Оглавление

Введение

Актуальность

Повсеместное внедрение компьютерных сетей, успехи в развитии оптоволоконных и беспроводных средств связи споровождаются непрерывной сменой сетевых технологий, направленной на повышение быстродействия и надёжности сетей. Однако создание опытного образца сети для оценки её эффективности не всегда является оправданным с точки зрения времени и трудоёмкости, поэтому разработка математических моделей является актуальной задачей.

Для непрерывного количественного и качественного роста компьютерных сетей необходимо развитие фундаментальной теории в этой области и создание инженерных методов анализа, направленных на сокращение сроков и повышение качества проектирования компьютерных сетей.

В качестве такой теории выступает теория систем и сетей массового обслуживания. Математические методы этой теории обеспечивают возможность решения многочисленных задач расчёта характеристик качества функционирования различных компонентов компьютерных сетей.

Цель

В данной работе рассматривается анализ критериев времени и надёжности доставки информации в информационно-вычислительных сетях (ИВС) с множественным методом доступа без коллизий, построенных на основе технологий семейства Ethernet.

Задачи

В задачи исследования входит:

1. Изучение методики разработки моделей сетей.
2. Разработка аналитических математических моделей ИВС.
3. Разработка программы для вычисления стационарных и интегральных вероятностных характеристик заданной ИВС.
4. Проведение модельного эксперимента.

Методы

Модельный эксперимент и математические модели фрагментов сетей основываются на математическом аппарате и методах теории систем и сетей массового обслуживания.

Значимость

Разработанная программа должна автоматизировать рутинную работу по вычислению стационарных и интегральных вероятностных характеристик, плотностей распределения сообщений в маршрутах сети и среднего количества маршрутов между любыми двумя узлами сети.

Глава 1. Теоретичские основы разработки математической модели ИВС

1. 1. Однородные экспоненциальные сети

Предметом изучения сетей массового обслуживания (СеМО) являются методы количественного анализа очередей при взаимодействии множества центров обслуживания и потоков сообщений.

СеМО представляет собой совокупность конечного числа M обслуживающих центров, в которой циркулируют сообщения, переходящие в соответствии с маршрутной матрицей (см. 1.1.2) из одного центра сети в другой. Центром обслуживания является система массового обслуживания, состоящая из A ($1 \leqslant A \leqslant \infty$) одинаковых приборов и буфера объёмом C ($0 \leqslant C \leqslant \infty$). Если в момент поступления сообщения все обслуживающие приборы центра заняты, то сообщение занимает очередь в буфере и ожидает обслуживания [1, стр. 90].

В дальнейшем будем полагать, что объём буфера в центре обслуживания $C = \infty$, время обслуживания заявок распределено по экспоненциальному закону, а распределение входящего потока имеет распределение Пуассона.

СеМО с такими распределениями длительности обслуживания и входящего потока являются однородными экспоненциальными сетями или сетями Джексона [1, стр. 94]. Такая модель даёт верхнюю границу оценки (худший вариант) и стационарные вероятности состояний сети имеют муль-

типликативную форму.

В данной работе используются открытые сети Джексона, обрабатывающие F входящих потоков. В открытую сеть сообщения поступают из внешнего источника, могут покидать сеть после завершения обслуживания и интенсивность входного потока не зависит от состояния сети.

1. 1. 1. Пуассоновский поток

Предположение о том, что входящий поток является Пуассоновским, значительно облегчает математические выкладки при достаточной точности.

Пуассоновский поток имеет следующие свойства [8, стр. 12] [7, стр. 7]:

1. Стационарность — вероятность появления k событий на любом промежутке времени зависит только от числа k и от длительности t промежутка.

2. Ординарность — вероятность наступления за элементарный промежуток времени более одного события мала по сравнению с вероятностью наступления за этот промежуток не более одного события и ей можно пренебречь.

3. Независимость — вероятность появления k на любом промежутке времени не зависит от того, появлялись или не появлялись события в моменты времени, предшествующие началу рассматриваемого промежутка.

1. 1. 2. Маршрутная матрица

Маршрутная матрица задаёт структуру соединений узлов сети (топологию) и вероятности переходов сообщения из одного центра сети, после завершения обслуживания в нём, в другой. Для открытой сети в качестве внешнего источника вводится новый центр с индексом 0. Таким образом маршрутная матрица имеет вид $P = \| P_{ij} \|$, где [5, стр. 17]:

$i, j = \overline{0, n}$, n - число узлов в сети,

P_{0j} - вероятность поступления сообщения в M_j узел сети из внешнего источника,

P_{i0} - вероятность покидания сообщением сети после окончания обработки в M_i узле,

$P_{\substack{ij \\ i \neq j}}$ - вероятность перехода сообщения в узел M_j после обработки в M_i.

$P_{00} = 0$.

В маршрутной матрице должно выполняться равенство $\sum_{j=0}^{n} P_{ij} = 1, i = \overline{1,n}$. Т.е. событие, состоящие в том, что сообщение после обработки в узле сети перейдёт в другой узел или покинет сеть — достоверное.

Для сети, обрабатывающей F входящих потоков, необходимо задать F маршрутных матриц P^m, $m = \overline{1,F}$ для каждого входного потока.

1. 2. Методика разработки математической модели ИВС

Одним из самых распространёных методов для разработки аналитической математической модели ИВС является приближённая декомпозиционная модель сети массового обслуживания, основанная на составлении уравнений баланса средних для класса мультипликативных сетей. Эта модель допускает простую декомпозицию всей сети на отдельные элементы и обратную операцию - композицию. Такой подход позволяет проводить анализ каждого фрагмента сети независимо, а затем объединять эти фрагменты, получая обобщённые характеристики.

Первый этап методики состоит в декомпозиции сети на отдельные фрагменты. В зависимости от уровня детализации выделяют 4 уровня декомпозиции [5, стр. 15]:

1. Состоит из набора функциональных элементов (терминалов, моноканалов, канальных станций), каждый из которых может быть представлен в виде отдельной СМО. Самый подробный уровень декомпозиции.

2. Учитывает особенности взаимодействия отдельных элементов 1-го

уровня в пределах всей ИВС.

3. Учитывает взаимодействие нескольких ИВС элементарных топологий (физическая шина, физическое кольцо), связанных между собой в единую ИВС простой топологии (дерево, звезда, и т.д.).

4. Учитывает взаимодействие любых ИВС 2-го и 3-го уровня, связанных в единую ИВС произвольной топологии.

Декомпозиция позволяет преодолеть трудности анализа ИВС большой размерности за счёт разделения ИВС на иерархический набор более простых моделей [3, стр. 11]

Второй этап методики независимо от уровня декомпозиции заключается в разработке отдельных математических моделей всех составляющих на всех уровнях декомпозиции и состоит из следующих подэтапов [5, стр. 16]:

1. Составление уравнений баланса интенсивностей потоков.

2. Вычислеие коэффициентов передачи из уравнений баланса.

3. Вычисление стационарных вероятностно-временных характеристик (ВВХ) для каждого отдельного элемента СеМО.

4. Вычисиление интегральных ВВХ при взаимодействии двух любых абонентов сети.

Исходными параметрами модели являются интенсивности обслуживающих узлов сети μ_i^m, интенсивности поступления сообщений из внешнего источника λ_i^m и маршрутная матрица P^m для каждого входного потока $m = \overline{1, F}$.

1. 2. 1. Уравнения баланса

Уравнения баланса позволяют найти общие интенсивности потоков $\lambda_i'^m$ сообщений в стационарном режиме открытой СеМО (стационарным режимом называется состояние сети, при $t \to \infty$ и $\rho \leqslant 1$ (см. 1.2.2)).

$$\lambda_i^{'m} = e_i^m \lambda_0^m$$

e_i^m - коэффициенты передачи, получаемые при решении уравнений баланса, $\lambda_0^m = \sum_{i=1}^{n} \lambda_i^m$ - суммарная интенсивность всех внешних потоков типа m.

Общая интенсивность птоков складывается из интенсивностей поступления сообщений в M_i узел из внешнего источника $P_{0i}^m \lambda_0^m$, $P_{0i}^m = \frac{\lambda_j^m}{\lambda_0^m}$ и интенсивностей поступления сообщений от других узлов $e_j^m P_{ji}^m \lambda_0^m$ [5, стр. 17].

$$e_i^m \lambda_0^m = P_{0i}^m \lambda_0^m + \sum_{j=1}^{n} e_j^m P_{ji}^m \lambda_0^m, i = \overline{1,n}, m = \overline{1,F}$$

Видно, что можно сократить обе части уравнения на λ_0^m.

$$e_i^m = P_{0i}^m + \sum_{j=1}^{n} e_j^m P_{ji}^m, i = \overline{1,n}, m = \overline{1,F}$$

$$\Updownarrow$$

$$\begin{cases} e_1^m = P_{01}^m + e_1^m P_{11}^m + \cdots + e_n^m P_{n1}^m \\ \vdots \\ e_n^m = P_{0n}^m + e_1^m P_{1n}^m + \cdots + e_n^m P_{nn}^m \end{cases}$$

После решения этой системы уравнений получаем e_i^m для каждого узла, что позволяет рассчитать $\lambda_i^{'m}$.

1. 2. 2. Коэффициент загрузки

Коэффициент загрузки для узла M_i вычисляется по формуле

$$\rho_i = \sum_{m=0}^{F} \rho_i^m, \ \rho_i^m = \frac{\lambda_i^{'m}}{\mu_i^m}, \ i = \overline{1,n}$$

ρ_i^m- коэффициент использования узла, характеризующий соотношение интенсивности входящего потока к интенивности обработки [?, стр. 34].

Для существования стационарного распределения числа сообщений в системе необходимо выполнение условия

$$0 \leqslant \rho_i, \rho_i^m \leqslant 1,\ i = \overline{1,n}$$

что согласуется с интуитивными соображениями: для того, чтобы в системе не накапливалась бесконечная очередь, необходимо, чтобы в среднем сообщения в системе обслуживались быстрее, чем они туда поступают [1, стр. 35].

1. 2. 3. Стационарные вероятностно-временные характеристики

Для каждого узла M_i сети определяются четыре вероятностно-временные характеристики [5, стр. 19]:

1. Средняя длительность ожидания обслуживания.

$$W_i = \frac{\frac{1}{2}\sum\limits_{m=0}^{F}\frac{\rho_i^m(1+{V_i^m}^2)}{\mu_i^m}}{1-\rho_i} = \left| \begin{array}{l} V_i^m = 1 \\ \text{для распределения Пуассона} \end{array} \right| = \frac{\sum\limits_{m=0}^{F}\frac{\rho_i^m}{\mu_i^m}}{1-\rho_i}$$

   V_i^m - коэффициент вариации времени обработки.

2. Средняя длительность пребывания сообщения в узле для потока m.

$$U_i^m = W_i + \frac{1}{\mu_i^m}$$

   $\frac{1}{\mu_i^m}$ - время обработки сообщения.

3. Средняя длина очереди сообщений в узле для потока m.

$$L_i^m = {\lambda_i'}^m W_i$$

4. Среднее число сообщений в узле для потока m, хакатеризующее количество сообщений, находящихся в очереди на обработку, и количество

сообщений, обрабатывающихся в узле.

$$N_i^m = \lambda_i^{'m} U_i^m$$

Эти формулы справедливы для многих моделей СМО и называются формулами Литтла [1, стр. 37].

1. 2. 4. Интегральные вероятностно-временные характеристики

Для определения интегральных ВВХ используются стационарные ВВХ, полученные для каждого узла сети, и анализ маршрутов движения сообщений между двумя абнентами A_i и A_j, $i \neq j$.

Любой маршрут между двумя любыми абонентами принадлежит к одному из трёх типов [5, стр. 20]:

1. Последовательная обработка сообщений на конечном числе элементов сети.

Рис. 1.1. Последовательная обработка

Маршрут состоит из последовательно соединённых узлов сети (рис. 1.1). Интегральные характеристики маршрута вычисляются как сумма соответсвующих характеристик узлов, входящих в маршрут.
Среднее время ожидания обслуживания в маршруте.

$$W = \sum_{i=1}^{n} W_i$$

Среднее время пребывания требования в маршруте для потока m.

$$U^m = \sum_{i=1}^{n} U_i^m$$

Средняя длина очереди требований в маршруте для потока m.

$$L^m = \sum_{i=1}^{n} L_i^m$$

Среднее число требований в маршруте для потока m.

$$N^m = \sum_{i=1}^{n} N_i^m$$

2. Параллельные варианты обработки с определёнными значениями вероятности выбора рассматриваемого варианта.

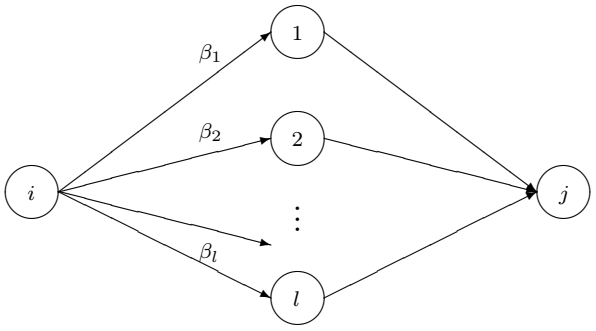

Рис. 1.2. Параллельные варианты

Маршрут зависит от одного из выбранного параллельного варианта (рис. 1.2). Заданы параллельные варианты обработки с вероятностями выбора (см. 1.2.5) β_k, $k = \overline{1,l}$, l - количество альтернатив и должно выполняться нормирующее условие $\sum\limits_{k=1}^{l} \beta_k = 1$.

Интегральные ВВХ определяются следующим образом.

Среднее время ожидания обслуживания в маршруте.

$$W = W_i + \sum_{k=1}^{l} \beta_k W_k + W_j$$

Среднее время пребывания требования в маршруте для потока m.

$$U^m = U_i^m + \sum_{k=1}^{l} \beta_k U_k^m + U_j^m$$

Средняя длина очереди требований в маршруте для потока m.

$$L^m = L_i^m + \sum_{k=1}^{l} \beta_k L_k^m + L_j^m$$

Среднее число требований в маршруте для потока m.

$$N^m = N_i^m + \sum_{k=1}^{l} \beta_k N_k^m + N_j^m$$

3. Комбинация первых двух типов.

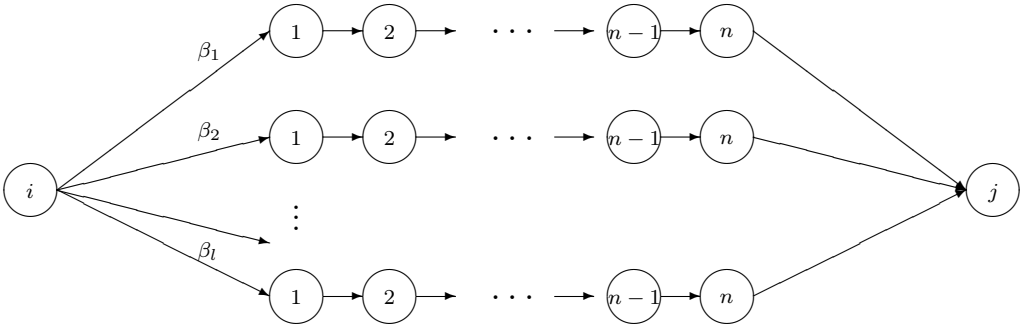

Рис. 1.3. Комбинированный тип обработки

Маршрут зависит от одного из выбранных параллельных маршрутов (рис. 1.3).

Комбинированный тип обработки включает в себя рассмотрение сложных альтернатив, представляющих собой конечную комбинацию двух предыдущих топологических типов.

Интегральные ВВХ в этом случае определяются как в предыдущем случае, но вместо альтернативных вариантов рассматриваются альтернативные маршруты.

Среднее время ожидания обслуживания в маршруте.

$$W = W_i + \sum_{k=1}^{l} \left(\beta_k \sum_{x=1}^{n} W_x \right) + W_j$$

Среднее время пребывания требования в маршруте для потока m.

$$U^m = U_i^m + \sum_{k=1}^{l} \left(\beta_k \sum_{x=1}^{n} U_x^m \right) + U_j^m$$

Средняя длина очереди требований в маршруте для потока m.

$$L^m = L_i^m + \sum_{k=1}^{l} \left(\beta_k \sum_{x=1}^{n} L_x^m \right) + L_j^m$$

Среднее число требований в маршруте для потока m.

$$N^m = N_i^m + \sum_{k=1}^{l} \left(\beta_k \sum_{x=1}^{n} N_x^m \right) + N_j^m$$

1. 2. 5. Вероятность выбора маршрута

Для вычисления интегральных ВВХ необходимо определить вероятности выбора альтернативных маршрутов (см. 1.2.4).

Для этого нужно учитывать вероятности перехода между узлами, заданные маршрутной матрицей P^m (см. 1.1.2), и нормирующее условие

$$\sum_{k=1}^{l} \beta_k = 1$$

указывающее, один из маршрутов будет обязательно выбран.

Вероятность выбора маршрута определяется отношением произведения вероятностей перехода требований из узла $M_{R_j^i}$ в узел $M_{R_{j+1}^i}$ i - го маршрута к сумме произведений вероятностей переходов требований всех альтерна-

тивных маршрутов (нормировочной величине).

$$\beta_i = \frac{\prod\limits_j P_{R_j^i, R_{j+1}^i}}{S}$$

$S = \sum\limits_i \left(\prod\limits_j P_{R_j^i, R_{j+1}^i} \right)$ - нормировочная величина.

R^i - совокупность узлов, составляющих альтернативный маршрут i.

$P_{R_j^i, R_{j+1}^i}$ - вероятность перехода между узлами, задаваемая маршрутной матрицей.

1. 2. 6. Плотность распределения количества сообщений в маршруте

Плотность распределения количества сообщений для произвольного маршрута определяется следующим способом:

$$g_i(t) = \sum_{i=1}^{n} H_i(\mu_i - \lambda_i^{'}) e^{-(\mu_i - \lambda_i^{'})t}$$

$$H_i = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{\mu_j - \lambda_j^{'}}{\mu_j - \lambda_j^{'} - \mu_i - \lambda_i^{'}}$$

где n - количество узлов в маршруте.

1. 3. Определение интенсивностей обслуживающих приборов, работающих на основе технологий семейства Ethernet

Для определения интенсивности обслуживания μ, которая характеризует максимальное количество кадров в секунду [1] , обрабатываемых обслуживающим прибором, рассмотрим кадр формата Ethernet Version 2, т.к. на

[1]При указании производительности сетей термины "кадр"и "сообщение"обычно используются как синонимы. Следовательно единицы измерения производительности кадр/с и сообщение/с являются аналогичными. [4]

практике в оборудовании Ethernet используется только он [6, стр. 361].

Стандарт Ethernet определяет длину служебных полей кадра и преамбулу по 18 и 8 байт соответственно. От сюда следует что длина кадра в битах равна $8 * (X + 26)$, X - длина поля данных. Поле данных может иметь длину от 46 до 1500 байт [6, стр. 361]. Таким образом минимальная длина кадра равна $46 + 26 = 72$ байта, а максимальная 1526 байт.

Время передачи битов кадра определяется битовыми интервалами $bt = \frac{1}{S}$, S - битовая скорость. Для передачи кадра длиной X потребуется время равное $8 * (X + 26) * bt$. Прибавив межкадровый интервал $IFG = 96 * bt$ получим период следования кадров с длиной поля данных X.

$$T = 8 * (X + 26) * bt + IFG$$

Итенсивность обслуживания μ, которая характеризует максимальное количество кадров в секунду, обрабатываемых обслуживающим прибором, есть величина обратно пропорциональная периоду T.

$$\mu = \frac{1}{T} = \frac{1}{8 * (X + 26) * bt + IFG}$$

Проведя не сложные расчёты можно получить интенсивности обслуживания для пакетов различной длины с учётом выбранной технологии передачи данных (табл. 1.1).

1. 4. Структурная надёжность сетей

Надёжность какого-либо объекта – свойство, заключающееся в способности выполнять поставленные задачи в определённых условиях эксплуатации. Состояние объекта, при котором он способен выполнять заданные функции, сохраняя значения основных параметров в пределах, установленных нормативно-технической документацией, называют работоспособностью, а состояние, в котором объект удовлетворяет указанным требованиям, – его исправностью. Событие, заключающееся в нарушении работоспособности объекта, называют отказом.

Технология Ethernet	Битовая скорость	Длина кадра (байт)	Интенсивность μ (кадр/мс)
Fast Ethernet	100 Мбит/с	72	148.800
		1526	8.127
Gigabit Ethernet	1 Гбит/с	72	1488.095
		1526	81.274
10G Ethernet	10 Гбит/с	72	14880.952
		1526	812.744
40G Ethernet	40 Гбит/с	72	59523.800
		1526	3250.975
100G Ethernet	100 Гбит/с	72	148809.524
		1526	8127.438

Таблица 1.1. Интенсивности обслуживания для пакетов различной длины с учётом выбранной технологии передачи данных

Для информационых сетей, являющихся сложными системами, состоящими из элементов разнородных по своим свойствам, показателям надёжности, назначению, дате изготовления, сроку ввода в эксплуатацию и т.п., выделяют два основных аспекта надёжности [9]:

1. Аппаратурный. Под ним понимают проблему надёжности отдельных элементов, входящих в узлы и линии сети.

2. Структурный. Связан с возможностью существования в сети путей доставки информации.

В данной работе рассматривается структурная надёжность, определяемая средним количеством маршрутов маршрутов между двумя произвольными вершинами сети. Эта характеристика вполне хорошо отражает структурную надёжность сети, т.к. чем больше альтернативных маршрутов существует между любыми двумя узлами сети, тем при большем количестве неисправных участков сети она будет оставаться работоспособной и способной передачать сообщени между любыми двумя узлами. Более строгие модели надёжности сетей будут рассмотрены в магистерской диссертации.

Глава 2. Программная реализация

2. 1. Постановка задачи

Программа должна удовлетворять следующим требованиям:

1. Вычислять стационарные и интегральные ВВХ для заданной пользователем модели ИВС.
2. Проверять корректность введённой пользователем модели ИВС.
3. При невозможности вычисления каких-либо характеристик сообщать об этом пользователю.

Программа будет выполнена в виде консольного приложения, т.к. этого достаточно для поставленной задачи и позволяет сконцентрироваться на реализации и достижении поставленных требований. Также это позволит программе без проблем запускаться на различных операционны системах.

2. 2. Программная платформа

Выбрана программная платформа Mono — кроссплатформенная реализация Microsoft .Net Framework с открытым исходным кодом. Mono поддерживает Windows, Linux, BSD, Mac OS X и другие операционные системы и множетсво процессорных платформ, что позволяет писать переносимые приложения на C#. Среда разработки — MonoDevelop.

2. 3. Описание модели ИВС

Модель ИВС одонозначно задаётся интенсивностями обслуживания μ_i^m, интенсивностями поступления сообщений λ_i^m и маршрутной матрицей P^m для каждого входного потока $m = \overline{1, F}$ (см. 1.2). Таким образом надо решить как пользователь будет передавать программе описание модели ИВС.

Для этой цели используется XML файл, в котором задаются параметры модели. XML имеет простой синтаксис, удобный как для составления и чтения файлов человеком, так и для машинной обработки и генерации.

```
1   <NetworkConfiguration Name="Linear topology">
2     <RoutingMatrix>
3           <Row>0; -;  -;  -;  -</Row>
4           <Row>0.25;  0;  0.75;   0;  0</Row>
5           <Row>0.25;  0.375;  0;  0.375;  0</Row>
6           <Row>0.25;  0;  0.375;  0;  0.375</Row>
7           <Row>0.25;  0;  0;  0.75;   0</Row>
8     </RoutingMatrix>
9
10    <Nodes Count="4">
11          <Lambda>17; 14; 13; 16</Lambda>
12
13          <Mu>
14            <Ethernet Type="Gigabit" FrameLength="Max"/>
15          </Mu>
16    </Nodes>
17  </NetworkConfiguration>
```

Рис. 2.1. Описание ИВС линейной топологии из 4-х узлов и одним потоком

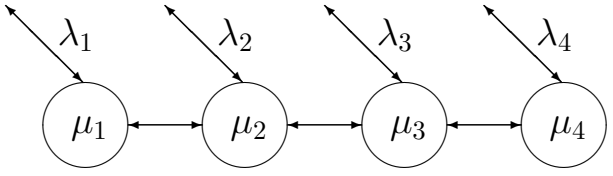

Рис. 2.2. Сеть из 4-х узлов с линейной топологией

На рис. 2.1 показан XML файл, задайющий модель ИВС, состоящей из 4-х узлов с линейной топологией и единственным потоком (рис. 2.2).

При задании интенсивностей обслуживания (рис. 2.2, строка 14) атрибут `Type` обязателен и задаёт технологию Ethernet. Возможные значения этого атрибута Fast, Gigabit, _10G, _40G, _100G. Атрибут `FrameLength` задаёт длину используемых кадров. Его значение может быть Min, Max, что соответствует минимальной (72) или максимальной (1526) длине кадра в байтах, или любое целое число от 72 до 1526.

Интенсивности поступления сообщений из внешнего источника здаются через точку с запятой для каждого узла сети, либо одним целым числом, если интенсивность для всех узлов одна, любо в процентах от интенсивности обслуживания, определённой в элементе `Mu`.

Если в сети несколько потоков, то элементы `Lambda` и `Mu` задаются для каждого потока отдельно внутри элементов `Stream` с атрибутом `Index` (индексация с 0), указывающим к какому потоку относятся интенсивности (пример на рис. 2.8).

В данном случае интенсивность обслуживания μ задаётся для пакетов максимальной длины технологии Gigabit Ethernet (см. 1.3), интенсивность поступления сообщений λ задаётся в явном виде для каждого узла сети (пакеты/мс).

$$\mu = (81.274, 81.274, 81.274, 81.274),\ \lambda = (17, 14, 13, 16)$$

Со 2-й по 8-ю строки задаётся маршрутная матрица.

$$P = \begin{pmatrix} 0 & - & - & - & - \\ 0.25 & 0 & 0.75 & 0 & 0 \\ 0.25 & 0.375 & 0 & 0.375 & 0 \\ 0.25 & 0 & 0.375 & 0 & 0.375 \\ 0.25 & 0 & 0 & 0.75 & 0 \end{pmatrix}$$

Маршрутная матрица это квадратная матрица с размерностью $(n + 1 \times n + 1), n$ - количество узлов сети. Один дополнительный узел это внешний источник.

Маршрутная матрица должна обладать следующими свойствами:

1. на главной диагонали должны быть нули

2. сумма каждой строки должна быть равна 1

Это гарантирует, что сообщение после обработки в узле его покинет.

Прочерки в строках задаваемой матрицы указывают программе, что надо вычислить соответствующие вероятности. Прочерки в первой строке указывают на необходимость вычисления вероятностей поступления сообщений из внешнего источника. Если прочерки есть и в других строках, то в соответствии со свойством 2 алгоритм присвоит пропускам значение

$$\frac{1 - \text{сумма элементов строки}}{\text{количество пропусков}}$$

В данном случае будут вычисляться только вероятности поступления сообщений из внешнего источника.

Парсинг (разбор) XML файла осуществляется с помощью стандартных средств языка C# из пространств имён System.Xml и System.Xml.Linq.

2. 4. Аргументы командной строки

При вызове программе передаются несколько обязательных аргументов и один опциональный.

Обязательные аргументы `netconfig`, `startnode` и `targetnode` задают путь к конфигурационному файлу, индексы начального и конечного узлов (индексация с 0) соответственно.

Необязательный аргумент `log` обозначает необходмость создать лог, в который будет выведена вся информация, если вычисления закончились успешно, `console` задаёт необходимость вывода информации в консоль, а `matrix` указывает, надо ли выводить маршрутную матрицу. Аргумент `matrix` полезен, когда маршрутная матрица имеет большую размерность и не корректно отображается при выводе.

Аргументы могут идти в любом порядке. Если обязательный аргумент пропущен, то произойдёт исключение и в консоли появится текст ошибки и какой аргумент пропущен.

2. 5. Алгоритм работы программы

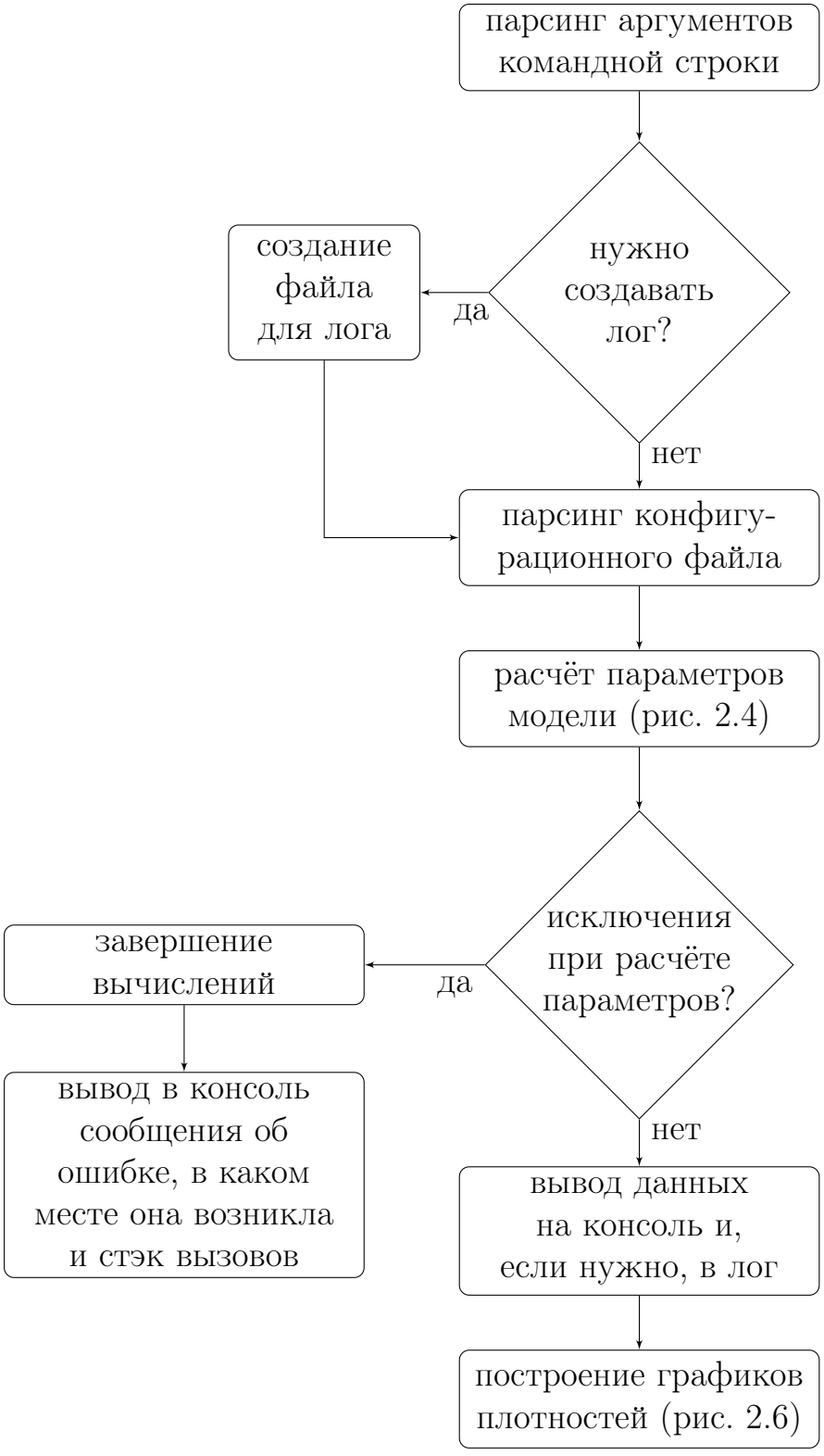

Рис. 2.3. Общая блок-схема программной реализации

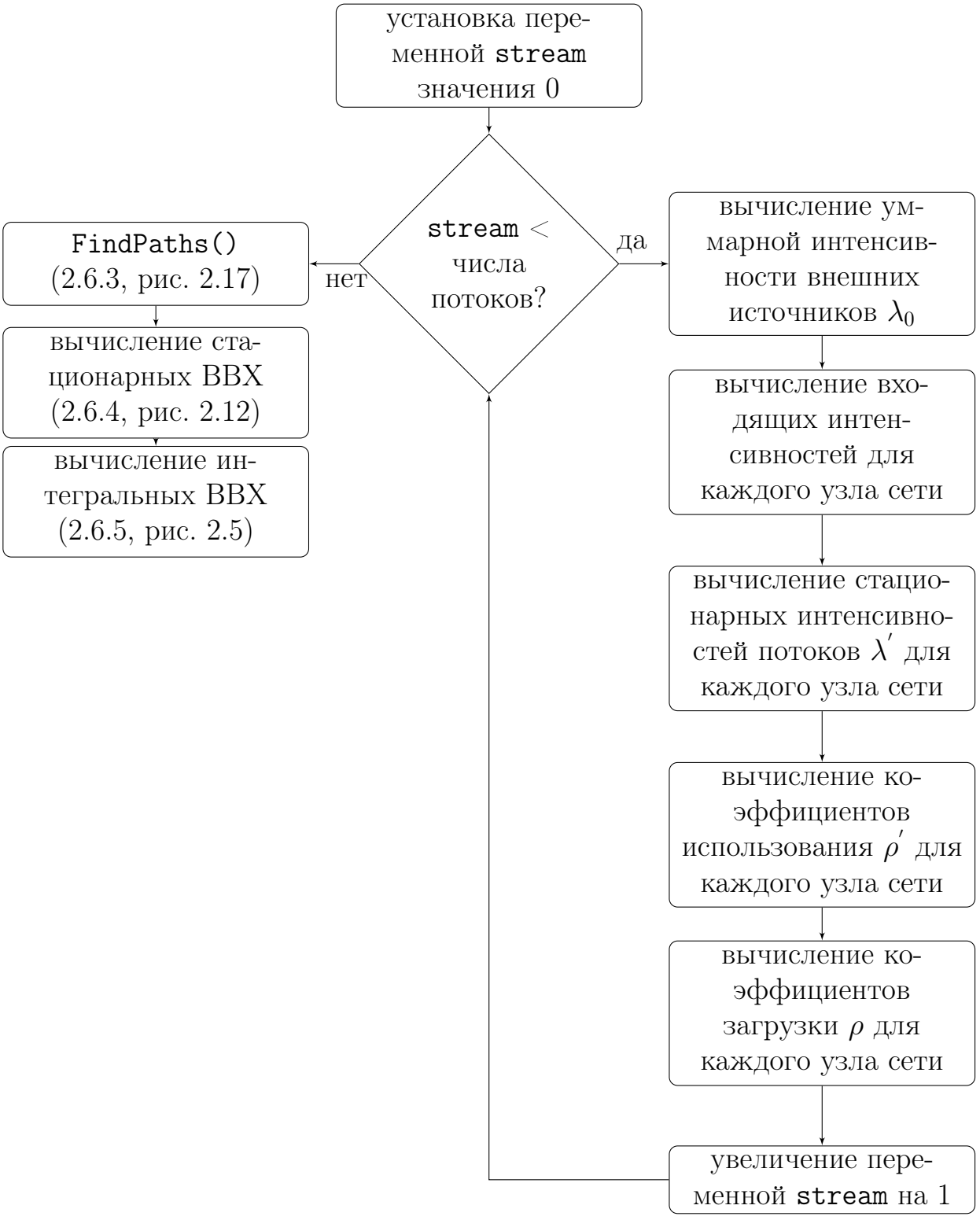

Рис. 2.4. Рассчёт параметров модели

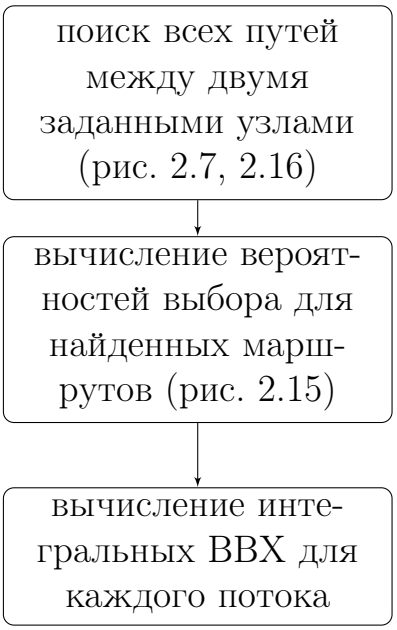

Рис. 2.5. Рассчёт интегральных ВВХ

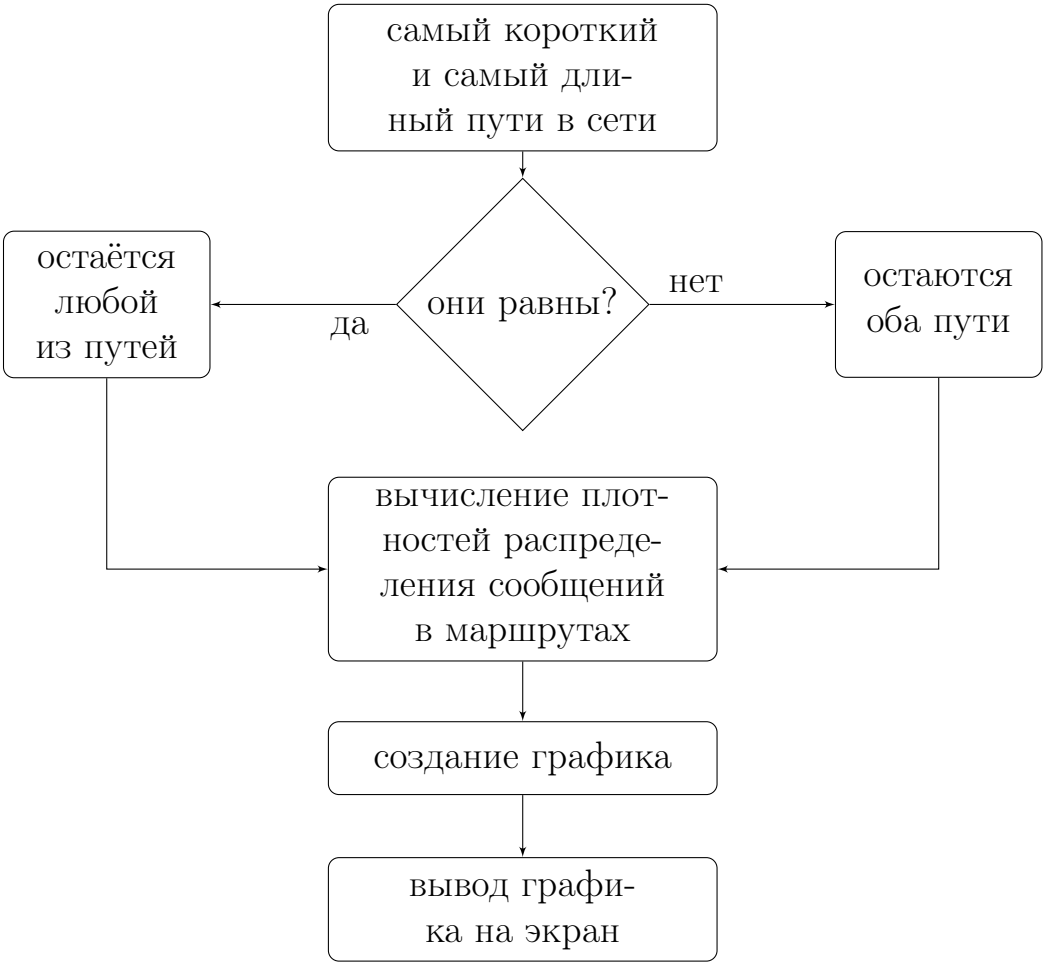

Рис. 2.6. Построение графиков плотностей

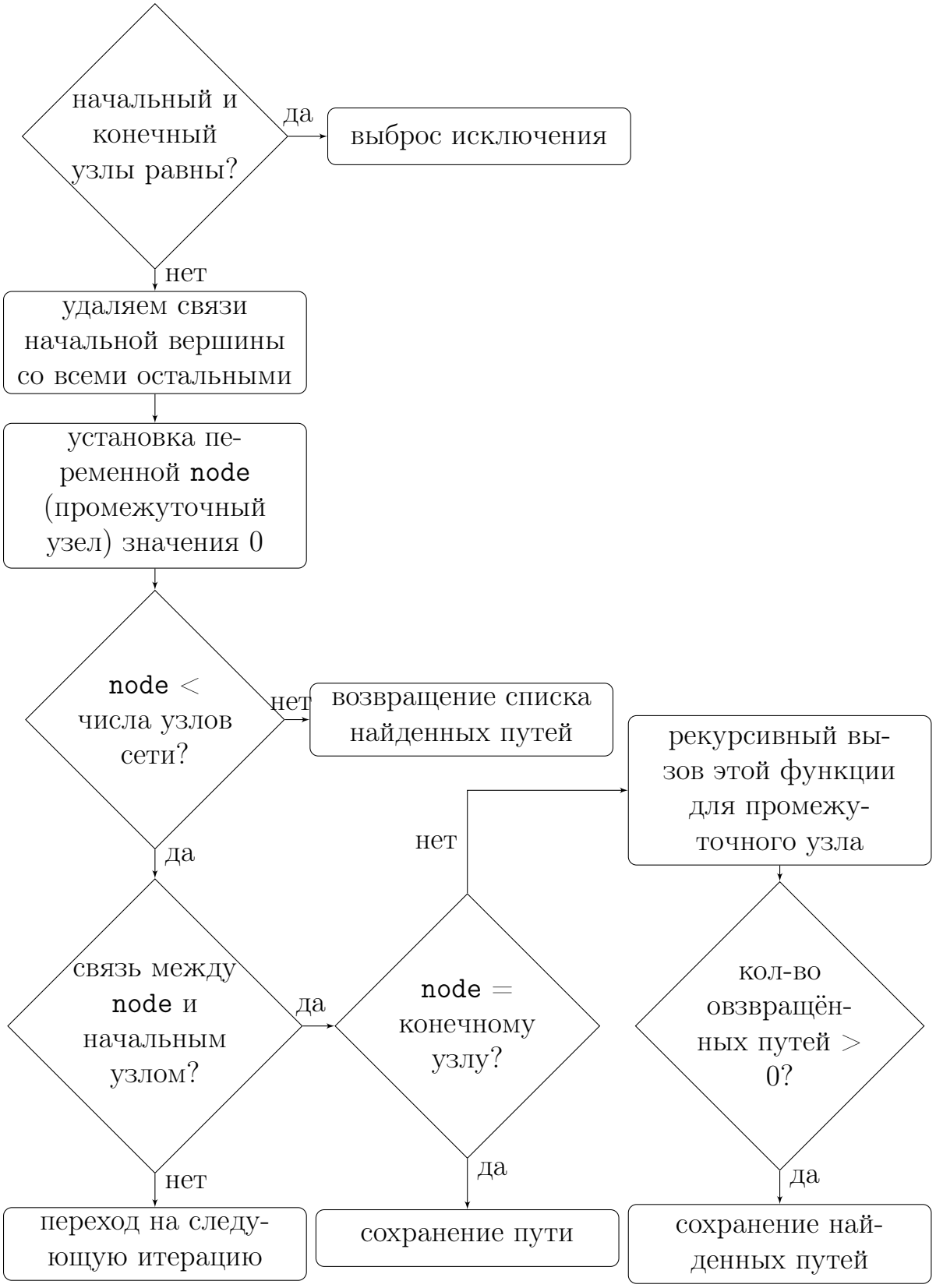

Рис. 2.7. Поиск всех путей между двумя узлами сети

2. 6. Основные моменты реализации

2. 6. 1. Функция расчёта параметров модели

Функция `ComputeParameters()` (рис. 2.10) основная функция, которая после парсинга конфигурационного файла высчитывает следующие параметры:

- суммарная интенсивность внешних потоков λ_0^m (рис. 2.10, строка 4)
- вероятности поступления сообщений из внешнего источника $P_{0,j}^m$ (рис. 2.10, строка 11)
- коэффициенты передачи e_i^m (рис. 2.10, строка 14)
- общие интенсивности потоков $\lambda_i'^m$ (рис. 2.10, строка 17)
- коэффициенты использования узлов ρ_i^m и коэффициенты загрузки ρ_i (рис. 2.10, строка 20)
- поиск всех путей между двумя заданными узлами, поис минимального и максимального путей в сети, поиск среднего числа путей между любыми двумя узлами сети (рис. 2.10, строка 33).

Эти параметры дают достаточно инормации для дальнейших вычислений стационарных и интергральных ВВХ и плотностей распределения сообщений в маршрутах.

2. 6. 2. Составление и решение уравнений баланса

Расширенная матрица системы уравнений баланса получается из матрицы маршрутизации следующим способом:

1. Транспонируем матрицу маршрутизации.
2. Вычёркиваем нулевую строчку.
3. Умножаем нулевой столбец на -1.

4. Переносим нулевой столбец в конец.

5. Проставляем -1 на главной диагонали.

Получаем расширенную матрицу для системы уравнений следующего вида и решаем её с помощью матода Гаусса:

$$\begin{cases} -e_1^m + e_1^m P_{11}^m + \cdots + e_n^m P_{n1}^m = -P_{01}^m \\ \vdots \\ -e_n^m + e_1^m P_{1n}^m + \cdots + e_n^m P_{nn}^m = -P_{0n}^m \end{cases}$$

В программной реализации (рис. 2.10, строка 15) вызывается функция `GetExtendedMatrix(int streamIndex)` (рис. 2.11), которая преобразует матрицу маршрутизации заданного потока.

2. 6. 3. Функция FindPaths()

Функция `FindPaths()` представлена на рисунке 2.17. Эта функция находит все пути между заданными узлами, минимальный и максимальный пути и среднее количество путей между любыми двумя узлами сети.

Эта функция для каждой пары узлов сети, если они не равны, находит пути между ними (рис. 2.7, 2.16). Если эти узлы — узлы, которые задал пользователь, то все пути сохраняются для дальнейшего их использования при вычислении интегральных ВВХ. Переменная `pathsCount` увеличивается на количество найденных на данной итерации путей, а `pairsCount` инкрементируется на 1. Далее сравниваются минимальный и максимальный пути, найденные на данной итерации, с сохранёнными минимальным и максимальным путями. Если найденный минимальный путь меньше сохранённого, то сохраняем его вместо предыдущего. Если найденный максимальный путь больше сохранённого, то сохраняем его вместо предыдущего. После завершения обхода всех пар узлов, считается среднее количество путей между любыми двумя узлами сети как частное

$$\text{среднее количество путей } = \frac{pathsCount \text{ (количество всех путей в сети)}}{pairsCount \text{ (количество пар узлов)}}$$

2. 6. 4. Вычисление стационарных ВВХ

Функция вычисления стационарных ВВХ представлена на рис. 2.12. Первый цикл, который расположен с 3 по 10 строку вычисляет среднюю длительность ожидания обслуживания W_i для узла i. Эти расчёты проводятся отдельно от остальных, потому что, как видно по формулам (1.2.3), каждому узлу сети соответствует один такой параметр, не зависимо от количества обрабатываемых потоков.

Следующий цикл, расположенный с 12 по 19 строку, считает остальные стационарных ВВХ. Здесь используются реализованные мной функции работы с векторами `AddElementWise` и `MultiplyElementWise`, позволяющие складывать и умножать векторы поэлементно.

2. 6. 5. Вычисление интегральных ВВХ

Вычисление интегральных ВВХ несколько сложнее, чем стационарных. Для их вычисления сначала надо получить вероятности выбора маршрутов, что достигается с помощью функции
`ComputeTransitionProbabilities()` (рис. 2.15). Затем вычисляются интегральные ВВХ с помощью вспомогательной функции
`ComputeIntegralChar` (рис. 2.14), реализующей алгоритм вычисления интегральных ВВХ при комбинированном типе обработки (см. 1.2.4).

2. 6. 6. Поиск всех путей между двумя узлами сети

Основная идея этого алгоритма – последовательный перебор узлов для нахождения путей на основе матрицы смежности. Каждая пройденная вершина учитывается только один раз, что исключает маршрутные петли. Алгоритм представлен на рис. 2.7 (блок-схема) и 2.16 (реализация).

2. 7. Пример работы программы

Рассмотрим работу программы на полносвязной сети из 4-х узлов с двумя потоками сообщений.

Запускаем программу со следующими аргументами:

```
/netconfig:"TestConfigurations/network_demo4.netconfig"
/startnode:0 /targetnode:3 /log
```

```
1   <NetworkConfiguration Name="Full-mesh topology">
2     <RoutingMatrix>
3           <Row>0; -; -; -; -</Row>
4           <Row>0.25; 0; 0.25; 0.25; 0.25</Row>
5           <Row>0.25; 0.25; 0; 0.25; 0.25</Row>
6           <Row>0.25; 0.25; 0.25; 0; 0.25</Row>
7           <Row>0.25; 0.25; 0.25; 0.25; 0</Row>
8     </RoutingMatrix>
9
10    <Nodes Count="4">
11          <Stream Index="0">
12            <Lambda>860; 930; 670; 710</Lambda>
13                  <Mu>
14                    <Ethernet Type="_10G" FrameLength="128"/>
15                  </Mu>
16            </Stream>
17
18          <Stream Index="1">
19            <Lambda>161; 153; 170; 167</Lambda>
20                  <Mu>
21                    <Ethernet Type="_10G" FrameLength="1024"/>
22                  </Mu>
23            </Stream>
24    </Nodes>
25  </NetworkConfiguration>
```

Рис. 2.8. Конфигурационный файл для полносвязной сети из 4-х узлов с двумя потоками

Как видно из конфигурационного файла (рис. 2.8), в данной сети находятся два потока сообщений:

1. Интенсивность поступления сообщений
   $\lambda^1 = (860, 930, 670, 710)$ кадр/мс.

Интенсивность обслуживания μ_i^1 у всех узлов одинаковая. Соответствует максимальной пропускной способности для кадров длиной 128 байт технологии 10G Ethernet и равна 8929.571 кадр/мс.

2. Интенсивность поступления сообщений
   $\lambda^2 = (161, 153, 170, 167)$ кадр/мс.
   Интенсивность обслуживания μ_i^2 у всех узлов одинаковая. Соответствует максимальной пропускной способности для кадров длиной 1024 байт технологии 10G Ethernet и равна 1206.564 кадр/мс.

При запуске мы передали в программу аргумент `log`, значит все результаты вычислений записались в файл, содержимое которого представленно на рис. 2.18 на страницах 38 - 42.

Вывод разделён по потокам и первые строки (4 - 21) занимает уже известная информация о матрице маршрутизации (`ROUTING MATRIX`), интенсивности поступления сообщений (`LAMBDA`, кадр/мс) и интенсивности обслуживания (`MU`, кадр/мс).

Далее (строки 23 - 56) выводится вычисленные суммарные интенсивности внешнего потока (`LAMBDA0`, кадр/мс), коэффициенты передачи (`E`), общие интенсивности потока (`LAMBDA'`, кадр/мс), коэффициенты использования узлов (`RO'`) и коэффициенты загрузки (`ROTOTAL`), среднее число путей между любыми двумя узлами сети (`THE AVERAGE NUMBER OF PATHS`).

Следующая часть посвящена стационарным ВВХ (строки 58 - 81). Здесь находятся средняя длительность ожидания обслуживания в узлах (`W`, мс), средняя длительность пребывания требования в узлах (`U`, мс), средняя длина очереди требований в узлах (`L`, мс) и среднее число требований в узлах (`N`, кадр).

После стационарных идут интегральные ВВХ (строки 82 - 100). Здесь представлены все пути между заданными (в аргументах командной строки) узлами и вероятности их выбора, среднее время ожидания обслуживания в маршруте (`W`, мс), среднее время пребывания требования в маршруте (`U`, мс), средняя длина очереди требований в маршруте (`L`, мс), среднее число требований в маршруте (`N`, кадр).

Всё тоже-самое представленно и для второго потока.

Графики плотностей распределения количества сообщений в маршруте выводятся на экран.

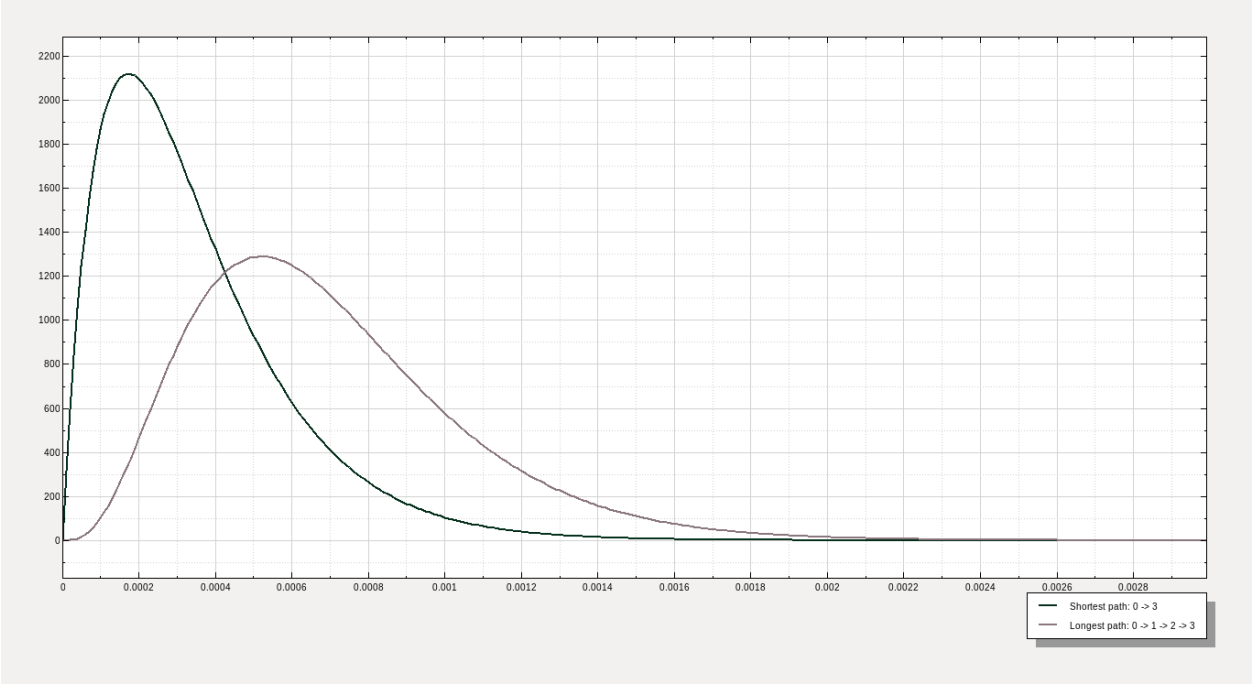

Рис. 2.9. Графики плотностей распределения количества сообщений в самом коротком и самом длинном маршрутах сети

```
1   void ComputeParameters()
2   {
3     // вычисление суммарной интенсивности внешних потоков
4     Lambda.ForEach(
5           lambdas => Lambda0.AddElement(lambdas.Sum()));
6     RoTotal = new Vector<double>(NodesCount);
7
8     for (int stream = 0; stream < StreamsCount; stream++)
9     {
10          // вычисление входных вероятностей
11          InputProbability.Add(Lambda[stream].Clone()
12            .InsertElement(0, 0).Divide(Lambda0[stream]));
13          // вычисление коэфициентов передачи
14          E.Add(
15            GaussMethod.Solve(GetExtendedMatrix(stream)));
16          // вычисление общих интенсивностей потоков
17          LambdaBar.Add(
18            E[stream].Multiply(Lambda0[stream]));
19          // вычисление коэффициентов использования узлов
20          RoBar.Add(
21            LambdaBar[stream].DivideElementWise(Mu[stream]));
22          RoTotal = RoTotal.AddElementWise(RoBar.Last());
23
24          if(RoBar[stream].Any(roBar => roBar > 1))
25            throw new ArgumentOutOfRangeException(
26            string.Format("RoBar, stream {0}", stream),
27            "Some Ro' is greater than zero.");
28    }
29    if (RoTotal.Any(roTotal => roTotal > 1))
30          throw new ArgumentOutOfRangeException("RoTotal",
31          "Some RoTotal is greater than zero.");
32
33          FindPaths();
34  }
```

Рис. 2.10. Функция расчёта параметров модели

```
1   Matrix<double> GetExtendedMatrix(int streamIndex)
2   {
3     var matrix = GetRoutingMatrix(streamIndex);
4     matrix.Transpose();
5     matrix.RemoveRow(0);
6     matrix.AddColumn(matrix.GetColumn(0).Negate());
7     matrix.RemoveColumn(0);
8     matrix.ReplaceDiagonalElements(-1);
9
10    return matrix;
11  }
```

Рис. 2.11. Функция преобразования мартрицы маршрутизации в расширенную матрицу системы уравнений баланса

```
1   void ComputeStationaryPTC()
2   {
3     for (int i = 0; i < NodesCount; i++)
4     {
5           double sum = 0.0;
6           for (int stream = 0; stream < StreamsCount; stream++)
7             sum += RoBar[stream][i] / Mu[stream][i];
8
9           Ws.AddElement(sum / (1 - RoTotal[i]));
10    }
11
12    for (int stream = 0; stream < StreamsCount; stream++)
13    {
14          Us.Add( Ws.AddElementWise(Mu[stream].Pow(-1)) );
15          Ls.Add(
16            LambdaBar[stream].MultiplyElementWise(Ws) );
17          Ns.Add(
18            LambdaBar[stream].MultiplyElementWise(Us[stream]));
19    }
20  }
```

Рис. 2.12. Функция вычисления стационарных ВВХ

```
1  void ComputeIntegratedPTC()
2  {
3    ComputeTransitionProbabilities();
4
5    Wi = ComputeIntegralChar(Ws);
6    for (int stream = 0; stream < StreamsCount; stream++)
7    {
8          Ui.Add(ComputeIntegralChar(Us[stream]));
9          Li.Add(ComputeIntegralChar(Ls[stream]));
10         Ni.Add(ComputeIntegralChar(Ns[stream]));
11   }
12 }
```

Рис. 2.13. Функция вычисления интегральных ВВХ

```
1  double ComputeIntegralChar(Vector<double> staticChar)
2  {
3    var integralChar =
4          staticChar[StartNode] + staticChar[TargetNode];
5
6    for (int i = 0; i < Paths.Count(); i++)
7    {
8          var temp = 0.0;
9          for (int j=1; j < Paths.ElementAt(i).Count()-1; j++)
10           temp += staticChar[j];
11
12         integralChar += TransitionProbabilities[i] * temp;
13   }
14
15   return integralChar;
16 }
```

Рис. 2.14

```
1  void ComputeTransitionProbabilities()
2    {
3          var matrix = RoutingMatrix.Clone().RemoveColumn(0);
4
5          var pathsProbabilities =
6          Paths.Select(
7          path => path.Where(
8          (item, index) => index < path.Length - 1)
9          .Select((item, index) => new {item, index})
10         .Aggregate(1.0, (accumulate, anon) => accumulate *=
11         matrix[anon.item, path[anon.index + 1]]));
12
13         var s = pathsProbabilities.Sum();
14
15         TransitionProbabilities = pathsProbabilities
16         .Select(prob => prob / s).ToList();
17 }
```

Рис. 2.15. Функция вычисления вероятностей выбора маршрутов

```
1  List<Vector<int>> GetPathsBetween(
2    Matrix<double> matrix, int startNode, int targetNode)
3  {
4    if (startNode == targetNode)
5    throw new ArgumentException(
6    "stratNode and targetNode must be different.");
7
8    var paths = new List<Vector<int>>();
9
10   // исключение циклов
11   matrix.ForEachRow(row => row[startNode] = 0);
12
13   for (int intermediateNode = 0;
14   intermediateNode < matrix.RowsCount; intermediateNode++)
15   {
16         if (matrix[startNode, intermediateNode] <= 0)
17           continue;
18
19         if (intermediateNode == targetNode)
20           paths.Add(
21           new Vector<int>(new[] { startNode, targetNode }));
22         else
23         {
24           List<Vector<int>> localPaths = GetPathsBetween(
25           matrix.Clone(), intermediateNode, targetNode);
26           if (localPaths.Any())
27                 localPaths.ForEach(path =>
28                 {
29                   path.InsertElement(0, startNode);
30                   paths.Add(path);
31                 });
32           }
33     }
34
35   return paths;
36 }
```

Рис. 2.16. Функция поиска всех путей между двумя вершинами

```
1   void FindPaths()
2   {
3     var matrix = RoutingMatrix.Clone().RemoveColumn(0);
4     var pathsCount = 0;
5     var pairsCount = 0;
6
7     for (int startNode = 0; startNode < matrix.RowsCount;
8           startNode++)
9           for (int targetNode = 0;
10          targetNode < matrix.RowsCount;targetNode++)
11            if (startNode != targetNode)
12            {
13                  var paths = GraphHelper.GetPathsBetween(
14                    matrix.Clone(), startNode, targetNode);
15
16                  if (startNode == StartNode
17                    && targetNode == TargetNode)
18                      Paths = paths;
19
20                  pathsCount += paths.Count();
21                  pairsCount++;
22
23                  if (MinPath == null
24                    || paths.Min().Length < MinPath.Length)
25                      MinPath = paths.Min();
26                  if (MaxPath == null
27                    || paths.Max().Length > MaxPath.Length)
28                      MaxPath = paths.Max();
29            }
30
31    AveragePaths = pathsCount / pairsCount;
32  }
```

Рис. 2.17. Функция поиска путей, минамального и максимального пути, среднее количество путей

```
NAME = Full-mesh topology
STREAMS COUNT = 2
==================== STREAM 0 ====================
ROUTING MATRIX
0.000   0.271   0.293   0.211   0.224
0.250   0.000   0.250   0.250   0.250
0.250   0.250   0.000   0.250   0.250
0.250   0.250   0.250   0.000   0.250
0.250   0.250   0.250   0.250   0.000

LAMBDA
860.000000
930.000000
670.000000
710.000000

MU
8928.571429
8928.571429
8928.571429
8928.571429

RO
0.096320
0.104160
0.075040
0.079520

LAMBDA0
3170

E
1.017035
1.034700
0.969085
0.979180

LAMBDA'
3224.000000
3280.000000
3072.000000
3104.000000
```

```
RO'
0.361088
0.367360
0.344064
0.347648

ROTOTAL
0.899476
0.900444
0.888420
0.890015

THE AVERAGE NUMBER OF PATHS
5.000000

STATIONARY PROBABILITY-TIME CHARACTERISTICS
W
0.004841
0.004851
0.004389
0.004441

U
0.004953
0.004963
0.004501
0.004553

L
15.608149
15.911909
13.482236
13.785019

N
15.969237
16.279269
13.826300
14.132667
```

```
INTEGRATED PROBABILITY-TIME CHARACTERISTICS
PATHS, 5
0 -> 3 : 0.615384615384615
0 -> 1 -> 3 : 0.153846153846154
0 -> 2 -> 3 : 0.153846153846154
0 -> 1 -> 2 -> 3 : 0.0384615384615385
0 -> 2 -> 1 -> 3 : 0.0384615384615385

W
0.011486

U
0.011761

L
36.550228

N
37.426723
==================================================

==================== STREAM 1 ====================
ROUTING MATRIX
0.000   0.247   0.235   0.261   0.257
0.250   0.000   0.250   0.250   0.250
0.250   0.250   0.000   0.250   0.250
0.250   0.250   0.250   0.000   0.250
0.250   0.250   0.250   0.250   0.000

LAMBDA
161.000000
153.000000
170.000000
167.000000

MU
1206.563707
1206.563707
1206.563707
1206.563707
```

```
RO
0.133437
0.126806
0.140896
0.138410

LAMBDA0
651

E
0.997849
0.988018
1.008909
1.005223

LAMBDA'
649.600000
643.200000
656.800000
654.400000

RO'
0.538388
0.533084
0.544356
0.542367

ROTOTAL
0.899476
0.900444
0.888420
0.890015

THE AVERAGE NUMBER OF PATHS
5.000000

STATIONARY PROBABILITY-TIME CHARACTERISTICS
W
0.004841
0.004851
0.004389
0.004441
```

```
164  U
165  0.005670
166  0.005680
167  0.005218
168  0.005270
169
170  L
171  3.144868
172  3.120287
173  2.882530
174  2.906223
175
176  N
177  3.683256
178  3.653371
179  3.426886
180  3.448590
181
182  INTEGRATED PROBABILITY-TIME CHARACTERISTICS
183  PATHS, 5
184  0 -> 3 : 0.615384615384615
185  0 -> 1 -> 3 : 0.153846153846154
186  0 -> 2 -> 3 : 0.153846153846154
187  0 -> 1 -> 2 -> 3 : 0.0384615384615385
188  0 -> 2 -> 1 -> 3 : 0.0384615384615385
189
190  W
191  0.011486
192
193  U
194  0.013526
195
196  L
197  7.472934
198
199  N
200  8.800595
201  ================================================
```

Рис. 2.18. Лог для полносвязной сети из 4-х узлов с двумя потоками

Глава 3. Модельный эксперимент

3. 1. Задачи

В задачи проведения модельного эксперимента входит сравнение разных топологий сетей по характеристикам времени и надёжности доставки сообщений, определение топологий с самым малым временем доставки сообщений и самой большой надёжностью доставки.

3. 2. Методология

Для качетсвенного проведения эксперимента надо исключить любые факторы, которые могут повлиять на его ход и результаты. В данном случае, для сравнения именно топологий сетей и как они влияют на храктеристики времени и надёжности доставки сообщений, будем сравнивать сети с одним потоком сообщений, с одинаковым количеством узлом, одинаковыми интенсивностями внешних источников и одинаковыми интенсивностями обработки собщений. У всех сравниваемых моделей будет отличаться только структура связей между узлами.

Для определения самой эффективной по времени доставки топологии, сети будем использовать интегральную вероятностно-временную характеристику U (1.2.4), характеризующей среднее время пребывания требования в маршруте.

Для определения топологии, предоставляющей самую надёжную доставку сообщений, будем сравнивать сети по среднему числу путей между любыми двумя узлами сети.

Для этого эксперимента будем использовать модели сетей с 10 узлами, интенсивностью обработки сообщений соответствующей технологии 10G Ethernet с кадрами максимальной длины, интенсивностью поступления сообщений из внешнего источника равной 7% (5.689207 сообщений/мс) от интенсивности обработки и следующими топологиями:

1. Шина
2. Звезда
3. Кольцо
4. Дерево
5. Двухменая квадратная решётка
6. Двухмерная треугольная решётка
7. Трёхмерная квадратная решётся
8. Пирамидальная топология
9. Полносвязная топология

3. 3. Анализ полученных данных

В результате моделирования получены данные, приведённые в таблице 3.1. Из них видно, что от топологии сети зависит многое. При одинаковом числе узлов, входных интенсивностях и интенсивностях обработки сообщений, различные топологии дают совешенно различные результаты по скорости и надёжности доставки сообщений.

У базовых топологий (шина, звезда, кольцо и дерево) самое минимальное возможное среднее число путей, что делает их совершенно не устойчивым к отказам. Шина также имеет самое большое время доставки.

Полносвязная топология самая надёжная с средним числом путей между любыми двумя вершинами 109601, но время доставки сообщений в ней не лучшее.

Топологией с самой быстрой доставкой сообещений (0.008404 мс) — пирамидальная топология.

Пирамидальная топология лучшая по скорости доставки сообщений и вторая по надёжности, а полносвязная лучшая по надёжности и вторая по скорости доставки.

Также можно сравнить топологии по количеству связей, образующих их.

Топологии шина, звезда, кольцо и дерево имеют минимальное необходимое количество связей для сооединения 10-ти узлов. Именно из-за отсутствия избыточности эти топологии имееют самую низкую надёжность.

Если сравнивать полносвязную и пирамидальную топологии по числу связей, то у полносвязной их $\frac{n*(n-1)}{2} = \frac{10*9}{2} = 45$, а у пирамидальной 24, что в 1.875 раза меньше. С этой точки зрения пирамидальная топология выглядит привлекательнее полносвязной, потому что при почти в два раза меньшем количестве связей эта топология имеет лучшее время доставки и второй показатель надёжности.

У двухмерной квадратной решётки, как и у пирамидальной, 24 связи, но хуже время доставки и малое среднее число путей между любыми двумя узлами.

Двухмерная треугольная и трёхмерная квадратная решётки имеют хорошее соотношение между временем доставки, надёжностью и количеством связей.

Топология	Среднее время пребывания требования в маршруте (мс)	Среднее число путей между любыми двумя узлами	Количество связей
Шина	0.153211	1	9
Звезда	0.055320	1	9
Кольцо	0.014307	1	10
Дерево	0.064059	1	9
Двухменая квадратная решётка	0.082925	4.844444	24
Двухмерная треугольная решётка	0.089373	50.733333	18
Трёхмерная квадратная решётся	0.069486	24.2	15
Пирамидальная топология	0.008404	503.522222	24
Полносвязная топология	0.013876	109601	45

Таблица 3.1. Результаты модельного эксперимента

Заключение

В дипломной работе выполнено следующее:

1. Изучена методика построения моеделей информационно-вычислительных сетей.

2. Разработанна программа, автоматизирующая вычисления стационарных и интегральных вероятностно-временных характеристик, плотностей распределения сообщений в маршрутах сети и среднего количества маршрутов между любыми двумя узлами сети на основе заданной модели сети. Она будет полезна при:

   - предварительной оценке характеристик проектируемой ИВС
   - оценке характеристик уже существующих ИВС
   - изучении влияния изменений топологии и/или оборудования на характеристики ИВС

3. Корректность работы программы и адекватность результатов проверена на моделях из [5].

4. Разработаны модели сетей с раными топологиями и с помощью разработанной программы проведён модельный эксперимент по их сравнению по времени и надёжности доставки сообщений.

Планирую дальнейшее развитие разработанной программы, улучшение функционала и производительности, поддержка различных методов аналитического моделирования для разных моделей сетей.

Список литературы

1. Вишневский В.М. Теоретические основы проектирования компьютерных сетей: монография. — Москва: Техносфера, 2003. — 512 с.

2. Клейнрок Л. Теория массового обслуживания. — Москва: Машиностроение, 1979. — 432 с.

3. Климанов В.П. Методология анализа вероятностно-временных характеристик локальных вычислительных сетей составных топологий на основе аналитического моделирования: автореферат диссертации на соискание учёной степени доктора технических наук. — Москва: Московский энергетический институт, 1993. — 40 с.

4. Климанов В.П. Методы разработки аналитических моделей для анализа локальных вычислительных сетей, используемых в управлении технологическим процессом: учебное пособие — Москва: Московский энергетический институт, 1995. — 115 с.

5. Климанов В.П., Руделёв Р.А. Моделирование информационных систем. Математические модели для разработки информационных систем: методика и решения: учебное пособие. — Москва: ФГБОУ ВПО МГТУ «СТАНКИН», 2014. — 45 с.

6. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов. — 4-е изд. — Санкт-Петербург: Питер, 2010. — 944 с.

7. Писарев В.Н. Применение теории массового обслуживания в задачах

инженерно-авиационного обеспечения. — Типография ВВИА имени проф. Н.Е. Жукова, 1965. — 45 с.

8. Хинчин А.Я. Работы по математической теории массового обслуживания — Москва: Физматгиз, 1963. — 236 с.

9. Файловый архив для студентов. — [Электронный ресурс]. Дата обновления: 11.04.2015. — URL: http://www.studfiles.ru/preview/2948129/ (дата обращения: 06.06.2015).