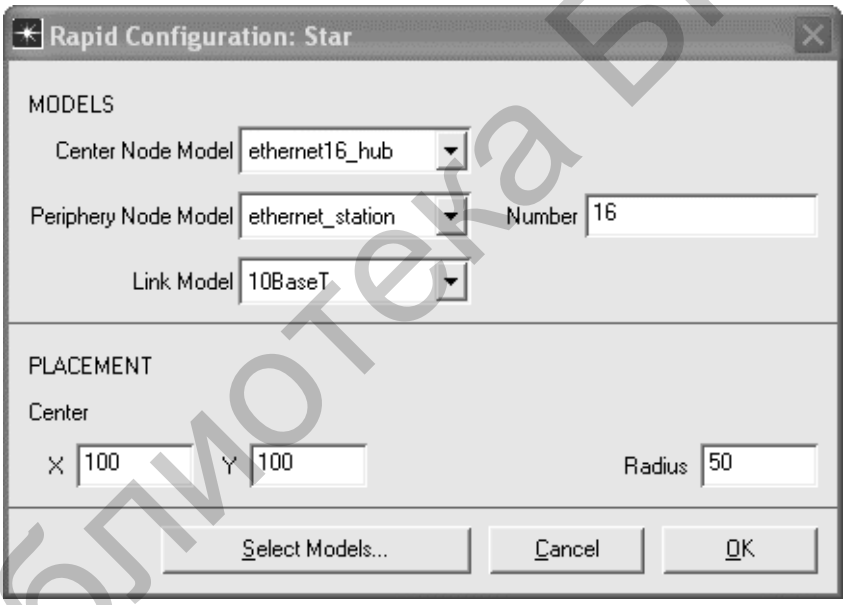


Рисунок 7.52 – Задание параметров сети

Теперь переименуем наш концентратор. Для этого щелкнем на нем правой кнопкой мыши, выберем из меню **Set Name** и зададим новое имя – **Hub10**. После этих действий проект должен выглядеть, как показано на рисунке 7.53.

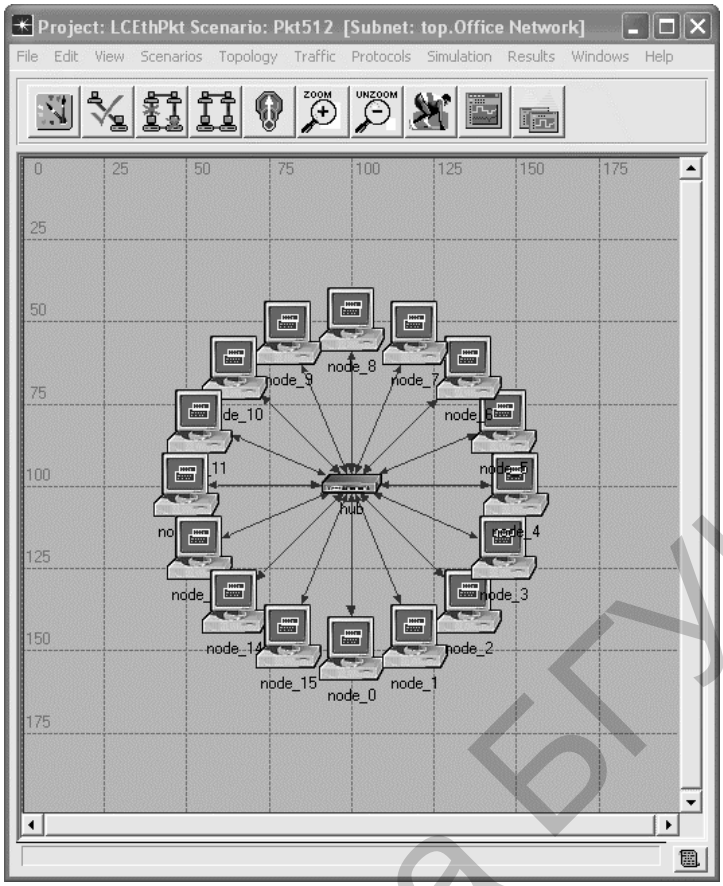


Рисунок 7.53 – Конфигурация сети

Теперь приступим к конфигурации хостов.

Для этого выберем любой из наших хостов, после чего вызовем контекстное меню, щелкнув правой кнопкой клавиши мыши, где найдем пункт **Select Similar Nodes** («Выделить все похожие хосты»). В данном случае для удобства настройки необходимо выделить все 16 хостов, чтобы сконфигурировать один и присвоить такие же значения всем.

После выделения всех хостов необходимо щелкнуть правой кнопкой мыши на любом из них и из меню выбрать пункт **Edit attributes**.

Перед нами окно конфигурирования (рисунок 7.54). Для начала необходимо поставить галочку напротив пункта **Apply Changes to Selected Objects** (внизу окна). Это важный момент, т. к. нам не потребуется настраивать каждый хост отдельно.

Нажмите знак «+» напротив пункта **Traffic Generation Parameters**, для того чтобы открыть вложенные параметры. Мы увидим список настроек, как показано на рисунке 7.54.

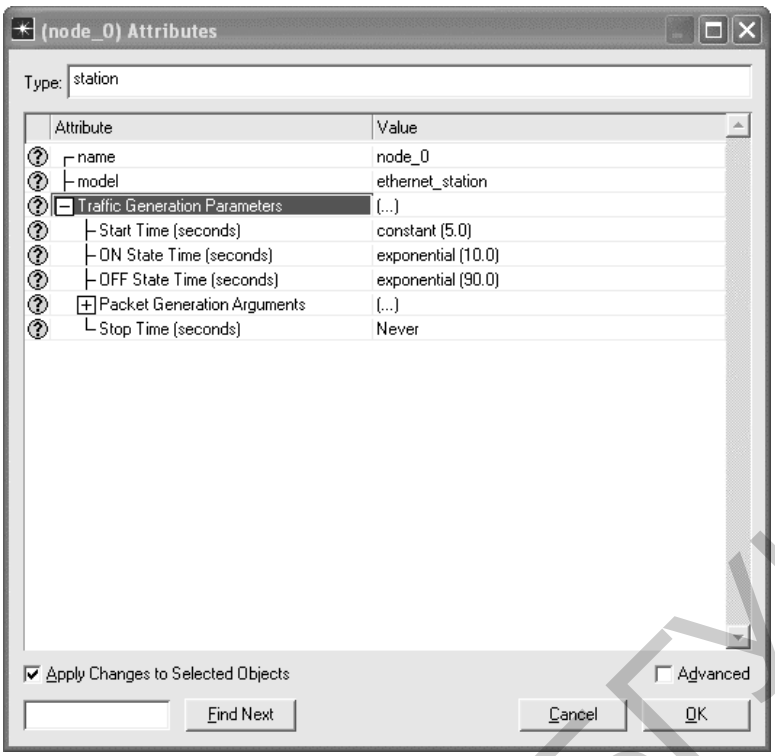


Рисунок 7.54 – Задание параметров трафика

Сконфигурируем время начала моделирования. Его необходимо настроить так, чтобы моделирование началось сразу после запуска. Для этого нажмем на строку напротив параметра **Start Time (seconds)**. В появившемся окне зададим следующие настройки: полю **Mean Outcome** присвоим значение **1**, остальные поля оставим по умолчанию. Нажмем **ОК**. Это будет означать, что в нашем моделировании трафик начнет «идти» через 1 с после запуска (рисунок 7.55).

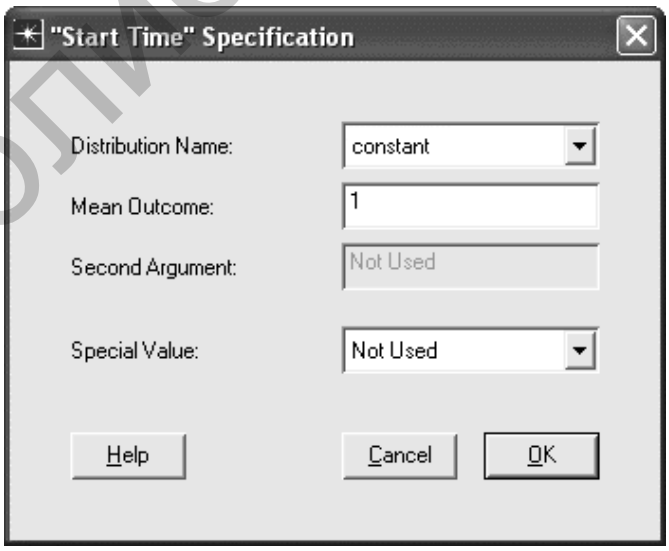


Рисунок 7.55 – Задание времени начала моделирования

Далее настроим параметр **ON State Time** точно так же, как и в пункте 3, вызвав настройки. Необходимо задать значение **Mean Outcome** в **100**, остальные поля оставим по умолчанию. Нажмем **ОК** (рисунок 7.56).

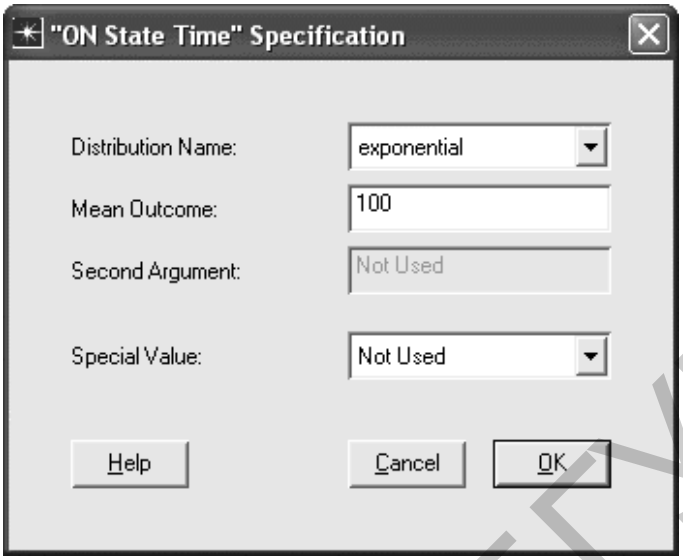


Рисунок 7.56 – Задание параметра **Mean Outcome**

После чего необходимо настроить параметр **OFF State Time**, вызвав его настройки. Зададим значение **Mean Outcome** равным **0**, остальные поля оставим по умолчанию. Нажмем **ОК** (рисунок 7.57).

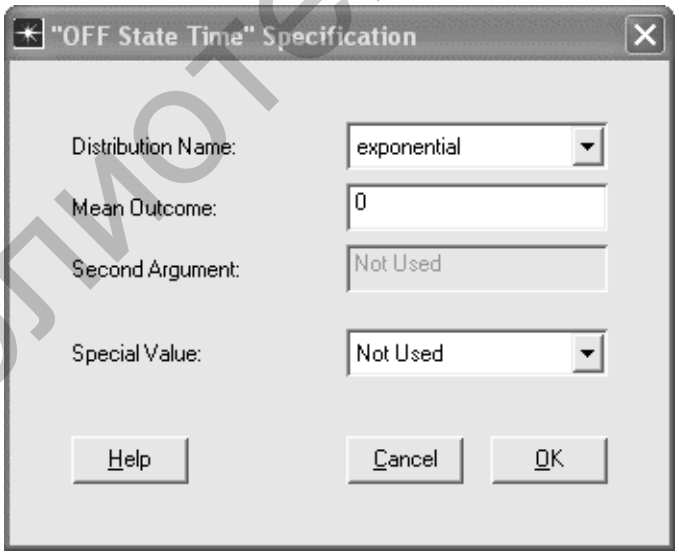


Рисунок 7.57 – Задание параметра **OFF State Time**

Данная настройка позволит нам генерировать трафик с нагрузкой в 100 % с самого начала моделирования и не снижать ее в течение всего необходимого времени моделирования.

Далее приступим к настройке параметров **Packet Generation Arguments**, нажав на знак «+» перед названием. В данных настройках нас интересует пункт **Interarrival Time (seconds)** – временной интервал между пакетами. Вызываем настройки этого пункта двойным щелчком мыши и изменяем значение **Mean outcome** с **1** на **.02**, как показано на рисунке 7.58 (это будет соответствовать интервалу в 0,02 с). После чего нажимаем **ОК**.

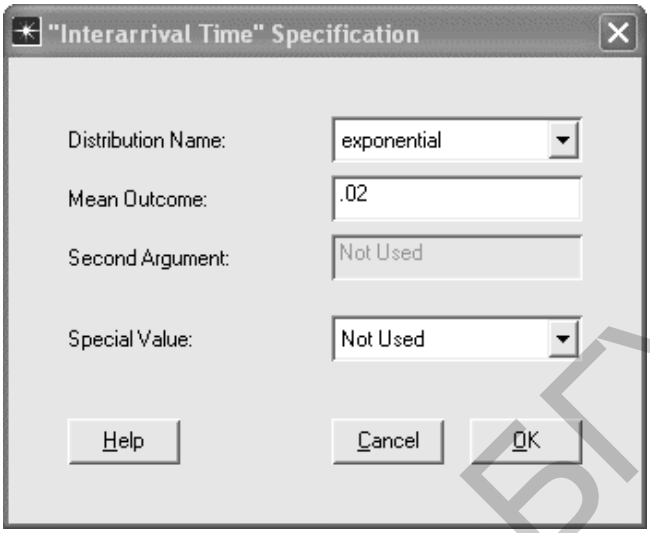


Рисунок 7.58 – Задание параметра **Interarrival Time**

Переходим к пункту **Packet Size (bytes)** – это непосредственно размер пакета. Вызвав его настройки, приступим к конфигурации. Нам необходимо задать размер пакета равным **512** таким образом, чтобы в процессе моделирования размер пакета не менялся, а был постоянным (если вы помните, в начале работы над проектом мы создали сценарий **Pkt512**, он имеет такое название специально для этих целей).

Для настройки пакета нам необходимо изменить параметр **Distribution Name** на **constant**, а параметру **Mean Outcome** присвоить значение 512 (рисунок 7.59). На этом настройка закончена, нажимаем **ОК**.

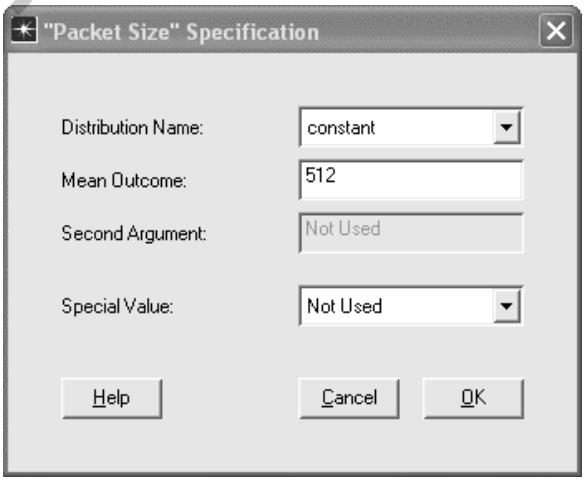


Рисунок 7.59 – Задание параметра **Distribution Name**

Проверьте, все ли параметры выставлены правильно, так, как показано на рисунке 7.60. Напротив параметра **Apply Changes to Selected Objects** (в нижней части окна) должна стоять галочка. После этого нажимаем **ОК**.

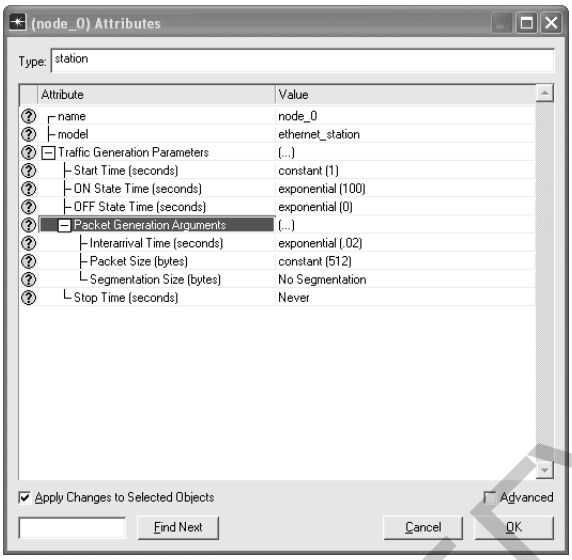


Рисунок 7.60 – Проверка параметров модели

Теперь необходимо сохранить все изменения в нашем проекте. Для этого в главном меню выбираем **File/Save**.

Конфигурация моделирования. Для начала необходимо определиться со статистикой, какие данные нам понадобятся для сравнения после окончания моделирования. Главный упор необходимо сделать на параметры пропускной способности (**throughput**), задержки (**delay**) и коллизий (**collisions**) (рисунок 7.61).

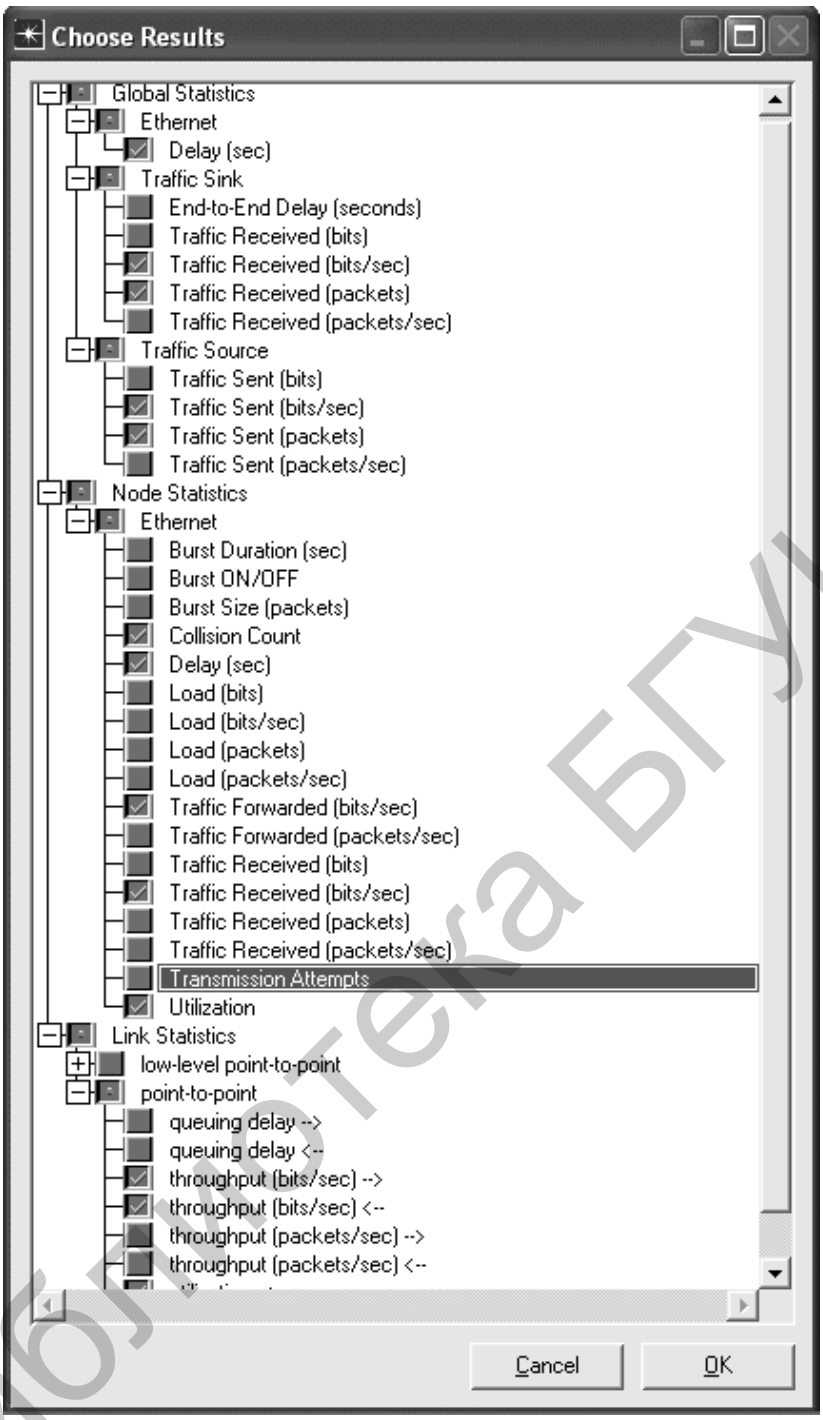


Рисунок 7.61 – Задание конфигурации моделирования

Для настройки статистики щелкните правой кнопкой мыши в любое место рабочей области нашего проекта для вызова контекстного меню, из которого нам понадобится пункт **Choose Individual Statistics**. Нажмите знак «+» напротив параметра **Global** для просмотра вложенных параметров. Нажмите на пункт **Ethernet** и поставьте галочку напротив **Delay**.

Далее раскроем параметр **Traffic Sink**, где нужно поставить галочки напротив **Traffic Received (bits/sec)**, **Traffic Received (packets)**. Затем раскроем параметр **Traffic Source** и выберем **Traffic Sent (bits/sec)** и **Traffic Sent (packets)**.

Нажмем знак «+» напротив **Node Statistics** для раскрытия параметров, после чего также нажмем знак «+» напротив **Ethernet**. Здесь нам понадобятся параметры **Collision Count**, **Delay**, **Traffic Forwarded (bits/sec)**, **Traffic Received (bits/sec)**, **Utilization** – ставим напротив них галочки.

Последними параметрами, которые понадобятся нам для статистики, будут **Link Statistics** и **point-to-point**. В открывшемся перечне необходимо выбрать пункты **throughput (bits/sec) ->**, **throughput (bits/sec) <-**, **utilization ->** и **utilization <-**. Сверьте все отмеченные параметры с рисунком 7.61 и нажмите **ОК**.

Запуск моделирования. Нажмите кнопку **Configure and Run** (пиктограмма в виде спортсмена на старте). Необходимо указать значения продолжительности выполнения моделирования. Для этого зададим параметр **Duration** равным **1**, а напротив него из выпадающего списка выберем **minute(s)** (рисунок 7.62).

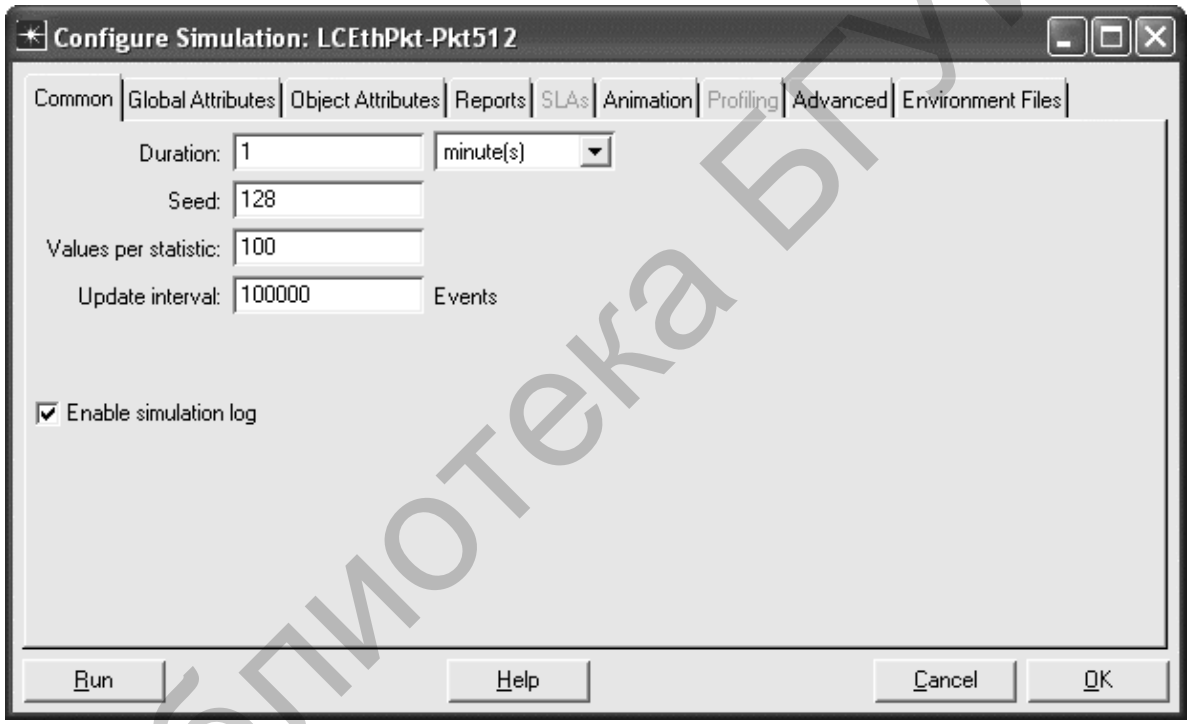


Рисунок 7.62 – Запуск моделирования

Это будет соответствовать времени моделирования в **одну минуту** (на самом деле выполнение пройдет быстрее, но система смоделирует нагрузку, равную одной минуте). Затем нажимаем кнопку **Run** («Запуск»). В зависимости от производительности вашего персонального компьютера это может занять определенное количество времени.

Мы посмотрим и оценим результаты после того, как создадим еще один сценарий. Сохраним текущие изменения в главном меню **File/Save**.

Создание нового сценария для моделирования. Нам необходимо создать новый сценарий для моделирования, чтобы увидеть, что происходит, когда размер

пакета увеличивается. Мы будем использовать максимально возможный размер пакета, равный **1500** байт (это обусловлено рамками технологии Ethernet).

Для этого из главного меню выберем пункт **Scenarios/Duplicate**. Тем самым мы сделаем дубликат уже имеющегося у нас сценария. В появившемся окне необходимо задать имя новому сценарию, рекомендуется назвать его **Pkt1500** (для удобства). После этого нажимаем **ОК**.

Теперь необходимо изменить размер пакета для каждого из наших хостов. Для этого возьмем любой из хостов, после чего вызовем контекстное меню щелчком правой кнопки мыши, в котором выберем пункт **Select Similar Nodes** (выделить все похожие хосты, в данном случае все 16). Это необходимо для удобства настройки, чтобы сконфигурировать один хост и присвоить такие же значения всем остальным.

После выделения всех хостов также на любом из них щелкнем правой кнопкой мыши и из меню выберем пункт **Edit attributes**. В появившемся окне перед началом работы поставим галочку напротив **Apply Changes to Selected Objects** (в нижней части окна).

Перейдем непосредственно к настройке. Нажмем знак «+» напротив пункта **Traffic Generation Parameters**, а затем знак «+» напротив **Packet Generation Parameters**. Здесь нам понадобятся настройки размера пакета **Packet Size (bytes)**. В открывшемся окне изменим параметр **Mean Outcome** на **1500** (что и будет соответствовать размеру пакета). Все изменения должны выглядеть в соответствии с рисунком 7.63. После этого в окне с изменениями необходимо нажать **ОК** и в основном окне настройки тоже.

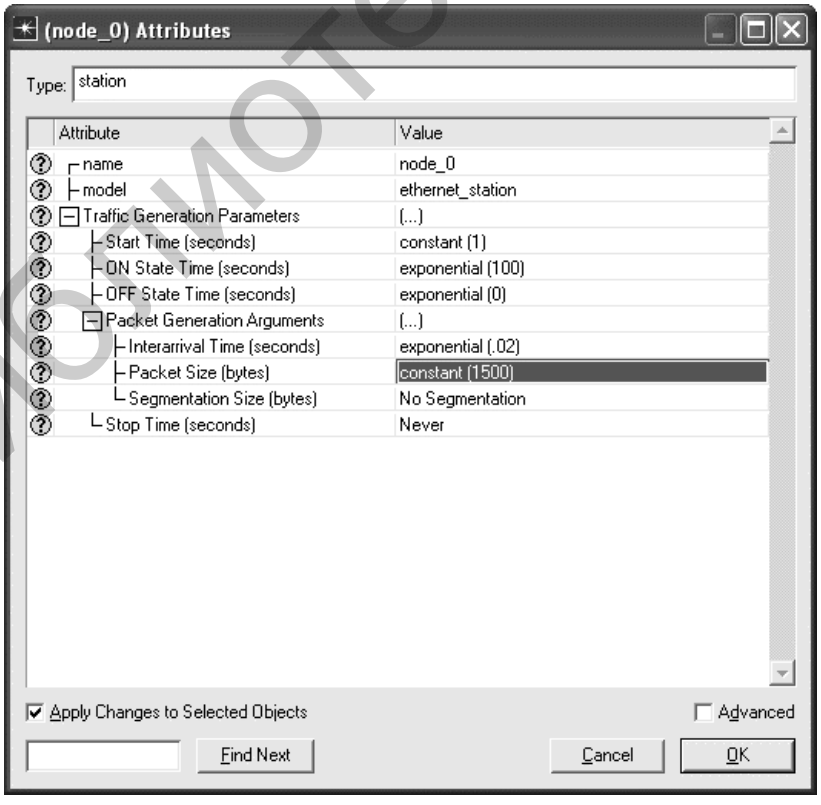


Рисунок 7.63 – Задание атрибутов нового сценария

Для того чтобы проверить, произошли изменения в настройках всех 16 хостов или нет (для этого мы и нажимали галочку напротив **Apply Changes to Selected Objects**), можно посмотреть в статусную строку главной рабочей области (внизу окна). Она должна содержать следующее: «16 objects changed» (16 объектов были изменены). Кроме того, при необходимости это можно проверить «вручную» путем вызова настроек каждого из хостов.

Далее приступим к запуску моделирования. Для этого из главного меню выбираем **Simulation/Run Discrete Event Simulation**. Это может занять несколько минут в зависимости от производительности компьютера. После окончания симуляции можем приступать к сравнению результатов выполнения двух сценариев.

Для этого из главного меню выберем пункт **Results/Compare Results** («Сравнение результатов»). В появившемся окне справа снизу выберем пункт **All Scenarios**, затем из выпадающего списка – **time_average**. Наибольший интерес для нас будет представлять график, находящийся в пункте **Traffic Sink**, под названием **Traffic Received (bits/sec)**. Нажмем кнопку **Show**. Если настройки выдачи результатов были выбраны правильно, будут отображены два графика разных цветов, один из них для пакета **512**, другой для пакета **1500**, пример показан на рисунке 7.64.

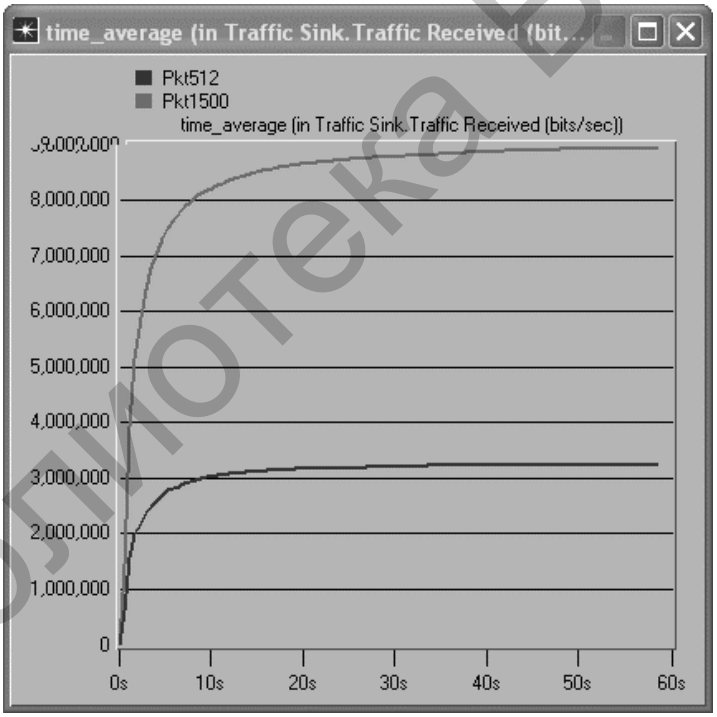


Рисунок 7.64 – Отображение результатов моделирования

Если вы закончили работу с графиком, то его можно закрыть, нажав на крестик в верхнем правом углу. После этого можно скрыть график (**Hide**) или удалить его (**Delete**). Можно нажать кнопку **Delete** (так или иначе результаты все равно будут сохранены).

Кроме того, можно менять сценарий по своему усмотрению. Для этого в главном меню выберите **Scenarios/Manage Scenarios**. Появится окно с настройками ваших сценариев.

Можно выбрать в поле **Results** вариант **recollect** (заново собрать данные), затем указать время моделирования для новых данных, например, можно выбрать 1 день вместо 1 мин и т. д.

После настройки данных для двух сценариев необходимо нажать **ОК** и моделирование запустится. После его окончания можно приступать к работе с результатами.

Возможность возникновения ошибки в ходе моделирования. В ходе моделирования может возникнуть ошибка, связанная со временем итерации **Interarrival Time (seconds)**. Это было установлено опытным путем.

Мы используем значение этого параметра для того, чтобы задавать временной интервал между пакетами, равный 0,02 с. Значение, которое мы используем для настройки этого параметра, выглядит следующим образом: **.02**. Именно на этом этапе выполнение моделирования может «зациклиться», что приведет к сбою работы программы OpNet.

Данная ошибка возникает не на всех системах и версиях операционной системы **Windows**, но если у вас русскоязычная версия Windows, а в региональных настройках выбрана **Россия** или **Беларусь**, то скорее всего вы получите ошибку в ходе моделирования.

Это связано с тем, что программа считывает код цифрового разделителя дробной части от целой для текущих региональных настроек. На постсоветском пространстве используется запятая для отделения дробной части от целой, а во всем остальном мире – точка в качестве разделителя. Именно с этим связана проблема моделирования.

Способ устранения ошибки. Необходимо зайти в панель управления Windows, найти региональные настройки. Обычно они находятся, где настройки языка, раскладки и т. д. Необходимо найти закладку **Форматы** и нажать кнопку **Дополнительные параметры**. Там изменить разделитель целой и дробной части – вместо запятой ставим точку. Затем нажать **Применить**.

Примечание – В разных версиях Windows путь к этим настройкам может отличаться, но суть такая же.

После устранения ошибки попробуйте повторить выполнение моделирования проекта.

7.5 Моделирование сети в масштабе города

Краткая теоретическая справка. При построении сети в масштабе города необходимо учитывать услуги, которые планируется предоставлять на базе данной сети. Как правило, это услуги **Triple Play**, т. е. услуги по передаче речи, данных и видео. Каждый из перечисленных типов трафика предъявляет свои требования к показателям качества обслуживания. Для обеспечения необходимого **QoS** для всех услуг выбрана следующая трехуровневая топология построения сетей класса

Metro: уровень ядра, уровень агрегации (уровень распределения), уровень доступа.

Уровень доступа осуществляет физическую концентрацию абонентских линий, организует разделение абонентов с использованием виртуальных сетей, обеспечивает ограничение скорости передачи данных на входе в сеть и осуществляет базовые функции безопасности. Предоставляет высокоскоростной доступ абонентов к уровню распределения. Совокупность узлов уровня доступа, объединенных в единую физическую структуру и предоставляющих доступ к общему узлу уровня распределения, принято называть доменом доступа.

Уровень распределения терминирует виртуальные сети уровня доступа с использованием протокола IP, предоставляет доступ к локальным ресурсам и позволяет вести обмен трафиком между различными узлами уровня распределения. Домен распределения предоставляет доступ к оборудованию предоставления сервисов. Осуществляет расширенные функции обеспечения безопасности и управления качеством обслуживания сети. Предоставляет высокоскоростной доступ к узлам уровня ядра сети. Совокупность узлов уровня распределения, объединенных в единую физическую структуру, называют доменом распределения.

Транспортный уровень (уровень ядра) сети служит высокоскоростной и надежной магистралью, объединяющей домены распределения.

Для обеспечения повышенной надежности и резервирования широко применяется топологическая модель кольца на уровне доступа и ядра. В некоторых случаях не исключается возможность использования топологии «звезда» на уровне доступа для подключения абонентов.

Как часть механизмов распределения ресурсов каждый маршрутизатор должен реализовывать различную дисциплину организации очереди, которая определяет, как пакеты будут буферизироваться до передачи. Различные способы дисциплины организации очереди могут быть использованы для контроля передачи информации, например, какие пакеты будут переданы, а какие отброшены. Дисциплины организации очереди также определяют время задержки пакета в очереди до передачи. Далее мы рассмотрим общие дисциплины организаций очереди, такие, как «первый вошел – первый вышел» (**FIFO**), приоритет одних пакетов над другими, стоящими в очереди (**PQ**), взвешенные справедливые очереди (**WFQ**).

Идея очереди «первый вошел – первый вышел» (**FIFO**) состоит в том, что первый пакет, поступивший в маршрутизатор, будет обработан в первую очередь. Учитывая, что буферное пространство каждого маршрутизатора ограничено, если пакет прибывает и очередь (буферное пространство) заполнена, маршрутизатор отбрасывает пакет. Степень важности пакета не учитывается.

Приоритет в очереди (**PQ**) – это несколько видоизмененное **FIFO**. Приоритет в очереди одних пакетов над другими означает, что пакеты с приоритетом при передаче помечаются. Вследствие этого маршрутизаторы реализуют несколько очередей, каждая очередь, имеющая свой класс приоритета, организуется способом **FIFO**. Этот способ дисциплин организации очереди позволяет пакетам с высоким приоритетом быстрее обрабатываться.

Взвешенные справедливые очереди (**WFQ**) автоматически разбивают трафик на потоки. Если потоков больше, чем очередей, то в одну очередь помещается несколько потоков. Обработчик **WFQ** обеспечивает равномерное разделение полосы между существующими потоками. Для этого доступная полоса делится на число потоков и каждый получает равную часть. Кроме того, каждый поток получает свой вес с некоторым коэффициентом, обратно пропорциональным IP-приоритету (**TOS**). В итоге **WFQ** автоматически справедливо распределяет доступную пропускную способность, дополнительно учитывая **TOS**. Потоки с одинаковыми IP-приоритетами **TOS** получат равные доли полосы пропускания; потоки с большим IP-приоритетом – большую долю полосы.

Построим сеть, использующую три приложения: **FTP**, **Video** и **VoIP**. Проанализируем, как изменение дисциплин организации очереди в маршрутизаторе влияет на производительность приложений и использование сетевых ресурсов.

Создание проекта. Запустите приложение **OpNet IT Guru Academic Edition**. Создайте новый проект из главного меню: **File/New/Project**.

Поменяйте название нового проекта на **xx_Queues** (где **xx** – это ваши инициалы). Измените название будущего сценария на **FIFO**.

В диалоговом окне **Startup Wizard Initial Topology** выберите **Create Empty Scenario**. Из списка **Network Scale** выберите **Campus**. Нажмите **Next** три раза, затем **OK**.

Создание и настройка сети. На экране вашего компьютера должно появиться диалоговое окно **Object Palette**, если оно отсутствует, откройте его, щелкнув на значке  . В диалоговом окне **Object Palette** выберите **internet_ toolbox**.

В рабочую зону проекта добавьте следующие объекты: **Application Config**, **Profile Config**, **QoS Attribute Config**, пять **ethernet_wkstn**, один **ethernet_server** и два маршрутизатора **ethernet4_slip8_gtwy**.

Соедините маршрутизаторы двунаправленным кабелем **PPP_DS1**.

Соедините рабочие станции и сервер с маршрутизатором, используя двунаправленный кабель **10Base_T** (рисунок 7.65).

Переименуйте добавленные объекты, как на рисунке 7.65, и сохраните проект.

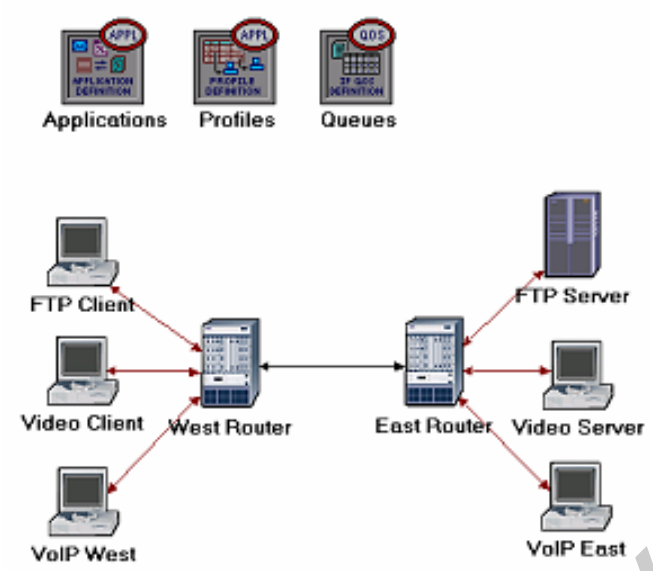


Рисунок 7.65 – Построение сети

Настройка Application. Щелчком правой кнопки мыши на объекте **Application** вызовите контекстное меню и выберите **Edit Attributes**, разверните **Application Definitions**, нажав на знак «+», установите **rows**, равный **3**. Присвойте следующие название рядам: **FTP Application**, **Video Application**, **VoIP Application**.

Отредактируйте ряд **FTP Application** следующим образом: разверните **Description**, установите **High Load** на **Ftp**, нажмите на значение **High Load** и в развернутом меню выберите **Edit**, установите **Constant(10)** в **Inter-Request Time**, установите **Constant(1000000)** в **File Size**. Установите **Type of Service (ToS)** как **Best Effort(0)**.

Отредактируйте ряд **Video Application** следующим образом: разверните **Description**, установите **Low Resolution Video** в **Video Conferencing**, нажмите на значение **Low Resolution Video** и выберите **Edit**, отредактируйте значение в поле **Type of Service** (появившееся окно **Configure TOS/DSCP)**, в развернутом меню установите **Streaming Multimedia (4) в ToS**, нажмите **OK** дважды.

Отредактируйте ряд **VoIP Application** следующим образом: разверните **Description**, установите **PCM Quality Speech** в **Voice**. После того как вы отредактировали это значение, убедитесь, что в поле **ToS** установилось **Interactive Voice (6)** (рисунок 7.66).

Нажмите **OK** и сохраните ваш проект.

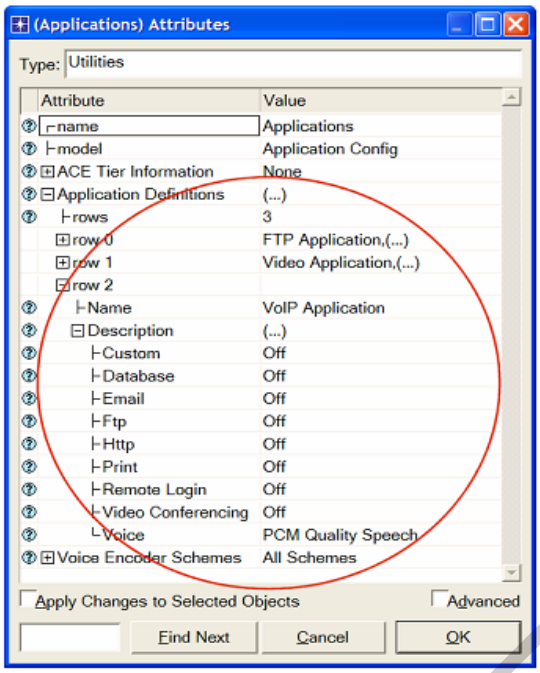


Рисунок 7.66 – Настройка **Application**

Настройка Profiles. Щелчком правой кнопки мыши на объекте **Profiles** вызовите контекстное меню и выберите **Edit Attributes**, разверните **Profile Configuration**, установите **rows**, равный **3**.

Отредактируйте **row 0**, как показано на рисунке 7.67.

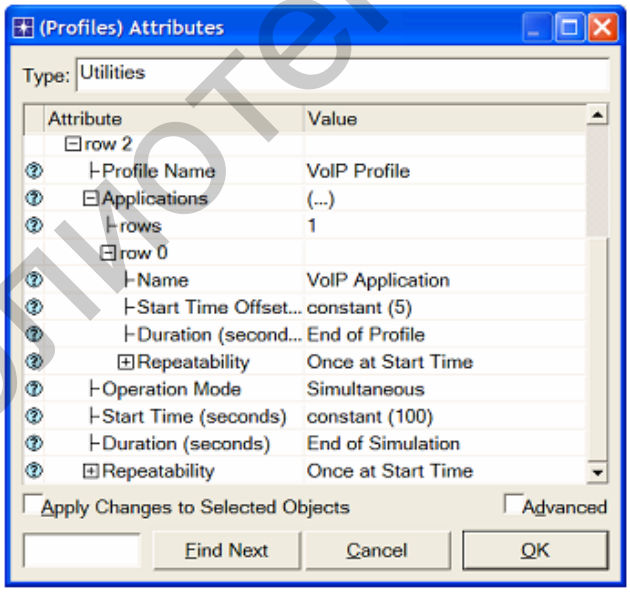


Рисунок 7.67 – Настройка **Profiles**

Отредактируйте **row 1**, как показано на рисунке 7.68.

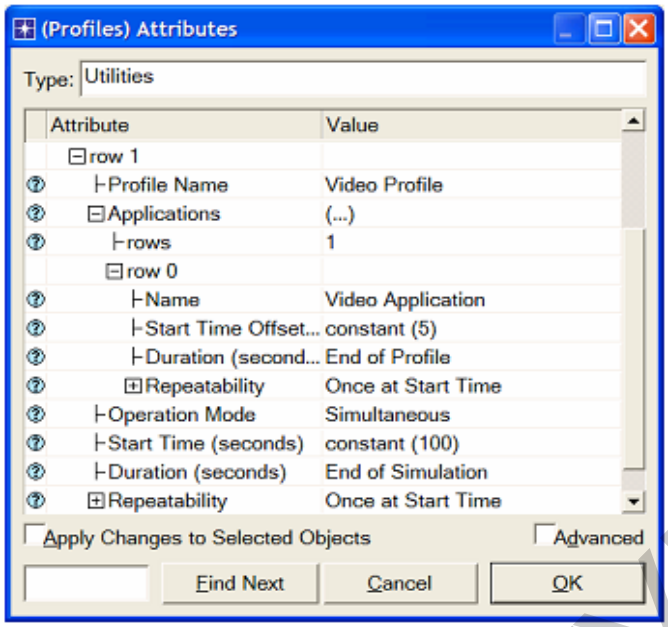


Рисунок 7.68 – Окно установки для **row1**

Отредактируйте **row 1**, как показано на рисунке 7.69.

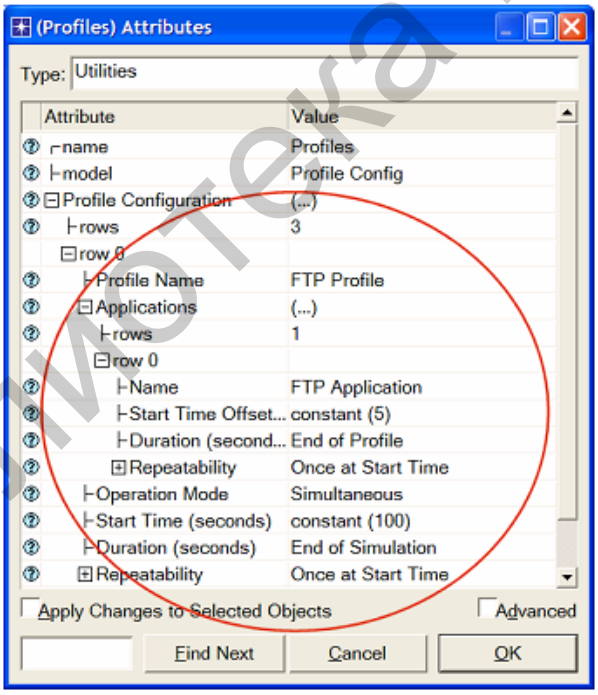


Рисунок 7.69 – Установка параметров **row1**

Нажмите **OK** и сохраните проект.

Настройка Queues. Объект **Queues** оставьте без изменений, это нужно для того, чтобы вы могли проверять конфигурации профилей **FIFO**, **PQ** и **WFQ**.

Настройка Workstations и Servers. Щелчком правой кнопки мыши выберите объект **FTP Client/Edit Attributes**, разверните **Application: Supported**

Profiles, установите **row**, равный **1**, установите **Profile Name как FTP Profile**, нажмите **OK**.

Щелчком правой кнопки мыши выберите объект **Video Client/Edit Attributes**, разверните **Application: Supported Profiles**, установите **row**, равный **1**, установите **Profile Name** как **Video Profile**, нажмите **OK**.

Щелчком правой кнопки мыши выберите объект **VoIP West/Edit Attributes**. Разверните **Application: Supported Profiles**, установите **row**, равный **1**, установите **Profile Name** как **VoIP Profile**.

Отредактируйте значение **Application: Supported Services** следующим образом: нажмите на поле **Value**, в развернутом меню выберите **Edit**, установите **row**, равный **1**, установите **Service Name** как **VoIP Application**, нажмите **OK** дважды.

Щелчком правой кнопки мыши выберите объект **VoIP East/Edit Attributes**. Разверните **Application: Supported Profiles**, установите **row**, равный **1**, установите **Profile Name** как **VoIP Profile**.

Отредактируйте значение **Application: Supported Services** следующим образом: нажмите на поле **Value**, в развернутом меню выберите **Edit**, установите **row**, равный **1**, установите **Service Name** как **VoIP Application**, нажмите **OK** дважды.

Щелчком правой кнопки мыши выберите объект **FTP Server/Edit Attributes**, отредактируйте значение **Application: Supported Services**, установите **row**, равный **1**, установите **Service Name** как **FTP Application**, нажмите **OK** дважды.

Щелчком правой кнопки мыши выберите объект **Video Server/ Edit Attributes**, отредактируйте значение **Application: Supported Services**, установите **row**, равный **1**, установите **Service Name** как **Video Application**, нажмите **OK** дважды.

Сохраните ваш проект.

Настройка Routers. Нажмите на линию связи между маршрутизаторами **East** и **West**, в главном меню программы выберите **Protocols/IP/QoS/Configure QoS**.

Установите необходимые настройки, как показано на рисунке 7.70.

Примечание – Так как флажок установлен в **Visualize QoS Configuration**, линия связи используется и должна окраситься в цвет параметра, выбранного в окне **QoS** (для **FIFO** голубой цвет).

Нажмите **OK**. Сохраните проект.

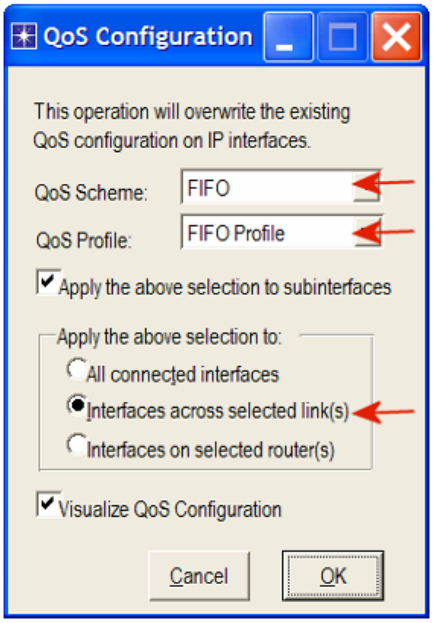


Рисунок 7.70 – Настройка конфигурации **QoS**

Выбор статистик. Правой кнопкой мыши щелкните по рабочей зоне проекта и выберите **Choose Individual Statistics**.

В диалоговом окне **Choose Results** установите статистики в соответствии с рисунком 7.71.

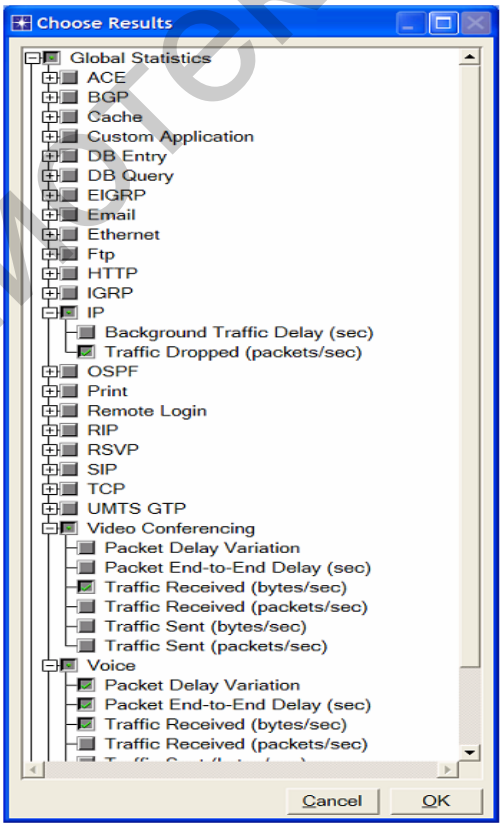


Рисунок 7.71 – Выбор статистик

Нажмите **OK** и сохраните ваш проект.

Настройка моделирования. Для настройки продолжительности эксперимента выполните следующее:

1) нажмите на значок  , появится окно **Configure Simulation**;

2) установите продолжительность эксперимента 150 с;

3) нажмите **OK** и сохраните проект.

Дублирование эксперимента. Созданная нами сеть использует только дисциплину организации очереди **FIFO**. Для анализа различных дисциплин организации очереди мы должны создать еще два сценария – один для тестирования **PQ**, а другой для **WFQ**.

В главном меню выберите **Scenarios** и установите **Duplicate Scenario**, название сценария **PQ**, нажмите **OK**.

Нажмите на линию связи между маршрутизаторами **East** и **West**, в главном меню программы выберите **Protocols/IP/QoS/Configure QoS**.

Установите необходимые настройки, как показано на рисунке 7.72.

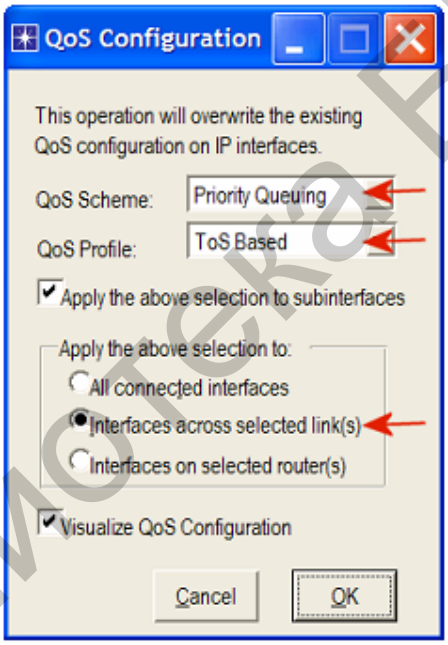


Рисунок 7.72 – Настройки конфигурации **QoS** для сценария **PQ**

Примечание – Так как флажок установлен в **Visualize QoS Configuration**, линия связи используется и должна окраситься в цвет параметра, выбранного в окне **QoS** (для **PQ** оранжевый цвет). Сохраните проект.

В главном меню выберите **Scenarios** и установите **Duplicate Scenario**, название сценария **WFQ**, нажмите **OK**.

Нажмите на линию связи между маршрутизаторами **East** и **West**, в главном меню программы выберите **Protocols/IP/QoS/Configure QoS**.

Установите необходимые настройки, как показано на рисунке 7.73.

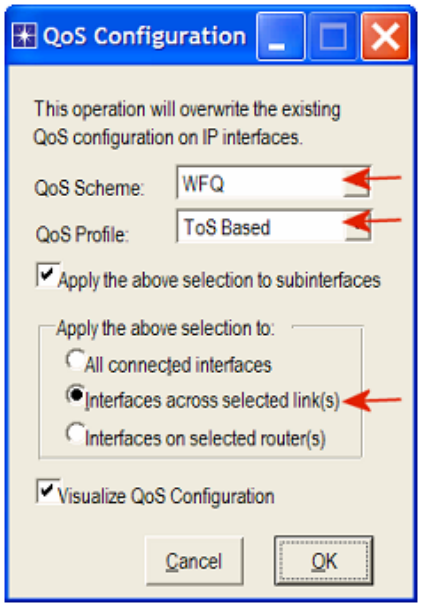


Рисунок 7.73 – Настройка конфигурации **QoS** для сценария **WFQ**

Примечание – Так как флажок установлен в **Visualize QoS Configuration**, линия связи используется и должна окраситься в цвет параметра, выбранного в окне **QoS** (для **WFQ** зеленый цвет).

Сохраните проект.

Запуск моделирования. Для трех сценариев запуск эксперимента будет происходить одновременно.

В главном меню выберите **Scenarios**, установите **Manage Scenarios**.

Измените значения в столбце **Results** на **<collect>** (или **<recollect>**) для трех сценариев, как показано на рисунке 7.74.

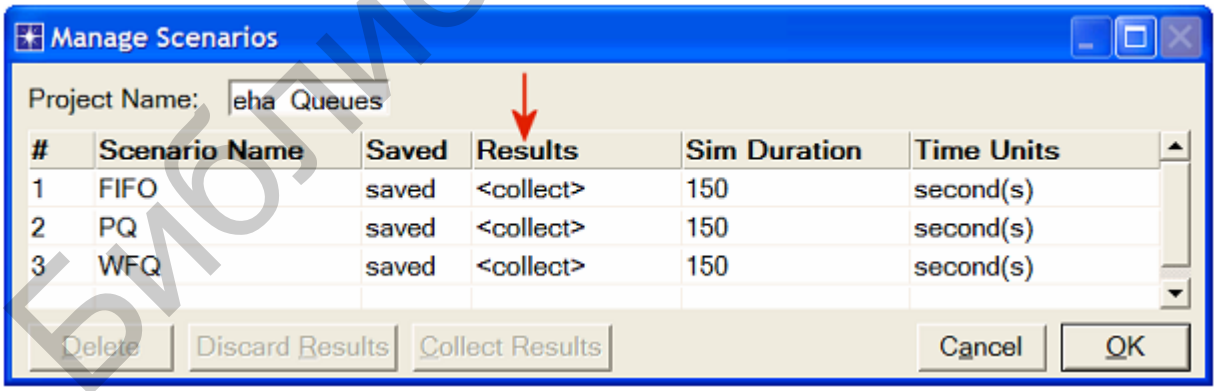


Рисунок 7.74 – Окно запуска моделирования

Нажмите **OK** для запуска эксперимента. В зависимости от скорости вашего процессора завершение может занять несколько минут.

После завершения эксперимента нажмите **Close**.

Сохраните ваш проект.

Анализ результатов. Проанализируйте полученные результаты.

В главном меню выберите **Results/Compare Results**.

Выберите **IP Traffic Droppe** и нажмите **Show**. Итоговый график, получившийся в ходе эксперимента, должен соответствовать представленному на рисунке 7.75.

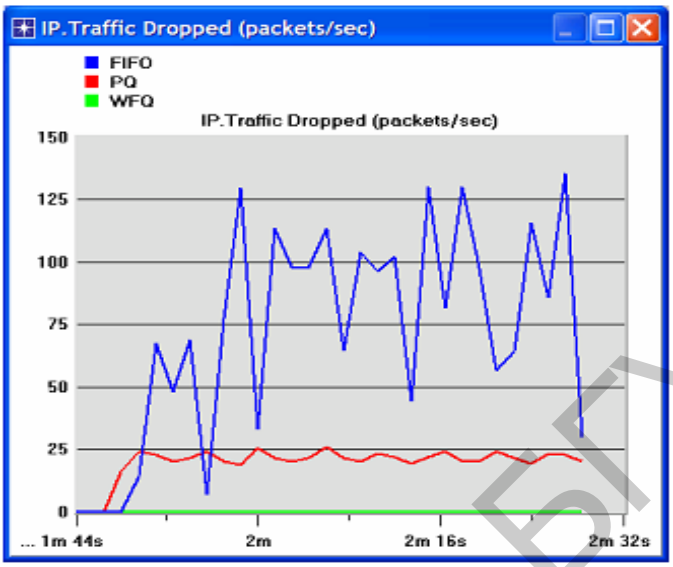


Рисунок 7.75 – Графики для **IP Traffic**

График для **Video Conferencing Traffic Received** представлен на рисунке 7.76.

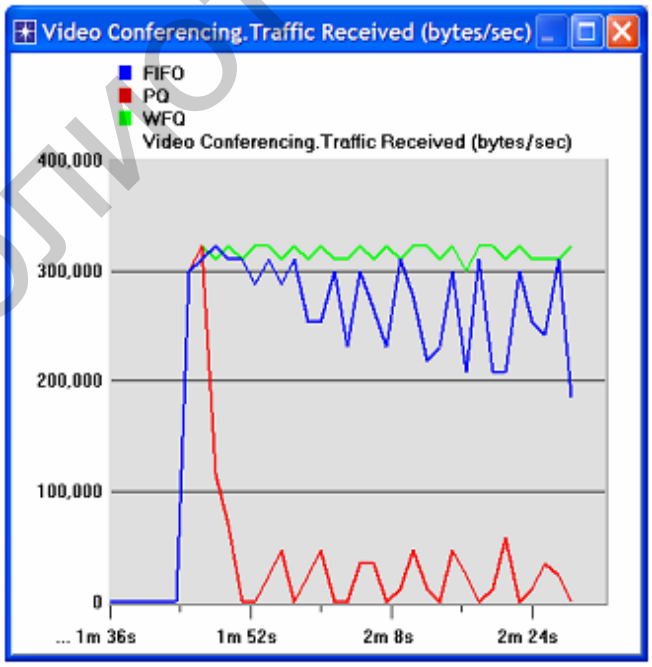


Рисунок 7.76 – График для **Video Conferencing**

График для **Voice Traffic Received** представлен на рисунке 7.77.

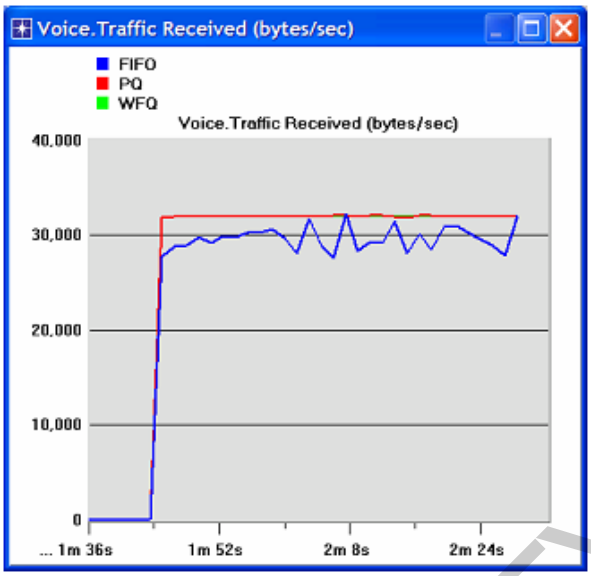


Рисунок 7.77 – График для **Voice Traffic**

Графики для **Voice Packet End-to-End Delay** и **Voice Packet Delay Variation** мы видим на рисунках 7.78, 7.79 соответственно.

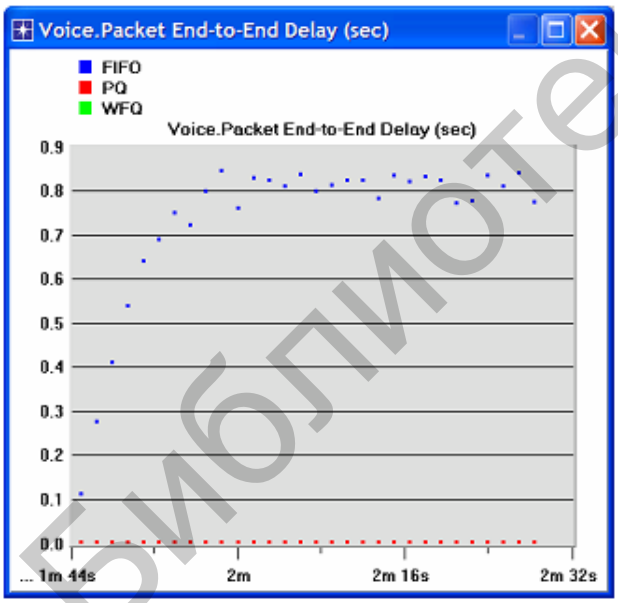


Рисунок 7.78 – Графики для **Voice Packet End-to-End Delay**

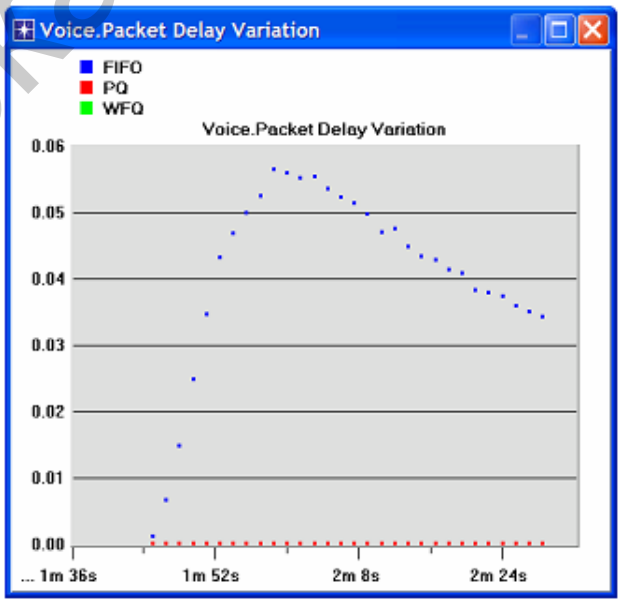


Рисунок 7.79 – Графики для **Voice Packet Delay Variation**

Примечание – На представленных рисунках мы не наблюдаем графика для **WFQ**, т. к. он перекрывается графиком **PQ**.

Проанализируйте полученные графики, обратите внимание на графики **Voice Packet End-to-End Delay** и **Voice Packet Delay Variation**, которые перекрываются друг другом. Сравните три дисциплины организации очереди, дайте свою оценку каждой.

Отредактируйте объект **Queues** и проверьте профиль назначенных дисциплин **FIFO**, **PQ** и **WFQ**.

Для всех сценариев выберите **queuing delay** – статистику для линии связи между маршрутизаторами **East** и **West**. Перезапустите эксперимент, откройте график, где сравнивается задержка в очереди для всех сценариев (**queuing delay** является статистикой только для топологии **point-to-point** (точка – точка).

7.6 Качество обслуживания (Quality of Service): влияние политики очереди

Краткая теоретическая справка. В сетях с буферизацией пакетов маршрутизаторы способны обслуживать одну и более очередей для каждой выходной линии. Создание очереди необходимо при занятости линии для сохранения пакетов. Политика очереди (**Queuing Policy**) представляет собой набор правил, по которым пакеты выстраиваются в очередь и отсылаются в линию. «Первый вошел – первый вышел» (**First-In**, **First-Out**, **FIFO**) является традиционной политикой, которую легко реализовать. При этом все потоки данных являются равнозначными. Только что прибывший пакет помещается в конец очереди и ждет отправки.

В настоящее время Интернет также используется для передачи речи и видеоданных, поэтому простого механизма **FIFO** недостаточно. Голосовые и видеоприложения требуют предельно допустимых задержек и дрожаний фаз сигналов при передаче пакетов. Это достигается путем обработки пакетов в пределах очереди по-разному. При осуществлении политики честной очереди с весовыми коэффициентами (**Weighted Fair Queuing**, **WFQ**) используется механизм планирования пакетных потоков данных с различными приоритетами. Весовые коэффициенты соответствуют классам, разделенным по важности. То есть очереди обслуживаются по правилам, основанным на их весовых коэффициентах. Например, если очереди А был назначен весовой коэффициент один, а очереди В – весовой коэффициент два, то при посылке двух пакетов из очереди В из очереди А посылается только один пакет. Использование весовых коэффициентов с большим значением при обслуживании очередей дает преимущество голосовым и видеопотокам перед стандартным трафиком данных.

В очереди с приоритетом различные очереди обслуживаются на основе классов приоритета, назначенных пакетам. В этом случае все пакеты с высшим приоритетом отсылаются раньше, чем пакеты с низшим. Если мы имеем две очереди, одна формируется на основе трафика второго класса, другая – на основе трафика первого класса. Очередь на основе трафика второго класса будет обслуживаться до того момента, пока не опустеет. Передача очереди на основе трафика первого

класса может прекратиться в случае прихода очереди на основе трафика второго класса.

Построение имитационной модели. Запустите **OpNet IT Guru Academic Edition**. Выберите закладку **File/New**, выберите **Project** и нажмите **OK**. Измените **Project Name** на **xx_QoS_Queuing** (где **xx** – ваши инициалы). Назовите **Scenario Name** именем **PQ** и нажмите **OK**. В появившемся окне **Initial Topology** выберите **Create Empty Scenario** и нажмите **Next**. В окне **Choose Network Scale** выберите **Choose from Maps** и нажмите **Next**. В окне **Choose Map** выберите **europe** и нажмите **Next**. В окне **Select Technologies** нажмите **Next**. В окне **Review** нажмите **OK** (рисунок 7.80).

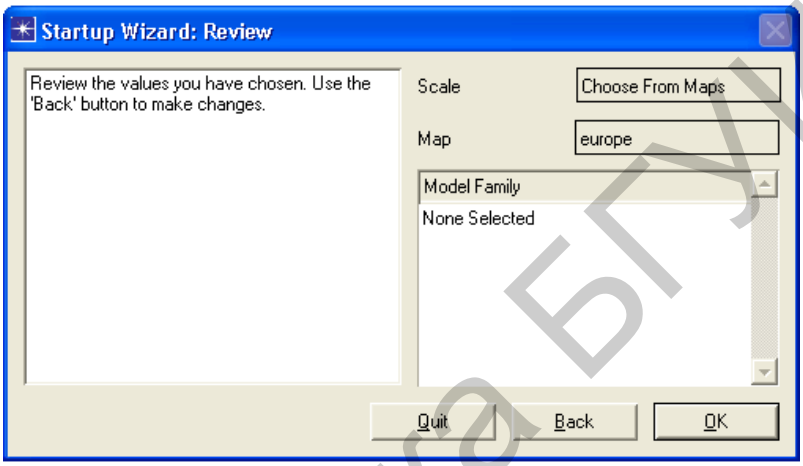


Рисунок 7.80 – Задание параметров модели

Для начала сконфигурируйте два идентичных **FTP**-приложения, отличающихся только по приоритету.

Перенесите объект **Application Config** из поля окна **Object Palette** на рабочую область. Щелкните на нем правой кнопкой мыши и выберите **Edit Attribute**.

В поле **Value** напротив атрибута **name** напишите **Applications**.

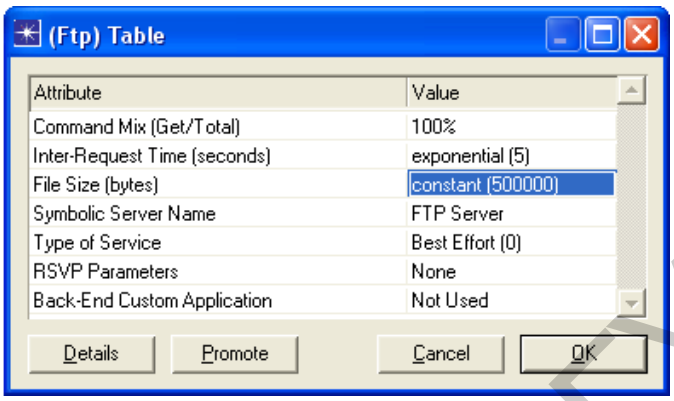
Раскройте **Application Definitions** (нажмите знак «+») и установите для атрибута **rows** значение, равное **2**. Далее раскройте **rows 0** и для атрибута **Name** напишите название **FTX_Low_Priority_Application**. Раскройте **Description** и в ячейке **value** атрибута **FTP** выберите **edit**. Установите для **Inter-Request Time (seconds)** атрибут **exponential (5)**, для **File Size (bytes)** установите **constant (500000)**. В обоих случаях в поле **Special Value** выберите **Not Used** для ввода значений. Ввод значений осуществляется в окне **Mean Outcome**. Убедитесь, что в поле **Type of Service** установлено **Best Effort (0)**.

Best Effort (0) представляет собой низший уровень приоритета. Приложение, которое вы сейчас настроили, будет передавать файлы один за другим размером 500 Кбайт с периодом между передачами, равным 5 с. Нажмите **OK** для закрытия окна.

Теперь раскройте атрибут **row 1** и поменяйте имя на **FTP_High_Priority_Application**. Снова установите для **Inter-Request Time (seconds)** атрибут **exponential (5)**, для атрибута **File Size (bytes) – constant (500000)**. Далее в поле **Type of Service** выберите **Excellent Effort (3)**. **Excellent Effort (3)** обеспечивает приоритет выше, чем для **Best Effort** (рисунок 7.81).

Нажмите **OK** для закрытия окна.

а



б

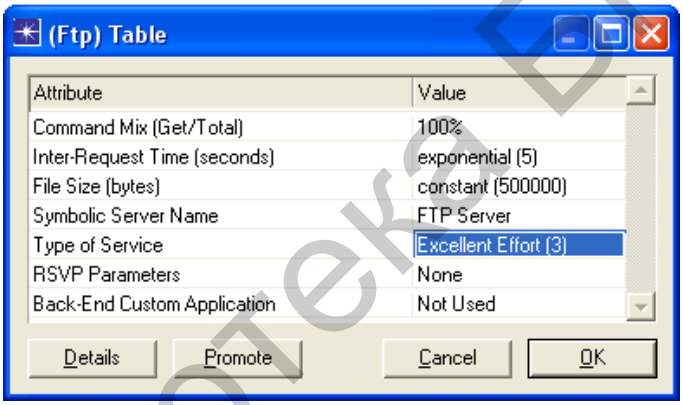


Рисунок 7.81 – Выбор атрибутов модели

Перенесите объект **Profile Config** из поля окна **Object Palette** на рабочую область. Щелкните на нем правой кнопкой мыши и выберите **Edit Attribute**. В поле **Value** напротив атрибута **name** напишите **Profiles**. Раскройте **Profile Configuration** и установите для атрибута **rows** значение, равное **2**. Далее раскройте атрибут **rows 0** и установите в **Profile Name** название **FTX_Low_Priority_Profile**. Раскройте **Applications** и установите для атрибута **rows** значение, равное **1**. Раскройте **row 0** и присвойте ему значение **FTP_Low_Priority_Application**. Выберите для **Duration (seconds)** атрибут **End of Last Task**. Раскройте **Repeatability** и установите для **Inter-repetition** характеристики **constant (0)**.

Раскройте атрибут **row 1** и запишите для **Profile Name** имя **FTP_High_Priority_Profile**. Раскройте атрибут **Applications** и установите для атрибута **rows** значение, равное **1**. Раскройте **row 0** и присвойте ему значение **FTP_High_Priority_Application**. Для характеристики **Duration(seconds)** выберите

End of Last Task. Раскройте атрибут **Repeatability** и для характеристики **Inter-repetition Time(seconds)** выберите **constant (0)** (рисунок 7.82).

Нажмите **OK** для закрытия окна.

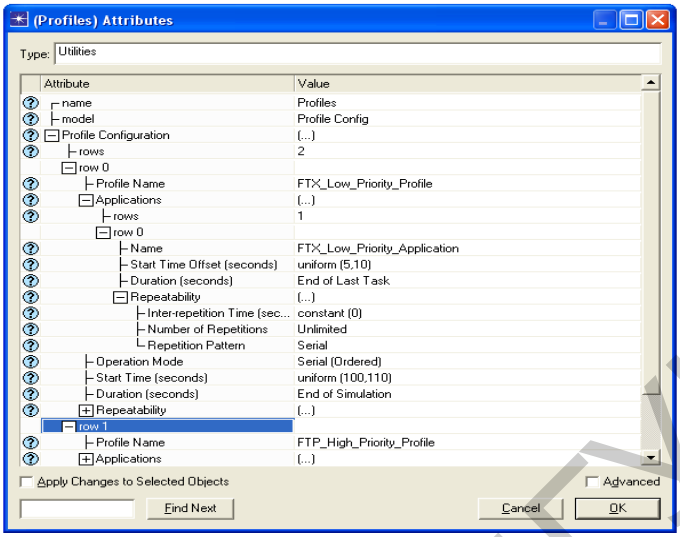


Рисунок 7.82 – Установка атрибутов профиля

Так как мы создали профиль, пришло время создать сеть.

Перенесите устройство **ppp_wkstn** из поля окна **Object Palette** в рабочую область. Щелкните правой кнопкой мыши на станции и выберите **Edit Attribute**. Измените имя (**name**) устройства на **FTP Low Client**. Отредактируйте **Application: Supported Profiles**. Присвойте атрибуту **rows** значение **1**, раскройте **row 0** и измените имя профиля (**Profile Name**) на **FTP_Low_Priority_Profile** (рисунок 7.83). Нажмите **OK** для закрытия окна.

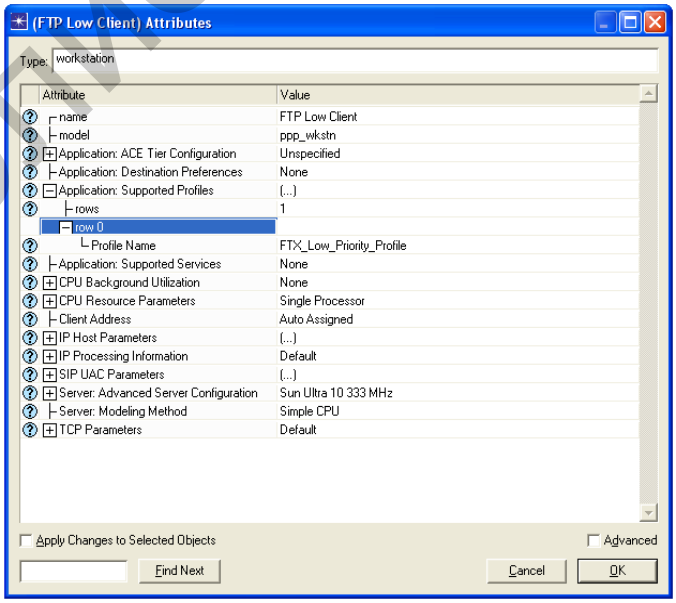


Рисунок 7.83 – Установка атрибутов **FTP Low Client**

Перенесите еще одно устройство **ppp_wkstn** из поля окна **Object Palette** в рабочую область. Щелкните правой кнопкой мыши на станции и выберите **Edit Attribute**. Измените имя (**name**) устройства на **FTP High Client**. Отредактируйте **Application: Supported Profiles**. Присвойте атрибуту **rows** значение **1**, раскройте **row 0** и измените имя профиля (**Profile Name**) на **FTP_Higt_ Priority_Profile**. Нажмите **OK** для закрытия окна.

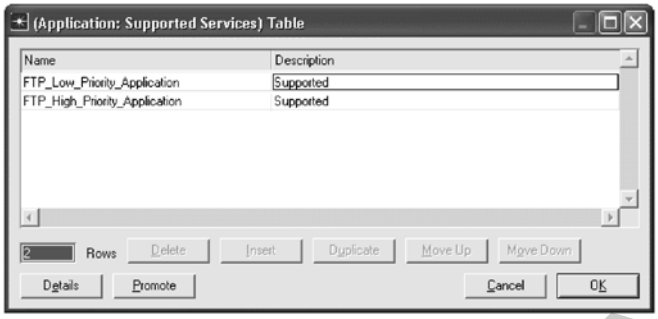


Рисунок 7.84 – Установка атрибутов **FTP Low Client**

Перенесите устройство **ppp_server** из поля окна **Object Palette** в рабочую область. Сейчас мы настроим сервер для поддержки FTP-приложений. Щелкните правой кнопкой мыши на устройстве и выберите **Edit Attribute**. Измените имя (**name**) устройства на **FTP Server**. Отредактируйте **Application: Supported Services**. Присвойте атрибуту **rows** значение **2**, установите первому **row** имя (**Name**) **FTP_Low_Priority_Application**, второму – **FTP_High_Priority_Application** (рисунок 7.85). Нажмите **OK** два раза для закрытия окон.

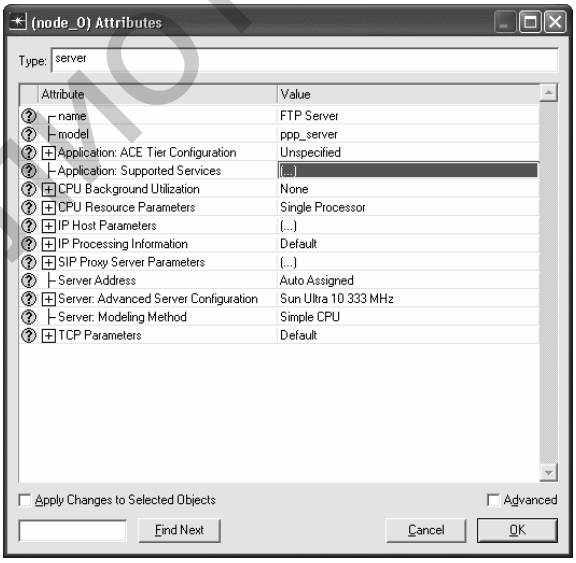


Рисунок 7.85 – Атрибуты **ppp server**

Перенесите второе устройство **ethernet4_slip8_gtwy** из поля окна **Object Palette** в рабочую область. Щелкните правой кнопкой мыши на маршрутизаторе

и выберите **Set Name**. Присвойте ему имя **Router 1**. Щелкните правой кнопкой мыши на втором маршрутизаторе и выберите **Set Name**. Присвойте ему имя **QoS Router**. Перенесите линки **PPP_DS1** из поля окна **Object Palette** и используйте их для соединения **FTP** клиентов с **Router 1** и для соединения маршрутизаторов между собой. Перенесите линк **PPP_DS3** из поля окна **Object Palette** и используйте для соединения **FTP Server** и **QoS Router** (рисунок 7.86).

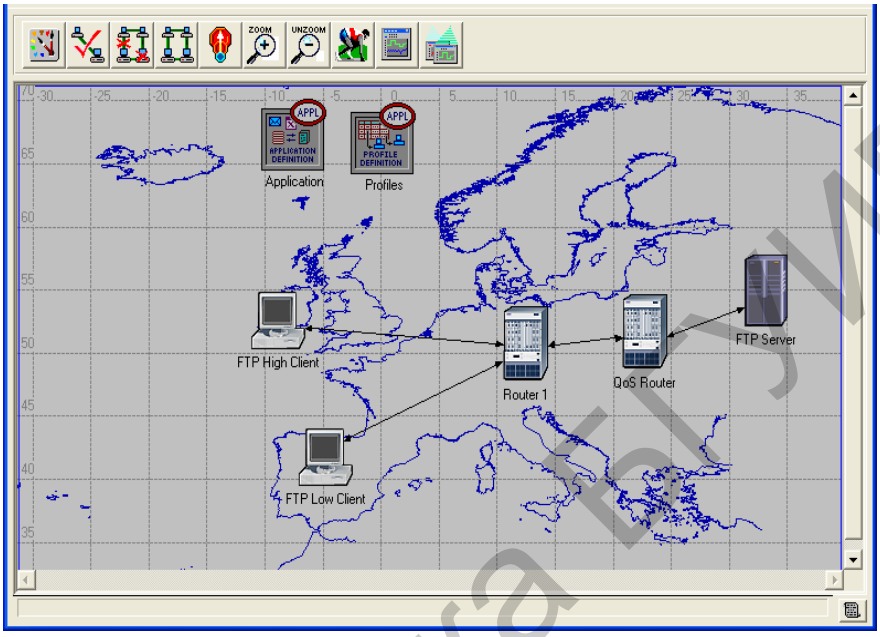


Рисунок 7.86 – Конфигурация сети

Теперь нам нужно настроить механизмы очереди, которые будут использоваться маршрутизаторами. Щелкните правой кнопкой мыши на **QoS Router** и выберите **Select Similar Nodes**. Это позволит нам делать изменения в настройках обоих маршрутизаторов одновременно.

Выберите закладку **Protocols/IP/ QoS/Configure QoS**. Для **QoS Scheme** выберите **Priority Queuing**. **ToS Based** было выбрано для **QoS profile**. Это означает, что маршрутизатор использует поле **Type of Service** в заголовке IP-пакета для определения пакета с приоритетом. **Best Effort** и **Exellent Effort –** приоритеты, определенные нами ранее, используются в поле **ToS**.

Далее выберите в представленном списке **Apply the above selection to: Interfaces on selected router(s)** (рисунок 7.87).

Нажмите **OK** для закрытия окна.

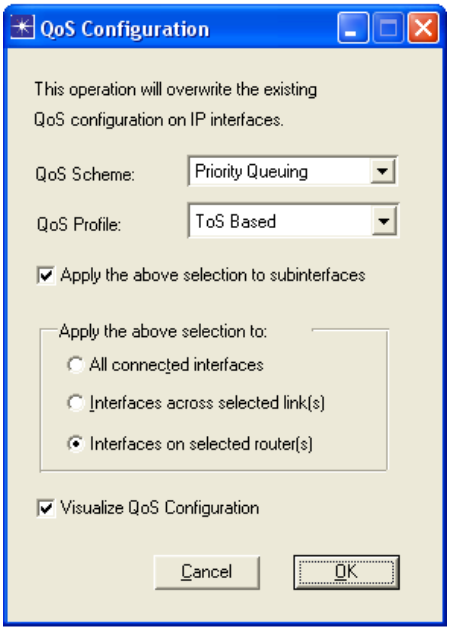


Рисунок 7.87 – Настройка **QoS**

Теперь мы настроили **QoS Router** для использования механизмов очереди с приоритетом. Щелкните правой кнопкой мыши на линк **PPP**, соединяющий **QoS Router** и **Router1**, и выберите **Edit Attribute**. Проверьте **port a** и **port b** и выясните, какой интерфейс используется в **QoS Router** (в нашем случае **IF10**). Ваша конфигурация может отличаться от представленной на рисунке 7.88, это зависит от того, как вы расположили линк **PPP**. Нажмите **OK** для закрытия окна.

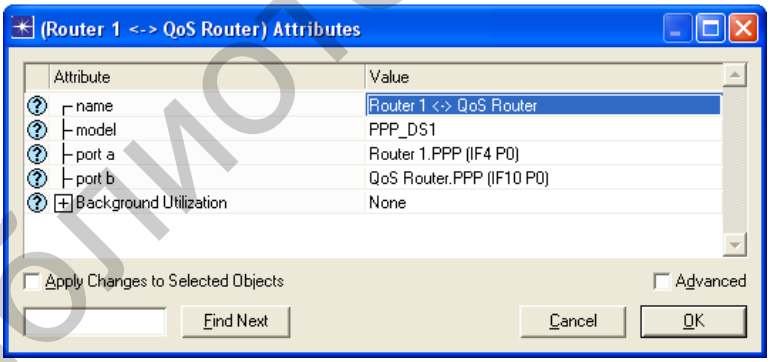


Рисунок 7.88 – Окно настройки **QoS router**

Щелкните правой кнопкой мыши на **QoS Router** и выберите **Edit Attribute**. Раскройте **IP Routing Parameters**, затем **Interface Information**, найдите **row 10**. Раскройте **QoS Information**. Отредактируйте **Buffer Size (bytes)**, установив значение (**value**) **100000** (рисунок 7.89).

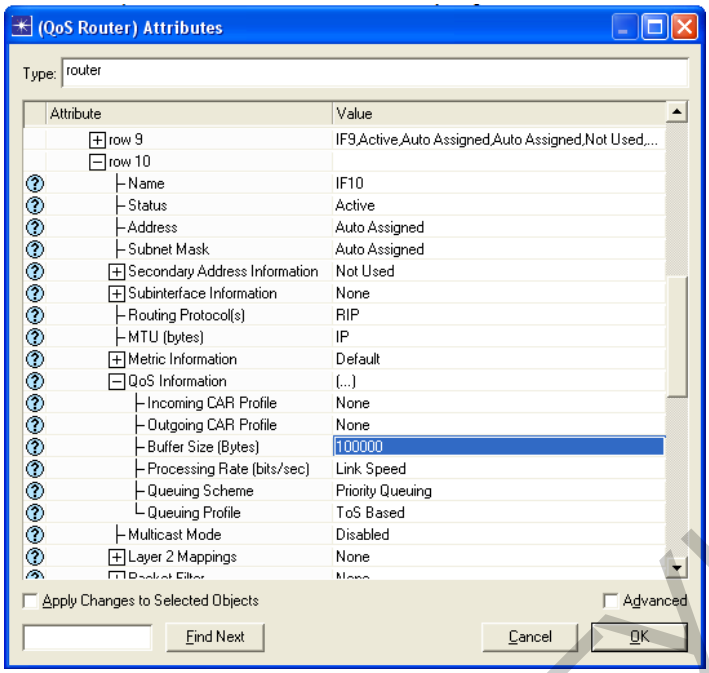


Рисунок 7.89 – Задание атрибутов **QoS Router**

Тем самым мы делаем размер буфера интерфейса сравнительно малым, переполнение буфера становится более частым. Это позволяет нам увидеть более четкую картину результатов использования различных механизмов создания очереди. В конечном счете переполнение случается при передаче сервером **FTP Server** большого количества трафика по линии **DS3** к маршрутизатору **QoS Router**, который передает трафик клиентам по линии **DS1**, не имеющей такой полосы пропускания как **DS3**.

Конфигурирование и выполнение. Щелкните правой кнопкой мыши на **QoS Router** и выберите **Choose Individual Statistics**. Раскройте **IP Interface**. Щелкните правой кнопкой мыши на **QoS Router**. Щелкните правой кнопкой мыши на **QoS Router**, выберите **Buffer Usage (packets)**, затем **– Queue Delay Variation (sec)**, **Queuing Delay (sec)** и статистику **Traffic Dropped (packet/sec)** (рисунок 7.90). Нажмите **OK** для закрытия окна.

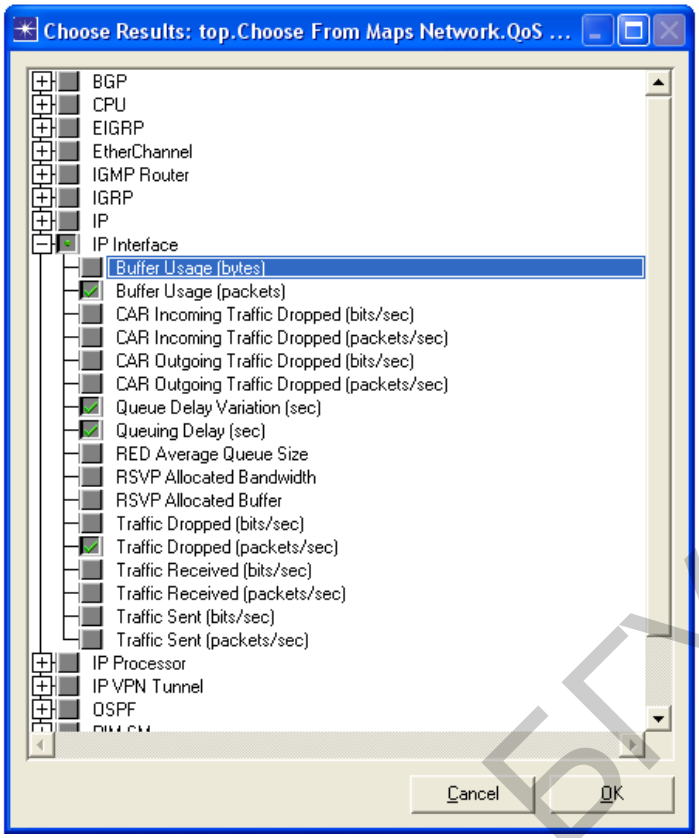


Рисунок 7.90 – Выбор статистики

Нажмите **Simulation/Configure Simulation**. В закладке **Common** установите для **Duration** значение **10**, измеряемое в минутах (рисунок 7.91).

Нажмите **Run** для запуска моделирования. Когда выполнение закончится, нажмите **Close** для закрытия окна.

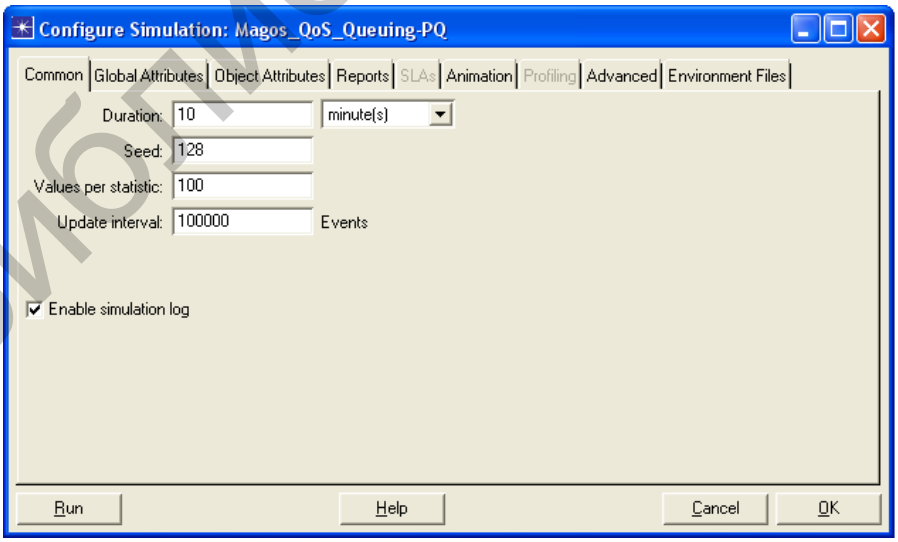


Рисунок 7.91 – Задание параметров моделирования

Проверка и анализ результатов. Выберите закладку **Results/View Results**. Выберите и раскройте **Object Statistics**, затем **Choose From Maps Network**, **QoS Router** и **IP Interface**. Выберите статистики **PQ Traffic Dropped (packets/sec) IF10 Q1** и **PQ Traffic Dropped (packets/sec) IF10 Q0 (Default Queue)**. Отметим, что интерфейс **10 (IF10)** принадлежит **QoS Router** и соединяется с **Router 1**. Если во время установки конфигурации вы обнаружили, что другой интерфейс был использован в модели, замените его интерфейсом **IF10** для оставшейся части анализа результатов. Используйте режим **As Is** для просмотра статистики. Режим **As Is** расположен внизу окна справа. Выбранная статистика показывает, сколько пакетов было утеряно при переполнении буфера. **Q1** соответствует более приоритетному трафику, **Q0 –** менее приоритетному. Отметим, что очередь с более приоритетным трафиком имеет меньшее количество утерянных пакетов, чем с менее приоритетным.

Нажмите на **Object Statistics** снова, чтобы отключить просмотр (рисунок 7.92).

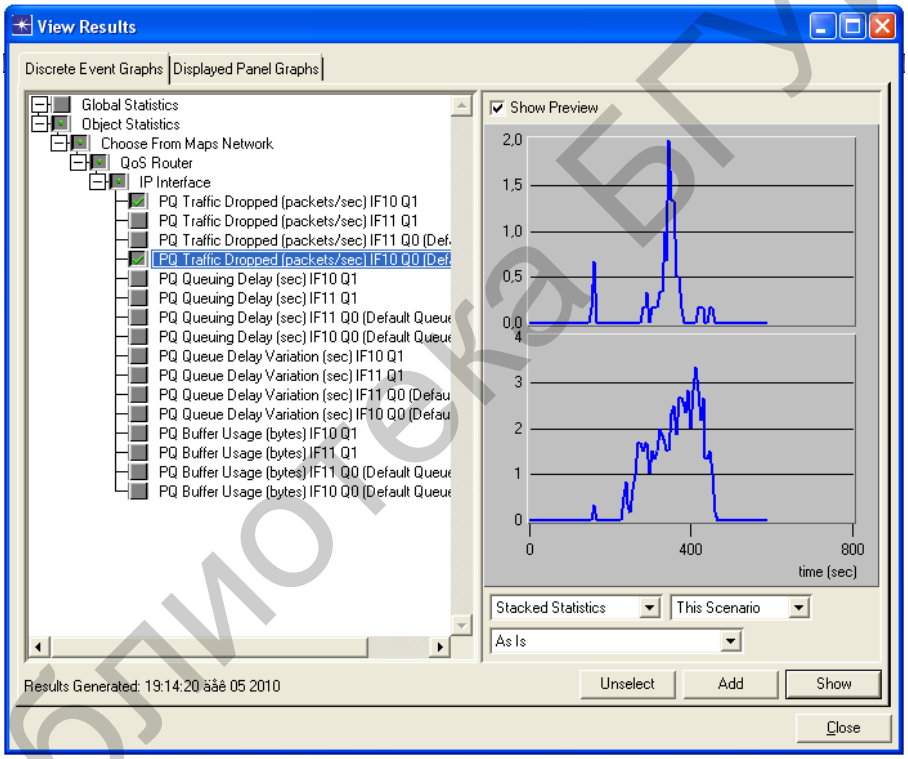


Рисунок 7.92 – Статистика потери пакетов

Выберите статистики **PQ Queuing Delay (sec) IF10 Q1** и **PQ Queuing Delay (sec) IF10 Q0 (Default Queuing)**. Эти статистики показывают, как долго пакеты находились в очереди перед отправкой. Заметим, что пакеты с низким приоритетом находятся дольше в очереди, чем пакеты с высоким приоритетом (рисунок 7.93).

Нажмите на **Object Statistics** снова, чтобы отключить просмотр.

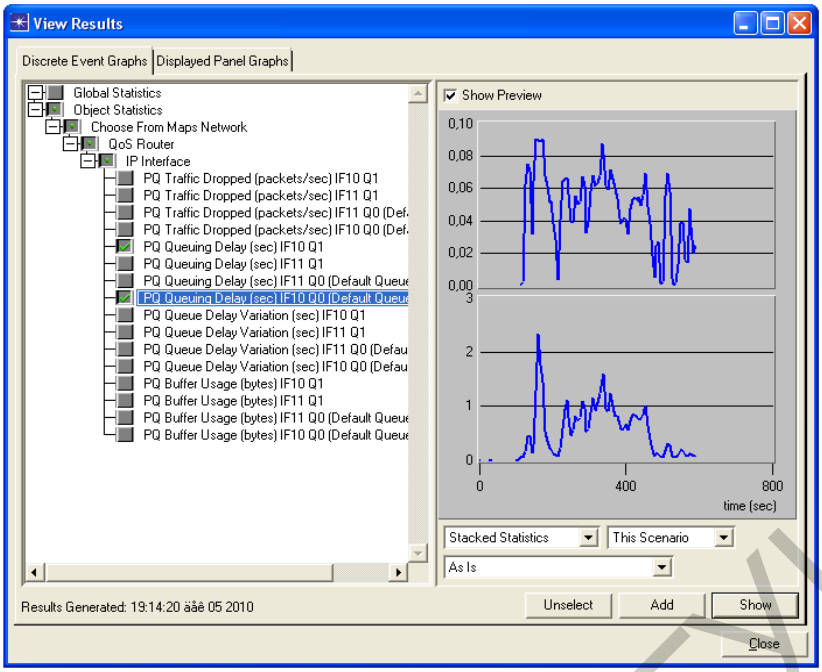


Рисунок 7.93 – График задержки пакетов

Выберите статистики **PQ Queuing Variation (sec) IF10 Q1** и **PQ Queuing Variation (sec) IF10 Q0 (Default Queuing)**. Эти статистики показывают изменения дрожания фаз (джиттер – фазовое отклонение передаваемого сигнала). Заметим, что трафик с низким приоритетом имеет большее значение джиттера, чем трафик с высоким приоритетом (рисунок 7.94).

Нажмите на **Object Statistics** снова, чтобы отключить просмотр.

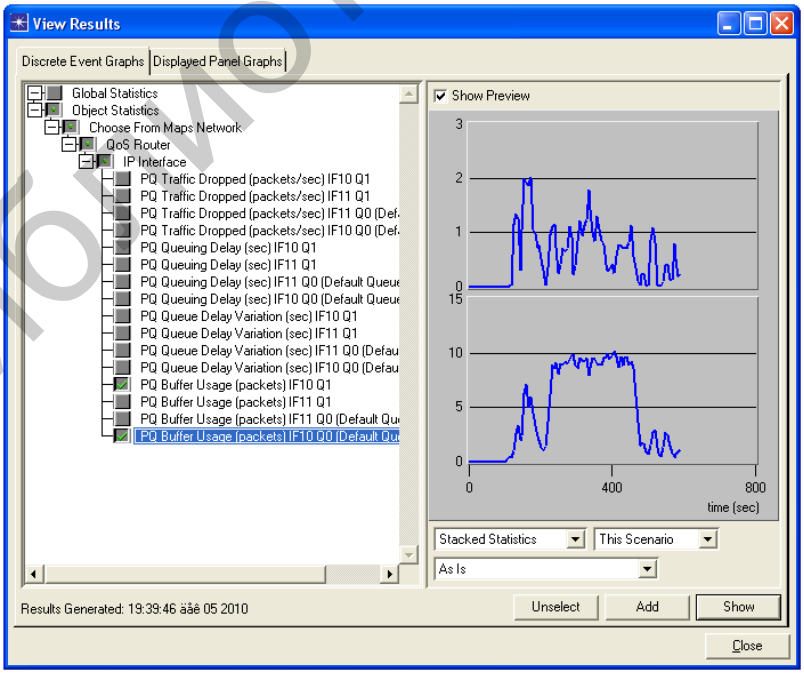


Рисунок 7.94 – График джиттера пакетов

Выберите статистики **PQ Buffer Usage (sec) IF10 Q1** и **PQ Buffer Usage (sec) IF10 Q0 (Default Queuing)**. Эти статистики показывают количество пакетов, стоящих в очереди, во время моделирования. И опять мы видим преимущество пакетов с высшим приоритетом (рисунок 7.95).

Нажмите на **Object Statistics** снова, чтобы отключить просмотр.

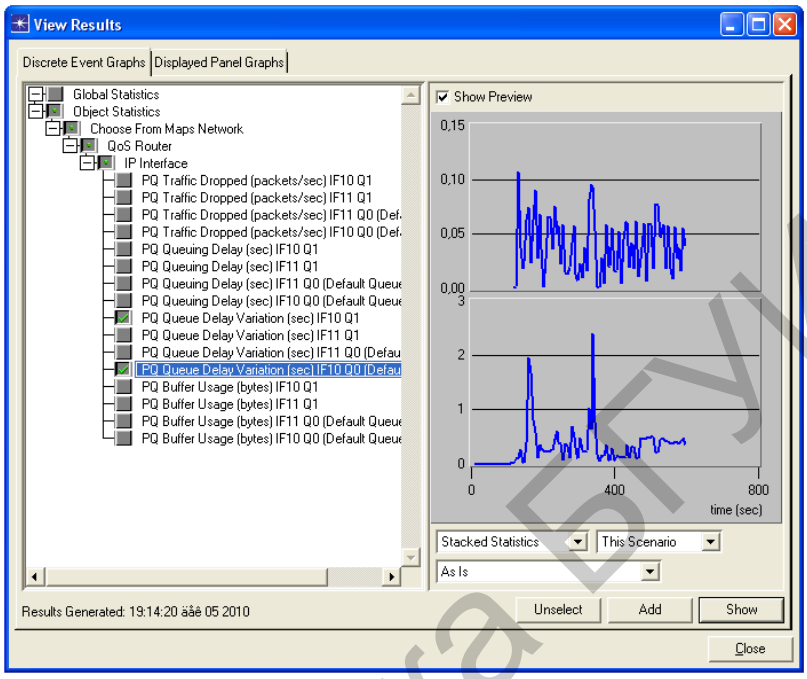


Рисунок 7.95 – Количество пакетов в очереди

Сохраните свою модель и закройте все окна.

Продублируйте ваш сценарий и назовите его **WFQ** (Weighted Fair Queuing – «честная очередь с весовыми коэффициентами»). Выберите два маршрутизатора и перейдите к закладке **Protocols/IP/QoS/Configure QoS**. Для **QoS Scheme** выберите **WFQ**. Напомним, что **ToS Based** было выбрано для **QoS Profile**. Это означает, что маршрутизатор использует поле **Type of Service** в заголовке IP-пакета для определения пакета с приоритетом. Далее выберите **Apply the above selection to** в представленном списке **Interfaces on selected router(s)**.

Нажмите **OK** для закрытия окна.

Запустите повторно моделирование и исследуйте уровень потерь пакетов, задержку, джиттер и загруженность буфера.

Объясните свои результаты.

Продублируйте ваш сценарий и назовите его **FIFO**. Выберите два маршрутизатора и перейдите к закладке **Protocols/IP/QoS/Configure QoS**. Для **QoS Scheme** выберите **FIFO**. При организации механизма **FIFO** для маршрутизатора все пакеты являются одинаковыми, нет никаких приоритетов.

Далее выберите в представленном списке **Apply the above selection to: Interfaces on selected router(s)**.

Нажмите **OK** для закрытия окна.

Запустите повторно моделирование и исследуйте уровень потерь пакетов, задержку, джиттер и загруженность буфера.

Сравните «честную очередь с весовыми коэффициентами» (**Weighted Fair Queuing**), «приоритетную очередь» (**Priority Queuing**) и механизм образования очереди «первый вошел – первый вышел» (**First-In, First-Out**). Назовите преимущества и недостатки каждой из них.

Отредактируйте атрибуты объекта **QoS Parameters**. Посмотрите в **WFQ Profiles** на **ToS Based**. Как значения весового коэффициента связаны с различными значениями **ToS** в **WFQ Scheme**? Как влияют значения этих весовых коэффициентов на работу маршрутизатора?

Отредактируйте атрибуты объекта **QoS Parameters**. Посмотрите в **Priority Queuing Profiles** на **ToS Based**. Сколько уровней приоритета там определено? Как значения **ToS** связаны с каждым уровнем приоритета?

Отредактируйте атрибуты объекта **QoS Parameters**. Посмотрите оба профиля **Priority Queuing Profiles** и **WFQ Profiles**. Как еще, кроме использования поля **ToS**, пакеты могут быть классифицированы для приоритетного обслуживания?

Список использованных источников

1 Советов, Б. Я. Моделирование систем. Практикум : учеб. пособие для вузов / Б. Я. Советов, С. А. Яковлев. – 2-е изд., перераб. и доп. – М. : Высш. шк., 2003. – 295 с.

2 Шелухин, О. И. Моделирование информационных систем : учеб. пособие для вузов / О. И. Шелухин, А. М. Тенякшев, А. В. Осин. – М. : Сайнс-Пресс, 2005. – 364 с.

3 Гургенидзе, А. Т. Мультисервисные сети и услуги широкополосного доступа / А. Т. Гургенидзе, В. И. Кореш. – М. : Наука и техника, 2003. – 390 с.

4 Филимонов, А. Ю. Построение мультисервисных сетей Ethernet / А. Ю. Филимонов. – СПб. : БХВ-Петербург, 2007. – 592 с.

5 Ершов, В. В. Мультисервисные телекоммуникационные сети / В. В. Ершов, Н. А. Кузнецов. – М. : МГТУ им. Н. Э. Баумана, 2003. – 432 с.

6 IT Guru Academic Edition. OPNET Technologies [Электронный ресурс]. – 2005. – Режим доступа : http//www.opnet.com/servise/university/guru_acdemic_edition.html.

7 Росляков, А. В. Сети следующего поколения NGN / А. В. Росляков. – М. : Эко-Трендз, 2008. – 424 с.

8 Бакланов, И. Г. NGN: принципы построения и организации / И. Г. Бакланов ; под ред. Ю. Н. Чернышова. – М. : Эко-Трендз, 2008. – 400 с.

9 Томашевский, В. М. Имитационное моделирование в среде GPSS / В. М. Томашевский, Е. В. Жданова. – М. : Бестселлер, 2003. – 416 с.

10 Хачатурова, С. М. Математические методы системного анализа : учеб. пособие / С. М. Хачатурова. – Новосибирск : НГТУ, 2004. – 124 с.

11 Шрайбер, Т. Дж. Моделирование на GPSS / Т. Дж. Шрайбер. – М. : Машиностроение, 1980. – 593 с.

12 Сайт, посвященный системе моделирования GPSS [Электронный ресурс]. – 2003. – Режим доступа : http://www.gpss.ru.

13 Интернет-университет информационных технологий [Электронный ресурс]. – 2008. – Режим доступа : http://www.intuit.ru.

Св. план 2017, поз. 43

Учебное издание

Лапшин Сергей Михайлович
Цветков Виктор Юрьевич

МОДЕЛИРОВАНИЕ УСТРОЙСТВ И СИСТЕМ ТЕЛЕКОММУНИКАЦИЙ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Е. С. Юрец*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *М. В. Касабуцкий*

Подписано в печать 12.06.2018. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 9,65. Уч.-изд. л. 10,5. Тираж 60 экз. Заказ 357.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6