МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Математические и компьютерные модели симуляции распределенных P2P-систем

Студент Э.Т. Кадирова _____
(И.О. Фамилия) (личная подпись)
Руководитель А.В. Очеповский _____
(И.О. Фамилия) (личная подпись)
Консультанты Н.В. Андрюхина _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

Выпускная квалификационная работа посвящена исследованию методов математического и компьютерного моделирования симуляций распределенных P2P-систем.

Целью бакалаврской работы являются анализ и практическое применение методов математического и компьютерного моделирования распределенных P2P-систем.

Объект исследования бакалаврской работы являются распределенные P2P-системы.

Предмет исследования бакалаврской работы – математические и компьютерные модели симуляций распределенных P2P-систем.

В первой главе представлен анализ методов математического моделирования распределенных P2P-систем.

Вторая глава посвящена компьютерному моделированию распределенных P2P-систем.

В третьей главе представлены примеры использования современных P2P-симуляторов для создания имитационных моделей распределенных P2P-систем.

Результаты бакалаврской работы могут быть рекомендованы для решения задач математического и компьютерного моделирования распределенных P2P-систем.

Бакалаврская работа включает 34 рисунка, 3 таблицы, 25 источников. Общий объем работы – 48 с.

ABSTRACT

This bachelor’s thesis is devoted to the study of methods of mathematical and computer simulation models of distributed P2P-systems.

The goal of this bachelor’s thesis are analysis and practical application of the methods of mathematical and computer modeling of distributed P2P systems.

The objects of the study are distributed P2P systems.

The subjects of the study are mathematical and computer simulation models of distributed P2P systems.

The first chapter of the bachelor’s thesis presents an analysis of the methods of mathematical modeling of distributed P2P-systems.

The second chapter is devoted to computer modeling of distributed P2P-systems.

The third chapter presents examples of utilizing modern P2P simulators for creating simulation models of distributed P2P systems.

The results of bachelor's thesis can be recommended for solving problems of mathematical and computer simulation of distributed P2P systems.

The bachelor’s thesis contains 34 figures, 3 tables and 25 references. The whole volume of the work is 48 pages.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Одноранговые «Peer-to-Peer» (P2P) системы являются формой распределенных вычислений, направленных на самоорганизацию, децентрализацию и автономию участвующих узлов (пиров).

Самоорганизация, автономность и децентрализация позволяют создавать высоко адаптивные, надежные и масштабируемые сети, что делает P2P все более интересным способом проектирования распределенных систем.

Однако, как показывает практика, при разработке приложения распределенной P2P-системы могут возникнуть проблемы из-за сложного и плохого понимания взаимозависимостей между пользователями, приложением, протоколом и физической сетью.

Кроме того, собственная топология P2P и базовая топология сети могут значительно влиять на поведение P2P-системы.

Для решения данной проблемы необходимо на начальном этапе проектирования исследовать особенности конкретной P2P-системы с помощью математической и компьютерной моделей ее симуляции [7].

Таким образом**, актуальность** бакалаврской работы обусловлена необходимостью математического и компьютерного моделирования распределенных P2P-систем.

Объектом исследования бакалаврской работы являются распределенные P2P-системы.

Предмет исследования бакалаврской работы – математические и компьютерные модели симуляций распределенных P2P-систем.

Целью бакалаврской работы является анализ и практическое применение методов математического и компьютерного моделирования распределенных P2P-систем.

Для достижения поставленной цели необходимо решить следующие задачи:

− проанализировать научную и учебно-методическую литературу по теме бакалаврской работы;

− проанализировать математические методы моделирования распределенных P2P-систем;

− проанализировать методы компьютерного моделирования распределенных P2P-систем;

− произвести анализ и представить примеры применения программных средств имитационного моделирования распределенных P2P-систем.

Методы исследования: методы математического и компьютерного моделирования распределенных P2P-систем.

Практическая значимость бакалаврской работы заключается в практическом применении программных средств имитационного моделирования распределенных P2P-систем.

В первой главе представлен анализ методов математического моделирования распределенных P2P-систем.

Вторая глава посвящена компьютерному моделированию распределенных P2P-систем.

В третьей главе представлены примеры применения современных P2P-симуляторов для создания имитационных моделей распределенных P2P-систем.

В заключении подводятся итоги исследования, формируются окончательные выводы по изучаемой тематике.

Бакалаврская работа состоит из 48 страниц и включает 34 рисунка, 3 таблицы, 25 источников.

Глава 1 МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕННЫХ P2P-СИСТЕМ

1.1 Архитектура P2P-систем

P2P (peer-to-peer) - вид сети, в которой компьютеры взаимодействуют друг с другом напрямую, а не через управляемые центральные серверы и сети [14].

Термин P2P также относится к системам и приложениям, которые функционируют децентрализовано.

Одноранговые (P2P) вычисления или сети - это распределенная прикладная архитектура, которая распределяет задачи или рабочие нагрузки между узлами (пирами) [8] (рисунок 1.1).

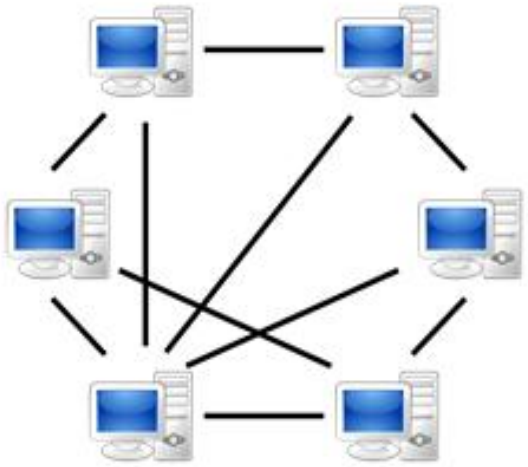


Рисунок 1.1 – Модель простой P2P-сети

Пиры - это равноправные участники приложения, образующие P2P-сеть узлов.

P2P-узлы предоставляют часть своих ресурсов, таких как вычислительная мощность, дисковое хранилище или пропускная способность сети в доступ другим участникам сети, без необходимости центральной координации с помощью серверов или стабильных хостов.

Пиры являются как поставщиками, так и потребителями ресурсов, в отличие от традиционной модели клиент-сервер, в которой потребление и предложение ресурсов разделены.

К основным преимуществам подхода P2P относятся:

− улучшение масштабируемости за счет устранения зависимости от централизованных точек;

− устранение потребности в дорогостоящей ИТ-инфраструктуре путем обеспечения прямой связи между клиентами;

− возможность агрегации ресурсов.

Более полная оценка достоинств и недостатков P2P-сетей предполагает сравнение с архитектурой клиент / сервер.

P2P-сети имеют клиентов с такими ресурсами, как пропускная способность, объем памяти и вычислительная мощность. Поскольку к каждому узлу предъявляется повышенный спрос, емкость всей системы увеличивается. Это объясняет существенное повышение безопасности системы и механизмов проверки файлов, что делает большинство сетей P2P устойчивыми практически к любому типу атак.

Для сравнения, типичная сеть клиент / сервер разделяет потребности, но не ресурсы. По мере того, как к системе присоединяются дополнительные клиенты, каждому из них становится доступно меньше ресурсов.

P2P-сети, как правило, реализуют некоторую форму виртуальной оверлейной сети поверх топологии физической сети, где узлы в оверлее формируют подмножество узлов в физической сети.

Обмен данными осуществляется по базовому протоколу TCP/IP, но на прикладном уровне узлы могут обмениваться данными друг с другом напрямую через логические оверлейные каналы (каждый из которых соответствует пути через базовую физическую сеть) [1].

Наложения используются для индексации и обнаружения одноранговых узлов и делают систему P2P независимой от топологии физической сети.

Основываясь на том, как узлы связаны друг с другом в оверлейной сети, и как ресурсы индексируются и располагаются, P2P-сети классифицируются как неструктурированные или структурированные (или как гибридная форма).

Неструктурированные P2P-сети не навязывают конкретную структуру наложенной сети, а скорее формируются узлами, которые случайным образом формируют соединения друг с другом (рисунок 1.2).

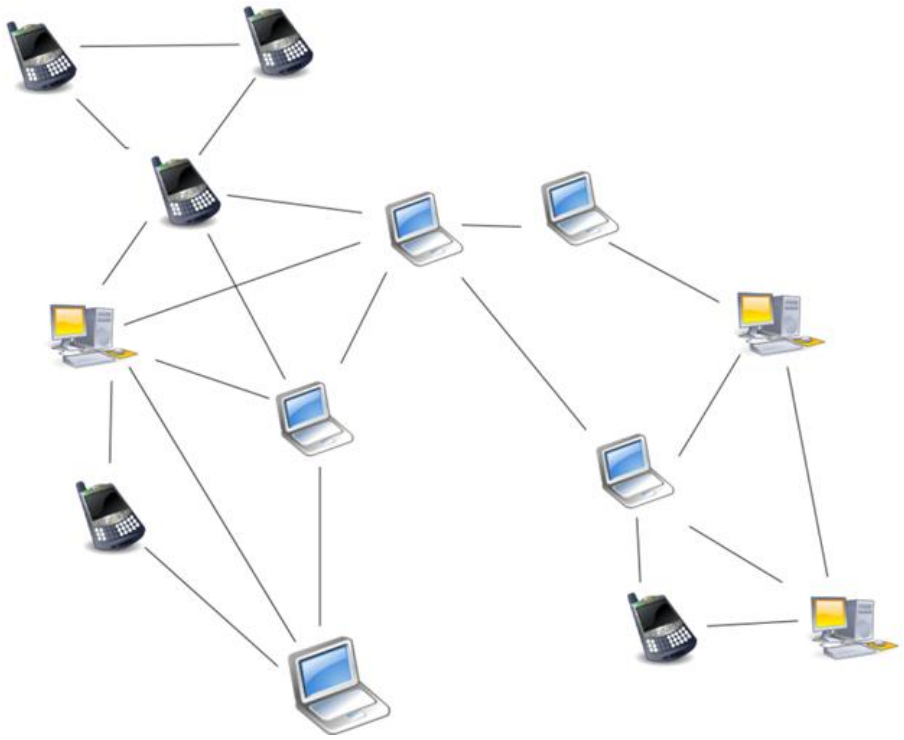


Рисунок 1.2 – Архитектура неструктурированной P2P-сети

К неструктурированным P2P-сетям относятся Gnutella, Gossip, KazaA, BitTorrent.

Существует три модели неструктурированной архитектуры компьютерной сети P2P:

- чистая P2P-сеть;
- гибридная P2P-сеть;
- централизованный P2P-сеть.

В структурированных P2P-сетях оверлей организован в рамках специфической топологии, при этом протокол гарантирует, что любой узел

может эффективно искать в сети файл / ресурс, даже если ресурс крайне редок (рисунок 1.3).

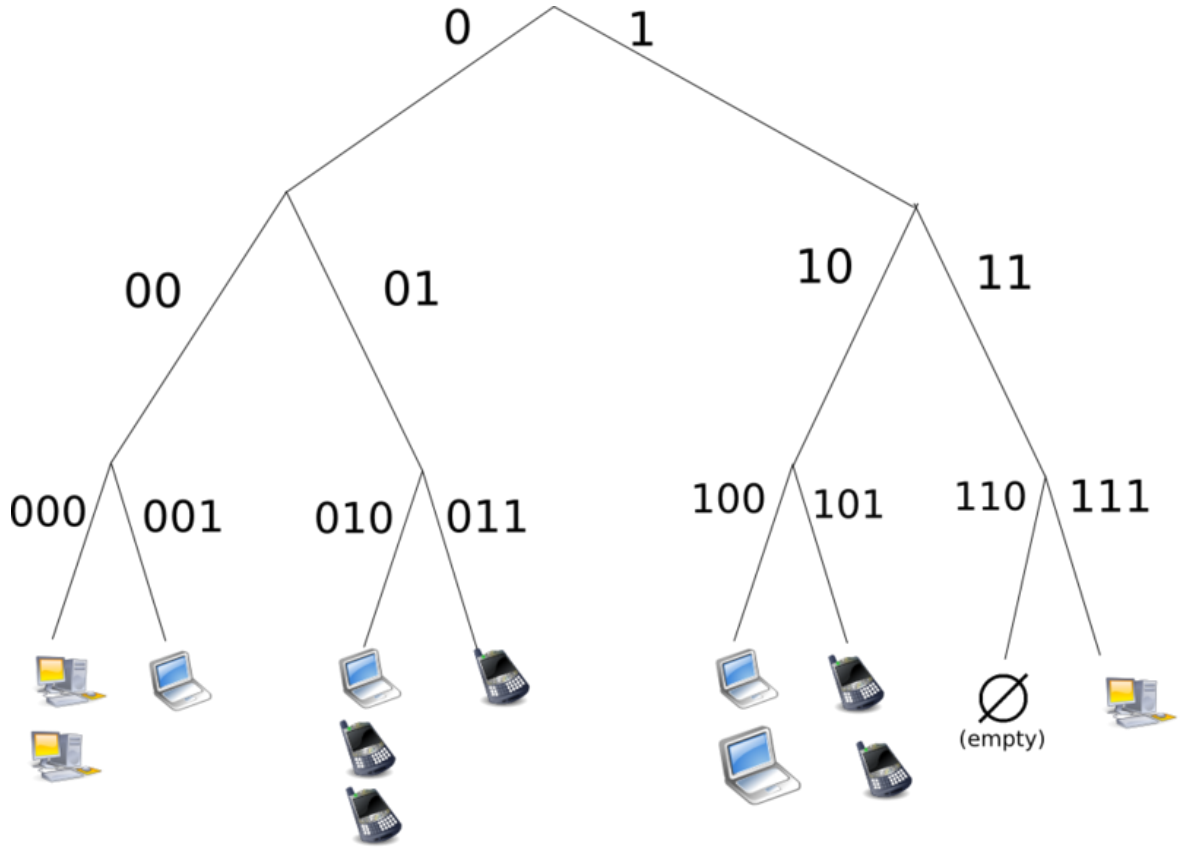


Рисунок 1.3 – Архитектура структурированной P2P-сети

К структурированным P2P-сетям относятся CAN, Chord, Pastry и Tapestry [17].

Наиболее популярной формой применения P2P-сетей является распространение контента.

Это включает публикацию и распространение программного обеспечения, сети доставки контента, потоковое мультимедиа и повторную передачу для многоадресных потоков, что облегчает доставку контента по требованию. Другие приложения включают научные, сетевые, поисковые и коммуникационные сети. Даже Министерство обороны США начало исследования приложений для P2P-сетей для современных стратегий сетевой борьбы [16].

1.2 Математические модели P2P-систем

1.2.1 Математическая модель распространения файла в пиринговой файлообменной сети

Рассмотрим модель распространения файла в пиринговой файлообменной сети, построенная на основе обыкновенных дифференциальных уравнений [3].

Эта модель в некотором приближении описывает распространение файла в пиринговой сети и может быть использована, например, для описания процесса обмена файлами в BitTorrent-сети.

BitTorrent – это пиринговый протокол для кооперативного обмена файлами (рисунок 1.4).

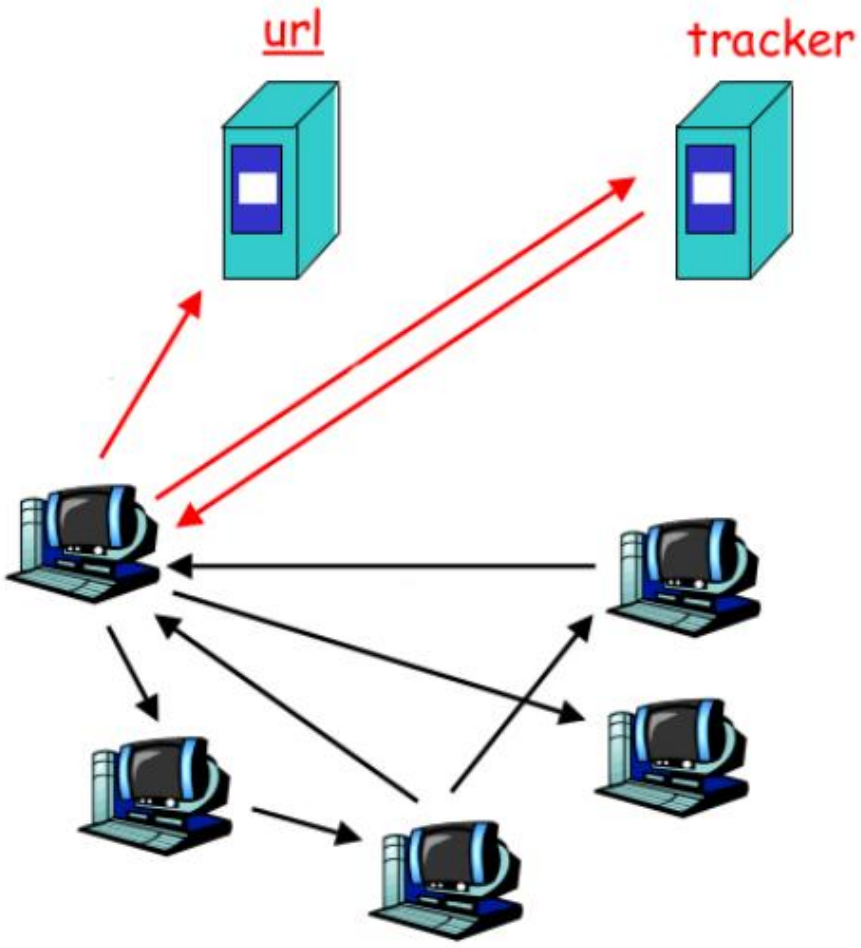


Рисунок 1.4 – Архитектура сети BitTorrent

В реальной файлообменной сети одновременно происходит как скачивание, так и приток новых пользователей и отток старых (4), так что её динамика описывается уравнениями:

$$\begin{cases} \dot{s} = C(s,l) - \beta_1 \cdot s \cdot e^{-\mu_1 l} \\ i = \alpha \cdot s \cdot (N - l - s) - C(s,l) - \beta_2 \cdot l \cdot e^{-\mu_2 (s+l)}, \end{cases} \quad (1.1)$$

где $C(s, l) = \gamma \bullet s \bullet l$;

$\gamma > 0$ - характеризует пропускную способность каналов раздачи-скачивания;

$\alpha > 0$ и $N > 0$ — параметры притока новых личеров (N — общее число пользователей ресурса, заинтересованных в файле, α характеризует привлекательность раздачи);

β_1, μ_1 и β_2, μ_2 — параметры оттока сидеров (пиры, содержащие файлы) и личеров (пиры, выполняющих скачивание), соответственно.

Выбором единиц измерения s и l можно любой из этих параметров сделать единицей, поэтому примем $\gamma = 1$ и далее будем рассматривать систему в следующем виде:

$$\begin{cases} \dot{s} = s \cdot l - \beta_1 \cdot s \cdot e^{-\mu_1 l} \\ i = \alpha \cdot s \cdot (N - l - s) - s \cdot l - \beta_2 \cdot l \cdot e^{-\mu_2 (s+l)}, \end{cases} \quad (1.2)$$

где: $\alpha > 0$, $N > 0$, $\beta_{1,2} > 0$, $\mu_{1,2} > 0$.

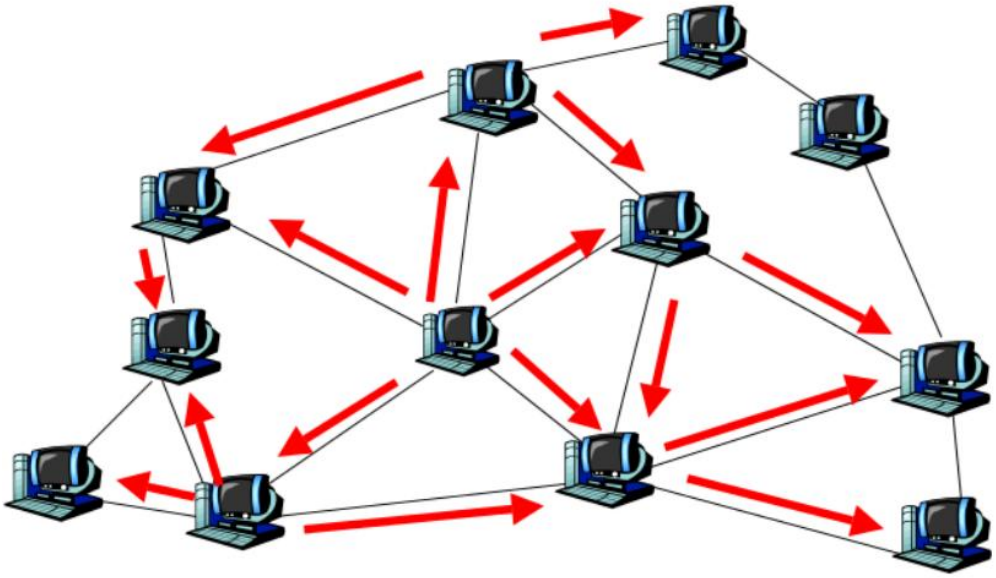
Следует иметь в виду, что, кроме детерминированных воздействий, на реальную раздачу действуют также случайные возмущения, которые влияют как на смещение текущего состояния раздачи на фазовой плоскости, так и на мгновенные значения параметров α, N, β_1, β_2, μ_1 и μ_2.

1.2.2 Модель сети Gnutella

Наиболее распространенными моделями P2P-систем являются модели на графах, описывающих состояние системы.

Рассмотрим в качестве примера математическую модель сети Gnutella.

Сеть Gnutella (рисунок 1.5) предоставляет чисто распределенное решение для обмена файлами [23].



Риунок 1.5 – Архитектура сети Gnutella

Пользователи могут запускать программное обеспечение, которое реализует протокол Gnutella для того, чтобы делиться файлами и искать новые файлы.

В таблице 1.1 представлены сообщения протокола Gnutella.

Таблица 1.1 - Сообщения протокола Gnutella

Сообщение	Описание
Широковещательные сообщения	
Ping	Исходное сообщение
Query	Шаблон поиска и TTL (время жизни)
Ответные сообщения	
Pong	Ответ на Ping, содержащий информацию об узле
Query response	Содержит информацию о компьютере, на котором находится нужный файл

Продолжение табл. 1.1

Сообщения между узлами	
PUSH	Загрузка файла

Рассмотрим алгоритм моделирования механизма поиска в сети Gnutella.

Шаг 1. Узел 2 инициализирует поиск файла A (рисунок 1.6).

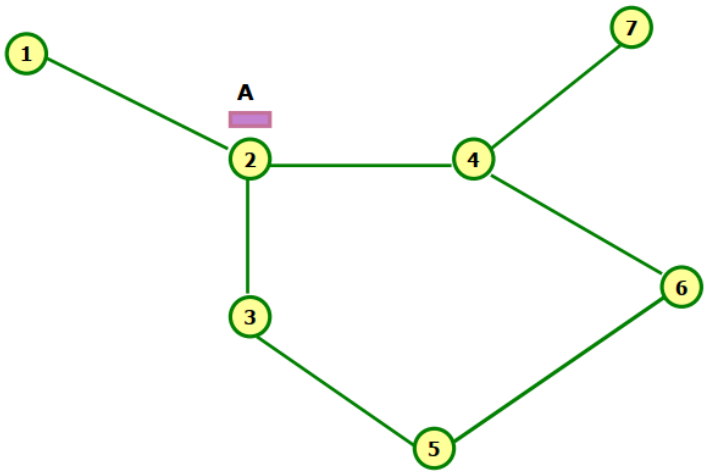


Рисунок 1.6 – Начальное состояние P2P-системы

Шаг 2. Узел 2 отправляет запрос Query соседним узлам сети (рисунок 1.7).

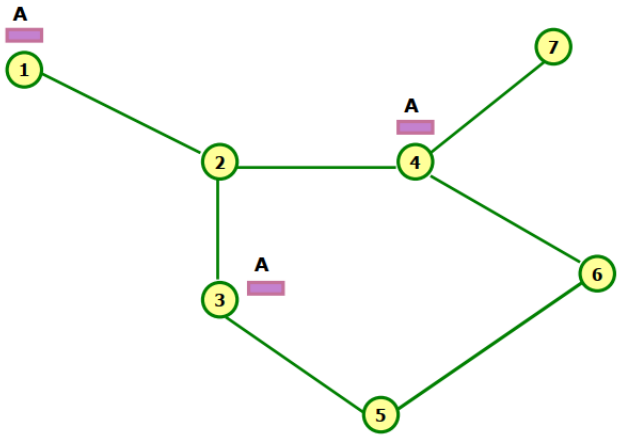


Рисунок 1.7 – Состояние P2P-системы на шаге 2

Шаг 3. Файл A не найден, отправляет сообщение Query response (рисунок 1.8).

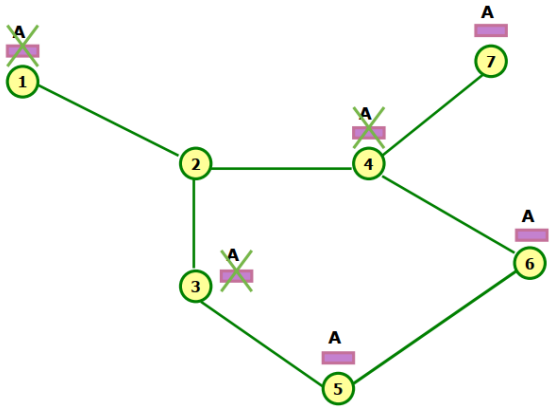


Рисунок 1.8 – Состояние P2P-системы на шаге 3

Шаг 4. Файл найден на узле 7 (рисунок 1.9).

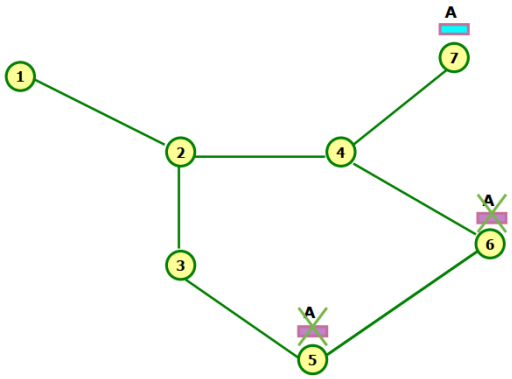


Рисунок 1.9 – Состояние P2P-системы на шаге 4

Шаг 5. Получено ответно сообщение на запрос (рисунок 1.10).

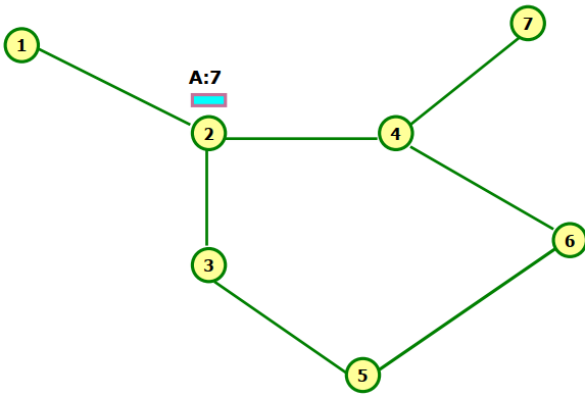


Рисунок 1.10 – Состояние P2P-системы на шаге 5

Шаг 6. Загрузка файла (рисунок 1.11).

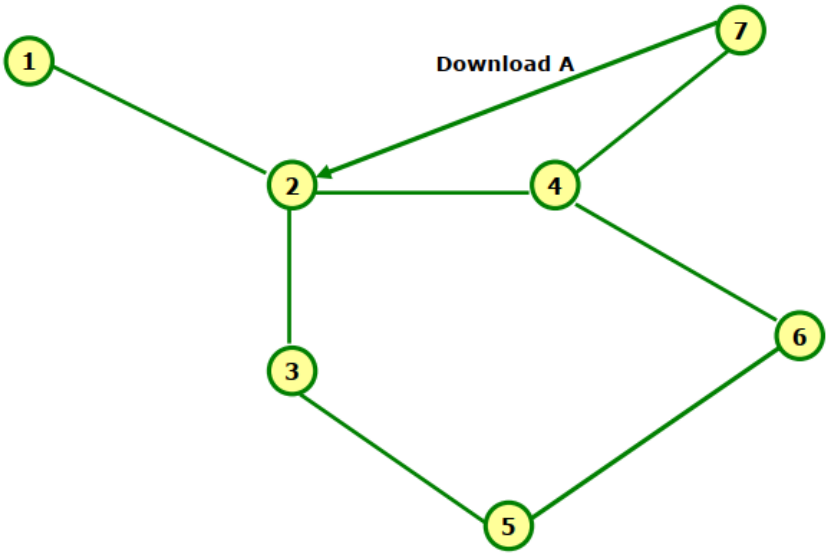


Рисунок 1.11 – Состояние P2P-системы на шаге 6

Представленный алгоритм моделирования положен в основу программного обеспечения, позволяющего пользователям искать и обмениваться файлами в сети Gnutella.

1.2.3 Математическая модель системы кабельного вещания IPTV

IPTV представляет собой принципиально новую форму коммуникации, которая успешно сочетает в себе информационную полноту и насыщенность сети Интернет с богатыми графическими и акустическими возможностями современных телевизионных систем.

Физическая модель IPTV - сервиса представлена на рисунке 1.12 [19].

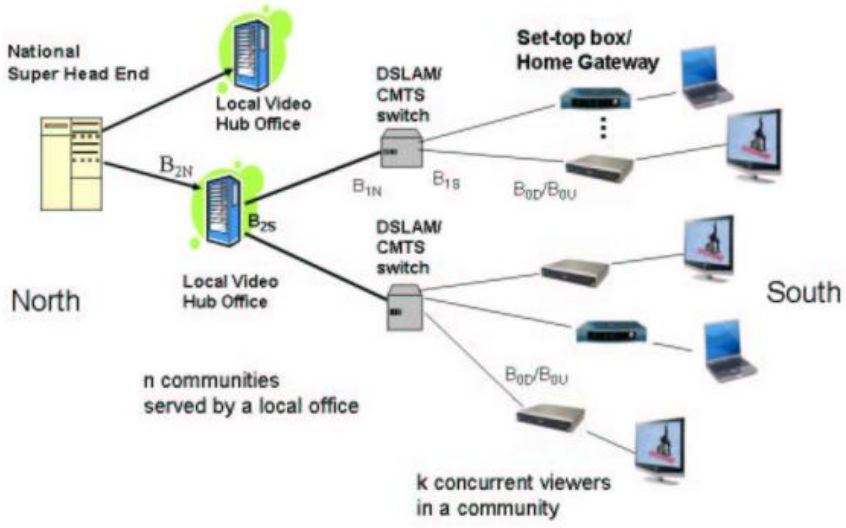


Рисунок 1.12 - Физическая модель IPTV - сервиса

В последние годы запущен в эксплуатацию ряд крупномасштабных P2P систем передачи потокового видео. Например, CoolStreaming, PPLive, PPStream, UUSee, SoftCast. Все эти системы используют общий принцип передачи информации, получивший название «модель изолированного канала» (ISO, Isolated channel model).

Математическая модель P2P-сети может быть построена по отношению к n-му пользователю, использующего m-й канал, или же с использованием m-го канала, количественно учитывающего соответствующую нагрузку [4].

В 1-м случае имеем матрицу состояний $X=(x_{nm})$, где x_{nm} принимает значение 1, если n-й пользователь использует m-й канал, и 0 в противном случае.

Данной модели может быть придан вероятностный смысл $P_m(k)$, показывающий с какой вероятностью на k-м шаге будет задействован m-й канал:

$$P_m(k) = P\{x \in X_m(k)\} = \sum_{x - x_{m(k)}}^{x} P(x) \qquad (1.3)$$

Для 2-й модели, отображающей загрузку m-го канала, требуется динамическое представление в виде состояний x на каждом k-м шаге.

Состояние P2P-сети может быть представлено вектором:

$$\overline{P_m^T} = (x_m^{(1)}, x_m^{(2)}, ..., x_m^{(n)}) \qquad (1.4)$$

Разностное уравнение отображающее скорость изменения состояния m-го канала на (k+1)-м шаге можно представить в виде:

$$X_m(k+1)=F(k+1,k)x(k) + G(k)\xi(k), \qquad (1.5)$$

где:

$F(k+1,k)$ - переходная вероятность между состояниями на k-м и (k+1)-м шаге, соответственно;

$G(k)$ - масштабируемый множитель, определяющий интенсивность изменения состояния;

$\xi(k)$ - порождающий гауссов белый шум, являющийся источником случайных изменений состояний.

В соответствии с методом переменных состояния на основании уравнения (1.5) могут быть построены процедуры в виде фильтра Калмана-Бьюси или алгоритмов стохастической аппроксимации, что позволит дальше на основании этих оценок построить управление состоянием каналов, т.е. произвести оценку загрузки сети и перераспределение ее ресурсов.

1.2.4 Моделирование структурированных P2P-систем

В основу моделей структурированных P2P-систем положена концепция распределенной хеш-таблицы Distributed hash table (DHT).

DHT - это класс децентрализованной распределенной системы, который предоставляет службу поиска, аналогичную хеш-таблице: пары (ключ, значение) хранятся в DHT, и любой участвующий узел может эффективно извлечь значение, связанное с данным ключом.

Ключи - это уникальные идентификаторы, которые отображаются на конкретные значения, которые, в свою очередь, могут быть любыми: от адресов, документов до произвольных данных.

Ответственность за поддержание соответствия между ключами и значениями распределяется между узлами таким образом, что изменение набора участников приводит к минимальным нарушениям.

Это позволяет DHT масштабироваться до чрезвычайно большого числа узлов и обрабатывать постоянные прибытия, отъезды и сбои узлов.

Рассмотрим алгоритм моделирования DHT [20].

Шаг 1. Переход от ключей и узлов к идентификаторам (ID).

Ключи и узлы представлены идентификаторами, взятыми из одного и того же пространства идентификаторов.

Ключевые идентификаторы вычисляются с помощью хэш-функции (например, SHA-1):

ID (k) = SHA1(k);

− идентификаторы узла назначаются случайным образом или вычисляются на с помощью хэш-функции:

ID(n) = SHA1 (IP-адрес n).

Шаг 2. Разделение ID-пространства.

Каждый узел в DHT хранит несколько пар k,v.

Делим пространство идентификаторов на зоны в зависимости от идентификаторов узлов (рисунок 1.13):

- пара (k,v) хранится в узле n так, что (примеры):

− идентификатор ID(n) наиболее близок к идентификатору (k);

− идентификатор ID(n) является наибольшим идентификатором узла меньше, чем ID(k).

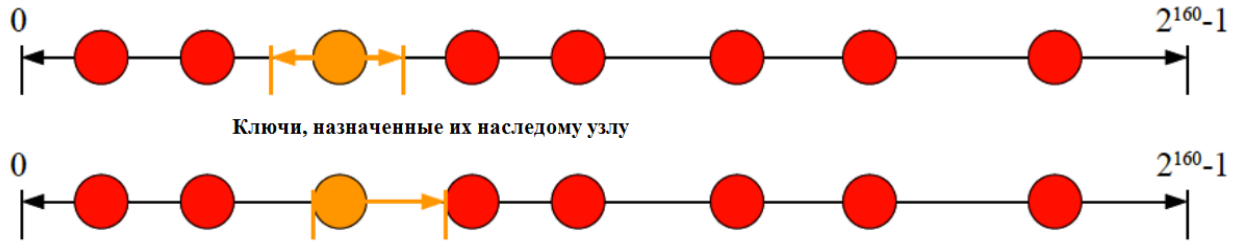


Рисунок 1.13 - Переход от ключей и узлов к идентификаторам

Шаг 3. Построение оверлейной сети.

Каждый узел DHT управляет O(log n) - ссылками на другие узлы, где n -

количество узлов в системе (рисунок 1.14).

Каждый узел имеет два набора соседей:

− ближайшие соседи по ключевому пространству (листья);

− дальние соседи.

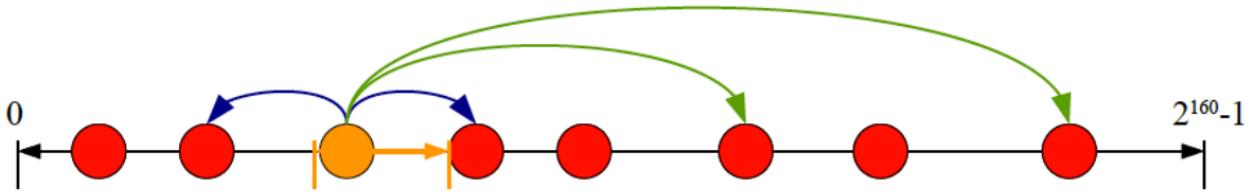


Рисунок 1.14 - Построение оверлейной сети

Шаг 4. Установить маршрут «поместить/ получить» через оверлей.

Рекурсивная маршрутизация: инициатор запускает процесс, а связавшиеся узлы пересылают сообщение (рисунок 1.16).

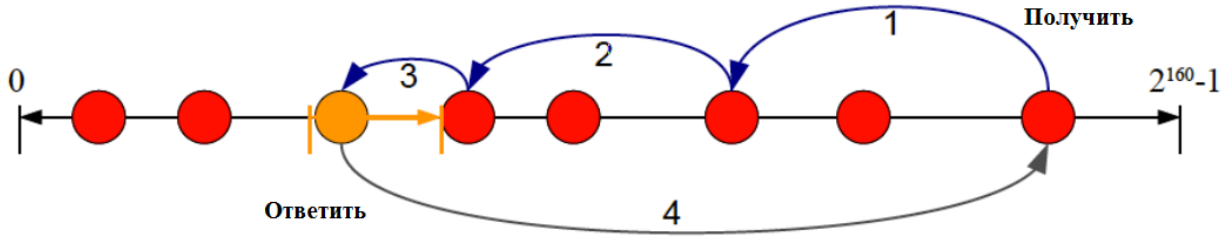


Рисунок 1.15 - Рекурсивная маршрутизация

Итеративная маршрутизация: инициатор лично связывается с узлами на каждом этапе маршрута (рисунок 1.16).

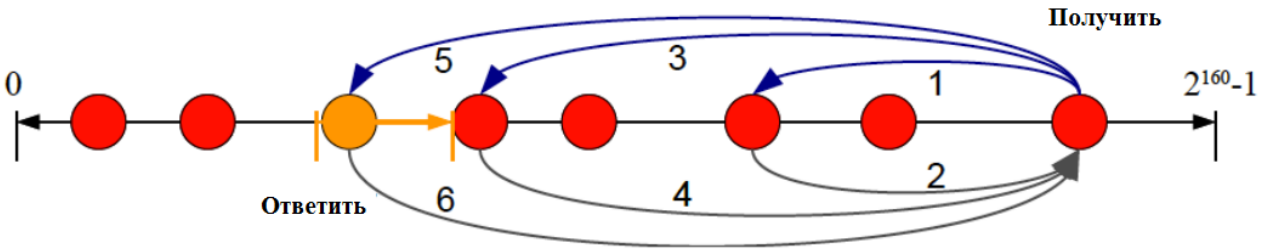


Рисунок 1.16 - Итеративная маршрутизация

Примером структурированной P2P-сети является Chord (рисунок 1.17).

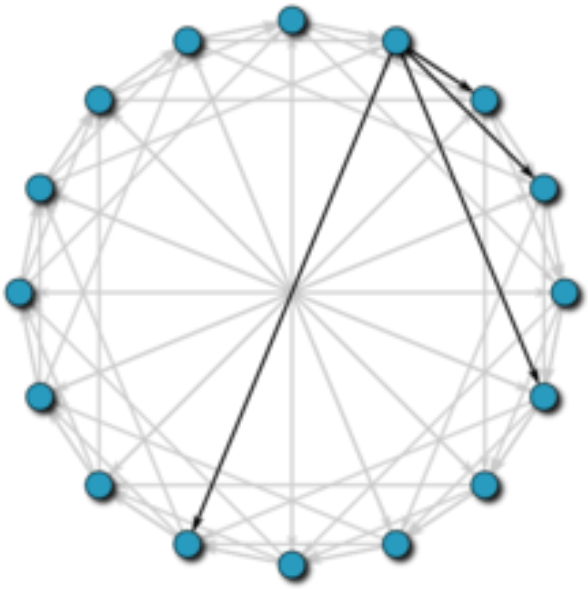


Рисунок 1.17 – Архитектура сети Chord

На рисунке 1.18 изображен пример добавления нового узла X в кольцо сети Chord [24].

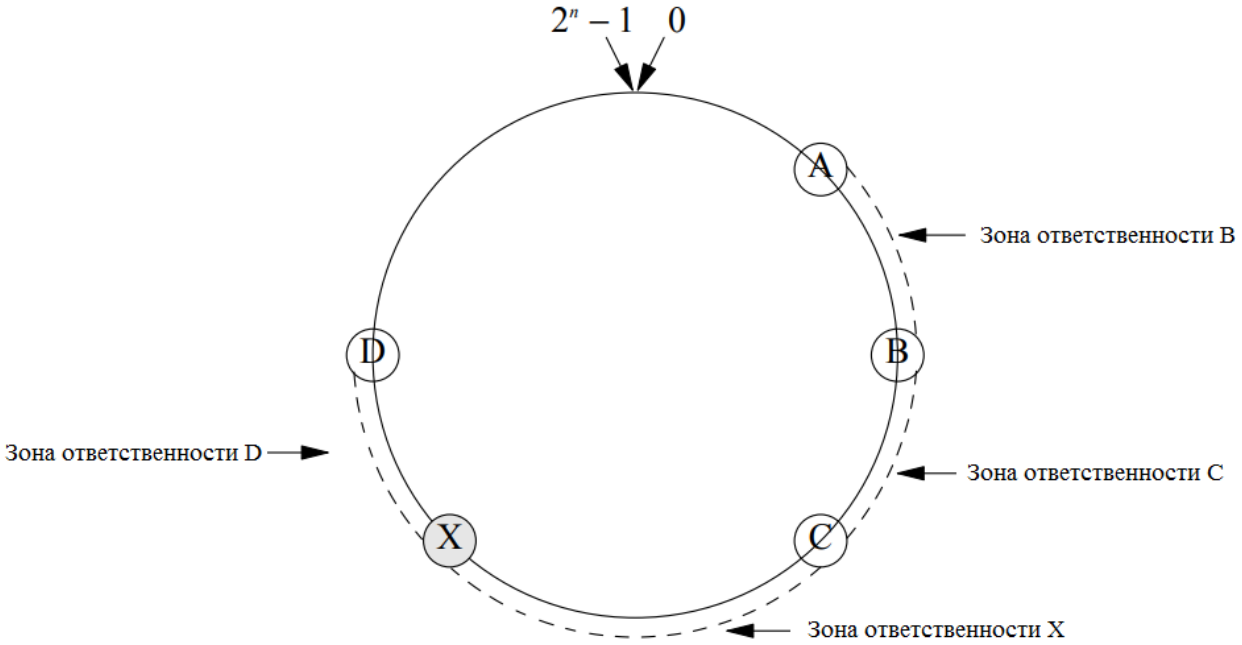


Рисунок 1.18 – Кольцо Chord

Главным недостатком DHT является проблема безопасности сетей, так как вызывает сложности проверка целостности данных и не обеспечена безопасность маршрутизации.

Следует учесть, что DHT-технология предназначена для решения более традиционных проблем распределенных систем, таких как балансировка нагрузки, целостность данных и производительность (в частности, обеспечение быстрого выполнения таких операций, как маршрутизация и хранение или извлечение данных).

Выводы к первой главе

1. С учетом характера взаимосвязей пиров в оверлейной сети и принципов индексации и расположения ресурсов P2P-сети классифицируются как неструктурированные, структурированные и гибридные.

2. Модель распространения файла в пиринговой файлообменной сети, построенная на основе обыкновенных дифференциальных уравнений в

некотором приближении описывает распространение файла в неструктурированных пиринговых сетях.

3. Наиболее распространенными математическими моделями P2P-систем являются модели на графах, описывающих состояние системы.

4. В основу моделей структурированных P2P-систем положена концепция распределенной хеш-таблицы - Distributed hash table (DHT).

Глава 2 КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕННЫХ P2P-СИСТЕМ

Для проведения вычислительных экспериментов с распределенными P2P-системами используются их компьютерные симуляции – имитационные модели данных систем, реализованные с помощью специализированных программных комплексов [2, 12].

Создание компьютерной модели, адекватно отображающей исследуемую P2P-систему, является сложной задачей.

Это обуславливает потребность в применении высокоуровневых инструментальных средств автоматизации данного процесса, построенных по методологическому принципу «модель-алгоритм-программа» [5].

Следует отметить, что в настоящее время разработано и широко применяется достаточное количество проблемно-ориентированных средств компьютерной симуляции – P2P-симуляторов, позволяющих исследовать различные аспекты распределенных вычислительных P2P-сетей [6].

Архитектура типового P2P-симулятора представлена на рисунке 2.1 [25].

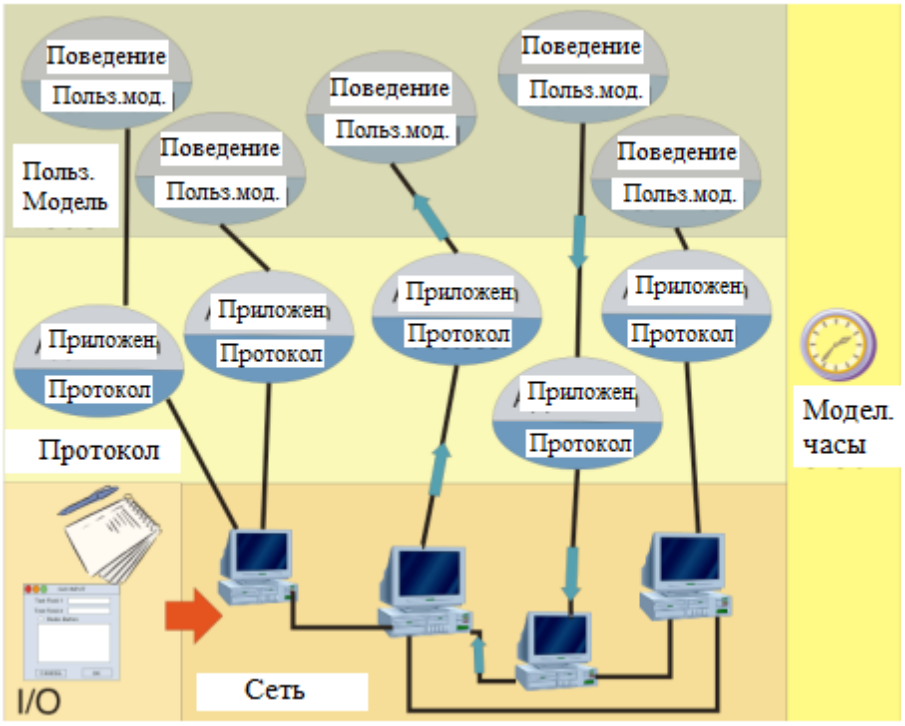


Рисунок 2.1 – Архитектура типового P2P-симулятора

Статические пошаговые часы используются для моделирования времени.

Благодаря отделению сети, протокола и приложения друг от друга становится возможным моделирование различных сетевых топологий для различных протоколов, приложений и пользовательских моделей.

Таким образом, определяются следующие уровни моделирования:

- уровень сети (внизу);
- уровень протокола (средний);
- уровень пользователя (вверху).

Связь может устанавливаться только между напрямую связанными уровнями.

Уровень протокола, который отвечает за моделирование протокола с желаемым приложением, действует как интерфейс между уровнем пользователя и уровнем сети.

Входная информация от пользователя подается на уровень сети через графический интерфейс или файл.

В настоящее время пользователям предлагается широкий выбор P2P-симуляторов с открытым программным кодом: SimGrid, PeerSim, NeuroGrid, 3LS, P2PSim, GPS и др. [21].

Рассмотрим и сравним возможности этих средств на предмет создания адекватных компьютерных моделей симуляции P2P-систем.

Для этого предварительно обозначим характеристики, на которые необходимо обратить внимание при выборе P2P-симулятора:

- модели P2P –протоколов, которые поддерживаются симулятором;
- максимальное количество узлов (пиров) моделируемой P2P-системы;
- используемый язык программирования;
- формат файла результатов;
- юзабилити.

При необходимости этот перечень может быть расширен.

2.1 Универсальная среда для симуляции SimGrid

SimGrid - это инструментарий, который предоставляет основные функции для моделирования распределенных приложений в гетерогенных распределенных средах. Конкретной целью проекта является содействие исследованиям в области параллельных и распределенных крупномасштабных систем, таких как сети, системы P2P и облачные технологии [11].

Его варианты использования включают эвристическую оценку, создание прототипов приложений или даже разработку и настройку реальных приложений.

Концепция моделирования в SimGrid основана на ориентированном ациклическом графе (ОАГ) G(υ, ε), где:

$\upsilon = \{ v_i \mid i = 1,\ldots,V\}$ – множество вершин, представляющих задачи;

$\varepsilon = \{ e_{i,j} \mid (i,j) \in \{1,\ldots,V\} \times \{1,\ldots,V\}\}$ – множество ребер, представляющих ограничения приоритета и / или перемещения данных между задачами (рисунок 2.2).

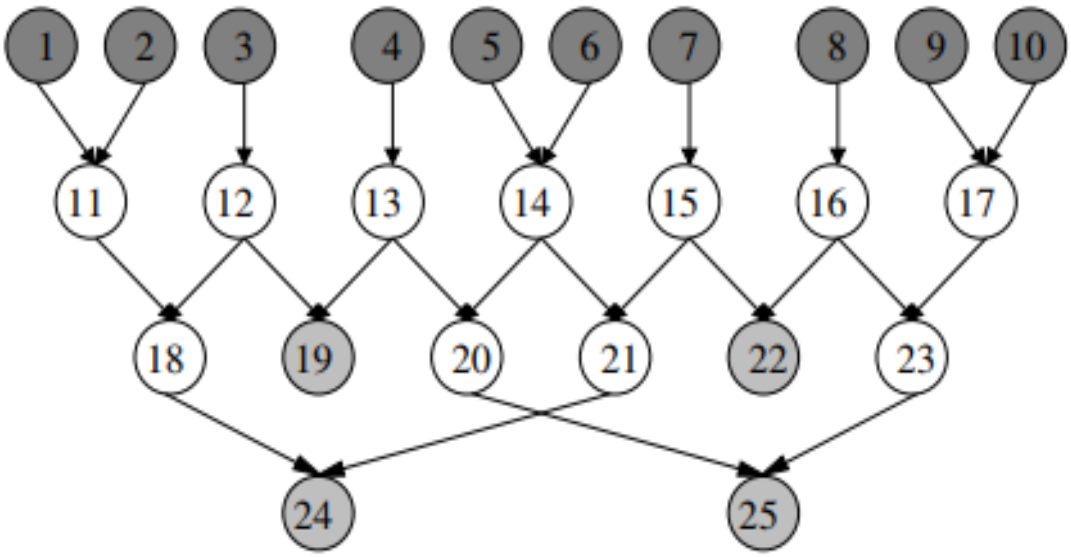


Рисунок 2.2- Пример ОАГ

Архитектура SimGrid представлена на рисунке 2.3.

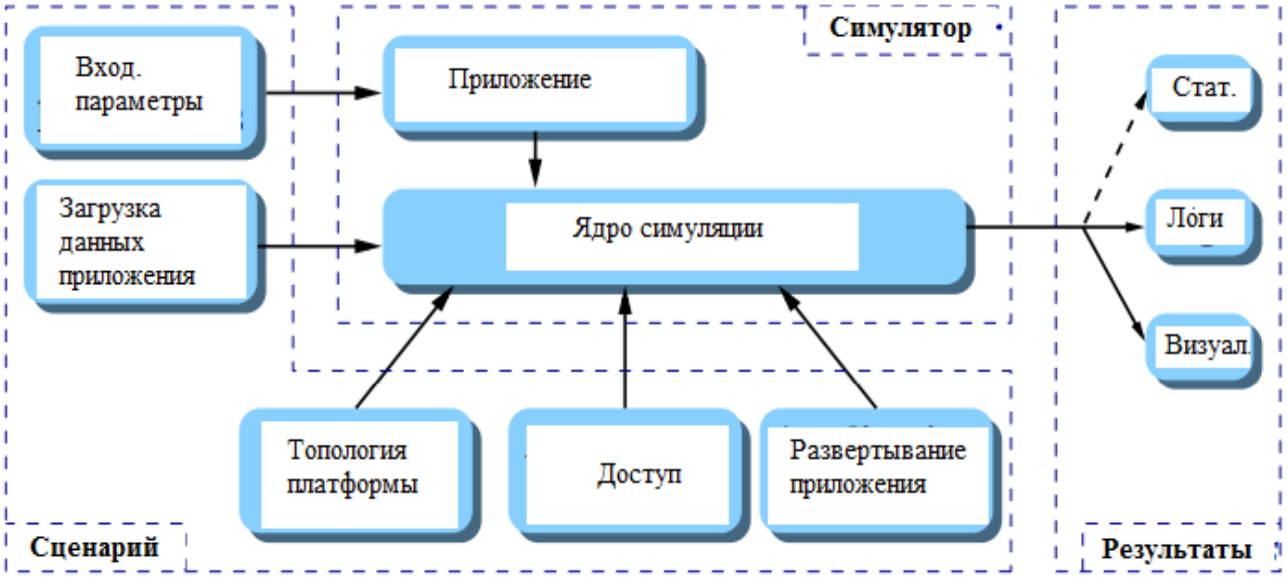


Рисунок 2.3 – Архитектура SimGrid

SimGrid включает следующие доступные пользователю компоненты (рисунок 2.4):

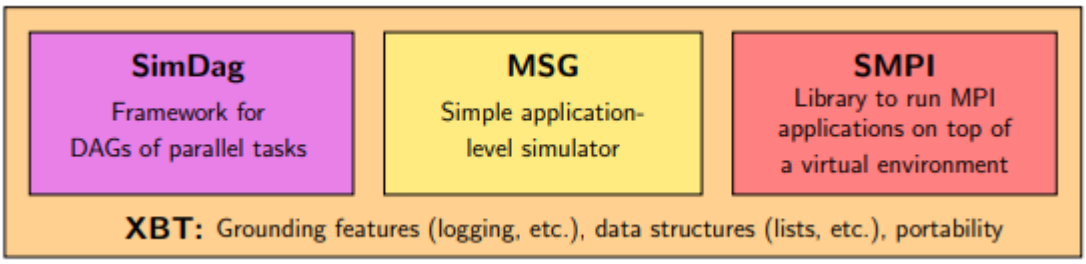


Рисунок 2.4 – Пользовательские компоненты SimGrid

MSG - эвристика как параллельные последовательные процессы; Java-привязки (+ lua);

SimDag - эвристика как ОАГ (параллельные) задачи;

SMPI: моделирование реальных приложений, написанные с помощью MPI-библиотеки.

К преимуществам SimGrid можно отнести:

− свободное распространение;

− возможность моделирования больших инфраструктур;

− отсутствие проблем, связанных с распределенными системами, таких, как синхронизация времени на всех узлах или обработки ошибок на узлах;

− повышение эффективности вычислительного эксперимента и др.

Проект SimGrid существует уже 15 лет. За это время система приобрела широкую популярность среди специалистов как качественная среда для создания компьютерных симуляций.

2.2 Симулятор PeerSim

P2P-системы могут быть чрезвычайно большого масштаба (миллионы узлов). Узлы включаются и выходят из сетей непрерывно. Эксперименты с протоколом в такой среде представляют сложную задачу.

Стимулятор PeerSim был разработан, чтобы справиться с этими проблемами, обеспечить масштабируемость и поддержку исследования системы в динамике [10].

Кроме того, структура PeerSim основана на компонентах и облегчает быстрое создание прототипа протокола, объединяющего различные подключаемые строительные блоки, которые на самом деле являются объектами Java.

На рисунке 2.5 изображены различные уровни протоколов P2P-сети в концепции PeerSim.

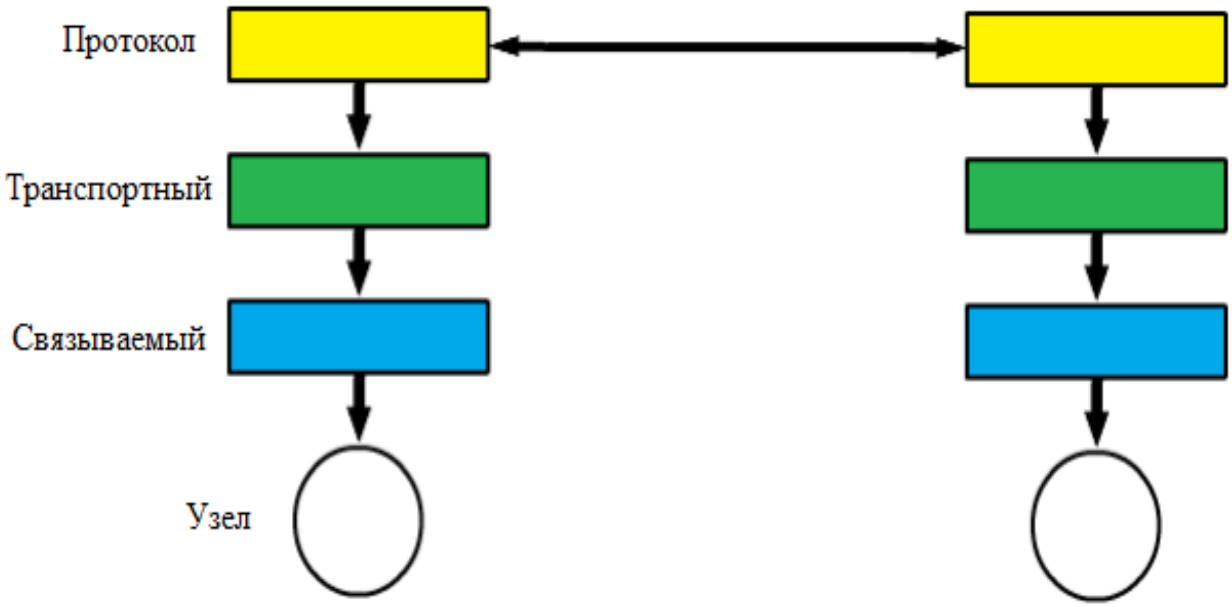


Рисунок 2.5 – Структура стека протоколов P2P-сети в концепции PeerSim

По мнению разработчиков, в PeerSim, почти для всех настраиваемых симуляций будет достаточно научиться описать только два типа компонентов:

1. Protocol - класс, реализующий интерфейс протокола;

2. Control - класс, реализующий интерфейс управления.

Первый тип компонентов четко имеет дело с протоколами, с которыми проводится эксперимент, и вспомогательными для них, а второй – с супервизорами, которые следят за соблюдением протоколов, изменяя состояние последних.

Принципиальное отличие заключается в том, как они видят сеть.

Объекты класса Control видят все узлы и сеть в целом, поэтому они имеют доступ ко всей информации последней;

Объекты класса Protocol содержится в каждом узле, они могут действовать как на сам узел, так и на соседние узлы, которые они могут просматривать.

PeerSim был разработан для использования модульного программирования на основе объектов. Каждый блок легко заменяется другим компонентом, который реализует тот же интерфейс (функциональность).

Общий алгоритм имитационного моделирования в PeerSim имеет следующий вид:

− выбрать размер сети (количество узлов);

− выбрать один или несколько протоколов для экспериментов и инициализировать их;

− выбрать один или несколько объектов управления, чтобы отслеживать интересующие исследователя свойства и изменить некоторые параметры во время моделирования (например, размер сети, внутреннее состояние протоколов и т. д.);

− запустить симуляцию, вызвав класс Simulator с файлом конфигурации, который содержит представленную выше информацию.

PeerSim предлагает следующие базовые модели для моделирования P2P:

− OverStat (протоколы агрегации для предоставления статистической информации в сети);

− SG-1 (самоорганизация и поддержание основанной на суперпире топологии);

− T-Man (используется для построения топологий с использованием функции ранжирования, определяющей предпочтение каждого узла для соседей).

PeerSim поддерживает два подхода к имитационному моделированию: моделирование на основе событий и на основе циклов.

Вторая модель проще, что позволяет достичь максимальной масштабируемости (более 1 млн. узлов) и производительности за счет некоторой потери адекватности.

Однако многие простые протоколы могут пренебречь этой потерей без каких-либо проблем.

Упрощенные гипотезы модели, основанной на циклах, - это отсутствие моделирования транспортного уровня и отсутствие конкуренции.

Другими словами, узлы обмениваются данными друг с другом напрямую, и узлам периодически предоставляется проверка в последовательном порядке, чтобы они могли выполнять произвольные действия.

Следует отметить, что в случае необходимости относительно легко перенести любое моделирование на основе цикла в механизм на основе событий.

2.3 Симулятор NeuroGrid

NeuroGrid изначально был разработан в качестве альтернативной модели маршрутизации для Gnutella. Основное внимание уделяется минимизации отправки сообщений за счет увеличения затрат на обработку

запросов для пиров. Он предназначен для того, чтобы обеспечить метод связи, который будет добавлен к http [9].

В концепции NeuroGrid предполагается, что ресурсы в сети P2P связаны с набором ключевых слов.

Использование модели децентрализованной маршрутизации обеспечивает более эффективную маршрутизацию за счет дополнительных издержек обработки.

В NeuroGrid используется событийное имитационное моделирование.

На рисунке 2.6 представлена таблица событий с действиями (actions), связанными с различными временными шагами (timesteps).

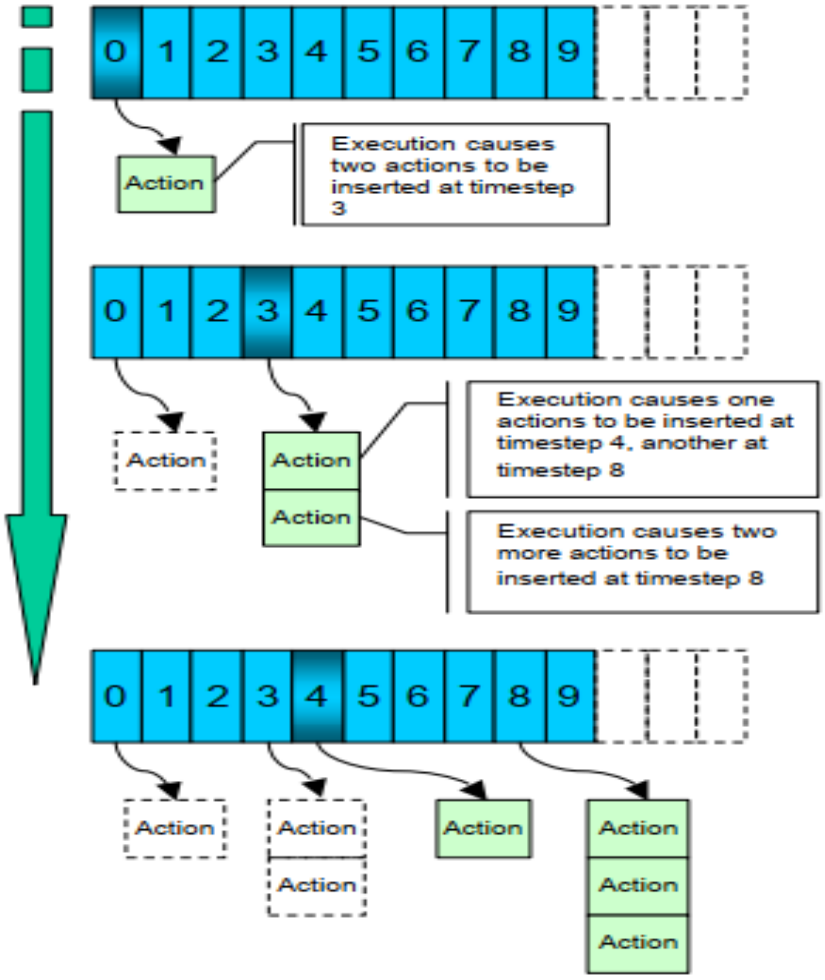


Рисунок 2.6 - Таблица событий NeuroGrid

Для удобства анализа характеристик P2P-симуляторов составим сводную таблицу 2.1 [18].

Таблица 2.1 – Сравнительный анализ характеристик P2P-симуляторов

Характеристика/ Симулятор	SimGrid	PeerSim	NeuroGrid
Модели P2P -протоколов	Нет данных	Коллекция моделей P2P собственной разработки	Gnutella, NeuroGrid, Pastry, FreeNet
Максимальное количество узлов	Нет данных	>1 млн.	300 тыс.
Язык программирования	C, C++, Ruby, Lua	Java	Java
Формат файла результатов	Текст	Тест,GML и др.	Нет данных
Юзабилити (макс. оценка 3)	3	2	3

Анализ показал, что существует большое разнообразие симуляторов P2P-систем с открытым исходным кодом.

Однако, как следует из предлагаемого анализа, они не удовлетворяют всем установленным требованиям для решения задач компьютерной симуляции P2P-систем. Это объясняет появление большого количество заказных разработок симуляторов, оценить функциональные возможности которых по представленным в Интернете статьям и описаниям очень сложно.

Вместе с тем, анализ подтвердил достаточно высокий спрос на функции симуляторов, позволяющих создавать компьютерные симуляции

P2P-сетей с различной топологией сети, пользовательскими моделями и приложениями.

Выводы ко второй главе

1. Для проведения вычислительных экспериментов с распределенными P2P-системами используются имитационные модели данных систем, реализованные с помощью специализированных программных комплексов – P2P-симуляторов.
2. Анализ показал, что известные P2P-симуляторы с открытым кодом не соответствуют всем требованиям для решения задач компьютерной симуляции P2P-систем.
3. Анализ подтвердил достаточно высокий исследовательский интерес к симуляторам, позволяющим создавать компьютерные симуляции P2P-сетей с различной топологией сети, пользовательскими моделями и приложениями.

Глава 3 ИСПОЛЬЗОВАНИЕ P2P-СИМУЛЯТОРОВ ДЛЯ СОЗДАНИЯ КОМПЬЮТЕРНЫХ СИМУЛЯЦИЙ РАСПРЕДЕЛЕННЫХ P2P-СИСТЕМ

Для проверки возможностей P2P-стимуляторов по созданию адекватных имитационных моделей P2P-систем рассмотрим примеры разработки с их помощью компьютерных симуляций популярных P2P-сетей.

3.1 Имитационное моделирование сети Chord на симуляторе PeerSim

В таблице 3.1 представлены классы симулятора PeerSim.

Таблица 3.1 – Описание классов симулятора PeerSim

Класс	Наименование
peersim	Основные точки входа для Peersim
peersim.cdsim	Симулятор на основе циклов
peersim.config	Управление конфигурацией
peersim.core	Классы ядра
peersim.dynamics	Control классы для инициализации сети или ее изменения во время симуляции.
peersim.edsim	Симулятор на основе событий
peersim.graph	Граф структуры данных, утилиты ввода-вывода и алгоритмы
peersim.rangesim	Симулятор, который позволяет определять диапазоны для переменных.
peersim.reports	Классы и интерфейсы для моделирования связи на транспортном уровне в оверлейных сетях
peersim.transport	Классы и интерфейсы для

Класс	Наименование
	моделирования связи на транспортном уровне в оверлейных сетях
peersim.util	Классы-утилиты
peersim.vector	Классы, которые позволяют модифицировать и анализировать числовые векторы, определяемые вектором экземпляров протокола в сети

Для построения конфигурации сети используется класс Cogfiguration (рисунок 3.1).

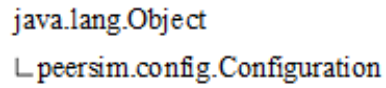


Рисунок 3.1- Представление класса Cogfiguration

Это полностью статический класс для хранения информации о конфигурации. Пример кода файла конфигурации config-chord.cfg приведен на рисунке 3.2 [13].

```
// начальное число, используемое для генерации псевдослучайного числа
# random.seed 1234567890
simulation.endtime 10^6 /* время, затрачиваемое на моделирование*/
simulation.logtime 10^6 /* время, затрачиваемое на регистрацию результатов*/
simulation.experiments 1/* число экспериментов*/
network.size 5000 /* количество узлов*/
// Транспортный протокол для отправки сообщений между узлами
protocol.tr UniformRandomTransport
{
        mindelay 0
        maxdelay 0
}
//Протокол, имитирующий поведение приложения Chord в каждом узле
protocol.my ChordProtocol
{
        transport tr
}
control.traffic TrafficGenerator
{
        protocol my
        step 100
}
```

Рисунок 3.2 - Код конфигурации сети Chord

```
// Создание экземпляра подкласса управления, который будет генерировать сообщения
//поиска (источник и пункт назначения выбираются случайным образом) на каждом
//заранее заданном шаге времени.
init.create CreateNw
{
        protocol my
        idLength 128
        succListSize 12 /* размер списка преемников*/
}
//Наблюдатель, который вычисляет такие значения, как среднее количество переходов
при //обработке поиска и т.д.. Эти результаты печатаются при каждом указанном шаге.
control.observer MessageCounterObserver
{
        protocol my
        step 90000
}

control.dnet DynamicNetwork
{
        add 20
        add -25
        minsize 3000 /* минимальный размер сети */
        maxsize 7000 /* максимальный размер сети */
        step 100000 / *шаг динамического управления*/
        init.0 ChordInitializer
        {
                protocol my
        }
```

Продолжение рисунка 3.2

Чтобы протестировать протокол в состоянии оттока (узлы удалены и добавлены новые), используется предоставляемый PeerSim элемент управления DynamicNetwork, который можно настроить, указав количество добавляемых (или удаляемых) узлов, границы сети размерность и время шага.

Код класса CordInitializer данного элемента представлен в приложении А.

Чтобы созданный протокол Chord мог принимать новые узлы в сети, каждый узел должен быть инициализирован линиями инициализации.

Узел, который уже является частью кольца Chord, выбирается случайным образом и поможет новому найти преемников и записи таблицы пальцев.

Выполнение протокола в идеальном состоянии (без оттока, надежная передача) привело к результатам, которые были предсказуемы: максимальное количество переходов, которое должно пройти сообщение поиска, прежде чем достигнуть правильного пункта назначения, ограничено $\log_2 N$ (количество узлов в сети). Это число должно быть округлено до целого числа. Среднее число переходов также предсказуемо мало (поиск от узла к самому себе может быть сгенерирован).

Эти параметры должны были фиксированными и представлены выше.

Для выполнения программного кода используется команда:

java -cp peersim-1.0.3.jar:djep-1.0.0.jar:jep-2.3.0.jar peersim.Simulator config-chord.txt.

Результаты моделирования, представленные в виде диаграмм Excel, изображены на рисунке 3.3.

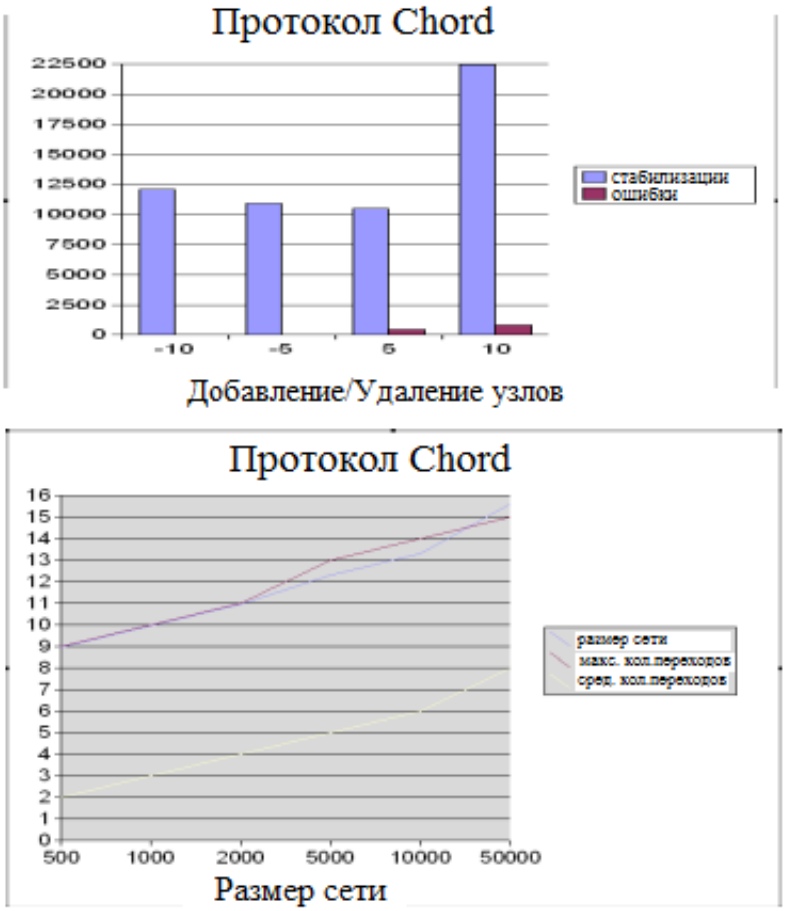


Рисунок 3.3 - Результаты моделирования протокола Chord

Представленная модель соответствует общепринятым представлениям о сети Chord, что подтверждает ее адекватность и широкие функциональные возможности симулятора PeerSim.

3.2 Имитационное моделирование сети Gnutella на симуляторе NeuroGrid

В NeuroGrid используются следующие абстрактные классы:

- Keyword;
- Document;
- Message;
- Node;
- Network;
- MessageHandler.

Для имитационного моделирования сети Gnutella используем представленную на рисунке 3.4 модель наследования абстрактных классов NeuroGrid [22].

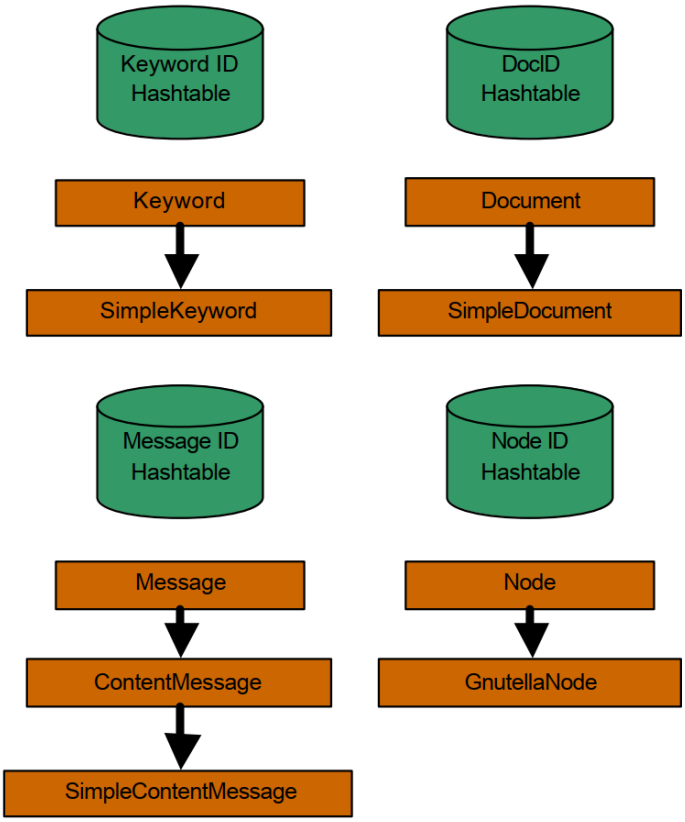


Рисунок 3.4 - Модель наследования абстрактных классов NeuroGrid для моделирования сети Gnutella

Функциональность конечного узла достигается за счет комбинации абстрактного класса Node и типа MessageHandler, с которым он связан.

Абстрактный класс Message содержит ряд переменных, общих для сообщений, в частности счетчик времени жизни (Time To Live, TTL) и ссылки на Node.

Переменные массива Document и Keyword SimpleMessage реализует второй конструктор, который используется, когда узлы пересылают сообщения (рисунок 3.5) [15].

```
public SimpleMessage(Message p_message)
   throws Exception
 {
   if(p_message == null) throw new Exception("Message is null");
   o_message_ID = p_message.getMessageID();
   o_TTL = p_message.getTTL() - 1;
   o_keywords = p_message.getKeywords();
   o_document = p_message.getDocument();
   etc …
 }
```

Рисунок 3.5 – Код метода SimpleMessage

Класс SimpleNode реализует следующий метод (рисунок 3.6).

```
public void processMessage(Message p_message, boolean p_start)
   throws Exception
 {
   if(p_message == null) throw new Exception("p_message is null");
   String x_previous = (String)(o_seenGUIDs.get(p_message.getMessageID()));
   if(x_previous != null)
      return;
       o_seenGUIDs.put(p_message.getMessageID(),p_message.getMessageID());
   etc …
```

Рисунок 3.6 – Код метода processMessage

Для создания узлов, которые будут отображать Gnutella-подобную функциональность, класс GnutellaNode расширяет абстрактный класс Node, добавляет некоторую репрезентативную реализацию и выбирает GnutellaMessageHandler.

Важными методами интерфейса MessageHandler являются следующие (рисунок 3.7).

```
public boolean handleMessage(Message message,
Node node)
        throws Exception;

public boolean injectMessage(Message message,
Node node)
        throws Exception;
```

Рисунок 3.7 – Методы интерфейса MessageHandler

Именно метод handleMessage реализует маршрутизацию в стиле Gnutella и, таким образом, изменяет работу сети так же просто, как редактирование или переопределение этого метода и создание узлов с соответствующими методами MessageHandler.

Код интерфейса MessageHandler (Java) представлен в приложении Б.

На рисунке 3.8 представлена модель поиска в сети Gnutella, разработанная в симуляторе NeuroGrid.

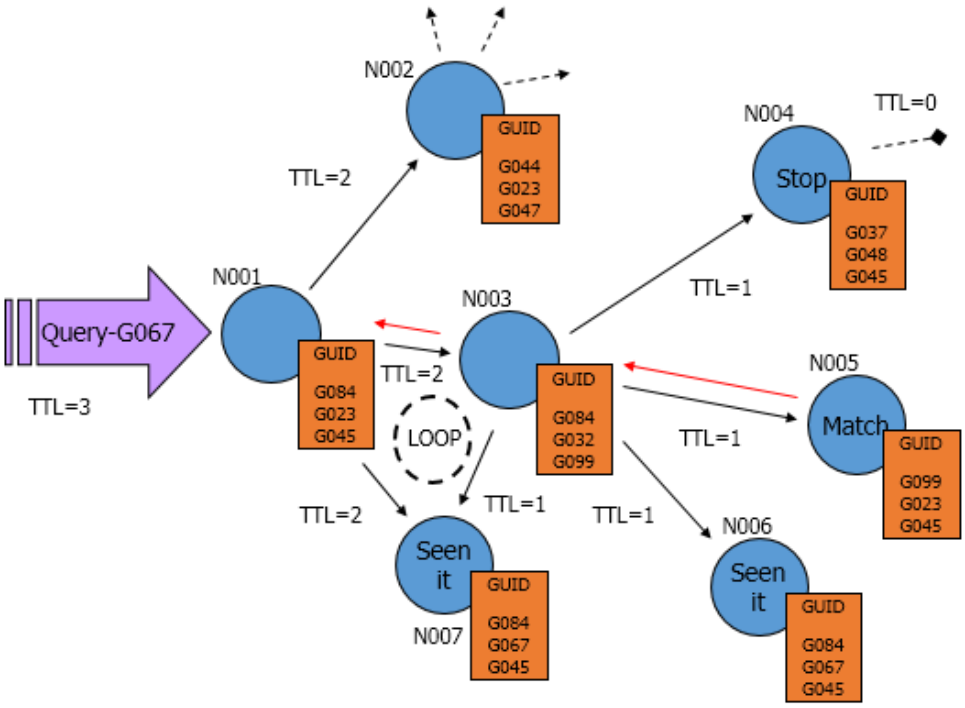


Рисунок 3.8 – Модель поиска сети Gnutella

Данная модель отражает алгоритм поиска в сетях, построенных на протоколе Gnutella, что подтверждает ее адекватность и широкие функциональные возможности симулятора NeuroGrid.

Выводы к третей главе

1. Для проверки возможностей P2P-стимуляторов по созданию адекватных имитационных моделей P2P-систем рассмотрены примеры разработки с их помощью компьютерных симуляций популярных P2P-сетей.

2. Результаты моделирования в симуляторе PeerSim сети Chord подтвердили ее адекватность и широкие функциональные возможности симулятора PeerSim.

3. Разработанная в симуляторе NeuroGrid модель отражает алгоритм поиска в сетях, построенных на протоколе Gnutella, что подтверждает ее адекватность и широкие функциональные возможности данного симулятора.

ЗАКЛЮЧЕНИЕ

Представленная бакалаврская работа посвящена актуальной проблеме анализа и практического применения методов математического и компьютерного моделирования распределенных P2P-систем.

В ходе выполнения бакалаврской работы достигнуты следующие результаты.

1. Проанализирована научная и учебно-методическая литература по исследуемой проблеме.

2. Проанализированы математические методы моделирования распределенных P2P-систем. Как показал анализ, наиболее распространенными математическими моделями P2P-систем являются модели на графах, описывающих состояние системы. В основу моделей структурированных P2P-систем положена концепция распределенной хеш-таблицы.

3. Проанализированы методы компьютерного моделирования распределенных P2P-систем. Как показал анализ, для проведения вычислительных экспериментов с распределенными P2P-системами используются имитационные модели данных систем, реализованные с помощью P2P-симуляторов.

4. Произведен анализ и представлены примеры применения программных средств имитационного моделирования распределенных P2P-систем. Представленные примеры подтверждают широкие возможности доступных P2P-стимуляторов PeerSim и NeuroGrid для создания имитационных моделей популярных P2P-систем.

Результаты бакалаврской работы могут быть рекомендованы для решения задач математического и компьютерного моделирования распределенных P2P-систем.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 34.936-91 Информационная технология. Локальные вычислительные сети. Определение услуг уровня управления доступом к среде.

2. ГОСТ Р 57412-2017 Компьютерные модели в процессах разработки, производства и эксплуатации изделий. Общие положения.

Научная и методическая литература

3. Кононова А. И. Динамическая модель процессов информационных обменов в пиринговой сети / А.И. Кононова // Моделирование и анализ информационных систем. -2018. -Т. 25. - № 4. -С. 421–434.

4. Ощепков М.Ю. Математическое моделирование пиринговых сетей / М.Ю. Ощепков, Е.О. Поповская // Наука и мир. -2014. -Т.1. - №3(7). –С. 195-197.

5. Самарский А.А. Математическое моделирование: Идеи. Методы. Примеры / А.А. Самарский, Михайлов А.П. – М.: Физматлит, 2001. – 320 с.

6. Феоктистов А. Г. Инструментальные средства имитационного моделирования предметно-ориентированных распределенных вычислительных систем / А. Г. Феоктистов, А. С. Корсуков, Ю. А. Дядькин // Системы управления, связи и безопасности. -2016. - №4. – С.30-49.

Электронные ресурсы

7. Замятина О. М. Моделирование сетей [Электронный ресурс] : учебное пособие / О. М. Замятина. — Томск : Томский политехнический университет, 2012. — 160 с. — Режим доступа: http://www.iprbookshop.ru/34683.html (дата обращения 23.04.2019).

8. Одноранговая сеть [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%80%D0%B0%D0%BD%D0%B3%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C (дата обращения 23.04.2019).

9. Симулятор NeuroGrid [Электронный ресурс]. – Режим доступа: https://sourceforge.net/projects/neurogrid/ (дата обращения 23.04.2019).

10. Симулятор PeerSim [Электронный ресурс]. – Режим доступа: http://peersim.sourceforge.net (дата обращения 23.04.2019).

11. Универсальная среда для симуляции SimGrid [Электронный ресурс]. – Режим доступа: https://simgrid.org/ (дата обращения 23.04.2019).

12. Черняева С. Н. Имитационное моделирование систем [Электронный ресурс] : учебное пособие / С. Н. Черняева, В. В. Денисенко ; под ред. Л. А. Коробова. — Воронеж : Воронежский государственный университет инженерных технологий, 2016. — 96 с. — Режим доступа: http://www.iprbookshop.ru/50630.html (дата обращения 23.04.2019).

13. An implementation of the Chord protocol for Peersim [Электронный ресурс]. – Режим доступа: http://peersim.sourceforge.net/#download (дата обращения 23.04.2019).

14. Gartner research and advisory company [Электронный ресурс]. – Режим доступа: https://www.gartner.com/en (дата обращения 23.04.2019).

15. P2P Simulation and Reality. Application Technology Workshop [Электронный ресурс]. – Режим доступа: https://apan.net/meetings/.../p2pgrid/SamJoseph-P2P-SimulationAndReality.ppt (дата обращения 23.04.2019).

16. Peer-to-Peer Architecture [Электронный ресурс]. – Режим доступа: https://www.techopedia.com/definition/454/peer-to-peer-architecture-p2p-architecture (дата обращения 23.04.2019).

17. Structured P2P Networks [Электронный ресурс]. – Режим доступа: inst.eecs.berkeley.edu/~cs268/sp03/notes/Lecture22.pdf (дата обращения 23.04.2019).

Литература на иностранном языке

18. Brown A., Kolberg M. Tools for Peer to Peer Network Simulation, IRTF P2pRG, 2006.

19. Chen Y-F. and others. When is P2P Technology Beneficial for IPTV Services? Proc. of ACM NOSSDA, 2007.

20. Dragoni N. Introduction to P2P Computing, Embedded Systems Engineering DTU Compute, 2014.

21. Ebrahim M. and others. Peer-to-Peer Network Simulators: an Analytical Review, Asian Journal of Engineering Science and Technology AJEST, Vol. 2(1), 2012.

22. Joseph S. An Extendible Open Source P2P Simulator, P2P Journal, November, 2003.

23. Mahanta A. and Theeramunkong T. A Comparative Simulation Analysis of P2P System Architectures. The 4th International Symposium on Parallel and Distributed Computing (ISPDC'05), Lille, 2005, pp. 51-57.

24. Rescorla E. Introduction to distributed hash tables, IETF-65 Technical Plenary, 2006.

25. Ting N. A Generic Peer-to-Peer Network Simulator. In: Proceeding of the 2003-2004 Grad Symposium, CS, Dept., University of Saskatchewan, 7-8, 2003.

ПРИЛОЖЕНИЕ А

Код реализации класса ChordInitializer (Java)

```
package peersim.chord;
import java.math.BigInteger;
import java.util.Random;
import peersim.config.Configuration;
import peersim.core.CommonState;
import peersim.core.Network;
import peersim.core.Node;
import peersim.dynamics.NodeInitializer;
public class ChordInitializer implements NodeInitializer {
	private static final String PAR_PROT = "protocol";
	private int pid = 0;
	private ChordProtocol cp;
	public ChordInitializer(String prefix) {
		pid = Configuration.getPid(prefix + "." + PAR_PROT);
	}
	public void initialize(Node n) {
		cp = (ChordProtocol) n.getProtocol(pid);
		join(n);
	}
	public void join(Node myNode) {
		Random generator = new Random();
		cp.predecessor = null;
		// search a node to join
		Node n;
		do {
			n = Network.get(generator.nextInt(Network.size()));
		} while (n == null || n.isUp() == false);
		cp.m = ((ChordProtocol) n.getProtocol(pid)).m;
		cp.chordId = new BigInteger(cp.m, CommonState.r);
```

```
                ChordProtocol cpRemote = (ChordProtocol) n.getProtocol(pid);

                Node successor = cpRemote.find_successor(cp.chordId);
                cp.fails = 0;
                cp.stabilizations = 0;
                cp.varSuccList = cpRemote.varSuccList;
                cp.varSuccList = 0;
                cp.succLSize = cpRemote.succLSize;
                cp.successorList = new Node[cp.succLSize];
                cp.successorList[0] = successor;
                cp.fingerTable = new Node[cp.m];
                long succId = 0;
                BigInteger lastId = ((ChordProtocol) Network.get(Network.size() - 1)
                                .getProtocol(pid)).chordId;
                do {
                        cp.stabilizations++;
                        succId = cp.successorList[0].getID();
                        cp.stabilize(myNode);
                        if (((ChordProtocol)
cp.successorList[0].getProtocol(pid)).chordId
                                        .compareTo(cp.chordId) < 0) {
                                cp.successorList[0] = ((ChordProtocol)
cp.successorList[0]
                                                .getProtocol(pid)).find_successor(cp.chordId
);
                        }
                        // controllo di non essere l'ultimo elemento della rete
                        if (cp.chordId.compareTo(lastId) > 0) {
                                cp.successorList[0] = Network.get(0);
                                break;
```

```
                }
            } while (cp.successorList[0].getID() != succId
                    || ((ChordProtocol)
cp.successorList[0].getProtocol(pid)).chordId
                            .compareTo(cp.chordId) < 0);
            cp.fixFingers();
        }
}
```

ПРИЛОЖЕНИЕ Б

Код интерфейса MessageHandler (Java)

```
public boolean handleMessage(Message message, Node
node)
throws Exception;
public boolean injectMessage(Message message,
Node node)
throws Exception;
```

Код реализации метода GnutellaMessageHandler представлен ниже.

```
public boolean handleMessage(ContentMessage message, Node
node)
throws Exception
{
// 1. Обновить статистику пересылки сообщений
Network.o_message_transfers++;
// 2. Проверить, было ли сообщение просмотрено
if (seenMessage(message, node) == true
{ return false; }
// 3. Обновление статистики активности узла
node.setActive();
// 4. Удалить сообщение из входящего ящика узла.
node.removeFromInbox(message);
// 5. Проверить на местное соответствие.
checkLocalMatch(message, node);
// 6. Проверить сообщение счетчика TTL.
if (checkMessageTTL(message, node) == false)
{ return false; }
// 7. Сообщить всем связанным узлам.
// (except the one we received from )
return forwardMessage(message, node);
}
```

w