Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Московский государственный технический университет им. Н. Э. Баумана

23 0102

Согласовано

к.т.н. доцент

_____ *Постников В.М.*

“____” _____ _____

Утверждаю

к.т.н. доцент

_____ *Постников В.М.*

“____” _____ _____

Курсовая работа по дисциплине «Эксплуатация АСОИиУ»

Студент группы ИУ5-92

_____ *Гуща А. В*

“____” _____ _____

2014

1 Реферат

Данный документ представляет собой расчетно-пояснительную записку к курсовой работе по курсу «Эксплуатация АСОИиУ».

Целью курсовой работы является разработка проектного решения на распределенную систему обработки данных (РСОД), объединяющую центральный офис и два удаленных филиала фирмы. В процессе выполнения курсовой работы предстоит решить следующие задачи:

- Выбор сетевого, телекоммуникационного оборудования и программного обеспечения для РСОД.
- Настройка рабочих параметров сетевой ОС и СУБД.
- Распределение предметных баз данных (ПБД) по узлам сети.
- Предложены возможные варианты для модернизации и реорганизации корпоративной сети фирмы.
- Аналитическое и имитационное моделирование РСОД.

Графическая часть содержит следующие материалы:

- Структурные схемы ЛВС центрального и удаленных филиалов фирмыи организации их удаленной связи (лист 1).
- Исходные данные и окончательные варианты распределения предметных баз данных по узлам сети (лист 2).
- Структурная схема моделируемой сети, исходные данные и результаты моделирования (лист 3). Листы выполнены в виде плакатов.

Содержание

2 Техническое задание

Исходные данные для выполнения курсовой работы представлены в таблице 1.

Таблица 1 – Исходные данные для выполнения курсовой работы.

№	Центральный офис	1-ый филиал	2-ой филиал	СУБД и выбор оборудования	Диски сервера	Диски сервера	Модель
8	5(1)/2(1)	3(3)/2(3)	18	23/33/43	53	64 (5) (0.9)	73

2.1 Наименование

Проектирование распределённой АСОИиУ фирмы.

2.2 Основание для разработки

Основанием для разработки является учебный план кафедры ИУ5 МГТУ им. Н.Э. Баумана.

2.3 Цель разработки

Целью разработки является создание проектного решения на распределенную сеть обра- ботки информации, объединяющую все подразделения фирмы, состоящей из центрального и двух удаленных офисов.

2.4 Задачи, подлежащие решению

В процессе выполнения курсовой работы необходимо решить следующие задачи:

1. Разработать блок-схему распределенной АСОИиУ фирмы и структурные схемы ЛВС цен- трального и удаленных офисов фирмы;
2. Описать правила построения всех сетей фирмы;
3. Выбрать и обосновать вариант удаленной связи отдельных ЛВС фирмы;
4. Выбрать требуемое оборудование для ЛВС фирмы;
5. Описать настройку рабочих параметров сетевой ОС, под управлением которой работают ЛВС фирмы;
6. Описать настройку рабочих параметров СУБД, которая установлена в ЛВС фирмы;
7. Провести распределение предметных баз данных по узлам сети;
8. Выполнить аналитическое и имитационное моделирование ЛВС фирмы и провести сравнительный анализ результатов моделирования;
9. Разработать и представить рекомендации по модернизации и реорганизации распределенной АСОИиУ фирмы.

2.5 Требования к составу технических средств

Фирма состоит из центрального офиса и двух филиалов.

В центральном офисе фирмы расположены ЛВС 100Base-TX, содержащая один концетратор, и ЛВС 10Base-2, содержащая один сегмент. Обе сети подключены к коммутатору, к нему также подключен удаленный маршрутизатор с двумя модемами.

В первом филиале фирмы расположены ЛВС 10Base-T, содержащая 3 концетратора, и ЛВС 10Base-2, содержащая 3 сегмента. Обе сети подключены к коммутатору, к нему также подключен удаленный маршрутизатор с одним модемом.

Во втором филиале фирмы расположена ЛВС Token Ring на комбинированной среде (ВОЛС (волоконно-оптическая линия связи) и ЭВП (экранированная витая пара)).

2.6 Требования к составу программных средств

ЛВС работает под управлением сетевой ОС Windows Server 2003 и СУБД SQL Server.

2.7 Техническая документация, предъявляемая по окончанию работы

По окончании работы должна быть предъявлена следующая документация:

- структурная схема объединённой сети фирмы;
- распределение предметных баз данных по узлам сети;
- формализованная схема сети и результаты моделирования;
- пояснительная записка.

2.8 Порядок приема работы

Приём работы осуществляется путем проверки соответствия выполненной работы пунктам технического задания.

2.9 Дополнительные условия

В ЛВС установлен сервер на базе двухядерного процессора с дисковой подсистемой RAID-6, содержащая 5 базовых диска (без учета уровня RAID), при условии, что вероятность безотказной работы одного диска равна 0.9 и все диски одинаковые.

Необходимо провести сравнительный анализ серверов отдела на базе двухядерных процессоров и выбрать наилучший.

Подробно описать особенности эксплуатации системного блока сервера.

Провести моделирование системы, содержащей ПЭВМ,канал и два сервера. Модель должна позволять проводить ее настройку на два варианта:

- на общий вид моделируемой ЛВС, чтобы можно было провести исследование любого варианта;
- на вариант задания (исходные данные - только данные задания).

Результаты моделирования должны соответствовать варианту задания, поэтому необходимо провести моделирование системы, содержащей 8 рабочих станций, ПЭВМ, канал и два сервера.

3 Архитектура объединенной сети фирмы

3.1 Общая схема сети

Объединенная сеть состоит из сети центрального офиса, сети 1 филиала и сети 2 филиала. Её общая схема изображена на рисунке 1. В графических обозначениях схема изображена на рисунке 2.

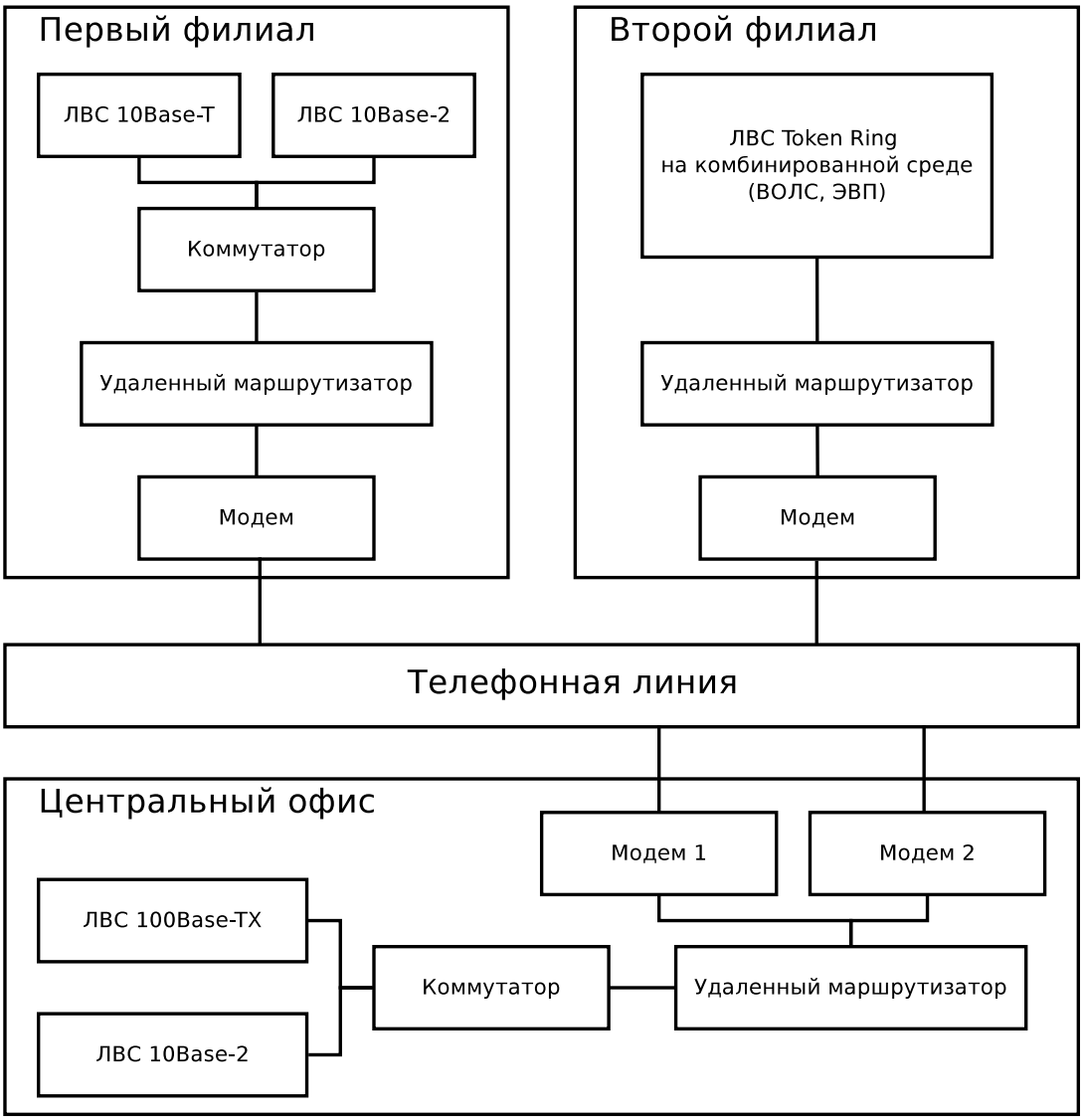

Рисунок 1 – Общая схема объединенной сети.

В центральном офисе фирмы расположены ЛВС 100Base-TX и ЛВС 10Base-2. Обе сети подключены к коммутатору, к которому также подключён удаленный маршрутизатор с двумя модемами.

В первом филиале фирмы расположены ЛВС 10Base-T и ЛВС 10Base-2. Обе сети подключены к коммутатору, к нему также подключен удаленный маршрутизатор с одним модемом.

Во втором филиале фирмы расположена ЛВС Token Ring на комбинированной среде (ВОЛС и ЭВП).

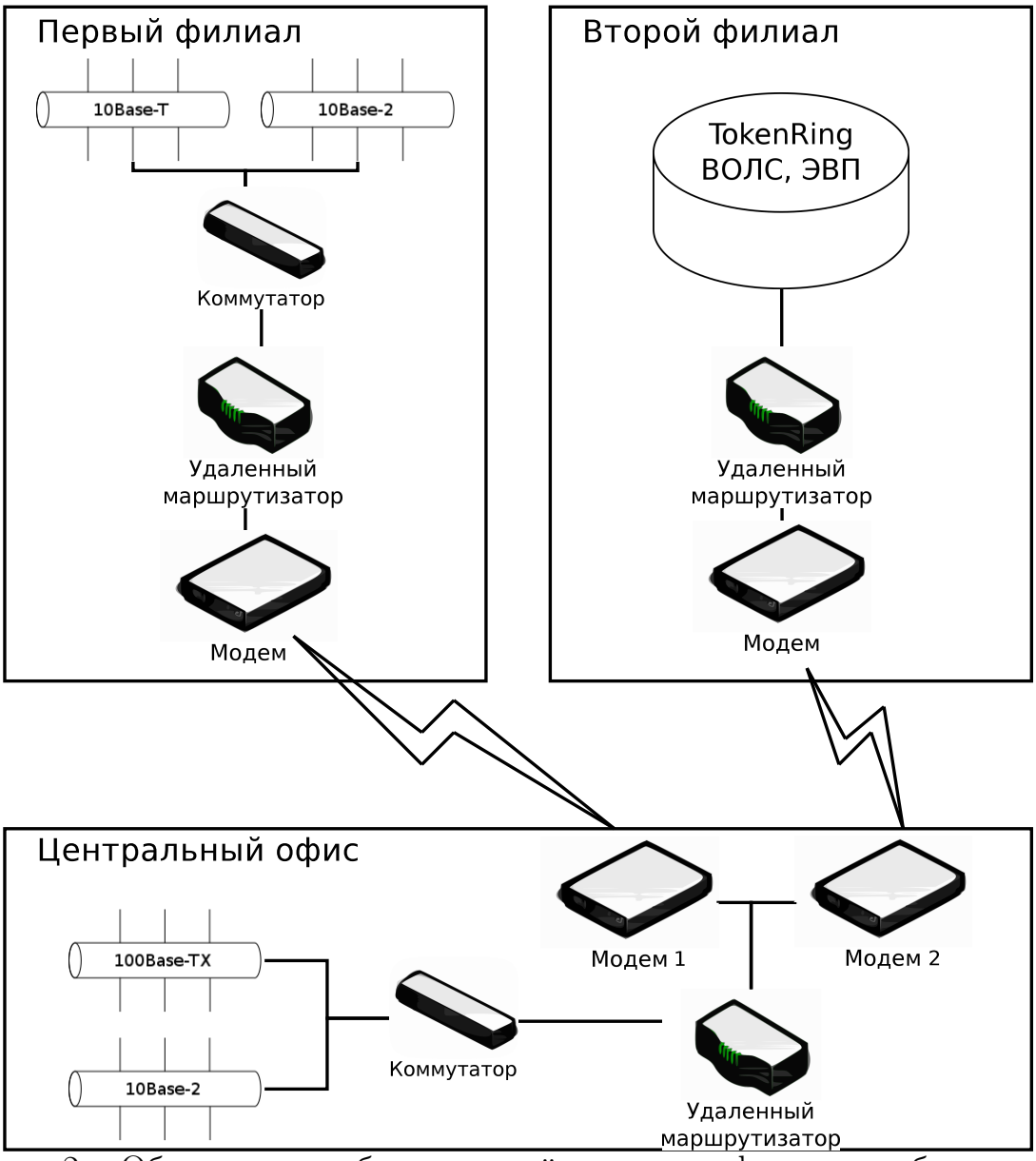

Рисунок 2 – Общая схема объединенной сети в графических обозначениях.

3.2 Схема сети центрального офиса

В центральном офисе расположены ЛВС 100Base-TX, содержащая один концентратор, и ЛВС 10Base-2, содержащая один сегмент. Обе сети подключены к коммутатору, к нему также подключены удаленный маршрутизатор с двумя модемами. В главном офисе расположен сервер фирмы. Схема сети представлена на рисунке 3.

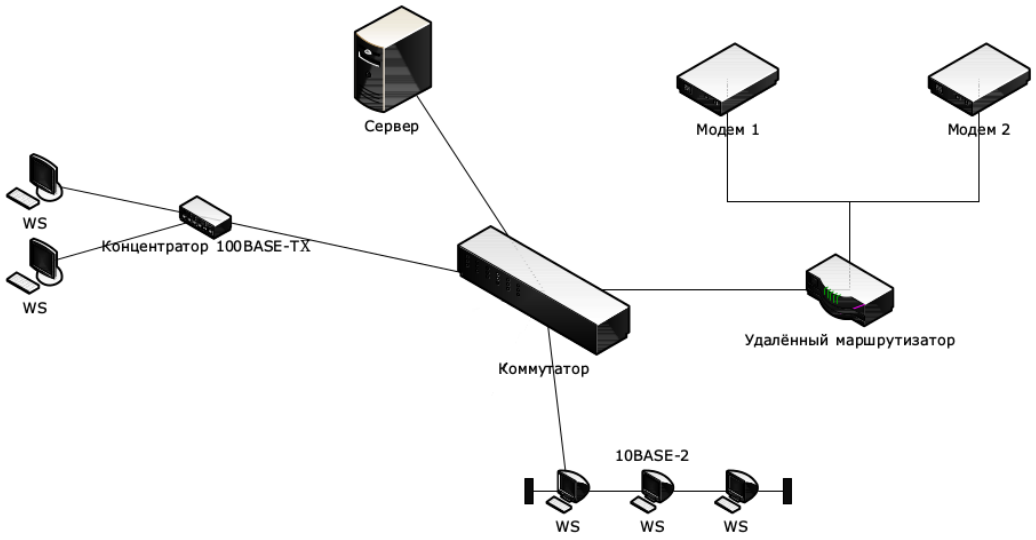

Рисунок 3 – Схема сети центрального офиса.

Во втором филиале фирмы расположена ЛВС Token Ring на комбинированной среде (ВОЛС (волоконно-оптическая линия связи) и ЭВП (экранированная витая пара)).

3.3 Схема сети первого филиала

В первом филиале фирмы расположены ЛВС 10Base-T, содержащая 3 концетратора, и ЛВС 10Base-2, содержащая 3 сегмента. Обе сети подключены к коммутатору, к нему также подключен удаленный маршрутизатор с одним модемом. Схема сети прдеставлена на рисунке 4.

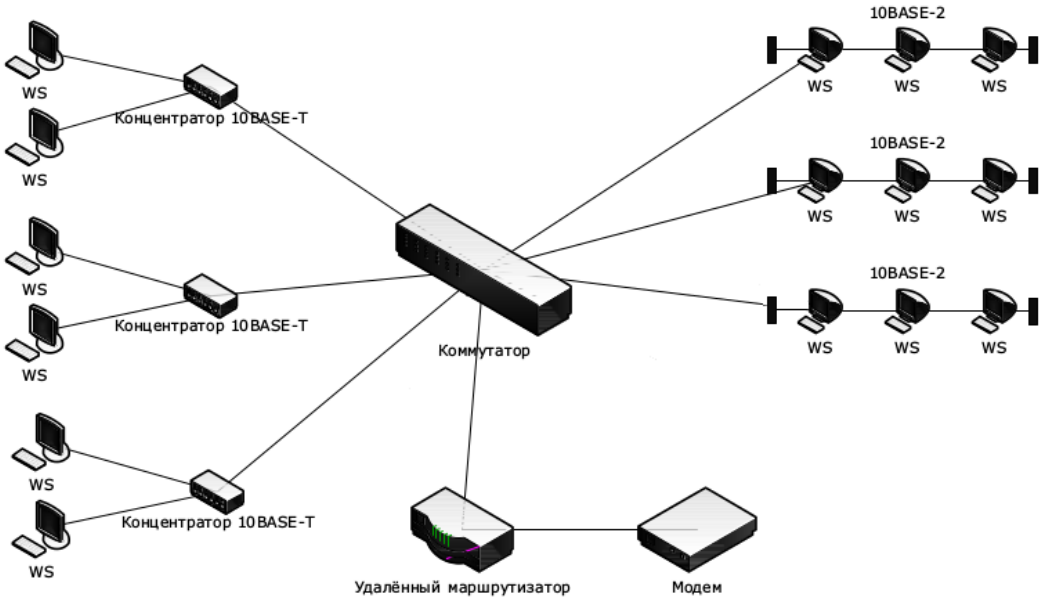

Рисунок 4 – Схема сети первого филиала.

3.4 Схема сети второго филиала

Во втором филиале фирмы расположена ЛВС Token Ring на комбинированной среде (ВОЛС (волоконно-оптическая линия связи) и ЭВП (экранированная витая пара)). Схема сети прдеставлена на рисунке 5.

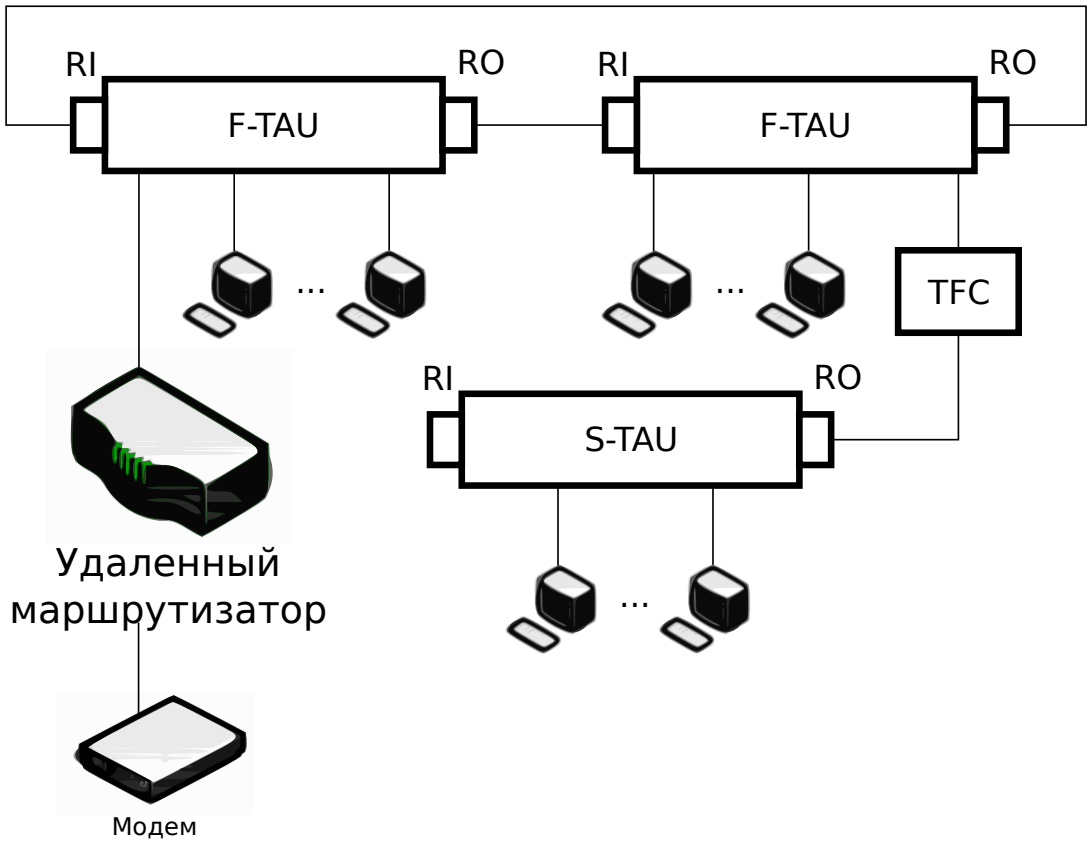

Рисунок 5 – Схема сети второго филиала.

3.5 Правила построения сети

Для построения работоспособной сети следует придерживаться ряда правил.

3.5.1 Правила построения сетей 100Base-TX

Для построения сети на основе 100Base-TX применяются следующие правила:

- используется витая пара 5 категории;
- в кабеле используются только 2 пары;
- длина луча сети должна быть не более 100 метров;
- между двумя узлами сети расстояние не может быть больше 205 м;
- между любыми двумя узлами должно быть не более 5 сегментов и 4 повторителей;

3.5.2 Правила построения сетей 10Base-T

При построении сети на основе 10BaseT применяются следующие правила:

- длина сегмента – 100 метров;
- между двумя узлами может быть не выше 5 лучей;
- общее количество станций не должно превышать 1024;
- максимальное число концентраторов между любыми двумя станциями сети - 4;
- максимальный диаметр сети - 500 м;
- используются коннекторы RJ-45;
- не допускается образование колец;
- используется неэкранированная витая пара 3 категории.

3.5.3 Правила построения сетей 10Base-2

При построении сети на основе 10Base2 применяются следующие правила:

- используется тонкий коаксиальный кабель с терминаторами на обоих концах;
- для усиления сигнала используются повторители;
- повторители должны быть подключены к источнику электропитания;
- рабочие станции подключаются к кабелю с помощью сетевых адаптеров с разъёмом BNC и Т-коннекторов;
- один из терминаторов каждого сегмента заземляется;
- Т-коннектор подключается к терминатору непосредственно или через расстояние кратное 2.5 м;
- минимальное расстояние между двумя Т-коннекторами составляет 2.5 м;
- расстояние между двумя Т-коннекторами должно быть кратно 2.5 м;
- между любыми двумя узлами должно находиться не более 5 сегментов, 4 повторителей;
- длина сегмента не более 185 м;
- длина сети не более 925 м;
- к сегменту может быть подключено не более 30 узлов.

3.5.4 Правила построения сетей Token Ring

При построении сети на основе Token Ring ЭВП и ВОЛС применяются следующие правила:

- в качестве кабеля используется экранированная витая пара (ЭВП) для S-TAU и волоконно оптические линии связи (ВОЛС) для F-TAU;
- для доступа к сети используется F-TAU и S-TAU;

- максимум в сети может быть не больше 255 узлов;
- для точки соединения между средами передачи сигнала используется TFC (Token ring optic Fiber Converter);
- максимальное расстояние между устройствами доступа S-TAU составляет 100 м, между устройствами доступа F-TAU 3000 м;

4 Построение сети

4.1 Принципы построения производительных сетей

Производительность системы определяется сочетанием её аппаратно- программных средств. Повышение производительности может быть достигнуто путем использования аппаратных средств, обладающих лучшими характеристиками производительности по сравнению с уже применяющи- мися.

Повышение производительности сервера, следует производить в соответствии с предвари- тельными расчетами «узких мест» – аналитическими расчетами, либо с помощью моделирования его работы. Эти расчеты показывают целесообразность увеличения производительности того или иного узла сервера - процессора, дисковой подсистемы. Аналогично для ЛВС можно определить, например, актуальность увеличения пропускной способности каналов связи.

Производительность сервера зависит от наличия:

- количества центральных процессоров;
- шин PCI и их большой производительности;
- большого объема памяти ОЗУ;
- высокоскоростного дискового интерфейса;
- организация дисковых подсистем с использованием RAID, обеспечивающих увеличение производительности.

4.2 Принципы построения отказоустойчивых сетей

Отказоустойчивость сети определяется двумя факторами:

- уровень избыточности сетевой инфраструктуры;
- время восстановления сети, то есть время, необходимое для переключения потоков данных на работоспособные части сети в случае отказа ее части.

При построении отказоустойчивой системы необходимо учесть следующее:

1. Архитектура сетевого оборудования:

    - дублирование блоков питания;
    - возможность "горячей"замены компонентов;
    - дублирование управляющего модуля;
    - дублирование коммутационной матрицы/шины.

2. Дублирование соединений:

    - использование нескольких дублирующих соединений;
    - не рекомендуется использовать протокол Spanning Tree, так как в сети появляется много неработающих (заблокированных) соединений, а время восстановления очень большое;
    - желательно использовать технологии Multi-Link Trunk (MLT) и Split-MLT (автоматиче- ская балансировка потоков данных между всеми работоспособными соединениями; вос- становление сети за доли секунды);
    - возможно внедрение протоколов балансировки нагрузки и дублирования на уровне марш- рутизации;
    - разнесение окончания каналов - окончание каналов на разных модулях ввода/вывода и/или на разных узлах для дополнительного дублирования;
    - разнесение каналов - использование различных носителей и различных путей для кри- тичных соединений;

3. Высоконадежное сетевое оборудование - устройства с высоким временем наработки на отказ;
4. Отказоустойчивость сервера:

    - использование технологии PCI Hot Plug замены отдельных узлов;
    - многопроцессорные серверы;
    - организация дисковых подсистем с использованием RAID, обеспечивающих увеличение надежности;

- дублирование дискового контроллера RAID;
- дублирование сетевых адаптеров;
- установка резервных вентиляторов для охлаждения процессора, ОЗУ, дисков, плат;
- организация резервного электропитания центрального процессора;
- резервирование источников питания;
- съемная плата кэш-памяти для диска со встроенной батареей;
- наличие заводского BIOS (ПЗУ) и рабочего BIOS (ППЗУ).

4.3 Расчет вероятности безотказной работы дисковой подсистемы сервера

Согласно ТЗ, необходимо определить вероятность безотказной работы дисковой подсисте- мы сервера, построенной на базе RAID-6, содержащей 5 базовых диска (без учета уровня RAID), при условии, что вероятность безотказной работы одного диска равна 0.9 и все диски одинаковые.

RAID 6 имеет высокую степень надёжности — под контрольные суммы выделяется ёмкость 2 дисков, рассчитываются 2 суммы по разным алгоритмам. Обеспечивает работоспособность после одновременного выхода из строя двух дисков — защита от кратного отказа. То есть, массив выходит из строя только при отказе трёх и более дисков. Схема дискового массива RAID-6 представлена на рисунке 6.

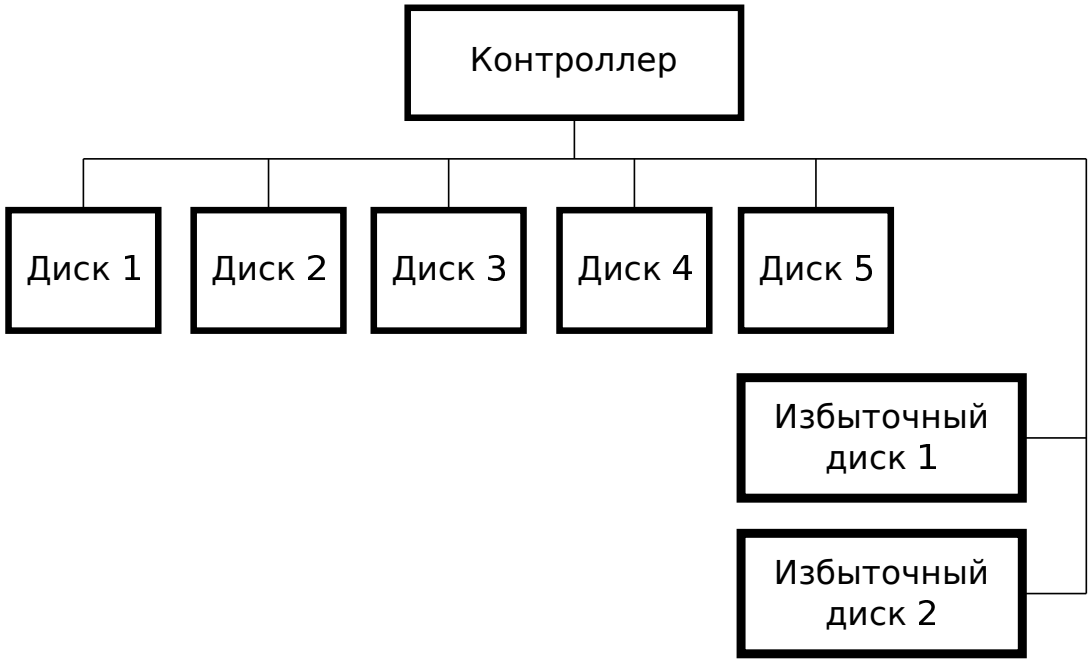

Рисунок 6 – Схема дискового массива RAID-6.

Вероятность безотказной работы массива RAID-6 для n дисков определяется по следующей формуле:

$$P = p^n + n(1-p)p^{n-1} + \binom{n}{2}(1-p)^2 p^{n-2}$$

Количество сочетаний вычисляется по следующей формуле:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Для $n = 5$ и $p = 0.9$ получаем следующую формулу:

$$P = 0.9^5 + 5 * 0.1 * 0.9^4 + \binom{5}{2} * 0.1^2 * 0.9^3 \quad (1)$$

$$= 0.59049 + 0.32805 + 10 * 0.00729 = 0.99144 \quad (2)$$

Вероятность безотказной работы дисковой подсистемы сервера на базе RAID-6 согласно формуле 1 равняется 0.99144.

4.4 Рекомендации по модернизации сети

В центральном офисе и в первом филиале в сети используются устаревшие технологии: 10BASE-T и 10BASE-2. И если первую ещё можно модернизировать путём замены сетевых адаптеров и концентраторов с возможностью использования уже проложенного кабеля, то вторая потребует ещё и замены всего оборудования плюс демонтаж старого и прокладку нового кабеля.

Если для работы фирмы достаточно текущей скорости передачи данных в сети и её пропускной способности, то не рекомендуется проводить модернизацию, так как это выльется в существенные финансовые и временные затраты.

5 Организация связи с филиалами

5.1 Выбор технологии

Мною было рассмотрено три варианта организации удалённых связей сети фирмы: VPN через Интернет, Frame Relay и ADSL.

VPN (Virtual Private Network — виртуальная частная сеть) - является современным способом создавать дешевые и защищенные каналы передачи данных между локальными сетями через сеть Интернет. Вместо модемов используются аппаратные или программные шлюзы, подключенные к сети Интернет. Данный подход позволяет сократить затраты на подключение, так как филиалы и главный офис разделяют соединение к сети Интернет вместе с виртуальными каналами VPN.

Стоимость организации VPN сети в среднем составляет 10 000 р., в которую входят стоимость аппаратного обеспечения и настройки программного обеспечения. Абонентская плата для скорости 10 Мбит/с составляет 2 700 р. в месяц.

Сети Frame Relay - сравнительно новые сети, которые гораздо лучше подходят для передачи пульсирующего трафика локальных сетей по сравнению с сетями X.25. Преимущество сетей Frame Relay заключается в их низкой протокольной избыточности и дейтаграммном режиме работы, что обеспечивает высокую пропускную способность и небольшие задержки кадров. Надежную передачу кадров технология Frame Relay не обеспечивает. Сети Frame Relay специально разрабатывались как общественные сети для соединения частных локальных сетей. Они обеспечивают скорость передачи данных до 2 Мбит/с.

Сети Frame Relay следует применять только при наличии на магистральных каналах волоконнооптических кабелей высокого качества. Стоимость использования данной сети достаточно высока. Подключение стоит 57000 рублей и абонентская плата составляет 9000 рублей в месяц.

ADSL модемы передают данные намного быстрее, чем обычные аналоговые модемы, используя те же телефонные линии. ADSL обеспечивает высокоскоростной широковещательный доступ. Скорость передачи данных достигает 6 Мб/с. ADSL работает по имеющейся телефонной линии и при этом не занимает телефонный канал. Таким образом, возможно одновременно пользоваться телефонным аппаратом и осуществлять доступ в интернет. Фактически, ADSL модем образует три канала:

- входящий канал, скорость до 6 Мбит/с,
- исходящий канал, скорость до 1 Мб/с,
- обычный канал телефонной связи, по которому ведутся телефонные разговоры.

Скорость передачи данных зависит от используемого оборудования, длины и качества телефонной линии. ADSL не требует, как аналоговый модем, набора номера для установления соединения с сетью. Обычно, телефонные сервисы используют минимальную часть пропускной способности телефонной линии, ADSL занимает оставшуюся часть для реализации высокоскоростной передачи данных. Подключение стоит 3 000 рублей, а абонентская плата составляет 2 500 рублей в месяц.

Качественные характеристики рассматриваемых технологий сведены в таблицу 2.

Таблица 2 – Качественные характеристики технологий передачи данных

Параметр	VPN	Frame Relay	ADSL
Скорость передачи, Мб/с	10	2	6
Надёжность передачи данных	Отличная	Отличная	Хорошая
Возможность масштабирования	Хорошая	Отличная	Плохая
Стоимость подключения, рублей	10 000	57 000	3 000
Абонентская плата, рублей	2 700	9 000	2 500

Количественные характеристики рассматриваемых технологий сведены в таблицу 3.

Таблица 3 – Количественные характеристики технологий передачи данных

Параметр	VPN	Frame Relay	ADSL
Скорость передачи, Мб/с	10	2	6
Надёжность передачи данных	5	5	4
Возможность масштабирования	4	5	2
Стоимость подключения, рублей	10 000	57 000	3 000
Абонентская плата, рублей	2 700	9 000	2 500

Теперь пронормируем параметры и добавим весовые коэффициенты. Результаты приведены в таблице 4.

Таблица 4 – Расчет весовых коэффицентов технологий передачи данных

Параметр	α	VPN	Frame Relay	ADSL
Скорость передачи, Мб/с	0.3	1	0.2	0.6
Надёжность передачи данных	0.3	1	1	0.8
Возможность масштабирования	0.2	0.8	1	0.4
Стоимость подключения, рублей	0.1	0.3	0.05	1
Абонентская плата, рублей	0.1	0.93	0.28	1
Итог	1.0	0.88	0.59	0.7

Согласно таблице 4 необходимо выбрать VPN.

5.2 Выбор оборудования

5.2.1 Выбор маршрутизатора

Выбираем из трех маршрутизаторов:

1. ASUS RT-N10P
2. TP-LINK TL-R470T+
3. D-link DIR-100

Качественные характеристики рассматриваемых маршрутизаторов сведены в таблицу 5.

Таблица 5 – Качественные характеристики рассматриваемых маршрутизаторов

Параметр	ASUS RT-N10P	TP-LINK TL-R470T+	D-link DIR-100
Количество портов	4	8	8
Легкость настройки	Отлично	Хорошо	Отлично
Размер таблицы маршрутизации	20	25	15
Гарантия, месяцев	36	24	24
Стоимость, рублей	720	1 790	1 550

Количественные характеристики рассматриваемых маршрутизаторов сведены в таблицу 7.

Таблица 6 – Качественные характеристики рассматриваемых маршрутизаторов

Параметр	ASUS RT-N10P	TP-LINK TL-R470T+	D-link DIR-100
Количество портов	0.5	1	1
Легкость настройки	1	0.8	1

Размер таблицы маршрутизации	0.8	1	0.6
Гарантия, месяцев	1	0.67	0.67
Стоимость, рублей	1	0.4	0.46

Теперь пронормируем параметры и добавим весовые коэффициенты. Результаты приведены в таблице 7.

Таблица 7 – Расчет весовых коэффицентов рассматриваемых маршрутизаторов

Параметр	α	ASUS RT-N10P	TP-LINK TL-R470T+	D-link DIR-100
Количество портов	0.4	0.5	1	1
Легкость настройки	0.1	1	0.8	1
Размер таблицы маршрутизации	0.1	0.8	1	0.6
Гарантия, месяцев	0.1	1	0.67	0.67
Стоимость, рублей	0.3	1	0.4	0.46
Итог	1.0	0.78	0.767	0.765

Следует предпочесть ASUS RT-N10P.

5.2.2 Выбор модема (шлюза)

На предыдущих этапах была выбрана технология VPN для соединения филиалов фирмы с главным офисом. Поэтому вместо модема производится выбор криптографического шлюза для организации VPN туннелей через сеть Интернет.

Выбор производится из следующих вариантов:

1. S-Terra L2 для CSP VPN Gate 100 - специализированное оборудование для поддержки нагруженных VPN сетей.
2. Слабомощный сервер на базе TopComp WO 3251432 с виртуальным VPN-шлюзом - VPN шлюз может быть развернут на любом ЭВМ, подключенном к сети Интернет, использование подходящей рабочей станции позволяет выдержать баланс между производительностью и стоимостью решения.
3. Использование маршрутизатора как VPN-шлюз - выбранный маршрутизатор содержит фукнцию VPN шлюза. Данный вариант не предполагает дополнительных расходов на приобретение оборудования, но создает серьезную нагрузку на маршрутизатор.

Качественные характеристики рассматриваемых шлюзов сведены в таблицу 8.

Таблица 8 – Качественные характеристики рассматриваемых шлюзов

Параметр	S-Terra L2	TopComp WO	Маршрутизатор
Количество туннелей	10	5	5
Легкость настройки	Плохо	Хорошо	Отлично
Масштабируемость	Отлично	Хорошо	Плохо
Гарантия, месяцев	36	24	36
Стоимость, рублей	6 750	4 190	1

Количественные характеристики рассматриваемых шлюзов сведены в таблицу 9.

Таблица 9 – Количественные характеристики рассматриваемых шлюзов

Параметр	S-Terra L2	TopComp WO	Маршрутизатор
Количество туннелей	1	0.5	0.5
Легкость настройки	0.2	0.8	1
Масштабируемость	1	0.8	0.2
Гарантия, месяцев	1	0.67	1
Стоимость, рублей	0.00015	0.00024	1

Теперь пронормируем параметры и добавим весовые коэффициенты. Результаты приведены в таблице 10.

Таблица 10 – Расчет весовых коэффицентов рассматриваемых шлюзов

Параметр	α	S-Terra L2	TopComp WO	Маршрутизатор
Количество туннелей	0.1	1	0.5	0.5
Легкость настройки	0.05	0.2	0.8	1
Масштабируемость	0.6	1	0.8	0.2
Гарантия, месяцев	0.05	1	0.67	1
Стоимость, рублей	0.2	0.00015	0.00024	1
Итог	1.0	0.76	0.60	0.47

Следует предпочесть специализированное оборудование: S-Terra L2 для CSP VPN Gate 100.

5.2.3 Выбор сервера

Выбор сервера производится из следующих вариантов серверов с двухядерным центральным процессором:

1. Preon Office 064
2. HP ProDesk 400 MT F4Q24EA
3. Lenovo ThinkCentre Edge 73 SFF 10AU0080RU

Выбор весовых коэффициентов параметров сравнения серверов будет проводиться с использованием метода анализа иерархий, а выбор сервера - с использованием метода взвешенной суммы параметров сравнения.

Качественные характеристики рассматриваемых серверов сведены в таблицу 11.

Таблица 11 – Качественные характеристики рассматриваемых серверов

Параметр	Preon	HP ProDesk	Lenovo ThinkCentre
Частота процессора, ГГц	2.5	2.7	2.7
Кеш процессора, Кб	2048	512	2048
Оперативная память, Гб	2	4	2
Объем жесткого диска, Гб	250	500	500
Мощность блока питания, Вт	350	420	650
Сетевой адаптер	LAN	LAN	LAN,WIFI
Стоимость, рублей	10003	12004	14020

Количественные характеристики рассматриваемых серверов сведены в таблицу 12.

Таблица 12 – Количественные характеристики рассматриваемых серверов

	Параметр	Preon	HP ProDesk	Lenovo ThinkCentre
К1	Частота процессора, ГГц	2.5	2.7	2.7
К2	Кеш процессора, Кб	2048	512	2048
К3	Оперативная память, Гб	2	4	2
К4	Объем жесткого диска, Гб	250	500	500

К5	Мощность блока питания, Вт	350	420	650
К6	Сетевой адаптер	4	4	5
К7	Стоимость, рублей	10003	12004	14020

Отсортируем критерии по их важности:

$$K_1 \succeq K_2 \succeq K_3 \succeq K_7 \succeq K_4 \succeq K_5 \succeq K_6$$

Весовые коэффициенты определяем методом анализа иерархий. Полученные результаты приведены в таблице 13.

Таблица 13 – Весовые коэффициенты

	К1	К2	К3	К7	К4	К5	К6	Вектор	α
К1	1	2	3	4	5	7	8	3.5218	0.34
К2	0.5	1	2	3	5	7	7	2.5673	0.25
К3	0.33	0.5	1	2	3	5	7	1.6594	0.16
К7	0.25	0.33	0.5	1	3	5	7	1.2329	0.12
К4	0.14	0.2	0.33	0.33	1	3	5	0.6436	0.06
К5	0.125	0.14	0.2	0.2	0.33	1	3	0.3537	0.03
К6	0.2	0.14	0.14	0.14	0.2	0.33	1	0.232	0.02
								10.2107	1.0

Собственный вектор рассчитывается по следующей формуле:

$$C_i = \sqrt[7]{K_{i1}K_{i2}K_{i3}K_{i7}K_{i4}K_{i5}K_{i6}}$$

Весовой коэффициент считается по следующей формуле:

$$\alpha_i = \frac{C_i}{\sum_{j=0}^{7} C_j}$$

Оцениваем отношение согласованности для данных весовых коэффициентов ($m_f = 7$, $R = 1.32$):

$$\lambda_{max} = \sum_{i=1}^{m_f}(\alpha_i \sum_{j=1}^{m_f} y_{ij}) = 7.36499$$

$$\text{4ОЦ} = \frac{\lambda_{max} - m_f}{(m_f - 1)R} = \frac{7.36499 - 7}{6 * 1.32} = 0.046$$

Как видим, $0.046 < 0.1$, значит матрица согласована.

Теперь пронормируем параметры и добавим весовые коэффициенты. Результаты приведены в таблице 14.

Таблица 14 – Количественные характеристики рассматриваемых серверов

	Параметр	α	Preon	HP ProDesk	Lenovo ThinkCentre
K1	Частота процессора, ГГц	0.34	0.93	1	1
K2	Кеш процессора, Кб	0.25	1	0.25	1
K3	Оперативная память, Гб	0.16	0.5	1	0.5
K4	Объем жесткого диска, Гб	0.06	0.5	1	1
K5	Мощность блока питания, Вт	0.03	0.54	0.65	1
K6	Сетевой адаптер	0.02	0.8	0.8	1
K7	Стоимость, рублей	0.12	1	0.83	0.71
		1.0	0.8284	0.7576	0.8652

Исходя из проведенного анализа, мы должны выбрать Lenovo ThinkCentre Edge 73 SFF 10AU0080RU.

5.2.4 Выбор системного блока

Выбор производится из следующих моделей:

- Corsair Carbide Series Air 540
- ZALMAN Z9 PLUS
- InWin ENR027BL

Качественные характеристики рассматриваемых системных блоков сведены в таблицу 15.

Таблица 15 – Качественные характеристики рассматриваемых системных блоков

Параметр	Corsair Carbide	ZALMAN Z9 PLUS	InWin ENR027BL
Слоты для плат расширения	8	7	4
Количество отсеков 5.25"	2	3	2
Количество портов USB	5	4	2
Вентиляторы	3	4	0
Стоимость, рублей	6660	3010	2360

Теперь пронормируем параметры и добавим весовые коэффициенты в таблицу 16.

Таблица 16 – Количественные характеристики рассматриваемых системных блоков

Параметр	α	Corsair Carbide	ZALMAN Z9 PLUS	InWin ENR027BL
Слоты для плат расширения	0.2	1.0	0.875	0.5
Количество отсеков 5.25"	0.2	0.67	1.0	0.67
Количество портов USB	0.2	1.0	0.8	0.4
Вентиляторы	0.1	0.75	1.0	0
Стоимость, рублей	0.3	0.35	0.78	1.0
	1.0	0.714	0.869	0.614

Из таблицы 16 видно, что следует выбрать системный блок ZALMAN Z9 PLUS.

6 Настройка рабочих параметров сетевой ОС

По условию технического задания используется Windows Server 2003.

6.1 Список рабочих параметров сетевой ОС

Разрешить менять права доступа к файлам и каталогам. С помощью этого параметра Вы можете разрешить или запретить возможность изменения прав доступа к локальным и разделяемым файлам и каталогам.

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa`

Forceguest: значение 1 или 0

Оптимизация процесса загрузки (Boot defrag). Суть boot defrag заключается в дефрагментации тех файлов, что нужны для старта операционной системы. Выключение этой функции позволит на некоторое время уменьшить время загрузки, но со временем она будет становиться все медленнее.

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Dfrg\BootOptimizeFunction`

Enable: значение y или n

Встроенная функция записи дисков С помощью этого параметра Вы можете разрешить или запретить использование встроенной функции записи CD-дисков.

`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion`
`\Policies\Explorer`

NoCDBurning: значение 1 или 0

Режим командной строки и обработки bat-файлов Параметр позволяет разрешить или запретить режим командной строки и обработку bat-файлов

`HKCU\Software\Policies\Microsoft\Windows\System`

DisableCMD: значение 2 или 1 или 0

Размер кэша Указывает максимальный объем места на диске, который может использоваться для файлового кэша защиты файлов Windows.

Защита файлов Windows добавляет защищенные файлы в кэш до тех пор, пока не будет достигнут предел квоты. Если квота больше 50 МБ, тогда защита файлов Windows добавляет другие важные файлы Windows 2003 в кэш до тех пор, пока не будет достигнут предел квоты.

`HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows NT`
`\Windows File Protection`

SfcQuota

Расширенный режим командного процессора Параметр позволяет запретить расширенный режим командного процессора (cmd.exe). Например, в расширенном режиме существуют такие команды как del, erase, chdir, goto.

`HKCU\Software\Microsoft\Command Processor`

EnableExtensions: значение 1 или 0

Ограничить пользователя единственным удаленным сеансом Ограничивает пользователей служб терминалов единственным сеансом.

По умолчанию, сервер терминалов позволяет пользователям неограниченное количество одновременных активных или отключенных сеансов для каждого удаленного пользователя. Эта политика используется для ограничения количества одновременных сеансов для каждого пользователя.

Если эта политика включена, пользователи, выполнившие удаленный вход на сервер терминалов, могут использовать только один сеанс на этом сервере. Если пользователь оставляет сеанс в отключенном состоянии, он будет автоматически переподключен к этому сеансу при следующем входе на сервер терминалов.

Если эта политика отключена, принудительно применяется поведение по умолчанию для неограниченного количества одновременных подключений.

Если эта политика не задана, количество пользовательских сеансов не определяется на уровне групповой политики.

`HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT`
`\Terminal Services`

fSingleSessionPerUser: значение 1 или 0

IP-адреса Указывает DNS-серверы, к которым компьютер посылает запросы на разрешение имен.

Предупреждение: определенный в этой политике список DNS-серверов подавляет списки DNS-серверов, которые были заданы локально или с помощью DHCP. Этот список DNS-серверов применяется для всех сетевых подключений многосетевых компьютеров, к которым применяется эта политика.

Чтобы использовать эту политику, выберите "Включить"и введите разделенный пробелами список IP-адресов (в разделенном точками формате) в соответствующее поле. Если эта политика включена, необходимо ввести по крайней мере один IP-адрес.

Если эта политика не задана, то она не применяется ни к каким компьютерам и компьютеры используют свои локальные или заданные с помощью DHCP параметры.

`HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows NT`
`\DNSClient`

NameServer: строка форматирования %s

7 Настройка рабочих параметров СУБД

По условию ТЗ используется СУБД SQL Server. Рассмотрим настройки рабочих параметров данной СУБД.

Вы можете задавать параметры конфигурирования одним из двух способов: путем выбора свойств в окне SQL Server Enterprise Manager или путем использования системной хранимой процедуры sp configure. Для использования хранимой процедуры sp configure нужно запустить следующую команду:

```
sp_configure 'имя_параметра', значение
```

affinity mask Параметр affinity mask (маска "родственности") является битовой переменной, которая указывает, на каких ЦП может выполнять свою работу SQL Server. Значение 0 (принятое по умолчанию) позволяет планировщику Microsoft Windows NT/2000 определять, на каких ЦП будет работать SQL Server. Поскольку это битовая переменная, то двоичное представление этого значения определяет, какие ЦП будут использоваться. Ниже приводятся пять первых двоичных значений.

```
0=0000
1=0001
2=0010
3=0011
4=0100
```

Например, если вы используете 4-процессорную систему, то можете задать для параметра affinity mask значение 15 (1111), чтобы SQL Server работал на всех ЦП.

awe enabled Параметр awe enabled (активизирована awe) разрешает использовать средство расширенной памяти Address Windowing Extensions (AWE) в Microsoft Windows 2000. Если для awe enabled задано значение 1, то разрешается использование памяти свыше 4 Гб.

index create memory Параметр index create memory (память для создания индекса) указывает количество памяти, используемое для сортировок при создании

индекса. Значение по умолчанию, равное 0, указывает, что это значение будет определять SQL Server.

max degree of parallelism Параметр max degree of parallelism (максимальная степень распараллеливания) указывает максимальное количество потоков, которые могут быть выделены для параллельного выполнения. Значение по умолчанию, равное 0, указывает, что будут использоваться все ЦП в данной системе, что делает количество потоков равным количеству ЦП в системе. Значение 1 запрещает параллельное выполнение. Поскольку распараллеливание может повысить производительность запросов, ограниченных возможностями ввода-вывода, вам, возможно, потребуется задать более высокое значение для параметра max degree of parallelism. Максимальное значение – 32.

max server memory Параметр max server memory (максимальная память сервера) используется для задания максимального количества памяти, которое может быть динамически выделено в SQL Server. Этот параметр используется в сочетании с параметром min server memory. Количество памяти, выделяемое в SQL Server, будет находиться между значениями, заданными для параметров min server memory и max server memory. Если вы хотите зарезервировать дополнительное пространство для процессов, отличных от SQL Server, то можете использовать этот параметр. Значение по умолчанию, равное 0, указывает, что SQL Server будет выделять память автоматически.

max worker threads Параметр max worker threads (максимальное количество потоков) указывает максимальное количество потоков Windows, которые может использовать SQL Server. Этот параметр можно изменять, чтобы предоставлять больше потоков для обработки в SQL Server. Но если SQL Server использует слишком много потоков, это приводит к перегрузке операционной системы.

media retention Параметр media retention (хранение носителя) указывает количество дней хранения носителя резервной копии. SQL Server не перезаписывает этот носитель резервной копии, пока не пройдет указанное количество дней.

min memory per query Параметр min memory per query (минимальная память на один запрос). Этот параметр указывает минимальное количество памяти, которое будет выделяться на один запрос. Значение по умолчанию – 1024, но вы

можете задать значение от 512 байтов до 2 Гб. Задание этого параметра таким образом, чтобы при запуске запроса выделялось указанное количество памяти, может способствовать повышению производительности при больших сортировках и операциях хеширования.

min server memory Параметр min server memory (минимальная память сервера) используется в сочетании с параметром max server memory для задания минимального и максимального количества памяти, которое будет использовать SQL Server. Значение по умолчанию, равное 0, указывает, что SQL Server будет выделять память автоматически.

query governor cost limit Параметр query governor cost limit (предел оценки стоимости запроса в секундах) указывает максимальное количество времени (в секундах), допустимое для выполнения запроса. Прежде чем запустить запрос, оптимизатор запросов оценивает длительность выполнения этого запроса. При соответствующих значениях этот параметр препятствует запуску слишком больших запросов.

query wait При недостатке памяти для запуска запроса SQL Server помещает этот запрос в очередь, пока не освободятся необходимые ресурсы. По умолчанию время ожидания в 25 раз превышает стоимость запроса. Задавая параметр query wait (время ожидания), вы указываете, тем самым, значение тайм-аута.

recovery interval Параметр recovery interval (интервал восстановления) очень важен. Значение параметр recovery interval указывает максимальное количество времени, которое может потратить SQL Server на восстановление (воспроизведение) базы данных в случае отказа системы, и, тем самым, этот параметр определяет, насколько часто будут создаваться контрольные точки. Принятое по умолчанию значение 0 указывает, что SQL Server будет определять это значение автоматически.

remote login timeout Параметр remote login timeout (тайм-аут удаленного входа) указывает допустимое время ожидания (в секундах) при входе по удаленной login-записи. Значение по умолчанию равно 5.

set working set size Параметр set working set size (установить размер рабочего набора) действует совместно с параметрами min server memory и max server memory. вы хотите, чтобы SQL Server захватывал память динамически, не задавайте для этого параметра значение 1. Принятое по умолчанию значение 0 отключает параметр set working set size.

show advanced options Если для параметра show advanced options(показать дополнительные параметры) задано значение 1, то вы можете использовать хранимую процедуру sp configure для показа дополнительных параметров.

user connections Параметр user connections (количество подсоединений пользователей) указывает максимальное количество пользователей, которые могут одновременно подсоединяться к SQL Server. По умолчанию SQL Server динамически регулирует допустимое количество подсоединений пользователей, но это динамическое регулирование создает дополнительную нагрузку. Данный параметр позволяет вам задавать статически допустимое количество подсоединений пользователей.

8 Распределение предметных БД по узлам сети

Согласно ТЗ, необходимо распределить предметные БД по узлам сети без учёта реплика- ции.

Необходимо определить вариант рационального размещения предметных баз данных в распределенной информационной системе для случая, когда каждая база данных размещается только в одном узле сети, а обрабатывающие процессы (приложения) не являются распределенными. При этом следует считать, что если некоторый процесс обращается за данными к базе, находящейся в другом узле, сетевые затраты на одно обращение составляют t секунд, независимо от местонахождения узла в сети и дисциплины обслуживания. Если процесс обращается к базе данных, находящейся в том же узле, где выполняется процесс, то следует считать, что $t = 0$.

Таблица 17 показывает использование предметных баз данных обрабатывающими процес- сами (приложениями) в течение временного интервала и интенсивности их обращений к базам данных (среднее число обращений за рассматриваемый интервал времени).

Таблица 17 – Использование предметных баз данных

	БД1	БД2	БД3	БД4	БД5	БД6	БД7	БД8	БД9	БД10
П1	100			60		150				140
П2		400	300					250		
П3	30		300		80		400		20	180
П4		300	150			100				
П5					85		300		30	
П6							200	300		110
П7	50			70					40	150
П8			200	60	75					
П9		350	300			100		400		
П10			240		90				40	

Таблица 18 показывает распределение обрабатывающих процессов по узлам распределён- ной сети.

Таблица 18 – Таблица, показывающая распределение обрабатывающих процессов по узлам

	П1	П2	П3	П4	П5	П6	П7	П8	П9	П10
У1			1.4				0.6		0.9	
У2						0.7	1.0			0.95
У3			1.05				1.15		0.55	0.7
У4			0.9				0.9		0.5	0.8
У5			1.3			1.6	1.1			
У6						1.6			0.6	0.7

Коэффициенты в таблице 18 используются для получения количества обращений к базе данных в исходном варианте задания по формуле:

$$N_1 = Nk$$

Где:

- N - значение из таблицы 17
- k - значение из таблицы 18
- N_1 - результирующее значение для таблицы учебного варианта задания

На основании данных из таблицы 17 и таблицы 18 для исходного варианта была сформирована сводная таблица исходных данных – таблица 19. Каждое значение этой таблицы есть среднее количество обращений к базе данных ($БД_i$) определенного процесса ($П_j$) из определенного узла сети ($У_k$).

Таблица 19 – Результаты затрат на обработку конкретной БД конкретными процессами в конкретных устройствах без учёта репликации

	$П_j$	k	БД1	БД2	БД3	БД4	БД5	БД6	БД7	БД8	БД9	БД10
	П3	1.4	42		420		112		560		28	252
У1	П7	0.6	30			42					24	90
	П9	0.9		315	270			90		360		
	П6	0.7							140	210		77
У2	П7	1.0	50			70					40	150
	П10	0.95			228		85				38	
	П3	1.05	31		315		84		420		21	189
	П7	1.15	57			80					46	172
У3												

	П9	0.55		192	165			55		220		
	П10	0.7			168		63				28	
У4	П3	0.9	27		270		72		360		18	162
	П7	0.9	45			63					36	135
	П9	0.5		175	150			50		200		
	П10	0.8			192		72				32	
У5	П3	1.3	39		390		104		520		26	234
	П6	1.6							320	480		176
	П7	1.1	55			77					44	165
У6	П6	1.6							320	480		176
	П9	0.6		210	180			60		240		
	П10	0.7			168		63				28	

Составляем таблицу 20, в которой указываем все возможные варианты размещения баз данных по узлам сети. В каждую клетку этой таблицы записываем число, которое определяет суммарное количество всех запросов от всех процессов всех узлов к данной БД, при условии, что эта БД находится в данном узле.

Таблица 20 – Таблица, показывающая распределение обрабатывающих процессов по узлам

	БД1	БД2	БД3	БД4	БД5	БД6	БД7	БД8	БД9	БД10
У1	304	577	2226	290	543	165	2062	1830	357	1636
У2	326	892	2688	262	570	255	2482	1980	331	1751
У3	288	700	2268	252	508	200	2202	1970	314	1617
У4	304	717	2304	269	511	205	2262	1990	323	1681
У5	282	892	2526	255	551	255	1800	1710	339	1403
У6	376	682	2568	332	592	195	2302	1470	381	1802
max	376	892	2688	332	592	255	2482	1990	381	1802

Используем правило: "Базу данных помещаем в тот узел, где она максимально используется, то есть суммарное количество обращений к ней со стороны других узлов минимально". Поэтому в каждом столбце, соответствующем одной конкретной БД, отыскиваем наименьшее значение. Это и будет соответствовать оптимальному варианту размещения этой БД, поскольку чем меньше это значение, тем меньше суммарное количество обращений от всех процессов всех других

узлов к данной БД.

Полученные результаты, показывающие оптимальные варианты размещения БД по узлам сети, записываем в таблицу 21.

Таблица 21 – Оптимальные варианты размещения БД по узлам сети

Вариант	БД1	БД2	БД3	БД4	БД5	БД6	БД7	БД8	БД9	БД10
1	У5	У1	У1	У3	У3	У1	У5	У6	У3	У5
$N_{\text{обр}}$	282	577	2226	252	508	165	1800	1470	314	1403

Итого получили единственное оптимальное распределение БД по узлам сети, суммарные затраты составляют 8997.

9 Моделирование сети

Необходимо провести моделирование системы, содержащей 8 рабочих станций, канал и два сервера.

Общая формализованная схема РСОД в виде сети массового обслуживания (СМО) приведена на рисунке 7.

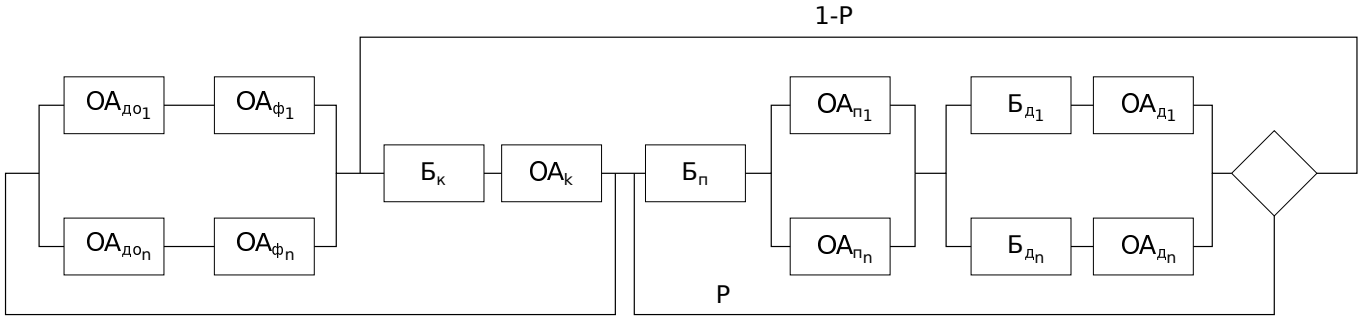

Рисунок 7 – Общая формализованная схема РСОД

В схеме 7 используются следующие обозначения:

- $ОА_{до_i}$ - обслуживающий аппарат, имитирующий дообработку на i-ой рабочей станции сети запроса от этой станции к серверу после обработки запроса на сервере;
- $ОА_{ф_i}$ - обслуживающий аппарат, имитирующий формирование запроса от i-ой рабочей станции к серверу;
- $Б_к$ - буфер, имитирующий очередь запросов к каналу;
- $ОА_к$ - обслуживающий аппарат, имитирующий задержку при передаче данных в канале;
- $Б_п$ - буфер, имитирующий очередь запросов к процессорам;
- $ОА_п$ - обслуживающий аппарат, имитирующий работу i-ого процессора;
- $Б_{д_i}$ - буфер, имитирующий очередь к i-ому диску;
- $ОА_{д_i}$ - обслуживающий аппарат, имитирующий работу i-ого диска;
- P - вероятность обращения запроса к ЦП после обработки на диске. Обслуживание заявок во всех ОА подчиняется экспоненциальному закону.

Формализованная схема рассматриваемой РСОД в виде СМО приведена на рисунке 8.

На схеме 8 используются следующие обозначения:

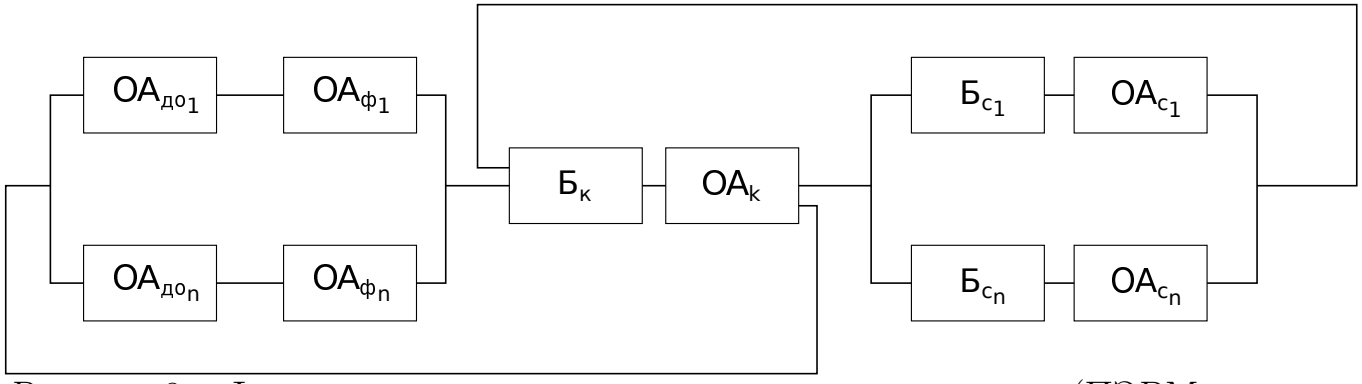

Рисунок 8 – Формализованная схема рассматриваемой модели (ПЭВМ, канал и два сервера)

- $OA_{\text{до}_i}$ - обслуживающий аппарат, имитирующий дообработку на i-ой рабочей станции сети запроса от этой станции к серверу после обработки запроса на сервере;
- $OA_{\text{ф}_i}$ - обслуживающий аппарат, имитирующий формирование запроса от i-ой рабочей станции к серверу;
- Б$_\text{к}$ - буфер, имитирующий очередь запросов к каналу;
- $OA_\text{к}$ - обслуживающий аппарат, имитирующий задержку при передаче данных в канале;
- Б$_{\text{с}_i}$ - буфер, имитирующий очередь запросов к i-ому серверу;
- $OA_{\text{с}_i}$ - обслуживающий аппарат, имитирующий работу i-ого сервера.

Таблица 22 – Исходные данные модели

Параметр	Описание
T_o	Среднее время дообработки на рабочей станции сети запроса от этой станции к серверу
T_p	Среднее время формирования запроса от рабочей станции сети к серверу
t_k	Среднее значение времени передачи запроса по каналу
m	Число серверов
n	Число рабочих станций в сети
t_s	Среднее время обработки запроса в сервере

Таблица 23 – Выходные данные модели

Параметр	Описание
$T_{\text{реак}}$	Среднее время реакции системы
ρ_{pc}	коэффициент загрузки ОА, имитирующего работу рабочей станции сети
ρ_k	коэффициент загрузки ОА, имитирующего работу канала передачи данных
$\rho_{\text{с}_i}$	коэффициент загрузки ОА, имитирующего работу i-ого сервера

9.1 Аналитическое моделирование

Введем следующие обозначения:

- $\lambda_{\text{ф}_i}$ - среднее значение суммарной интенсивности фонового потока запросов, выходящих из ОА, имитирующих работу рабочих станций, в канал (на i-ом шаге алгоритма);
- $t_k = \frac{t_{k1}+t_{k2}}{2}$ - среднее время обработки запроса в канале передачи данных, где t_{k1} и t_{k2} - соответственно среднее время передачи запроса по каналу в прямом и обратном направлениях;
- $P_i = \frac{1}{m}$ - вероятность обращения к i-ому серверу.

При расчете используется приближенный итерационный алгоритм нахождения значения выходных характеристик рассматриваемой системы.

1) Определяем начальное значение для $\lambda_{\text{ф}_i}$:

$$\lambda_{\text{ф}_i} = K_1 \min\left\{\frac{1}{2t_k}, \frac{1}{P_i t_s}\right\} \frac{n-1}{n},\ K_1 \in [0.995..0.99995]$$

Определяем средние времена пребывания запроса в узлах системы (канале, сервере):

$$T_k = \frac{2t_k}{1 - 2\lambda_{\text{ф}_i} t_k}$$

$$T_s = \frac{t_s}{1 - P_i \lambda_{\text{ф}_i} t_s}$$

2) Определяем интенсивность фонового потока после очередной итерации:

$$\lambda'_{\text{ф}_{i+1}} = \frac{n+1}{T_o + T_p + T_k + T_s}$$

3) Сравниваем $\lambda_{\text{ф}_i}$ и $\lambda'_{\text{ф}_{i+1}}$. Если:

$$\frac{|\lambda_{\text{ф}_i} - \lambda'_{\text{ф}_{i+1}}|}{\lambda_{\text{ф}_i}} < \delta$$

То переходим на пункт 5, иначе на пункт 4;

4) Определяем новое приближенное значение для $\lambda_{\text{ф}_i}$:

$$\lambda_{\text{ф}_{i+1}} = \lambda_{\text{ф}_i} - \frac{\lambda_{\text{ф}_i} - \lambda'_{\text{ф}_{i+1}}}{K_2},\ K_2 \in [10..1000]$$

Переход на пункт 2.

5) Определяем выходные результаты аналитической модели:

$$T_{\text{цикла}} = T_o + T_p + T_k + T_s$$

$$\lambda = \frac{N}{T_{\text{цикла}}}$$

Определяем загрузку основных узлов системы:

Рабочая станция:

$$\rho_{pc} = \frac{T_o + T_p}{T_{\text{цикла}}}$$

Пользователя:

$$\rho_{\text{польз}} = \frac{T_p}{T_{\text{цикла}}}$$

Канала передачи данных:

$$\rho_k = 2\lambda t_k$$

Сервера:

$$\rho_s = \lambda P_i t_s$$

Результаты аналитического моделирования представлены в таблице 24.

Таблица 24 – Результаты аналитического моделирования

Проведенные эксперименты					
	1	2	3	4	5
Количество рабочих станций	8	8	8	8	8
Среднее время дообработки запроса на PC	80	160	240	160	80
Среднее время формирования запроса на PC	80	160	240	160	80
Количество серверов	2	2	2	2	2
Среднее время обработки запроса на сервере	10	20	20	10	20
Среднее время передачи запроса по каналу	10	20	20	10	20
Результаты моделирования					
Среднее время реакции системы	225	450	580	364	362
Коэффициент загрузки рабочей станции	0.71	0.71	0.83	0.88	0.44
Коэффициент загрузки канала передачи данных	0.71	0.71	0.55	0.44	0.88
Коэффициент загрузки сервера	0.18	0.18	0.14	0.11	0.22
Коэффициент загрузки пользователя	0.36	0.36	0.41	0.44	0.22

9.1.1 Исходный код программы для аналитического моделирования

Использовался язык программирования D:

```
import vibe.data.json;

import std.array;
import std.getopt;
import std.file;
import std.stdio;
import std.math;

struct Input
{
    size_t wokrstationsCount;
    size_t serversCount;
    double afterProcessTime;
    double queringProcessTime;
    double responseProcessTime;
```

```
    double sendingProcessTime;

    static void genExample(string filename)
    {
        auto example = Input(8, 2, 80, 80, 10, 10).serializeToJson;
        auto builder = appender!string;

        writePrettyJsonString(builder, example);

        auto file = File(filename, "w");
        scope(exit) file.close();
        file.write(builder.data);
    }
}

struct Output
{
    double systemResponseTime;
    double workstationLoad;
    double cableLoad;
    double serverLoad;
    double userLoad;

    void save(string filename)
    {
        auto builder = appender!string;
        writePrettyJsonString(builder, this.serializeToJson);

        auto file = File(filename, "w");
        scope(exit) file.close();
        file.write(builder.data);
    }

    void toString(scope void delegate(const(char)[]) sink) const
    {
        writePrettyJsonString(sink, this.serializeToJson);
    }
}

void main(string[] args)
{
    string exampleInputName = "";
    bool needHelp = false;
```

```
    string inputFileName = "";
    string outputFileName = "output.json";

    getopt(args,
        "example", &exampleInputName,
        "input", &inputFileName,
        "output", &outputFileName,
        "help|h", &needHelp
    );

    if(needHelp)
    {
        writeln(helpMsg);
        return;
    }

    if(exampleInputName != "")
    {
        Input.genExample(exampleInputName);
        return;
    }

    if(inputFileName == "")
    {
        writeln("Input file isn't specified!");
        writeln(helpMsg);
        return;
    }

    immutable input = deserializeJson!Input(readText(inputFileName));
    writeln("Readed input: ", input);
    input.solve.save(outputFileName);
}

T min(T)(T a, T b)
{
    return a <= b ? a : b;
}

Output solve(const Input input)
{

    enum K1 = 0.995;
```

```
enum K2 = 100;
enum delta = 0.0001;

immutable Pi = 1 / cast(double)input.serversCount;
double background = K1 * min( 1 / (2*input.sendingProcessTime), 1 /
    ↪ (Pi*input.responseProcessTime) ) * ((input.wokrstationsCount -
    ↪  1)/cast(double)input.wokrstationsCount);
writeln("Initial background", background);

Output finalCalc(double Tk, double Ts)
{
    Output output;
    output.systemResponseTime = input.afterProcessTime + input.
        ↪ queringProcessTime + Tk + Ts;
    immutable lambda = input.wokrstationsCount / output.
        ↪ systemResponseTime;

    output.workstationLoad = (input.afterProcessTime + input.
        ↪ queringProcessTime) / output.systemResponseTime;
    output.userLoad = input.queringProcessTime / output.
        ↪ systemResponseTime;
    output.cableLoad = 2 * lambda * input.sendingProcessTime;
    output.serverLoad = lambda * Pi * input.responseProcessTime;

    writeln("Output calculated: ", output);
    return output;
}

enum maxIterations = 10000;
size_t lastIteration = 0;
double lastTk, lastTs, lastDiff;
foreach(i; 0..maxIterations)
{
    writeln("Iteration ", i);

    immutable Tk = (2*input.sendingProcessTime)/(1 - 2*background*
        ↪ input.responseProcessTime); lastTk = Tk;
    immutable Ts = input.responseProcessTime / (1 - Pi*background*
        ↪ input.responseProcessTime); lastTs = Ts;
    immutable backgroundTest = (input.wokrstationsCount - 1) / (
        ↪ input.afterProcessTime + input.queringProcessTime + Tk +
        ↪ Ts);
```

```
        immutable diff = abs(background - backgroundTest) / background;
            ↪ lastDiff = diff;

        writeln("Tk = ", Tk);
        writeln("Ts = ", Ts);
        writeln("New background = ", backgroundTest);
        writeln("diff = ", diff);

        if(diff < delta)
        {
            writeln("Diff is sufficient");
            return finalCalc(Tk, Ts);
        }

        writeln("Diff isn't sufficient");
        background = background - (background - backgroundTest) / cast(
            ↪ double)K2;
        writeln("Setting background to ", background);
        lastIteration = i;
    }

    writeln("Stopping at iteration ", lastIteration, " with diff = ",
        ↪ lastDiff);
    return finalCalc(lastTk, lastTs);
}

immutable helpMsg =
`Tool for analytic modeling for coursework IU5-92 Gushcha Anton 2014.

Arguments:
    --example=name - generate example input file at name
    --input=name - load input data from name
    --output=name - save result to name, default is 'output.json'
    --help or -h - show this message
`;
```

9.2 Имитационное моделирование

Иммитационное моделирование рассматриваемой РСОД было проведено с помощью языка D.

Результаты имитационного моделирования представлены в таблице 25.

Таблица 25 – Результаты имитационного моделирования

Проведенные эксперименты					
	1	2	3	4	5
Количество рабочих станций	8	8	8	8	8
Среднее время дообработки запроса на PC	80	160	240	160	80
Среднее время формирования запроса на PC	80	160	240	160	80
Количество серверов	2	2	2	2	2
Среднее время обработки запроса на сервере	10	20	20	10	20
Среднее время передачи запроса по каналу	10	20	20	10	20
Результаты моделирования					
Среднее время реакции системы	220	459	574	373	346
Коэффициент загрузки рабочей станции	0.78	0.70	0.84	0.89	0.46
Коэффициент загрузки канала передачи данных	0.67	0.76	0.56	0.40	0.97
Коэффициент загрузки сервера	0.16	0.17	0.14	0.11	0.21
Коэффициент загрузки пользователя	0.42	0.37	0.40	0.50	0.25

9.2.1 Исходный код програмы для иммитационного моделирования

Использовался язык программирования D:

```
module sim;

import std.algorithm;
import std.datetime;
import std.concurrency;
import std.container.dlist;
import std.typecons;
import std.random;
import core.thread;

import data;
import util;
```

```
struct Query
{
    size_t workstation;
    size_t server;

    TickDuration startStamp;
}

struct Response
{
    size_t workstation;

    TickDuration startStamp;
}

void workstationProcess(Duration To, Duration Tp, size_t workstationId,
    ↪ size_t serversCount)
{
    Thread.getThis.isDaemon = true;
    Tid cableProcessTid = receiveOnly!Tid;

    Duration totalLifetime = 0.dur!"msecs";
    size_t totalResponses = 0;
    auto startStamp = Clock.currAppTick;
    Duration loadTime = 0.dur!"msecs";
    Duration userLoadTime = 0.dur!"msecs";

    void postResponse(Response r)
    {
        loadTime += waitExp(To);
        totalLifetime += Clock.currAppTick - r.startStamp;
        totalResponses += 1;
    }

    void genQuery()
    {
        auto query = Query(workstationId, uniform(0, serversCount), Clock
            ↪ .currAppTick);
        auto t = waitExp(Tp.exp);
        loadTime += t;
        userLoadTime += t;
        cableProcessTid.send(query);
    }
```

```
        genQuery();


        while(true) {
            receive(
                (Response r) {
                    postResponse(r);
                    genQuery();
                },
                (Tid asker) {
                    double reactionTime = totalLifetime.total!"msecs" / cast
                        ↪ (double)totalResponses;
                    auto totalTime = cast(double)(cast(Duration)(Clock.
                        ↪ currAppTick - startStamp)).total!"msecs";
                    double loadFactor = loadTime.total!"msecs" / totalTime;
                    double userLoadFactor = userLoadTime.total!"msecs" /
                        ↪ totalTime;
                    asker.send(reactionTime, loadFactor, userLoadFactor);
                }
            );
        }
    }

    void cableProcess(Duration tk, const Tid[] workstations, const Tid[]
        ↪ servers)
    {
        Thread.getThis.isDaemon = true;
        auto startStamp = Clock.currAppTick;
        Duration loadTime = 0.dur!"msecs";

        while(true) {
            receive(
                (Response r) {
                    loadTime += waitExp(tk);
                    (cast()workstations[r.workstation]).send(r);
                },
                (Query q) {
                    loadTime += waitExp(tk);
                    (cast()servers[q.server]).send(q);
                },
                (Tid asker) {
```

```
                auto totalTime = cast(double)(cast(Duration)(Clock.
                    ↪ currAppTick - startStamp)).total!"msecs";
                asker.send(loadTime.total!"msecs" / totalTime);
            }
        );
    }
}

void serverProcess(Duration ts)
{
    Thread.getThis.isDaemon = true;
    Tid cableProcessId = receiveOnly!Tid;
    TickDuration startStamp = Clock.currAppTick;
    Duration loadTime = 0.dur!"msecs";

    while(true) {
        receive(
            (Query q) {
                loadTime += waitExp(ts);
                cableProcessId.send(Response(q.workstation, q.startStamp
                    ↪ ));
            },
            (Tid asker) {
                auto totalTime = cast(double)(cast(Duration)(Clock.
                    ↪ currAppTick - startStamp)).total!"msecs";
                asker.send(loadTime.total!"msecs" / totalTime);
            }
        );
    }
}

Duration mapTime(double v)
{
    return dur!"msecs"(cast(long)v);
}

Output startSimulation(Input input, Duration maxSimTime)
{
    Tid[] workstations;
    foreach(i; 0..input.wokrstationsCount)
    {
        workstations ~= spawn(&workstationProcess, input.
            ↪ afterProcessTime.mapTime, input.queringProcessTime.mapTime
```

```
        ↪ , i, input.serversCount);
}

Tid[] servers;
foreach(i; 0..input.serversCount)
{
    servers ~= spawn(&serverProcess, input.responseProcessTime.
        ↪ mapTime);
}

Tid cableProcessId = spawn(&cableProcess, input.sendingProcessTime.
    ↪ mapTime, cast(immutable)workstations, cast(immutable)servers);

foreach(w; workstations) w.send(cableProcessId);
foreach(s; servers) s.send(cableProcessId);

Thread.sleep(maxSimTime);
std.stdio.writeln("Simulation ended!");

Tuple!(double, double, double)[] times;
foreach(w; workstations)
{
    w.send(thisTid);
    times ~= receiveOnly!(double, double, double);
}
auto wStat = times.mean;

cableProcessId.send(thisTid);
auto cableLoadFactor = receiveOnly!double();

double[] serverTimes;
foreach(s; servers)
{
    s.send(thisTid);
    serverTimes ~= receiveOnly!double;
}
auto serverLoadFactor = serverTimes.mean;

Output output;
output.systemResponseTime = wStat[0];
output.workstationLoad = wStat[1];
output.userLoad = wStat[2];
output.cableLoad = cableLoadFactor;
```

```
    output.serverLoad = serverLoadFactor;

    return output;
}
```

9.3 Сравнительный анализ результатов моделирования

Сравнение результатов аналитического и имитационного моделирования приведено ниже в таблице 26.

Таблица 26 – Сравнение результатов аналитического и имитационного моделирования

№	Модель	Цикл	Рабст	Канал	Сервер	Польз
1	Аналитическая	225	0.71	0.71	0.18	0.36
	Иммитационная	220	0.78	0.67	0.16	0.42
2	Аналитическая	450	0.71	0.71	0.18	0.36
	Иммитационная	459	0.70	0.76	0.17	0.37
3	Аналитическая	580	0.83	0.55	0.14	0.41
	Иммитационная	574	0.84	0.56	0.14	0.40
4	Аналитическая	364	0.88	0.44	0.11	0.44
	Иммитационная	373	0.89	0.40	0.11	0.50
5	Аналитическая	362	0.44	0.88	0.22	0.22
	Иммитационная	346	0.46	0.97	0.21	0.25

Сравнительный анализ приведенных результатов разных моделей показывает разницу в значения не больше 10%, что является хорошим результатом для приближенных моделей.

Разница в различиях может быть объяснена следующим:

- При расчете аналитической модели использовался приближенный итерационных алгоритм;
- При иммитационном моделировании задавалось конечное время моделирования. Использовалась приближенная функция экспоненциального

распределения времени обслуживания. Также дополнительную погрешность добавляет системное окружение и системный планировщик потоков.

10 Выводы

В данной работе было разработан проект на построение распределенной АСОИиУ фирмы с дополнительными филиалами и получены следующие результаты:

- выбрана структура сетей для центрального офиса и филиалов в соответствии с заданными параметрами;
- простроена структурные схемы ЛВС главного офиса и удаленных филиалов;
- описаны правила простроения сетей фирмы;
- для удаленной связи между офисом и фиалами выбрана технология VPN через сеть Internet, как наиболее подходящяя для поставленных целей;
- произведено сравнение оборудования разных производителей и выбран оптимальный вариант;
- приведены методы повышения производительности и отказоустойчивости серверов;
- описана оптимизация рабочих параметров ОС Windows Server 2003;
- описана оптимизация рабочих параметров СУБД SQL Server;
- выполнено распределение предметных баз по узлам сети;
- произведено аналитическое и имитационное моделирование функционирования ЛВС с последующим сравнением полученных результатов.

11 Литература

1) Методические указания к куросовой работе по дисциплине «Эксплуатация АСОИиУ»;
2) Постников В.М., Лекции по курсу «Эксплуатация АСОИиУ», 2014;
3) Галкин В.А., Григорьев Ю.А., Телекоммуникация и сети;
4) Черненький В.М., Лекции по курсу «Иммитационное моделирование», 2013.