ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

На правах рукописи

Маличенко Дмитрий Александрович

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДОВ ХРАНЕНИЯ И ПЕРЕДАЧИ ИНФОРМАЦИИ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Специальность 05.12.13 —
«Системы, сети и устройства телекоммуникаций»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Крук Евгений Аврамович

Санкт-Петербург — 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования. Системы хранения данных очень востребованы в современном мире. С каждым годом объем обрабатываемой информации существенно возрастает. Этому способствует рост производительности и развитие электронных устройств для создания и обработки информации, а также развитие сетей связи. В последние годы большую популярность набирает направление, называемое “большие данные” (от англ. Big Data), которое связано с обработкой и хранением огромного количества данных.

Требования, предъявляемые к системам хранения, остаются неизменными:

- надежность – записанные данные не должны быть испорчены;
- доступность – разрешенным пользователям должен быть обеспечен бесперебойный доступ к системе;
- производительность – возможность работы с большим количеством клиентов и большими объемами данных (основными показателями производительности являются количество одновременных операций ввода/вывода и время их выполнения);
- масштабируемость – способность увеличивать объем памяти и количество поддерживаемых клиентов без ущерба для других характеристик системы.

С ростом объемов данных и количества клиентов выполнять эти требования становится сложнее, также возникают новые задачи. Ответом на эти трудности стало появление распределенных систем хранения, в которых данные распределены по сети узлов, выполняющих функции хранения. Такие системы могут использоваться в службах бронирования билетов, системах электронной коммерции (интернет-магазины, биржи), в банках, медицинских учреждениях и во многих других случаях. Распределенные системы хранения являются ключевым элементом сетей доставки “содержимого” (от англ. Content Delivery Network или CDN) и облачных сервисов.

Появление все большего количества устройств, которые могут хранить, обрабатывать и передавать информацию, создает потенциал для дальнейшего развития распределенных систем хранения. Активно развивается направление под названием “интернет вещей” (от англ. Internet of Things, сокр. IOT), подразуме-

вающее подключение большого количества новых устройств (“вещей”) к сети. Уже в настоящее время исследуется использование в качестве узлов системы относительно простых устройств, доступных на потребительском рынке, однако в большинстве случаев узлы распределенной системы хранения все же представляют собой сложный объект.

В решениях, востребованных на предприятиях, это интеллектуальная система, состоящая из памяти разного типа, серверов и операционной системы, в которой заложены алгоритмы обработки данных и управления. С одной стороны, в самих узлах распределенной системы возникают задачи, связанные с эффективным хранением данных, с другой стороны, возникают задачи, связанные с передачей информации между узлами и клиентами системы.

В данной работе рассматривается распределеная система хранения. В узлах системы находятся многоуровневые хранилища данных. Имеется набор приложений, работающих с системой, для которых задано требование к задержке.

Степень разработанности темы. Вопросам хранения и передачи информации в распределенных системах хранения данных посвящено большое количество работ. Много работ посвящено использованию корректирующих кодов для организации памяти и передачи информации по сети. Разработаны коды специально для распределенных систем хранения такими учеными как М. Луби, А. Шокролахи, А. Барг, П. Гопалан, В. Кадамбе. Давно известны фундаментальные исслодования Ч.К. Чоу, посвященные организации многоуровневых хранилищ данных.

Вопрос передачи информации рассматривается в работах таких российских ученых как Э.М. Габидулин, В.В. Зяблов, К.Ш. Зигангиров, Г.А. Кабатянский, Е.А. Крук, С.В. Семенов. Среди зарубежных исследователей – Н. Максемчук, А.Г. Димакис, М. Медард, М. Митценмахер, М. Герами.

Целью диссертационного исследования является разработка методов для повышения эффективности работы распределенной системы хранения данных.

В соответствии с целью были поставлены следующие **задачи**:

1. Разработка алгоритма распределения памяти между приложениями на каждом уровне памяти таким образом, чтобы удовлетворить требованиям на задержку для каждого приложения.

2. Исследование метода транспортного кодирования как средства уменьшения задержки в сети и разработка модификаций данного метода с целью повышения его эффективности.

Объект и предмет исследования. Объектом исследования является распределенная система хранения данных.

Предметом исследования является распределение памяти в многоуровневом хранилище данных и задержка при передаче информации по сети.

Методология и методы исследования. При получении основных результатов работы использовались методы теории вероятности и математической статистики, методы имитационного моделирования, теории кодирования и методы построения алгоритмов.

Научная новизна диссертационной работы заключается в следующем.

1. Разработана модель потока запросов к системе хранения, которая параметризуется распределением стековых расстояний и распределением частот появления адресов.
2. Разработан алгоритм распределения памяти между приложениями, удовлетворяющий требованиям на задержку приложения, который адаптируется к изменяемым характеристикам входного потока запросов.
3. Разработаны модели сети с коммутацией пакетов, учитывающие неэкспоненциальный характер задержки пакетов, изменения емкостей каналов сети и ограниченного времени жизни пакетов.
4. Исследована эффективность транспортного кодирования при различных распределениях задержки пакетов, а также при условиях, возникающих в реальных сетях связи, таких как изменяемые со временем емкости каналов и ограниченное время жизни пакетов. Представлены зависимости выигрыша транспортного кодирования от таких параметров как средняя загрузка сети, скорость кодирования, количество информационных пакетов, длина маршрутов.
5. Предложена модификация транспортного кодирования, которая позволяет повысить его эффективность в сетях с нерегулярной структурой. Проведен анализ выигрыша от кодирования.

Теоретическая и практическая значимость диссертационной работы определяется тем, что полученные в ней результаты позволяют повысить эффек-

тивность организации памяти в системах хранения данных и расширить область применения транспортного кодирования в реальных сетях связи.

Предложенный алгоритм распределения памяти между приложениями позволяет удовлетворить требованиям приложений к задержке системы, которая является важной характеристикой системы. Алгоритм повышает эффективность работы системы, т.к. выполняет расчет необходимого каждому приложению объема памяти, учитывая характеристики запросов к системе и заданные требования по задержке.

Проведенное исследование транспортного кодирования является продолжением и развитием предыдущих работ, посвященных данной теме. Уточнены условия выигрыша от кодирования при дополнительных эффектах, присутствующих в реальных сетях. Предложенная модификация транспортного кодирования повышает его эффективность в сетях с нерегулярной структурой.

Степень достоверности результатов работы обеспечивается корректным использованием математического аппарата, использованием имитационного моделирования для проверки корректности выполненных вычислений, а также апробацией полученных результатов на научно-технических конференциях. Основные результаты опубликованы в рецензируемых журналах, в том числе и в международных.

Апробация результатов. Основные результаты работы докладывались и обсуждались на следующих конференциях и симпозиумах: на 9-й Международной конфереции «FRUCT» (Петрозаводск, Россия, 2011); на Всероссийской научной конференции по проблемам информатики «СПИСОК» (Санкт-Петербург, Россия, 2012); на 14-м Международном симпозиуме «Problems of Redundancy in Information and Control Systems» (Санкт-Петербург, Россия, 2014); на 8-й Международной конференции «KES Conference on Intelligent Interactive Multimedia: Systems And Services» (Сорренто, Италия, 2015); на 15-м Международном симпозиуме «Problems of Redundancy in Information and Control Systems» (Санкт-Петербург, Россия, 2016)

Внедрение результатов. Результаты работы были использованы в научно-исследовательских работах Санкт-Петербургского государственного университета аэрокосмического приборостроения (ГУАП) и ЗАО «ИКТ». Теоретические результаты работы используются в учебном процессе кафедры безопасности информационных систем ГУАП.

Публикации. Основные результаты диссертационной работы опубликованы в 10 печатных работах [1–10]. Среди них 3 работы [1–3] опубликованы в изданиях, включенных в перечень ВАК, а также 2 работы [5], [6] опубликованы в изданиях, индексируемых Scopus.

Основные положения, выносимые на защиту.

1. Алгоритм перераспределения памяти в многоуровневой системе хранения данных, который позволяет удовлетворить требованиям на задержку системы и адаптироваться к входному потоку запросов.
2. Анализ эффективности транспортного кодирования с учетом особенностей реальных сетей, таких как изменяемые емкости каналов и неэкспоненциальный характер задержки пакетов, в результате которого получены зависимости эффективности кодирования от параметров сети.
3. Модифицированный алгоритм транспортного кодирования для нерегулярных сетей передачи информации, который позволяет уменьшить среднюю задержку сообщений.

Объем и структура работы. Диссертационная работа состоит из введения, трех разделов, заключения и списка литературы из 107 наименований. Полный объем диссертационной работы составляет 125 страниц с 68 рисунками, 4 таблицами и 1 приложением.

В первом разделе рассматривается распределеннная система хранения данных. Описывается архитектура и назначение элементов системы, а также используемые технологии. Приводятся задачи, решаемые системой, и предъявляемые к распределенному хранилищу требования.

Второй раздел посвящен задаче распределения памяти между приложениями на каждом уровне хранилища таким образом, чтобы удовлетворить требованиям приложений на задержку системы. Приводится структурное и функциональное описание многоуровневого хранилища данных. Описываются задачи, решаемые в работах, посвященных таким системам, и выполняется обзор существующих решений по задаче распределения памяти. Разработана и описана модель многоуровневой системы хранения. Построена математическая модель системы и сформулирована задача распределения памяти. Разработан адаптивный алгоритм распределения памяти. Разработана система моделирования для оценки эффективности алгоритма. Приводятся результаты имитационного моделирования, которые показывают эффективность предложенного алгоритма.

Третий раздел посвящен передаче данных по сети. Выполнен обзор существующих работ, посвященных механизмам взаимодействия узлов распределенной системы хранения. Рассмотрены основные задачи, возникающие при передаче данных. Описана известная модель сети с коммутацией пакетов. Дано определение транспортного кодирования и приведен существующий анализ задержки сообщений. Разработана новая модель сети, учитывающая ряд свойств, присутствующих в реальных сетях передачи данных. С помощью данной модели проведен анализ выигрыша от использования транспортного кодирования. Проведено исследование выигрыша транспортного кодирования в нерегулярных сетях. Предложена модификация транспортного кодирования, повышающая его эффективность в нерегулярных сетях.

В заключении кратко перечислены основные результаты, полученные в диссертационной работе, а также возможные направления дальнейшей разработки темы.

1 РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ

Крупные предприятия работают с большими объемами данных и имеют дело с гораздо большим количеством операций записи/чтения, чем обычный пользователь. Среди первых проблем, с которыми такие организации столкнулись, надежность данных и производительность системы хранения. При интенсивном использовании дисков они выходят из строя, что ведет к потере данных. Для решения этой проблемы появилась технология RAID (от англ. Redundant Array of Independent Disks). RAID-массивы объединяют множество дисков, причем один диск может дублировать другой. Таким образом, выход из строя одного из них не приведет к потере данных и остановке работы системы. RAID-массивы также позволяют увеличить скорость операций чтения/записи, работая одновременно с несколькими дисками. Изначально хранилища использовали прямое подключение к серверу или рабочей станции (DAS – Direct-attached storage) [11], однако,при таком подключении возникают трудности при организации совместного доступа и при росте хранилища, т.к. изначально хранилища находились внутри корпуса сервера. Новые требования к системам привели к появлению технологий NAS (от англ. Network Attached Storage) и SAN (от англ. Storage Area Network).

Сетевое устройство хранения данных NAS [11] представляет собой независимое от сервера или рабочей станции хранилище, подключаемое к существующей локальной сети и осуществляющее доступ к файлам с помощью сетевой файловой системы. Таким устройством могут пользоваться сразу несколько рабочих станций и его легче расширять новыми дисками, хотя оно тоже имеет ограничение на количество устанавливаемых дисков. Файловый доступ и передача по сети ограничивает скорость работы с такой системой. Преодолеть эти ограничения можно с помощью технологии SAN.

SAN – это сеть хранения данных. Она обладает наилучшей способностью к масштабированию. Можно подключать огромное количество устройств хранения, используя сеть и специальное сетевое оборудование. В сетях хранения данных используется оптоволоконный канал и доступ к хранилищам осуществляется на блочном уровне. Это дает гораздо более высокую скорость работы, чем NAS.

В существующих решениях новые технологии применяются совместно со старыми. Даже такие устаревшие хранилища как ленты до сих пор могут исполь-

зоваться для хранения архивов. Они хорошо подходят для хранения огромного количества архивной информации, т.к. обладают низкой стоимостью и такие данные запрашиваются крайне редко. Также технология RAID может использоваться совместно с другими, более новыми.

В данной работе рассматривается распределенная система хранения данных. Она состоит из узлов, которые представляют собой хранилища данных. Узлы соединены между собой вычислительной сетью. К сети имеют доступ приложения, которые записывают и читают данные из узлов. В качестве вычислительной сети может выступать как локальная сеть некоторой организации, так и сеть Интернет. Узлы могут находиться на разном удалении друг от друга.

1. Множество хранилищ находящихся на одной площадке в одном дата-центре.
2. Несколько объединенных посредством сети дата-центров в разных городах, регионах.
3. Несколько дата-центров, находящихся на значительном удалении друг от друга на разных континентах.

Функции, которые могут выполнять узлы.

1. Непосредственное хранение.
    (a) Хранение файлов.
    (b) Хранение частей (p2p, кодированное хранение).
2. Архивирование. Основной целью архивирования является создание резервных копий, которые могут быть использованы в случае потери информации.
3. Репликация. Основной целью репликации является создание полной копии узла, при выходе из строя которого, копия смогла бы быстро его заменить.
4. Кэширование. Основной целью кэширования является хранение наиболее востребованных данных как можно ближе к клиенту, чтобы сократить нагрузку и/или уменьшить задержку при обращении к системе хранения.

Требования, предъявляемые к системе хранения.

– Доступность системы. У клиента всегда должен быть доступ к данным. Если какой-то узел выходит из строя, то его должны заменить другие узлы. Процесс восстановления занимает некоторое время. Способность к

восстановлению данных достигается путем введения избыточности в систему. Задаче эффективного управления избыточностью посвящено много работ. Существует задача снижения количества данных, передаваемых по сети узлу, которому требуется восстановить данные.

- Целостность данных. Необходимо обеспечить сответствие прочитанных данных записанным. Другими словами, в процессе записи/чтения/хранения данные не должны измениться. Как правило, это обеспечивается применением корректирующих кодов.
- Производительность. Обеспечение высокой скорости работы системы. Может выражаться в количестве операций чтения/записи в единицу времени или во времени выполнения этих операций. Существует множество разных способов решения данной задачи. Например, развитие аппаратных средств хранения данных (быстрые диски), алгоритмы кэширования, использование избыточности и др.
- Безопасность. Задача состоит в исключении несанкционированного доступа к данным. Сюда входят задачи авторизации, шифрования и др.
- Масштабируемость. При нехватке ресурсов для хранения данных необходимо иметь возможность их нарастить. Важно сделать это за минимальное время и без прерывания работы всей системы. Также увеличение системы в объемах не должно уменьшать ее производительность.
- Управляемость. Чем больше и сложнее система хранения, тем сложнее ею управлять. Сложности также возникают при использовании решений от разных производителей. Сейчас активно развивается подход под названием программно определяемые хранилища (software defined storages или SDS в англоязычной литературе). Идея этого подхода заключается в том, чтобы использовать специальное порграммное обеспечение для организации и управления хранилищем независимо от аппаратных средств. Разделение программной и аппаратной части достигается за счет виртуализации.

Данные в распределенной системе хранения могут пересылаться как между клиентом и узлом, так и между самими узлами. Операции чтения и записи данных клиентом могут производиться с использованием сразу нескольких узлов. Работа клиентов с распределенной системой хранения может быть организована раз-

личным образом. Рассмотрим несколько возможных вариантов взаимодействия клиентов и узлов по сети передачи данных.

1. Запись данных: клиент –> узел.

   Простейший вид взаимодействия. Клиент записывает свои данные на один узел (рисунок 1.1). Запись может быть синхронной – операция завершается, когда данные записаны на диск, и асинхронной – операция завершается раньше фактической записи на диск.

Рисунок 1.1 — Запись данных клиента на один узел

2. Запись данных: клиент –> несколько узлов.

   Данные клиента распределяются среди нескольких узлов (рисунок 1.2). Существует несколько способов распределения данных. Например, путем создания полной копии данных на всех узлах или путем использования корректирующих кодов. В последнем случае избыточность данных меньше.

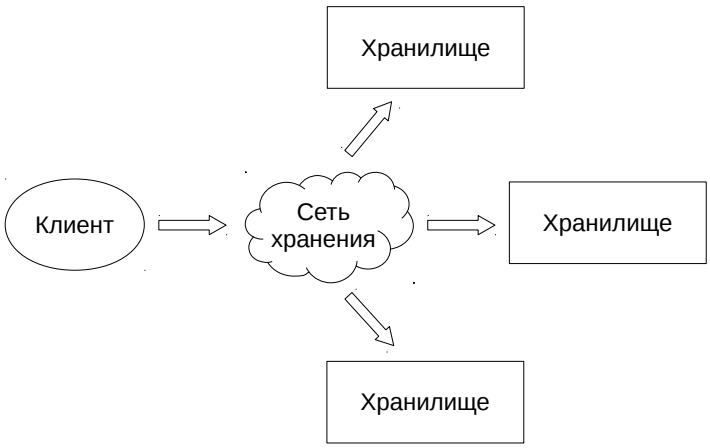


Рисунок 1.2 — Распределенная запись данных клиента на множество хранилищ

3. Запись данных: узел –> узел (репликация)

   Как правило при репликации создается некоторое множество узлов-копий (рисунок 1.3). Они могут использоваться, например, для баланси-

ровки нагрузки либо для восстановления утраченных данных. Если основной узел выйдет из строя, то один из узелов-копий сможет работать вместо него.

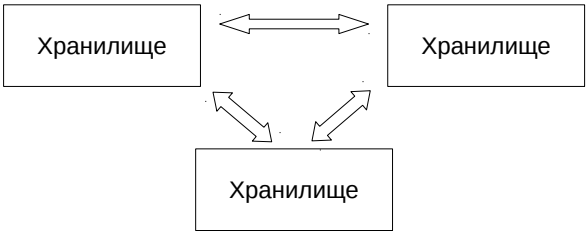


Рисунок 1.3 — Репликация данных клиента на множество хранилищ

4. Чтение: узел –> клиент
   Простейшая операция, аналогичная операции записи в п. 1
5. Чтение: несколько узлов –> клиент
   Такой вид взаимодействия происходит если информацию необходимо собрать с множества узлов (рисунок 1.4).

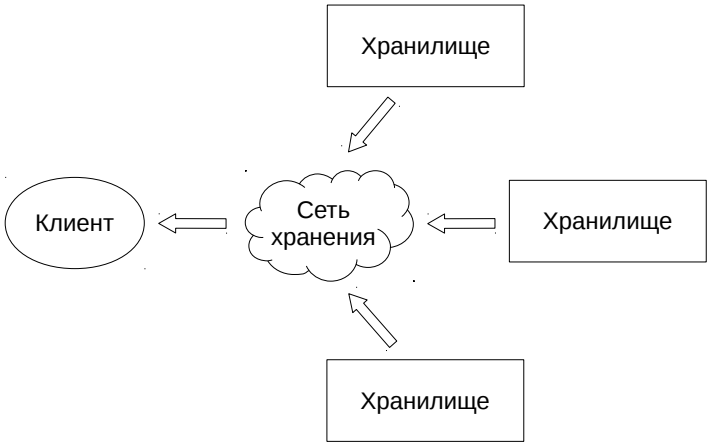


Рисунок 1.4 — Чтение данных из распределенной системы

Как было показно выше, некоторые узлы распределенной системы хранения могут использоваться для введения избыточности. В этом случае существует возможность получения одних и тех же данных из разных источников. В сети также может вводиться избыточность за счет дополнительных или неиспользуемых связей и за счет создания специальной топологии. Сетевая избыточность может быть использована для повышения производительности и надежности сети.

Одной из основных характеристик системы является ее производительность, выражаемая во времени доступа к данным. В описанной системе это время складывается из времени передачи по сети и времени доступа к информации на узле, следовательно, важными задачами являются организация соответствующим образом передачи данных по сети и обеспечение быстродействия на узле. Очевидно, что в большой системе запросы имеют разные требования по задержке. Поэтому в одной и той же системе для одних клиентов может быть задано более строгое требование к задержке, а для других менее строгое. Для эффективного использования ресурсов системы необходимо это учитывать. Данная работа посвящена уменьшению задержки в распределенной системе хранения. Раздел 2 посвящен анализу и уменьшению задержки на узле. Решается задача обеспечения узлом требуемой задержки для всех приложений. Раздел 3 посвящен задержке при передаче по сети. Исследуется метод транспортного кодирования как средство уменьшения задержки. Предлагается модификация транспортного кодирования, повышающая эффективность передачи в неравномерных сетях.

2 РАСПРЕДЕЛЕНИЕ ПАМЯТИ СРЕДИ ПРИЛОЖЕНИЙ В МНОГОУРОВНЕВЫХ ХРАНИЛИЩАХ ДАННЫХ

В данном разделе рассматривается не вся распределенная система хранения целиком, а ее узел – многоуровневая система хранения данных. Пусть имеется многоуровневое хранилище данных с заданным количеством уровней и заданными размерами этих уровней. Есть набор приложений, работающих с системой, для которых задано требование на среднюю задержку операций ввода/вывода. В разделе решается задача распределения памяти между приложениями на уровнях хранилища таким образом, чтобы выполнялись требования на задержку.

Для решения поставленной задачи разработана модель системы хранения на основе известных моделей, в которой представлен новый модуль, реализующий алгоритм расчета объемов памяти, необходимых для выполнения требования на задержку. Разработанный алгоритм является адаптивным. Он анализирует характеристики потока заявок и учитывает их при управлении системой. Рассматриваемая система хранения выполняет изолирование адресного пространства разных приложений друг от друга, путем деления его на секции. Входными данными алгоритма являются такие характеристики потока заявок как распределение стековых расстояний и распределение популярностей запрашиваемых блоков данных. Выходными данными являются размеры секций для каждого приложения на каждом уровне памяти кроме кэш-памяти и цель на хит-рейт для алгоритма управления кэш-памятью. Для оценки работы предложенного алгоритма использовались два типа потоков запросов. Во-первых, это потоки, собранные с реальных систем хранения и доступные в сети интернет. Во-вторых, это синтетические потоки. Для генерации синтетических потоков заявок к системе разработан алгоритм, позволяющий учесть желаемые распределения стековых расстояний и популярности запрашиваемых блоков. Также для оценки работы алгоритма распределения памяти разработан комплексный критерий, который демонстрирует, насколько близки задержки системы к заданным.

Раздел построен следующим образом. В подразделе 2.1 приведены общие знания о многоуровневой системе хранения. В подразделе 2.2 описаны основные задачи и существующие решения, посвященные задержке в многоуровневых системах хранения. В подразделе 2.3 анализируются специфические характеристи-

ки, свойственные потокам запросов к системам хранения данных. В подразделе 2.4 описана разработанная модель системы, которая собрана преимущественно из уже известных блоков, но имеет новый блок, в котором реализован разработанный алгоритм. В подразделе 2.7 приводится грубая оценка сложности алгоритма. В подразделе 2.8 описаны модели потока заявок, используемые для оценки предлагаемого алгоритма по критерию, описанному в подразделе 2.9. Результаты моделирования описанной системы хранения с предлагаемым алгоритмом приведены в подразделе 2.10. В подразделе 2.11 приведены выводы по разделу.

2.1 Многоуровневая система хранения

Многоуровневые системы хранения являются узлами распределенной системы хранения данных и, как правило, используются в решениях промышленного уровня. Они представляют собой комплексные и интелектуальные системы, в состав которых входит резервная система питания, резервная память для повышения надежности, отдельный сервер, назначение которого – координация работы всех систем хранилища. Уровни системы представляют собой хранилища данных, отличающиеся друг от друга скоростью работы, ценой и емкостью. Системы такого типа хорошо описаны в [11].

Распространенная структура уровней представляет собой кэш память и диски с разной скоростью доступа. Емкость более быстрых дисков обычно гораздо меньше, чем емкость медленных, из-за высокой стоимости. Размер кэша значительно меньше объема данных, хранимых на дисках. Кэш память самая быстрая и самая дорогая. Цель подобных систем состоит в том, чтобы получить большую скорость работы с данными по сравнению с обычными SATA дисками и при этом иметь решение более дешевое, чем использование исключительно более быстрых дисков. Чтобы этого добиться небольшой объем самых быстрых дисков используют только для наиболее востребованных данных. Таким образом, те данные, которые запрашиваются чаще всего, считываются быстро. Данные, которые запрашиваются редко, находятся на более медленных дисках. Количество уровней дисковой памяти не ограничено, но в реальных системах часто встречаются два или три уровня.

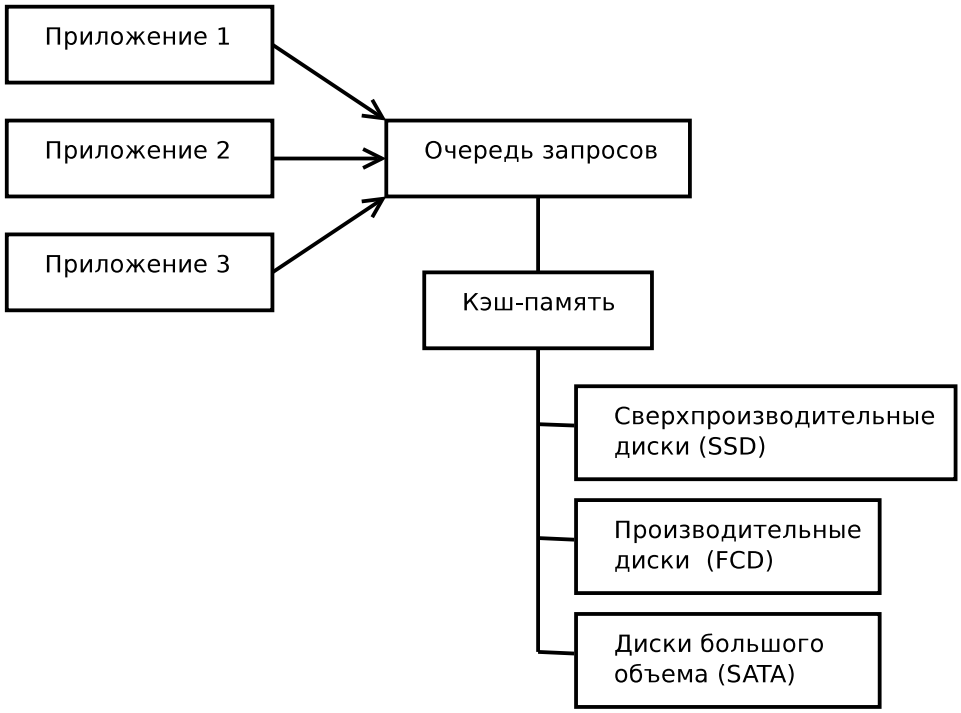


Рисунок 2.1 — Многоуровневая система хранения

В качестве иллюстрации на рисунке 2.1 представлена обобщенная схема многоуровневой системы. На ней изображены следующие компоненты.

- Очередь запросов, которая содержит запросы от разных клиентов на запись или чтение.
- Кэш-память – самая быстрая память в системе. Она предназначена для временного хранения наиболее востребованных данных.
- Сверхпроизводительные диски. Как правило, это диски на основе flash памяти. Они предназначены для хранения наиболее востребованных данных.
- Производительные диски. Как правило, это высокоскоростные диски с доступом по оптическому каналу. Они имеют на порядок меньшую скорость, чем flash память, но обладают существенно большим объемом.
- Диски большого объема. Они на порядок медленнее предыдущего типа дисков, но обладают большой емкостью и более низкой стоимостью. Они хорошо подходят для хранения данных, которые редко запрашиваются.

Похожие схемы можно видеть как в исследовательских работах [12], [13], [14], [15], так и в промышленных реализациях [16].

Рассмотрим работу такой системы на примере записи и чтения. Сначала любая заявка попадает в очередь, где она занимает свое место в соответствии с управляющим алгоритмом. Если поступила заявка на запись блока, то этот блок

сначала записывается в кэш-память, а потом происходит запись блока на один из дисков системы. Определением уровня, на который произойдет запись, занимается операционная система. Если поступила заявка на чтение блока, то система сначала ищет блок в кэш памяти и, если находит, то считывает блок оттуда. Если блок не найден, то система определяет, на каком диске он лежит и считывает его.

Время считывания блока данных зависит от того, на диске какого типа он расположен. Считанный блок отправляется инициатору запроса и записывается в кэш память. При скором повторном обращении к этому же блоку данных он будет считан уже из кэш памяти.

На рисунке 2.1 представлена структура, общая для многих систем хранения, однако функционирование каждого модуля зависит от сопутствующих алгоритмов управления. Существует множество различных решений, однако, все они строятся таким образом, чтобы наиболее актуальные данные оказывались на быстрых дисках или в кэш-памяти, а менее актуальные данные на медленных дисках.

Следующий подраздел посвящен описанию существующих решений, которые связаны с блоками, представленными на рисунке 2.1.

2.2 Обзор существующих решений

Все исследования, посвященные многоуровневым системам хранения, преследуют одну цель – повышение производительности и эффективности работы. Следующие характеристики являются главными при рассмотрении таких систем:

- задержка – время, необходимое для выполнения операции чтения/записи;
- пропускная способность системы – объем данных, который система может обработать за единицу времени;
- количество операций ввода/вывода в секунду, зависящее от предыдущих двух характеристик.

Чем меньше задержка и чем больше пропускная способность и количество операций чтения/записи в секунду, тем выше производительность системы. Чем выше производительность системы при одних и тех же затратах (в стоимости или объемах памяти на уровнях), тем выше эффективность работы системы.

Для достижения цели исследователи разрабатывают алгоритмы управления, основной список которых представлен на рисунке 2.2.

Рисунок 2.2 — Алгоритмы управления системой хранения

Алгоритм управления очередью позволяет уменьшить задержку системы путем учета свойств потоков запросов и переупорядочения запросов таким образом, чтобы соседние запросы относились к близким областям памяти [17] [18].

Управление кэш-памятью позволяет увеличить вероятность того, что очередной запрос окажется в кэше. Таким образом, задержка системы уменьшится. Работы [19], [20], [21], [22], [23], [24], [25] посвящены этой задаче.

Алгоритмы виртуализации отделяют логическое представление от конкретных устройств хранения. Виртуализация помогает повысить масштабируемость системы, простоту использования, а также производительность за счет грамотного распределения данных по хранилищам с разными характеристиками. Данные одного логического диска могут находиться на дисках, обладающих разной производительностью. Виртуализации посвящены работы [26], [27], [28].

Одним из способов виртуализации является подход, который называется программно-определяемые хранилища [28], в которых программное обеспечение и аппаратные средства представляют собой отдельные друг от друга слои, что позволяет изменять аппаратную часть, не меняя программной. В работе [29] рассматривается проблема большого числа параллельных запросов, предлагается параллельная файловая система. Влиянию программной части на производи-

тельность также посвящена работа [30], в ней рассматривается задача передачи данных на большие расстояния.

Алгоритм управления уровнями выполняет перераспределение дисковой памяти между уровнями и перемещение данных между ними, что позволяет поддерживать состояние, в котором наиболее востребованные данные находятся на быстрых дисках, а наименее востребованные данные – на медленных. Это позволяет увеличить количество запросов, которые будут иметь небольшую задержку. Такие алгоритмы рассматриваются в работах [31], [17], [16], [15]. Перемещение данных между уровнями рассмотрено в работах [12], [13], [14]. В работе [14] рассматривается всего 2 уровня памяти, один уровень состоит из обычных дисков, а другой из SSD дисков и используется в качестве кэша.

Производительность и надежность системы хранения можно улучшить благодаря улучшению соответствующих характеристик на дисках. Для этих целей создаются различные алгоритмы кэширования и управления дисками [26], [32]. Для дисков на основе flash памяти алгоритмы кэширования также позволяют увеличить надежность диска за счет уменьшения количества беспорядочных операций записи [33], [34].

В работах [35], [36] рассматривается задача обеспечения клиентам системы заданного качества обслуживания. В работе [37] построена аналитическая модель системы, по которой выполняется расчет необходимого количества уровней памяти, а также их размеры. В работе [15] выполняется расчет необходимого объема памяти на каждом уровне на основе собранной предварительно статистики.

Для оценки работы новых алгоритмов в работах, как правило, используется система моделирования собственной разработки [27], [17] из-за различий в дизайне самой системы хранения и необходимости реализовать новый модуль. В качестве метрик используются задержка операций ввода/вывода и количество операций в еденицу времени.

Тематика распределенных систем хранения популярна и среди диссертационных работ. Так, например, работа [38] посвящена уменьшению задержки в облачных системах.

2.3 Специфика запросов к системам хранения данных

Поток запросов к системам хранения данных имеет такие важные свойства как временная и пространственная локальности [39], [40], [41], [42]. Эти свойства учитываются при проектировании любых хранилищ данных, что позволяет эффективнее использовать уровни системы хранения [22], [23], [43]. В работе [44] предложена методика сбора характеристик потока с работающей системы для приложений в области вычислительных наук. Важно учитывать свойства потока запросов также при его моделировании, в работе [45] представлен ряд относительно простых моделей трафика. В последующих двух подразделах даются определения для временной и пространственной локальности. Для более подробного изучения свойств потока запросов можно обратиться к [45].

2.3.1 Временная локальность

Временная локальность – свойство потока запросов, состоящее в том, что если в данный момент запрошен некоторый адрес X, то в скором времени с большой вероятностью этот же адрес X будет запрошен еще раз. Количественно оценить временную локальность можно с помощью распределения стековых расстояний. Стековое расстояние адреса X – это количество уникальных адресов, которое запрашивается между двумя соседними запросами к адресу X, включая сам адрес X. Например, для последовательности адресов X,A,B,B,C,A,X стековое расстояние адреса X равно четырем, а стековое расстояние адреса B равно единице. Важной характеристикой является распределение вероятностей стековых расстояний $F_{\text{sd}}(x)$ – вероятность того, что стековое расстояние очередного запрошенного адреса не будет превышать x. Пример распределения стековых расстояний представлен на рисунке 2.3. Если кэш организовать в виде стека, то распределение стековых расстояний показывает вероятность того, что запрашиваемый блок данных окажется в кэше. Распределение, представленное на рисунке 2.3, говорит о том, что необходимо иметь кэш размером 20% от количества всех данных, чтобы вероятность использования кэша была равна 0.7. Вероятность попадания в

кэш называют хит-рейтом. Распределение стековых расстояний показывает, насколько может быть эффективен кэш, организованный по принципу стека, для данного потока запросов. Чем выше скорость роста кривой распределения стековых расстояний, тем выше эффективность использования кэш-памяти.

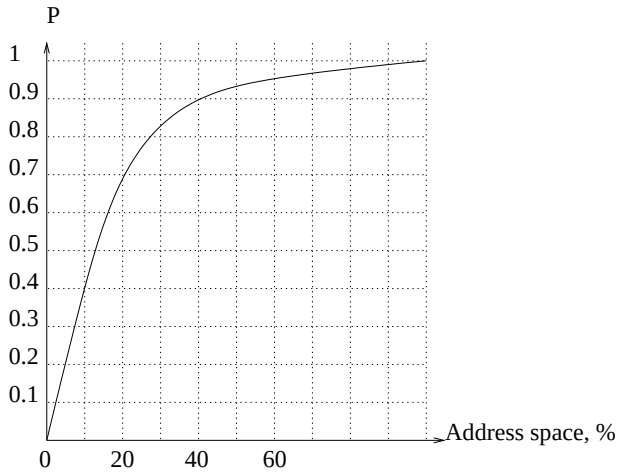


Рисунок 2.3 — Пример распределения стековых расстояний

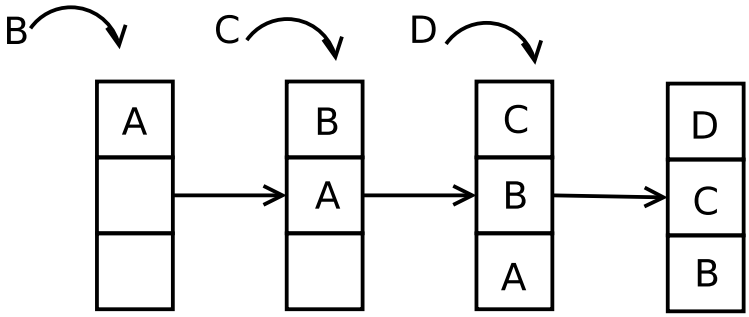


Рисунок 2.4 — Пример работы LRU кэша

Рассмотрим моделирование потока адресов с заданной гистограммой стековых расстояний. Чтобы сгенерировать поток запросов можно использовать LRU кэш (от англ. Least Recently Used – вытесненение давно неиспользуемых). Если кэш заполнен и приходит новый адрес, то он записывается вместо адреса, который запрашивался раньше всех остальных. Мы будем моделировать LRU кэш в виде стека. Тогда адреса, которые только что запрашивались, будут на вершине стека, а давно неиспользовавшиеся адреса будут на дне стека. Вытесняться из стека будет всегда самый нижний адрес, так как он самый старый в стеке. В качестве примера расммотрим кэш размером 3 ячейки. Пусть запрошены по очереди адреса A, B и C (рисунок 2.4). Пусть затем запрошен адрес D, тогда адрес A будет удален из кэша, а адрес D будет помещен в начало кэша как только что запрошенный адрес. Пусть задана гистограмма стековых расстояний, тогда алгоритм генерации адресного потока с помощью LRU кэша выглядит следующим образом.

1. Кэш заполняется всеми уникальными адресами из адресного пространства.
2. С помощью заданного распределения генерируется стековое расстояние r.
3. Адрес, находящийся в кэше на расстоянии r, записывается в выходной поток и перемещается в кэше со своего места на самое верхнее/первое место в кэше.
4. Если требуется сгенерировать еще адреса, то переходим на шаг 2.

Чем дольше производится генерация адресов, тем больше распределение частот адресов соответствует равномерному. Таким образом, этот метод генерации адресов подходит только для моделирования временной локальности, так как популярности адресов в конечном итоге будут одинаковы.

2.3.2 Пространственная локальность

Пространственная локальность – свойство потока адресов, состоящее в том, что если в данный момент запрошен некоторый адрес X, то в скором времени с большой долей вероятности будет запрошен адрес Y в окрестности адреса X, т.е. $X - d \leq Y \leq X + d$, где d небольшое число относительно размеров адресного пространства. Для потока запросов к блокам данных характерно наличие областей локализации наиболее часто запрашиваемых адресов. Важно знать не только, какие адреса наиболее востребованы, но и сколько их. Для оценки количества наиболее часто запрашиваемых адресов можно построить распределение популярности адресов $F_{\text{pop}}(x)$. Значение $F_{\text{pop}}(x)$ для конкретного x будет равно вероятности запроса блоков из диапазона адресов $0 - x$. Для того, чтобы сгруппировать наиболее востребованные адреса в начале области определения функции $F_{\text{pop}}(x)$, отсортируем все адреса в порядке убывания их популярности. Тогда x – это не адрес запрашиваемого блока, а индекс, соответствующий расположению адреса в отсортированной по популярности последовательности. В таком случае функция $F_{\text{pop}}(x)$ имеет наибольший рост в самом начале, а далее скорость роста функции только уменьшается. В дальнейшем будем иметь в виду только такие функции популярности. На рисунке 2.5 представлен пример распределения попу-

лярностей адресов. Кривая показывает, какую долю данных нужно положить на более быстрые диски, а какую можно оставить на медленных, чтобы увеличить производительность системы хранения при ограниченном количестве быстрых дисков. Для приведенного примера быстрой памяти объемом всего 0.1 от общего объема данных достаточно для того, чтобы более 40% обращений к данным происходило в быстрой памяти.

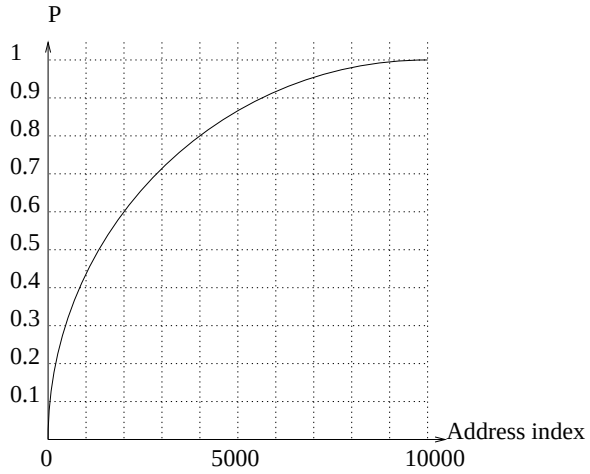


Рисунок 2.5 — Пример распределения популярности адресов

Рассмотрим генерацию потока адресов, обладающего пространственной локальностью, которую будем задавать с помощью распределения популярностей адресов, описанного выше. Если имеется история запрошенных адресов с реальной системы, то по ней можно посчитать оценки вероятностей появления адресов в потоке и использовать их при моделировании. Для этого необходимо пройти по файлу с запрошенными адресами и посчитать, сколько раз каждый адрес встретился, а после этого каждый счетчик поделить на количество запросов. Алгоритм генерации потока адресов будет выглядеть следующим образом.

1. С помощью плотности распределения популярности адресов сгенерировать индекс и соответствующий ему адрес записать в выходной поток.
2. Повторять пункт 1 до достижения заданного числа запросов, которое должно быть достаточным, чтобы сгенерированная последовательность была статистически близка к заданному распределению популярностей.

Полученный поток адресов будет иметь заданное распределение популярностей адресов, однако распределение стековых расстояний не будет контролироваться. Таким образом, данный метод позволяет учитывать только пространственную локальность.

2.4 Описание разработанной модели системы хранения

В данном разделе описывается выбранная модель системы хранения. Ее выбор сделан, исходя из имеющихся знаний о многоуровневых системах хранения, которые применяются в существующих решениях (например, [16]).

– Система содержит кэш память довольно большого объема.
– В некоторых системах применяется сегментирование кэш памяти, для изолирования работы приложений.
– Имеются диски разного типа для повышения эффективности работы.
– Для клиентов задается качество обслуживания (например, в виде длительности операций чтения/записи)

Состав модели представлен на рисунке 2.6.

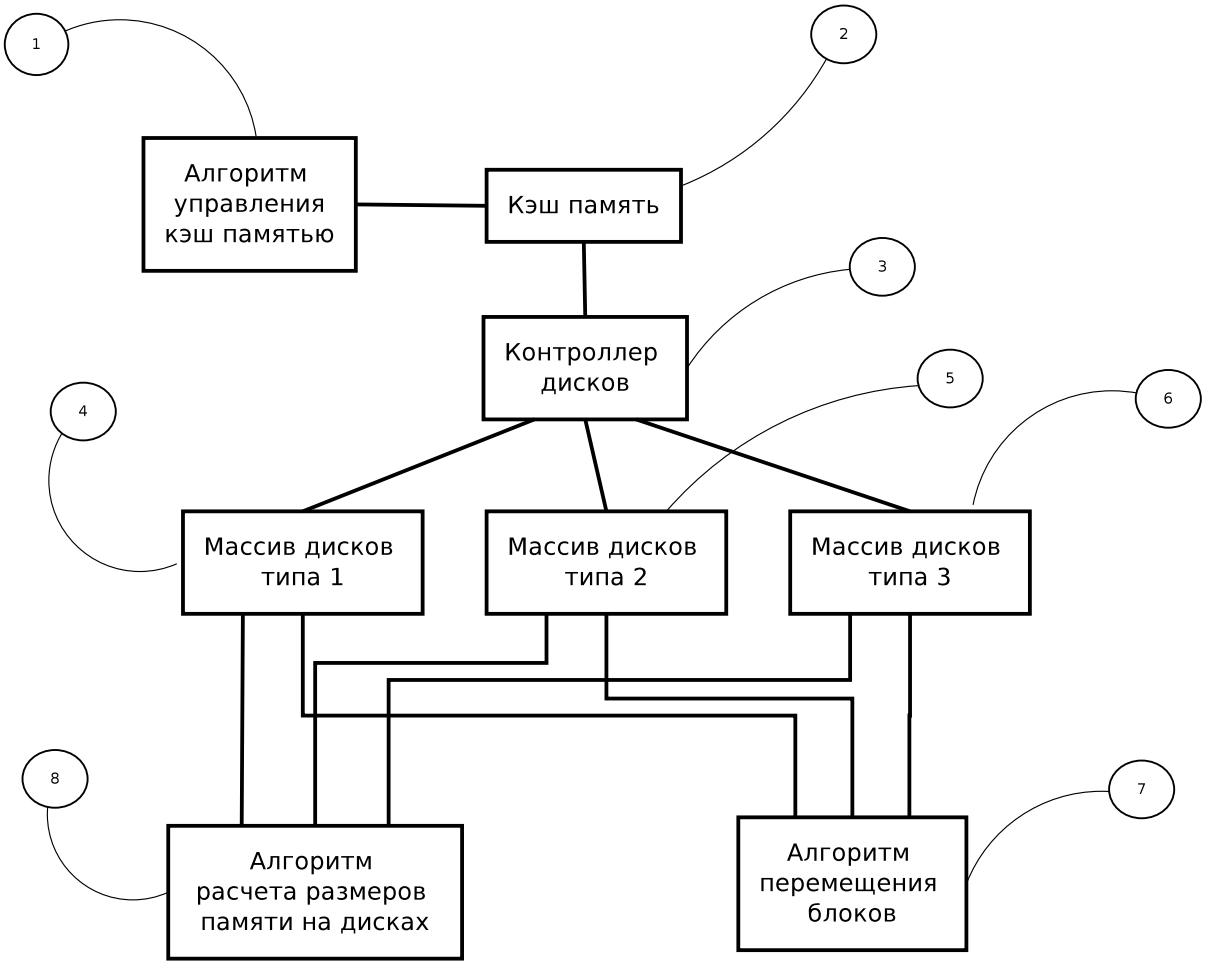


Рисунок 2.6 — Состав модели системы хранения.

Прежде чем перейти к описанию каждого блока, рассмотрим систему в целом. Для каждого клиентского приложения в системе будет выделена своя об-

ласть в кэш памяти и логический диск заданного объема. Логический диск представляет собой непрерывную область для записи и чтения данных приложением. Физически эти данные могут находиться на дисках разного типа. Диски определенного типа представляют собой один уровень системы. В рассматриваемой модели имеется 3 уровня дисковой памяти. Каждый уровень отличается скоростью работы с данными. Фактическое расположение данных скрыто от приложения и регулируется исключительно системой хранения.

Блок номер 1 – это алгоритм управления кэш памятью. Будем использовать алгоритм, который описан в работе [21]. Он делит всю область памяти на изолированные друг от друга сегменты, где каждый сегмент используется конкретным приложением или группой приложений, объединенных по некоторому принципу. На каждый сегмент кэша задается значение хит-рейта, которое должно достигаться во время работы кэша. Для этой цели в процессе работы алгоритма анализируется поток запросов и на основе анализа периодически пересчитываются размеры сегментов.

Блок номер 2 – это кэш память, которая работает по принципу “вытеснения давно неиспользуемых” блоков. В литературе такой кэш обычно называют LRU кэш (от Least Recently Used).

Блок номер 3 представляет собой контроллер дисков, который при запросе на запись определяет конкретный диск, куда будет записан блок данных. При запросе на чтение он определяет местоположение запрашиваемого блока.

Блок номер 4 представляет собой массив самых быстрых дисков в системе. Они предназначены для хранения наиболее востребованной информации. Примером в реальной системе могут служить SSD (Solid-state Drive) диски.

Блок номер 5 представляет собой массив более медленных и более дешевых дисков. Примером в реальной системе могут служить высокоскоростные диски с оптоволоконным каналом подключения.

Блок номер 6 представляет собой массив самых медленных дисков в системе. Их основное назначение заключается в длительном хранении данных. Примером в реальной системе могут служить SATA диски.

Блок номер 7 представляет собой алгоритм перемещения блоков данных между уровнями памяти (дисками разных типов). Для перемещения блоков выбран простой алгоритм, который сортирует все блоки данных по частоте использования и записывает их по очереди в рамках доступного места, начиная с самых

быстрых дисков. Таким образом, наиболее часто используемые блоки будут находиться на более быстрых уровнях системы хранения.

Блок номер 8 вычисляет размеры сегментов памяти для каждого приложения, работающего с системой, на каждом уровне памяти. Суммарный объем доступной памяти для каждого приложения остается неизменным, а распределение по уровням меняется.

Новизна в описанной модели заключается в алгоритме, заложенном в блоке номер 8. Его отличие от других подобных работ заключается в совокупности следующих функций:

- пересчет размеров памяти в процессе работы системы;
- учет переменчивости характера потока запросов в течение суток;
- удовлетворение требований на задержку операций ввода/вывода;
- учет эффективности работы кэш-памяти;
- выделение клиентам изолированных друг от друга областей памяти на дисках.

В подразделе 2.5 задача формулируется в математических терминах, а алгоритм описывается в подразделе 2.6.

2.5 Постановка задачи распределения памяти

Описание многоуровневой системы хранения с помощью математического аппарата была предпринята ранее в работе [37]. Однако описанная там модель была построена для решения других задач:

- определение оптимального количества уровней системы,
- определение сумарных размеров памяти на каждом уровне при заданных временах доступа к системе,
- определение времен доступа к системе при заданных размерах памяти на каждом уровне.

Отличие данной работы состоит в том, что уровни поделены на сегменты, которые выделяются приложениям и задачей является расчет размеров этих сегментов. Также отличие заключается в функции, определяющей вероятность попадания запроса на конкретный уровень системы. В работе [37] она считается задан-

ной. В данной работе она задается таблицей, полученной путем анализа входного потока запросов.

Постановка задачи, представленная в данном разделе, была описана автором в работе [1].

Пусть время доступа к блоку данных в кэше равно t_0, а времена доступа к дискам t_1, t_2, t_3 и $t_0 \ll t_1 \ll t_2 \ll t_3$, где 1,2,3 – номера уровней памяти. Пусть объемы памяти в кэше и на дисках соответственно равны S_0, S_1, S_2, S_3. С системой работает n приложений $A_1, \ldots, A_n$. Пусть объем памяти, выделенный для i-го приложения на j-м уровне памяти, равен s_j^i. Как было описано раннее, уровни отличаются быстродействием памяти. Тогда сумарный объем выделенной памяти не должен превышать весь имеющийся объем на каждом уровне:

$$\sum_{i=1}^{n} s_j^i \leq S_j. \tag{2.1}$$

Пусть p_j^i – вероятность того, что блок данных, запрошенный i-м приложением, окажется на j-м уровне памяти. Тогда среднее время доступа к блоку данных для i-го приложения равно

$$\overline{T}_i = p_0^i t_0 + p_1^i t_1 + p_2^i t_2 + p_3^i t_3, \tag{2.2}$$

где $p_0^i + p_1^i + p_2^i + p_3^i = 1$.

Пусть заданы ограничения на средние времена доступа к блоку данных для каждого приложения:

$$\overline{T}_i \leq R_i. \tag{2.3}$$

Следовательно, присутствует ограничение на вероятности p_j^i:

$$p_0^i t_0 + p_1^i t_1 + p_2^i t_2 + p_3^i t_3 \leq R_i. \tag{2.4}$$

Вероятности p_j^i связаны с объемами памяти s_j^i. Чем больше памяти приложение использует на j-м уровне относительно других уровней памяти, тем выше вероятность p_j^i. Обозначим эту зависимость:

$$\begin{aligned} p_0^i &= f_c^i\left(s_0^i\right), \\ p_j^i &= f_d^i\left(s_j^i, p_0^i\right), 1 \leq j \leq 3, \end{aligned}$$

где $f_c^i()$ и $f_d^i()$ – функции зависимости между размером занимаемой памяти и вероятностью попадания в нее для кэш-памяти и дисков соответственно.

Вероятность p_0^i является хит-рейтом, так как на 0-м уровне находится кэш память. В этом случае $f_c^i\left(s_0^i\right) = F_{\text{sd}}^i\left(s_0^i\right)$. Тогда

$$p_0^i = F_{\text{sd}}^i\left(s_0^i\right).$$

Так как на уровнях 1-3 находятся диски, то в этом случае для вычисления функции $f_d^i\left(s_j^i, p_0^i\right)$ будет использоваться функция популярности адресов $F_{\text{pop}}^i\left(x\right)$. Если имеется три диска разных типов с размерами S_1, S_2, S_3, и данные упорядочены по степени востребованности так, что самые востребованные данные хранятся на диске 1, а самые невостребованные данные на диске 3, то вероятность того, что данные будут запрошены с диска 1, равна

$$p_1^i = \frac{F_{\text{pop}}^i\left(s_1\right)}{1 - p_0^i} = f_d^1, \tag{2.5}$$

вероятность запроса данных с диска 2 равна

$$p_2^i = \frac{F_{\text{pop}}^i\left(s_2 + s_1\right) - F_{\text{pop}}^i\left(s_1\right)}{1 - p_0^i} = f_d^2, \tag{2.6}$$

а вероятность запроса данных с диска 3 равна

$$p_3^i = \frac{F_{\text{pop}}^i\left(s_3 + s_2 + s_1\right) - F_{\text{pop}}^i\left(s_2 + s_1\right) + F_{\text{pop}}^i\left(s_1\right)}{1 - p_0^i} = f_d^3, \tag{2.7}$$

Следует отметить, что функция $F_{\text{pop}}^i\left(x\right)$ должна соответствовать потоку запросов, уже прошедшему через кэш память. Функции популярностей потока запросов до кэш памяти и прошедшего через нее (адресов, не попавших в кэш) будут отличаться, так как в последнем случае популярность некоторых адресов будет ниже из-за того, что они попадали в кэш память. Для алгоритма распределения памяти на дисках важна именно популярность адресов, не попавших в кэш.

Таким образом, функции f_d^i для дисков разных уровней будут отличаться. В рассматриваемой системе хранения приложению выделена только часть s_j^i каж-

дого диска, т.е. $s_j^i < S_j$. Перепишем уравнения (2.5)-(2.7) другим образом:

$$s_1^i = {F_{\text{pop}}^i}^{-1}\left(p_1^i\left(1-p_0^i\right)\right), \tag{2.8}$$

$$s_2^i + s_1^i = {F_{\text{pop}}^i}^{-1}\left(\left(p_2^i + p_1^i\right)\left(1-p_0^i\right)\right), \tag{2.9}$$

$$s_3^i + s_2^i + s_1^i = {F_{\text{pop}}^i}^{-1}\left(\left(p_3^i + p_2^i + p_1^i\right)\left(1-p_0^i\right)\right). \tag{2.10}$$

При этом на вероятности p_j^i имеется ограничение (2.4), а на s_j^i ограничение (2.1).

Сформулируем задачу распределения места на дисках, как задачу вычисления размеров сегментов памяти s_i^j для всех i,j.

Пусть известны параметры S_0, S_1, S_2, S_3, t_0, t_1, t_2, t_3, N, $R_{i=1\ldots N}$. Требуется найти такие s_j^i, чтобы выполнялись заданные ограничения $R_{i=1\ldots N}$.

Кроме заданных параметров в выражениях (2.8) - (2.10) присутствуют функции $F_{\text{sd}}^i(x)$ и $F_{\text{pop}}^i(x)$, которые меняются в зависимости от потока запросов и для реальной системы их можно получить только численно, они задаются таблицей. Это приводит к тому, что поиск оптимального решения, во-первых, связан с переборными алгоритмами, а во-вторых, оптимальное решение возможно получить только, если функции $F_{\text{sd}}^i(x)$ и $F_{\text{pop}}^i(x)$ известны заранее. Строго говоря, на эти функции кроме входного потока влияют и сами алгоритмы управления кэшем и диском.

2.6 Описание разработанного алгоритма

Алгоритм, описываемый в данном разделе был представлен в работе [1].

Учитывая сказанное выше в подразделе 2.5, в данной работе предлагается адаптивный алгоритм решения задачи, который собирает статистику для определения функций $F_{\text{sd}}^i(x)$ и $F_{\text{pop}}^i(x)$ и на их основе выполняет расчет целевых значений s_j^i. Изменение фактических размеров сегментов памяти происходит путем постепенного приближения к рассчитанным s_j^i.

Для применимости адаптивного алгоритма необходимо, чтобы функции $F_{\text{sd}}^i(x)$ и $F_{\text{pop}}^i(x)$, используемые для вычисления границ s_i, были неизменными некоторое время, достаточное для того, чтобы система успела применить изменения и новые границы сегментов дали желаемый эффект, а именно, удовлетво-

рение требованиям на задержку. В противном случае, если функции $F^i_{\text{sd}}(x)$ и $F^i_{\text{pop}}(x)$, которые характеризуют поток запросов к системе, будут быстро меняться, то алгоритм не успеет адаптироваться.

Как было отмечено ранее, для реальной системы функции $F^i_{\text{sd}}(x)$ и $F^i_{\text{pop}}(x)$ будут задаваться таблицей и алгоритм поиска решения связан с полным перебором. В таком случае сложность предполагаемого алгоритма должна расти экспоненциально быстро с ростом количества приложений, следовательно, такое решение несовместимо с постоянно растущими системами хранения.

Разрабатываемый алгоритм также является переборным, однако, для уменьшения его сложности ниже предлагается ряд мер, позволяющий существенно сократить перебор.

Во-первых, разделим алгоритмы перераспределения места в кэш памяти и на дисках. Независимое вычисление границ в кэш памяти и на дисках позволяет сократить поиск решений. Считая, что характеристики потока запросов имеют стабильные периоды, средние хит-рейты кэша p^i_0 также должны сохраняться. Переменные p^i_0 будут исключены из перебора и будут использоваться фактические значения в алгоритме расчета s^i_j на дисках.

Во-вторых, введем обратную связь между разрабатываемым алгоритмом и алгоритмом управления кэш памятью путем введения цели на хит-рейт для алгоритма управлением кэшем. Таким образом, если алгоритму распределения памяти на дисках не будет хватать ресурсов, он подаст заявку на повышение хит-рейтов отдельным приложениям.

В-третьих, воспользуемся тем, что функция $F^i_{\text{pop}}(x)$ имеет дискретность. Размер дискрета зависит от размера блока памяти, для которого вычисляется статистика в системе. В существующих системах он относительно большой. Как правило, он измеряется в сотнях мегабайт, а объем данных приложений в терабайтах. Отличие составляет примерно два порядка. Следовательно, такого порядка будет размер таблицы.

Для того, чтобы выполнить описанное разделение управления кэшем и дисками, был выбран алгоритм из [21] в блоке 1. Цель алгоритма состоит в достижении заданных хит-рейтов для каждого приложения. Он вычисляет средние значения хит-рейтов для каждого приложения и передает их блоку 8 для расчета размеров памяти на дисках, как показано на рисунке 2.7. В свою очередь, блок 8 в конце своей работы выдает подправленные требования на хит-рейт блоку 1. На-

чальные значения для p_0^i и s_j^i задаются некоторым типовым образом, например, одинаковыми или пропорционально требованиям на отклик системы. До вычисления требуемых хит-рейтов алгоритм поиска в кэше не изменяет значения s_0^i.

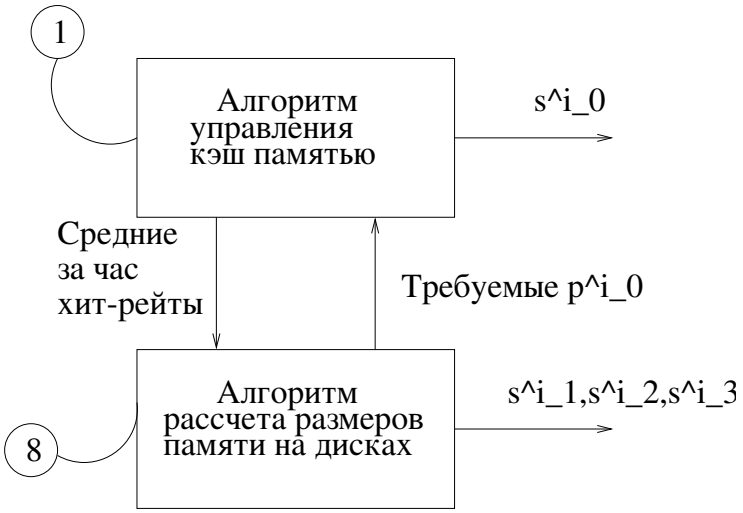


Рисунок 2.7 — Связь блоков управления кэшем и расчета размеров памяти на дисках

Перейдем к рассмотрению алгоритма расчета s_1^i, s_2^i, s_3^i предлагаемого в данной работе. Алгоритм вычисляет границы областей один раз в час. Это время выбрано исходя из того, что в существующих решениях перемещение данных между уровнями происходит не чаще упомянутого периода, что в свою очередь связано с затратами по времени и по ресурсам системы на выполнение данной операции. За час собирается статистика, вычисляется $F_{\text{pop}}^i(x)$ и вместе с текущими значениями границ по выражениям (2.5) – (2.7) рассчитываются вероятности p_1^i, p_2^i, p_3^i. Используя эти вероятности и p_0^i, которую выдает блок 1, с помощью выражения (2.2) вычисляем $\overline{T}_i$. Далее все $\overline{T}_i$ разделим на два списка. В первый список попадут $\overline{T}_i \geq R_i$, а во второй список $\overline{T}_i < \delta R_i$, где δ характеризует необходимый запас по требуемой задержке и представляет собой долю от R_i. В первый список T1 попадут задержки тех приложений, для которых требование R_i не выполнилось, а во второй список T2 задержки тех, для которых они выполнились с избытком. Если эти списки не пустые, то далее задача алгоритма состоит в таком изменении s_1^i, s_2^i, s_3^i, при котором для всех приложений требования R_i станут удовлетворяться. Для этого сначала списки T1 и T2 сортируется по возрастанию. Далее для приложений, чьи $\overline{T}_i$ оказались в начале списков, происходит изменение размеров областей на дисках. Изменения производятся пошагово, по одному слоту, начиная с

самой быстрой памяти. Сначала размер s_1^y приложения с избытком уменьшается на один слот, а размер s_1^x приложения с недостатком увеличивается на один слот. После этого по формуле (2.2) снова пересчитываются $\overline{T}_i$ для двух приложений. В таблице 2.1 представлены различные варианты значений $\overline{T}_i$ и дальнейшие шаги.

№	Условие	Действие
1	$\overline{T}_x > R_x, \overline{T}_y < \alpha R_y$	Продолжать увеличивать s_1^x и уменьшать s_1^y
2	$\overline{T}_x < R_x, \overline{T}_y < \alpha R_y$	Остановить перераспределение s_1^i
3	$\overline{T}_y > \alpha R_y$	Сделать шаг назад, т.е. уменьшить s_1^x и увеличить s_1^y

Таблица 2.1

Возможные события в результате перераспределения памяти

Параметр α необходим для того, чтобы приложение с запасом по $\overline{T}_i$ не стало приложением, не удовлетворяющим требованию. Если выполнилось условие 2 (таблица 2.1), то дальше выбирается следующая пара приложений с недостатком и избытком $\overline{T}_i$, причем, приложение с избытком может остаться тем же, что было на предыдущем шаге. Если выполнилось условие 3, то резерв быстрой памяти у приложения с избытком закончился. В этом случае после шага назад приложение с недостатком все еще не удовлетворяет требованию. Далее необходимо повторить процедуру перераспределения для следующего уровня памяти s_2^i. Если и в нем будет такой же результат, то выбирается следующее приложение с избытком и процедура перераспределения памяти повторяется с уровня s_1^i. Далее либо закончатся все приложения с недостатком быстрой памяти, что означает успешное завершение алгоритма, либо закончатся приложения с избытком, а с недостатком останутся. В последнем случае система не сможет удовлетворить текущим требованиям на задержку.

Работа алгоритма основана на предположении, что во время функционирования системы присутствуют достаточно длинные периоды стабильности $F_{\text{pop}}^i(x)$, тогда собрав за определенное время статистику, можно рассчитывать на то, что новые значения s_j^i будут актуальны.

Для того, чтобы быстрее реагировать на изменения потока запросов, в алгоритме предусмотрена процедура пересчета требуемого хит-рейта. Она работает каждые 10 минут, в то время, когда пересчет s_1^i, s_2^i, s_3^i не производится. По текущим значениям s_1^i, s_2^i, s_3^i с помощью формул (2.5) – (2.7) вычисляются p_1^i,

p_2^i, p_3^i. Далее из этих формул можно выразить вероятность p_0^i, которая и является хит-рейтом. Найденное значение передается в блок управления кэшем, который должен в результате работы выдерживать заданные p_0^i.

2.7 Сложность алгоритма

Представить точную оценку сложности алгоритма весьма затруднительно. При каждом запуске алгоритма количество выполненных шагов может сильно отличаться. Оно зависит от характеристик входного потока. Целью данного подраздела является вычисление очень грубой оценки для самой худшей ситуации, которая возможно даже не реализуема, чтобы получить представление о том, что данный алгоритм применим на практике.

Пусть имеется n приложений, работающих с системой. Как было описано ранее, по собранной статистике формируются два списка: приложения, не удовлетворяющие заданным требованиям, и приложения, удовлетворяющие заданным требованиям. Длина обоих списков порядка $O(n)$. В худшем случае для каждого приложения с недостатком необходимо выполнить перераспределение памяти с каждым приложением с избытком. Количество таких распределений порядка $O(n^2)$. Теперь оценим количество операций при каждом перераспределении памяти между парой приложений. Необходимо перебрать всего два уровня из трех, так как трейтий уровень получается изходя из размеров первых двух и общего объема памяти, выделенного приложению. Как было оценено ранее, размер таблицы для $F_{\text{pop}}^i(x)$ имеет порядок 10^2. Так как перебор по уровням происходит последовательно друг за другом, то количество операций суммируется. Следовательно, количество операций для двух уровней порядка $2 \cdot 10^2$.

Итак грубая оценка для полного количества операций по пересчета размеров сегментов памяти имеет порядок $O(2 \cdot 10^2 \cdot n^2)$.

Для $n = 10$, используемого в моделировании, количество операций имеет порядок $2 \cdot 10^4$.

2.8 Модель потока заявок

Представленная в данном разделе агентная модель была предложена автором в работе [7], где характеристики полученного потока заявок сравниваются с аналогичными характеристиками потока, полученного с помощью модели Тяпочкина К.

Описанные в подразделе 2.3 методы генерации потока адресов позволяют задать, либо только гистограму популярности либо гистограмму стековых расстояний. Предлагаемая агентная модель позволяет задавать обе гистограммы. Рассмотрим основную идею алгоритма. Все адреса разбиваются на группы по популярности. Для каждой группы задается гистограмма стековых адресов.

Для генерации очередного адреса сначала выбирается группа адресов, исходя из популярностей, а затем конкретный адрес генерируется с помощью стековой гистограммы. Одна группа и соответствующая ей гистограмма стековых расстояний представляет одного агента.

Агенты подобны пользователям системы хранения, работающим с определенной группой адресов в памяти. Для генерации потока адресов необходимо задать входные параметры модели:

- количество групп адресов, вероятности каждой группы;
- распределения стековых расстояний для каждой группы.

Для генерации потока необходимо выполнить следующие шаги:

1. выбрать группу адресов для запроса в соответствии с вероятностями групп;
2. использовать гистограмму стековых расстояний для выбранной группы адресов, чтобы сгенерировать очередной адрес;
3. перейти на шаг 1.

С помощью агентной модели можно сгенерировать поток адресов, подобный некоторому исходному потоку, т.е. гистограммы популярности и стековых расстояний нового потока будут близки к исходному потоку адресов. Для этого необходимо определить параметры модели по имеющемуся потоку адресов.

1. Подсчитать частоты появления каждого адреса в потоке.
2. Отсортировать адреса по частотам.

3. Разделить адреса на группы, чтобы в каждой группе адреса имели примерно одинаковые частоты.
4. Вычислить гистограмму стековых адресов, используя LRU-стек, как описано ранее. Для каждой группы используется отдельный стек. Например, адрес из 1-й группы попадает в 1-й стек, а адрес из 2-й группы попадает во 2-й стек.
5. Подсчитать вероятности появления в потоке адресов из каждой группы.

На рисунках 2.8, 2.9 представлены гистограммы стековых расстояний и популярности для исходного потока и потока, сгенерированного агентной моделью, а также для еще одного алгоритма, разработанного Тяпочкиным К. и описанного в работе [7]. Данный пример демонстрирует, что потоки отличаются друг от друга, но имеют схожие характеристики временной и пространственной локальности.

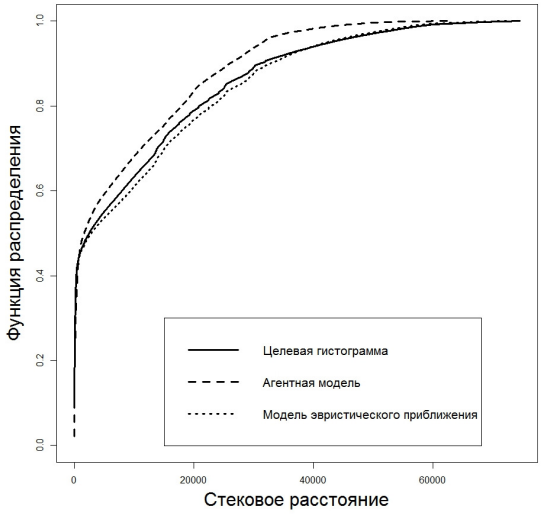


Рисунок 2.8 — Сравнение функций распределения стековых расстояний при разных алгоритмах генерации потоков

Достоинством агентной модели является относительно небольшая сложность генерации потока адресов. Агенты могут быть прототипами реальных приложений, работающих с системой хранения данных, что облегчает задание параметров модели.

К недостаткам модели можно отнести следующее. В случае, когда требуется задать общую для всего потока адресов гистограмму стековых расстояний, параметризация модели требует дополнительных шагов. С помощью модели невоз-

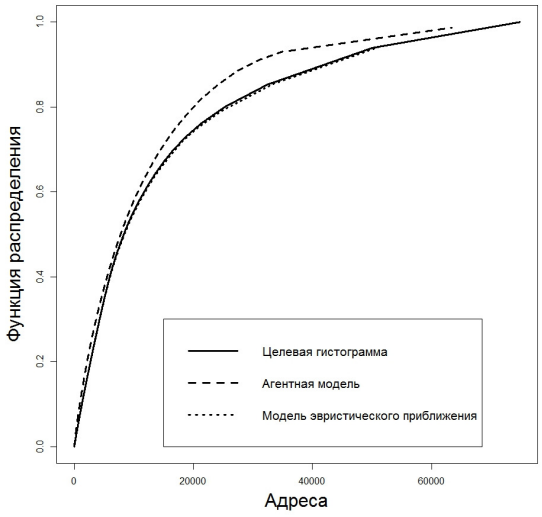


Рисунок 2.9 — Сравнение функций распределения популярностей при разных алгоритмах генерации потоков

можно сгенерировать серии последовательных чтений, которые могут встречаться среди запросов к системе хранения.

2.9 Критерий оценки алгоритма

Введем критерий, по которому будет происходить оценка работы предложенного алгоритма и сравнение его работы с системой без пересчета размеров областей на дисках. Введем коэффициент

$$q_i = \frac{N\left\{T_i \leq R_i\right\}}{N}, \tag{2.11}$$

где $N\left\{T_i \leq R_i\right\}$ – количество десятиминутных интервалов, в течение которых среднее время считывания блока данных не превышало заданное для приложения i, а N – общее количество десятиминутных интервалов работы системы. Введем коэффициент

$$Q = \frac{1}{n}\sum_{i=1}^{n} q_i, \tag{2.12}$$

который характеризует работу системы в целом по всем приложениям. Будем использовать его для сравнения работы системы с постоянными размерами s_j^i и работы системы с динамически изменяемыми размерами.

Для вычисления коэффициента Q будем использовать имитационное моделирование. Сама модель системы хранения описана в разделе 2.4.

2.10 Результаты моделирования системы хранения данных

Целью данного раздела является исследование эффективности работы предложенного алгоритма перераспределения памяти в многоуровневой системе хранения данных. В предыдущих разделах описан сам алгоритм, модель многоуровневой системы хранения и критерий оценки эффективности алгоритма.

Исходные данные для моделирования.

- Доля SSD дисков от общего объема памяти – 5%
- Доля производительных дисков от общего объема памяти – 20%
- Доля SATA дисков от общего объема памяти – 75%
- Доля уникальных адресов относительно объема SATA дисков – 30%
- Время доступа к SSD диску – $5 \cdot 10^{-5}$
- Время доступа к производительному диску – $5 \cdot 10^{-4}$
- Время доступа к SATA диску – 10^{-2}

Для оценки способности алгоритма адаптироваться к входному потоку, будем сравнивать его работу по описанному критерию с алгоритмом, у которого заранее есть вся информация о предстоящем потоке заявок. Второй алгоритм выполняет следующее:

- собирает статистику по заранее известному потоку заявок, чтобы получить распределения стековых расстояний и популярности адресов для каждого приложения за все время;
- вычисляет объемы памяти s_j^i для каждого клиента на каждом уровне памяти исходя из полученных распределений;
- значения s_j^i остаются неизменными в течение всей работы системы.

Таким образом, если значение Q для предложенного алгоритма будет равно таковому для неадаптивного алгоритма со статическими границами, значит алгоритм успевает адаптироваться к входному потоку.

Если значение Q для предложенного алгоритма будет выше, чем для неадаптивного алгоритма со статическими границами, значит за время работы системы поток настолько сильно менялся, что статические границы, полученные по усредненным характеристикам по всему потоку запросов, оказываются неэффективными на некоторых промежутках работы.

Сравнение будем производить как на синтетических, так и на реальных входных потоках, полученных от различных систем.

На рисунке 2.10 представлены пары коэффициентов Q для одного искусственного лога и для нескольких общедоступных логов реальных систем хранения. Один коэффициент получен при работе системы с выставленными s_j^i один раз в начале работы пропорционально требованиям на задержку (статические границы). Второй коэффициент получен при работе системы с предложенным алгоритмом пересчета s_j^i. Требование на задержку выбрано достаточно жестким ($Q < 1$), чтобы показать поведение алгоритмов в условиях нехватки ресурсов.

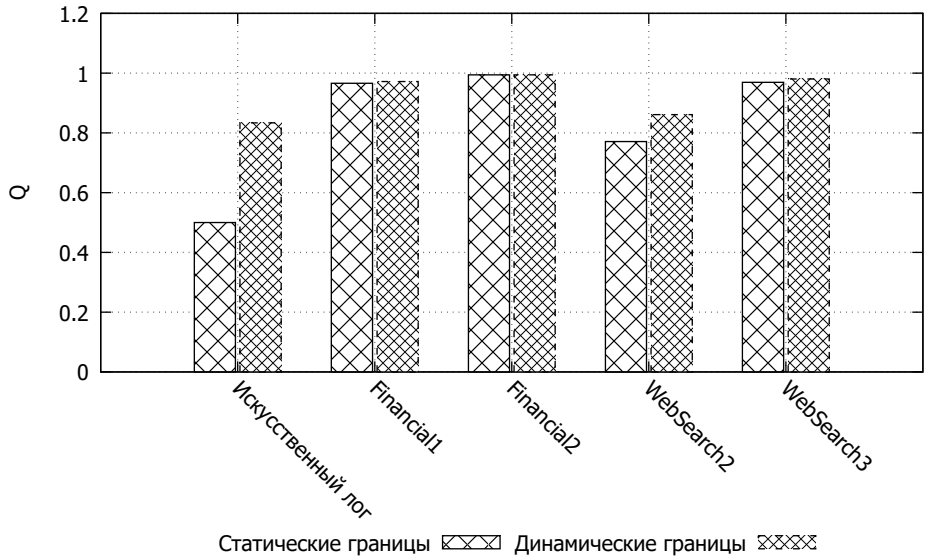


Рисунок 2.10 — Результаты моделирования предложенного алгоритма.

Логи Financial1, Financial2 сняты с систем обработки транзакций в реальном времени в финансовых организациях. Логи WebSearch2, WebSearch3 собраны с известных поисковых систем [46]. Как видно из рисунка, наибольший выигрыш получается на искусственном логе. Как было упомянуто выше, предла-

гаемый алгоритм опирается на предположение о наличии достаточно длинных периодов времени, на которых функция $F^i_{\text{pop}}(x)$ постоянна. В искусственном логе это условие строго выполняется, кроме того, он построен таким образом, что характеристики потока запросов очень сильно отличаются на первой половине работы системы и на второй. По этой причине выигрыш на синтетическом логе наибольший. В реальности поведение $F^i_{\text{pop}}(x)$ непредсказуемо.

На логах Financial1 и WebSearch3 наблюдается незначительный выигрыш предложенного алгоритма. На логе Financial2 разницы нет. Это связано с тем, что свободных для перераспределения ресурсов не было и выдерживались изначально выставленные границы. На логе WebSearch2 выигрыш предложенного алгоритма составляет 12%. Можно также отметить, что благодаря введенным параметрам α и δ, которые защищают от работы на грани, алгоритм не ухудшил работу системы хранения, по сравнению со статическим вариантом.

Рассмотрим теперь задержки системы, усредненные по часу, для адаптивного и статического алгоритма. На рисунках 2.11,2.12,2.13 представлены графики задержки операций чтения для потоков запросов Financial1, WebSearch2 и WebSearch3. Для каждого лога представлены данные лишь некоторых характерных приложений, в которых задержки отличаются для адаптивного и статического алгоритмов. На каждом графике представлены три кривые: задержка для адаптивного алгоритма, задержка для статического алгоритма и линия (обзначена как QoS), показывающая заданную задержку для приложения.

На графиках видно, что адаптивный алгоритм почти всегда держит кривую задержки ближе к заданной, что говорит о более эффективном использовании системы, т.к. если приложение имеет излишне низкую задержку, значит ему выделено больше быстрой памяти, чем требуется. Эти излишки направляются приложениям, задержка которых не укладывается в заданную.

2.11 Заключение и выводы по разделу

В разделе рассмотрена многоуровневая система хранения, которая является частью распределенной системы хранения данных. Поставлена и решена задача

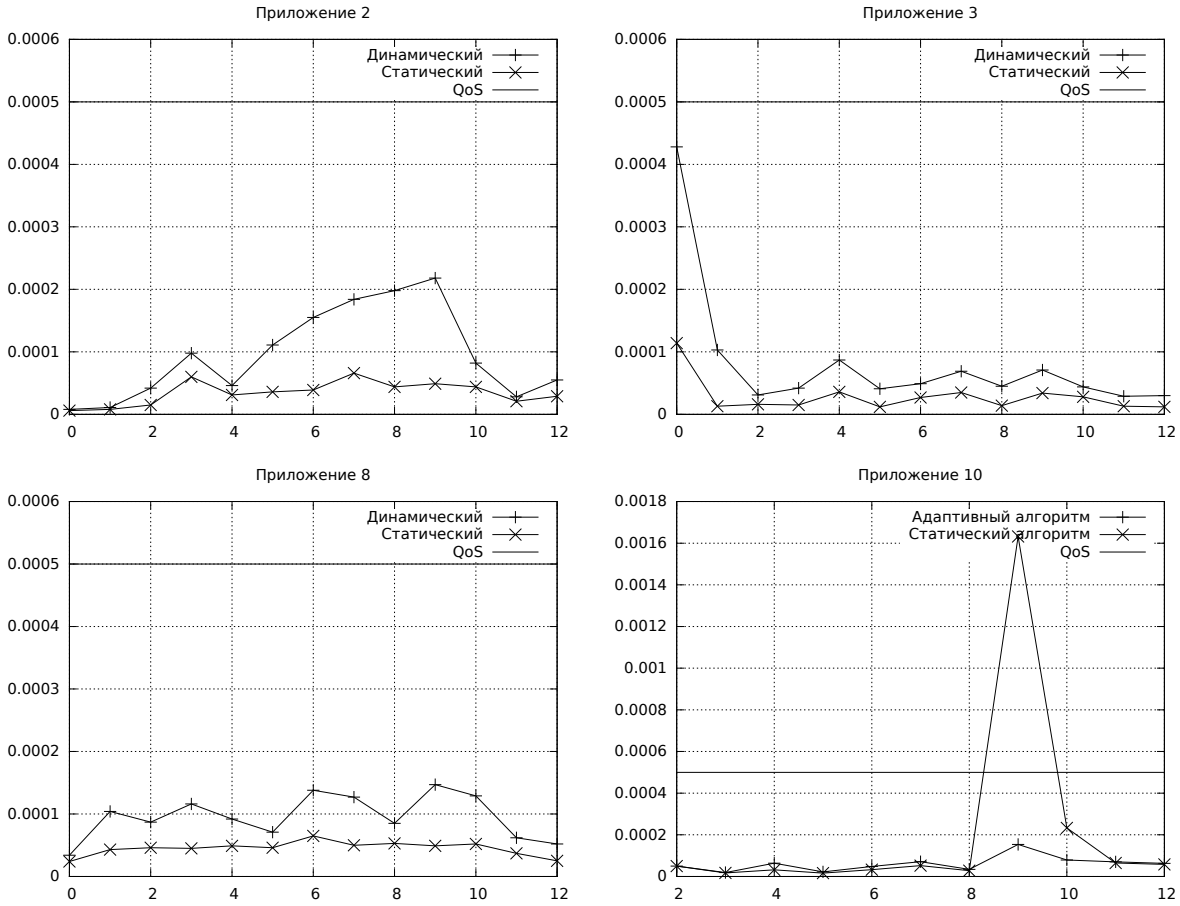


Рисунок 2.11 — Задержка системы для потока запросов Financial1

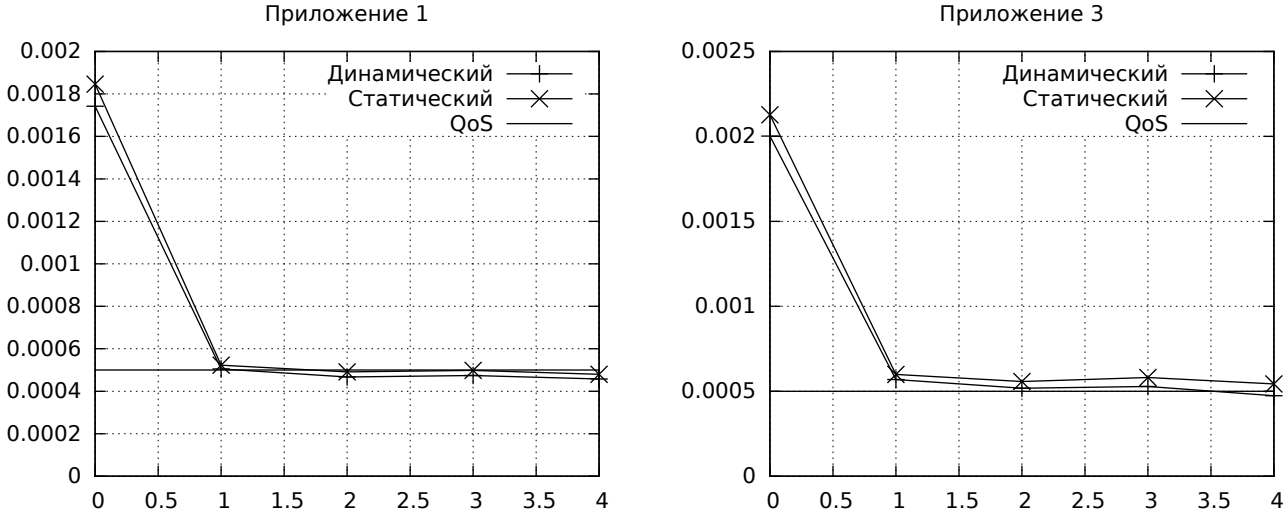


Рисунок 2.12 — Задержка системы для потока запросов WebSearch2

распределения памяти на дисках разного типа для каждого приложения с учетом заданных требований на задержку операций ввода/вывода.

Для решения задачи предложен алгоритм, который адаптируется к входному потоку заявок, оценивая распределения стековых расстояний и популярностей

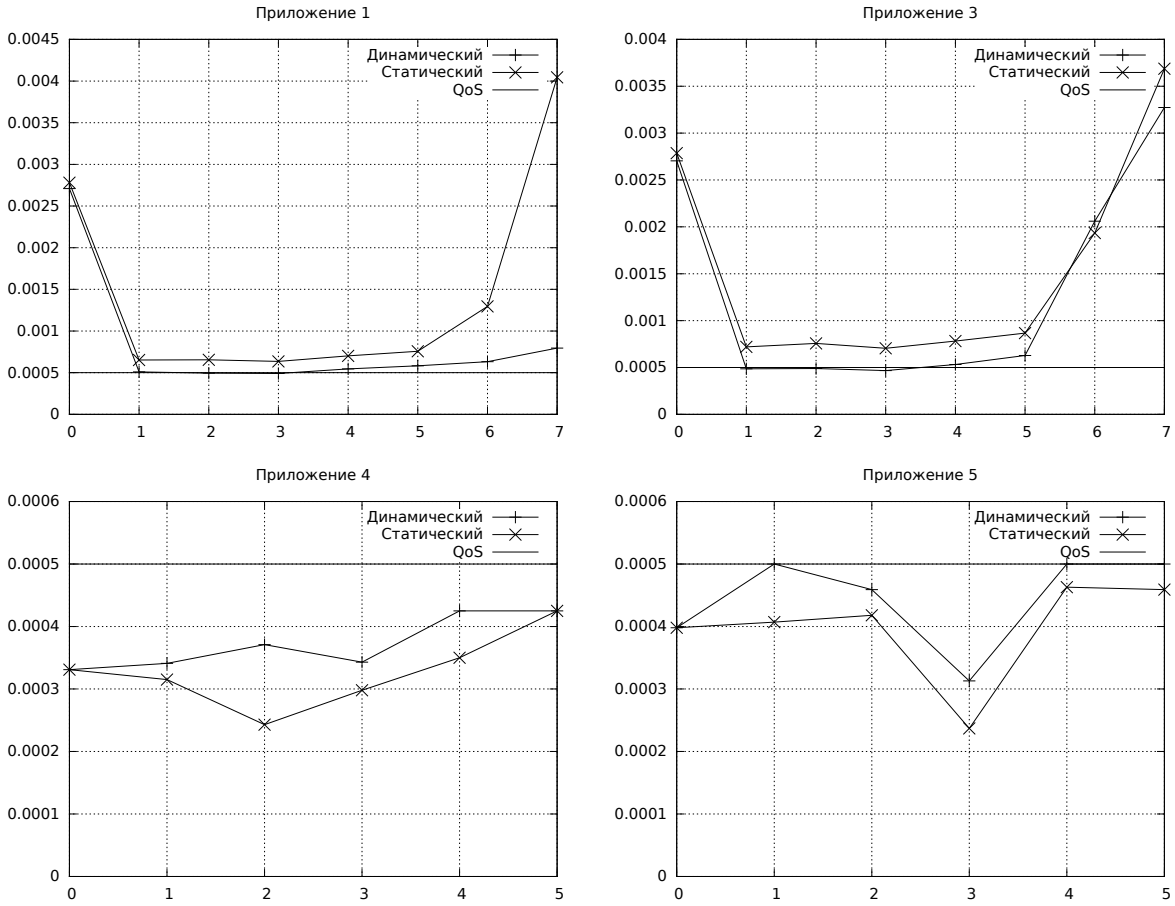


Рисунок 2.13 — Задержка системы для потока запросов WebSearch3

адресов. Оценка работы алгоритма проводилась с помощью представленной модели хранилища и предложенного критерия оценки.

Сравнение алгоритма расчета размеров памяти производилось с идеализированным алгоритмом, которому заранее известен весь входной поток запросов (чего обычно не бывает), что используется для вычисления соответствующих размеров памяти перед работой системы.

Предложен алгоритм генерации входного потока заявок с заданными распределениями стековых расстояний запросов приложений и популярностями групп адресов.

В качестве нагрузки использовались логи работы реальных систем хранения, а также синтетические логи, в том числе те, которые получены с помощью предложенной модели входного потока системы.

Предложенный алгоритм распределения памяти имеет свои ограничения применимости. Его работа основана на предположении о наличии более или менее длительных периодах стабильности распределения популярностей адресов. Если это выполняется, то алгоритм дает ощутимое преимущество. Это показа-

но с помощью моделирования на искусственном потоке запросов адресов. Однако использование потока запросов с реальных систем показало, что довольно часто алгоритм позволяет улучшить работу системы хранения и в этих случаях, а в остальных случаях не ухудшает ее. Положительный эффект достигается за счет того, что алгоритм находит приложения, у которых есть излишки быстрой памяти (требование на задержку выполняется с запасом) и распределяет эти излишки среди приложений, у которых требование на задержку не выполняется.

С ростом требований к задержке системы при прочих равных параметрах системы алгоритм в определенный момент перестанет выполнять это требование. В таком случае необходимо увеличивать общий объем быстрой памяти либо снижать требования. Сочетание таких параметров как объемы памяти на каждом уровне системы и заданные требования на задержку, при котором алгоритм перестает справляться с задачей, задают границы его применимости.

3 ТРАНСПОРТНОЕ КОДИРОВАНИЕ КАК СРЕДСТВО УМЕНЬШЕНИЯ ЗАДЕРЖКИ В СЕТИ

Как было отмечено раннее, в распределенной системе хранения задержка складывается из задержки на узле и задержки в сети. В данном разделе исследуется вопрос задержки в сети.

В результате обзора работ, посвященных данной теме, было выделено два метода передачи информации, на основе которых строится сетевое взаимодействие в распределенной системе хранения: сетевое кодирование и транспортное кодирование.

Данный раздел посвящен транспортному кодированию. Предлагается модификация модели сети с коммутацией пакетов, разработанная в [47], [48], которая снимает с прошлой модели часть ограничений, таких как экспоненциальная задержка пакетов, равномерные потоки в сети и одинаковые емкости каналов. Анализируется задержка сообщений при задержке пакетов, распределенной по закону Эрланга и по нормальному закону.

На новой модели проведено исследование эффективности транспортного кодирования при таких эффектах как изменение емкостей каналов и ограниченное время жизни пакетов, которые присутствуют в реальных сетях. Исследована эффективность кодирования на модели Клейнрока с топологией типа решетка.

Рассмотрено применение транспортного кодирования в нерегулярных сетях и предложена модификация транспортного кодирования, которая позволяет увеличить выигрыш от кодирования в сетях с нерегулярной структурой.

Структура раздела выглядит следующим образом. В подразделе 3.1 выполнен обзор работ, посвященных уменьшению задержки при передаче в распределенных системах хранения. Подраздел 3.2 посвящен описанию известной модели сетей с коммутацией пакетов, представленной Л. Клейнроком. В подразделе 3.3 описывается метод передачи с помощью кодирования на транспортном уровне. В подразделе 3.4 описывается математический аппарат порядковых статистик, который используется в подразделе 3.5 для анализа задержки сообщения при передаче с помощью транспортного кодирования. В подразделе 3.6 рассматриваются неэкспоненциальные модели задержки пакетов для анализа задержки сообщений, эффективность кодирования сравнивается с экспоненциальной моделью, описан-

ной в подразделе 3.5. Далее в подразделе 3.7 описывается имитационная модель сети Клейнрока. В подразделе рассматривается простейшая модель сети, в которой потоки сообщений и загрузка одинаково распределены по всей сети. Приводятся результаты моделирования, которые сравниваются с результатами из 3.5 и 3.6. В подразделе 3.8 рассматривается более сложная модель с топологией типа решетка. Эффективность транспортного кодирования сравнивается с рассмотренными ранее более упрощенными моделями. В подразделе 3.9 в модель сети добавляется эффект изменения емкостей каналов. Исследуется влияние изменения емкостей на применимость и эффективность транспортного кодирования. В подразделе 3.10 в модель сети добавляется ограничение на время жизни пакета и исследуется его влияние на транспортное кодирование. В подразделе 3.11 в модель сети добавляется нерегулярность и предлагается изменение метода передачи для повышения эффективности транспортного кодирования. Приводятся результаты моделирования. В подразделе 3.12 даются выводы по разделу.

3.1 Обзор существующих работ

При организации сетевого взаимодействия узлов распределенных систем хранения распространено применение кодов, исправляющих стирания. Это позволяет как повысить надежность передачи, так и уменьшить задержку. При этом используется схема, которая в российских работах применялась в рамках транспортного кодирования, а в зарубежных статьях известна под разными названиями: dispersity routing [49–53] – применительно к передаче информации по сети, fork-join queueing [54] – применительно к анализу распределенных систем.

Рассмотрим суть этой схемы на примере распределенной системы хранения. Пусть имеется n узлов в распределенной системе хранения. Каждый узел представляет собой хранилище данных. Тогда для совершения успешной операции чтения достаточно получить ответ от любых k узлов. Другими словами, информация распределяется по n узлам, но достаточно выполнить чтение из любых k. Это достигается за счет использования кодов, исправляющих стирания.

Информация делится на k блоков, затем кодируется, в результате чего получается n блоков, которые записываются на n узлах. Для реализации схемы k

из n необходимы так называемые разделимые коды с максимальным расстоянием или МДР коды [55]. Самыми известными представителями этого семейства являются коды Рида-Соломона [55]. Основным недостатком этих кодов является относительно высокая сложность декодирования, поэтому стремятся построить МДР коды или близкие к ним по характеристикам коды, обладающие низкой сложностью декодирования. К таким кодам относятся фонтанные коды и raptor коды [56–60].

В большинстве работ коды, исправляющие стирания, применяются для надежного хранения и являются альтернативой для репликации [61], [62], [63], [64]. Данное направление привело к созданию кодов для систем хранения [65], [66], [67], [68], [69], [70], [71]. Коды также используются для организации распределенного хранения [72], [73], в системах распространения мультимедиа данных [74] [75]. В работе [76] рассматривается задача хранения нескольких версий данных в распределенных системах.

При исследовании вопроса передачи в распределенных системах хранения во многих работах рассматривается сетевое кодирование. По сравнению с классическим методом передачи оно позволяет повысить пропускную способность сети. Впервые метод был представлен в работе Ahlswede [77]. Среди российских исследователей сетевому кодированию ряд работ посвятил Э.М. Габидулин [78], [79], [80], [81], [82], [83]. Применение сетевого кодирования к распределенным системам хранения рассмотрено в работах [84], [85], [86], [87].

Другой рассматриваемый в работах метод передачи – транспортное кодирование. Впервые транспортное кодирование было представлено Г.А. Кабатянским и Е.А. Круком в работах [47], [48]. Дальнейший анализ и развитие оно получило в работах [88], [89], [90], [91], [92], [93], [94], [95]. Транспортное кодирование было представлено еще до появления сетевого кодирования. В большинстве зарубежных работ по транспортному кодированию рассматривается задача надежной и быстрой передачи [53; 96–98]. Как правило, в качестве области применения рассматривают системы эффективной доставки мультимедиа данных нескольким адресатам [99; 100].

В данной работе рассматривается транспортное кодирование как средство уменьшения задержки сообщений.

3.2 Модель сети с коммутацией пакетов Клейнрока

Рассмотрим сеть ЭВМ — множество узлов, в которых располагаются вычислительные ресурсы, соединенные каналами передачи данных. Примерами узлов могут служить персональные компьютеры, планшетные компьютеры, смартфоны, сетевое оборудование. В качестве каналов передачи информации могут использоваться как проводные, так и беспроводные каналы связи. Сети бывают централизованные – доступом к ресурсам сети управляет особый узел, называемый сервером, и децентрализованные, или одноранговые – каждый узел может выполнять роль как сервера, так и рабочей станции. Сети могут быть мобильными и стационарными. В мобильных сетях со временем меняется топология сети, а в стационарных она фиксирована.

Сети условно можно разделить на три типа: сети с коммутацией каналов, сообщений и пакетов. В данной работе будут рассматриваться сети с коммутацией пакетов. В таких сетях сообщение делится на некоторое количество пакетов, которые нумеруются и снабжаются адресом. Каждый пакет прокладывает себе путь по сети. Передача каждого пакета осуществляется движением от узла к узлу согласно алгоритму маршрутизации, пока пакет не достигнет получателя. Если канал, ведущий к следующему узлу, занят, то сообщение стоит в очереди и ждет освобождения канала. Узел-получатель ждет прихода всех пакетов и собирает из них сообщение. Таким образом, в сети с коммутацией пакетов пакеты одного и того же сообщения передаются одновременно и могут быть доставлены узлу-получателю в любом порядке.

При оценке сетей с коммутацией пакетов рассматривают следующие количественные характеристики сети: задержка, скорость передачи, стоимость и надежность. В данной работе интерес представляет средняя задержка пакетов и сообщений. Задержка представляет собой полное время передачи от узла-источника к узлу-получателю. Известной моделью для анализа сетей ЭВМ является модель Л. Клейнрока [101].

Далее приводится описание этой модели, представленное в работе [88]. В последующих подразделах эта модель будет использоваться и модифицироваться.

Пусть сеть имеет M каналов и N узлов. В узлах могут образовываться сообщения для других узлов. Эти сообщения передаются по каналам от источника до получателя через промежуточные узлы. Каналы считаются абсолютно надежными и бесшумными. Пропускная способность i-го канала равна C_i бит/сек. Узлы выполняют прием и обработку пакетов, выбор маршрутов, хранение пакетов в буферах. Обработка в узлах считается безошибочной и мгновенной. Если канал, по которому нужно послать пакет, занят, то сообщение поступает в буфер. Так образуются очереди пакетов на каждый канал. Как только канал освобождается, в него посылается первый пакет из очереди. Трафик, поступающий в сеть от внешних источников в узел j и предназначенный узлу k, имеет интенсивность γ_{jk} (пакетов в секунду). Полный внешний трафик сети определяется как

$$\gamma = \sum_{j=1}^{N} \sum_{k=1}^{N} \gamma_{jk}. \tag{3.1}$$

В модели Клейнрока действует допущение о независимости длин пакетов. Оно позволяет упростить модель и дает возможность выполнить для нее расчет. Рассмотрим это допущение подробнее. По предположению о независимости длины всех пакетов независимы и распределены по экспоненциальному закону со средним значением $1/\mu$ бит. Это означает, что, каждый раз, когда пакет принимается узлом, генерируется новая длина для пакета по экспоненциальному закону распределения с плотностью:

$$p(b) = \mu e^{-\mu b}, b \geq 0 \tag{3.2}$$

Далее пакет передается с такой длиной следующему узлу, в котором повторяется выбор длины пакета. Это предположение делает время передачи по каналу и задержку пакета случайной величиной. Так как мы рассматриваем передачу по каналу как обслуживание заявок в сети массового обслуживания (СМО), важно иметь возможность рассматривать каждую СМО независимо от других. Для этого нужна независимость интервалов поступления заявок в СМО от времени их обслуживания. Рассматриваемое допущение дает такую независимость и возможность рассматривать каждый канал как СМО типа M/M/1.

В любой реальной сети пакет имеет постоянный размер и время прохождения по каналу величина постоянная, а времена поступления заявок зависят от

времени обслуживания. На практике эта зависимость может быть уменьшена, если пакеты приходят в буфер одного и того же канала из разных источников, либо если идущие друг за другом пакеты расходятся по разным каналам. Клейнроком было показано [101], что для сетей со средней связностью предположение о независимости справедливо. Также Клейнроком было показано, что влияние предположения о независимости на среднюю задержку пакета в сети пренебрежимо мало для большинства сетей.

Таким образом, будем рассматривать каналы в виде СМО, функционирующих независимо друг от друга. Обслуживателем является канал, а очередь образуют пакеты, поступающие в буфер перед каналом. Пусть по каждому каналу проходит в среднем λ_i пакетов в секунду. Тогда полный внутренний трафик сети определяется как

$$\lambda = \sum_{i=1}^{M} \lambda_i. \tag{3.3}$$

Считается, что трафик представляет собой пуассоновский поток. У Клейнрока проведен анализ средней задержки пакета в сети [101]. Общая формула для средней задержки пакета выглядит следующим образом:

$$T = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} T_i, \tag{3.4}$$

где T_i — это среднее время передачи по каналу с учетом нахождения в очереди. Так как в рассматриваемом случае канал можно представить в виде системы массового обслуживания M/M/1, то T_i можно найти из выражения:

$$T_i = \frac{1}{\mu C_i - \lambda_i}. \tag{3.5}$$

Тогда из формулы 3.4 получаем

$$T = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} \frac{1}{\mu C_i - \lambda_i}. \tag{3.6}$$

Существуют и другие модели сети. Так, например, в работе [102] рассматривается модель мобильной сети, в которой узлы перемещаются друг относительно друга.

3.3 Кодирование на транспортном уровне

Рассмотрим кодирование на транспортном уровне, предложенное в работах [47], [48]. Пусть требуется передать по сети сообщение, состоящее из k пакетов. Для этого k пакетам сообщения применяется кодирование некоторым (N, k) кодом, после чего пакетов становится N штук ($N > k$). В сеть вместо k пакетов посылается N, тем самым вносится избыточность в сообщение. Эта избыточность может быть использована для детектирования и исправления ошибок. Узел-получатель восстанавливает исходное сообщение по принятым пакетам, некоторые из которых могут быть искажены. Восстановление происходит путем декодирования (N, k) кода. При использовании разделимого кода с максимальным расстоянием (МДР код) достаточно получить любые k из N пакетов, чтобы восстановить сообщение. Таким образом, описанный метод позволяет бороться с искажениями и потерями пакетов в сети. На первый взгляд кажется, что применение данного метода в бесшумных и абсолютно надежных сетях бессмысленно, но в работе [89] показывается, что метод позволяет уменьшать задержку сообщений в сети. Так как разбиение сообщения на пакеты и сборка из пакетов обратно происходит на транспортном уровне, то и кодирование пакетов сообщения происходит тоже на транспортном уровне. Отсюда возникло название метода “транспортное кодирование”.

В работах [88], [89] приведен анализ задержки сообщений при использовании транспортного кодирования. При анализе задержки используется модель сети, основанная на описанной выше модели Клейнрока. В сети также имеется M каналов и N узлов. Предполагается, что все каналы имеют одинаковую емкость c. Внешний трафик, поступающий в i-й узел сети и предназначенный для j-го узла, представляет собой пуассоновский поток со средним γ_{00} пакетов в секунду. Тогда внешний трафик i-го узла определяется как

$$\gamma_0 = N\gamma_{00}, \tag{3.7}$$

а полный внешний трафик сети вычисляется по формуле

$$\gamma = N\gamma_0 = N^2\gamma_{00}. \tag{3.8}$$

Считается, что увеличение внешнего трафика сети приводит к такому же увеличению нагрузки на ее отдельные каналы. Считается, что поток проходящий по всем каналам одинаков и имеет среднее значение λ пакетов в секунду. Тогда полный внутренний трафик сети равен

$$\lambda_0 = M\lambda. \tag{3.9}$$

Теперь вычислим среднюю задержку отдельного пакета по формуле 3.6

$$\begin{aligned} T &= \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} \frac{1}{\mu C_i - \lambda_i} = \sum_{i=1}^{M} \frac{\lambda}{\gamma} \frac{1}{\mu c - \lambda} = \\ &= \frac{M\lambda}{\gamma} \frac{1}{\mu c - \lambda} = \frac{\lambda_0}{\gamma} \frac{1}{\mu c - \lambda}. \end{aligned} \tag{3.10}$$

Выразим эту формулу через нагрузку на канал $\rho = \frac{\lambda}{\mu c}$:

$$\bar{t}(\rho) = \frac{\lambda_0}{\mu c \gamma} \frac{1}{1 - \rho}. \tag{3.11}$$

Для расчета задержки сообщения при использовании транспортного кодирования используется допущение об экспоненциальном законе распределения задержки пакетов [88]:

$$F_\rho(t) = 1 - e^{-t/\bar{t}(\rho)}, t \geq 0. \tag{3.12}$$

Также используется математический аппарат порядковых статистик. Он описан в следующем разделе. Дальнейший анализ средней задержки сообщения при использовании транспортного кодирования приведен в разделе 3.5.

3.4 Порядковые статистики

Для оценки задержки сообщения необходимо оценивать время k-го по счету пакета. В данном разделе описываются порядковые статистики [103], которые нужны для расчета времени прихода k-го по счету пакета.

Пусть X_1, X_2, …, X_n некоторые случайные величины. Упорядочим их по возрастанию значений:

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)}. \tag{3.13}$$

Тогда X_i называется i-й порядковой статистикой. В дальнейшем будем обозначать i-ю порядковую статистику как $X_{i:n}$, чтобы показать объем выборки.

Пусть $P(x) = P\left(X \leq x\right)$ — функция распределения случайной величины X, а $p(x)$ — плотность этого распределения. Тогда функция распределения $X_{k:n}$ выглядит следующим образом [103]:

$$P_{k:n}(x) = P\left\{X_{(k)} \leq x\right\} = \sum_{i=k}^{n} \binom{n}{i} P^i(x) \left(1 - P(x)\right)^{n-i}. \tag{3.14}$$

В частном случае, когда $k = n$, функция распределения будет

$$P_{n:n}(x) = P^n(x). \tag{3.15}$$

Математическое ожидание $X_{k:n}$ можно найти по формуле

$$\begin{aligned} M\left[X_{k:n}\right] &= \int_{-\infty}^{\infty} x f_k(x) dx = \\ &= n\binom{n-1}{k-1} \int_{-\infty}^{\infty} x [P(x)]^{k-1} \left[1 - P(x)\right]^{n-k} dP(x). \end{aligned} \tag{3.16}$$

После подстановки $u = P(x)$ в предыдущую формулу получим

$$M\left[X\right] = n\binom{n-1}{k-1} \int_0^1 P^{-1}\left(u\right) u^{k-1} (1-u)^{n-k} du, \tag{3.17}$$

где $u = P(x)$.

3.5 Транспортное кодирование как средство уменьшения задержки сообщений

Для модели, описанной выше, вычислим задержку сообщения при кодированной и некодированной передаче. Приводимый ниже анализ был представлен в работах [89], [88].

Пусть сообщение состоит из k пакетов. Обозначим время прихода i-го пакета как t_i. В рассматриваемой модели величины t_i независимы и распределены по

экспоненциальному закону $F_\rho(t)$. При некодированной передаче задержка сообщения T равна времени прихода k-го пакета из k. Таким образом время прихода некодированного сообщения — это k-я порядковая статистика совокупности из k величин. Средняя задержка сообщения $\bar{T}$ может быть выражена как математическое ожидание $T_{k:k}$:

$$\bar{T} = M[T_{k:k}]. \tag{3.18}$$

Согласно (3.12) и (3.15) функция распределения для величины $T_{k:k}$ равна

$$F_{k:k}(t) = \left(1 - e^{-\frac{t}{\bar{t}(\rho)}}\right)^k. \tag{3.19}$$

Тогда плотность распределения

$$f_{k:k}(t) = k\left(1 - e^{-\frac{t}{\bar{t}(\rho)}}\right)^{k-1} \frac{1}{\bar{t}\,(\rho)} e^{-\frac{t}{\bar{t}(\rho)}}. \tag{3.20}$$

Получаем

$$M[T_{k:k}] = \int_{-\infty}^{\infty} t \cdot k\left(1 - e^{-\frac{t}{\bar{t}(\rho)}}\right)^{k-1} \frac{1}{\bar{t}(\rho)} e^{-\frac{t}{\bar{t}(\rho)}} dt. \tag{3.21}$$

После вычисления интеграла и упрощения выражения получим

$$\bar{T} = M[T_{k:k}] = \bar{t}(\rho) \sum_{i=1}^{k} i^{-1}. \tag{3.22}$$

Далее можно воспользоваться формулой для суммы гармонического ряда [104]

$$\sum_{i=1}^{k} i^{-1} = \ln k + \epsilon + \frac{1}{2k} - \sum_{l=2}^{\infty} \frac{A_l}{k(k+1)\ldots(k+l-1)}, \tag{3.23}$$

где $\epsilon = 0.577$ — это постоянная Эйлера, а

$$A_l = \frac{1}{l} \int_0^1 x(1-x)\ldots(l-1-x)dx.$$

Согласно (3.11) и (3.23) выражение (3.22) может быть переписано:

$$\bar{T} \geq (\ln k + \epsilon)\bar{t}(\rho) = \frac{\lambda}{\mu c \gamma} \frac{1}{1-\rho} (\ln k + \epsilon). \tag{3.24}$$

Средняя задержка для кодированных сообщений представляет собой математическое ожидание k-й порядковой статистики из n независимых случайных, величин распределенных экспоненциально:

$$\bar{T} = M\left[T_{k:n}\right]. \tag{3.25}$$

При использовании кодирования загрузка каждого канала возрастает, так как каждый узел добавляет избыточность в посылаемые сообщения. Для учета этого факта положим загрузку равной ρ/R, где R — скорость кода. Согласно (3.12) и (3.14) функция распределения для величины $T_{k:n}$ равна

$$F_{k:n}(t) = \sum_{i=k}^{n} \binom{n}{i} \left(1 - e^{-\frac{t}{\bar{t}(\rho)}}\right)^i e^{-\frac{(n-i)t}{\bar{t}(\rho)}}. \tag{3.26}$$

Тогда плотность распределения $T_{k:n}$

$$f_{k:n}(t) = n\binom{n-1}{k-1} \left(1 - e^{-\frac{t}{\bar{t}(\rho)}}\right)^{k-1} \frac{1}{\bar{t}(\rho)} e^{-\frac{(n-i)t}{\bar{t}(\rho)}} e^{-\frac{t}{\bar{t}(\rho)}}. \tag{3.27}$$

Средняя задержка кодированного сообщения согласно (3.25) и (3.17)

$$\begin{aligned} \bar{T} &= M\left[T_{k:n}\right] = \int_{-\infty}^{\infty} t \cdot n\binom{n-1}{k-1} \left(1 - e^{-\frac{t}{\bar{t}(\rho/R)}}\right)^{k-1} \cdot \\ &\cdot \frac{1}{\bar{t}(\rho/R)} e^{-\frac{(n-i)t}{\bar{t}(\rho/R)}} e^{-\frac{t}{\bar{t}(\rho/R)}} = \\ &= \frac{1}{\bar{t}(\rho/R)} n\binom{n-1}{k-1} \cdot \Big|_{-\infty}^{\infty} \sum_{i=0}^{k-1} t\binom{k-1}{i} \cdot \\ &e^{-\frac{t\bar{t}(\rho/R)(k-1)}{(n-k+1)(k-1-i)+n\cdot i}\left(-\bar{t}(\rho/R)\frac{(k-1)}{(n-k_1)(k-1-i)+n\cdot i} - \bar{t}(\rho/R)^2 \frac{(k-1)^2}{((n-k_1)(k-1-i)+n\cdot i)^2}\right)}. \end{aligned} \tag{3.28}$$

Упростив это выражение [103], получим

$$\bar{T} = \bar{T}_{k:n} = \bar{t}(\rho/R) \sum_{i=n-k+1}^{n} i^{-1} = \bar{t}(\rho/R) \left(\sum_{i=1}^{n} i^{-1} - \sum_{i=1}^{n-k} i^{-1}\right). \tag{3.29}$$

Используя (3.23), для разности получим

$$\sum_{i=1}^{n} i^{-1} - \sum_{i=1}^{n-k} i^{-1} \leq \ln n - \ln(n-k) = \ln \frac{1}{1-R}. \tag{3.30}$$

Так как объем трафика возрастает в $1/R$ раз, то

$$\bar{t}(\rho/R) = \frac{\lambda/R}{\gamma/R} \frac{1}{\mu c - \lambda/R} = \frac{\lambda}{\gamma\mu c} \frac{1}{1-\rho/R} = \frac{\lambda}{\gamma\mu c} \frac{R}{R-\rho}. \tag{3.31}$$

Используя (3.29) и это выражение, получим

$$\bar{T} = \bar{T}_{k:n} \leq \min_{R} \left\{ \frac{\lambda}{\gamma\mu c} \frac{R}{R-\rho} \ln \frac{1}{1-R} \right\} \tag{3.32}$$

Определим выигрыш от кодирования как отношение задержек сообщений:

$$T_1/T_2 = \frac{\bar{T}_{k:k}}{\bar{T}_{k:n}}. \tag{3.33}$$

На рисунках 3.1-3.3 представлены графики для выигрыша от кодирования в зависимости от скорости кода. Как видно из графиков, существует оптимальная скорость кода, при которой выигрыш максимален. При разных загрузках сети значение оптимальной скорости кода разная.

Максимальный выигрыш от кодирования обозначим как

$$G_{max} = \max_{R} \frac{\bar{T}_{k:k}}{\bar{T}_{k:n}}. \tag{3.34}$$

Минимизируя по R выражение (3.32), получим значение скорости кода, при которой $\bar{T}_{k:n}$ минимальна:

$$R = 2\rho/(1+\rho). \tag{3.35}$$

Подставив полученное значение скорости кода в выражение (3.32), получим

$$\bar{T}_{k:n} \leq \frac{\lambda}{\gamma\mu c} \frac{2}{1-\rho} \ln \frac{1+\rho}{1-\rho} \leq \frac{\lambda}{\gamma\mu c} \frac{4\rho}{(1-\rho)^2} \tag{3.36}$$

Транспортное кодирование выгодно использовать, когда выполняется неравенство $\bar{T}_{k:n} < \bar{T}$. Учитывая выражения (3.24) и (3.36), условие выгодности кодиро-

вания принимает вид

$$\frac{2}{1-\rho}\ln\frac{1+\rho}{1-\rho} < \ln k + \epsilon. \tag{3.37}$$

Из выражения (3.37) видно, что выигрыш от кодирования зависит от нагрузки сети ρ и числа пакетов в сообщении k. На рисунке 3.4 видно, что выигрыш от кодирования уменьшается с ростом загрузки сети. График на рисунке 3.5 показывает, что выигрыш от кодирования возрастает с ростом числа k информационных пакетов.

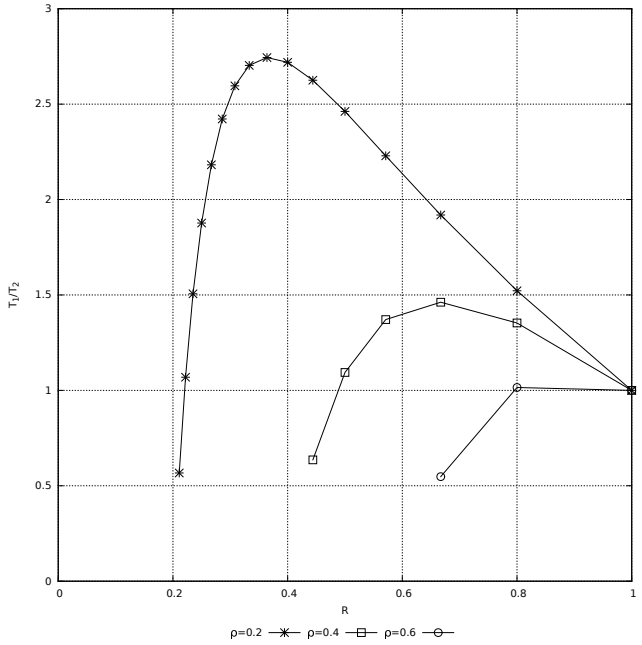


Рисунок 3.1 — Выигрыш от кодирования в зависимости от скорости кода R при $k = 4$.

3.6 Неэкспоненциальные модели сети

В работе [47] принято допущение об экспоненциальном распределении задержки пакетов в сети. Оно позволяет получить простые аналитические зависимости для выигрыша от кодирования. С одной стороны ввод этого допущения имеет свое обоснование [88]. С другой стороны для модели сетей Клейнрока, на которой основана модель из [47], не доказано, что распределение задержки сообщения носит экспоненциальный характер. Более того, получение аналитического закона распределения для сетей Клейнрока представляется сложной задачей.

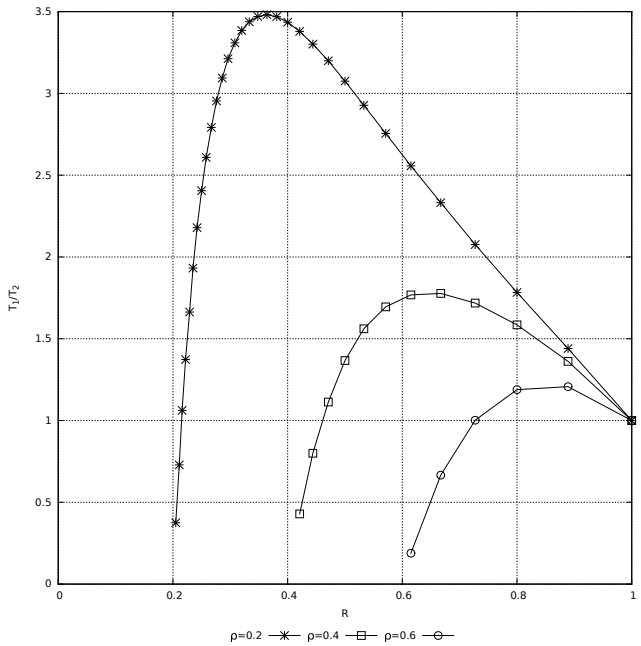


Рисунок 3.2 — Выигрыш от кодирования в зависимости от скорости кода R при $k = 8$.

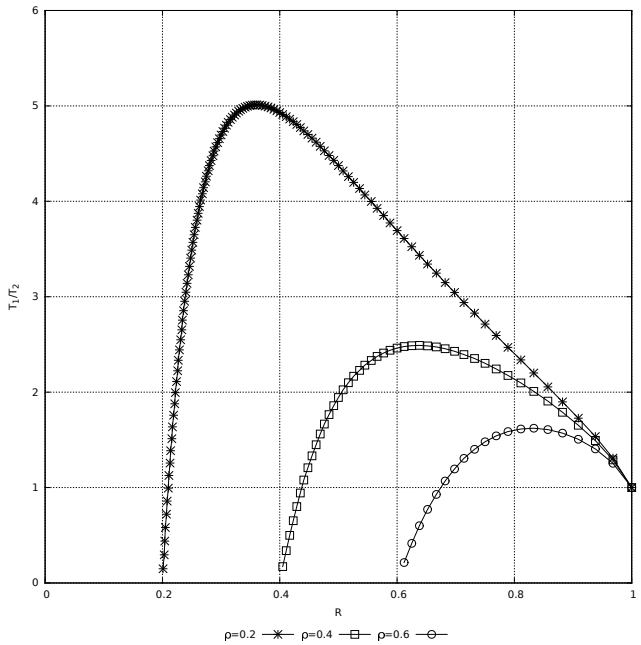


Рисунок 3.3 — Выигрыш от кодирования в зависимости от скорости кода R при $k = 30$.

В данном подразеделе рассматриваются другие законы распределения задержки пакетов, отличные от экспоненциального. Проводится анализ выигрыша от транспортного кодирования при данных распределениях. В подразделе 3.6.1 рассматривается распределение Эрланга, т.к. оно при некоторых параметрах сети соответствует распределению задержки пакетов для сети Клейнрока, что будет показано. При длине пути равной единице, этот закон распределения сводится к

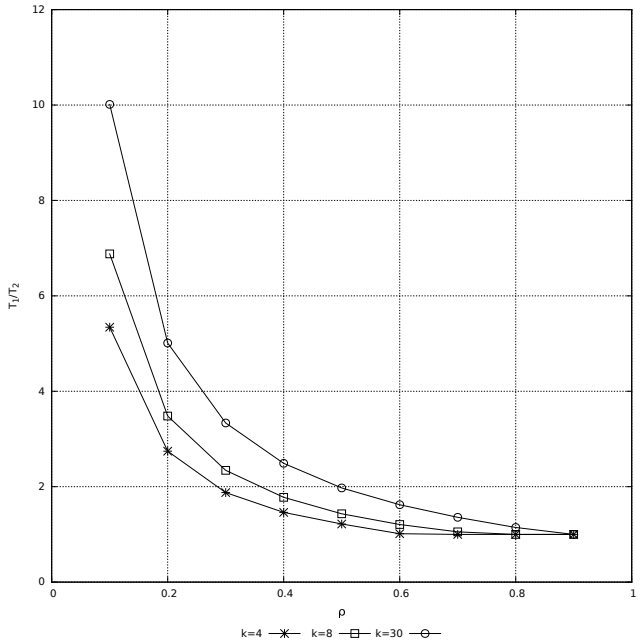


Рисунок 3.4 — Выигрыш от кодирования в зависимости от загрузки сети.

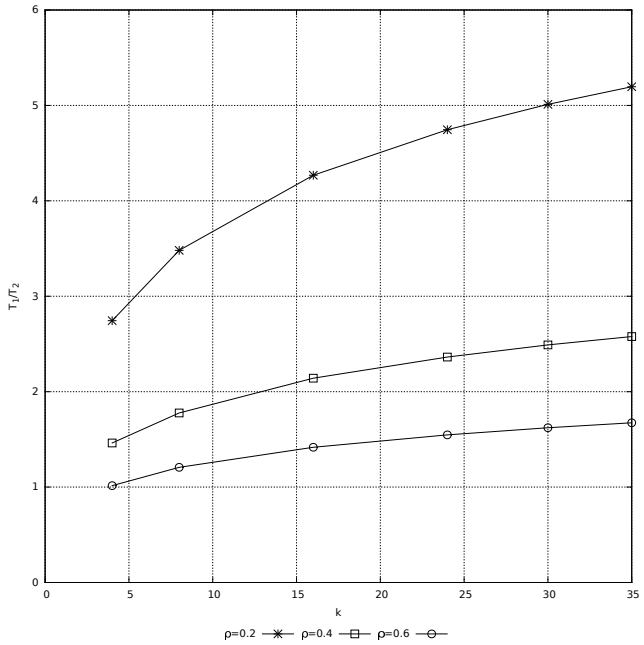


Рисунок 3.5 — Выигрыш от кодирования в зависимости от k.

экспоненциальному. В подразделе 3.6.2 рассматривается нормальное распределение задержки пакетов. Нормальное распределение возникает при рассмотрении закона Эрланга при достаточно больших длинах путей, так как известно, что с ростом порядка распределение Эрланга стремится к нормальному.

Результаты представленные в данном подразделе были изложены автором в работах [3; 5].

3.6.1 Распределение задержки пакетов по закону Эрланга

Рассмотрим, из чего складывается задержка пакета в модели сети Клейнрока. Для простоты рассмотрения начнем с передачи одного пакета по пустой сети. Как известно, передача между соседними узлами моделируется в сети Клейнрока с помощью системы массового обслуживания типа M/M/1. Следовательно, время передачи между соседними узлами складывается из ожидания в очереди и времени обслуживания. На рисунке 3.6 проиллюстрирована передача пакета от узла номер 4 к узлу с номером 21. Сначала пакет появляется на узле 4. Процедура маршрутизации определяет следующий на маршруте узел, это узел с номером 17. Сначала пакет попадает в буфер, где должен ждать своей очереди. Так как сеть пуста, то он сразу же попадает на обслуживатель. Время обслуживания пакета распределено по экспоненциальному закону с некоторым средним, которое мы обозначим как m_e. После обслуживания пакет попадает в узел номер 17. Время передачи между узлами составило $t = 0 + m_e = m_e$. Далее процедура маршрутизации решает отправить пакет на узел номер 21. Снова пакет попадает сначала в буфер, а затем на обслуживатель. Пусть среднее время обслуживания также будет m_e. Значит после m_e единиц времени (в среднем) пакет попадет в узел 21, который и является получателем пакета. Итак, задержка пакета на маршруте равна $t = 0 + m_e + 0 + m_e = 2m_e$.

Для произвольной длины пути l задержку можно записать как $t = lm_e$. Так как в рассмотренном примере пакет передается по пустой сети (пустые буферы) и все обслуживатели СМО работают по экспоненциальному закону с одними и теми же параметрами, то задержка каждого пакета представляет собой сумму, состоящую из одинакового числа l независимых случайных величин распределенных по одному и тому же экспоненциальному закону распределения. Как известно [105], [106] величина являющаяся суммой независимых случайных величин, распределенных по одному и тому же экспоненциальному закону, распределена по закону Эрланга. Если хотя бы одно условие будет нарушено, то распределение задержки пакета строго говоря не будет Эрланговым. Например, если при передаче между парой узлов пакеты будут передаваться по маршрутам разной длины, задержка будет суммой переменного числа случайных величин и соответствующее условие будет нарушено.

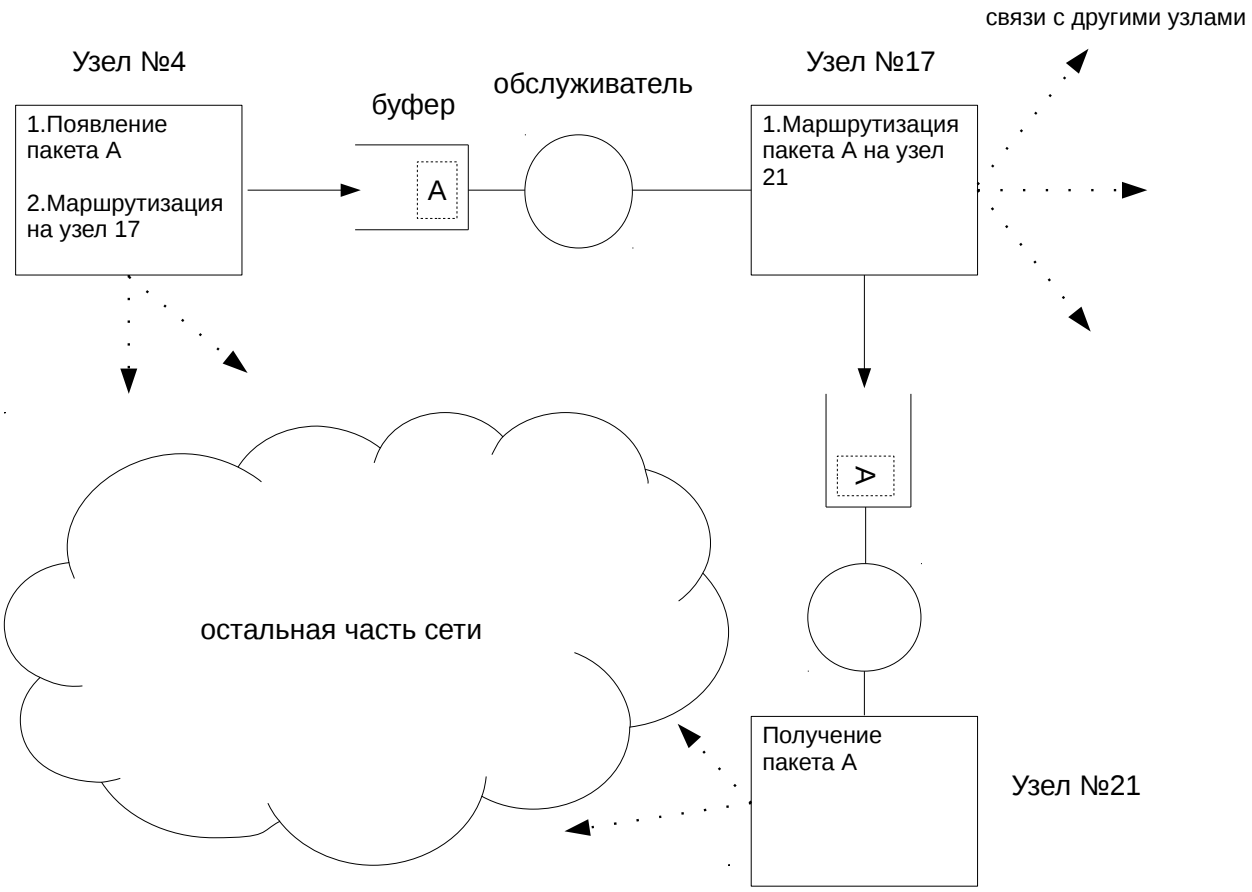


Рисунок 3.6 — Передача одного пакета по пустой сети Клейнрока.

Рассмотрим теперь подробнее передачу по сети Клейнрока, когда буферы непустые. Известно [107], что время пребывания заявки в системе M/M/1 распределено по экспоненциальному закону с средним $\bar{T} = 1/(\mu - \lambda)$. Следовательно, задержка пакета при передаче по каналу распределена по экспоненциальному закону со средним $\bar{t}(\rho)$ из выражения (3.11). Воспользуемся допущением Клейнрока о независимости, позволящим независимо рассматривать каждую СМО. Тогда задержка пакета на всем маршруте является суммой независимых случайных величин распреленных по одному и тому же экспоненциальному закону распределения. Как было замечено ранее, в таком случае задержка пакетов имеет распределение Эрланга. Обратим внимание на то, что утверждение о независимости отдельных СМО в сети является допущением и на самом деле зависимость присутствует. Однако в [101] приводятся условия, при которых эта зависимость может быть уменьшена.

Подытожим сказанное про закон распределения задержки пакетов. С одной стороны, при некоторых условиях задержка действительно имеет распределение Эрланга. С другой стороны, при других условиях задержка имеет иное распреде-

ление. В целом, представляется, что рассмотрение закона Эрланга представляет интерес, т.к. иногда он выполняется, а при небольшом отклонении от необходимых условий возможно станет неплохой аппроксимацией.

Для расчета средней задержки сообщения при использовании транспортного кодирования в сети, где задержка пакета распределена по закону Эрланга, воспользуемся той же методикой вычисления с помощью порядковых статистик, что и в подразделах 3.4, 3.5. Для этого необходимо воспользоваться формулой (3.17)

$$M\left[X\right] = n\binom{n-1}{k-1}\int_0^1 P^{-1}\left(u\right)u^{k-1}(1-u)^{n-k}du,$$

где в качестве u подставить функцию распределения Эрланга

$$F_r(x) = 1 - e^{-\alpha x}\sum_{i=0}^{r-1}\frac{(\alpha x)^i}{i!}, \tag{3.38}$$

где r — порядок Эрланга, а α — параметр экспоненциального распределения. В нашем случае порядок Эрланга равен длине пути, т.е. $r = l$, а $1/\alpha$ – среднее время передачи пакета по каналу, следовательно $\alpha = 1/\bar{t}\left(\rho\right)$. После необходимых преобразований получим

$$M\left[T_{k:n}\right] = n\binom{n-1}{k-1}\int_{-\infty}^{\infty} -t\left(1 - e^{-\alpha t}\sum_{i=0}^{r-1}\frac{(\alpha t)^i}{i!}\right)^{k-1} \tag{3.39}$$
$$\left(e^{-\alpha t}\sum_{i=0}^{r-1}\frac{(\alpha t)^i}{i!}\right)^{n-k} e^{-\alpha t}\sum_{i=0}^{r-1}\left((\alpha t)^i\left[\frac{i}{ti!} - \frac{\alpha}{i!}\right]\right)dt.$$

Перепишем формулу 3.39, подставляя определенные раннее r и α:

$$\bar{T}_{k:n}\left(l,\rho\right) = n\binom{n-1}{k-1}\int_{-\infty}^{\infty} -t\left(1 - e^{-t/\bar{t}(\rho)}\sum_{i=0}^{l-1}\frac{\left(t/\bar{t}\left(\rho\right)\right)^i}{i!}\right)^{k-1} \tag{3.40}$$
$$\left(e^{-t/\bar{t}(\rho)}\sum_{i=0}^{l-1}\frac{\left(t/\bar{t}\left(\rho\right)\right)^i}{i!}\right)^{n-k} e^{-t/\bar{t}(\rho)}\sum_{i=0}^{l-1}\left(\left(\frac{t}{\bar{t}\left(\rho\right)}\right)^i\left[\frac{i}{ti!} - \frac{1}{\bar{t}\left(\rho\right)i!}\right]\right)dt$$

Вычислим значение выигрыша от кодирования как отношение времени передачи сообщения без кодирования ко времени передачи сообщения с кодированием. Так как при кодировании появляются избыточные пакеты, то загрузка сети стано-

вится больше в $1/R$ раз, что необходимо учитывать при вычислении выигрыша от кодирования.

$$T_1/T_2 = \frac{\bar{T}_{k:k}(l,\rho)}{\bar{T}_{k:n}(l,\rho/R)}, \tag{3.41}$$

где R – скорость кода. Для сравнения с экспоненциальным распределением задержки необходимо переписать формулы (3.22) и (3.29) с учетом длины пути:

$$\bar{T}_{k:k}(l,\rho) = l \cdot \bar{t}(\rho) \sum_{i=1}^{k} i^{-1}, \tag{3.42}$$

$$\bar{T}_{k:n}(l,\rho) = l \cdot \bar{t}(\rho) \left(\sum_{i=1}^{n} i^{-1} - \sum_{i=1}^{n-k} i^{-1} \right). \tag{3.43}$$

Тогда и для экспоненциального распределения задержки пакета выигрыш от кодирования можно также вычислить по формуле (3.41).

На рисунках 3.7, 3.8 представлены графики зависимости выигрыша от кодирования при длине пути $l = 3$ (вычисленного по формуле (3.41)) от скорости кода R. Для сравнения представлены кривые для экпоненциального распределения вместе с распределением Эрланга.

Проанализировав кривые, можно сделать следующие выводы:

– При Эрланговой модели задержки выигрыши от кодирования значительно ниже, но все равно присутствуют. Для экспоненциального распределения выигрыш варьировался в диапазоне $T_1/T_2 = 0 \ldots 10$, а для Эрланга – в диапазоне $T_1/T_2 = 1 \ldots 3$.

– Значения оптимальных скоростей кода, при которых выигрыш максимален, отличаются для двух распределений. Для Эрланга оптимальная скорость несколько выше при прочих равных параметрах.

– С ростом загрузки сети ρ зависимости, полученные для обоих распределений, становятся ближе друг к другу. Выигрыш от кодирования ожидаемо снижается.

– С ростом количества информационных пакетов k для обоих распределений наблюдается рост выигрыша от кодирования. При распределении Эрланга скорость роста ниже.

– С ростом длины пути l выигрыш при экспоненциальном распределении не изменяется, при распределении Эрланга уменьшается. Это связано с

особенностями модели сети. При экспоненциальной задержке закон распределения не изменяется с ростом длины пути, другие параметры сети в силу равномерности сетевой нагрузки также не меняются. При распределении Эрланга с ростом длины пути меняется распределение, так как изменяется порядок Эрланга.

– Если бы сетевая нагрузка не была равномерно распределена по всей сети, то, вероятно, в зависимости выигрыша от длины пути при распределении Эрланга не было бы такого явного тренда на снижение.

3.6.2 Нормальное распределение задержки пакетов

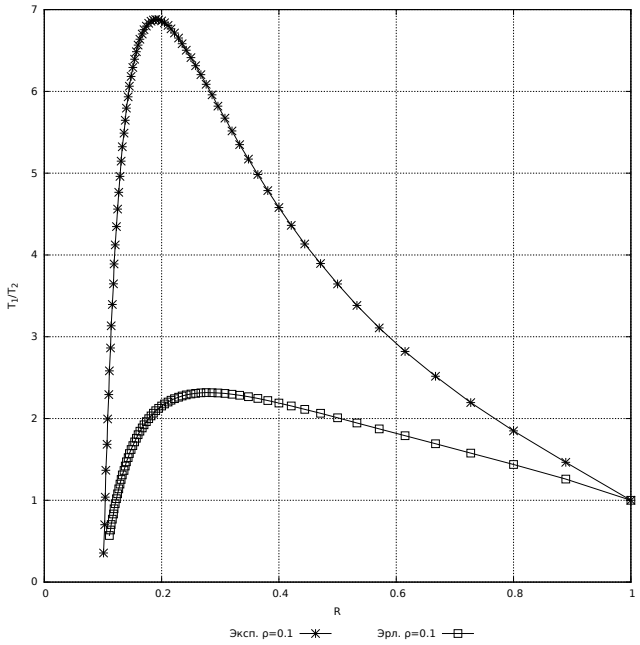
Рассмотрим выигрыш от кодирования при нормальном распределении задержки пакетов. Нормальное распределение вероятностей интересно для рассмотрения по следующим причинам. С ростом порядка в распределении Эрланга (равносильно росту длины пути) это распределение стремится к нормальному. Поэтому зависимости, полученные для нормального распределения, интересны и могут быть использованы для прогноза выигрыша от кодирования на очень больших длинах путей. Также нормальный закон распределения очень популярен и распространен в реальной жизни.

Выполним вычисление средней задержки сообщения при нормальном распределении задержки пакета. Для этого, так же как и в предыдущем подразделе, воспользуемся аппаратом порядковых статистик. Воспользуемся формулой (3.17):

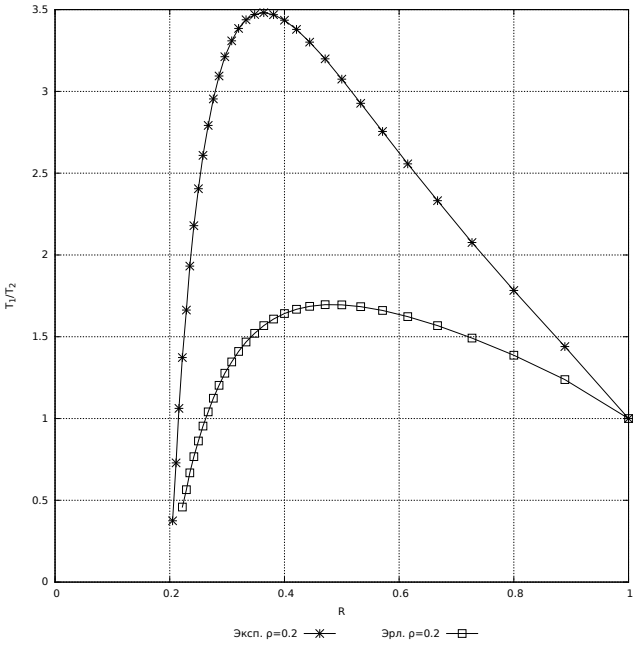
$$\begin{aligned} M\left[X_{k:n}\right] &= \int_{-\infty}^{\infty} x f_k(x) dx = \\ &= n\binom{n-1}{k-1} \int_{-\infty}^{\infty} x[P(x)]^{k-1} \left[1-P(x)\right]^{n-k} dP(x), \end{aligned}$$

где в качестве функции $P(x)$ необходимо подставить функцию нормального распределения:

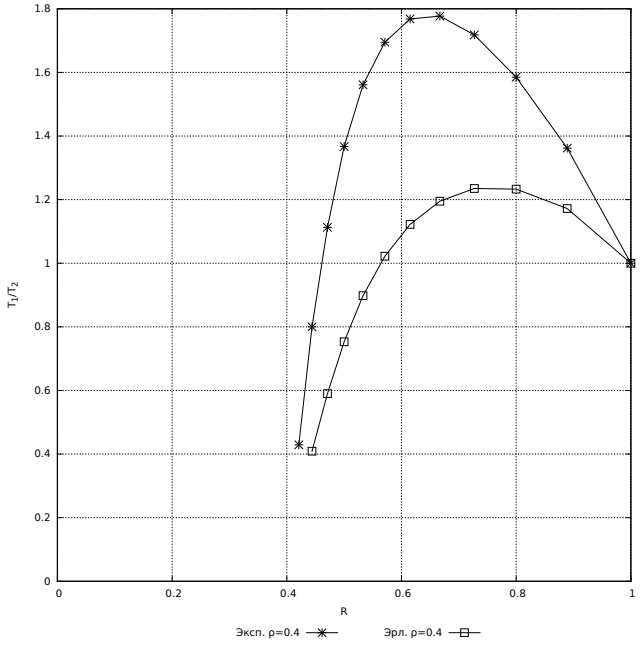
$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx. \tag{3.44}$$



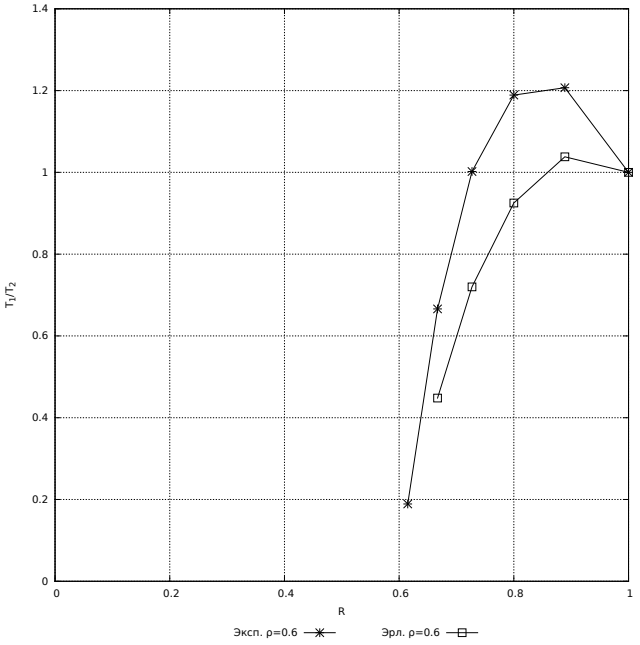
а) ρ=0.1



б) ρ=0.2



в) ρ=0.4

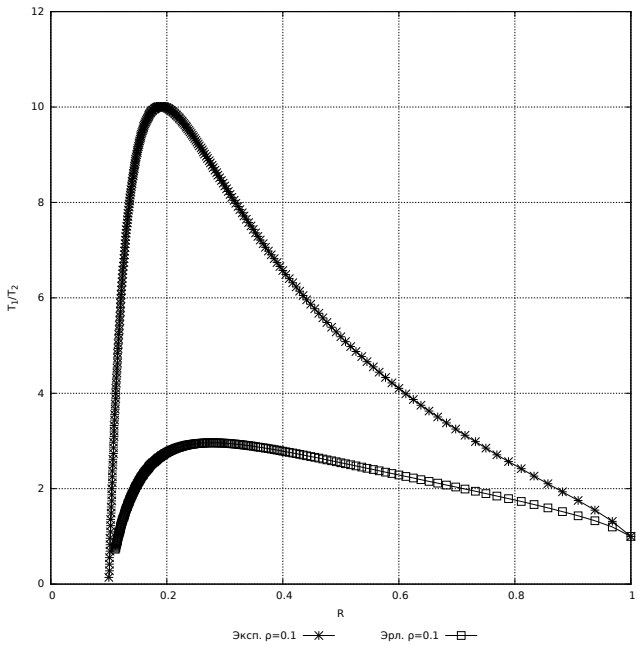


г) ρ=0.6

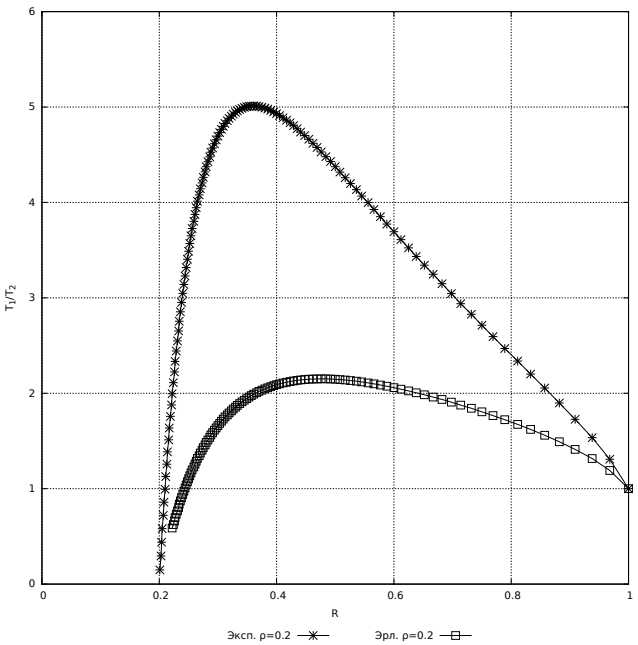
Рисунок 3.7 — Зависимость выигрыша от скорости кода при k=8 и длине пути равной 3

Тогда формула для расчета задержки сообщения примет вид

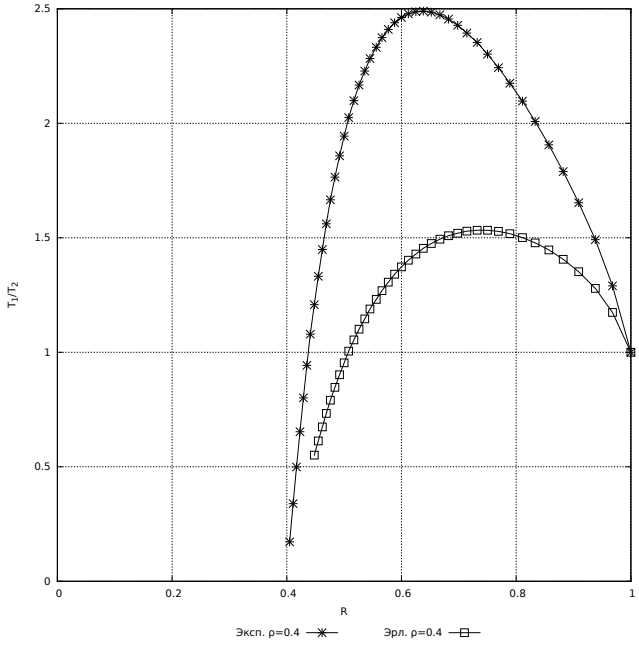
$$\bar{T}_{k:n} = n\binom{n-1}{k-1}\int_{-\infty}^{\infty}\left[x\left(\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx\right)^{k-1}\right.$$
$$\left.\left(1-\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x}e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx\right)^{n-k}\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right]dx. \quad (3.45)$$



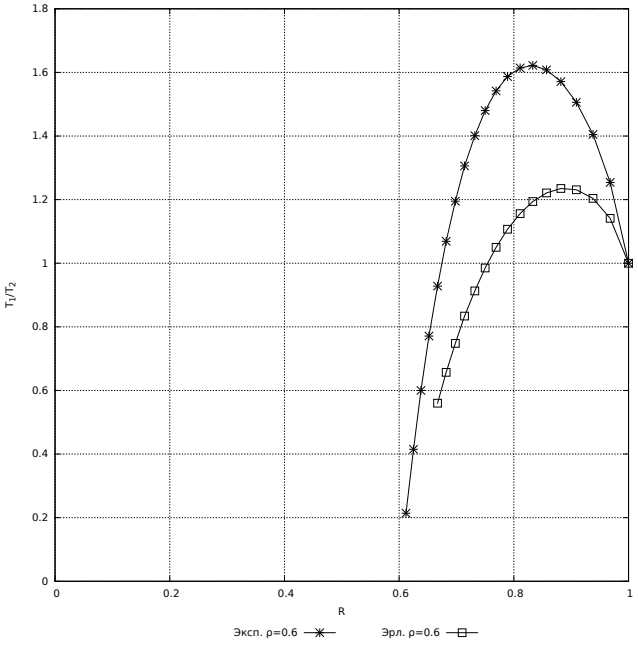
а) ρ=0.1



б) ρ=0.2



в) ρ=0.4

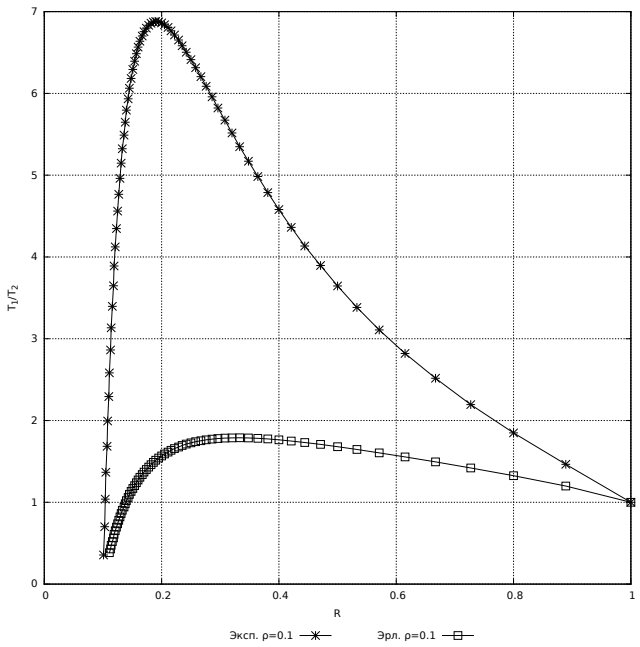


г) ρ=0.6

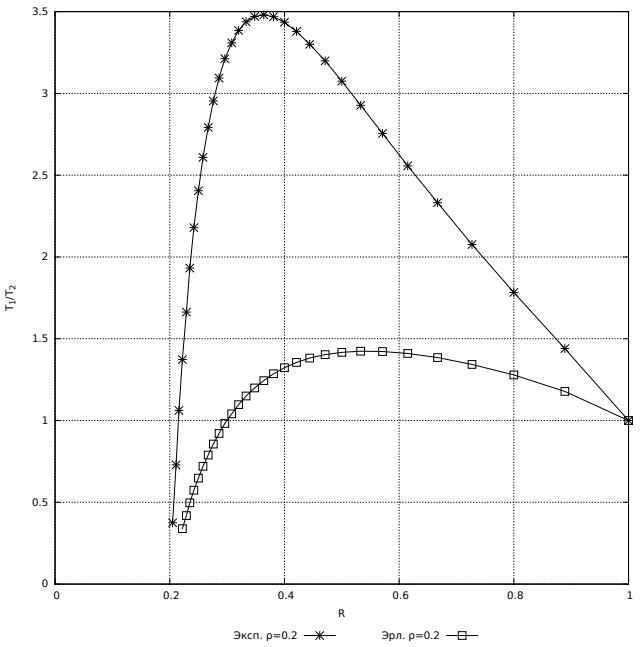
Рисунок 3.8 — Зависимость выигрыша от скорости кода при k=30 и длине пути равной 3

В выражении (3.45) присутствуют параметры нормального распределения μ — математическое ожидание и σ — среднеквадратическое отклонение. В данном случае параметр μ является средней задержкой пакета на всем пути, т.е.

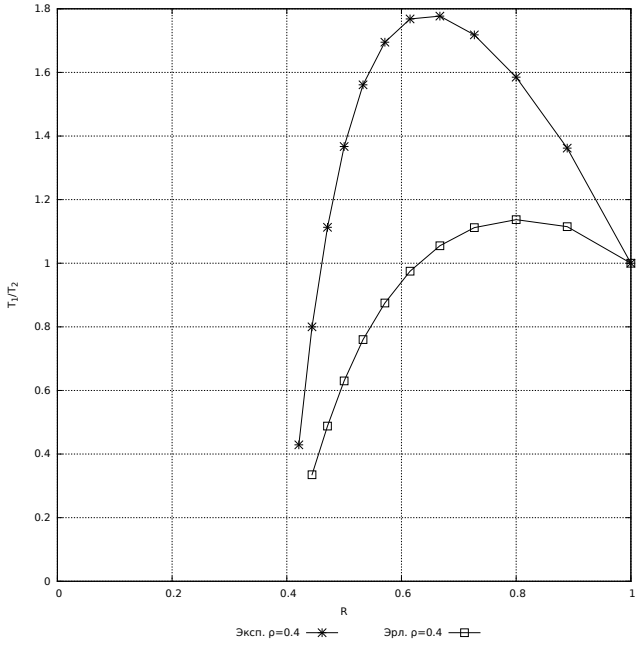
$$\mu = l \cdot \bar{t}\,(\rho)\,.$$



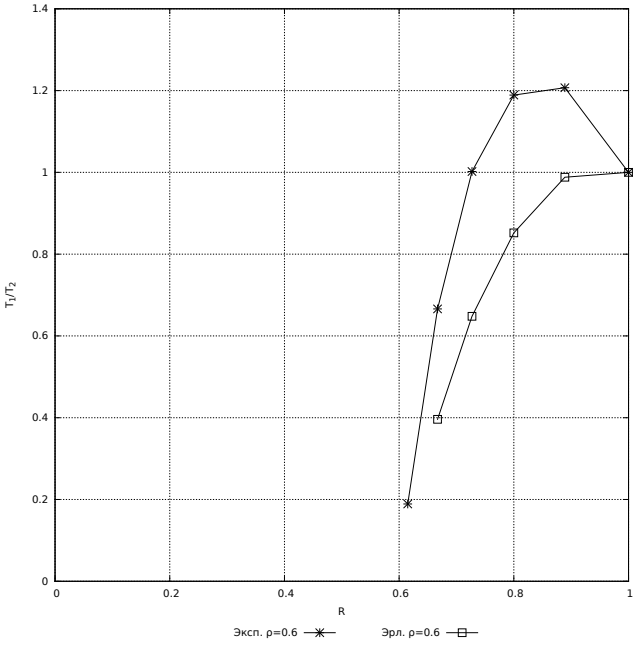
а) ρ=0.1



б) ρ=0.2



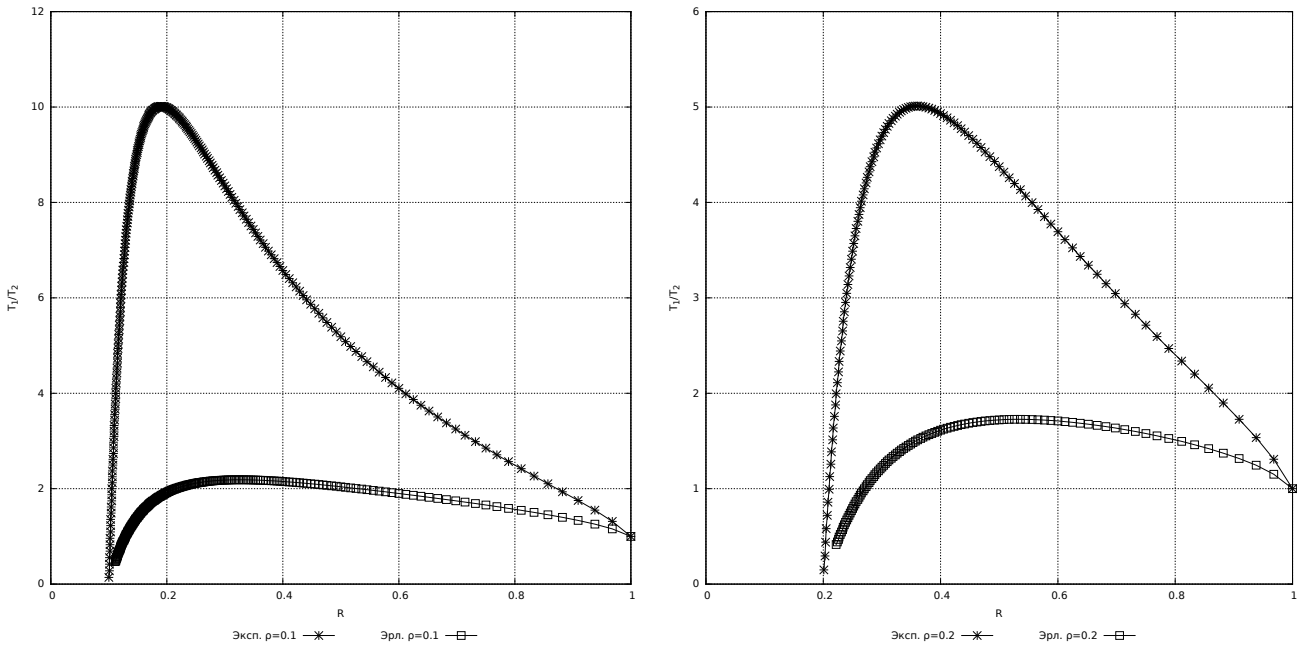
в) ρ=0.4



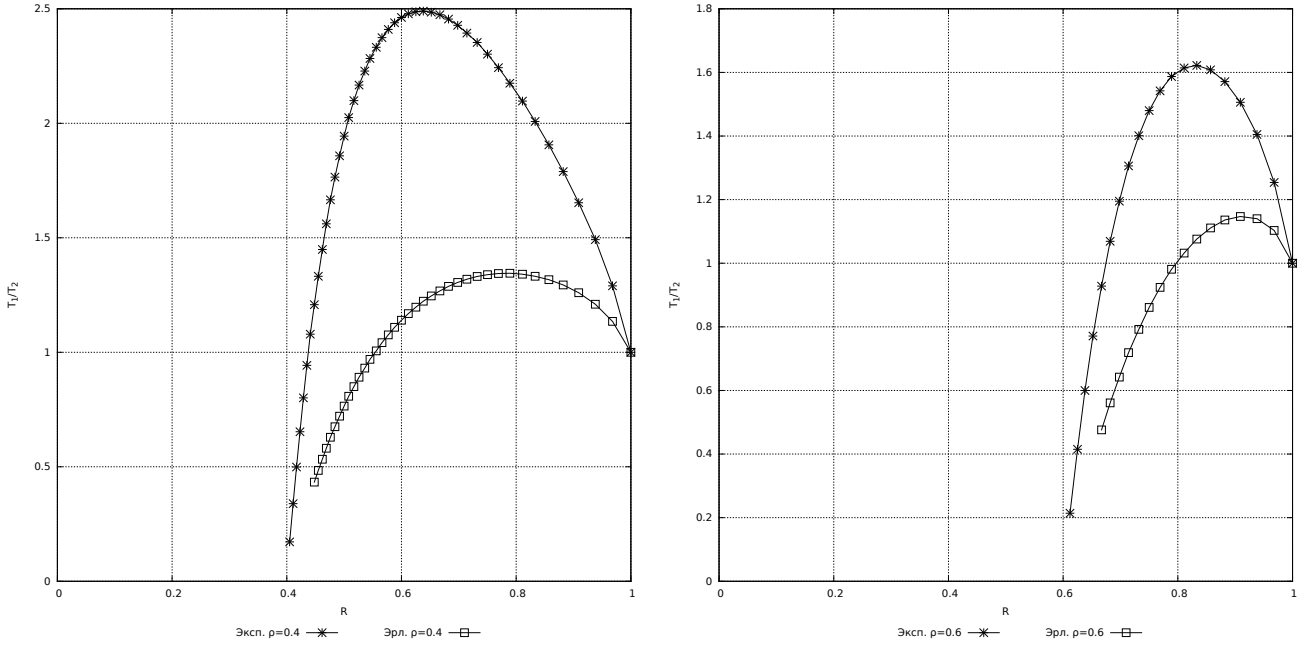
г) ρ=0.6

Рисунок 3.9 — Зависимость выигрыша от скорости кода при k=8 и длине пути равной 5

Необходимо также задать среднеквадратическое отклонение для задержки пакетов. Известно [101], что время пребывания одной заявки в системе массового обслуживания M/M/1 распределено по экспоненциальному закону со средним $\frac{1}{\mu-\lambda}$. Как известно, дисперсия экспоненциального распределения равна квадрату математического ожидания $\frac{1}{(\mu-\lambda)^2}$.



а) ρ=0.1 б) ρ=0.2

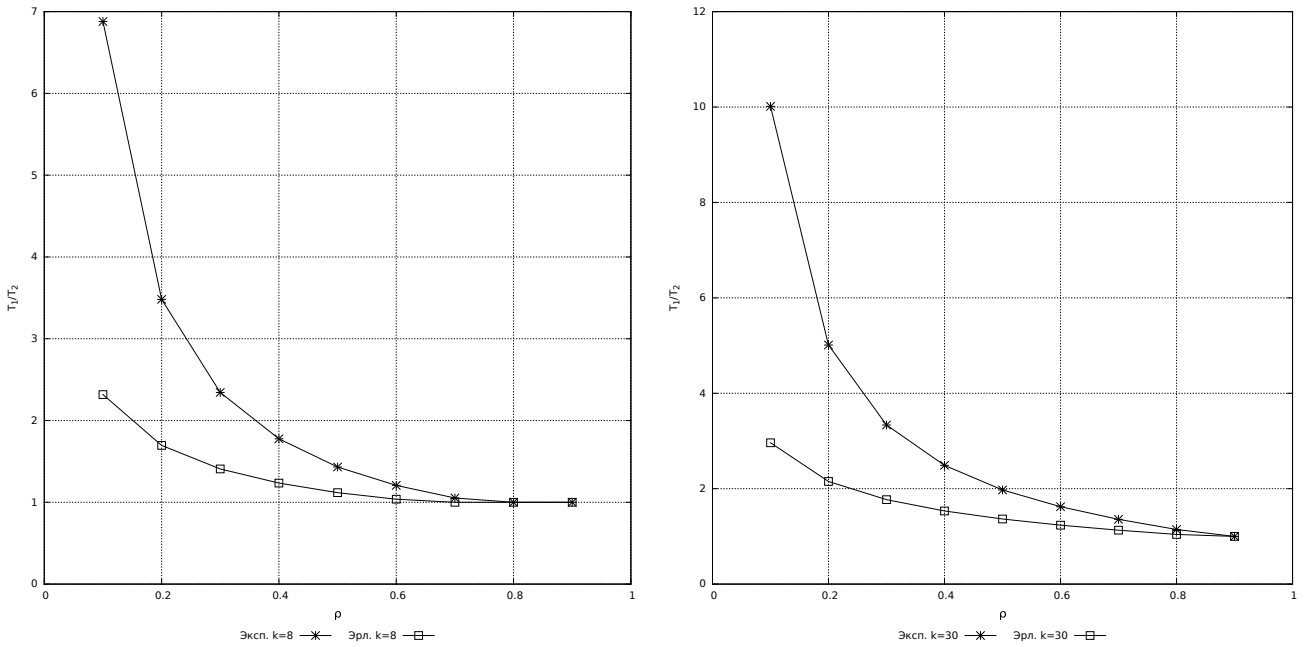


в) ρ=0.4 г) ρ=0.6

Рисунок 3.10 — Зависимость выигрыша от скорости кода при k=30 и длине пути равной 5

В рассматриваемой модели сети дисперсия задержки пакета в канале равна $\frac{1}{\bar{t}(\rho)^2}$. Однако, задержка пакета в сети является суммой задержек в каналах. Известно, что только для независимых случайных величин выполняется следующее свойство дисперсий

$$D[X_1 + X_2 + \ldots + X_i] = D[X_1] + \ldots + D[X_i].$$



а) k=8 б) k=30

Рисунок 3.11 — Зависимость выигрыша от загрузки сети при длине пути равной 3

Воспользуемя допущением Клейнрока о независимости, которое позволяет рассматривать каналы независимо друг от друга. Тогда дисперсия задержки в сети будет равна

$$D\left[\bar{T}_{k:n}\right] = \frac{l}{\bar{t}\left(\rho\right)^2}. \tag{3.46}$$

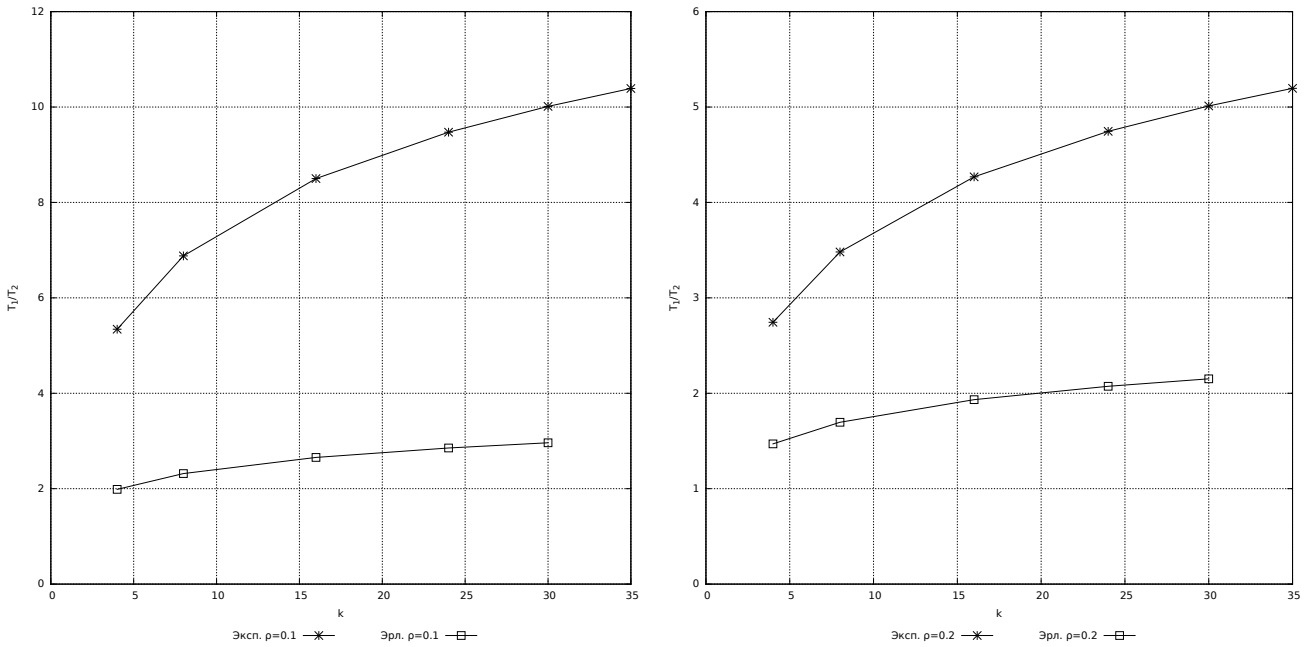
Перепишем выражение (3.45) в зависимости от l и ρ:

$$\bar{T}_{k:n}\left(l,\rho\right) = n\binom{n-1}{k-1}\int_{-\infty}^{\infty}\left[x\left(\frac{\bar{t}\left(\rho\right)}{\sqrt{2\pi l}}\int_{-\infty}^{x}e^{-\frac{(x\bar{t}(\rho)-l)^2}{2l}}dx\right)^{k-1}\right.$$
$$\left.\left(1-\frac{\bar{t}\left(\rho\right)}{\sqrt{2\pi l}}\int_{-\infty}^{x}e^{-\frac{(x\bar{t}(\rho)-l)^2}{2l}}dx\right)^{n-k}\frac{\bar{t}\left(\rho\right)}{\sqrt{2\pi l}}e^{-\frac{(x\bar{t}(\rho)-l)^2}{2l}}\right]dx. \tag{3.47}$$

Выражение (3.47) будем использовать в формуле (3.41) для вычисления выигрыша от кодирования.

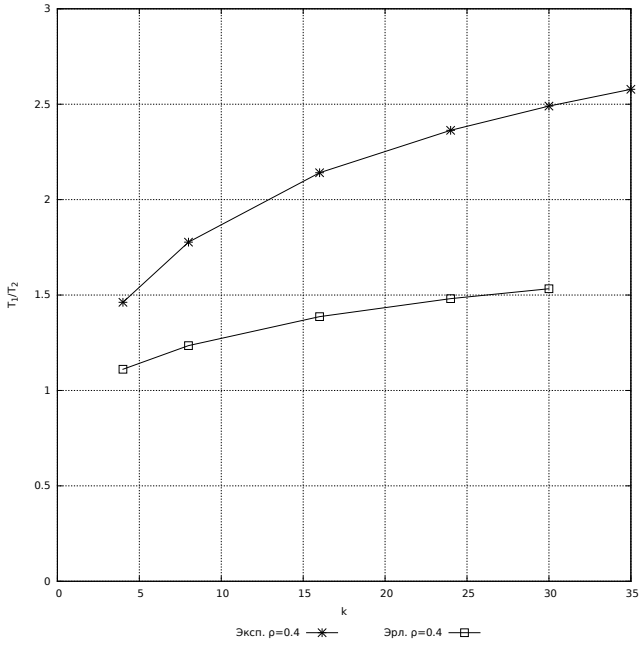
Сравним выигрыш от кодирования при задержке пакета, распределенной по нормальному закону, с выигрышем при экспоненциальном и Эрланговом распределении.

На рисунках 3.14-3.15 представлены графики зависимости выигрыша от кодирования при длине пути $l = 3$, вычисленного по формуле (3.41), от скорости

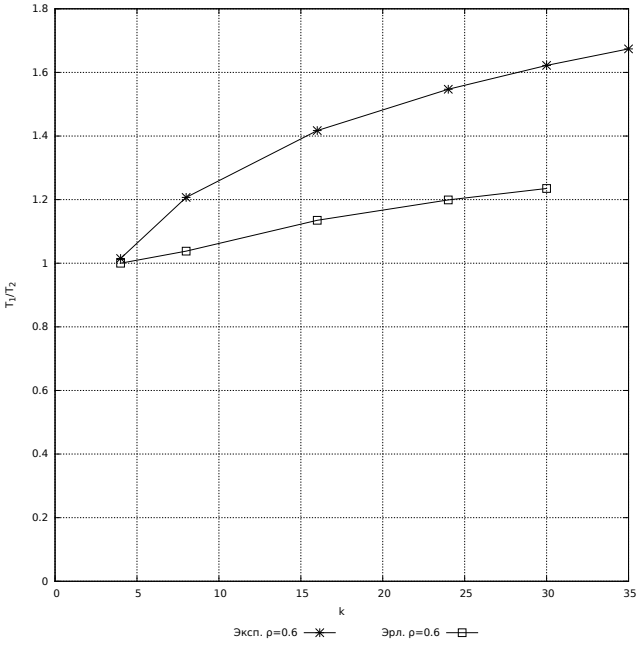


а) ρ=0.1

б) ρ=0.2



в) ρ=0.4



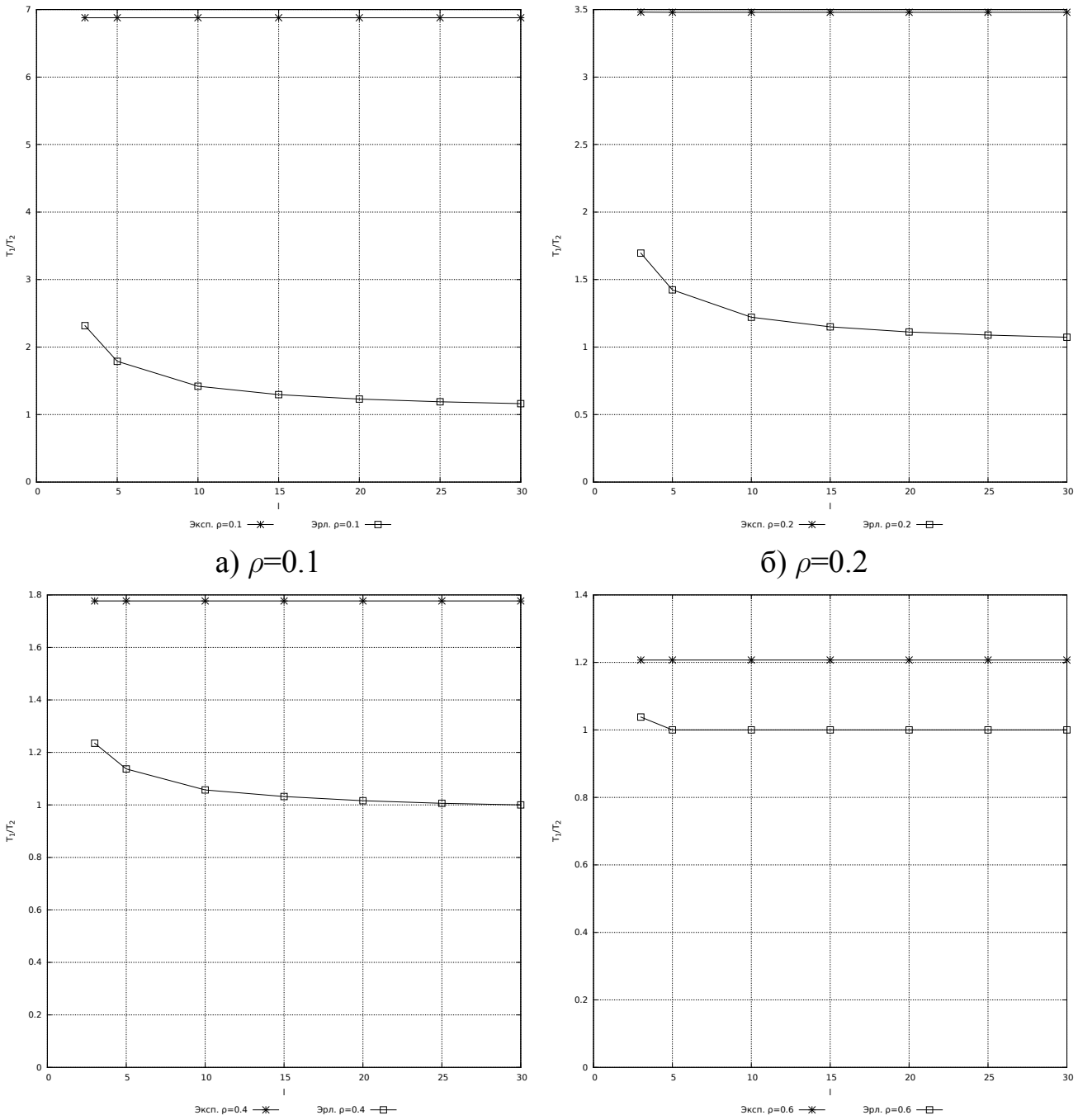
г) ρ=0.6

Рисунок 3.12 — Зависимость выигрыша от k при длине пути равной 3

кода R при использовании нормального, экпоненциального распределения и распределения Эрланга.

Проанализировав кривые, можно сделать следующие выводы.

1. При нормальном распределении задержки пакета, выигрыши стали сильно ниже и для k выигрыш заметно снижается уже при $\rho >= 0.2$, а при $k = 30$ выигрыша не наблюдается вообще.
2. С ростом загрузки сети ρ выигрыш уменьшается.



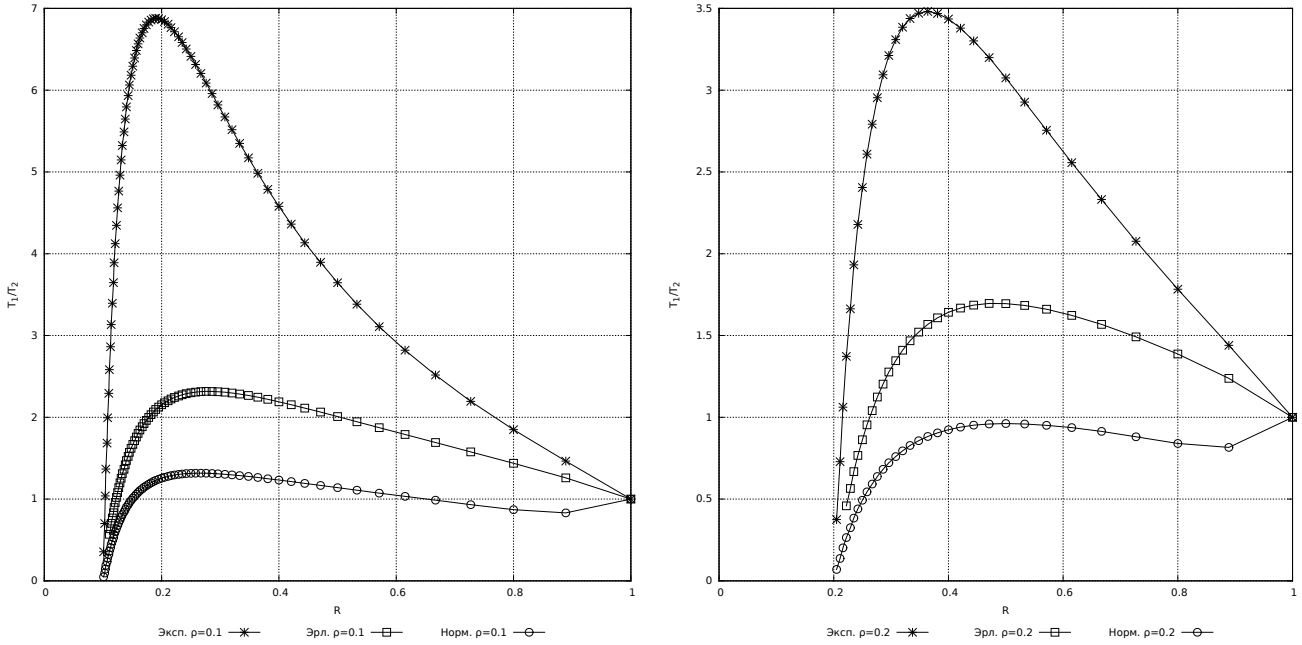
а) ρ=0.1 б) ρ=0.2

в) ρ=0.4 г) ρ=0.6

Рисунок 3.13 — Зависимость выигрыша от длины пути при k=8

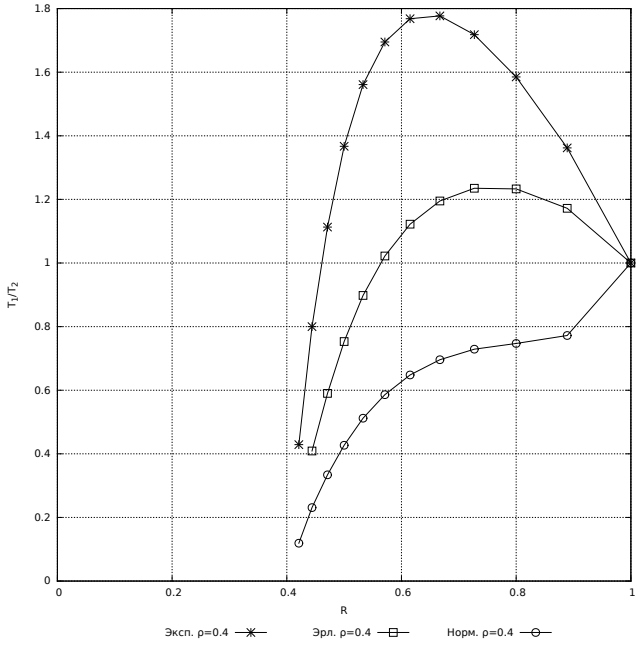
3. С ростом k при нормальном распределении выигрыш уменьшается вплоть до полного отсутствия, в то время как другие распределения демонстрируют рост выигрыша.
4. С ростом длины пути l, как и ожидалось, выигрыши при распределении Эрланга и нормальном становятся ближе и в какой-то момент становятся одинаковыми. Это согласуется с тем, что с ростом порядка распределение Эрланга стремится к нормальному.

5. Учитывая выводы из пунктов 3 и 4, можно предположить, что при очень больших длинах путей при Эрланговом распределении выигрыш начнет уменьшаться и в этом случае следует уменьшить количество информационных пакетов k, чтобы его увеличить.

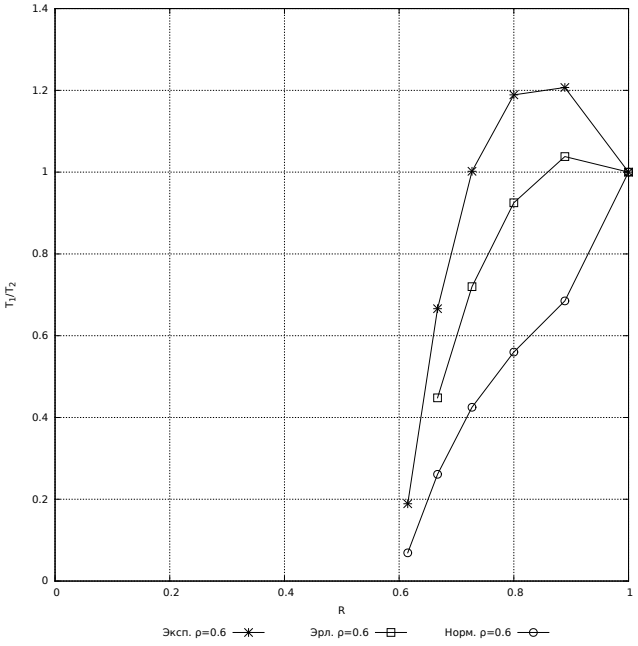


а) ρ=0.1

б) ρ=0.2

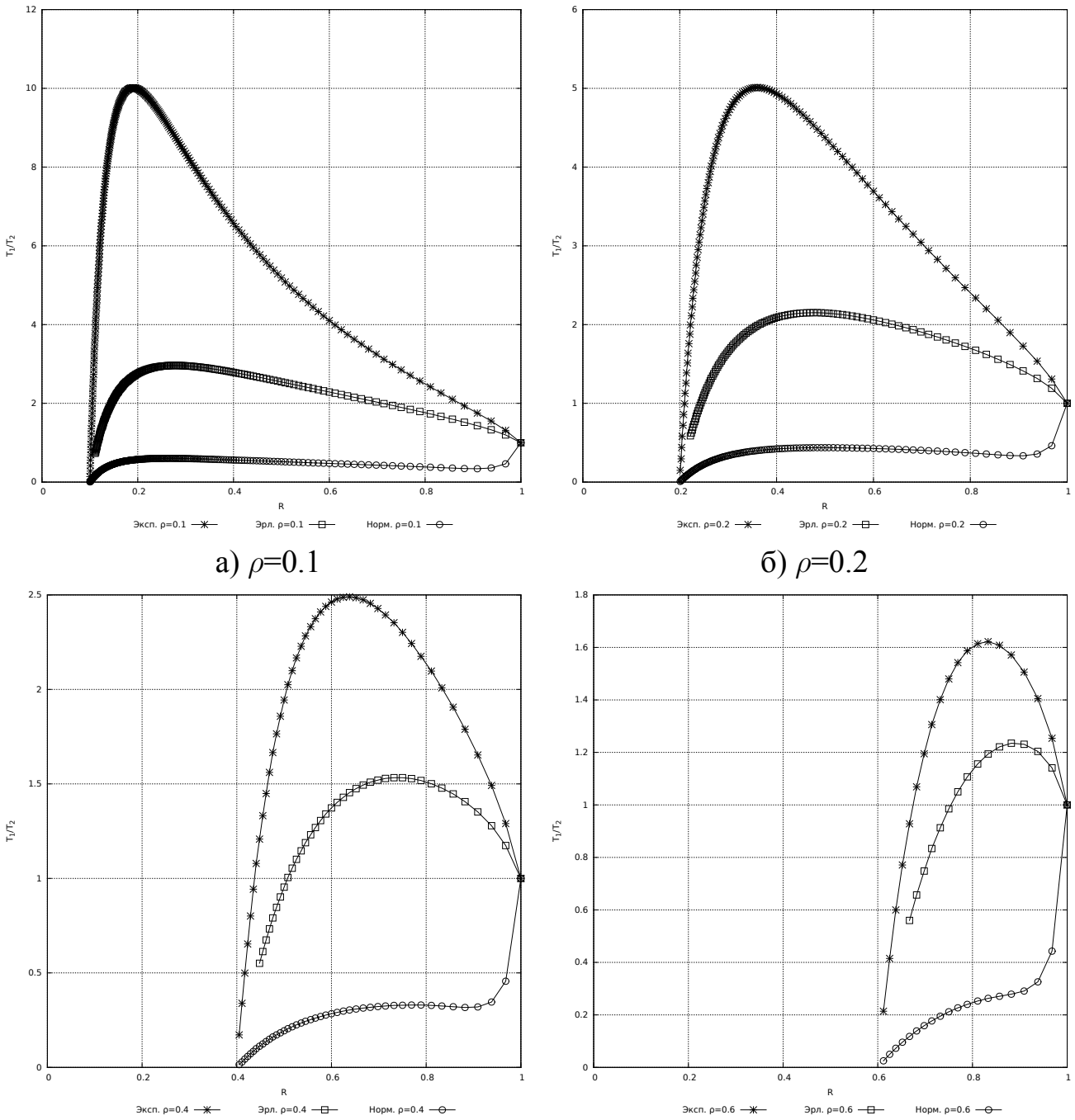


в) ρ=0.4



г) ρ=0.6

Рисунок 3.14 — Зависимость выигрыша от скорости кода при k=8 и длине пути, равной 3



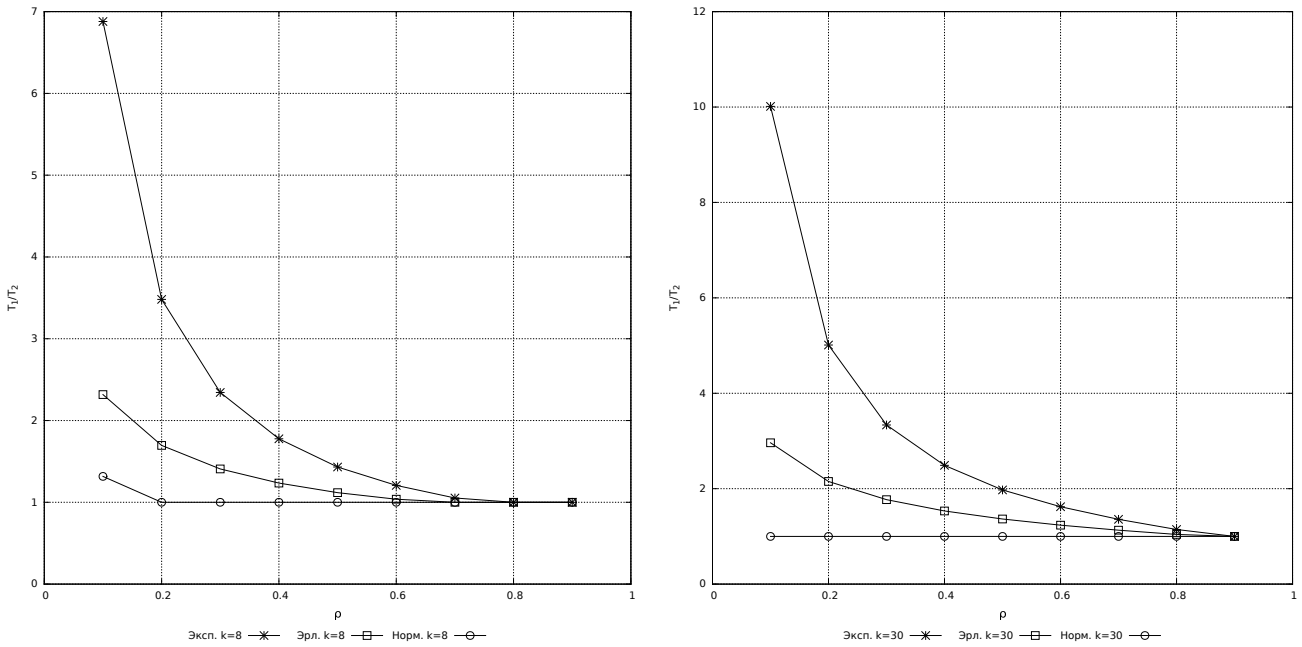
а) ρ=0.1 б) ρ=0.2

в) ρ=0.4 г) ρ=0.6

Рисунок 3.15 — Зависимость выигрыша от скорости кода при k=30 и длине пути, равной 3

3.7 Моделирование сетей с коммутацией пакетов

Для дальнейшего исследования выигрыша от транспортного кодирования будем использовать имитационную модель сети Клейнрока. С ее помощью будем переходить к более общим моделям сети по сравнению с моделями, рассмотрен-



а) k=8 б) k=30

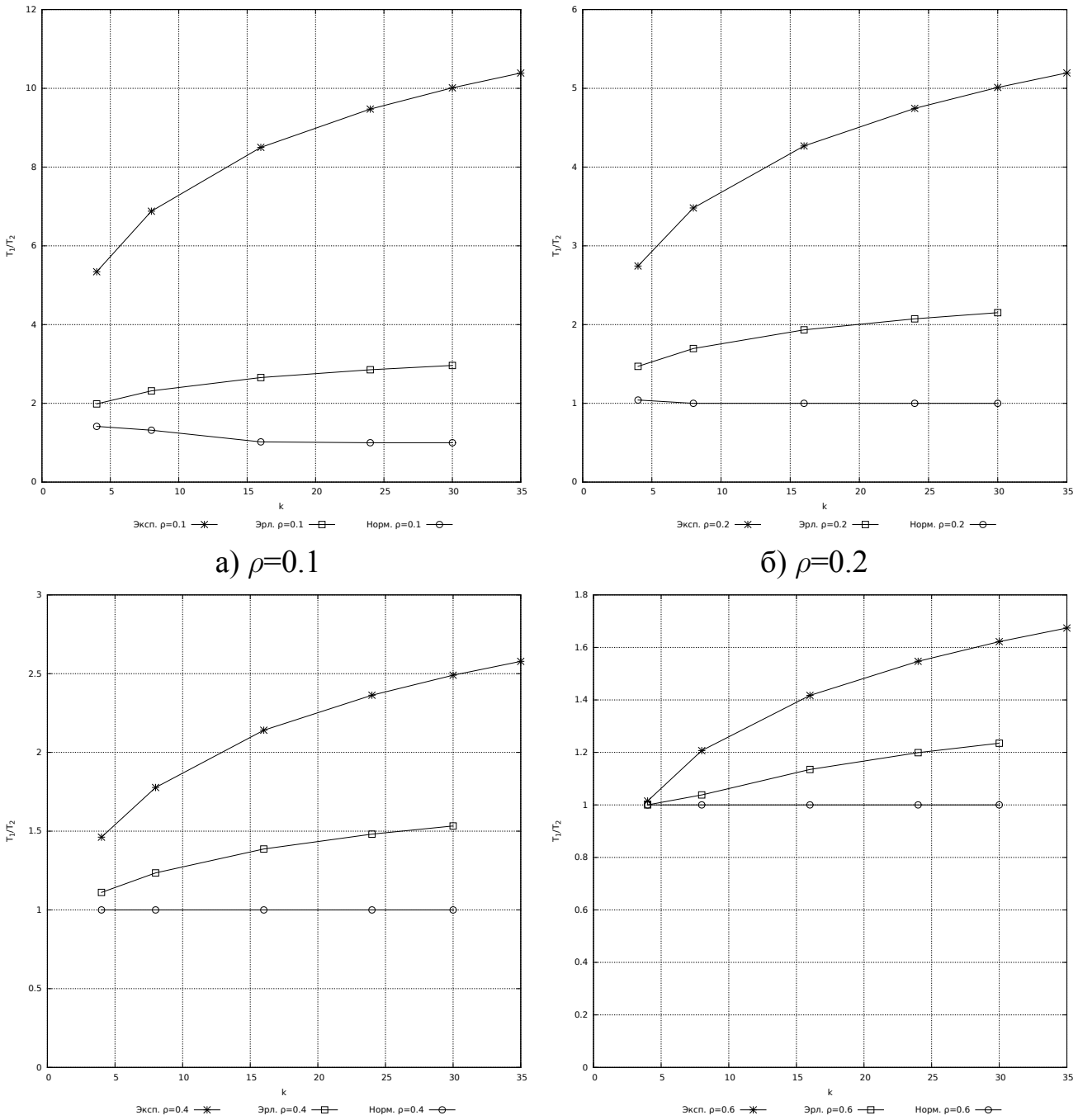
Рисунок 3.16 — Зависимость выигрыша от загрузки сети при длине пути, равной 3

ными в предыдущем разделе. Так как модель сети постепенно усложняется, то вычисление задержки сообщений становится сложной задачей. В данном разделе будет описана такая система моделирования.

На основе имитационной модели в данном подразделе будет исследован выигрыш от кодирования при условиях, максимально близким к рассмотренным в прошлом разделе. Результаты моделирования будут сравнены с полученными ранее. Различия в моделях главным образом состоят в том, что распределение задержки будет получено с помощью имитационного моделирования и должно будет соответствовать таковому в сети Клейнрока.

3.7.1 Описание имитационной модели сети Клейнрока

Рассматриваемая система моделирования работает по принципу дискретно-событийного моделирования. В модели есть такие сущности как узел и канал. На узле создаются многопакетные сообщения с заданной интенсивностью λ. Интервалы между появлениями сообщений распределены по экспоненциальному закону со средним $1/\lambda$. Все пакеты сообщения вбрасываются в сеть одновременно.

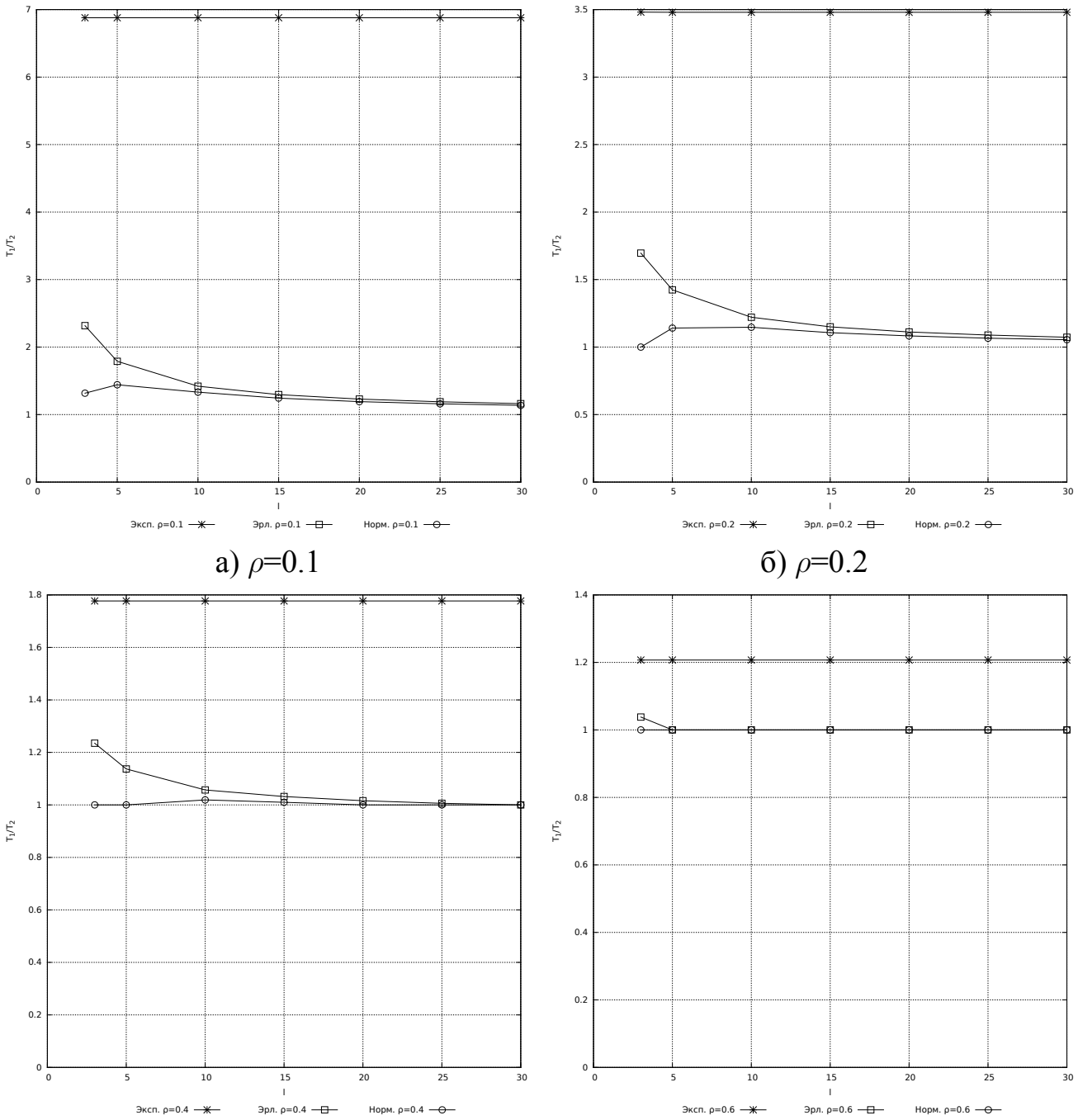


а) ρ=0.1 б) ρ=0.2

в) ρ=0.4 г) ρ=0.6

Рисунок 3.17 — Зависимость выигрыша от k при длине пути, равной 3

На узле принимается решение, по какому каналу посылать пакет дальше, если он еще не достиг узла-получателя. Также на узлах принимается решение о том, что сообщение полностью принято, т.е. получено заданное количество пакетов. Каналы передают пакеты, из которых состоят сообщения, от узла к узлу. Они моделируются с помощью СМО типа M/M/1. Буфер имеет неограниченную длину. Интенсивность обслуживания заявок равна μC, где C – емкость канала, а $1/\mu$ –



а) ρ=0.1 б) ρ=0.2

в) ρ=0.4 г) ρ=0.6

Рисунок 3.18 — Зависимость выигрыша от длины пути при k=8

средняя длина пакета. Для проведения моделирования необходимо задать следующие параметры:

– Топология сети, заданная матрицей смежности.
– Матрица интенсивностей $\Lambda = [\lambda_{i,j}]$, где в i-й строке и j-м столбце стоит значение интенсивности потока сообщений от узла i к узлу j.
– Матрица емкостей каналов $C = [c_{i,j}]$, где в i-й строке и j-м столбце стоит значение емкости канала, соединяющего узлы i и j.

– Количество информационных пакетов k в сообщении.
– Скорость кода для транспортного кодирования R.
– Средняя длина пакета $1/\mu$.
– Ограничение времени моделирования в условных единицах времени.
– Ограничение времени моделирования в количестве принятых сообщений в сети.

Рассмотрим процедуру маршрутизации. Для моделирования эффекта прихода пакетов одного и того же сообщения в разном порядке используется многопутевая маршрутизация. На каждом узле имеется таблица маршрутизации, в которой для каждого возможного узла-получателя задан следующий по маршруту узел и указана длина пути по данному маршруту. Запоминается несколько маршрутов. Для каждого пришедшего на узел пакета по таблице маршутизации выбирается следующий узел и пакет отправляется этому узлу. Выбор конкретного маршрута из нескольких происходит по очереди. Таблицы маршрутизации вычисляются перед выполнением моделирования. Для этого методом обхода в ширину выполняется поиск маршрутов. Первыми будут найдены все кратчайшие маршруты. Процедура поиска останавливается, когда выполнено одно из двух условий:

– достигнута максимальная заданная длина маршрутов,
– достигнуто максимальное заданное количество маршрутов.

3.7.2 Выигрыш от кодирования в простейшей сети

Рассмотрим сеть, в которой все потоки сообщений распределены равномерно по всем каналам. Будем считать, что увеличение нагрузки на сеть при использовании кодирования происходит пропорционально во всей сети. Пусть емкости всех каналов равны. Будем считать, что между источником и получателем имеется столько маршрутов, сколько имеется пакетов. В таком случае, нет необходимости моделировать всю сеть целиком. Достаточно смоделировать только одну пару источник-получатель. Остальные пары источник-получатель будут идентичны. Это упрощение позволяет не рассматривать конкретную топологию и вопрос маршрутизации. На рисунке 3.19 эта модель проиллюстрирована в виде сети

СМО. Фактически моделируются только маршруты от узла источника сообщений до узла получателя.

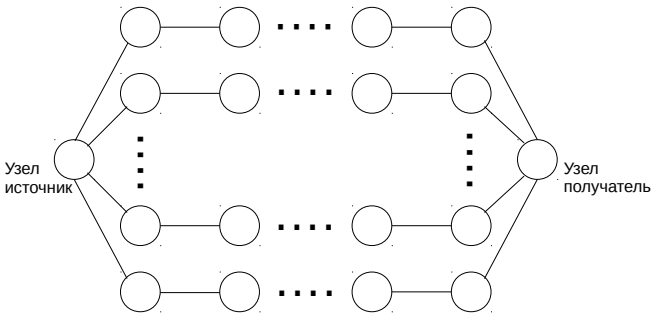


Рисунок 3.19 — Модель сети с параллельными каналами.

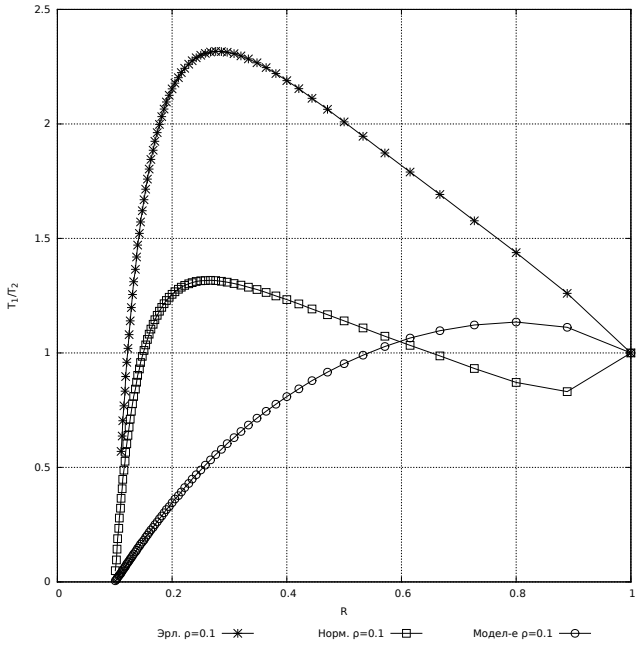
Описанная модель сети максимально приближена к рассматриваемым в предыдущих разделах. Единственное отличие заключается в законе распределения задержки пакетов. Здесь он точно такой, как в сети Клейнрока, а в рассмотренных ранее использовались экспоненциальное, Эрланга и нормальное распределения.

Сравним результаты имитационного моделирования с расчетами, полученными ранее. На рисунках 3.20, 3.21 представлены графики зависимости выигрыша от кодирования при длине пути $l = 3$, вычисленного по формуле (3.41), от скорости кода R при использовании нормального, экпоненциального распределения и распределения Эрланга.

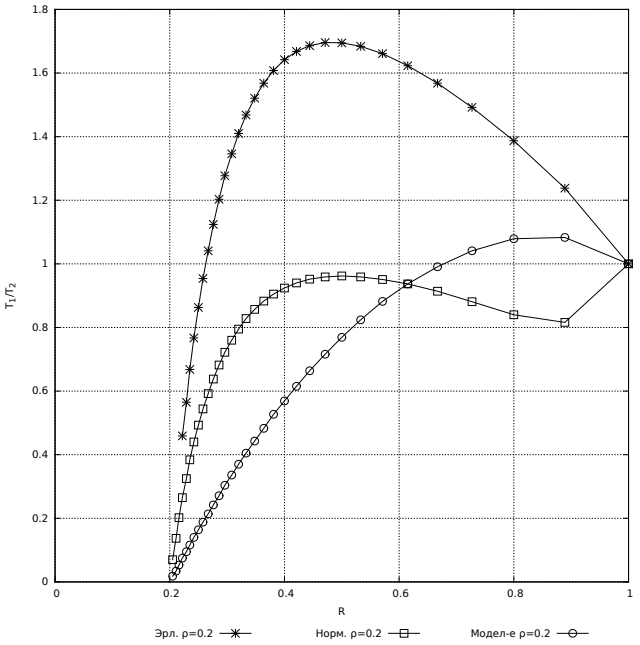
Проанализировав кривые можно сделать следующие выводы.

– Оптимальная скорость кодирования, при которой выигрыш наибольший, отличается от рассчитанной по распределениям Эрланга и нормальному. Для имитационной модели она выше, чем при распределении Эрланга.
– Выигрыш от кодирования присутствует, но стал меньше, чем рассчитанный при распределении Эрланга. Однако он в большинстве случаев выше, чем при нормальном распределении. При рассмотренных $k = 8$,$k = 30$ выигрыш наблюдается при всех рассмотренных значениях загрузки сети $\rho = 0.1$,0.2,0.4,0.6.
– Кривая выигрыша от кодирования в зависимости от загрузки сети для имитационной модели проходит между соответствующими кривыми для нормального распределения и распределения Эрланга.
– Как и для распределений экспоненциального и Эрланга, выигрыш растет с ростом k.

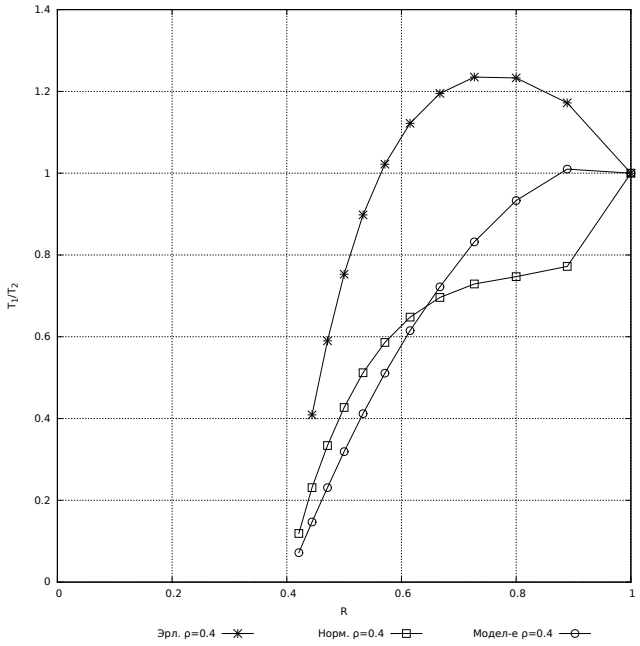
– С ростом длины пути l выигрыш от кодирования уменьшается вплоть до его полного отсутствия.



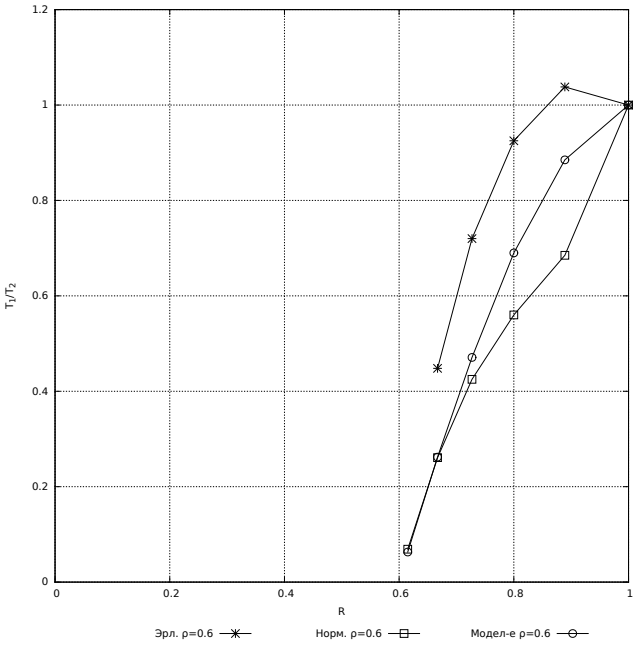
а) ρ=0.1



б) ρ=0.2

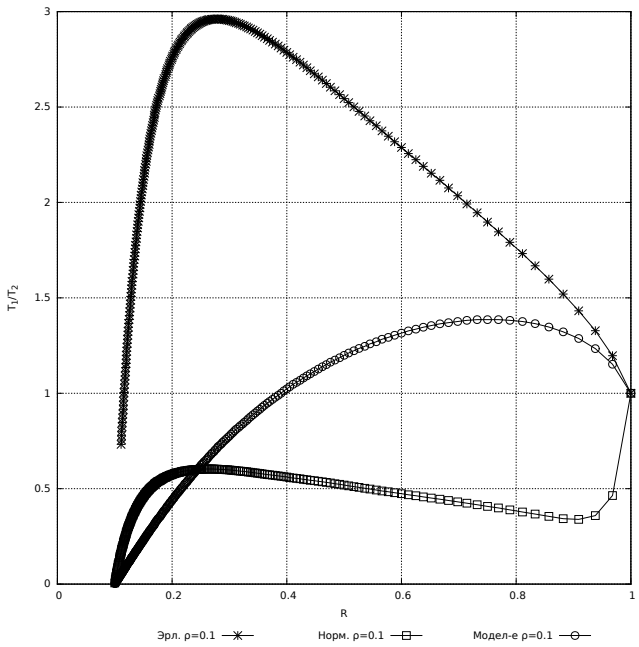


в) ρ=0.4

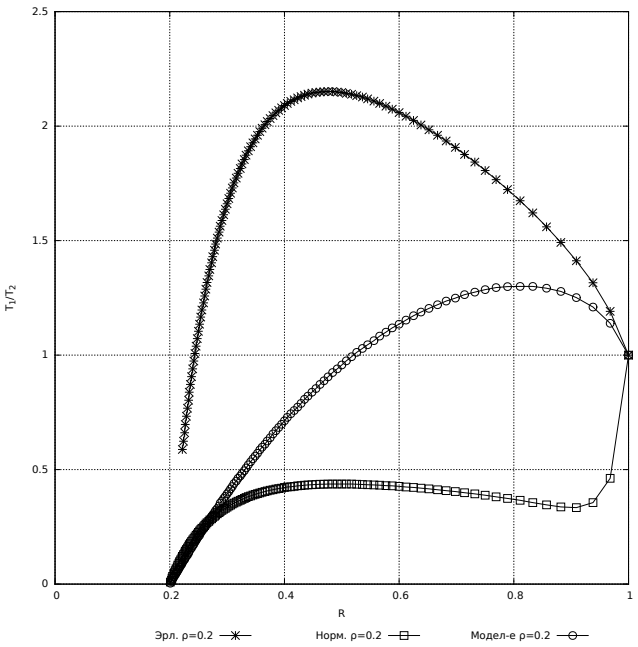


г) ρ=0.6

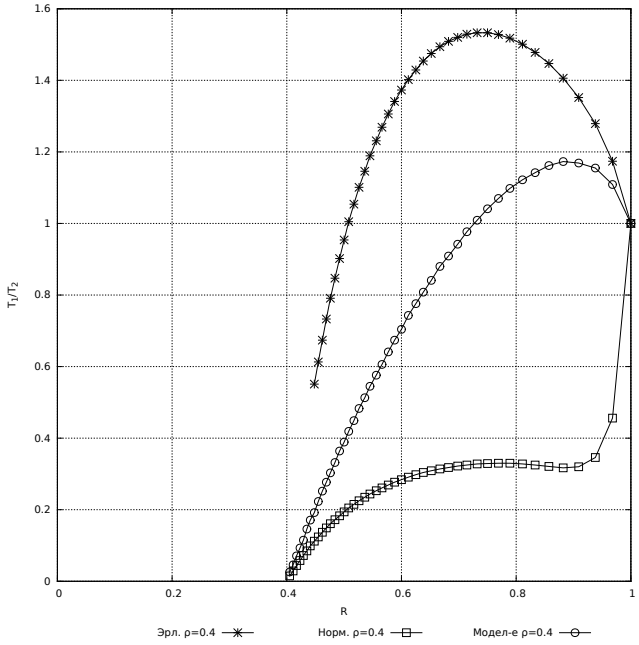
Рисунок 3.20 — Зависимость выигрыша от скорости кода при k=8 и длине пути, равной 3



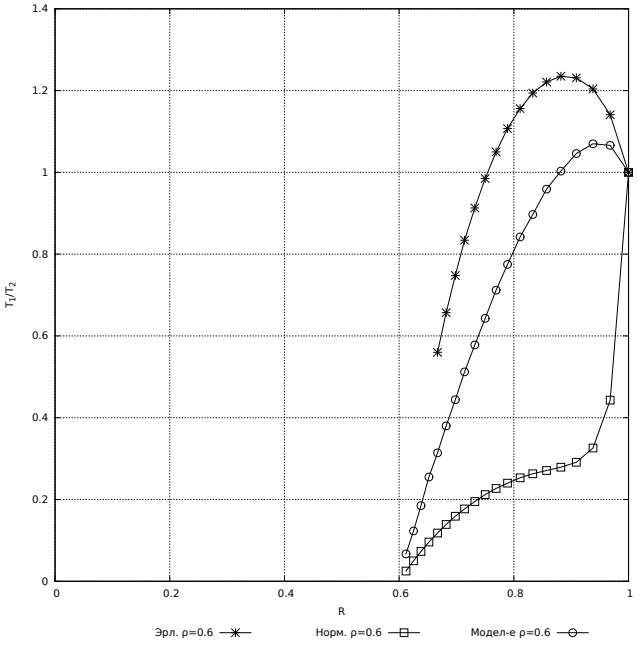
а) ρ=0.1



б) ρ=0.2



в) ρ=0.4

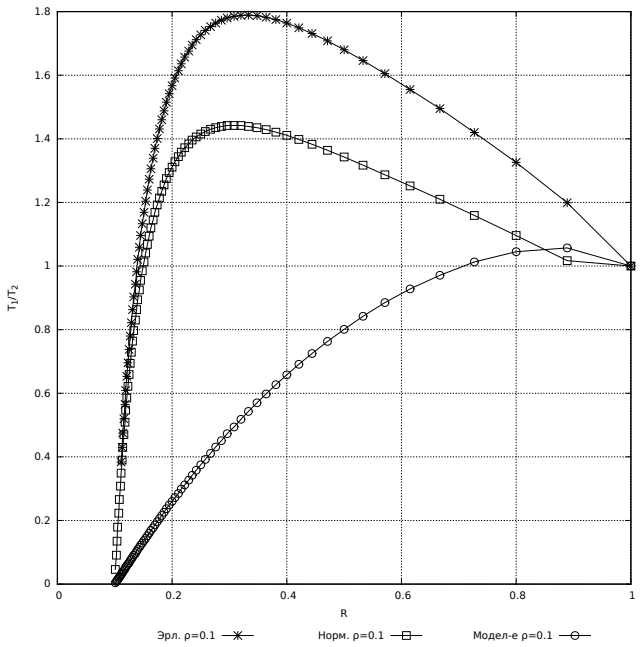


г) ρ=0.6

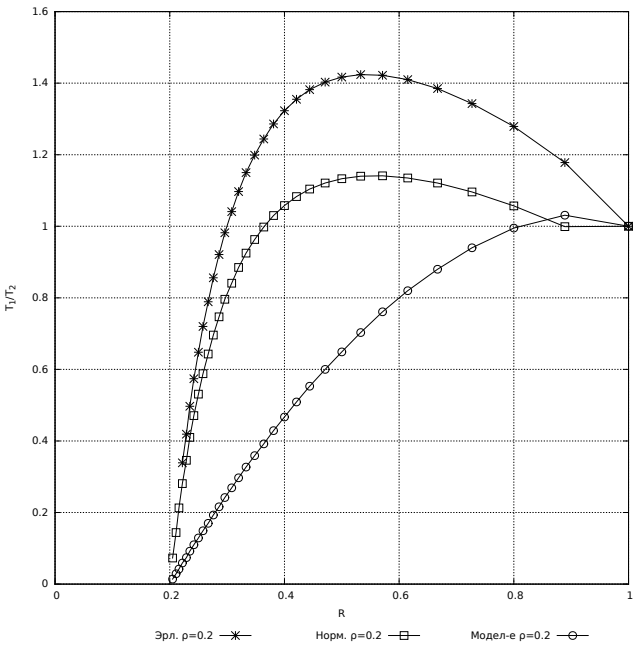
Рисунок 3.21 — Зависимость выигрыша от скорости кода при k=30 и длине пути, равной 3

3.8 Исследование задержки в сети с топологией типа решетка

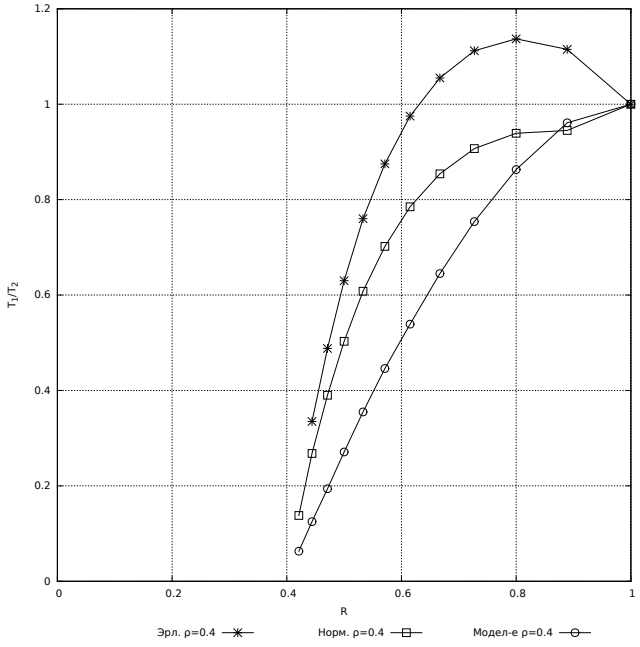
Перейдем к рассмотрению сетей с конкретной топологией. Введение топологии влечет за собой необходимость решать такие задачи как распределение потоков по сети (маршрутизация), задание емкостей каналов. Эти вопросы важны и



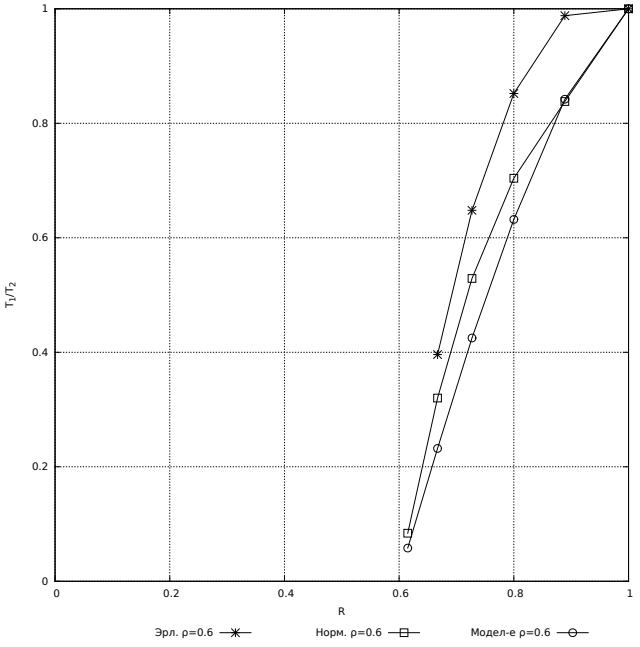
а) ρ=0.1



б) ρ=0.2



в) ρ=0.4

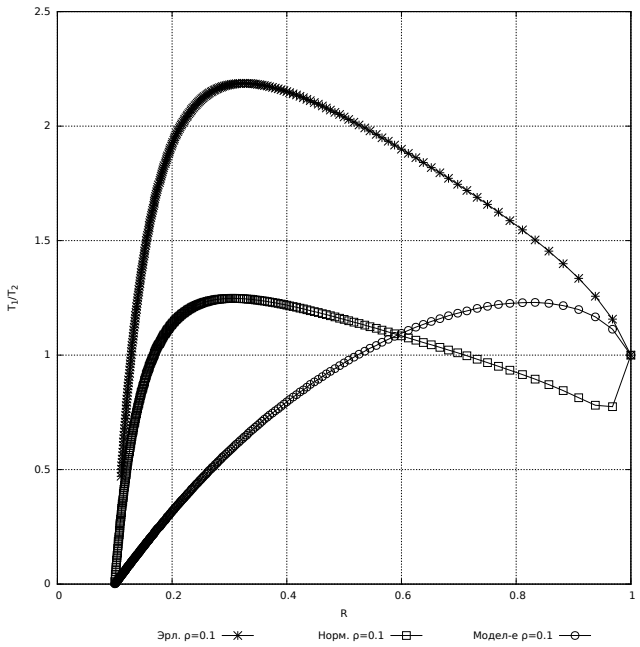


г) ρ=0.6

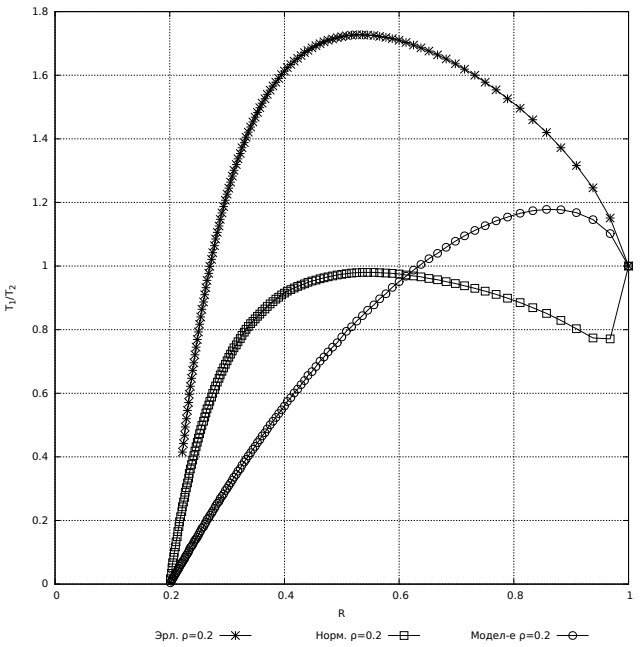
Рисунок 3.22 — Зависимость выигрыша от скорости кода при k=8 и длине пути, равной 5

требуют отдельного рассмотрения и изучения. Модель в данном подразделе выбирается таким образом, чтобы по возможности упростить решение этих вопросов.

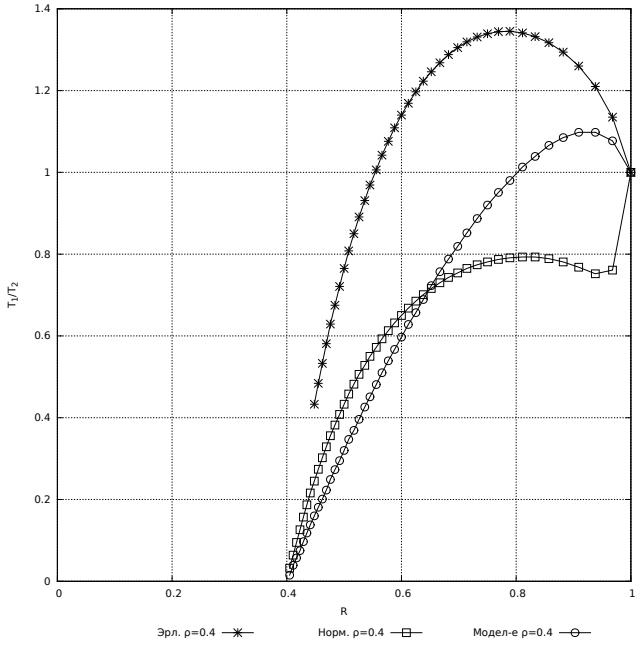
Существует множество различных топологий. В данном разделе мы будем рассматривать топологию типа решетка (рисунок 3.27). Выбор данной топологии обусловлен следующими причинам:



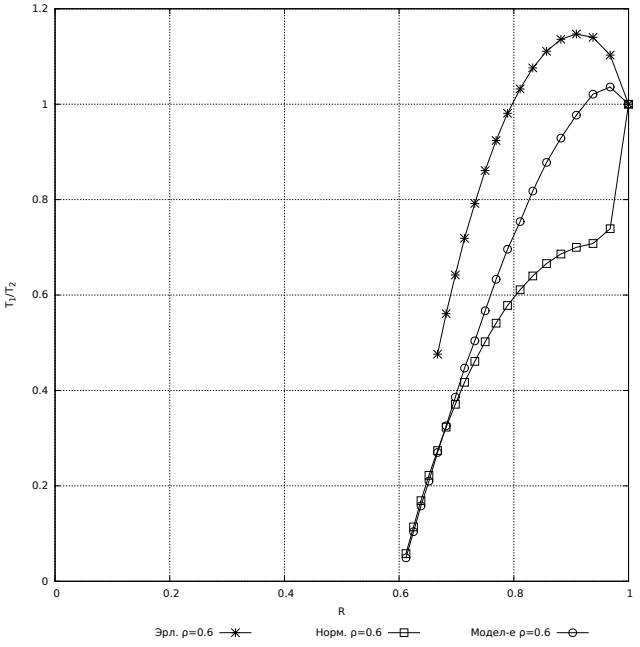
а) ρ=0.1



б) ρ=0.2



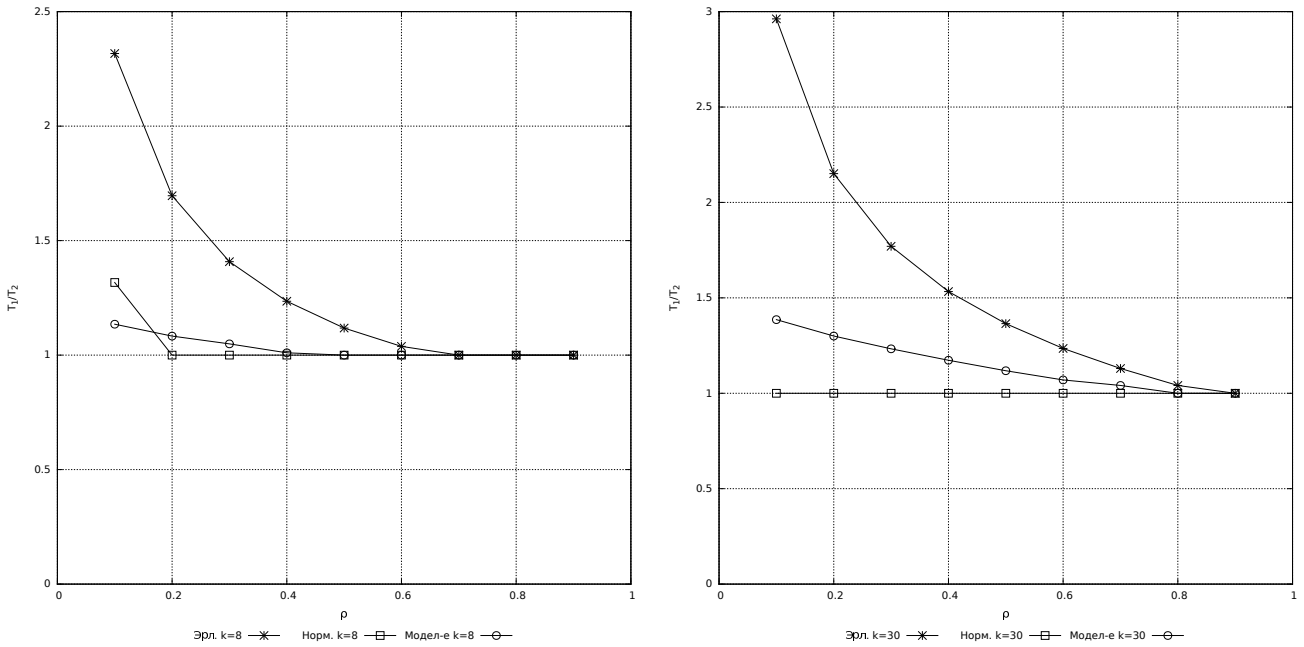
в) ρ=0.4



г) ρ=0.6

Рисунок 3.23 — Зависимость выигрыша от скорости кода при k=30 и длине пути, равной 5

– решетка имеет регулярную структуру, что несколько упрощает рассмотрение,
– между несоседними узлами существует более одного кратчайшего маршрута, что позволяет моделировать эффект прихода пакетов одного сообщения не по порядку.

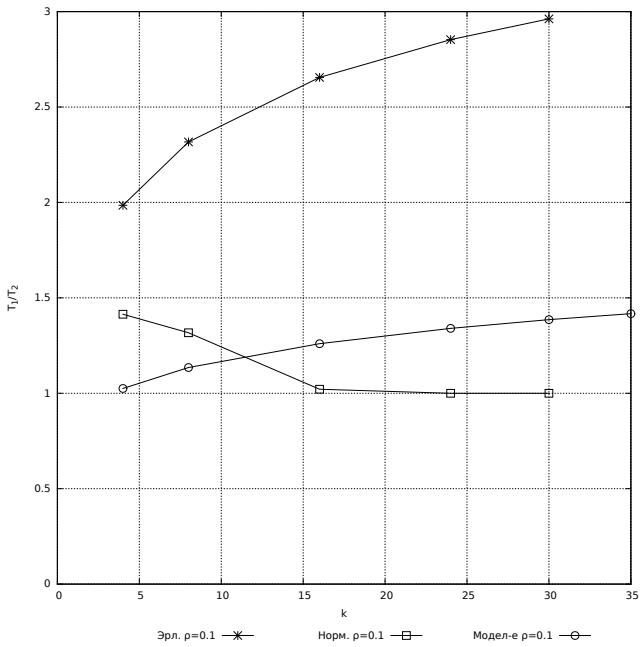


а) k=8 б) k=30

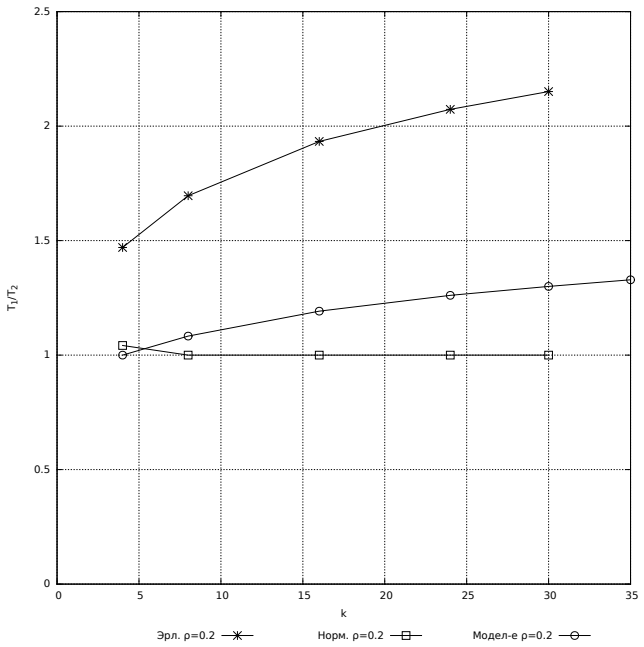
Рисунок 3.24 — Зависимость выигрыша от загрузки сети при длине пути, равной 3

На распределение потоков сообщений по сети влияет процедура маршрутизации. Воспользуемся процедурой, описанной в подразделе 3.7.1. Она позволяет найти несколько кратчайших маршрутов. При такой маршрутизации одни каналы могут быть более нагружены чем другие. Таким образом, снимается ограничение на равномерность потоков в сети, которое присутствовало в рассмотренных ранее моделях. На распределение потоков влияет также выбор узлов, в которых появляются новые сообщения.

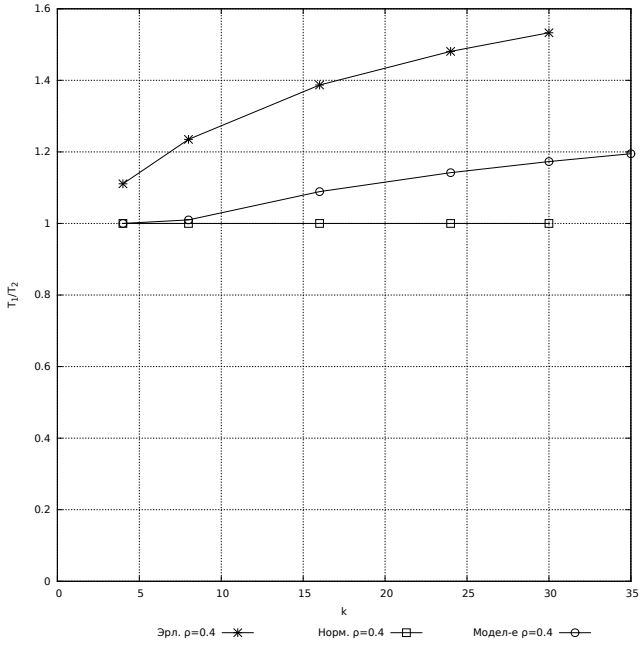
Как было показано ранее, выигрыш от кодирования может отличаться на разных длинах путей. Кроме того, оптимальная скорость кодирования, при которой достигается максимальный выигрыш, тоже может отличаться. Вопросу выбора оптимальной скорости кода посвящена работа [8]. Для иллюстрации на рисунке 3.27 показаны решетки разных размеров и посчитано количество маршрутов из левого нижнего угла в правый верхний. Ограничивая длину пути, мы ограничиваем также количество доступных маршрутов и наоборот. Удобной особенностью решетки является то, что между любой парой узлов все кратчайшие пути имеют одинаковую длину. Приведенные ниже результаты были получены при ограничении в 12 маршрутов между каждой парой передающих узлов. Это означает, что если между некоторой парой узлов меньше путей, то в модели они не генериру-



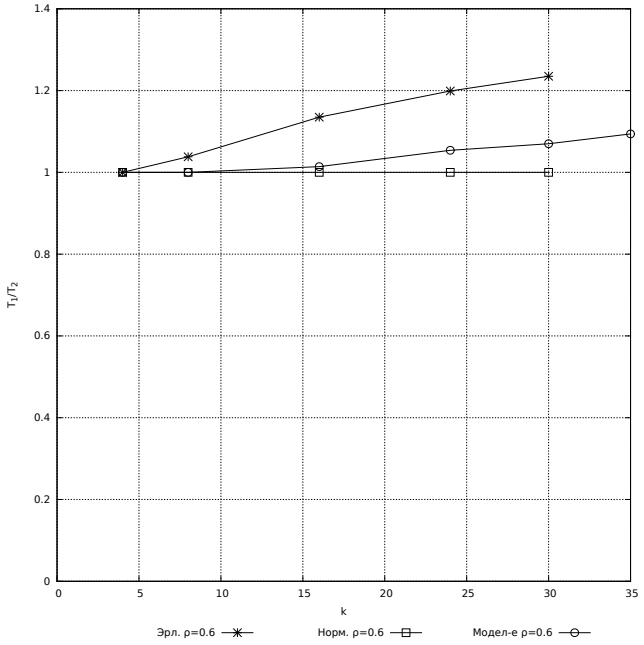
а) ρ=0.1



б) ρ=0.2



в) ρ=0.4



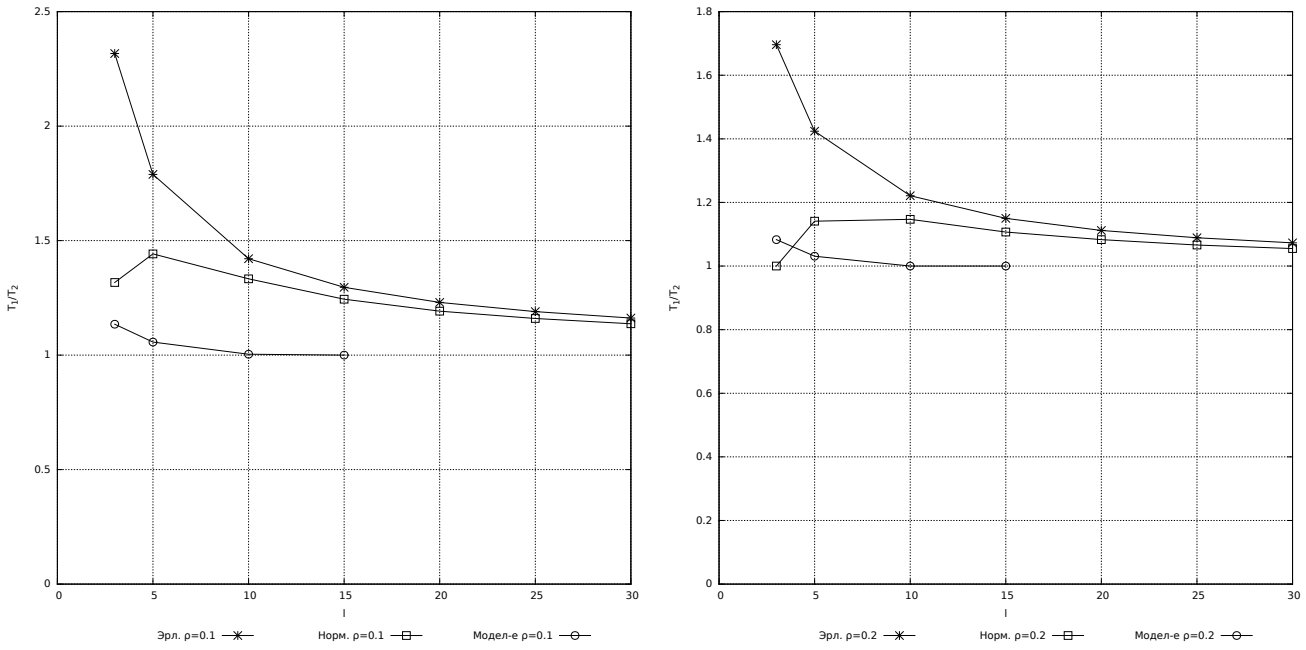
г) ρ=0.6

Рисунок 3.25 — Зависимость выигрыша от k при длине пути, равной 3

ют сообщения друг другу. Если между некоторой парой узлов путей больше, то в таблицу маршрутизации вносятся только 12.

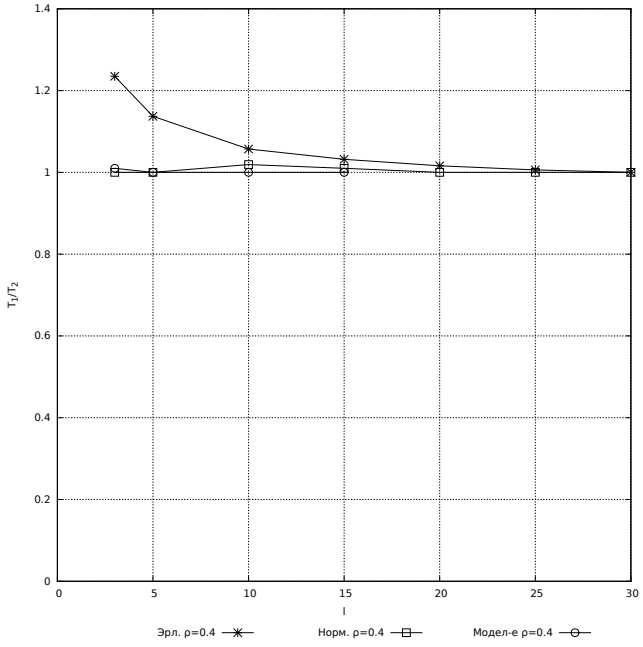
Для удобства сравнения с результатами из предыдущих подразделов, будем выбирать емкости каналов таким образом, чтобы загрузка по всей сети была одинакова. При расчете загрузки сети используется интенсивность потока соответствующая некодированной передаче.

Соберем сведения об описанной модели вместе:

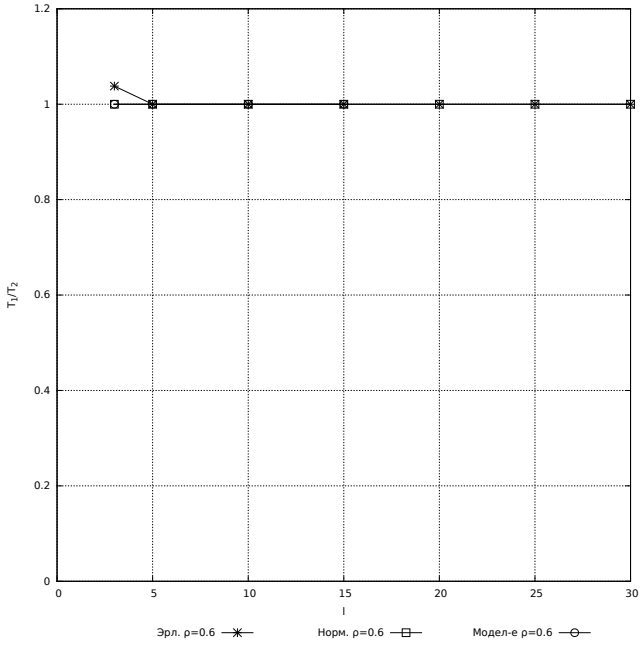


а) ρ=0.1

б) ρ=0.2



в) ρ=0.4



г) ρ=0.6

Рисунок 3.26 — Зависимость выигрыша от длины пути при k=8

– моделируются маршруты разной длины;
– маршруты зависимы;
– интенсивности потоков на каналах не равномерны;
– каналы имеют разные емкости, однако они выбраны так, что загрузка ρ на всех каналах одинаковая.

Приведенный список является перечислением ограничений, снятых с предыдущих рассмотренных моделей.

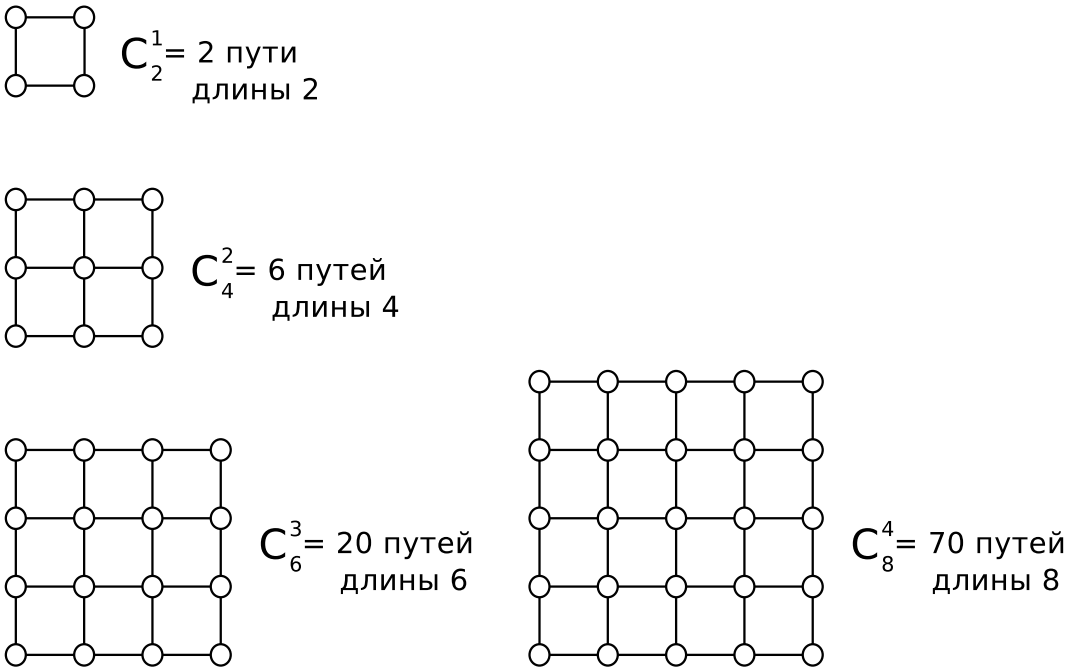
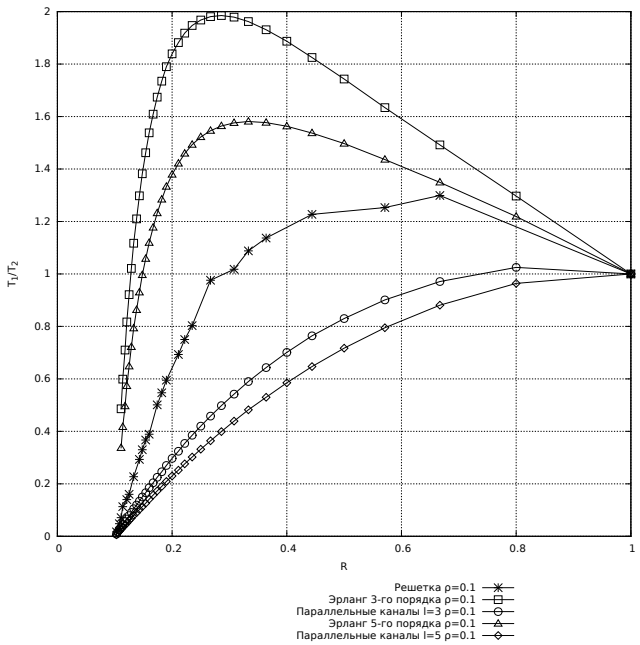


Рисунок 3.27 — Количество путей от крайнего левого снизу до крайнего правого сверху узла

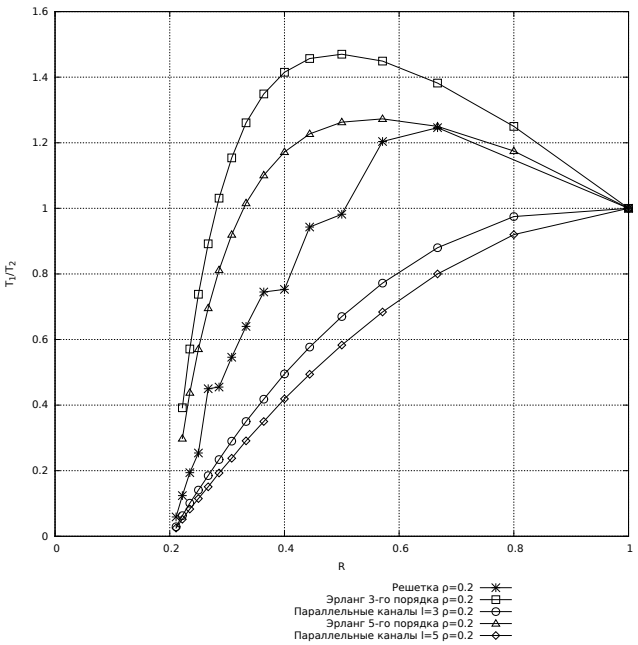
На рисунке 3.28 и 3.29 представлены зависимости выигрыша от скорости кода. На рисунке 3.29 представлены зависимости выигрыша от загрузки сети. Для сравнения на графики помещены соответствующие кривые для моделей, рассмотренных ранее. Представляет интерес сравнение порядка выигрыша и при каких параметрах он присутствует, учитывая, что в имитационной модели снят ряд ограничений, присутствовавших в предыдущих моделях.

Проанализировав кривые, можно сделать следующие выводы:

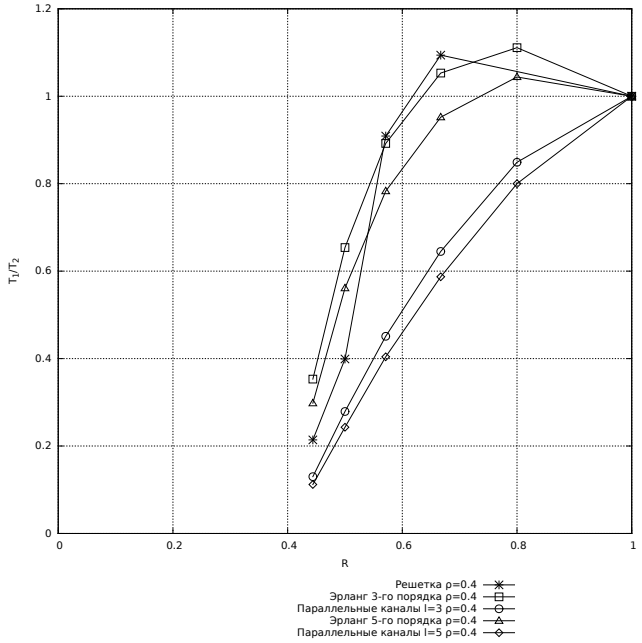
– Несмотря на снятые ограничения выигрыш присутствует.
– С ростом загрузки сети ρ выигрыш уменьшается; при $k = 4$ выигрыш исчезает при $\rho = 0.6$.
– Сравнивая случаи $k = 4$ и $k = 8$ отметим, что при большем k выигрыш снижается. Это может быть связано с тем, что в решетке у узлов относительно небольшое количество соседей и увеличение числа пакетов в сообщении приводит к перегрузке сети.



а) ρ=0.1



б) ρ=0.2

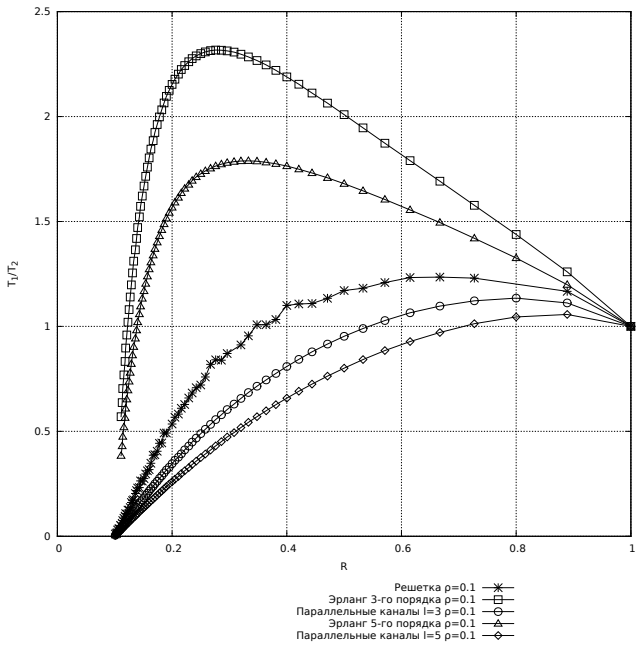


в) ρ=0.4

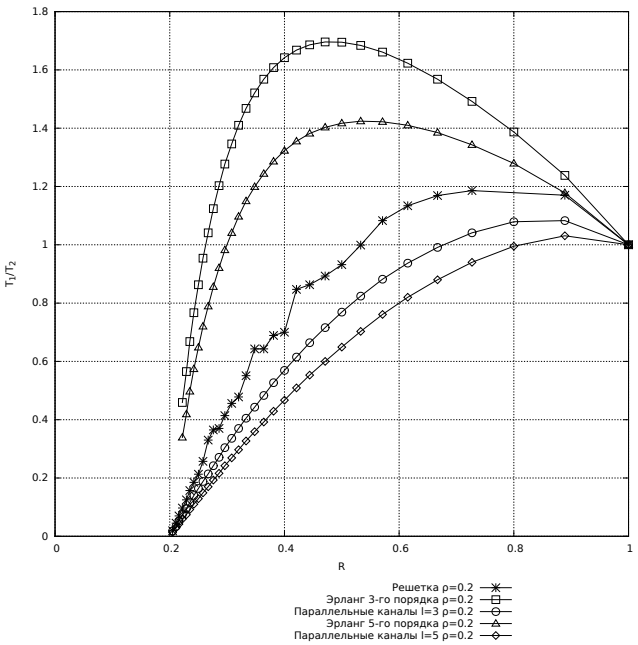
Рисунок 3.28 — Зависимость выигрыша от скорости кода при k=4

3.9 Эффективность транспортного кодирования при изменяемых емкостях каналов

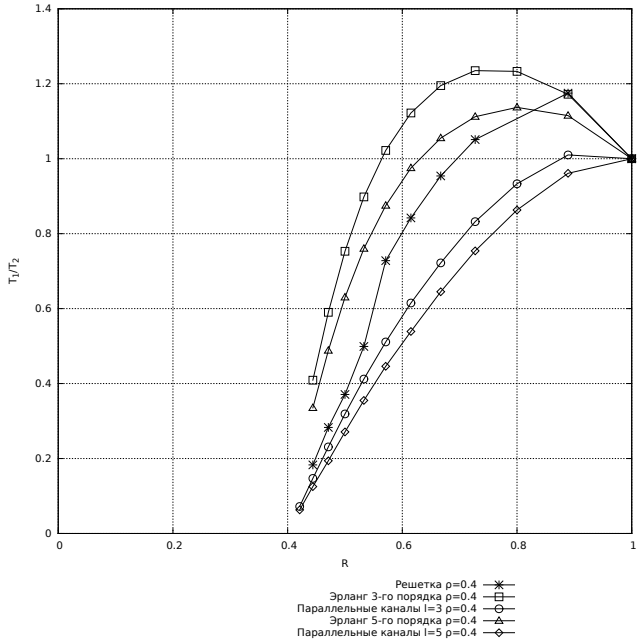
Для исследования эффективности транспортного кодирования при изменяемых емкостях каналов будем использовать имитационную модель сети, описанную в 3.7.1 и модифицированную таким образом, чтобы учесть эффект изменения



а) ρ=0.1



б) ρ=0.2



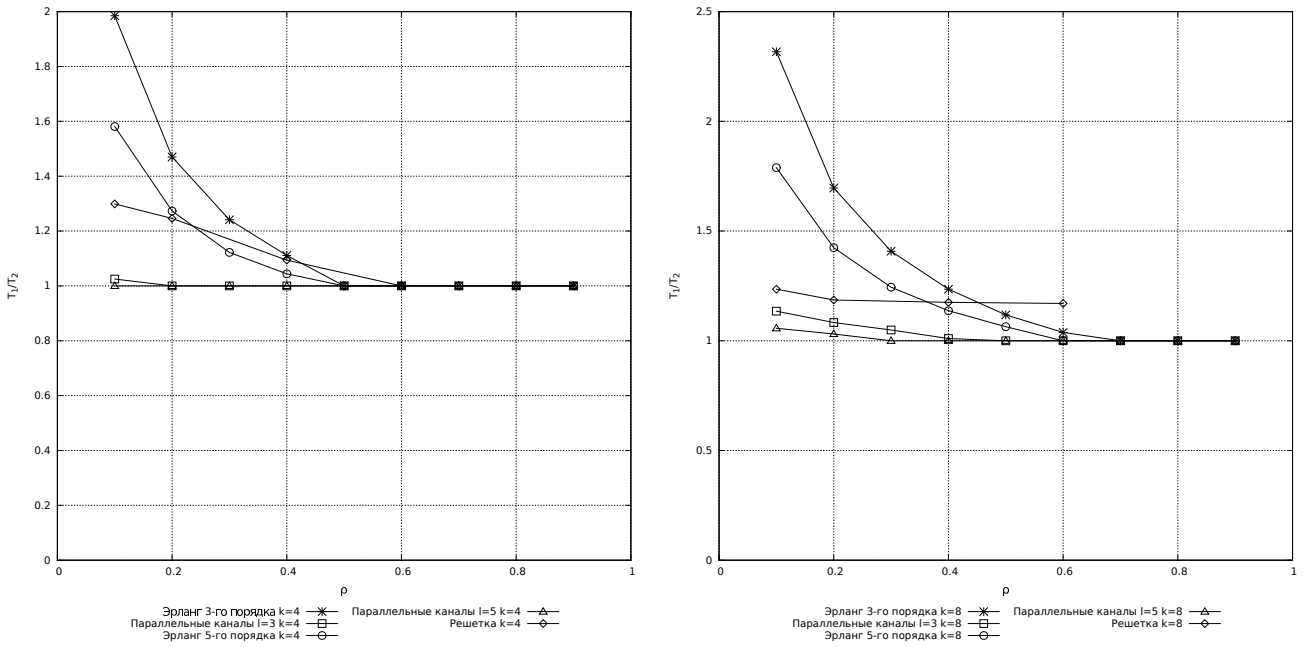
в) ρ=0.4

Рисунок 3.29 — Зависимость выигрыша от скорости кода при k=8

емкостей каналов. Результаты описанные в данном подразделе были изложены в работе [4].

Для моделирования эффекта изменения емкости введем два состояния канала:

– нормальное состояние – значение емкости канала задано матрицей емкостей $C[i][j]$;



а) k=4 б) k=8

Рисунок 3.30 — Зависимость выигрыша от загрузки сети

– “плохое” состояние – значение емкости канала устанавливается равным $C[i][j]\delta_{LC}/100$, где $\delta_{LC} = 0 \ldots 100\%$.

Пусть длительность плохого состояния является константой и устанавливается параметром t_{LC}. Длительность нормального состояния распределена по экспоненциальному закону со средним $1/\lambda_{LC}$, где λ_{LC}. Параметры δ_{LC},λ_{LC},t_{LC} задаются в начале моделирования.

Описанные изменения модели сделаны таким образом, чтобы описать ситуации, когда в основном канал находится в нормальном состоянии, но иногда случаются кратковременные ухудшения, связанные с падением пропускной способности. Так происходит, например, в подвижной сети при беспроводной передаче, когда на пути распространения сигнала возникает какое-то препятствие.

В качестве топологии сети выбрана решетка размером 5x5 (рисунок 3.27). Такая топология позволяет иметь несколько маршрутов между отправителем и получателем и обладает регулярной структурой. Подмножество узлов, в которых появляются новые сообщения, выбрано таким образом, чтобы количество маршрутов было равно 10. Интенсивность появления новых маршрутов равна 100 сообщений в единицу времени. Длина одного сообщения k равна 4 пакета. Для данного сценария была найдена скорость кодировани, равная $R = 0.6$, при которой средняя задерка сообщения наименьшая.

Емкости каналов вычисляются таким образом, чтобы получить среднюю загрузку сети, равную заданной ρ, по формуле

$$\rho = \frac{\lambda}{\mu C},$$

где λ – интенсивность потока пакетов посылаемых по каналу, $1/(\mu C)$ – среднее время передачи пакета по каналу. Тогда емкость канала между узалми i и j может быть выражена следующим образом.

$$C[i][j] = \frac{\lambda[i][j]}{\rho\mu},$$

где $\lambda[i][j]$ – сумма интенсивностей всех потоков, передаваемых по каналу.

Параметр λ_{LC} может меняться от сценария к сценарию, но везде будет задаваться как процент от интенсивности потока, проходящего по каналу. Это означает, что каналы с разной интенсивностью потока трафика будут иметь разную интенсивность перехода в состояние с низкой емкостью. Такое правило выбора значения для λ_{LC} позволяет согласовать его с интенсивностью потока и избежать сильных различий между ними.

Параметр t_{LC} будем задавать кратным длительности передачи одного пакета по каналу. Это означает, что каналы с разными емкостями будут иметь разную длительность плохого состояния, однако соотношение t_{LC} и времени передачи пакета будет единым для всей сети.

Для исследования эффективности кодирования в зависимости от значений параметров δ_{LC},λ_{LC},t_{LC} подготовлены три различных сценария. Для каждого сценария моделирования были построены кривые средней задержки сообщения и выигрыша от кодирования. Средняя задержка сообщения вычислена по всей сети. Выигрыш от кодирования будем вычислять отличным от предыдущих разделов образом как $\left(\bar{T}_{unc} - \bar{T}_{enc}\right) 100/\bar{T}_{unc}$, где $\bar{T}_{unc}$ – средняя задержка сообщения без кодирования, $\bar{T}_{enc}$ – средняя задержка сообщения с кодированием. Такая оценка позволит лучше видеть малые выигрыши от кодирования.

Параметры первого сценария заданы в таблице 3.1. Цель данного сценария заключается в изучении средней задержки сообщения и выигрыша от кодирования в зависимости от частоты перехода каналов в состояние низких емкостей λ_{LC}. Рассмотрим сначала ситуацию с низкой загрузкой сети $\rho = 0.1$ (рисунки 3.31,

3.33). Как видно из графиков, чем чаще происходит переход в состояние низких емкостей, тем сильнее отличаются задержки сообщений при использовании кодирования и без него. Таким образом, чем чаще переход в плохое состояние, тем больше передача с кодированием выигрывает у передачи без кодирования.

Таблица 3.1

Сценарий 1: изменение интенсивности перехода в состояние низких емкостей

R	0.6
ρ	0.1, 0.3
δ_{LC}	30
t_{LC}	3
λ_{LC}	$5, \ldots, 40$

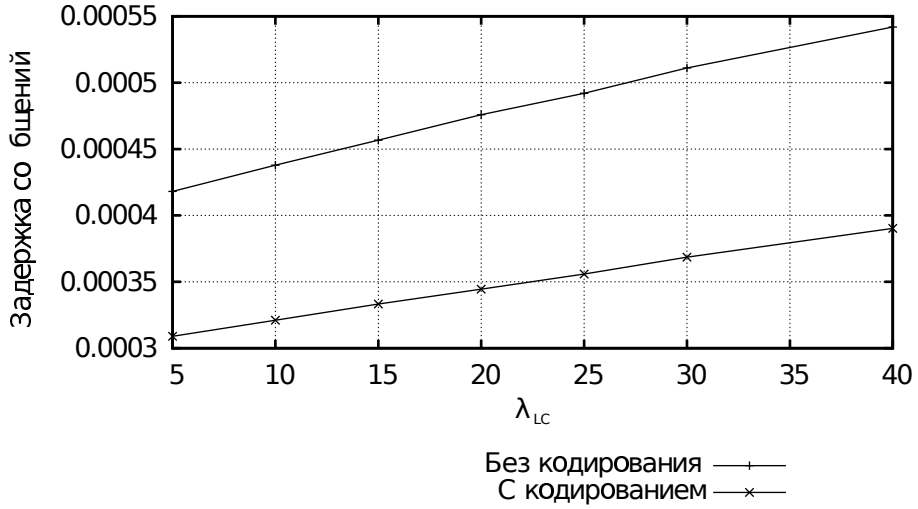


Рисунок 3.31 — Сценарий 1: зависимость средней задержки сообщений от интенсивности перехода в состояние низких емкостей при $\rho = 0.1$

В случае более высокой загрузки сети $\rho = 0.3$ эффективность передачи с кодированием гораздо ниже (рисунки 3.32, 3.34). Применение кодирования увеличивает избыточность в сети и как следствие увеличивает загрузку сети, что приводит к увеличению задержки. В случае $\rho = 0.3$ негативный эффект от увеличения избыточности выше в сравнении с ситуацией при $\rho = 0.1$ и выигрыш от кодирования получается ниже.

Параметры, определяющие второй сценарий, заданы в таблице 3.2. Цель данного сценария состоит в изучении зависимости средней задержки сообщения и выигрыша от кодирования от того, насколько сильно падает емкость каналов (параметр δ_{LC}) в состоянии низкой емкости. Рассмотрим ситуацию с небольшой загрузкой сети $\rho = 0.1$ и загрузкой сети $\rho = 0.3$. Результаты моделирования,

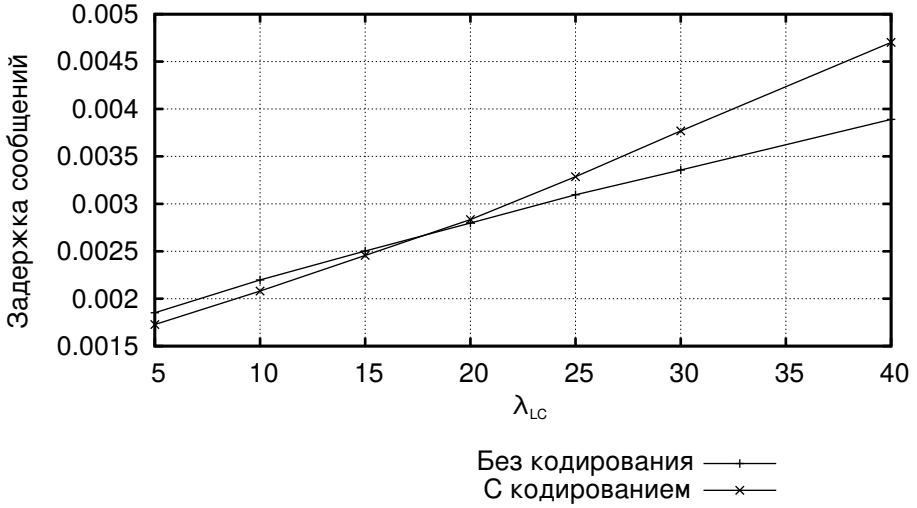


Рисунок 3.32 — Сценарий 1: зависимость средней задержки сообщения от интенсивности перехода в состояние низких емкостей при $\rho = 0.3$

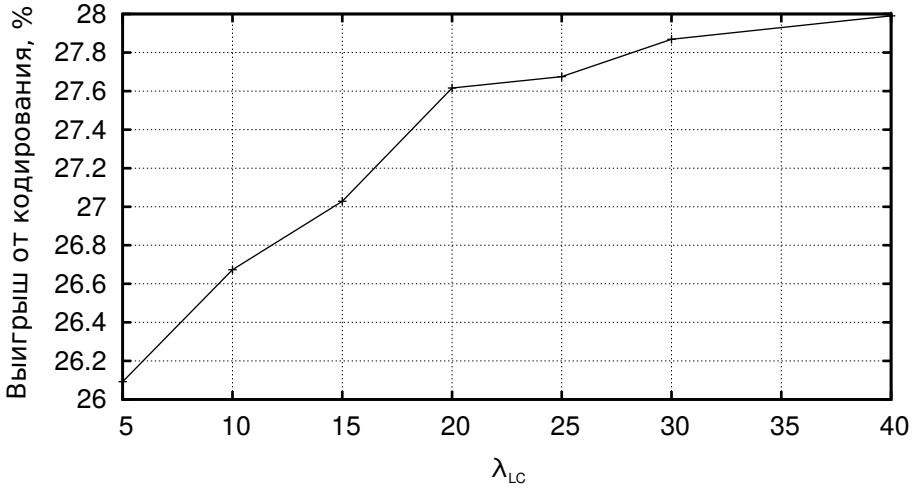


Рисунок 3.33 — Сценарий 1: зависимость выигрыша от кодирования от интенсивности перехода в состояние низких емкостей при $\rho = 0.1$

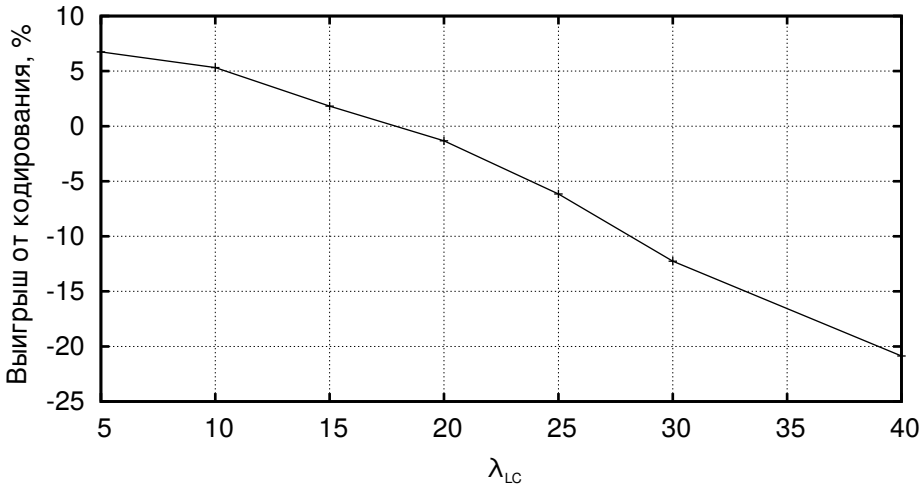


Рисунок 3.34 — Сценарий 1: зависимость выигрыша от кодирования от интенсивности перехода в состояние низких емкостей при $\rho = 0.3$

представленые на рисунках 3.35-3.38, показывают, что чем сильнее проседает емкость каналов, тем больше выигрыш от применения кодирования. Другими словами, переход в состояние низких емкостей канала меньше влияет на задержку сообщения при использовании кодирования. Передача без кодирования более чувствительна к падению емкостей каналов. Когда загрузка сети становится равной $\rho = 0.3$, выигрыш от кодирования все еще присутствует только в случае небольшого проседания емкостей. В первом сценарии мы увидели, что значение параметра $\lambda_{LC} = 30$ не является самым лучшим для загрузки $\rho = 0.3$, однако все равно, при $\delta_{LC} = 90\%$ удается получить ненулевой выигрыш.

Таблица 3.2

Сценарий 2: изменение величины проседания емкостей

R	0.6
ρ	0.1
t_{LC}	3
λ_{LC}	30
δ_{LC}	10, 30, 50, 70, 90

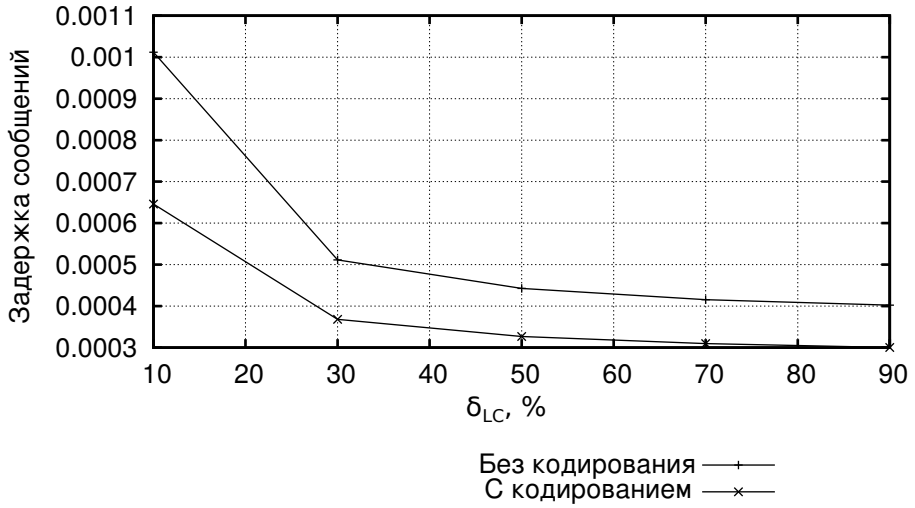


Рисунок 3.35 — Сценарий 2: зависимость средней задержки от δ_{LC} при $\rho = 0.1$

Параметры третьего сценария заданы в таблице 3.3. Его цель заключается в изучении средней задержки и выигрыша от кодирования в зависимости от длительности пребывания в состоянии низких емкостей t_{LC}. Результаты моделирования представлены на рисунках 3.39-3.42. При загрузке сети $\rho = 0.1$ существует оптимальное значение для величины t_{LC}, при котором выигрыш от кодирования принимает наибольшее значение. Вспомним, что значение $t_{LC} = 3$ означает, что

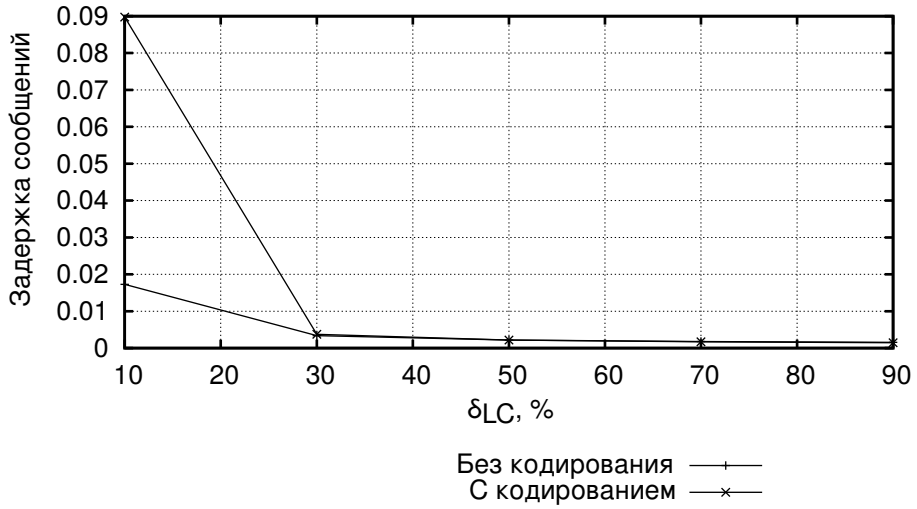


Рисунок 3.36 — Сценарий 2: зависимость средней задержки от δ_{LC} при $\rho = 0.3$

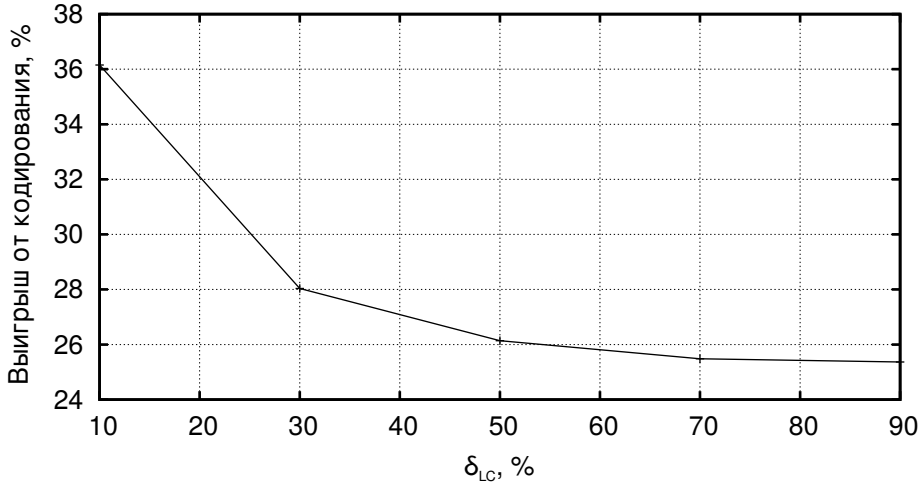


Рисунок 3.37 — Сценарий 2: зависимость выигрыша от кодирования от δ_{LC} при $\rho = 0.1$

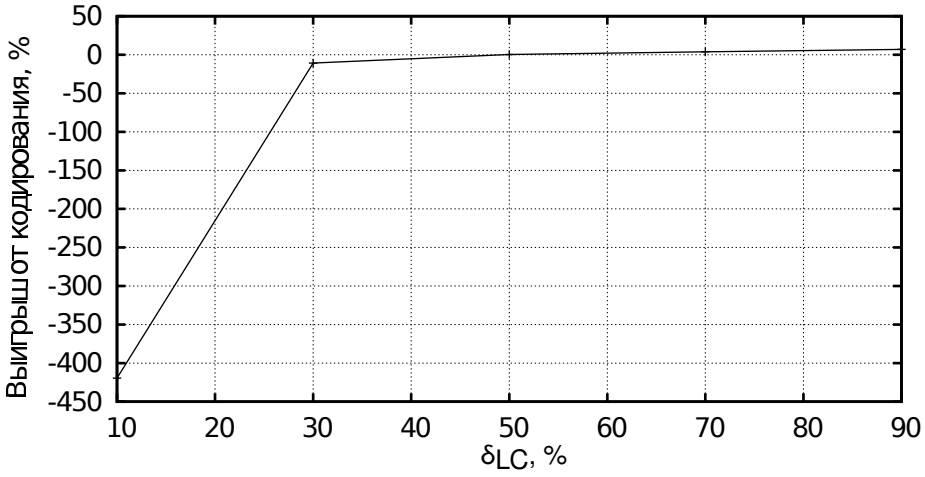


Рисунок 3.38 — Сценарий 2: зависимость выигрыша от кодирования от δ_{LC} при $\rho = 0.3$

длительность состояния с низкими емкостями в три раза больше времени передачи одного пакета по каналу.

Как и в предыдущих сценариях, передача с кодированием более чувствительна к параметрам, задающим состояние с низкими емкостями при загрузке сети $\rho = 0.3$.

Таблица 3.3

Сценарий 3: изменение длительности состояния с низкими емкостями

R	0.6
ρ	0.1
λ_{LC}	30
δ_{LC}	10
t_{LC}	1, 3, 6, 8, 10, 13, 16

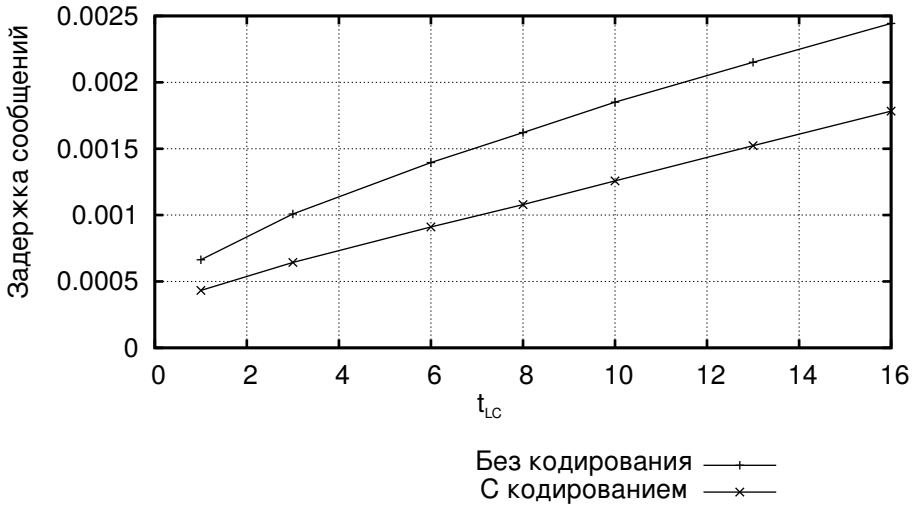


Рисунок 3.39 — Сценарий 3: зависимость средней задержки от t_{LC} при $\rho = 0.1$

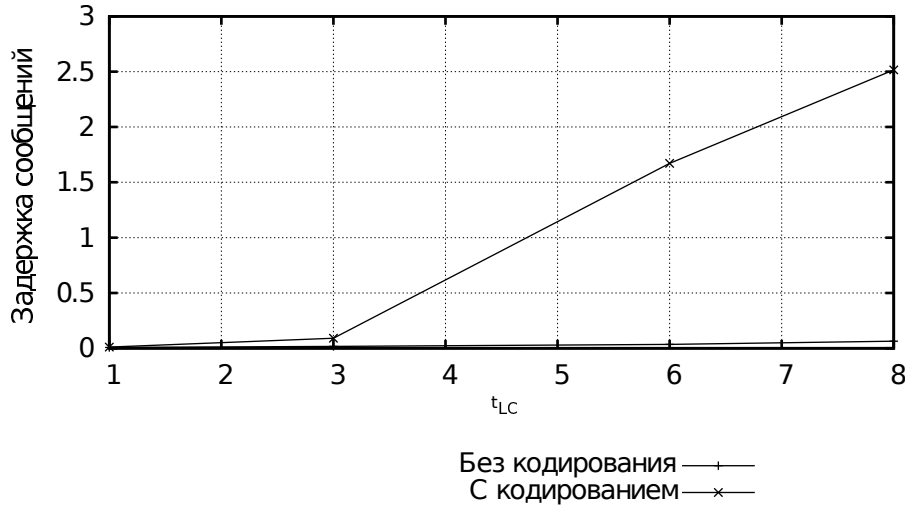


Рисунок 3.40 — Сценарий 3: зависимость средней задержки от t_{LC} при $\rho = 0.3$

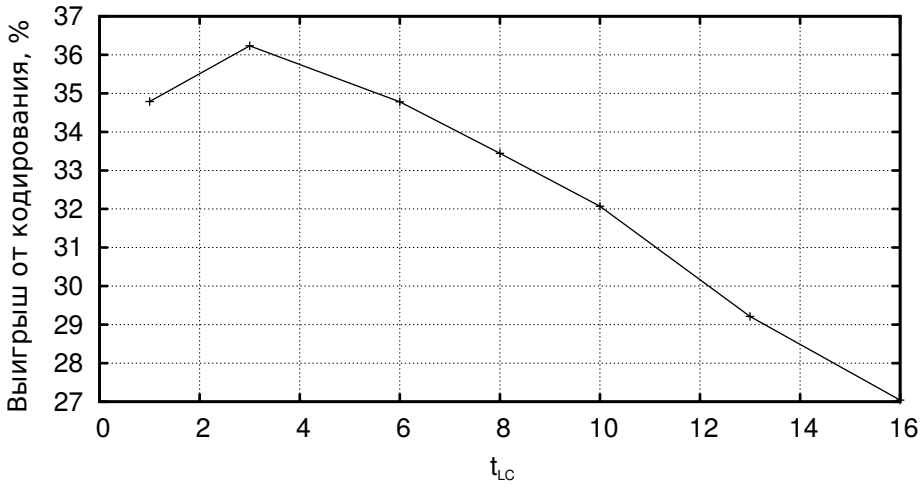


Рисунок 3.41 — Сценарий 3: зависимость выигрыша от кодирования от t_{LC} при $\rho = 0.1$

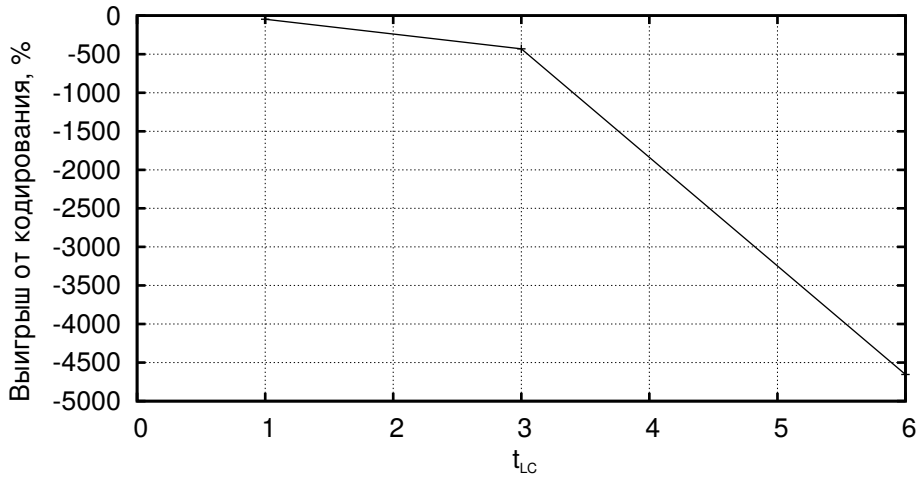


Рисунок 3.42 — Сценарий 3: зависимость выигрыша от кодирования от t_{LC} при $\rho = 0.3$

После изучения представленных сценариев можно сделать следующие выводы.

– Эффективность транспортного кодирования при изменяемых емкостях каналов зависит от загрузки сети. Чем она выше, тем ниже эффективность кодирования.
– Чем чаще происходит падение емкостей каналов тем выше выигрыш от кодирования. Однако стоит заметить, что, возможно, при других параметрах сети будет существовать точка экстремума для интенсивности “плохого состояния”.
– Наличие “плохих” каналов имеет меньшее влияние при передаче с кодированием, чем без нее. Кроме того, передача без кодирования более чувствительна к величине падения емкостей каналов.

– Чем сильнее происходит падение емкостей каналов, тем выше выигрыш от кодирования, однако, возможно, при других параметрах сети будет существовать точка экстремума для величины падения емкостей.
– Изученные сценарии показывают наличие оптимального значения для длительности состояния с низкими емкостями. При этом значении выигрыш от кодирования максимален.

3.10 Введение ограничения на время жизни пакетов

Время жизни пакета в реальных сетях ограничено и обозначается как TTL (от англ. time to live). Это время, по истечении которого, пакет удаляется из сети, если он еще не доставлен до адресата. В данном подразделе исследуется вероятность доведения сообщения до адресата и выигрыш от кодирования при ограничении на время жизни пакетов. Вопросы, связанные с временем жизни пакетов, рассматривались автором в работах [10] и [9].

Если TTL больше времени прихода k-го пакета на узле-получателе, то сообщения будут успешно приняты. Можно допускать удаление или потерю до $n-k$ пакетов, чтобы успешно декодировать сообщение. Если TTL меньше, то можно говорить о вероятности доведения сообщения до адресата $P_{\text{дов.}}$. Таким образом,

$$P_{\text{дост.}} = P\{\text{принято} \geq k \text{ пакетов}\}.$$

Для аналитических моделей, рассмотренных ранее, $P_{\text{дов.}}$ можно вычислить, используя функцию распределения задержек сообщений, т.к.

$$P\{\text{принято} \geq k \text{ пакетов}\} = P\{\text{время прихода k-го пакета} < TTL\},$$

$$P\{\text{время прихода k-го пакета} < TTL\} = F_{k:n}\left(TTL\right),$$

где $F_{k:n}\left(t\right)$ – функция распределения задержки сообщений.

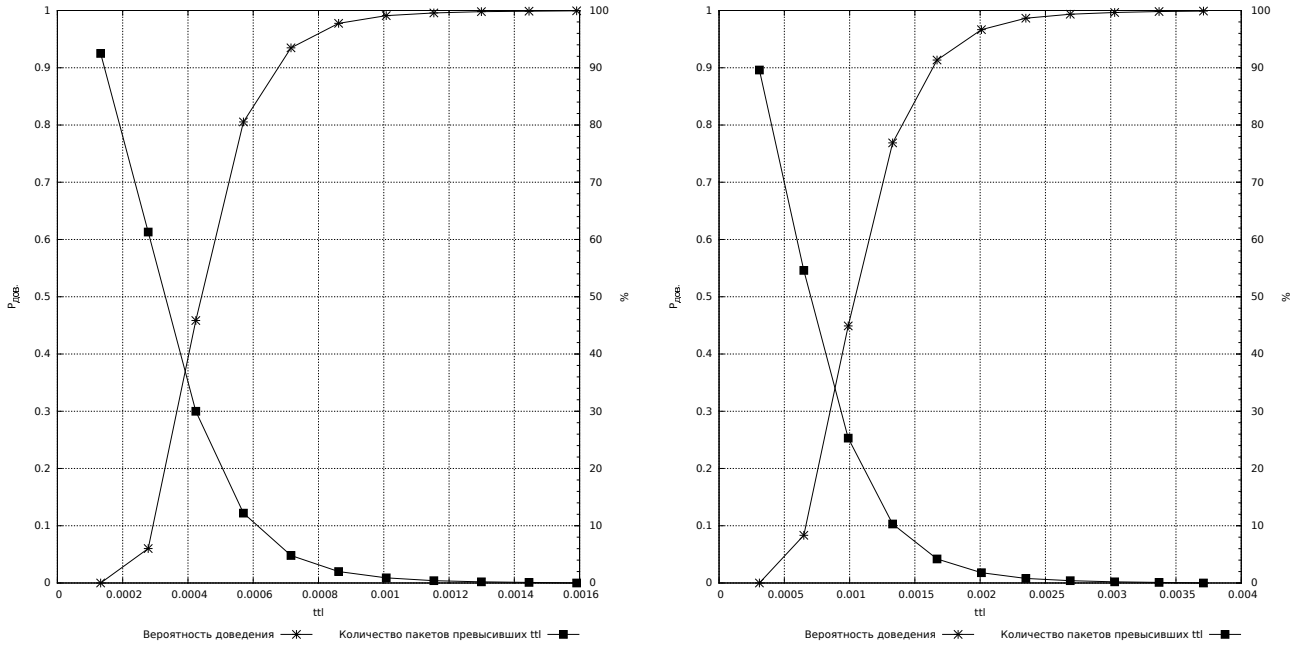
Обычно время жизни пакета вводится для того, чтобы в случае некорректной работы сети пакеты не блуждали по ней бесконечно долго и не перегружали ее таким образом. В случае использования транспортного кодирования TTL мог бы использоваться для уменьшения ненужной избыточности, удаляя те пакеты,

принадлежащие сообщению, которые находятся в сети после получения адресатом k-го пакета из сообщения, что способствовало бы еще уменьшению задержки пакетов.

Для исследования эффектов от введения времени жизни пакетов воспользуемся имитационной моделью, описанной в подразделе 3.7.1. Будем изменять величину TTL в пределах $[0.3\bar{T}_{k:n} \ldots 4\bar{T}_{k:n}]$, где $\bar{T}_{k:n}$ – средняя задержка сообщений в сети при использовании кодирования. Для всех пакетов сети вне зависимости от отправителя и получателя будем устанавливать единое значение TTL.

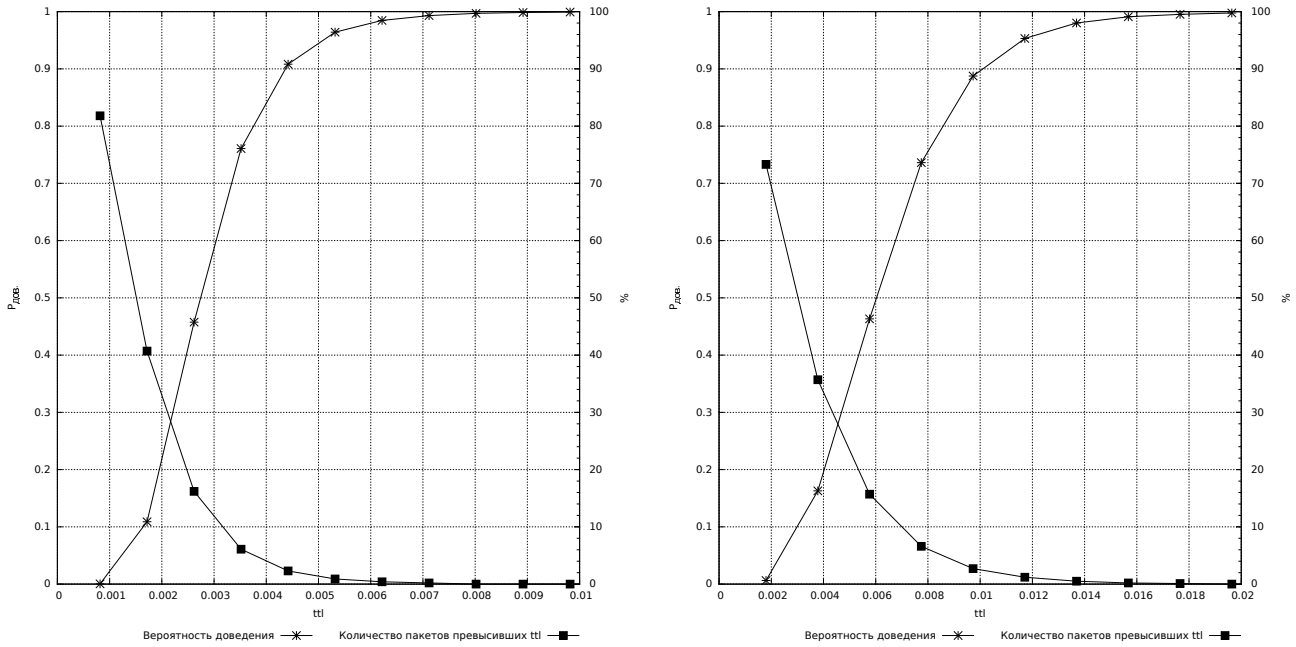
На рисунках 3.43, 3.44 показана зависимость между вероятностью доведения сообщения до адресата и конкретным значением TTL, а также зависимость числа удаленных из сети пакетов от TTL. На рисунке 3.45 показана зависимость выигрыша от кодирования при различных TTL. Проанализировав кривые, можно сделать следующие выводы:

– при уменьшении TTL вероятность $P_{\text{дов.}}$ спадает очень медленно и имеет достаточно длинный участок высоких значений;
– существенного снижения трафика на участке высоких $P_{\text{дов.}}$ для рассматриваемой модели сети не наблюдается;
– на участке высоких $P_{\text{дов.}}$ для рассматриваемой модели сети выигрыш от кодирования практически не меняется, следовательно не имеет смысла уменьшать TTL для повышения выигрыша;



а) ρ=0.1 б) ρ=0.2

Рисунок 3.43 — Вероятность доведения сообщения до адресата при разных TTL



а) ρ=0.4 б) ρ=0.6

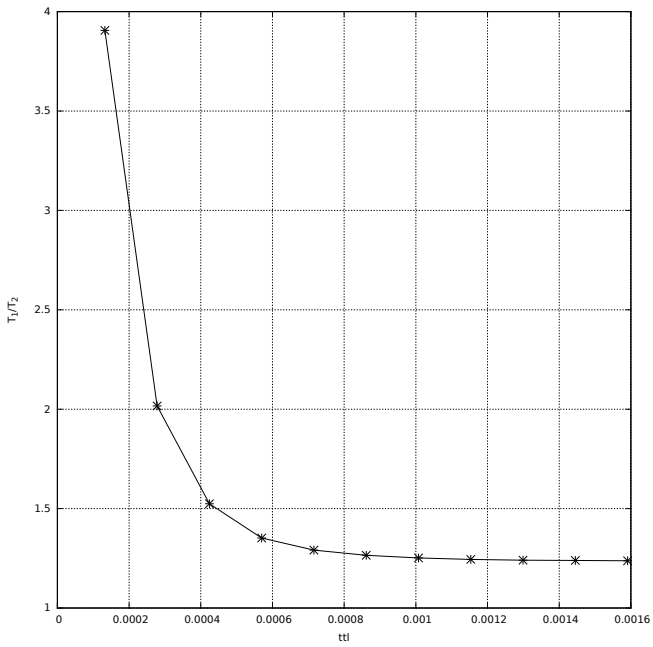
Рисунок 3.44 — Вероятность доведения сообщения до адресата при разных TTL

3.11 Кодирование с переменной скоростью кода

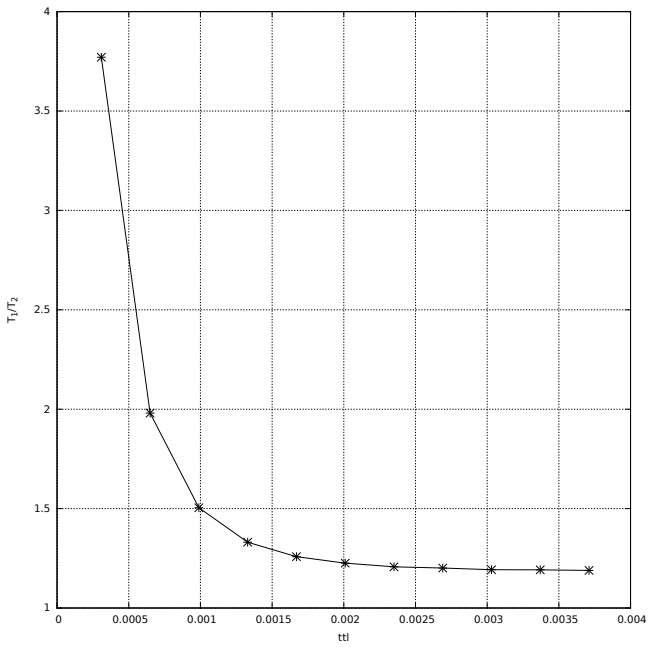
До сих пор в работе скорость кода $R = k/n$ в транспортном кодировании была фиксирована при передаче от источника до получателя. В данном подразделе рассматривается передача в нерегулярных сетях, предлагается метод передачи, при котором используется транспортное кодирование с переменной скоростью кода.

Под нерегулярной сетью понимается сеть, которая состоит из неодинаковых подсетей. Подсети могут отличаться, например, загрузкой, топологией. В предлагаемом методе в промежуточных узлах, соединяющих подсети, выбирается новое значение скорости, адаптированное для следующей подсети, далее происходит перекодирование сообщения в промежуточном узле с новой скоростью. Так как выбор новой скорости происходит неоднократно и с учетом параметров конкретной подсети, то ожидается, что такой способ передачи будет более выгодным в сравнении со случаем с фиксированной скоростью кода. Алгоритм выбора перекодирующих узлов в данном разделе не рассматривается. Результаты представленные в данном подразделе были изложены автором в работах [2; 6].

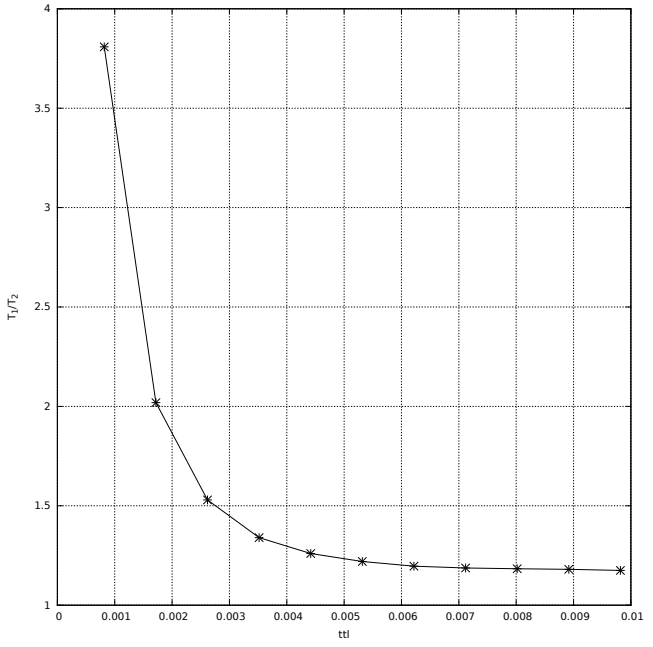
Далее в работе исследуется эффективность передачи с использованием перекодирования. Для этого рассматривается несколько примеров и производится



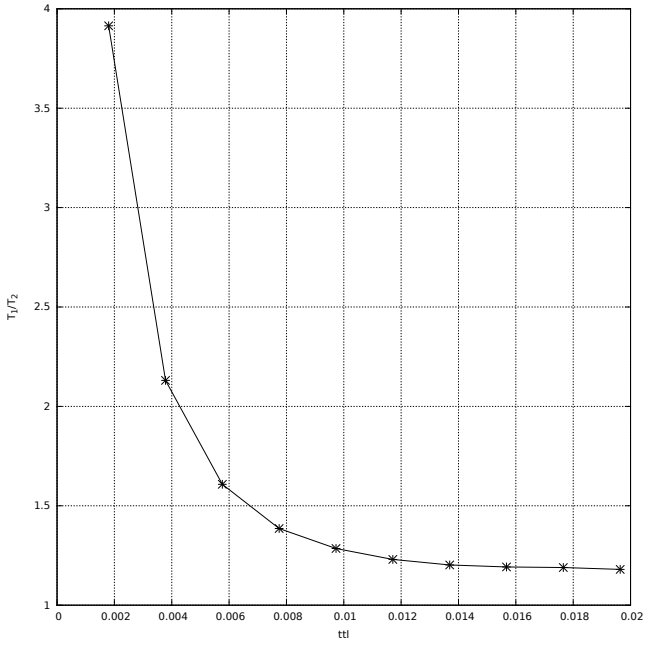
а) ρ=0.1



б) ρ=0.2



в) ρ=0.4



г) ρ=0.6

Рисунок 3.45 — Зависимость выигрыша от кодирования при различных TTL

расчет задержки сообщений, сравниваются задержки для случая с перекодированием и без него.

3.11.1 Исследование выигрыша от использования перекодирования

Для простоты рассмотрим две сети $S1$ и $S2$, соединенных через одну общую точку. Пусть в данной точке будет происходить перекодирование. Пусть загрузка одной сети равна ρ_1, а загрузка второй сети ρ_2. Тогда средняя задержка сообщений в одной сети равна $\bar{t}\,(\rho_1)$, а средняя задержка сообщений во второй сети $\bar{t}\,(\rho_2)$.

Посчитаем отношение средней задержки сообщений при передаче без перекодирования к средней задержке сообщений при передаче с использованием перекодирования:

$$G = \frac{T_0}{T_1 + T_2}, \tag{3.48}$$

где T_0 — средняя задержка сообщения без использования перекодирования, T_1 — средняя задержка сообщения в сети $S1$, T_2 — средняя задержка сообщения в сети $S2$. Будем использовать отношение (3.48) для оценки выигрыша от использования перекодирования. Для вычисления T_0, T_1, T_2 воспользуемся выражением, полученным в подразделе 3.5:

$$T = \min_R \left\{ \bar{t}\,(\rho/R) \sum_{i=k/R-k+1}^{k/R} i^{-1} \right\};$$

тогда

$$T_0 = \min_R \left\{ \bar{t}\,(\rho_1/R) \sum_{i=k/R-k+1}^{k/R} i^{-1} + \bar{t}\,(\rho_2/R) \sum_{i=k/R-k+1}^{k/R} i^{-1} \right\}, \tag{3.49}$$

$$T_1 = \min_R \left\{ \bar{t}\,(\rho_1/R) \sum_{i=k/R-k+1}^{k/R} i^{-1} \right\}, \tag{3.50}$$

$$T_2 = \min_R \left\{ \bar{t}\,(\rho_2/R) \sum_{i=k/R-k+1}^{k/R} i^{-1} \right\}. \tag{3.51}$$

Подставим выражения для T_0, T_1, T_2 в отношение (3.48), получим

$$G = \frac{\bar{t}\,(\rho_1/R_0)\sum_{i=k/R_0-k+1}^{k/R_0} i^{-1} + \bar{t}\,(\rho_2/R_0)\sum_{i=k/R_0-k+1}^{k/R_0} i^{-1}}{\bar{t}\,(\rho_1/R_1)\sum_{i=k/R_1-k+1}^{k/R_1} i^{-1} + \bar{t}\,(\rho_2/R_2)\sum_{i=k/R_2-k+1}^{k/R_2} i^{-1}}, \tag{3.52}$$

Тогда условие выгодности перекодирования принимает вид

$$\begin{aligned} &\bar{t}\,(\rho_1/R_0)\sum_{i=k/R_0-k+1}^{k/R_0} i^{-1} + \bar{t}\,(\rho_2/R_0)\sum_{i=k/R_0-k+1}^{k/R_0} i^{-1} > \\ &\bar{t}\,(\rho_1/R_1)\sum_{i=k/R_1-k+1}^{k/R_1} i^{-1} + \bar{t}\,(\rho_2/R_2)\sum_{i=k/R_2-k+1}^{k/R_2} i^{-1}, \end{aligned} \tag{3.53}$$

На рисунке 3.46 представлен график зависимости выигрыша G от ρ_2 при фиксированных значениях ρ_1, полученного расчетом по формуле (3.52). Графики показывают, что с ростом $\rho_2 - \rho_1$ выигрыш от использования перекодирования возрастает.

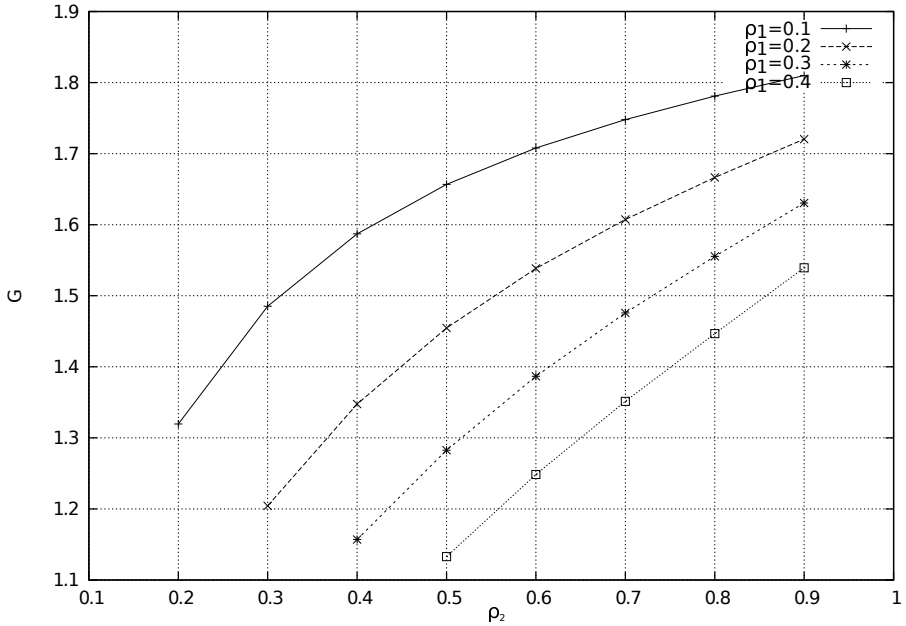


Рисунок 3.46 — Выигрыш от перекодирования при разных значениях загрузки подсетей ρ_1, ρ_2

Рассмотрим другую модель сети. Пусть имеются две соседние сети, которые можно представить в виде системы параллельных каналов (см. рисунок 3.19 из подраздела 3.7.2). Будем считать, что все маршруты из одной сети в другую проходят через один общий узел. Рассмотрим передачу из одной такой подсети

в другую с использованием перекодирования. Топология рассматриваемого примера представлена на рисунке 3.47.

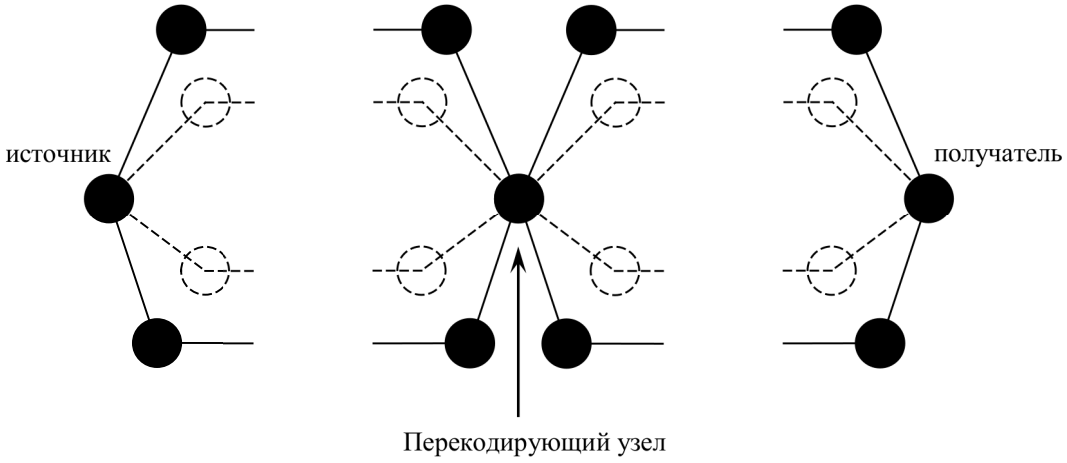


Рисунок 3.47 — Пример двух соседних подсетей

Для расчета выигрыша от перекодирования будем считать, что емкости всех каналов одинаковы, и средняя длина одного пакета $1/\mu$ зафиксирована. Задержка одного пакета на пути от источника до получателя представляет собой сумму задержек на каждом канале пути. Для расчета средней задержки сообщения воспользуемся моделью задержки, разработанной в подразделе 3.6.1, в которой задержка пакета распределена по закону Эрланга. Для расчета задержки кодированного сообщения воспользуемся выражением (3.39). Положим $\mu c = 1$, тогда

$$\alpha = \frac{l}{\bar{t}\,(\rho)} = \frac{l\mu c\,(1-\rho)}{l} = \mu c\,(1-\rho)\,. \tag{3.54}$$

Перепишем эту формулы для средней задержки сообщений в зависимости от l, R, ρ. Тогда средняя задержка сообщения для передачи без перекодирования равна

$$T_0 = \min_R \left\{T\,(l_1 + l_2, R, \rho)\right\}. \tag{3.55}$$

Средняя задержка сообщения на участке от источника сообщений до перекодирующего узла равна

$$T_1 = \min_R \{T(l_1, R, \rho)\}. \tag{3.56}$$

Средняя задержка сообщения на участке от перекодирующего узла до узла-получателя равна

$$T_2 = \min_R \{T(l_2, R, \rho)\}. \tag{3.57}$$

Тогда выигрыш от применения перекодирования равен

$$G = \frac{\min_R \{T(l_1 + l_2, R, \rho)\}}{\min_R \{T(l_1, R, \rho)\} + \min_R \{T(l_2, R, \rho)\}}. \tag{3.58}$$

На рисунках 3.48-3.51 представлены графики для выигрыша от перекодирования. В рассматривамой модели сети выполняются следующие условия.

1. Оптимальные скорости кодирования в подсетях отличаются.
2. Использование транспортного кодирования в подсетях дает разный выигрыш. Чем больше эта разница, тем больше будет выигрыш от использования перекодирования.

На рисунке 3.48 представлена зависимость выигрыша от использования перекодирования (3.58) от отношения количества путей в подсетях (N_2/N_1) при длине сообщения 8 пакетов и следующих длинах путей в подсетях: $l_1 = l_2 = 3$, $l_1 = l_2 = 6$. Как видно из рисунка, наличие перекоса в количестве путей в подсетях приводит к выигрышу при использовании перекодирования, с ростом различия количества путей в подсетях $N_2 - N_1$ растет выигрыш от перекодирования.

На рисунке 3.49 представлена та же зависимость для случая, когда длина путей в подсетях отличается. Этот график требует более подробного разъяснения. Кривые с легендами $k = 8$,$l_1 = 6$,$l_2 = 3$,$N_1 = 8$ и $k = 8$,$l_1 = 3$,$l_2 = 6$,$N_1 = 8$ на первый взгляд соответствуют идентичной конфигурации сети и выигрыши должны быть одинаковые, однако это не так. В первом случае $l_1 = 6$, $l_2 = 3$, а во втором случае $l_1 = 3$, $l_2 = 6$. Подсети отличаются количеством путей. В первой подсети оно фиксировано $N_1 = 8$, а во второй подсети больше чем в первой. Значение N_2/N_1 отложено по оси X. Увеличение количества маршрутов для случая коротких и длинных путей дает разный выигрыш от транспортного кодирования, поэтому и выигрыш от перекодирования сильно отличается.

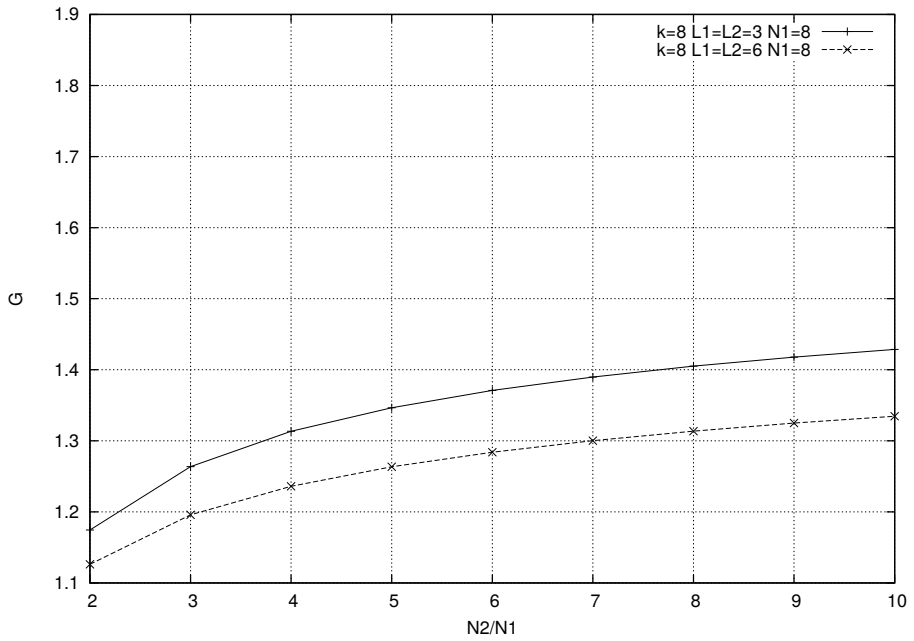


Рисунок 3.48 — Выигрыш от перекодирования для системы с параллельными каналами в зависимости от отношения количества путей в подсетях, случай равных длин путей

Для этой же модели сети рассмотрим выигрыш от перекодирования для разных загрузок сети. Для этого необходимо переписать формулу (3.58), чтобы учесть различие в загрузке:

$$G = \frac{\min\limits_{R} \{T(l_1, R, \rho_1) + T(l_1, R, \rho_2)\}}{\min_R \{T(l_1, R, \rho_1)\} + \min_R \{T(l_1, R, \rho_2)\}} \tag{3.59}$$

На рисунке 3.50 представлен выигрыш от перекодирования для случая разных значений загрузки в подсетях ($\rho_1 = 0.3$, $\rho_2 = 0.6$). На графике видно, что различие в загрузках делает выигрыш от перекодирования еще выше, чем в случае равных загрузок.

Графики, представленные на рисунках 3.48-3.50, построены по формулам (3.58) и (3.59). Для более точного расчета модели сети Клейнрока воспользуемся имитационной моделью. Результаты сравнения выигрыша от перекодирования, оцененного с помощью имитационной модели и по формуле (3.58), представлены на рисунке 3.51. На графиках видно, что имитационная модель также демонстрирует наличие выигрыша от перекодирования. Значение этого выигрыша ниже, чем рассчитанное по формуле (3.58), а разница в расчете и моделировании не велика при малых загрузках сети и увеличивается с ростом загрузки сети.

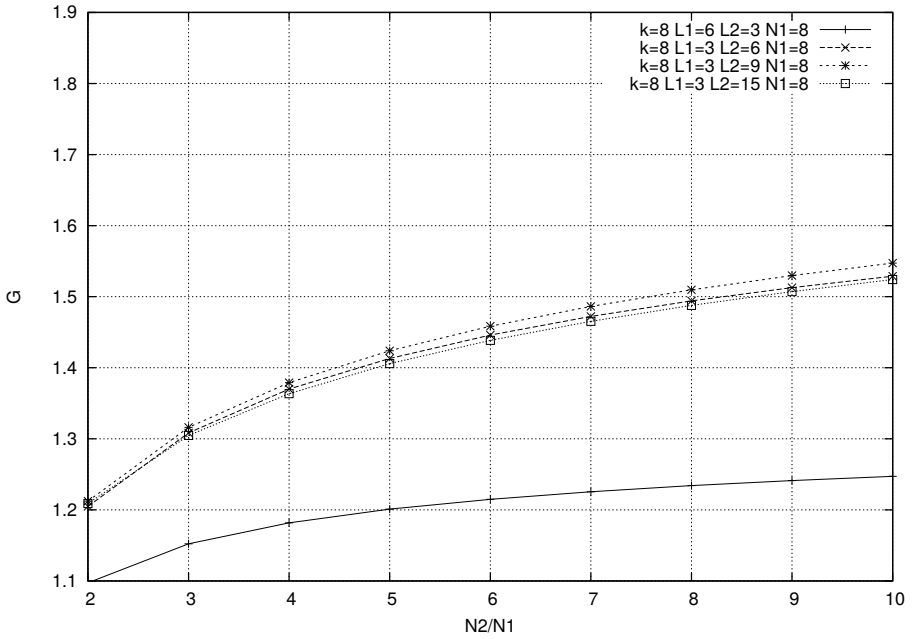


Рисунок 3.49 — Выигрыш от перекодирования для системы с параллельными каналами в зависимости от отношения количества путей в подсетях, случай различных длин путей

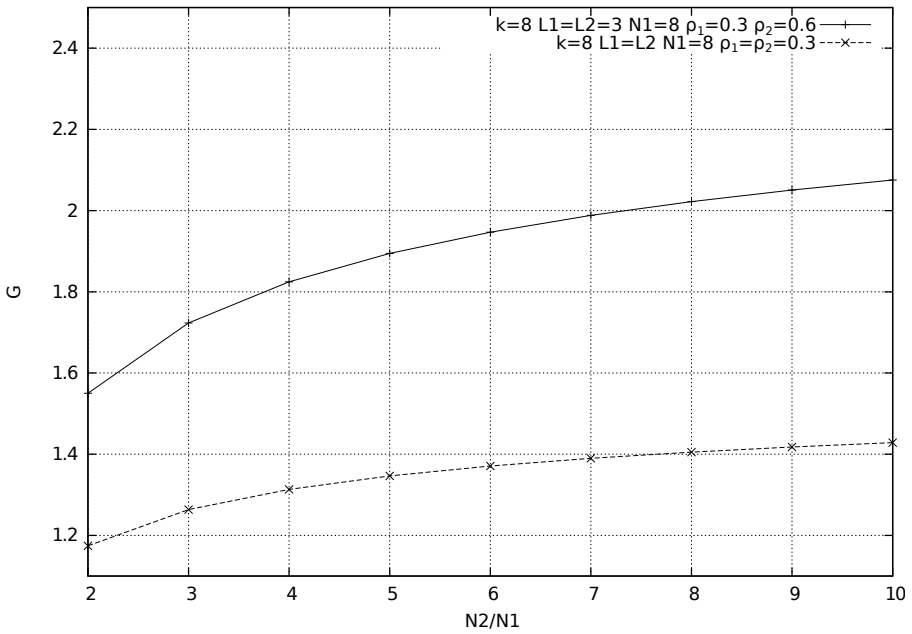


Рисунок 3.50 — Выигрыш от перекодирования для системы с параллельными каналами в зависимости от отношения количества путей в подсетях, случай равных длин путей и различных загрузок сети

3.12 Заключение и выводы по разделу

В данном разделе рассмотрена эффективность транспортного кодирования при различных эффектах сети, присутствующих в реальных сетях связи, которые

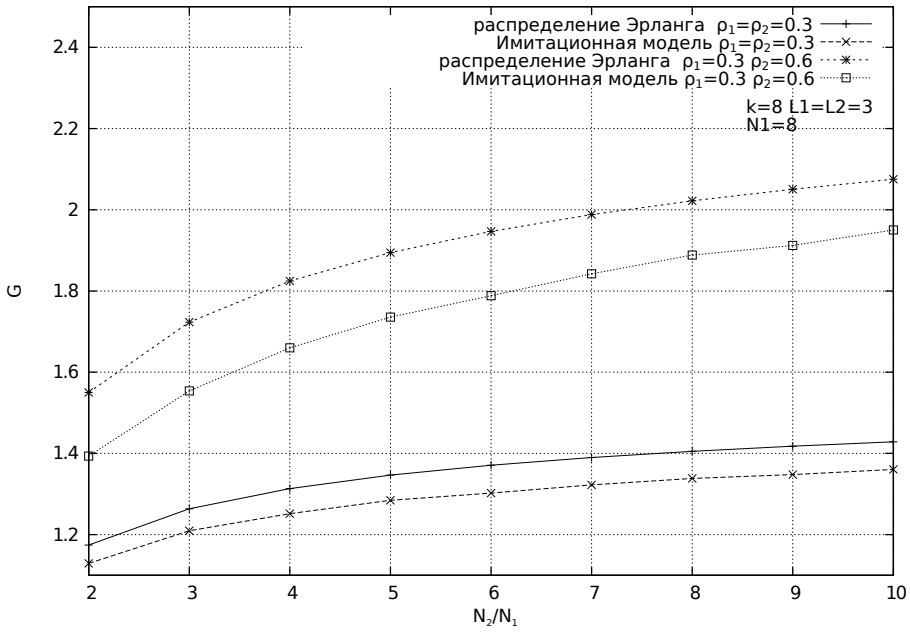


Рисунок 3.51 — Сравнение выигрыша от перекодирования, вычисленного аналитически и с помощью имитационной модели

ранее не рассматривались в рамках транспортного кодирования. Для этой цели были разработаны аналитические и имитационные модели сети путем эволюционных изменений известных ранее моделей.

Аналитические модели были получены путем замены распределения задержки пакетов в модели Кабатянского, Крука [88]. С помощью моделей проведен анализ выигрыша от кодирования, который позволил получить новые знания. Была получена зависимость выигрыша от кодирования от длины пути. Рассмотрен новый эффект уменьшения выигрыша от кодирования с ростом количества информационных символов k при нормальном распределении задержки. Совместный анализ распределения Эрланга и нормального привел к гипотезе о том, что увеличение выигрыша от кодирования, наблюдаемое с ростом k при распределении Эрланга, должно смениться убыванием при больших значениях k. Также на новой модели были построены известные ранее зависимости, а именно выигрыш от кодирования в зависимости от загрузки сети, количества информационных пакетов, скорости кода. Проведенный анализ показал, что выигрыш присутствует и в новой модели, однако значение выигрыша уменьшилось.

Имитационные модели были основаны на известной модели сети Клейнрока. Модель с параллельными каналами была получена упрощением модели Клейнрока и фиксированием части параметров. В модель сети с топологией типа решетка была добавлена многопутевая маршрутизация, при которой пакеты рас-

сылались по множеству кратчайших путей. Был сформулирован принцип выбора емкостей каналов. Для моделирования эффекта уменьшения емкости канала была введена модель с двумя состояниями. Введено ограничение на время жизни пакетов. Для данных моделей были построены зависимости, аналогичные тем, что получены для аналитических моделей. Результаты оказались согласованными друг с другом. Кроме этого, были получены новые зависимости. Был исследован выигрыш от кодирования при наличии следующих эффектов:

- зависимость маршрутов передачи,
- ограниченность количества маршрутов передачи,
- разные длины маршрутов,
- неравномерность информационных потоков в сети,
- ограниченное время жизни пакетов,
- уменьшение емкостей каналов.

Получены новые зависимости выигрыша от следующих параметров:

- интенсивность состояний, в которых уменьшается емкость каналов,
- длительность состояний, в которых уменьшается емкость каналов,
- величина проседания емкости каналов,
- время жизни пакета.

Кроме этого, получены зависимости вероятности доведения пакета и количества удаленных пакетов при ограничении на время жизни пакета. Полученные зависимости демонстрируют возможность эффективного применения транспортного кодирования при некоторых ограничениях.

С помощью аналитической и имитационной моделей было исследовано кодирование в нерегулярных сетях – сетях, состоящих из подсетей с регулярной структурой и различающимися параметрами. Был предложен новый метод кодирования на транспортном уровне с переменной скоростью кода. Для данного метода были получены зависимости для выигрыша от кодирования в зависимости от следующих параметров:

- различия в загрузках подсетей,
- различия в количестве маршрутов в подсетях,
- различия в длинах маршрутов в подсетях.

Полученные зависимости демонстрируют эффективность предложенного метода.

ЗАКЛЮЧЕНИЕ

В работе были рассмотрены вопросы, связанные с хранением и передачей данных в распределенных системах хранения. Основное внимание было уделено задержке системы. Рассмотрена задержка, возникающая на узле и в сети передачи информации.

В качестве узла распределенной системы рассматривались многоуровневые системы хранения. Была разработана модель такой системы. Предложен адаптивный алгоритм распределения памяти между приложениями на различных уровнях системы при ограничении на задержку системы. При разработке алгоритма были выбраны опорные характеристики потока запросов, на основе которых выполняется работа алгоритма. Этими характеристиками стали распределение стековых расстояний и распределение частот появления адресов. Для оценки эффективности алгоритма был предложен критерий, характеризующий соблюдение заданного требования на задержку. Оценка алгоритма производилась как на потоках запросов, собранных с реальных систем хранения, так и на синтетических потоках. Для генерации потока запросов была предложена агентная модель, позволяющая задавать распределение стековых расстояний и распределение частот появления адресов.

В качестве метода передачи информации по сети рассматривалось кодирование на транспортном уровне сети. Данный метод согласуется с текущей практикой использования кодов, исправляющих стирания, т.к. основан на их применении. Был исследован выигрыш от кодирования при различных эффектах, которые не рассматривались ранее совместно с транспортным кодированием, но которые присутствуют в существующих сетях связи. Для этого были разработаны аналитические и имитационные модели сети. Предложен метод транспортного кодирования с переменной скоростью кодирования для передачи в неравномерных сетях и показана его эффективность.

Основные результаты работы можно сформулировать следующим образом.

1. Разработан алгоритм расчета размеров областей памяти на каждом уровне системы хранения для каждого приложения, который позволяет при некоторых ограничениях обеспечить заданную для приложения

задержку системы. На рассмотренных сценариях была продемонстрирована эффективность использования алгоритма.

2. Предложенный алгоритм, кроме выполнения своей прямой задачи, мог бы использоваться для расчета общего объема уровней памяти для выполнения требований на задержку и предсказания граничных параметров, при которых система хранения перестанет удовлетворять требованиям на задержку.
3. Предложенный алгоритм имеет ограничения использования. Он основан на допущении, что в работе системы имеются периоды стабильности, в течение которых основные параметры входного потока не меняются. В случае быстрой изменчивости входного потока алгоритм не будет успевать адаптироваться. С другой стороны, применение алгоритма ограничено минимальными размерами уровней системы, при которых имеется возможность удовлетворять требованиям на задержку системы.
4. Рассмотренные сценарии работы многоуровневой системы хранения, основанные на потоках запросов с реальных систем хранения, продемонстрировали ситуации, когда предложенный адаптивный алгоритм отработал лучше, чем алгоритм, имеющий заранее информацию о потоке и использующий в работе средние характеристики, рассчитанные по всему потоку.
5. Разработаны модели сети с коммутацией пакетов, учитывающие такие эффекты как
    (a) зависимость маршрутов передачи,
    (b) ограниченность количества маршрутов передачи,
    (c) разные длины маршрутов,
    (d) неравномерность информационных потоков в сети,
    (e) ограниченное время жизни пакетов,
    (f) уменьшение емкостей каналов.
6. Разработанные модели позволили получить зависимости выигрыша от кодирования от следующих параметров:
    (a) длина маршрутов,
    (b) интенсивность появления состояний, в которых падает емкость каналов,
    (c) длительность состояний, в которых падает емкость каналов,

(d) величина проседания емкости каналов,

(e) время жизни пакета.

7. Полученные зависимости для выигрыша могут быть использованы при выборе метода передачи по сети, они позволяют оценить границы применимости транспортного кодирования.
8. Предложен метод кодирования на транспортном уровне с переменной скоростью кода. На разработанных аналитической и имитационной моделях была показана эффективность предложенного метода.

Возможные направления дальнейшей разработки темы.

1. В рассмотренном алгоритме перераспределения памяти каждому приложению выдяляется отдельный сегмент памяти. Представляется интересным вопрос группирования потоков различных приложений в один сегмент. Группирование позволило бы расширить границы применения алгоритма перераспределения памяти на большее количество приложений.
2. Исследование эффективности перераспределения памяти в зависимости от соотношения типов памяти и их общих размеров.
3. Исследование эффективности транспортного кодирования в мобильных сетях передачи информации.
4. Алгоритм поиска узлов, в которых целесообразно выполнять перекодирование при использовании предложенного алгоритма кодирования в нерегулярных сетях.

СПИСОК ЛИТЕРАТУРЫ

1. *Маличенко, Д.* Эвристический алгоритм расчета размеров памяти в многоуровневой системе хранения / Д. Маличенко // *Информационно-управляющие системы*. — 2015. — Т. 78, № 5.

2. *Маличенко, Д.А.* Кодирование сообщений на транспортном уровне в неравномерных сетях / Д.А. Маличенко // *Информационно-управляющие системы*. — 2014. — Т. 73, № 6.

3. *Крук, Е. А.* Расчет задержки при использовании кодирования на транспортном уровне сети передачи данных / Е. А. Крук, Д. А. Маличенко // *Известия высших учебных заведений. Приборостроение*. — 2013. — Т. 56, № 8.

4. *Malichenko, Dmitrii*. Efficiency of transport layer coding in networks with changing capacities / Dmitrii Malichenko // Problems of Redundancy in Information and Control Systems (REDUNDANCY), 2016 XV International Symposium / IEEE. — 2016. — Pp. 82–86.

5. *Malichenko, Dmitrii*. Estimation of the Mean Message Delay for Transport Coding / Dmitrii Malichenko, Evgenii Krouk // Intelligent Interactive Multimedia Systems and Services. — Springer, 2015. — Pp. 239–249.

6. *Malichenko, Dmitry*. Transport layer coding with variable coding rate / Dmitry Malichenko // Problems of Redundancy in Information and Control Systems (REDUNDANCY), 2014 XIV International Symposium on / IEEE. — 2014. — Pp. 71–73.

7. *Маличенко, Д.* Моделирование потока адресов в системе хранения данных / Д. Маличенко, Тяпочкин К. // Всероссийская научная конференция по проблемам информатики СПИСОК-2012. — 2012. — С. 502–507.

8. *Маличенко, Д.* К вопросу о выборе оптимальной скорости кода для кодирования на транспортном уровне / Д. Маличенко // Всероссийская научная конференция по проблемам информатики СПИСОК-2012. — 2012. — С. 211–216.

9. *Malichenko, D.* Optimization of Network Overhead for Transport Layer Coding / D. Malichenko // 9th Conference of Open Innovations Community FRUCT. — 2011.

10. *Маличенко, Д.* Введение времени жизни пакета при использовании транспортного кодирования / Д. Маличенко // *Вопросы передачи и защиты информации: сборник статей*. — 2011.

11. *Сомасундарам, Г.* От хранения данных к управлению информацией: [пер. с англ.] / Г. Сомасундарам, А. Шривастава. — Питер, 2010.

12. *Morenoff, E.* Application of level changing to a multilevel storage organization / E. Morenoff, J. B. McLean // *Communications of the ACM*. — 1967. — Vol. 10, no. 3. — Pp. 149–154.

13. *Ver Hoef, E. W.* Design of a multi-level file management system / E. W. Ver Hoef // Proceedings of the 1966 21st national conference / ACM. — 1966. — Pp. 75–86.

14. *Liu, X.* Hybrid storage management for database systems / X. Liu, K. Salem // *Proceedings of the VLDB Endowment*. — 2013. — Vol. 6, no. 8. — Pp. 541–552.

15. An object placement advisor for DB2 using solid state storage / M. Canim, G. A. Mihaila, B. Bhattacharjee et al. // *Proceedings of the VLDB Endowment*. — 2009. — Vol. 2, no. 2. — Pp. 1318–1329.

16. *Taneja Group Technology Analysts*. The State of the Core – Engineering the Enterprise Storage Infrastructure with the IBM DS8000. White Paper. — 2010.

17. Efficient QoS for Multi-Tiered Storage Systems. / A. Elnably, H. Wang, A. Gulati, P. J. Varman // HotStorage. — 2012.

18. A lightweight I/O scheme to facilitate spatial and temporal queries of scientific data analytics / Y. Tian, Z. Liu, S. Klasky et al. // Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on / IEEE. — 2013. — Pp. 1–10.

19. Improving I/O performance using soft-QoS-based dynamic storage cache partitioning / C. M Patrick, R. Garg, S. W. Son, M. Kandemir // Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on / IEEE. — 2009. — Pp. 1–10.

20. CacheCOW: QoS for storage system caches / P. Goyal, D. Jadav, D. S. Modha, R. Tewari // International Workshop on Quality of Service / Springer. — 2003. — Pp. 498–515.

21. *Дужин, В. С.* Алгоритм перераспределения кэша между приложениями в системе хранения данных / В. С. Дужин // Научная сессия ГУАП: сб. докл.: в 3 ч. — Т. I. Технические науки. — СПб.: ГУАП, 2012. — С. 73–75.

22. *Hill, M. D.* Cache considerations for multiprocessor programmers / M. D. Hill, J. R. Larus // *Communications of the ACM*. — 1990. — Vol. 33, no. 8. — Pp. 97–102.

23. *Lam, M. D.* The Cache Performance and Optimizations of Blocked Algorithms / M. D. Lam, E. E. Rothberg, M. E. Wolf // *SIGPLAN Not.* — 1991. — Vol. 26, no. 4. — Pp. 63–74. http://doi.acm.org/10.1145/106973.106981.

24. *Lai, P.* Makespan-Optimal Cache Partitioning / P. Lai, R. Fan // Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on / IEEE. — 2013. — Pp. 202–211.

25. *Yuan L.and Zhang, Y.* A Locality-based Performance Model for Load-and-Compute Style Computation / Y. Yuan, L.and Zhang // Cluster Computing (CLUSTER), 2012 IEEE International Conference on / IEEE. — 2012. — Pp. 566–571.

26. *Huang, L.* Stonehenge: A High-performance Virtualized Ip Storage Cluster with Qos Guarantees: Ph.D. thesis / State University of New York at Stony Brook. — Stony Brook, NY, USA, 2003. — AAI3102796.

27. *Gang, P.* Availability and fairness support for storage QoS guarantee / P. Gang, T. Chiueh // Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on / IEEE. — 2008. — Pp. 589–596.

28. IOFlow: a software-defined storage architecture / E. Thereska, H. Ballani, G. O'Shea et al. // Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles / ACM. — 2013. — Pp. 182–196.

29. Storage challenges at los alamos national lab / J. Bent, G. Grider, B. Kettering et al. // Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on / IEEE. — 2012. — Pp. 1–5.

30. A technique for moving large data sets over high-performance long distance networks / B. W. Settlemyer, J. D. Dobson, S. W. Hodson et al. // Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on / IEEE. — 2011. — Pp. 1–6.

31. Cost Effective Storage using Extent Based Dynamic Tiering. / J. Guerra, H. Pucha, J. S. Glider et al. // FAST. — Vol. 11. — 2011. — Pp. 20–20.

32. A new high-performance, energy-efficient replication storage system with reliability guarantee / J. Wan, C. Yin, J. Wang, C. Xie // Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on / IEEE. — 2012. — Pp. 1–6.

33. An adaptive partitioning scheme for DRAM-based cache in solid state drives / H. Shim, B. Seo, J. Kim, S. Maeng // Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on / IEEE. — 2010. — Pp. 1–12.

34. *Fan, Z.* H-ARC: A non-volatile memory based cache policy for solid state drives / Z. Fan, D. HC Du, D. Voigt // Mass Storage Systems and Technologies (MSST), 2014 30th Symposium on / IEEE. — 2014. — Pp. 1–11.

35. *Chuang, J. C.* Distributed network storage service with quality-of-service guarantees / J. C. Chuang, M. A. Sirbu // *Journal of Network and Computer Applications*. — 2000. — Vol. 23, no. 3. — Pp. 163–185.

36. *Chen, F.* Client-aware cloud storage / F. Chen, M. P. Mesnier, S. Hahn // Mass Storage Systems and Technologies (MSST), 2014 30th Symposium on / IEEE. — 2014. — Pp. 1–12.

37. *Chow, C. K.* Determination of cache's capacity and its matching storage hierarchy / C. K. Chow // *IEEE Transactions on Computers*. — 1976. — Vol. 100, no. 2. — Pp. 157–164.

38. *Joshi, G.* Efficient Redundancy Techniques to Reduce Delay in Cloud Systems: Ph.D. thesis / Massachusetts Institute of Technology. — 2016.

39. *Bunt, R.* A measure of program locality and its application / R. Bunt, J. M. Murphy, S. Majumdar // *ACM SIGMETRICS Performance Evaluation Review*. — 1984. — Vol. 12, no. 3. — Pp. 28–40.

40. *Citron, D.* Creating a wider bus using caching techniques / D. Citron, L. Rudolph // High-Performance Computer Architecture, 1995. Proceedings., First IEEE Symposium on / IEEE. — 1995. — Pp. 90–99.

41. Locality as a visualization tool / K. Grimsrud, J. Archibald, R. Frost, B. Nelson // *IEEE transactions on computers*. — 1996. — Vol. 45, no. 11. — Pp. 1319–1326.

42. *McKinley, K. S.* Quantifying loop nest locality using SPEC'95 and the perfect benchmarks / K. S. McKinley, O. Temam // *ACM Transactions on Computer Systems (TOCS)*. — 1999. — Vol. 17, no. 4. — Pp. 288–336.

43. *Anderson, J. M.* Global Optimizations for Parallelism and Locality on Scalable Parallel Machines / J. M. Anderson, M. S. Lam // *SIGPLAN Not.* — 1993. — Vol. 28, no. 6. — Pp. 112–125. http://doi.acm.org/10.1145/173262.155101.

44. Understanding and improving computational science storage access through continuous characterization / P. Carns, K. Harms, W. Allcock et al. // *ACM Transactions on Storage (TOS)*. — 2011. — Vol. 7, no. 3. — P. 8.

45. *Feitelson, D. G.* Workload modeling for computer systems performance evaluation / D. G. Feitelson. — Cambridge University Press, 2015.

46. UMass Trace Repository. — http://traces.cs.umass.edu/index.php/Storage/Storage. — дата обращения: 19.12.2016.

47. *Кабатянский, Г. А.* Кодирование уменьшает задержку / Г. А. Кабатянский, Е. А. Крук // *X Всесоюз. школа-семинар по вычислительным сетям.* — 1985. — Т. 2. — С. 23–26.

48. *Кабатянский, Г. А.* Об избыточном кодировании на транспортном уровне сети передачи данных / Г. А. Кабатянский, Е. А. Крук // *Помехоустойчивое кодирование и надежность ЭВМ. М.: Наука*. — 1987. — С. 143–150.

49. *Maxemchuk, N. F.* Dispersity routing / N. F. Maxemchuk // Proceedings of ICC. — Vol. 75. — 1975. — Pp. 41–10.

50. *Maxemchuk, N. F.* Dispersity routing: Past and present / N. F. Maxemchuk // Military Communications Conference, 2007. MILCOM 2007. IEEE / IEEE. — 2007. — Pp. 1–7.

51. *Maxemchuk, N. F.* Dispersity routing in high-speed networks / N. F. Maxemchuk // *computer networks and ISDN systems*. — 1993. — Vol. 25, no. 6. — Pp. 645–661.

52. *Maxemchuk, N. F.* Dispersity routing on ATM networks / N. F. Maxemchuk // INFOCOM'93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future, IEEE / IEEE. — 1993. — Pp. 347–357.

53. *Banerjea, A.* Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels / A. Banerjea // ACM SIGCOMM Computer Communication Review / ACM. — Vol. 26. — 1996. — Pp. 194–205.

54. *Joshi, G.* On the delay-storage trade-off in content download from coded distributed storage systems / G. Joshi, Y. Liu, E. Soljanin // *IEEE Journal on Selected Areas in Communications*. — 2014. — Vol. 32, no. 5. — Pp. 989–997.

55. *Блейхут, Р.* Теория и практика кодов, исправляющих ошибки / Р. Блейхут. — М.: Мир, 1982.

56. *Alon, N.* Linear time erasure codes with nearly optimal recovery / N. Alon, J. Edmonds, M. Luby // Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on / IEEE. — 1995. — Pp. 512–519.

57. Practical loss-resilient codes / M. Luby, M. Mitzenmacher, M. Shokrollahi et al. // Proceedings of the twenty-ninth annual ACM symposium on Theory of computing / ACM. — 1997. — Pp. 150–159.

58. *Luby, M.* LT Codes / M. Luby // Proceedings of the 43rd Symposium on Foundations of Computer Science. — FOCS ’02. — Washington, DC, USA: IEEE Computer Society, 2002. http://dl.acm.org/citation.cfm?id=645413.652135.

59. *Etesami, O.* Raptor codes on symmetric channels / O. Etesami, M. Molkaraie, A. Shokrollahi // Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on / IEEE. — 2004. — P. 38.

60. *Shokrollahi, A.* Raptor codes / A. Shokrollahi // *IEEE transactions on information theory*. — 2006. — Vol. 52, no. 6. — Pp. 2551–2567.

61. *Gerami, M.* Optimal-cost repair in multi-hop distributed storage systems / M. Gerami, M. Xiao, M. Skoglund // Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on / IEEE. — 2011. — Pp. 1437–1441.

62. *Gerami, M.* Repair for distributed storage systems with erasure channels / M. Gerami, M. Xiao // Communications (ICC), 2013 IEEE International Conference on / IEEE. — 2013. — Pp. 4058–4062.

63. *Hollmann, H.* On the minimum storage overhead of distributed storage codes with a given repair locality / H. Hollmann // Information Theory (ISIT), 2014 IEEE International Symposium on / IEEE. — 2014. — Pp. 1041–1045.

64. *Rashmi, K. V.* Enabling node repair in any erasure code for distributed storage / K. V. Rashmi, N. B. Shah, P. V. Kumar // Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on / IEEE. — 2011. — Pp. 1235–1239.

65. *Barg, A.* Locally recoverable codes on algebraic curves / A. Barg, I. Tamo, S. Vlăduț // Information Theory (ISIT), 2015 IEEE International Symposium on / IEEE. — 2015. — Pp. 1252–1256.

66. Erasure Coding in Windows Azure Storage / C. Huang, H. Simitci, Y. Xu et al. // Usenix annual technical conference / Boston, MA. — 2012. — Pp. 15–26.

67. Space-time storage codes for wireless distributed storage systems / C. Hollanti, D. Karpuk, A. Barreal, H. F. Lu // Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), 2014 4th International Conference on / IEEE. — 2014. — Pp. 1–5.

68. Polynomial length MDS codes with optimal repair in distributed storage / V. R. Cadambe, C. Huang, J. Li, S. Mehrotra // Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on / IEEE. — 2011. — Pp. 1850–1854.

69. Asymptotic interference alignment for optimal repair of MDS codes in distributed storage / V. R. Cadambe, S. A. Jafar, H. Maleki et al. // *IEEE Transactions on Information Theory*. — 2013. — Vol. 59, no. 5. — Pp. 2974–2987.

70. Codes with local regeneration / G. M Kamath, N. Prakash, V. Lalitha, P. V. Kumar // Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on / IEEE. — 2013. — Pp. 1606–1610.

71. Locality and availability in distributed storage / A. S. Rawat, D. S. Papailiopoulos, A. G. Dimakis, S. Vishwanath // *IEEE Transactions on Information Theory*. — 2016. — Vol. 62, no. 8. — Pp. 4481–4493.

72. *Altman, E.* Distributed storage in the plane / E. Altman, K. Avrachenkov, J. Goseling // Networking Conference, 2014 IFIP / IEEE. — 2014. — Pp. 1–9.

73. *Golrezaei, N.* Device-to-device collaboration through distributed storage / N. Golrezaei, A. G. Dimakis, A. F. Molisch // Global Communications Conference (GLOBECOM), 2012 IEEE / IEEE. — 2012. — Pp. 2397–2402.

74. *Chan, S. H. G.* Distributed storage to support user interactivity in peer-to-peer video streaming. — 2011. — US Patent 7,925,781.

75. *Milne, R. B.* System and method for distributing and accessing files in a distributed storage system. — 2014. — US Patent 8,768,981.

76. *Wang, Z.* On multi-version coding for distributed storage / Z. Wang, V. Cadambe // Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on / IEEE. — 2014. — Pp. 569–575.

77. Network information flow / R. Ahlswede, N. Cai, S-YR Li, R. W. Yeung // *IEEE Transactions on information theory*. — 2000. — Vol. 46, no. 4. — Pp. 1204–1216.

78. *Габидулин, Э. М.* Теория кодов с максимальным ранговым расстоянием / Э. М. Габидулин // *Проблемы передачи информации*. — 1985. — Т. 21, № 1. — С. 3–16.

79. Сетевое кодирование / Э. М. Габидулин, Н. И. Пилипчук, А. И. Колыбельников и др. // *Труды МФТИ*. — 2009. — Т. 1, № 2. — С. 3–28.

80. *Габидулин, Э. М.* Алгебраические коды для сетевого кодирования / Э. М. Габидулин, М. Боссерт // *Проблемы передачи информации*. — 2009. — Т. 45, № 4. — С. 54–68.

81. *Габидулин, Э. М.* Декодирование случайных сетевых кодов / Э. М. Габидулин, Н. И. Пилипчук, М. Боссерт // *Проблемы передачи информации*. — 2010. — Т. 46, № 4. — С. 33–55.

82. *Габидулин, Э. М.* Ранговые подкоды в многокомпонентном сетевом кодировании / Э. М. Габидулин, Н. И. Пилипчук // *Проблемы передачи информации*. — 2013. — Т. 49, № 1. — С. 46–60.

83. *Габидулин, Э. М.* Эффективность подпространственных сетевых кодов / Э. М. Габидулин, Н. И. Пилипчук // *Труды МФТИ*. — 2015. — Т. 7, № 1. — С. 104–111.

84. Network coding for distributed storage systems / A. G. Dimakis, P. B. Godfrey, Y. Wu et al. // *IEEE Transactions on Information Theory*. — 2010. — Vol. 56, no. 9. — Pp. 4539–4551.

85. A survey on network codes for distributed storage / A. G. Dimakis, K. Ramchandran, Y. Wu, C. Suh // *Proceedings of the IEEE*. — 2011. — Vol. 99, no. 3. — Pp. 476–489.

86. A survey on network codes for distributed storage / A. G. Dimakis, K. Ramchandran, Y. Wu, C. Suh // *Proceedings of the IEEE*. — 2011. — Vol. 99, no. 3. — Pp. 476–489.

87. Optimal-cost repair in multi-hop distributed storage systems with network coding / M. Gerami, M. Xiao, M. Skoglund et al. // *Transactions on Emerging Telecommunications Technologies*. — 2016. — Vol. 27, no. 11. — Pp. 1539–1549.

88. *Крук, Е. А.* Комбинаторное декодирование линейных блоковых кодов / Е. А. Крук. — ГУАП СПб., 2007.

89. *Krouk, E.* Application of coding at the network transport level to decrease the message delay / E. Krouk, S. Semenov // Proceedings of Third International Symposium on Communication Systems Networks and Digital Signal Processing. — 2002. — Pp. 109–112.

90. *Krouk, E.* Transmission of priority messages with the help of transport coding / E. Krouk, S. Semenov // Telecommunications, 2003. ICT 2003. 10th International Conference on / IEEE. — Vol. 2. — 2003. — Pp. 1273–1277.

91. *Krouk, E.* Transmission of a message during limited time with the help of transport coding / E. Krouk, S. Semenov // ICETE. — 2005. — Pp. 88–93.

92. *Kabatiansky, G.* Error correcting coding and security for data networks: analysis of the superchannel concept / G. Kabatiansky, E. Krouk, S. Semenov. — John Wiley & Sons, 2005.

93. *Krouk, E.* Application of Tornado Codes to Transport Coding / E. Krouk, S. Semenov // Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on / IEEE. — 2007. — Pp. 249–256.

94. Punctured Reed–Solomon codes at the transport layer of digital networks / V. Sidorenko, F. Shen, E. Krouk, M. Bossert // *Proc. Coding Theory Days in St. Petersburg*. — 2008.

95. *Vvedenskaya, N. D.* Large queueing system where messages are transmitted via several routes / N. D. Vvedenskaya // *Problemy Peredachi Informatsii*. — 1998. — Vol. 34, no. 2. — Pp. 98–108.

96. *Ernst, H.* Transport layer coding for the land mobile satellite channel / H. Ernst, L. Sartorello, S. Scalise // Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th / IEEE. — Vol. 5. — 2004. — Pp. 2916–2920.

97. Transport Layer Coding for Satellite-Based Audio and Multimedia Services to Vehicular Terminals in Ku-band / M. Berioli, H. Ernst, S. Scalise et al. // Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE / IEEE. — 2008. — Pp. 2907–2911.

98. *Lam, K.* Performance analysis and optimization of multipath TCP / K. Lam, J. M. Chapin, V. Chan // Wireless Communications and Networking Conference (WCNC), 2011 IEEE / IEEE. — 2011. — Pp. 695–700.

99. *Mitzenmacher, M.* Digital fountains: A survey and look forward / M. Mitzenmacher // Information Theory Workshop, 2004. IEEE / IEEE. — 2004. — Pp. 271–276.

100. Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks / J. Wu, C. Yuen, B. Cheng et al. // *IEEE Transactions on Mobile Computing*. — 2016. — Vol. 15, no. 9. — Pp. 2345–2361.

101. *Клейнрок, Л.* Вычислительные системы с очередями / Л. Клейнрок. — Мир, 1979.

102. *Neely, M. J.* Capacity and delay tradeoffs for ad hoc mobile networks / M. J. Neely, E. Modiano // *IEEE Transactions on Information Theory*. — 2005. — Vol. 51, no. 6. — Pp. 1917–1937.

103. *Дэйвид, Г.* Порядковые статистики / Г. Дэйвид. — Наука, 1979.

104. *Градштейн, И. С.* Таблицы интегралов, сумм, рядов и произведений / И. С. Градштейн, И. М. Рыжик. — Государственное издательство физико-математической литературы, 1963.

105. *Вентцель, Е. С.* Теория вероятностей и ее инженерные приложения / Е. С. Вентцель, Л. А. Овчаров. — Москва, ”Высшая школа”, 2000.

106. *Gentle, J. E.* Random Number Generation and Monte Carlo Methods / J. E. Gentle. Statistics and Computing. — Springer New York, 2013.

107. *Adan, I.* Queueing theory / I. Adan, J. Resing. — Eindhoven University of Technology Eindhoven, 2015.

ПРИЛОЖЕНИЕ А

АКТЫ О ВНЕДРЕНИИ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский государственный университет аэрокосмического приборостроения»
(ГУАП)

ул. Большая Морская, д. 67, лит. А, Санкт-Петербург, 190000, Тел. (812) 710-6510, факс (812) 494-7057,
E-mail: common@aanet.ru ОГРН 1027810232680, ИНН/КПП 7812003110/783801001

_____ № _____ На № _____ от _____

УТВЕРЖДАЮ

Ректор,
д-р экон. наук, доцент

_____ Ю. А. Антохина
«24» октября 2017 г.

АКТ

об использовании результатов диссертационной работы Маличенко Дмитрия Александровича на тему «Разработка и исследование методов хранения и передачи информации в распределенных системах»

Комиссия в составе: д.т.н., профессор Крук Е.А., д.т.н., доцент Тюрликов А.М., к.т.н., доцент Овчинников А.А. составила настоящий акт о том, что в рамках НИР №023-2Д от 24.01.2012г. и Программы научно-исследовательских и опытно-конструкторских работ №1 от 25.01.2012г. «Разработка алгоритма, который динамически перераспределяет пространство в кэше и на разных носителях между приложениями для того, чтобы выполнить требования по производительности этих приложений» были использованы следующие результаты диссертационной работы Маличенко Д.А.:

- модель потока запросов к системе хранения, которая параметризуется распределением стековых расстояний и распределением частот появления адресов;
- алгоритм распределения памяти между приложениями, удовлетворяющий требованиям на задержку приложения, который адаптируется к изменяемым характеристикам входного потока запросов.

Руководитель НИР № 023-2Д,
доктор технических наук, профессор _____ Е.А. Крук

Заведующий кафедрой
инфокоммуникационных систем,
доктор технических наук, доцент _____ А.М. Тюрликов

Заместитель заведующего кафедрой
безопасности информационных систем,
кандидат технических наук, доцент _____ А.А. Овчинников

ГУАП

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский государственный университет аэрокосмического приборостроения»
(ГУАП)

ул. Большая Морская, д. 67, лит. А, Санкт-Петербург, 190000, Тел. (812) 710-6510, факс (812) 494-7057,
E-mail: common@aanet.ru ОГРН 1027810232680, ИНН/КПП 7812003110/783801001

_____ № _____ На № _____ от _____

УТВЕРЖДАЮ
Ректор ФГАОУ ВО «Санкт-Петербургский государственный университет аэрокосмического приборостроения»
д-р экон. наук, доцент
_____ Ю. А. Антохина
«02» февраля 2017 г.

АКТ

об использовании результатов диссертационной работы Маличенко Дмитрия Александровича «Разработка и исследование методов хранения и передачи информации в распределенных системах».

Научно-техническая комиссия в составе: Е.А. Крук – директор института информационных систем и защиты информации; А.М. Тюрликов – заведующий кафедрой инфокоммуникационных систем; Т.М. Татарникова – профессор кафедры безопасности информационных систем, составила настоящий акт о том, что следующие результаты диссертационной работы:

- модели сети с коммутацией пакетов, учитывающие неэкспоненциальный характер задержки пакетов, изменения емкостей каналов сети и ограниченного времени жизни пакетов;
- анализ эффективности транспортного кодирования при различных распределениях задержки пакетов, а также при условиях, возникающих в реальных сетях связи, таких как изменяемые со временем емкости каналов и ограниченное время жизни пакетов;
- модифицированный алгоритм транспортного кодирования, который позволяет повысить его эффективность в сетях с нерегулярной структурой;

используются при проведении лекционных и лабораторных занятий учебных курсов «Основы построения инфокоммуникационных систем и сетей» и «Моделирование ИС», читаемых студентам кафедры безопасности информационных систем.

Директор института информационных систем и защиты информации
доктор технических наук, профессор _____ Е.А. Крук

Заведующий кафедрой инфокоммуникационных систем
доктор технических наук, доцент _____ А.М. Тюрликов

Профессор кафедры безопасности информационных систем
доктор технических наук, профессор _____ Т.М. Татарникова

Институт
Инфокоммуникационных
Технологий

Закрытое акционерное общество
«Институт инфокоммуникационных технологий»
195197, г. Санкт-Петербург, ул . Минеральная д. 13, литер К
ИНН 7804362290, КПП 780401001
ОГРН 1077847382238 ОКПО 80494677

19.01.17 № б/н

На №_____ от_____

УТВЕРЖДАЮ:
директор по развитию
_____ В.Б. Прохорова
«19» января 2017 г.

АКТ

об использовании результатов кандидатской диссертации Маличенко Д.А. на тему «Разработка и исследование методов хранения и передачи информации в распределенных системах»

Настоящий акт составлен о том, что в рамках НИР № 008/14-РД от 30.05.2014 г. теме: «Кодирование данных в распределенных СХД» были использованы следующие результаты диссертационной работы Маличенко Д.А.:

- Методика анализа распределения памяти в RAID-системах хранения информации.

Руководитель проектов

к.т.н. Е.М. Линский