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)
02.03.03 Математическое моделирование и администрирование информационных систем
(код и наименование направления подготовки, специальности)
Технология программирования
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Компьютерные модели анализа архитектур пиринговых систем

Студент	А.А. Упатов	
	(И.О. Фамилия)	(личная подпись)
Руководитель	А.В. Очеповский	
	(И.О. Фамилия)	(личная подпись)
Консультанты	К.А. Селиверстова	
	(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

Тема бакалаврской работы – «Компьютерные модели анализа архитектур пиринговых систем».

Цель работы: описать архитектуры и апробировать технологии моделирования архитектур децентрализованных пиринговых систем с использованием программных симуляторов.

Объект исследования: процесс функционирования пиринговых систем.

Предмет исследования: компьютерные модели архитектур распределенных пиринговых систем.

Работа посвящена исследованию актуальной проблеме в современной информатике – разработка методов и способов функционирования распределенных пиринговых систем, применение которых по результатам исследования аналитических обзоров в будущем будет только расти. Поэтому апробация методов компьютерного моделирования распределенных пиринговых систем является актуальной задачей.

В первой главе рассматривается проблема роста интернет трафика и вклад распределенных пиринговых систем в его формирование, а также приводится определение и классификация пиринговых систем.

Во второй главе анализируются архитектуры распределенных пиринговых систем и приводятся математические модели, описывающие эти архитектуры.

В третей главе приведено описание вычислительного эксперимента по моделированию архитектур распределенных пиринговых.

Результатами работы является апробированная технология компьютерного моделирования распределенных пиринговых систем с использованием P2P симуляторов.

Пояснительная записка выполнена на 53 страницах, включает 26 рисунков. Список используемых источников включает 23 источника, из них 9 на иностранном языке.

Abstract

The theme of bachelor's work is “Computer models of the analysis of peering systems”.

Objective: to describe the architecture and test the technology of modeling architectures of decentralized peer-to-peer systems using software simulators.

The object of study: the process of functioning of the peering systems.

Subject of research: computer models of distributed peer systems architecture.

The work is devoted to the study of the actual problem in modern computer science - the development of methods and approaches for the functioning of distributed peer-to-peer systems, the application of which according to the results of the research of analytical reviews will only grow in the future. Therefore, approbation of computer simulation methods for distributed peer-to-peer systems is an urgent task.

The first chapter deals with the problem of the growth of Internet traffic and the contribution of distributed peering systems to its formation, and also provides definition and classification of peering systems.

In the second chapter, architectures of distributed peering systems are analyzed and mathematical models describing these architectures are given.

The third chapter describes the computational experiment on the simulation of distributed peering architectures.

The results of the work are a proven technologies of computer simulation of distributed peering systems using P2P simulators.

Explanatory note was made on 53 pages, includes 26 figures. The list of sources used includes 23 sources, 9 of them in a foreign language.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня можно с уверенностью утверждать, что эпоха одноядерных вычислений ушла в прошлое. Распределенные базы данных, вычислительные кластеры, системы IP телевидения, массовые онлайн игры с одной стороны и бытовые устройства – с другой – все являются примерами многоядерных или многопроцессорных систем, причем огромная часть приведенных систем строится по распределённым архитектурам. Распределенные системы стали эквивалентом экономической глобализации в мире компьютеров и являются сегодня часто единственным возможным решением ряда задач.

За последние два десятилетия распределенные системы приняли множество видов и форм. Одним из представителей первого поколения стали компьютерные кластеры, цель которых состояла в предоставлении экономически эффективной альтернативы дорогостоящим параллельным машинам. Файловые серверы были первыми, кто эволюционировал на основе кластерной модели распределенной системы для удовлетворения растущего спроса на хранилища данных. Одноранговые системы появились как альтернативное решение технологии клиент-сервер и ее разновидности – глобальной сети Интернет. В дальнейшем облачные вычисления превратили распределенные системы в утилиту, предлагающую вычисления и хранение в качестве сервисов через Интернет. Одним из новых и наименее ожидаемых бенефициаров облачных вычислений станет мобильный мир смартфонов и персональных устройств, возможности которых можно расширить с помощью разгрузки вычислений. С другой стороны, беспроводные сети инициировали дальнейшее развитие распределенных систем в сенсорных сетях и встроенных устройствах. Кроме того, вопросы цифровой трансформации, поставленные в [1, 2] также тесно связаны темой работы.

Вместе с тем множеством реализаций распределенных систем породили широкий спектр проблем и соответствующих исследований, связанных с концептуальными и прикладными задачами. Одной из остающихся актуальной проблемой является исследование особенностей архитектурных решений для

распределенных пиринговых систем. Проблема усугубляется тем, что количество узлов сети может превышать 10^6, что значительно затрудняет аналитические методы анализа распределённых пиринговых систем. Поэтому тема работы является **актуальной** как с научной, так и с инженерной точек зрения.

Объектом исследований являются процессы функционирования распределённых пиринговых систем. **Предметом** исследований являются компьютерные модели пиринговых систем, построенных по различным архитектурам.

Целью бакалаврской работы является апробация способов компьютерного моделирования распределенных пиринговых систем с использованием P2P симуляторов.

Для достижения поставленной цели были выделены следующие задачи:

1. Провести анализ перспектив развития рынка распределенных пиринговых систем на основе изучения литературы.
2. Выполнить обзор видов и моделей архитектур построения распределенных пиринговых систем.
3. Провести моделирование структур распределенных пиринговых систем в использованием P2P симуляторов.

Новизна исследования заключается апробации технологий моделирования распределенных пиринговых систем с использованием P2P симуляторов.

Бакалаврская работа состоит из введения, трех глав, заключения, списка используемых источников.

В первой главе анализируются современное состояние P2P систем и прогнозы по их развитию. Также приводится определение и классификация пиринговых систем

Во второй главе рассматриваются виды и модели архитектур построения распределенных пиринговых систем.

В третьей главе проводится компьютерное моделирование структур распределенных пиринговых систем с использованием P2P симулятора PeerSim.

Бакалаврская работа выполнена по заданию Центра IT Student.

1 ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОНЯТИЯ РАСПРЕДЕЛЕННЫХ ПИРИНГОВЫХ СИСТЕМ

1.1 Понятие и классификация распределённых пиринговых сетей

Технологии глобальной сети Интернет и связанные с ней технологии интернета вещей (Internet of Thinks - IoT), IP телевидения, социальных сетей и т.д. прочно вошли в повседневную жизнь человечества.

Согласно индексу развития визуальных сетевых технологий (Visual Networking Index, VNI), опубликованному корпорацией Cisco [12] в 2022 году объем интернет трафика превысит сумму трафика за все предыдущие годы. Годовой глобальный IP-трафик достигнет 4,8 зеттабайт (ZB) к 2022 году, или 396 эксабайт (EB) в месяц (рис. 1.1). В 2017 году годовой показатель глобального трафика IP составлял 1,5 ZB в год, или 122 EB в месяц.

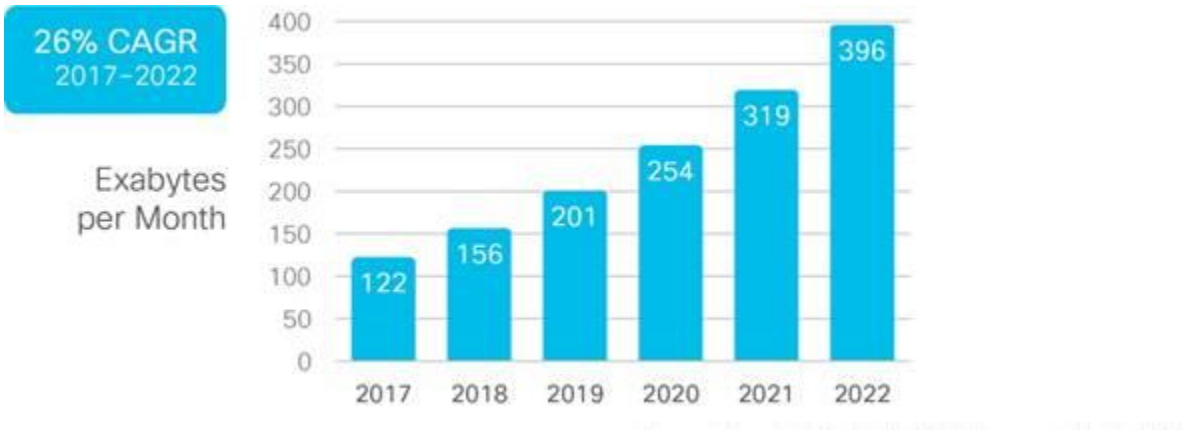


Рисунок 1.1 – Прогноз ежемесячного интернет трафика

Аналитика показывает, что каждые пять лет глобальный IP-трафик увеличивается втрое. К 2020 году ежемесячный IP-трафик достигнет 85 ГБ на одного человека, в то время как в 2017 году этот показатель составлял всего 29 ГБ. Основную долю интернет трафика будет составлять видео: 80% в 2022 году, по сравнению с 70% в 2017. В целом в 2022 году на видео, игры и мультимедиа придется более 85% всего интернет трафика (рис 1.2).

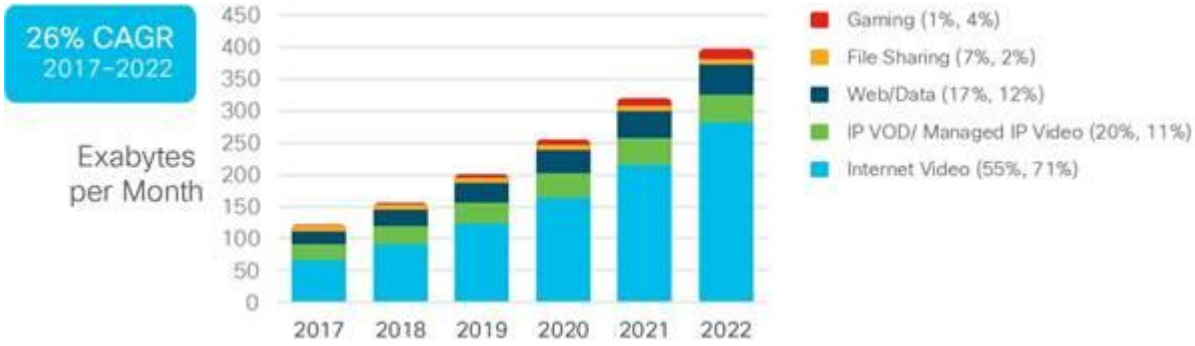


Рисунок 1.2 – Прогноз глобального видео трафика по сегментам

Для хранения огромных объемов данных, их передачи и обработки классические технологии, построенные по архитектурам клиент-сервер, становятся малопригодными. В последние годы большой интерес с научной и практической точек зрения представляют распределенные информационные системы в целом и пиринговые системы в частности.

Приведем определение распределенной информационной системы (РИС). РИС – это множество вычислительных узлов, взаимодействующих для выполнения общих задач и соединенных посредством коммуникационной сети [23].

В соответствии с приведённым определением к распределённым системам предъявляются следующие требования:

- для автономной работы узлов последние должны быть независимыми;
- узлы должны иметь связь с другими узлами (прямую или косвенную) и поэтому должны соединяться через коммуникационную сеть;
- для решения общих задач сети должен присутствовать механизм, согласующий деятельность узлов.

Распределенные информационные системы могут классифицироваться по многим показателям. Одной из важнейших классификаций является классификация РИС по архитектурному признаку [18]. В соответствии с данной классификацией выделяются следующие классы РИС:

- Клиент-сервер (Client-Server – CS);

– Ведущий-ведомый (Master-Slaves – MS);

– точка-точка (peer-to-peer – P2P).

Модель «клиент-сервер» (рис. 1.3) является естественным расширением удаленного вызова процедур – Remote Procedure Call (RPC). В начале 90-х годов 20 века такие технологии как CORBA расширили изначальный RCP механизм до объектно-ориентированного представления. Механизмы регистрации дали возможность динамической идентификации адреса сервера и избавили от необходимости разбора IP адреса в программе. Это важнейшее нововведение позволило повысить переносимость и масштабируемость распределенных систем.

Однако, модель «клиент-сервер» подразумевает использование протокола «точка-точка» (point-to-point), посредством которого клиент инициирует взаимодействие с сервером. Роль сервера относительно проста: обработка запроса клиента и формирование ответа клиенту.

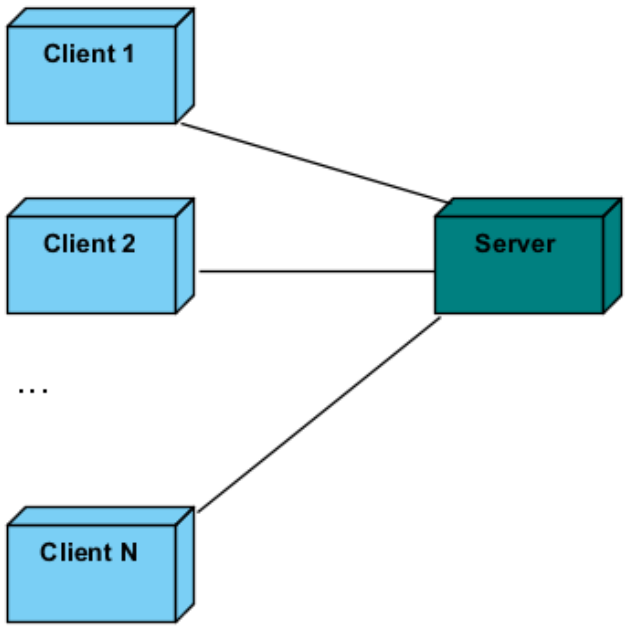


Рисунок 1.3 – Модель «Клиент-сервер»

Очевидно, что весь контент распределенной системы, построенной по технологии «Клиент-сервер», находится на сервере. Следовательно, для повышения отказоустойчивости системы сервер должен реплицироваться на

другие машины. И такая технология в настоящее время широко используется для больших серверных приложений.

Разновидностью модели «клиент-сервер» является модель «ведущий-ведомые» (Master-Slaves). В этой модели (рис. 1.4) роль распределителя задач берет на себя узел Master, который распределяет задачи между подчиненными узлами Slave. Один из подчиненных узлов берет на себя задачу, обеспечивая взаимодействие по схеме «точка-точка». Логика работы приложения реализуется Master узлом. Аналогично модели «клиент-сервер» контекст выполнения приложения и соответствующие данные регулярно должны резервироваться. В случае возникновения сбоя контент грузится с последней резервной точки и вычисления продолжаются.

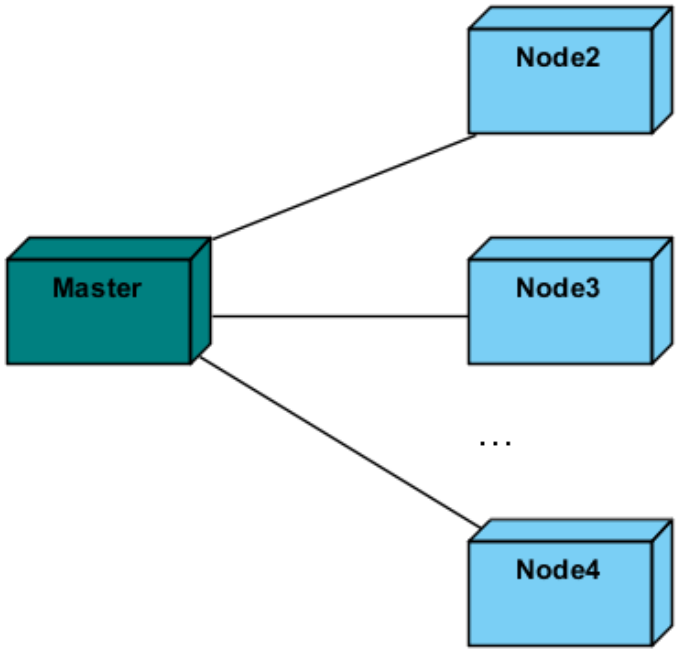


Рисунок 1.4 –Модель «Master-Slaves»

P2P-архитектуры являются альтернативным подходом к традиционным распределенным схемам, основанным на архитектуре «клиент-сервер», которые были особенно эффективны для поиска ресурсов в больших конфигурациях, состоящих из миллионов узлов.

Узлы P2P систем могут образовывать различные сетевые топологии, среди которых можно выделить: кольцо, звезда, дерево, полный или частичный граф. Использование определенных топологий позволяет определить постоянное положение элемента в системе и обеспечить маршрутизацию сообщений между элементами. Пример P2P системы, построенной по топологии неполного графа приведен на рисунке 1.5.

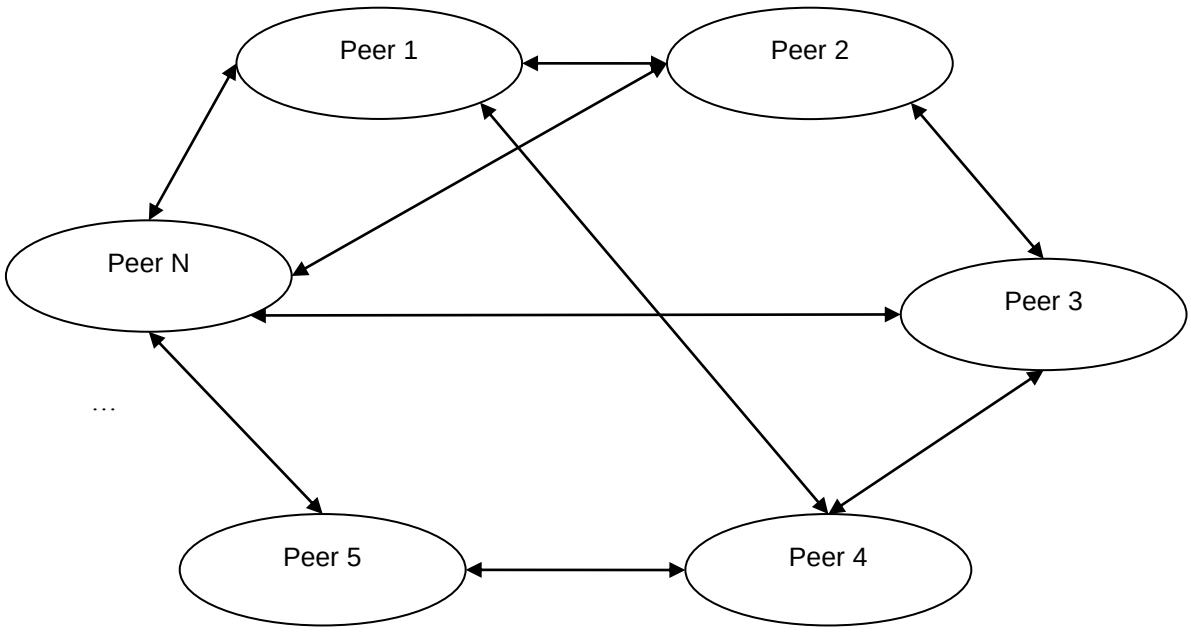


Рисунок 1.5 – Модель P2P. Неполный граф

1.2 Классификация P2P систем

Также как и РИС, пиринговые системы могут быть классифицированы по разным критериям. Основными классификациями согласно [18] являются классификации по степени централизации и структуре.

По степени централизации P2P системы делятся на:

- централизованные (Centralized);
- частично централизованные (Partially centralized);
- распределенные (Distributed).

В централизованной архитектуре выделяется группа узлов-серверов, выполняющих операции регистрации узлов, распределения ресурсов по узла,

поиска узлов и т.д. Для обеспечения высокой пропускной способности между центральными узлами используются высокоскоростные соединяя на основе LAN-технологий.

Централизованные пиринговые сети обладаю следующими преимуществами:

- организация и хранение метаданных сети;
- операций индексирования, упорядочивания, поиска ресурсов в сети выполняются с высокой скоростью;
- сеть характеризуется небольшим количеством поисковых запросов;
- балансировка нагрузки выполняется относительно просто;
- легкость управления центральными узлами.

К недостаткам централизованных P2P систем можно отнести:

- недостаточная масштабируемость рассматриваемой архитектуры;
- наличие единственной точкой отказа – группа центральных узлов;
- высокие требования по памяти и быстродействию к серверным узлам.

Что бы в некоторой мере устранить приведенные недостатки централизованных P2P сетей была разработана архитектура частично централизованных системы. Основной идеей этой архитектуры является то, что вместо одной группы узлов-серверов критические задачи выполняются несколькими группами серверных узлов. При двухуровневой иерархии в частично централизованных сетях выделяются несколько групп суперузлов. Суперузел – это мощный узел, соединённый высокоскоростными каналами связи с другими суперузлами. В свою очередь, простые узлы соединяются с одной или несколькими группами суперузлов обычными линями связи.

Однако частично централизованные системы имеют свои недостатки: сложность алгоритмов обработки метаданных и учета присоединения и отсоединения узлов к сети. Практика показывает, что частично централизованные пиринговые системы часто неэффективны, когда число

узлов становится большим или идет интенсивный процесс присоединения-отсоединения пиров из сети.

Для решения вышеупомянутых проблем применяются децентрализованные P2P системы, в которых не используется никакая иерархия узлов. В этой архитектуре каждый пир входит в состав глобального оверлея P2P сети и может хранить метаданные о ресурсах только своих близлежащий соседях. Кроме того, для повышения устойчивости сети в целом при отказе отдельного узла информация о метаданных реплицируется всем непосредственным соседям по оверлею.

Исследования показывают, что децентрализованные пиринговые сети могут быстро восстанавливаться после большого числа разнообразных сбоев в силу того, что в них отсутствуют единые точки отказа.

К недостаткам децентрализованных P2P сетей можно отнести сложность децентрализованных алгоритмов.

1.3 Оверлеи P2P сетей

Как уже отмечалось ранее P2P системы могут образовывать различные логические структуры – оверлейные сети. В соответствии с [18], оверлеи классифицируются на структурированные (structured) и неструктурированные (unstructured).

В неструктурированных оверлеях не определена никакая топология узлов и они строятся в недетерминированном порядке. Неструктурированные оверлеи достаточно просты в построении. Когда узел присоединяется к сети, то он просто соединяется с любым существующим узлом и использует его информацию о соседях для настройки соединения. Никакая информация о присоединенном узле не передается в сеть, за исключением информации небольшого объема о маршрутах к ближайшим соседям. Указанное обстоятельство делает неструктурированные оверлеи приемлемым решением в условиях большой текучести узлов (присоединение/отсоединение узлов).

Структурированные оверлеи строятся так, чтобы узлы образовывали определенную топологию. Коренным отличием структурированных оверлеев от неструктурированных является то, что узел не может произвольно менять свое местоположение. Именно топология оверлея определяет множество соседей, с которыми может общаться выбранный узел. Кроме того, топология оверлея определяет пространство имен, в котором каждому узлу присваиваться определённый уникальный идентификатор. Каждый узел поддерживает таблицу маршрутизации, связывающую идентификаторы и адреса соседних узлов в соответствии с топологией оверлея. Данная технология позволяет определять список узлов, за которые данный узел отвечает при поиске.

1.4 Беспроводные сети

Неструктурированные (Ad hoc) беспроводные сети обмениваются данными по беспроводным каналам, не имеющим постоянной архитектуры. Эти сети могут быть развернуты достаточно легко и быстро. Поэтому ad hoc сети применимы во многих областях таких как мониторинг, военные, специальные и спасательные операции.

1.4.1 Неструктурированные беспроводные сети

Вычислительные узлы беспроводной сети оснащаются средствами беспроводной связи и коммуникации. Беспроводные сети могут быть построены как инфраструктурные (infrastructured) или неструктурированные (Ad hoc). В инфраструктурных сетях стационарная проводная магистраль, состоящая из маршрутизаторов, точек доступа и серверов, используется для обеспечения связи между узлами сети, как показано на рис. 1.6. На рисунке пять мобильных хостов (МН) получают доступ к магистрали, которая состоит из трех маршрутизаторов (R) и сервера. В такой сети мобильные или стационарные узлы не обмениваются данными напрямую. В случае аварии

магистраль может не функционировать, что приводит к полной потере связи между хостами. Сотовые сети являются одним из типов инфраструктурных беспроводных сетей [19].

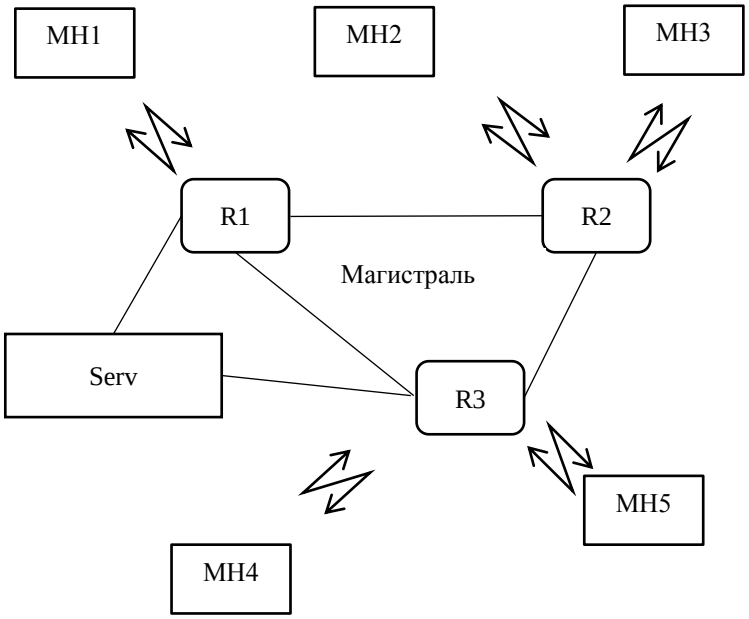


Рисунок 1.6 – Инфраструктурная беспроводная сеть

Неструктурированная (Ad hoc) беспроводная сеть, в свою очередь, не имеет какой-либо фиксированной структуры, и каждый узел участвует в обмене данными путем пересылки сообщений, как показано на рисунке 1.X. Ad hoc беспроводная сеть может состоять из разнородных или однородных узлов, которые имеют одинаковые средства беспроводной связи. Обычно используются два типа беспроводных специальных сетей: мобильные неструктурированные сети (mobile ad hoc networks) и беспроводные сенсорные сети (wireless sensor networks).

1.4.2 Мобильные неструктурированные сети

MANET сеть состоит из автономных мобильных узлов, которые могут свободно и произвольно перемещаться и осуществлять связь без какой-либо коммуникационной инфраструктуры. Примерами MANET являются

аварийные (поисково-спасательные) операции, операции по оказанию помощи при бедствиях, транспортные сети и военные сети.

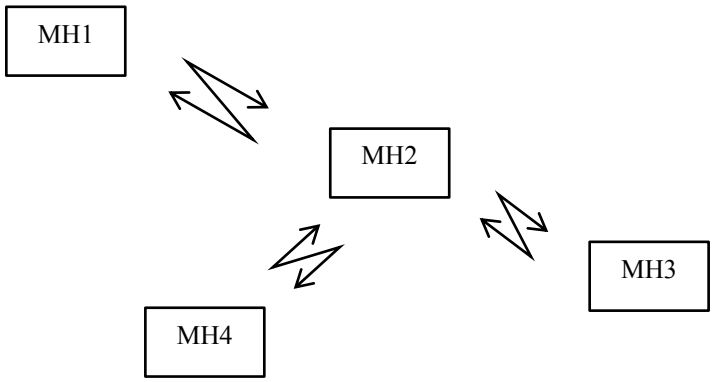


Рисунок 1.X – Неструктурированная беспроводная сеть

Каждый узел MANET является автономным и действует как хост и маршрутизатор для передачи сообщений. Если два узла MANET находятся в пределах зоны радиосвязи друг с другом, то они могут связываться напрямую. В противном случае они могут использовать многоскачковый (multi-hop) обмен данными, где промежуточные узлы используются для передачи сообщения между двумя узлами. Поскольку в MANET сети нет инфраструктуры и централизованного контроля, управление сетью осуществляется отдельными узлами. Однако MANET сетям присуще следующие проблемы:

- маршрутизация: маршрутизация с несколькими переходами обычно является единственным выбором связи в MANET. Кроме того, поскольку топология является динамической, таблицы маршрутизации необходимо часто обновлять. На рисунке 1.7 (а) показан пример, когда узлы u и v, перемещающиеся близко друг к другу, образуют линию связи, когда они обнаруживают друг друга посредством сообщений маяка (beacon messages). На рисунке (б)

третий узел w приближается к узлу u. В этот момент узел u передает данные о своем новом соседе w узлу v.

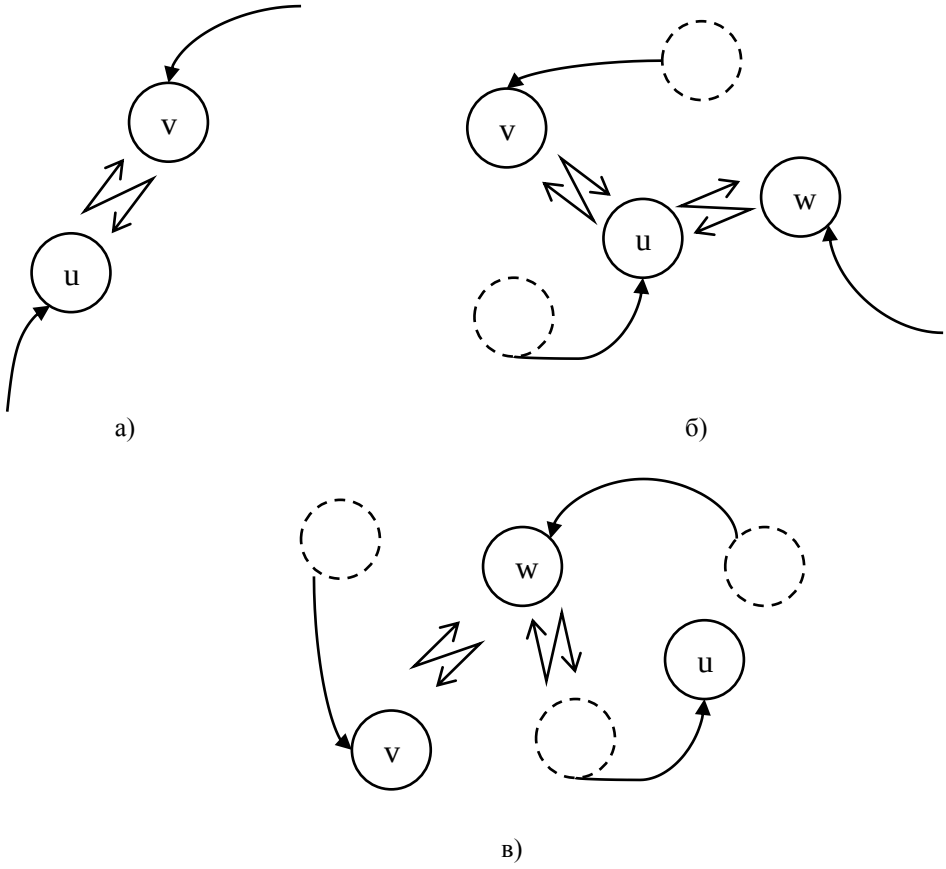


Рисунок 1.7 – Пример взаимодействия в MANET сети

Поскольку узел v не обнаружил узел w, он определяет, что не может напрямую связаться с узлом w, но может сделать это через узел u, поэтому он обновляет свою таблицу маршрутизации. На рисунке (в) по мере перемещения узлов ситуация меняется. Теперь узел u может достигать узла v через узел w, и таблицы маршрутов снова обновляются. Маршруты в мобильной сети могут периодически обновляться, так что всегда существует маршрут между любой парой отправитель-получатель, и в этом случае алгоритмы маршрутизации называются проактивными. В протоколах реактивной маршрутизации маршруты формируются только при необходимости. Оба метода имеют свои преимущества и недостатки. Маршрутизация через динамическую магистраль

с использованием связанного доминирующего набора является широко используемым методом;

- управление топологией: управление топологией является сложной проблемой в MANET сетях, где соединения быстро и непредсказуемо изменяются. Распространенным методом управления MANET топологией является построение графа с более редкими коммуникационными связями. Также кластеризация узлов обеспечивает эффективный метод управления ресурсами и маршрутизации в MANET;
- доступ к каналу (Channel Access): беспроводные каналы, через которые обмениваются узлы, подвержены шуму, затуханию и помехам. Когда два или более узла находятся в пределах зоны связи друг с другом, их одновременные передачи могут создавать помехи, вызывая коллизии пакетов. Узел может не успеть обнаружить передачу от соседа или правильно принять пакет резервирования и, следовательно, может начать передачу, что приводит к конфликтам. Описанная проблема является основным источником коллизий в MANET. Протоколы контроля среднего уровня должны обеспечивать механизмы для уменьшения коллизий. Использование отдельных каналов для пакетов управления и данных, а иногда и отдельных каналов для разных узлов является распространенным методом уменьшения коллизий. Также используются методы, основанные на синхронизации. Эти методы выделяют временные интервалы для обмена данными между узлами для предотвращения коллизий.
- качество обслуживания (Quality of Service – QoS): в MANET сетях из-за мобильности узлов и проблем с доступом к каналу обеспечение QoS также затруднено с точки зрения ограниченной задержки и минимальной полосы пропускания. Эта проблема особенно актуальна для мультимедийных приложений;

- безопасность: общая беспроводная среда открыта как для легитимных пользователей и злоумышленников. Отсутствие защищенных маршрутизаторов под централизованным контроллером приводит к тому, что злоумышленники могут вывести из строя MANET сеть. Эффективным способом достижения безопасности является использование шифрования с открытым ключом.

1.4.3 Беспроводные сенсорные сети

Беспроводная сенсорная сеть (Wireless Sensor Network – WSN) состоит из нескольких сенсорных географически распределенных узлов-датчиков. Каждый датчик имеет возможность связи по беспроводному каналу и способность осуществлять обработку сигналов. Многопереходная связь обычно является единственным выбором в WSN. Обычно некоторый центральный узел, называемый центром слияния, собирает все данные с датчиков и, возможно, выполняет вторичную обработку данных датчиков. Далее центр слияния может передавать данные на другой компьютер для окончательной обработки данных. Прием и передача данных осуществляется в форме радиосигналов.

В сенсорной сети выполняется связь многие-к-одному, когда данные от датчиков передаются в центр слияния для дальнейшей обработки, как показано на рис. 1.8. Узлы отправляют свои данные ближайшему соседу. Узел, который получил данные от соседа, может устранять избыточность данных перед их дальнейшей отправкой. Этот процесс называется агрегацией данных. Например, только среднее значение температуры, полученное от ряда датчиков в конкретной области, может быть передано в корню узла.

В процессе проектирования и реализации WSN возникают следующие проблемы:

- масштабируемость: сенсорная сеть может состоять из сотен или тысяч узлов. Протоколы и алгоритмы маршрутизация должны быть масштабируемыми;

- отказоустойчивость: отказ одного узла не должен влиять на общую работу сенсорной сети, и возможность подключения к сети должна поддерживаться с использованием новых каналов, если это необходимо;
- развертывание узлов: узлы могут быть развернуты случайным или детерминированным образом с целью обеспечения максимального охвата области, обслуживаемой датчиками. Также при развертывании должен реализоваться заданный граф сети;

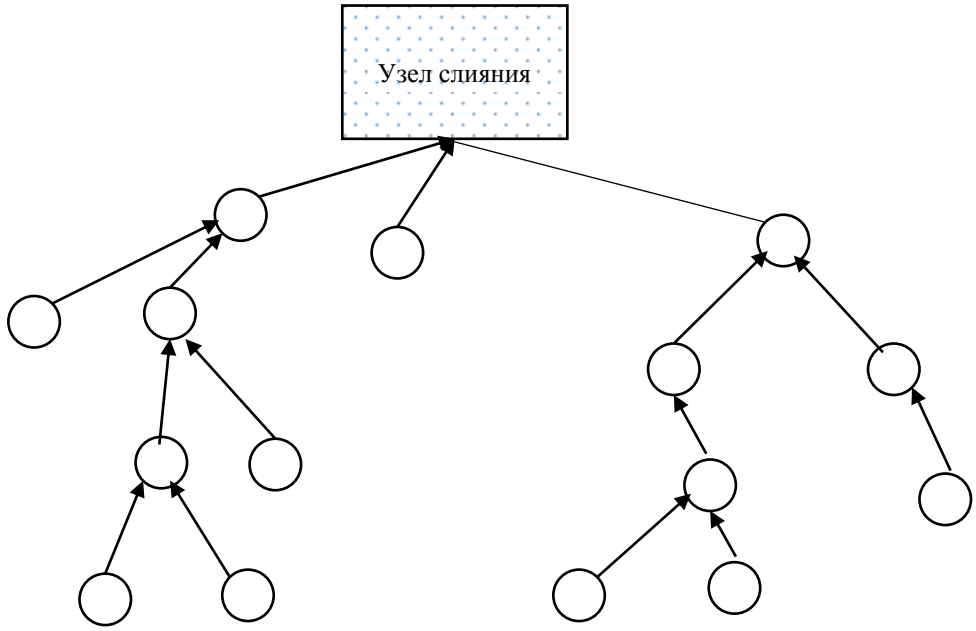


Рисунок 1.8 – Слияние в MANET сети

Управление питанием: датчики сенсорной сети используют батареи с ограниченным сроком службы, и замена этих батарей может быть затруднена или невозможна по многим причинам. Передача и прием сообщений потребляют гораздо больше энергии, чем локальные вычисления, поэтому протоколы и алгоритмы должны быть рассчитаны на работу с минимальным количеством сообщений. Эффективный способ экономии энергии – это перевод некоторых датчиков в спящий режим, когда не требуется действий.

Важным требованием в этих методах будет обеспечение полного охвата датчиков в процессе пробуждения.

Выводы по первой главе

Таким образом, по результатам первой главы можно сделать следующие выводы:

- в перспективе 5-10 лет будет наблюдаться расширение области применения пиринговых сетей, вызванной увеличением мультимедийного интернет трафика и развитием таких приложений, как IP телевидение;
- пиринговые сети могут строиться по централизованной, частично централизованной и децентрализованной архитектуре;
- логически узлы пиринговых сетей могут объединяться по различным оверлеям, которые принято делить на структурированные и неструктурированные;
- при построении децентрализованных сетей, таких как MANET и WSN, необходимо решать ряд присущих этих сетям проблем.

2 МОДЕЛИ ТОПОЛОГИЙ ПИРИНГОВЫЙ СЕТЕЙ

2.1 Основные модели топологий P2P сетей

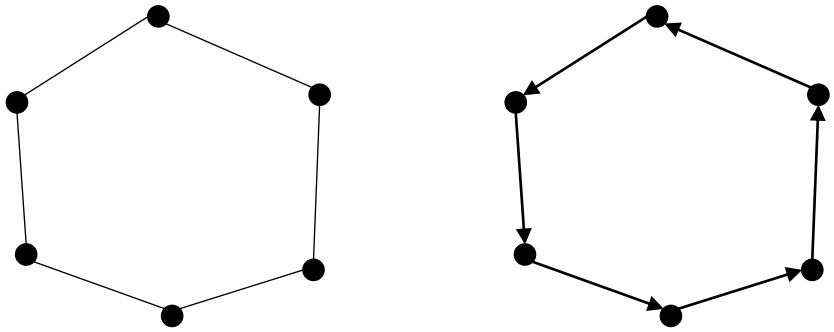
Отнесем к базовым топологиям следующие топологии [11]:

- кольцо;
- дерево;
- лес;
- решетка;
- тор;
- гиперкуб.

Кольцевая топология активно используется в распределенных вычислениях. В некоторых сетях узлы действительно образуют кольцо, в других кольцо используется как встроенная управляющая структура.

Определение. Кольцо – неориентированный связный регулярный граф степени два.

Для топологи кольцо истинно следующее утверждение: существует такое взаимно однозначное соответствие между множеством вершин V и множеством натуральных чисел $\{0, \dots, \mathrm{N}-1\}$, что соседями вершины, соответствующей числу i, являются вершины с номерами $i-1$ и $i+1$ (по модулю N). Пример сети, построенной по топологии кольцо представлен на рис. 2.1.



а) б)

Рисунок 2.1 – Распределённая сеть, построенная по топологии
а) кольцо б) ориентированное кольцо

Топологии типа «**дерево**» также являются часто используемыми при построении распределенных сетей как непосредственно, так и опосредовано.

Определение. Дерево – неориентированный связный ациклический граф.

Локальные топологии децентрализованных пиринговых сетей могут образовывать корневые деревья – деревья, в которых роль корня *r* выполняет один из узлов сети.

Важное значение имеют также топологии, построенные на основе остовных деревьев разного типа:

- остовные деревья малого диаметра;
- остовные деревья минимального веса;
- остовные деревья ограниченной степени;
- деревья поиска в глубину.

Следующей важной моделью топологий распределенных систем является обобщение понятия дерева – лес. Лес – это граф, который состоит из некоторого числа изолированных деревьев. Пример сети, построенной по топологии лес, приведен на рисунке 2.2. На рисунке показан пример IP-TV сети, в которой видео поток отдается тремя субпотоками через узлы-партнеры [10]

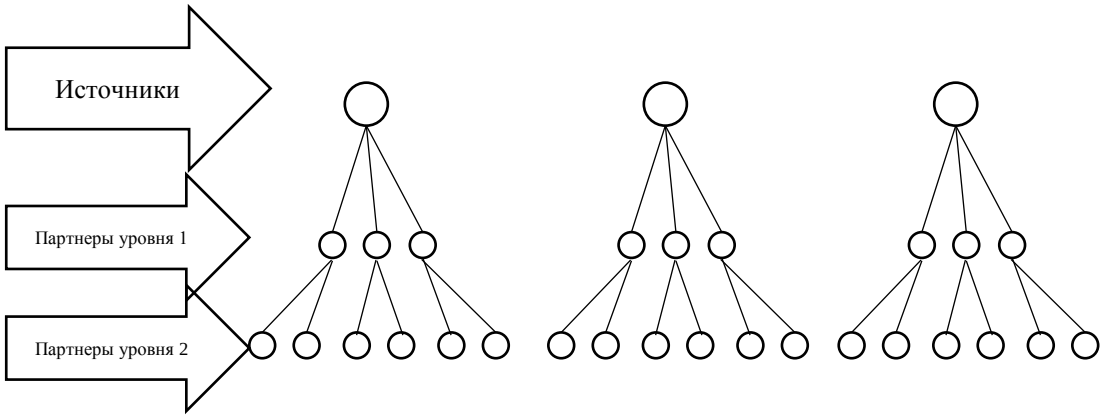
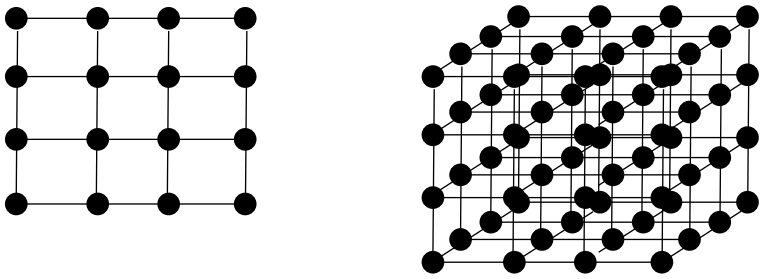


Рисунок 2.2 – Пример сети, построенной по топологии лес

Большое количество распределенных вычислительных систем строится на основе 2D и 3D **решеток**. Можно дать следующее определение 2D решетки [11].

Определение. Решеткой размера $n \times n$ называется граф, имеющий n^2 вершин, которые взаимно однозначно сопоставлены элементам из множества $\{\ i,j\ : 0 \leq i,j < n\}$ так, что вершины, помеченные парами (i,j) и (i',j'), являются смежными в том и только том случае, когда $(i = i') \wedge\ j = j' \pm 1$ или $(i = i' \pm 1) \wedge\ j = j'$ (операции выполняются по модулю *N*). Пример сети, построенной по топологии решетка приведен на рисунке 2.3.



а) б)

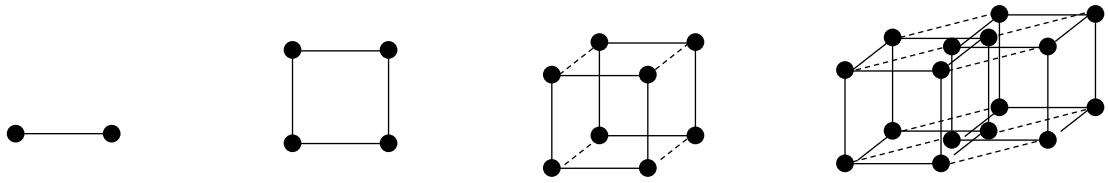
Рисунок 2.3 – Сеть, построенная по топологии а) 2D и б) 3D решетки (n=4)

В ряде случаев [18] децентрализованные системы могут строиться по топологии «**тор**». Под тором или тороидальным графом понимается граф, который можно уложить на торе.

Наряду с решетками и торами распределённые системы могут строится с использованием топологии гиперкуба.

Определение. Гиперкубом размерности *n* называется регулярный граф, имеющий *2n* вершин, $2^{n-1}n$ рёбер, причем в каждой вершине сходится ровно *n* рёбер. На рисунке 2.4 приведены примеров сетей, построенных по топологии гиперкуба.

Ряд децентрализованных пиринговых систем строится по специфическим топологиям [18]: бабочка, графы де Брюйна, деревья Плакстона и т.п.



а) б) в) г)

Рисунок 2.4 – Сеть, построенная по топологии гиперкуб

а) *n=1* б) *n=2* в) *n=3* г) *n=4*

Топологии рассматриваемых децентрализованных сетей могут строится на основе рассмотренных топологий непосредственно, либо на их множестве или комбинациях. Существуют еще иные топологии, используемые для построения специальных P2P сетей. Часть указанных топологий будет рассмотрена ниже.

2.2 Топологии массовых многопользовательских онлайн игр

Многопользовательские онлайн-игры (Massively multiplayer online games – MMOG) в последнее время стали популярным классом приложений с миллионами пользователей по всему миру, подключенных через Интернет для совместной игры [18]. Большинство этих игр предоставляют виртуальную среду, в которой игроки развиваются и взаимодействуют друг с другом. Когда игрок движется, перемещает объекты или выполняет какую-либо операцию, которая оказывает влияние на виртуальную среду, игроки вокруг него могут видеть его действия.

В MMOG каждый игрок запускает часть клиентского программного обеспечения на локальном устройстве (например, персональном компьютере или игровой станции), называемом впоследствии узлом. Программное обеспечение локального клиента отвечает за поддержание актуальной версии состояния виртуального мира, окружающего игрока, и за предоставление ему возможности выполнять над ним операции (то есть играть). Чтобы обеспечить приемлемый игровой мир, распределенное приложение должно отображать виртуальный мир, окружающий игрока, с минимальными задержками.

Игры, построенные по технологии клиент-сервер, при большом числе игроков не справляются с нагрузкой – сервера не успевают обрабатывать невероятно большие объемы запросов.

Для преодоления указанных ограничений появилось новое поколение децентрализованных сетевых виртуальных сред (Networked Virtual Environments – NVE), основанных на P2P оверлеях. В дополнение к масштабируемости и стоимости, эти новые решения обеспечивают лучшую доступность, поскольку можно продолжать играть, даже когда многие игроки присоединяются к игре и покидают ее. На самом деле, несколько игроков могут играть вместе, как только они подключены. Эти новые системы также предлагают больше свободы для игрока: становится возможным иметь игру без какой-либо центральной организации (компании), контролирующей ее.

Однако распространение таких приложений с использованием P2P парадигмы действительно сложно. Большинство существующих P2P оверлеев не удовлетворяют новым требованиям к приложениям, поскольку их задержки связи слишком высоки.

MMOG игры основаны на концепции виртуальной среды. Виртуальная среда - это n-мерное аппликативное пространство именований, в котором развиваются игроки. Обычно измерения отображаются в координатах виртуального мира (например, оси x, y и, возможно, оси z трехмерной карты), но также можно использовать другие понятия семантического расстояния (например, списки контактов игроков, скрытые ворота, которые позволяют игроку прыгать в другое место в виртуальном мире и т. д.).

Каждый узел игрока управляет объектом, представляющим игрока в игре. Эта сущность представляет собой особый объект виртуальной среды, обычно называемый аватаром. Каждому объекту и, следовательно, каждому аватару присваиваются координаты в виртуальном мире. Когда игрок играет, аватар игрока перемещается (и потенциально может перемещать объекты), и его координаты в виртуальной среде изменяются.

Таким образом, каждый узел управляет набором объектов, который определяет область знаний (Knowledge Area) узла, также называемую «зоной ответственности». Каждый раз, когда что-то меняется в его области знаний, узел должен обновлять его, либо уведомляя об изменениях других игроков, либо периодически проверяя узлы, с которыми аватар играет в своей области знаний (рис. 2.5).

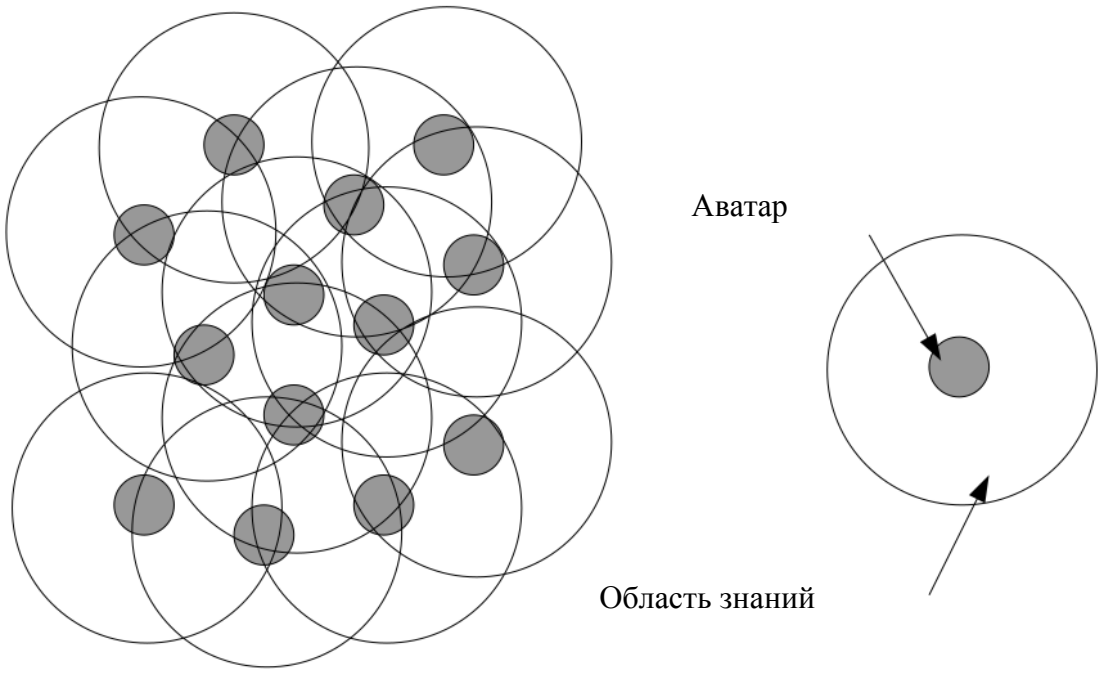


Рисунок 2.5 – Пересечение областей знаний

Клиентское программное обеспечение (работающее на узлах игроков) должно поддерживать актуальную игровую зону для игрока. Игровая зона - это зона вокруг аватара, которую необходимо отобразить игроку-человеку. Рисунок 2.6 иллюстрирует это понятие. Когда игрок играет, его аватар модифицирует локальную игровую зону. Изменения распространяются на узлы, которые управляют объектами в этой игровой зоне: это узлы, область знаний которых содержит части рассматриваемой игровой зоны.

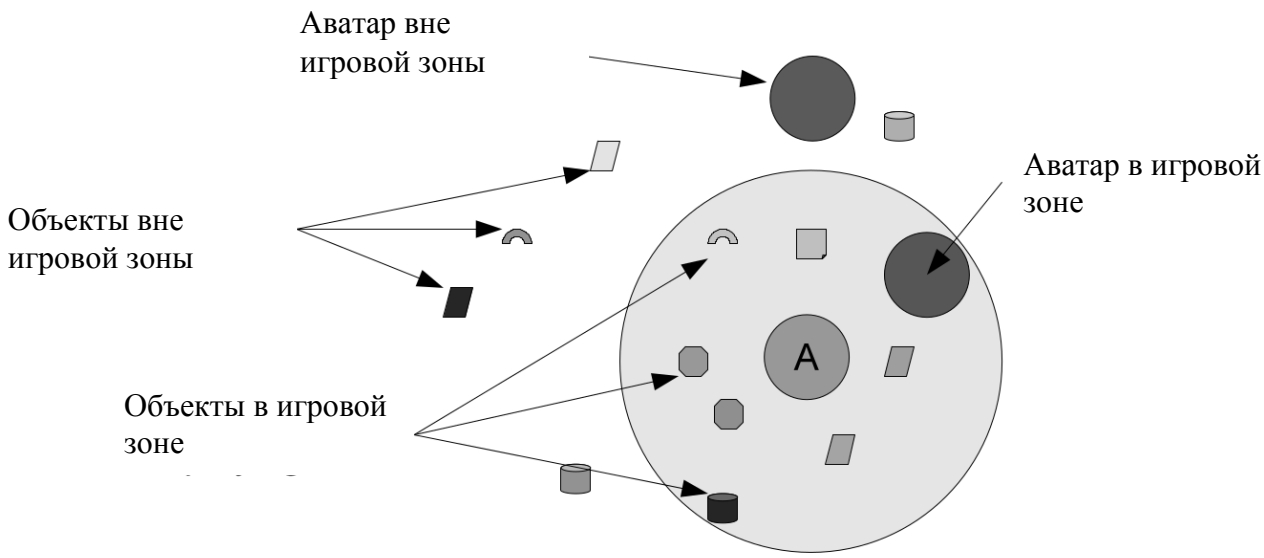


Рисунок 2.6 – Аватар, как игровая зона

Для управления областями знаний и игровыми областями разработаны ряд оверлеев, рассмотренных в [18]. Одним из игровых оверлеев является Solipsis.

Solipsis - это оверлей, предназначенный для поддержки распределенной виртуальной среды. Каждый узел оверлея Solipsis отвечает за один аватар. В Solipsis область знаний и игровая зона совпадают. Следовательно, объекты игровой зоны реплицируются на узлах, которые управляют аватарами в этой игровой зоне. В Solipsis уже коммуницировавшиеся узлы находятся в текущей игровой зоне: они являются узлами, которые управляют объектами в игровой зоне.

Solipsis поддерживает набор прямых соседей для каждого узла. Узлы обмениваются сообщениями, проходящими через оверлей: задержка увеличивается с расстоянием (измеряемым количеством прыжков) в оверлее. Solipsis старается поддерживать соседей для эффективного общения. Если два аватара A и B являются соседями в виртуальной среде, оверлей Solipsis адаптируется так, что B в конечном итоге окажется в окрестности A, и наоборот. Рисунок 2.7 показывает изменение узлов в оверлее при движении узла C.

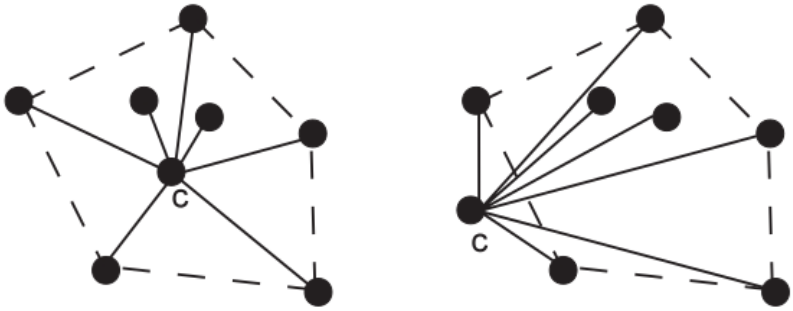


Рисунок 2.7 – Изменение контекста при движении узла

Таким образом, при реализации децентрализованных P2P сетей могут использоваться специализированные оверлеи, способные обрабатывать динамические аспекты поведения системы: присоединение и отсоединение узлов в сети; изменение набора соседних узлов при изменении координат выбранного узла.

Следуют отметить, что особые виды оверлеев используются при построении различных типов самоорганизующихся беспроводных сетей

2.3 Топологии самоорганизующихся беспроводных сетей

Распространенной моделью для самоорганизующихся беспроводных сетей (СПС), которая рассматривает широковещательную передачу, является модель в виде графа единичных кругов (Unit Disk Graph - UDG) [19].

Пусть V и E – множества вершин и ребер графа.

Определение. UDG граф это такой граф, у которого каждая вершина $u \in V$ имеет круг диаметром 1 и ребро $e \in E$ соединяет вершины u и v тогда и только тогда евклидово расстояние между вершинами $d(u, v) \leq 1$.

Таким образом, два узла u и v в СПС соединены, если расстояние между ними не более радиуса передачи, который считается одинаковым для всей сети. Данное условие выполняется в большом числе случаев, когда можно считать, что антенны узлов всенаправленные и радиус взаимодействия примерно одинаков по сети. Рисунок 2.8 приводит пример UDG топологии.

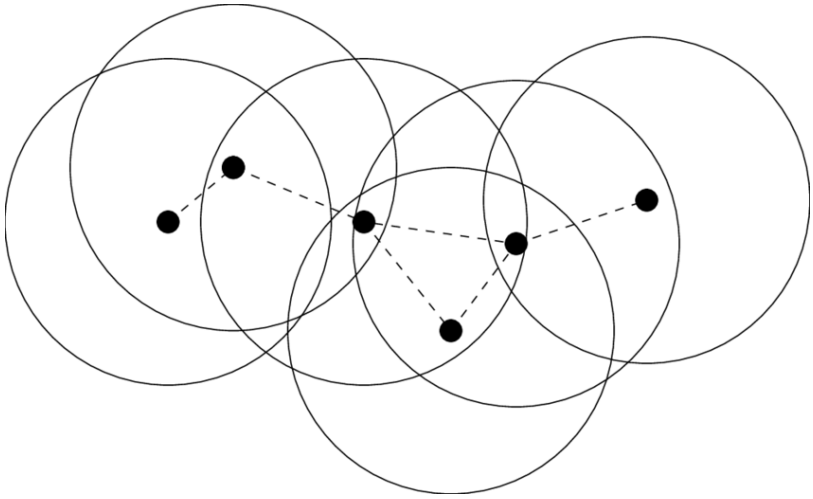


Рисунок 2.8 – Сеть, построенная по топологии UDG графа

UDG модель имеет два главных недостатка:

- в модели существует предположение о наличие узлов с одинаковыми характеристиками передачи. Данное условие может быть нарушено либо из-за разнородности самих узлов, либо из-за неодинаковости условий передачи (например, наличие препятствий);
- модель не учитывает веса узлов, которые бывают полезны для определения других параметров, таких как подвижность и остаточная энергия узлов.

Однако, несмотря на указанные недостатки, модель UDG графа является весьма распространенной при исследовании ad hoc беспроводных сетей.

Расширением UDG модели является граф квазиединичных кругов (Quasi Unit Disk Graph – QUDG). В этой модели каждый узел характеризуется двумя кругами внутренним радиуса r и внешним радиуса 1. Условие существования ребра между узлом u и соседним узлом v определяется следующим образом:

- ребро между узлами u и v существует, если $d(u, v) \leq r$;
- ребро между узлами u и v может существовать, если $r \leq d(u, v) \leq 1$;
- ребро между узлами u и v не существует, если $d\ u, v\ > 1$.

Рисунок 2.9 показывает сеть, построенную по топологии QUDG графа около узла *u* и его соседей *u, v, s, w*. В силу того, что расстояние между узлами *u* и *v* меньше *r*, то между ними связь существует и на рисунке она отображена сплошной линией. Между узлами *u* и *w* связь возможна и отображается пунктирной линией. Между узлами *u* и *s* связи нет.

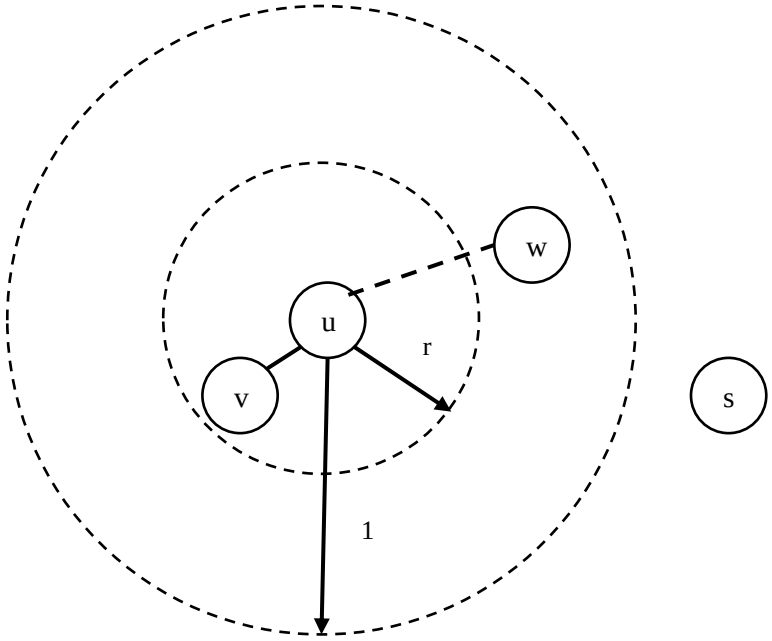


Рисунок 2.9 – Сеть, построенная по топологии QUDG графа

2.4 Топологии социальных сетей

Социальные сети являются актуальной разновидностью децентрализованных пиринговых сетей и являются объектом повышенного изучения.

В зависимости от того, как моделируется такие сети зависят и некоторые характеристики сети. Большие естественные сети рано или поздно образуют гигантский компонент (рисунок 2.10) [16].

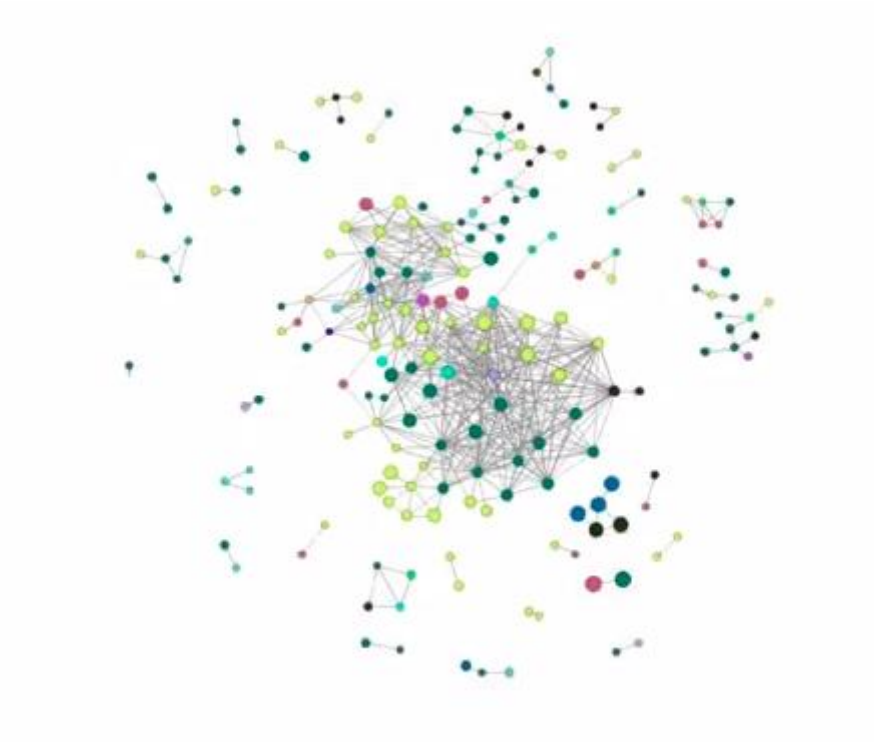


Рисунок 2.10 – Модель социальной сети

Для моделирования реального неравенства распределения ресурсов в сетях можно использовать топологию предпочтительного связывания (preferential attachment, cumulative advantage), которая позволяет моделировать следующие задачи [16]:

– почему богатые становятся богаче;

– почему у поп-звёзд так много поклонников и т.д.

В целом указанные задачи сводятся к тому, что чем больше у вершины есть ресурсов, тем легче ей получить дополнительные.

Модель пространственного предпочтительного присоединения (Spatial Preferential Attachment – SPA) была предложена в работе [17]

Эта модель сочетает предпочтительное присоединение и структуру метрического пространства, в котором лежат вершины [15]. У каждой вершины вводится понятие «сферы влияния». Параметрами модели являются вероятность создания ребра p и две константы A_1, A_2, удовлетворяющие следующим условиям:

$$0 < A_1 < \frac{1}{p},$$

$$A_2 > 0.$$

Указанные константы отвечают за объем сфер влияния вершин.

В модели SPA генерируется последовательность случайных графов $\{G_t\}$. Последовательность графов строится постепенно, начиная с пустого графа G_0 в момент времени $t = 0$. Далее, для каждого $t \geq 1$, граф G_t получается из графа G_{t-1} следующим образом. Сначала случайно и равномерно из полного множества вершин выбирается новая вершина v_t, которая добавляется к множеству вершин предыдущего шага V_{t-1}. Затем, независимо для каждой вершины из множества вершин предыдущего шага $u \in V_{t-1}$, направленное ребро v_t, u проводится с вероятностью p. Таким образом, вероятность того, что ребро v_t, u будет добавлено в момент времени *t*, равна $p|S\ u, t-1\ |$, где $S\ u, t-1$ – сфера влияния вершины u в момент времени *t-1*. Y На рисунке 2.11 представлен пример граф в SPA модели [15].

Выводы по второй главе

Был рассмотрен ряд моделей распределенных пиринговых систем, включая топологию предпочтительного связывания, нашедшую широкое применение при исследовании социальных сетей, когда некоторые узлы сети имеют значительно больше соседних узлов. Кроме этого, были рассмотрены особенности топологий массовых многопользовательских онлайн игр, самоорганизующихся беспроводных сетей.

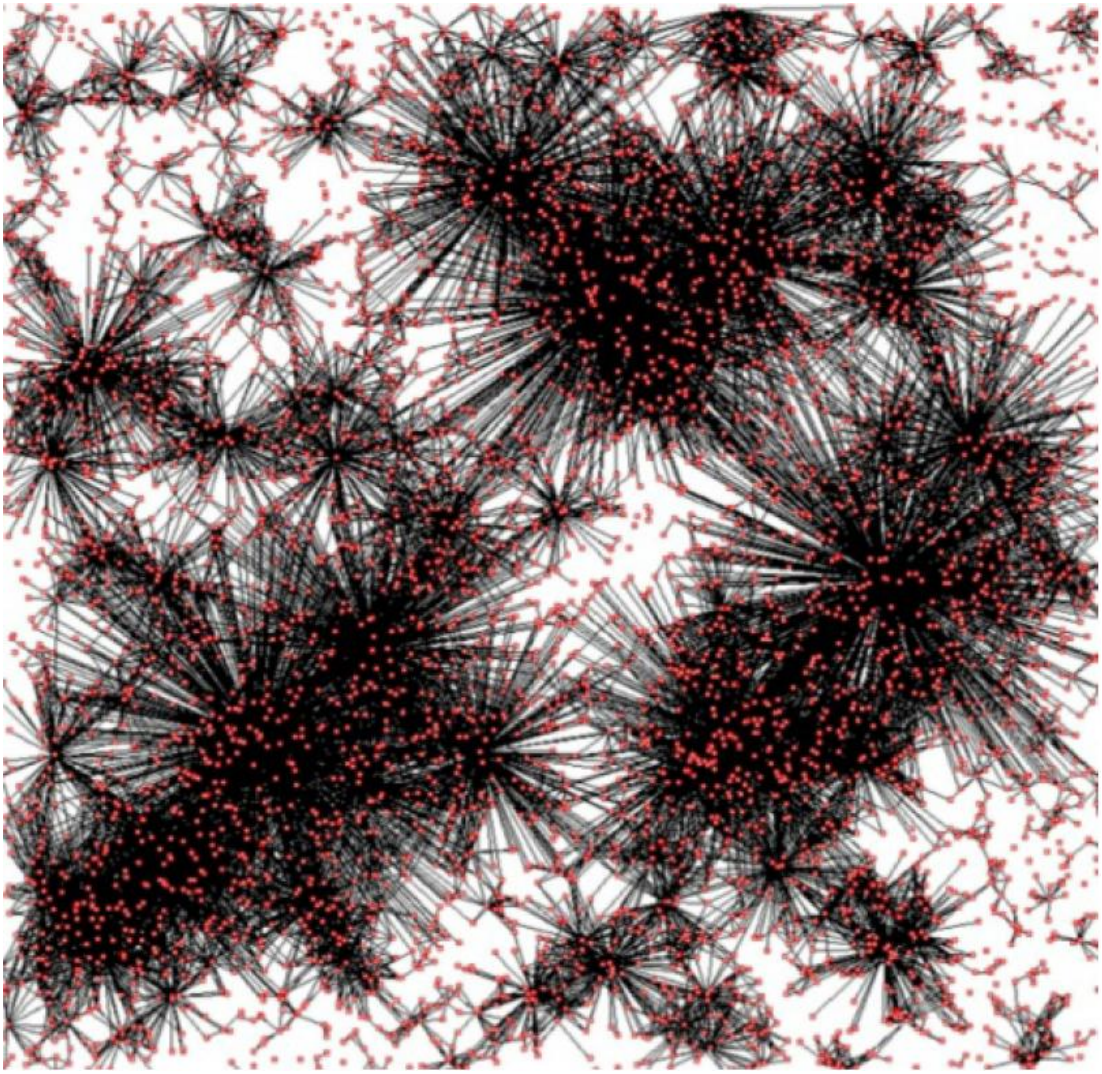


Рисунок 2.11 – Граф в SPA модели

3 КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ТОПОЛОГИЙ ДЕЦЕНТРАЛИЗОВАННЫХ ПИРИНГОВЫХ СИСТЕМ

3.1 Описание технологии моделирования

Моделирование и симуляция часто используются для оценки производительности пиринговых систем. Модель - это упрощенное представление системы, которая помогает понять и исследовать реальную систему.

Ключевыми особенностями одноранговых пиринговых систем являются масштабируемость и динамизм. Натурные эксперименты с P2P протоколами являются дорогими и характеризуются повышенной сложностью, поэтому компьютерное моделирование часто имеет решающее значение в исследованиях распределенных пиринговых систем [22].

Одним из известных P2P симуляторов является PeerSim – хорошо масштабируемая среда моделирования, которая поддерживает различные динамические сценарии, такие как отток узлов и т.п. Достоинством симулятора PeerSim является его реализация на языке Java, что позволяет разрабатывать сложные P2P модели и использовать преимущества многопоточного выполнения.

PeerSim имеет модульную структуру и достаточно простой механизм конфигурации. P2P сеть моделируется как набор узлов. Каждый узел имеет список протоколов, а симуляция – инициализаторы и элементы управления.

Инициализация осуществляется до симуляции, а элементы управления выполняются во время симуляции. И инициаторы, и элементы управления могут модифицировать или контролировать любой узел. Например, они могут добавлять новые узлы или уничтожать существующие; они могут действовать на уровне протоколов, предоставляя им внешний вход или изменяя соответствующие параметры. Элементы управления также могут использоваться для пассивного мониторинга процесса симуляции, например, они могут сообщать о скорости уменьшения дисперсии во время выполнения протокола агрегации на основе диффузии [20], или они могут сообщать о

теоретико-графических свойствах топологий наложения, таких как диаметр, кластеризация и т. д.

Механизм моделирования может быть основан на цикле (cycle-based) (протоколы выполняются в определенном порядке) или на основе событий (event-based). Симулятор читает файл конфигурации и загружает указанные классы во время выполнения. На основе файла конфигурации загружаются либо механизмы циклического, либо событийного моделирования. В первом случае для обеспечения масштабируемости используются некоторые упрощающие допущения, такие как игнорирование деталей транспортного уровня в стеке протокола связи. Второй механизм является менее эффективным, но более реалистичным. Кроме того, событийный механизм поддерживает моделирование транспортного уровня.

Каждое моделирование определяется файлом конфигурации в виде текстового файла, в котором на каждой строчке приведены пара «свойство – значение» (рис. 3.1). Свойства определяют общие настройки симуляции и реализации компонентов (определяют числовые или строковые параметры для компонентов). В приведенном примере циклическая симуляция сети в 1000 узлов выполняется в течении 100 циклов. Каждый узел запускает протокол под названием link, который реализуется классом IdleProtocol. Параметр инициализации init.rndlink, реализованном классом WireKOut, инициализирует оверлейные ссылки, управляемые сообщениями, для создания случайной топологии оверлея с постоянной степенью init.rndlink.k. IdleProtocol – это встроенный протокол PeerSim, который хранит информацию о соседях узла [21].

Так как PeerSim реализован на Java, то это позволяет выполнять симуляции посредством чтения файлов конфигурации и динамической загрузки классов с использованием механизма рефлексии Java. PeerSim также предоставляет простые компоненты, реализующие основные функции симулятора, которые могут замещаться пользовательскими реализациями.

```
# network size
SIZE 1000

# parameters of periodic execution
CYCLES 100
CYCLE SIZE*10000

# parameters of message transfer
# delay values here are relative to cycle length, in percentage,
# eg 50 means half the cycle length, 200 twice the cycle length, etc.
MINDELAY 0
MAXDELAY 0
# drop is a probability, 0<=DROP<=1
DROP 0

random.seed 1234567890
network.size SIZE
simulation.endtime CYCLE*CYCLES
simulation.logtime CYCLE

################### protocols ===========================

protocol.link peersim.core.IdleProtocol

protocol.avg example.edaggregation.AverageED
protocol.avg.linkable link
protocol.avg.step CYCLE
protocol.avg.transport tr

protocol.urt UniformRandomTransport
protocol.urt.mindelay (CYCLE*MINDELAY)/100
protocol.urt.maxdelay (CYCLE*MAXDELAY)/100

protocol.tr UnreliableTransport
protocol.tr.transport urt
protocol.tr.drop DROP

################### initialization ======================

init.rndlink WireKOut
init.rndlink.k 20
init.rndlink.protocol link
```

Рисунок 3.1 – Фрагмент файла инициализации PeerSim

В результате инициализации создается структура компонентов симулятора. На рис. 3.2 показан возможный вариант компонентов PeerSim.

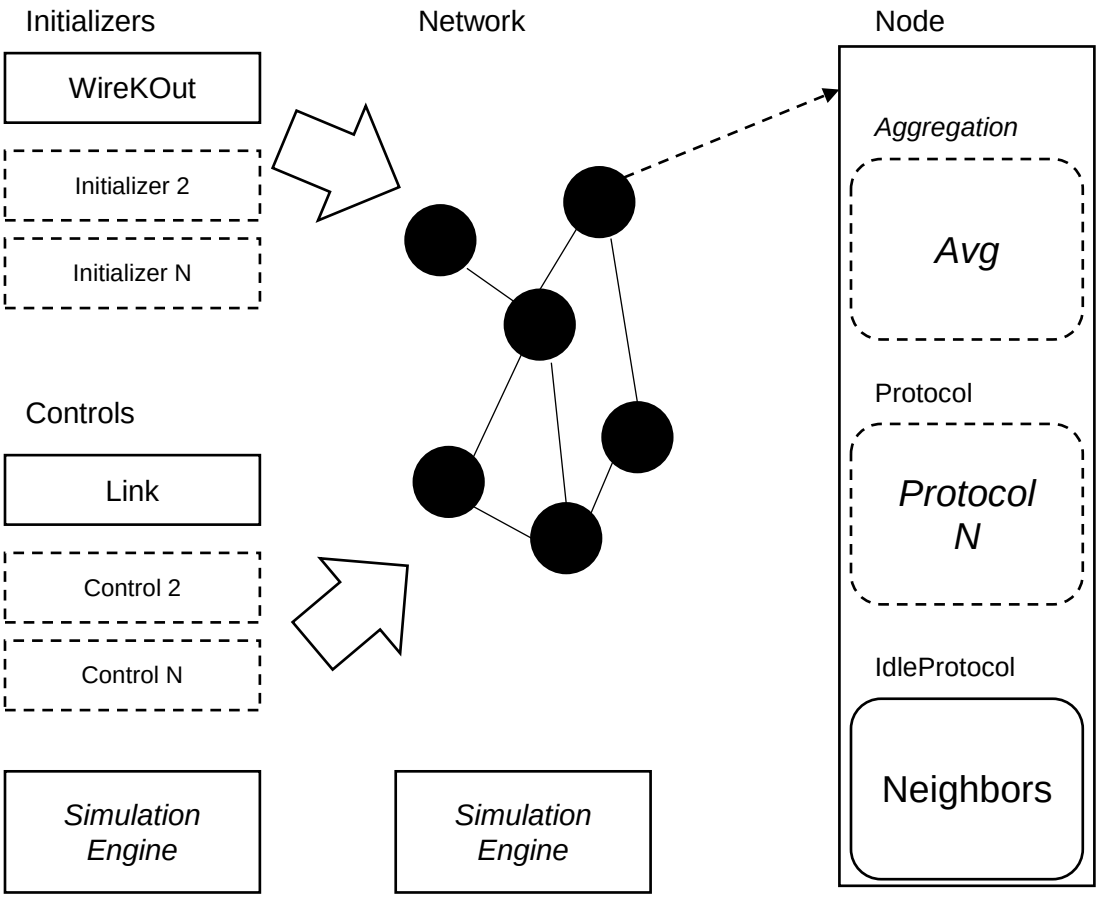


Рисунок 3.2 – Условный набор компонентов PeerSim после инициализации

Для инициализации модели могут использоваться различные инициаторы (Initializers). Например, выше приводился инициализатор, реализуемый классом WireKOut. Элементы управления (Controls), настраивают узлы сети (Networks), поведение которой разыгрывается механизмом симуляции (Simulation Engine). Каждый узел (Node) может включать в себя несколько протоколов, реализующих различный функционал, например вести таблицу соседей узла, рассчитывать агрегирующую функцию над данными узлов и т.д.

Очевидным преимуществом использования Peersim является необходимость программирования только двух типа компонент: протоколы, реализующие интерфейс Protocol, и элементы управления (супервизоры, controls) – классы, реализующие интерфейс Control [13].

Все узлы в сети имеют один и тот же набор протоколов. Протоколы узла имеют доступ только к протоколам узлов-соседей и могут объединяться в стеки. Супервизорам доступны все узлы сети. По умолчанию связи в PeerSim являются однонаправленными. После инициализации механизм симуляции вызывает все экземпляры Protocol и Control один раз в каждом цикле. В Peersim все объекты Protocol и Control присоединяются к объекту Scheduler, который определяет, когда они будут выполнены. По умолчанию все объекты выполняются в каждом цикле.

3.2 Моделирование топологии предпочтительного связывания

Как уже отмечалось выше для моделирования предпочтительного связывания необходимо переопределить несколько классов.

В поставке PeerSim уже имеется соответствующий класс, реализующий протокол сети (рис. 3.3) [21]. Данный класс просто сохраняет координаты узла.

```
public class InetCoordinates implements Protocol {
    private double x, y;

    public InetCoordinates(String prefix) {
        /* Un-initialized coordinates defaults to -1. */
        x = y = -1;
    }

    public Object clone() {
        InetCoordinates inp = null;
        try {
            inp = (InetCoordinates) super.clone();
        } catch (CloneNotSupportedException e) {
        } // never happens
        return inp;
    }
    public double getX() {
        return x;
    }
    public void setX(double x) {
        this.x = x;
    }
    public double getY() {
        return y;
    }
    public void setY(double y) {
        this.y = y;
    }
}
```

Рисунок 3.3 – Фрагмент кода протокола InetCoordinates

Класс инициализации InetInitializer, реализующий интерфейс Control, имеет единственный метод execute() и генерирует равномерно распределенные координаты, за исключением корня графа с номером 0 и координатами (0.5,0.5) (рис. 3.4).

```
public class InetInitializer implements Control {
    private static final String PAR_PROT = "protocol";
    private final int pid;
    public InetInitializer(String prefix) {
        pid = Configuration.getPid(prefix + "." + PAR_PROT);
    }

    public boolean execute() {
        // Set the root: the index 0 node by default.
        Node n = Network.get(0);
        InetCoordinates prot = (InetCoordinates) n
                .getProtocol(pid);
        prot.setX(0.5);
        prot.setY(0.5);

        // Set coordinates x,y
        for (int i = 1; i < Network.size(); i++) {
            n = Network.get(i);
            prot = (InetCoordinates) n.getProtocol(pid);
            prot.setX(CommonState.r.nextDouble());
            prot.setY(CommonState.r.nextDouble());
        }
        return false;
    }
}
```

Рисунок 3.4 – Фрагмент кода инициализатора

Класс WireInetTopology, расширяющий класс WireGraph из стандартной поставки пакета PerSim, формирует топологию предпочтительного связывания. В качестве класса, записывающего граф сети в файл будем использовать класс InetObserver, расширяющий класс GraphObserver.

Последним шагом подготовки симуляции является разработка файла конфигурации, содержимое которого приведено на рисунке 3.5.

```
simulation.cycles 1
network.size 10000

protocol.link IdleProtocol
protocol.coord example.hot.InetCoordinates

init.0 example.hot.InetInitializer
init.0.protocol coord
init.1 example.hot.WireInetTopology
init.1.protocol link #the linkable to be wired
init.1.coord_protocol coord
init.1.alpha 4

control.io example.hot.InetObserver
control.io.protocol link
control.io.coord_protocol coord
control.io.file_base graph
control.degree DegreeStats
control.degree.protocol link
control.degree.undir
```

Рисунок 3.5 – Содержимое файла конфигурации симуляции предпочтительного связывания

В результате выполнения выводится лог программы, приведенной на рисунке 3.6. Как видно, на диске сформирован файл *.dat, в который записана топология сети.

Для визуализации данных было использовано приложение GNUPLOT [14]. После запуска терминала GNUPLOT была выполнена команда

```
gnuplot> plot 'graph00000000.dat' with linespoint lt -1,
```

отображающая полученный результат на экран монитора компьютера. Для записи графика в файл *.png были использованы следующие команды

```
gnuplot> set term png
Terminal type set to 'png'
Options are 'nocrop font
"/usr/share/fonts/truetype/liberation/LiberationSans-
Regular.ttf,12" fontscale 1.0 size 640,480 '
gnuplot> set output "40.png"
gnuplot> replot
gnuplot> set term x11
```

```
java -cp "peersim-1.0.5.jar:jep-2.3.0.jar:djep-1.0.0.jar"
peersim.Simulator config2.txt
Simulator: loading configuration
ConfigProperties: File config2.txt loaded.
Simulator: starting experiment 0 invoking peersim.cdsim.CDSimulator
Random seed: 1560620789137

CDSimulator: resetting
Network: no node defined, using GeneralNode
CDSimulator: running initializers
- Running initializer init.0: class example.hot.InetInitializer
- Running initializer init.1: class example.hot.WireInetTopology
CDSimulator: loaded controls [control.degree, control.io]
CDSimulator: starting simulation
control.degree: 1.0 2157.0 10000 1.9998 707.5467546354635 9827 1
control.io: Writing to file graph00000000.dat
CDSimulator: cycle 0 done
```

Рисунок 3.6 – Информация о выполнении симуляции предпочтительного связывания

Результаты моделирования пиринговой сети по топологии предпочтительное связывание для различных значений параметра α приведен на рисунках 3.7-3.10.

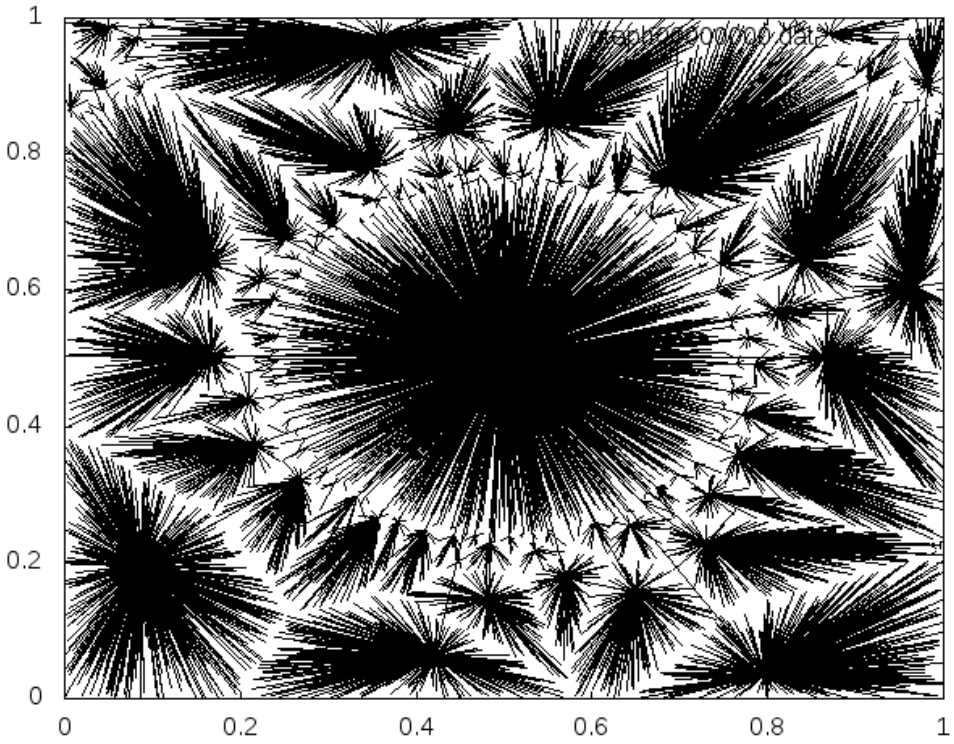


Рисунок 3.7 – Топология предпочтительного связывания при α=4

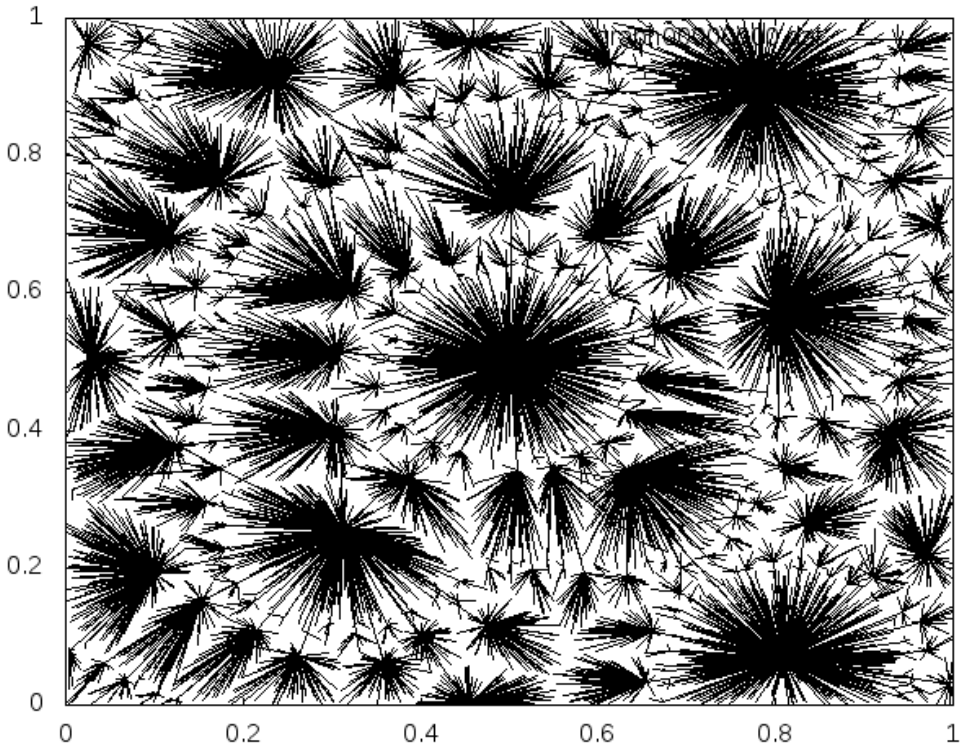


Рисунок 3.8 – Топология предпочтительного связывания при α=8

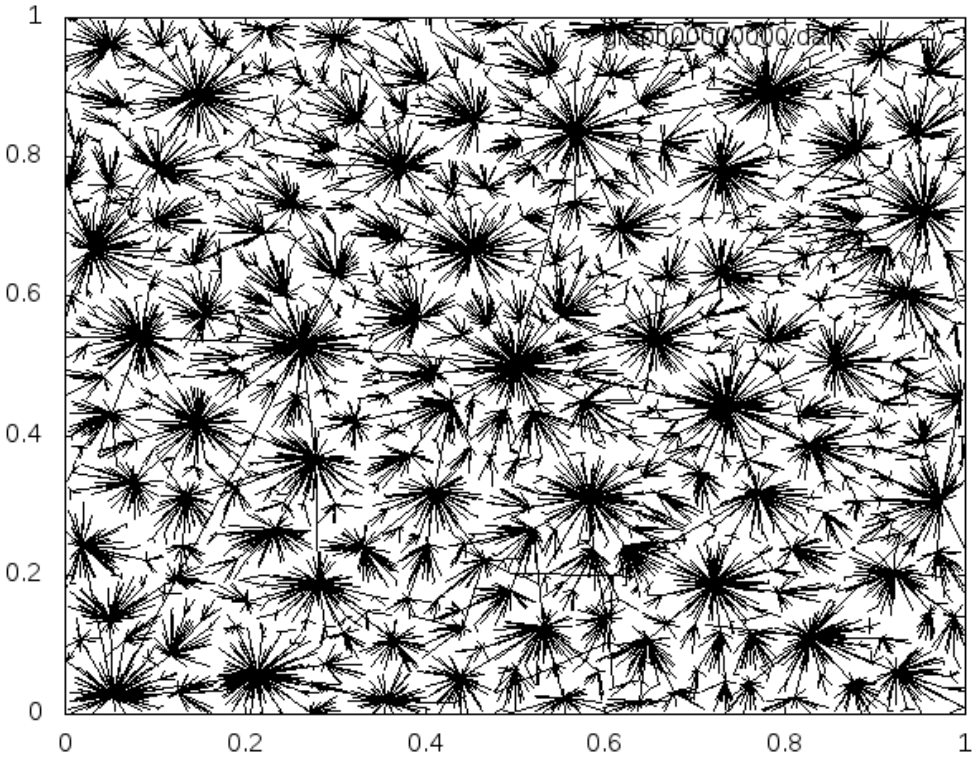


Рисунок 3.9 – Топология предпочтительного связывания при α=20

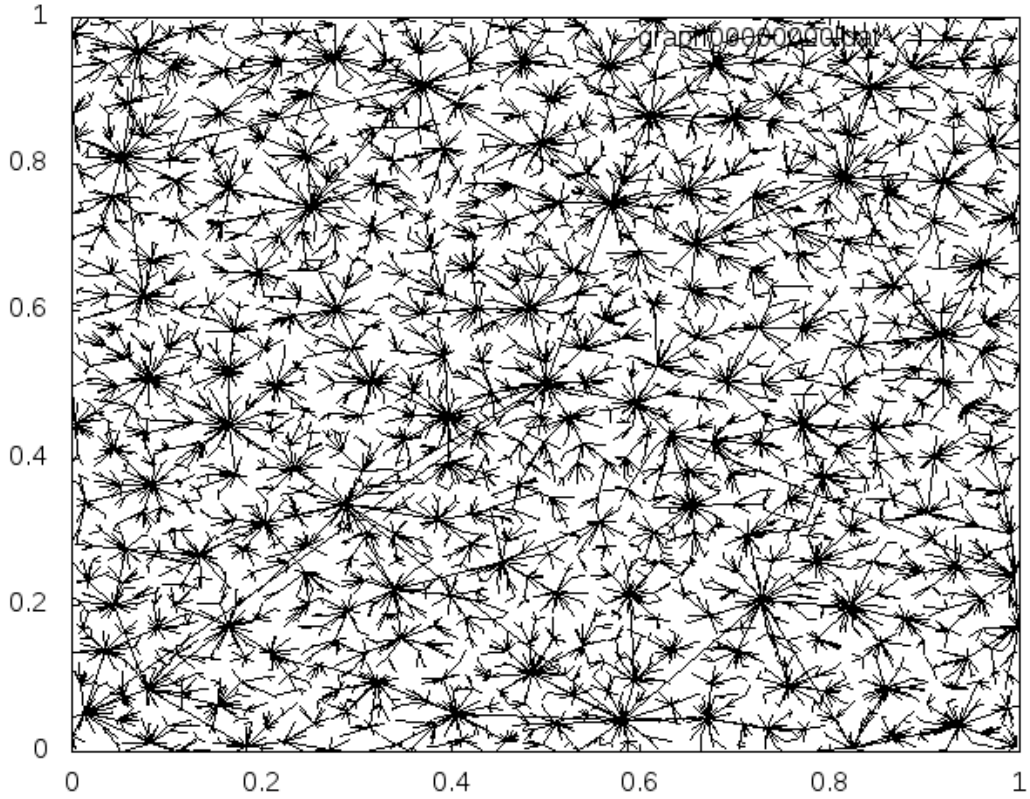


Рисунок 3.10 – Топология предпочтительного связывания при α=50

3.3 Моделирование топологии частично связанного графа

Основываясь на изложенной методике в ходе выполнения работы была произведена разработка класса, отвечающего за вывод топологии частично связанного графа (рис. 3.11). Код класса достаточно прост – метод execute() просто расставляет узлы пиринговой системы в узлы решетки размером 1x1.

На рисунке 3.12 приведен файл конфигурации эксперимента. После обработки данных утилитой GNUPLOT получены топологии пиринговой сети разной размерности N и коэффициента k, характеризующего количество связей узла с другими узлами сети (рис. 3.13 – 3.17).

Код класса RandomNetInitializer приведен в приложении А.

```
public class RandomNetInitializer implements Control{
    private static final String PAR_PROT = "protocol";
    private final int pid;
    public RandomNetInitializer(String prefix) {
        pid = Configuration.getPid(prefix + "." + PAR_PROT);
    }
    public boolean execute() {
      Node node;
        InetCoordinates prot;

        // Set coordinates x,y
        int networkSize = Network.size();
        //Calculate number nodes in row/column of grid
        int n = Double.valueOf(
                      Math.ceil(Math.sqrt(networkSize))).intValue();
        double d = 1. / (n-1);
        for (int i = 0; i < networkSize; i++) {
            int col = i / n;
            int row = i % n;
            node = Network.get(i);
            prot = (InetCoordinates) node.getProtocol(pid);
            prot.setX(col * d);
            prot.setY(row * d);
        }
        return true;
    }
```

Рисунок 3.11 – Фрагмент кода класса RandomNetInitializer

```
simulation.cycles 1
control.shf Shuffle #shuffles nodes order
network.size 144

protocol.link IdleProtocol #Store links to neighbors
protocol.coord example.hot.InetCoordinates #Store node coordinates

init.rnd WireKOut
init.rnd.protocol link
init.rnd.k 6

init.0 tltsu.RandomNetInitializer
init.0.protocol coord

control.io example.hot.InetObserver
control.io.protocol link
control.io.coord_protocol coord
control.io.file_base graph
control.degree DegreeStats
control.degree.protocol link
control.degree.undir
```

Рисунок 3.12 – Файл конфигурации симуляции топологии частично связанного графа (N=144, k=6)

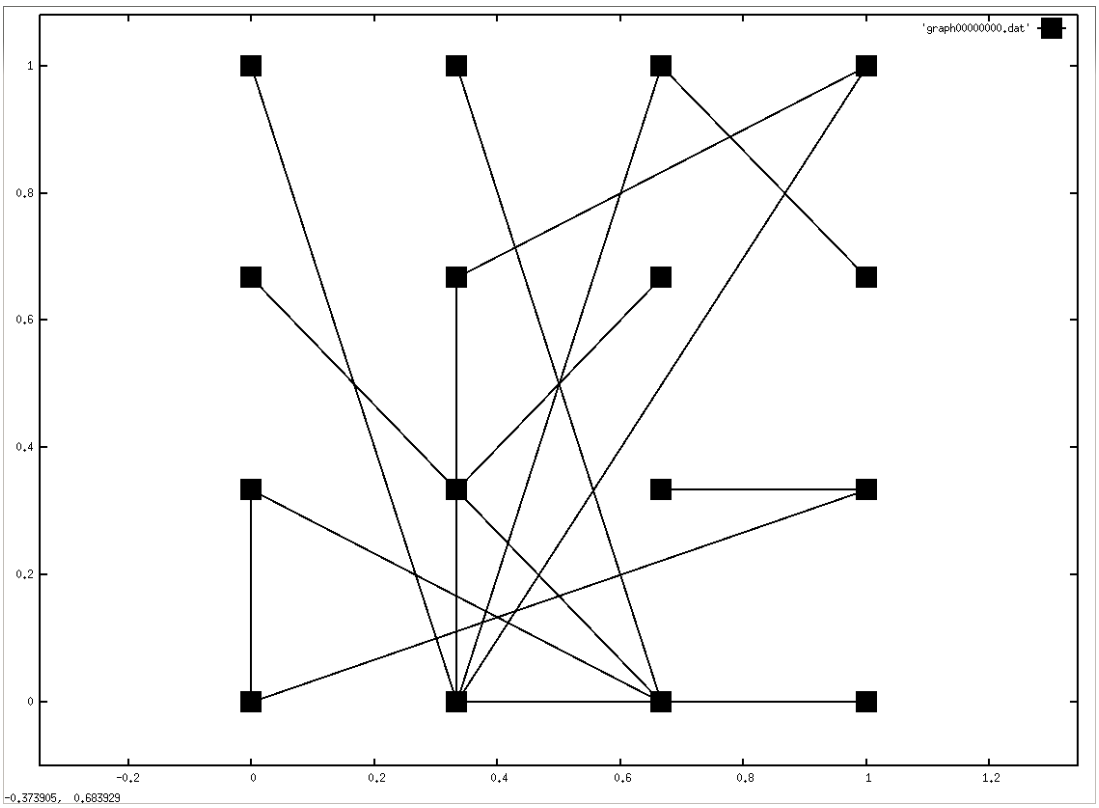


Рисунок 3.13 – Топология частично связанного графа (N=16, k=1)

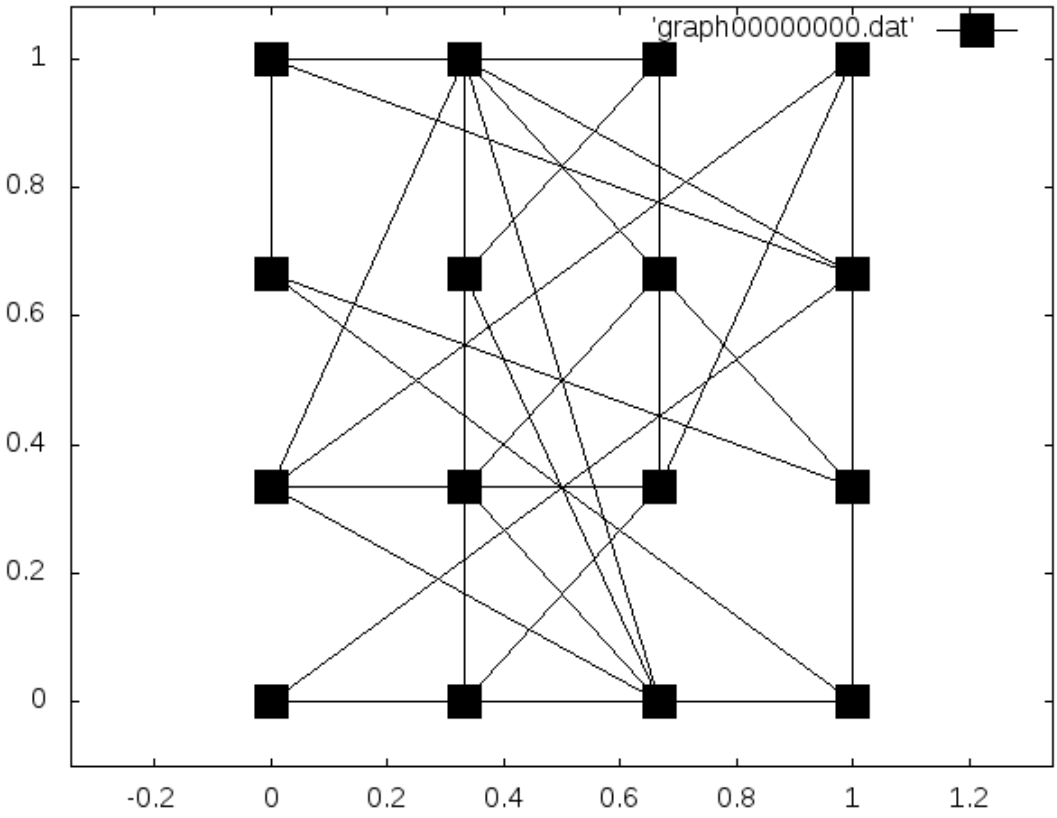


Рисунок 3.14 – Топология частично связанного графа (N=16, k=2)

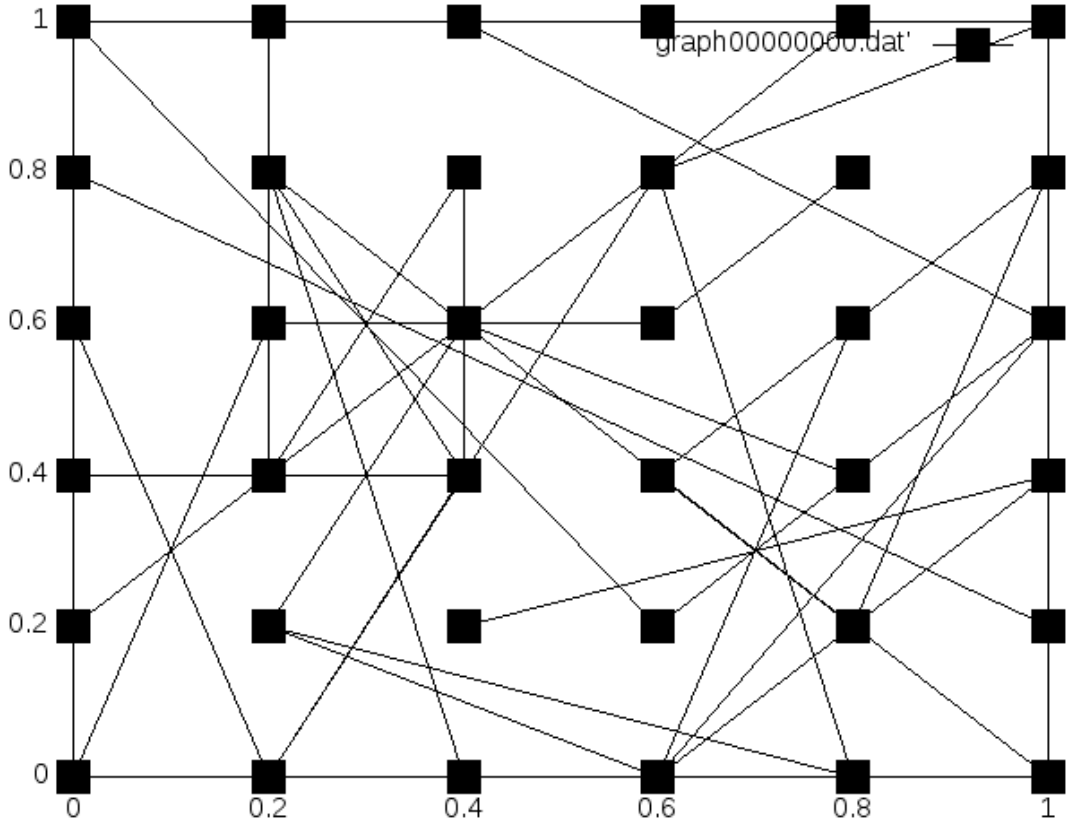


Рисунок 3.15 – Топология частично связанного графа (N=36, k=1)

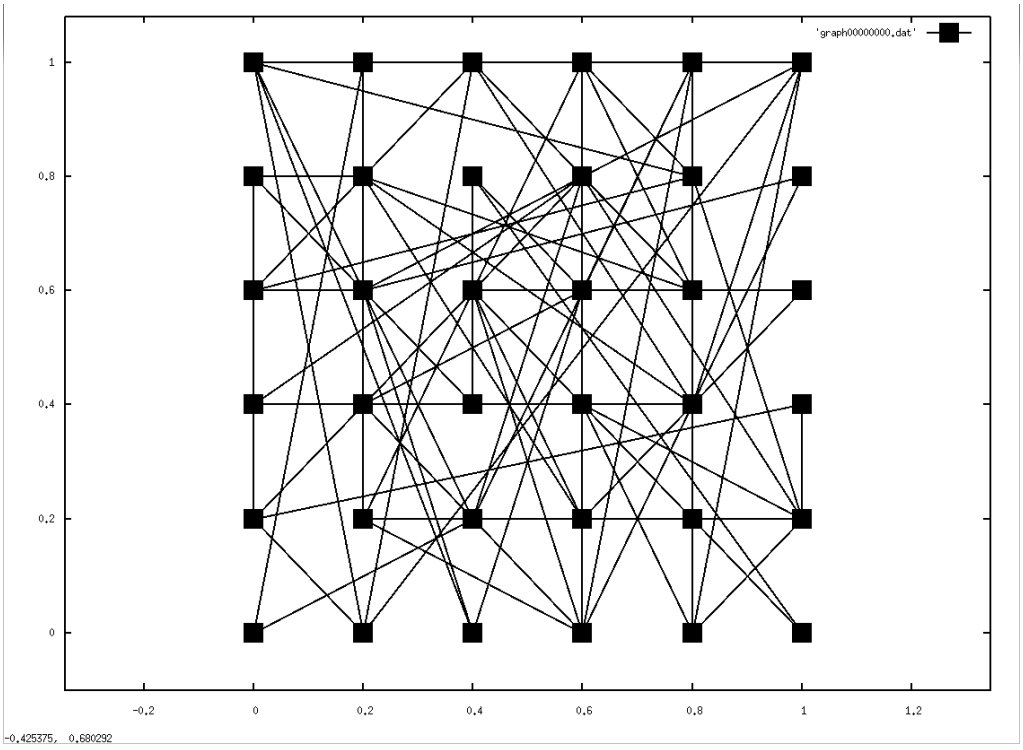


Рисунок 3.16 – Топология частично связанного графа (N=36, k=2)

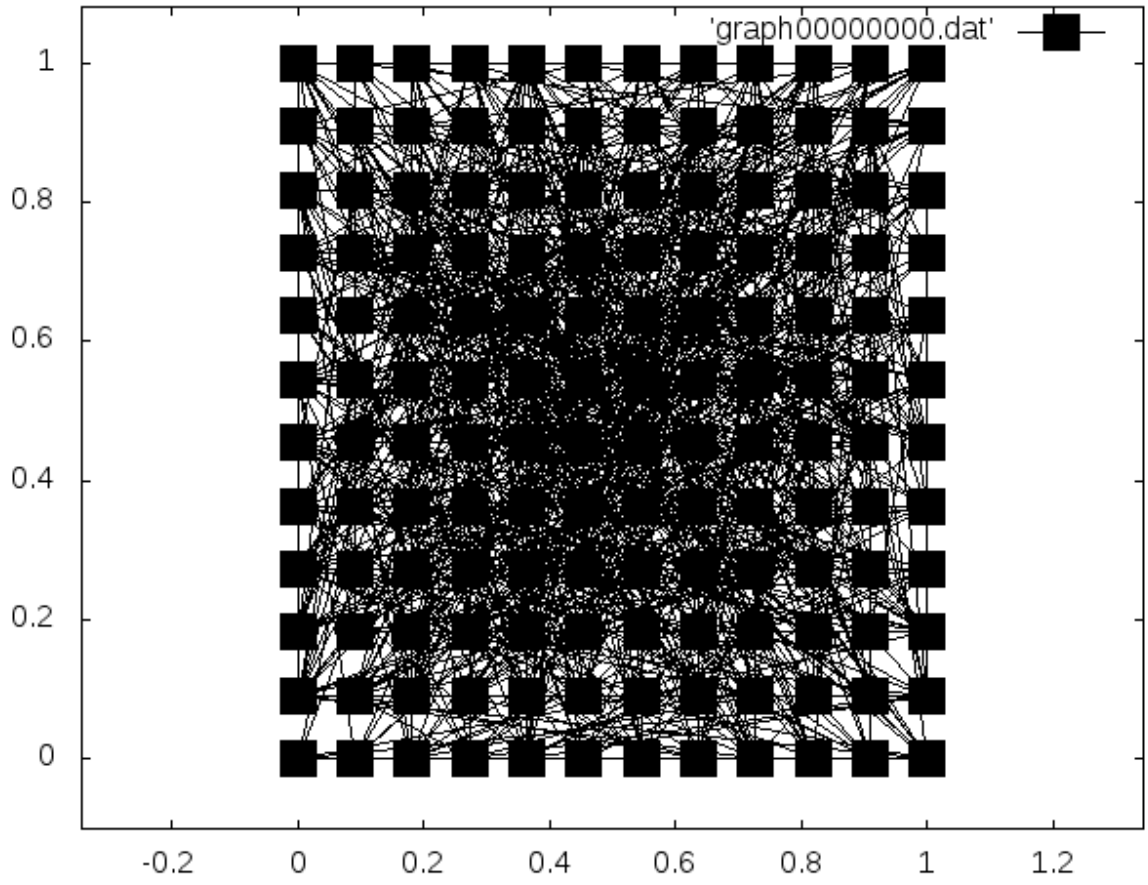


Рисунок 3.17 – Топология частично связанного графа (N=144, k=6)

Выводы по третьей главе

В результате третьей главы с использованием P2P симулятора PeerSim были выполнены все необходимы этапы процесса моделирования ряда топологий распределенных пиринговых систем.

Для визуализации полученных результатов было использовано средство GNUPLOT, позволяющая экспортировать графики результатов в различные графические форматы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы была доказана актуальность исследования свойств пиринговых систем, рассмотрены виды и свойства пиринговых систем.

Были рассмотрены архитектуры построения распределённых пиринговых систем и отмечено их большое разнообразие. Проведен анализ оверлеев P2P систем, определяющих структуру логических связей узлов в сети. В качестве примера были рассмотрены особенности построения неструктурированных беспроводных, мобильных неструктурированных и беспроводных сенсорных сетей.

Во второй главе рассмотрены часто используемые модели P2P систем, особое внимание было уделено применению использованию модели частично связанного графа и предпочтительного связывания, используемого для анализа социальных сетей.

В качестве средства моделирования был выбран P2P симулятор PeerSim, позволяющий проводить циклически и событийно ориентированное моделирование сетей с числом элементов более 10^6. Была апробирована технологии моделирования и визуализации топологии предпочтительного связывания с использованием компонентов стандартной поставки симулятора. Для визуализации топологии частично связанного графа был разработан программный компонент на языке Java и разработан файл конфигурации эксперимента.

Полученные результаты могут быть использованы для дальнейших исследований в области теории распределенных пиринговый сетей, технологий интернета вещей, IP TV и т.д.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. Паспорт национальной программы «Цифровая экономика Российской Федерации» [Электронный ресурс] – Режим доступа: http://static.government.ru/media/files/urKHm0gTPPnzJlaKw3M5cNLo6gczMkPF.pdf (дата обращения 10.06.2019).

2. Информационные материалы о национальной программе «Цифровая экономика Российской Федерации» [Электронный ресурс] – Режим доступа: http://static.government.ru/media/files/3b1AsVA1v3VziZip5VzAY8R-TcLEbdCct.pdf (дата обращения 10.06.2019).

Научная и методическая литература

3. Васильев И. Ю., Гайдамака Ю. В. Имитационная модель буферизации видеопотока в одноранговой сети с учетом геолокации и активности пользователей //Труды Второй молодежной научной конференции «Задачи современной информатики». – С. 58.

4. Виткова Л.А. Исследование распределенной компьютерной системы адаптивного действия // Наукоемкие технологии в космических исследованиях Земли. 2015. Т.7. №5. С. 44-48.

5. Гриценко А. В. Обзор принципа работы пиринговых сетей и изучение возможности их применения для коммутации в распределенных вычислениях //Science Time. – 2017. – №. 2. – С. 114-120.

6. Князьков В.С. Введение в теорию графов [Электронный ресурс] / В.С. Князьков, Т.В. Волченская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 76 с. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73674.html

7. Комаров И.И. Проектирование архитектуры инструментального средства моделирования поведения сети типа P2P / И.И. Комаров, А.А. Шлыков, М.В. Назыров и др. //Альманах научных работ молодых ученых XLV

научной и учебно-методической конференции Университета ИТМО. – 2016. – Том 5. – С. 274-276.

8. Кручинин С. В. Типы децентрализованных сетей и вариант построения децентрализованной сети полного стека протоколов //Известия Волгоградского государственного технического университета. – 2016. – №. 11 (190).

9. Математика. Дискретная математика [Электронный ресурс] : учебник / В.Ф. Золотухин [и др.]. — Электрон. текстовые данные. — Ростов-на-Дону: Институт водного транспорта имени Г.Я. Седова – филиал «Государственный морской университет имени адмирала Ф.Ф. Ушакова», 2016. — 129 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/57348.html

10. Ощепков М. Ю., Поповская Е. О. Математическое моделирование пиринговых сетей // М.Ю. Ощепков, Е.О. Поповская. // Наука и Мир. – 2014. – Т. 1. – №. 3. – С. 195-197.

11. Тель, Ж. Введение в распределенные алгоритмы: монография / Жерар Тель. – М.: Изд-во МЦНМО, 2009.–616 с.

Электронные ресурсы

12. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper [Электронный ресурс] – Режим доступа: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html (дата обращения 04.06.2019 г.).

13. Davide Malvestiti. Peersim - How to use it [Электронный ресурс] – Режим доступа: http://peersim.sourceforge.net/quickhowto/Peersim_eng.pdf (дата обращения 04.06.2019 г.).

14. Gnuplot homepage [Электронный ресурс] – Режим доступа: http://www.gnuplot.info/ (дата обращения 04.06.2019 г.).

15. Исхаков Л. Н., Prałat P., Kamiński B. , Прохоренкова Л.А. , Миронов М.С. Кластерный коэффициент в модели пространственного предпочтительного присоединения / Л.Н. Исхаков, P. Prałat, B. Kamiński, Л.А. Прохоренкова, М.С. Миронов.// Доклады Академии наук 2018. Т. 481. Номер 1 С. 10-13 [Электронный ресурс]. URL: http://ras.jes.su/dan/s207987840000242-5-1 (дата обращения: 10.06.2019).

16. Курапов А. Анализ сетей / А. Курапов [Электронный ресурс] – Режим доступа: https://kurapov.ee/rus/study/graphs/ (дата обращения 04.06.2019 г.).

Литература на иностранном языке

17. Aiello W. et al. A spatial web graph model with local influence regions //Internet Mathematics. – 2008. – Т. 5. – №. 1-2. – С. 175-196.

18. Distributed systems : design and algorithms / edited by Serge Haddad ... [et al.] // John Wiley & Sons, Inc, 2011. – 324 p.

19. Erciyes K. Distributed graph algorithms for computer networks. – Springer Science & Business Media, 2013.

20. Jelasity M., Montresor A., Babaoglu O. Gossip-based aggregation in large dynamic networks. / M. Jelasity, A. Montresor, and O. Babaoglu // ACM Transactions on Computer Systems, 23(3):219–252, August 2005.

21. Jesi G. P. PeerSim HOWTO: Build a new protocol for the PeerSim 1.0 simulator / G.P. Jesi // Peersim. surcefge. net. – 2005. http://peersim.sourceforge.net/tutorial1/tutorial1.pdf

22. Montresor A., Jelasity M. PeerSim: A scalable P2P simulator / A. Montresor, M. Jelasity //2009 IEEE Ninth International Conference on Peer-to-Peer Computing. – IEEE, 2009. – С. 99-100.

23. Varela C. A., Agha G. Programming Distributed Computing Systems: A Foundational Approach. – MIT Press, 2013.

ПРИЛОЖЕНИЕ А

КОД КЛАССА RANDOMNETINITIALIZER

```
public class RandomNetInitializer implements Control{
    // ------------------------------------------------------------------
--------
    // Parameters
    // ------------------------------------------------------------------
--------
    /**
     * The protocol to operate on.
     *
     * @config
     */
    private static final String PAR_PROT = "protocol";

    // ------------------------------------------------------------------
--------
    // Fields
    // ------------------------------------------------------------------
--------

    /** Protocol identifier, obtained from config property {@link
#PAR_PROT}. */
    private final int pid;

    // ------------------------------------------------------------------
--------
    // Constructor
    // ------------------------------------------------------------------
--------

    /**
     * Standard constructor that reads the configuration parameters.
Invoked by
     * the simulation engine.
     *
     * @param prefix
     *            the configuration prefix for this class.
     */
    public RandomNetInitializer(String prefix) {
        pid = Configuration.getPid(prefix + "." + PAR_PROT);
    }

    // ------------------------------------------------------------------
--------
    // Methods
```

```
    // ----------------------------------------------------------------
--------
    /**
     * Initialize the node coordinates.
     */
    public boolean execute() {
     Node node;
        InetCoordinates prot;

        // Set coordinates x,y
        int networkSize = Network.size();
        //Calculate number nodes in row/column of grid
        int n =
Double.valueOf(Math.ceil(Math.sqrt(networkSize))).intValue();
        //Calculate distance between two neighbors
        double d = 1. / (n-1);
        for (int i = 0; i < networkSize; i++) {
           int col = i / n;
           int row = i % n;
            node = Network.get(i);
            prot = (InetCoordinates) node.getProtocol(pid);
            prot.setX(col * d);
            prot.setY(row * d);
            System.out.printf("node %02d x:%5.2f y:%5.2f\n", i,
prot.getX(), prot.getY());
        }
        return false;
    }
```