ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

УЛЬЯНОВСКИЙ ИНСТИТУТ ГРАЖДАНСКОЙ АВИАЦИИ ИМЕНИ ГЛАВНОГО МАРШАЛА АВИАЦИИ Б.П. БУГАЕВА

КАФЕДРА ИНФОРМАТИКИ

КУРСОВАЯ РАБОТА

по дисциплине «Сети электронно-вычислительных машин и средства коммуникаций»

на тему: Сеть ЭВМ информационно – справочной системы аэропорта

Выполнил:

студент группы УК5-17-1 Юмангулова Л.Р.

Проверил:

руководитель курсовой работы Брежнев В.Г.

Работа защищена _____ с оценкой_____
(дата)

_____
(подпись)

Ульяновск 2018

График

выполнения курсовой работы по дисциплине
«Сети ЭВМ и средства коммуникаций»

№ п/п	Наименование работ	Дата представления
1	Получение, изучение темы курсовой работы.	*16.02.18 г.*
2	Ознакомление с рекомендуемой литературой, составление плана выполнения курсовой работы.	*до 28.02.18 г.*
3	Проведение анализа технологий, применяемых для построения современных кабельных ЛВС.	*до 10.03.18 г.*
4	Разработка модели функционирования ЛВС на основе современных технологий и анализ их технических характеристик.	*до 29.03.18 г.*
5	Разработка алгоритма клиент- серверного приложения.	*до 10.04.18 г*
6	Разработка клиент-серверного приложения.	*до 25.04.18 г.*
7	Оформление пояснительной записки и представление руководителю курсовой работы.	*до 7.05.18 г*
8	*Защита курсовой работы.*	*до 20.05.18 г.*

Студент Юмангулова Л..Р.

УЛЬЯНОВСКИЙ ИНСТИТУТ ГРАЖДАНСКОЙ АВИАЦИИ ИМЕНИ ГЛАВНОГО МАРШАЛА АВИАЦИИ Б.П. БУГАЕВА

КАФЕДРА ИНФОРМАТИКИ

КУРСОВАЯ РАБОТА

Задание № 7

Студент Юмангулова Л.Р. **группы** УК5-17-1

Тема «Сеть ЭВМ информационно – справочной системы аэропорта.»

Данные информационно-справочной службы аэропорта (ИСС) хранятся в виде текстовых файлов на серверах. В аэропорту имеется 12 рабочих мест, с которых данные поступают в ИСС аэропорта, для ее постоянного обновления. Средний объем данных для обновления 2300 Кбайт. Обновление должно осуществляться не более чем за 0,59 мин.

Провести анализ и дать рекомендации по построению ЛВС рабочих мест ИСС. Разработать клиент-серверное приложение, позволяющее передавать данные отчета на сервер с использованием средств протокола UDP.

Задание получил: «16» февраль 2018г
(фамилия и инициалы, подпись)

Задание выдал : «16» февраль 2018 г
(фамилия и инициалы, подпись)

Содержание

Введение

Ethernet является сегодня доминирующей технологией канального уровня в локальных сетях , где она практически вытеснила все остальные технологии. Технология Ethernet прошла длинный путь развития с тех пор, как была изобретена в 1973 году сотрудником компании XeroxPARC*Робертом Меткалфом* (Robert Metcalfe). Это была экспериментальная сеть на основе коаксиального кабеля, которая передавала данные со скоростью 3 Мбит/с с использованием протокола множественного доступа с контролем несущей и обнаружением конфликтов CSMA/CD (Carrier Sense)MultipleAccessCollisionDetect — CSMA/CD). Разработка предназначалась для локальных сетей со случайным, но иногда весьма интенсивным объемом передаваемых данных. Успех проекта сразу привлек к нему внимание, и в 1980 году консорциум трех компаний — Digital Equipment Corporation, Intel Corporation и Xerox Corporation — разработал спецификацию Ethernet 1.0 для передачи данных со скоростью 10 Мбит/с. Первый стандарт IEEE 802.3 был основан на спецификации Ethernet 1.0 и был очень похож на нее. Проект стандарта был одобрен рабочей группой по стандарту 802.3 в 1983 году, а в 1985 году опубликован как официальный стандарт (ANSI/IEEE Std. 802.3-1985). С тех пор был принят ряд дополнений к стандарту, отражающих новые достижения в технологиях и позволяющих поддерживать дополнительные сетевые среды и более высокие скорости передачи, а также поддерживающих несколько новых дополнительных возможностей управления доступом к сети.

Технология **Fast Ethernet** во многом совпадает с традиционной технологией Ethernet, но быстрее ее в 10 раз. Fast Ethernet или 100BASE-T работает со скоростью 100 Мбит/с вместо 10 Мбит/с для традиционного варианта Ethernet. Технология 100BASE-T использует кадры того же формата и длины, как Ethernet и не требует изменения протоколов высших уровней, приложений или сетевых ОС на рабочих станциях. Существует возможность

маршрутизировать и коммутировать пакеты между сетями 10 Мбит/с и 100 Мбит/с без трансляции протоколов и связанных с ней задержек. Технология Fast Ethernet использует протокол CSMA/CD подуровня MAC для обеспечения доступа к среде передачи. Большинство современных сетей Ethernet построены на основе топологии "звезда", где концентратор является центром сети, а кабели от концентратора тянутся к каждому компьютеру. Такая же топология используется в сетях Fast Ethernet, хотя диаметр сети несколько меньше по причине более высокой скорости. Fast Ethernet использует неэкранированный кабель из скрученных пар проводников (UTP), как указано в спецификации IEEE 802.3u для 100BASE-T. В 100BASE-TX одна пара используется для передачи данных, вторая - для обнаружения коллизий и приема.

В марте 1996 года, примерно через год после принятия стандарта 802.3u (Fast Ethernet), положившего начало масштабируемое Ethernet, принимается решение о разработке стандарта 802.3 (**Gigabit Ethernet**). В мае того же года 11 ведущих компаний, наиболее заинтересованных в разработке и внедрении этой технологии, основали Альянс Gigabit Ethernet. Внегопервоначальновошли 3Com Corp, Bay Networks, Inc., Cisco Systems, Inc., Compaq Computer Corp. Granite Systems,Inc., Intel Corporation, LSI Logic, Packet Engines, Inc., Sun Microsystems Computer Company, LJB Networks и VLSI Technology. В июне 1998г. принимается стандарт IEEE 802.3z, использующий одномодовые и многомодовые оптоволоконные кабели, а также STP категории 5 на короткие расстояния (до 25 м). Столь малое допустимое расстояние в случае применения UTP обуславливало сомнительную возможность практического применения такого варианта. Положение изменилось с принятием в июне 1999 г. стандарта IEEE 802.3ab для передачи 1000 Мбит/с по неэкранированной витой паре на расстояния до 100 м.

1. Анализ технологий построения современных локальных вычислительных сетей (ЛВС)

Ethernet - самая популярная в настоящее время сетевая архитектура. Фирменный сетевой стандарт Ethernet был разработан на основе коаксиального кабеля. Эта последняя версия фирменного стандарта послужила основой стандарта IEEE 802.3. Стандарт IEEE 802.3 имеет модификации, которые различаются типом используемой физической среды:

Таблица 1. Спецификации физической среды Ethernet

	l0Base-5	*l0Base-2*	*l0Base-T*	*l0Base-F*
Максимальная длина сегмента	500 м	185 м	100 м	2000 м
Макс. количество сегментов	5	5		
Макс. количество пользователей	100	30	1024	1024
Максимальное число повторителей	4	4	4	4
Макс. протяженность	2500 м	925 м	500 м	2500
Кабель	«толстый» коаксиал	"тонкий" коаксиал	UTP-3	оптика
Топология	Шина	шина	звезда, дерево	звезда

- *10Base-T* - Конечные узлы соединяются по топологии «точка-точка» с многопортовым повторителем с помощью двух витых пар. Преимущество l0Base-T: концентратор контролирует работу узлов и изолирует от сети некорректно работающие узлы.

- *l0Base-F* – «+» высокая помехоустойчивость,«–» сложность прокладки оптики.

10 - скорость передачи данных, Base - метод передачи на одной базовой частоте 10 МГц, последний символ - тип кабеля. Локальные сети, построенные по этому стандарту, обеспечи вают пропускную способность до 10 Мбит/с. Используемая топология - общая шина, "звезда" и смешанные структуры.

В стандарте 802.3, включая Fast Ethernet и Gigabit Ethernet, в качестве метода доступа к среде передачи данных используется метод коллективного доступа с опознаванием несущей и обнаружением коллизий (carrier-sense-miltiply-access with collision detection, CSMA/CD), метод CSMA/CD.

Этот метод используется в сетях, где все компьютеры имеют непосредственный доступ к общей шине и могут немедленно получить данные, которые посылаются любым компьютером. Простота этого метода позволила ему получить широкое распространение.

Данные передаются кадрами. Каждый кадр снабжается преамбулой (8 байт), которая позволяет синхронизировать работу приемника и передатчика. В заголовках кадра указывается адрес узла-получателя, который позволяет узлу-получателю распознать, что предаваемый кадр предназначен ему, и адрес узла-отправителя для отправки сообщения, подтверждающего факт получения кадра. Минимальная длина кадра - 64 байта, максимальная - 1518 байт. Минимальная длина кадра является одним из параметров, определяющих диаметр сети или максимальную длину сегмента сети. Чем меньше кадр, тем меньше диаметр сети.

Передача кадра возможна, когда никакой другой узел сети не передает свой кадр. Стандарт Ethernet не позволяет одновременную передачу/прием более одного кадра. На практике в сетях Ethernet возможны ситуации, когда два узла пытаются передать свои кадры. В таких случаях происходит искажение передаваемых данных, потому что методы стандарта Ethernet не позволяет

выделять сигналы одного узла из общего сигнала и возникает так называемая коллизия. Передающий узел, обнаруживший коллизию, прекращает передачу кадра, делает паузу случайной длины и повторяет попытку захвата передающей среды и передачи кадра. После 16 попыток передачи кадра кадр отбрасывается.

При увеличении количества коллизий, когда передающая среда заполняется повторными кадрами, реальная пропускная способность сети резко уменьшается. В этом случае необходимо уменьшить трафик сети любыми доступными методами (уменьшение количества узлов сети, использование приложений с меньшими затратами сетевых ресурсов, реструктуризация сети).

Технология Fast Ethernet

Развитие локальных сетей, появление новых более быстрых компьютеров привело к необходимости совершенствования стандарта Ethernet с целью увеличения пропускной способности сети до 100 Мбит/с.

Технология Fast Ethernet использует метод доступа CSMA/CD, такой же, как в технологии Ethernet, что обеспечивает согласованность технологий. Отличия Fast Ethernet от Ethernet наблюдаются только на физическом уровне. На канальном уровне изменений нет.

Таблица 2. Спецификации физической среды Fast Ethernet

	100Base-TX	100Base-T4	100Base-FX
Максимальная длина сегмента	100 м	100 м	412 м (полудуплекс) 2000 м (полный дуплекс)
Кодирование	-	8B/6T	4B/5B
Кабель	2 пары	4 пары	многомодовая оптика

	UTP-5, STP-1	UTP-3, 4, 5	
Топология	звезда, дерево	звезда, дерево	звезда, дерево

- 8B/6T - каждые 8 бит информации уровня MAC кодируются 6-ю троичными цифрами (3 состояния), группа из 6-ти троичных цифр передается на одну из 3 передающих витых пар, независимо и последовательно, 4 пара используется для прослушивания несущей частоты в целях обнаружения коллизии;
- 4B/5B: каждые 4 бита данных подуровня MAC представляются 5 битами.

Диаметр сети сократился до 200 метров, что связано с увеличением скорости передачи данных в 10 раз. Стандарты TX и FX могут работать как в полудуплексном режиме (передача ведется в двух направлениях, но попеременно во времени), так и в полнодуплексном режиме (передача ведется одновременно в двух направлениях) за счет использования двух витых пар или двух оптических волокон. Для отделения кадра Ethernet от символов Idle в спецификациях 100Base-FX/TX используется комбинация символов Start Delimiter (пара символов J (11000) и K (10001) кода 4B/5B, а после завершения кадра перед первым символом Idle вставляется символ T).

Спецификация Fast Ethernet включает также механизм автосогласования, позволяющий порту узла автоматически настраиваться на скорость передачи данных — 10 или 100 Мбит/с. Этот механизм основан на обмене рядом пакетов с портом концентратора.

Технология Gigabit Ethernet

Стандарт IEEE 802.3z Gigabit Ethernet был принят в 1998 году на основе согласованных усилий группы компаний, образовавших объединение Gigabit Ethernet Alliance. В качестве варианта физического уровня был принят

физический уровень технологии Fiber Channel. Разработчики стандарта максимально сохранили преемственность предыдущих стандартов Ethernet: сохраняются все форматы кадров, полудуплексная и полнодуплексная версии протоколов, поддерживаются коаксиальный кабель, витая пара категории 5, волоконно-оптический кабель.

Поддержка полудуплексного режима метода доступа CSMA/CD сокращает диаметр сети до 25 м. Для увеличения диаметра сети до 200 м разработчики изменили размер минимального кадра с 64 до 512 байт. Для сокращения накладных расходов по передаче длинных кадров стандарт разрешает передавать несколько кадров подряд, не дополняя их до 512 байт и не передавая доступ к среде другому узлу. Не поддерживает:

- качество обслуживания;
- избыточные связи;
- тестирование работоспособности узлов и оборудования.

Метод доступа CSMA/CD.

Таблица 3. Спецификациифизической среды Gigabit Ethernet

	1000BASE-T	1000BASE-TX	1000BASE-SX	1000BASE-LX	1000BASE-LX
Максимальная длина сегмента	100 м	100 м	550 м	5000 м	550 м
Кодирование	PAM5		PAM5		PAM5
Кабель	UTP-5e	STP-6	многомодовая оптика	одномодовая оптика	Многомодовая оптика

Топология	звезда, дерево	звезда, дерево	звезда, дерево	звезда, дерево	звезда, дерево

Многомодовый кабель – применяются излучатели, работающие на двух длинах волн: 1300 и 850 нм. Светодиоды с λ=850 нм – дешевле, чем с λ=1300 нм. Длина кабеля уменьшается – затухание на волне 850 м более чем в два раза выше, чем на волне 1300 нм.

Одномодовый кабель – применяются излучатели, работающие на длине волны: 1300.

Увеличение минимального размера кадра с 64 до 512 байт. Разрешается также передавать несколько кадров подряд, не освобождая среду.

Таблица 4. Сравнительный анализ технологий

№ п/п	Ethernet	Fast Ethernet	Gigabit Ethernet
Тип кабеля	10Base - 5 коаксиальный	100 Base TX Витая пара категории 5 UTP	1000 BASE-SX многомодовое оптоволокно
Тип адаптера	Ether Power 10/100	3CSOHO100-TX	SMC9452TX
Среда передачи	Коаксильный кабель	витая пара или оптоволоконный кабель	витая пара или оптоволоконный кабель
Способдоступа к каналу	CSMA/CD	CSMA/CD	CSMA/CD

Скорость передачи данных в канале связи	10	100	1000
Длина сегмент (max)	500м	100 м	550м
Основная топология	«шина»	«звезда»	«звезда»
Длина кадра/поля данных (байт)	46-1500	46-1500	46-1500
Стандарты	IEEE 802.3	IEEE 802.3u	IEEE 802.3z

2. Разработка модели и моделирование функционирования локальной вычислительной сети

Для разработки модели используются основные положения теории массового обслуживании. ЛВС представляется в виде системы массового обслуживания (СМО) (рис. 1):

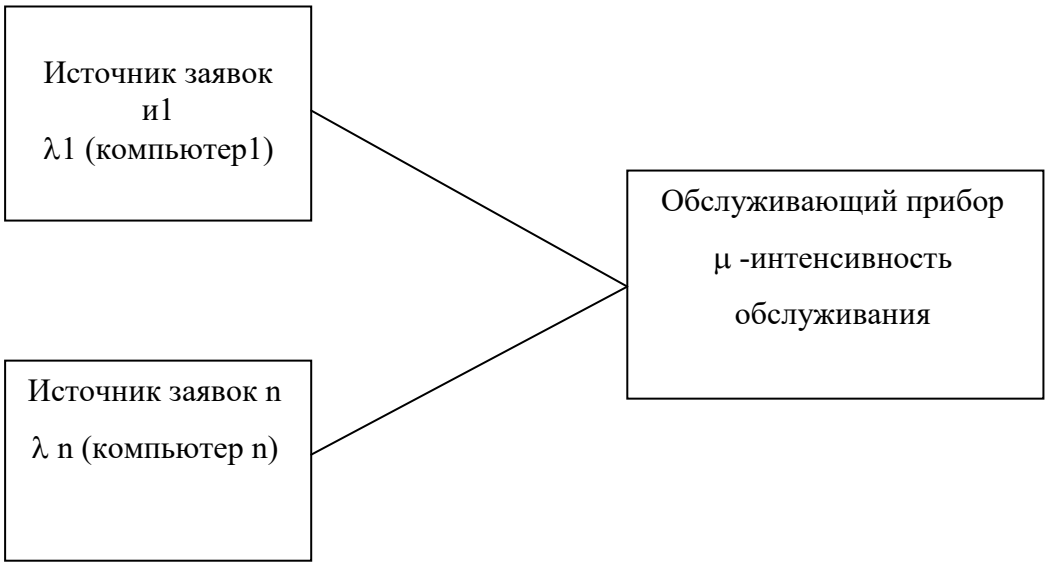


Рис. 1 Представление ЛВС в виде СМО

От источников заявок (ПК) заявки поступают в сеть в случайные моменты времени с интенсивностями λn. Поступившие заявки обрабатываются в обслуживающем приборе с интенсивностью μ. Обслуживающий прибор – канал передачи данных ЛВС. Функция распределения интервалов времени между поступлениями заявок и функция распределения времени обработки заявки в обслуживающем приборе соответствуют показательному закону распределения:

$F\left(t_{\text{поступления}}\right) = 1 - e^{-\lambda\tau}$- функция распределения интервалов времени между поступлениями заявок.

$F\left(t_{\text{поступления}}\right) = 1 - e^{-\mu t}$ - функция распределения времени

обработки заявки в обслуживающем приборе, где λ – интенсивность поступления заявок –расчет по формуле 1.

$$\lambda = \frac{V_{\text{опф}}}{L_{\text{дк}} * T_{\text{пф}}}, \qquad (1)$$

где Vопф – объем передаваемого файла, Lдк – длина кадра, Tпф – требуемое время передачи файла;

- интенсивность обработки заявок в канале ЛВС:

$$\mu = \frac{1}{L_{\text{дк}} * b}; \quad b = \frac{1}{U_{\text{нс}}}, \quad (2)$$

где Uнс – номинальная скорость передачи данных в канале; b – скорость передачи одного бита данных в канале;

- эффективная скорость в канале ЛВС:

$$C_{\text{Э}} = \frac{L_{\text{дпд}}}{L_{\text{дк}} * b}, \quad (3)$$

где Lдпд - длина поля данных пакета;

- реальная скорость передачи данных в моделируемом канале ЛВС:

$$C_{\text{Р}} = \frac{\sum_{i=1}^{m} L_{\text{дпд}i}}{T_{\text{м}}}, \quad (4)$$

где Lдпдi – длина поля данных кадра, переданного в канале модели ЛВС;

Tм – время моделирования работы сети, заданное в модели;

m - число переданных кадров в канале – определяется в результате моделирования работы сети в среде AnyLogic;

- коэффициент использования канала передачи данных ЛВС:

$$К_{исп} = \frac{С_р}{С_э}, \quad (5)$$

Данные формулы нужны для того чтобы произвести расчеты, которые будут внесены в Таблицу 5, где:

λ – интенсивность поступления заявок – расчет по формуле (1);

μ - интенсивность обработки заявок в приборе - расчет по формуле (2);

Сэ – эффективная скорость передачи данных в канале ЛВС - расчет по формуле (3).

Таблица 5. Расчет основных параметров модели

U – номинальная скорость (Мбит/с)	L- длина поля данных кадра (байт)	Сэ- эффек-тивная скорость	b	λ	μ	Ср- реальная скорость в канале	Кисп- Коэф-фициент исполь-зования канала
10	46	6,388889	0,1	924,043	0,01736	1,538462	0,240803
	100	7,936508	0,1	528,024	0,00992	4,166667	0,525
	500	9,505703	0,1	126,485	0,00238	3,846154	0,404615
	1000	9,746589	0,1	64,8451	0,00122	6,666667	0,684
	1497	9,829284	0,1	43,6842	0,00082	7,5	0,763026
100	46	63,88889	0,01	924,043	0,17361	1,111111	0,017391
	100	79,36508	0,01	528,024	0,09921	4,166667	0,0525
	500	95,05703	0,01	126,485	0,02376	3,846154	0,040462
	1000	97,46589	0,01	64,8451	0,01218	4,705882	0,048282
	1497	98,29284	0,01	43,6842	0,00821	6,923077	0,070433
1000	46	638,8889	0,001	924,043	1,73611	1,538462	0,002408
	100	793,6508	0,001	528,024	0,99206	4,166667	0,00525
	500	950,5703	0,001	126,485	0,23764	3,333333	0,003507
	1000	974,6589	0,001	64,8451	0,12183	4,210526	0,00432

	1497	982,9284	0,001	43,6842	0,08208	6,923077	0,007043

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	U – номинальная скорость (Мбит/с)	L- длина поля данных кадра (байт)	Сэ - эффективная скорость	b	λ	μ	Ср-реальная скорость в канале	Кисп-Коэф-фициент исполь-зования канала	U – номинальная скорость (бит/с)	время передачи (мин)	время передачи (сек)	L- кадра (бит)	L- длина поля данных кадра (бит)	время моделир ования	V (Кбайт)	V (бит)	sink
2	10	46	6,388888889	0,1	924,043	0,017361	1,538461538	0,24080268	10000000	0,59	35,4	576	368	1,3	2300	18841600	2
3	10	100	7,936507937	0,1	528,024	0,009921	4,166666667	0,525	10000000	0,59	35,4	1008	800	1,2	2300	18841600	5
4	10	500	9,505703422	0,1	126,485	0,002376	3,846153846	0,40461538	10000000	0,59	35,4	4208	4000	1,3	2300	18841600	5
5	10	1000	9,746588694	0,1	64,8451	0,001218	6,666666667	0,684	10000000	0,59	35,4	8208	8000	1,2	2300	18841600	8
6	10	1497	9,829284307	0,1	43,6842	0,000821	7,5	0,76302605	10000000	0,59	35,4	12184	11976	1,2	2300	18841600	9
7	100	46	63,88888889	0,01	924,043	0,173611	1,111111111	0,0173913	100000000	0,59	35,4	576	368	1,8	2300	18841600	2
8	100	100	79,36507937	0,01	528,024	0,099206	4,166666667	0,0525	100000000	0,59	35,4	1008	800	1,2	2300	18841600	5
9	100	500	95,05703422	0,01	126,485	0,023764	3,846153846	0,04046154	100000000	0,59	35,4	4208	4000	1,3	2300	18841600	5
10	100	1000	97,46588694	0,01	64,8451	0,012183	4,705882353	0,04828235	100000000	0,59	35,4	8208	8000	1,7	2300	18841600	8
11	100	1497	98,29284307	0,01	43,6842	0,008207	6,923076923	0,07043317	100000000	0,59	35,4	12184	11976	1,3	2300	18841600	9
12	1000	46	638,8888889	0,001	924,043	1,736111	1,538461538	0,00240803	1000000000	0,59	35,4	576	368	1,3	2300	18841600	2
13	1000	100	793,6507937	0,001	528,024	0,992063	4,166666667	0,00525	1000000000	0,59	35,4	1008	800	1,2	2300	18841600	5
14	1000	500	950,5703422	0,001	126,485	0,237643	3,333333333	0,00350667	1000000000	0,59	35,4	4208	4000	1,5	2300	18841600	5
15	1000	1000	974,6588694	0,001	64,8451	0,121832	4,210526316	0,00432	1000000000	0,59	35,4	8208	8000	1,9	2300	18841600	8
16	1000	1497	982,9284307	0,001	43,6842	0,082075	6,923076923	0,00704332	1000000000	0,59	35,4	12184	11976	1,3	2300	18841600	9

Для нахождения Ср (реальная скорость передачи данных в моделируемом канале ЛВС) и Кисп (коэффициент использования канала передачи данных ЛВС) построим модель в программной среде имитационного моделирования AnyLogic.

2.1.Построение модели в программной среде имитационного моделирования AnyLogic

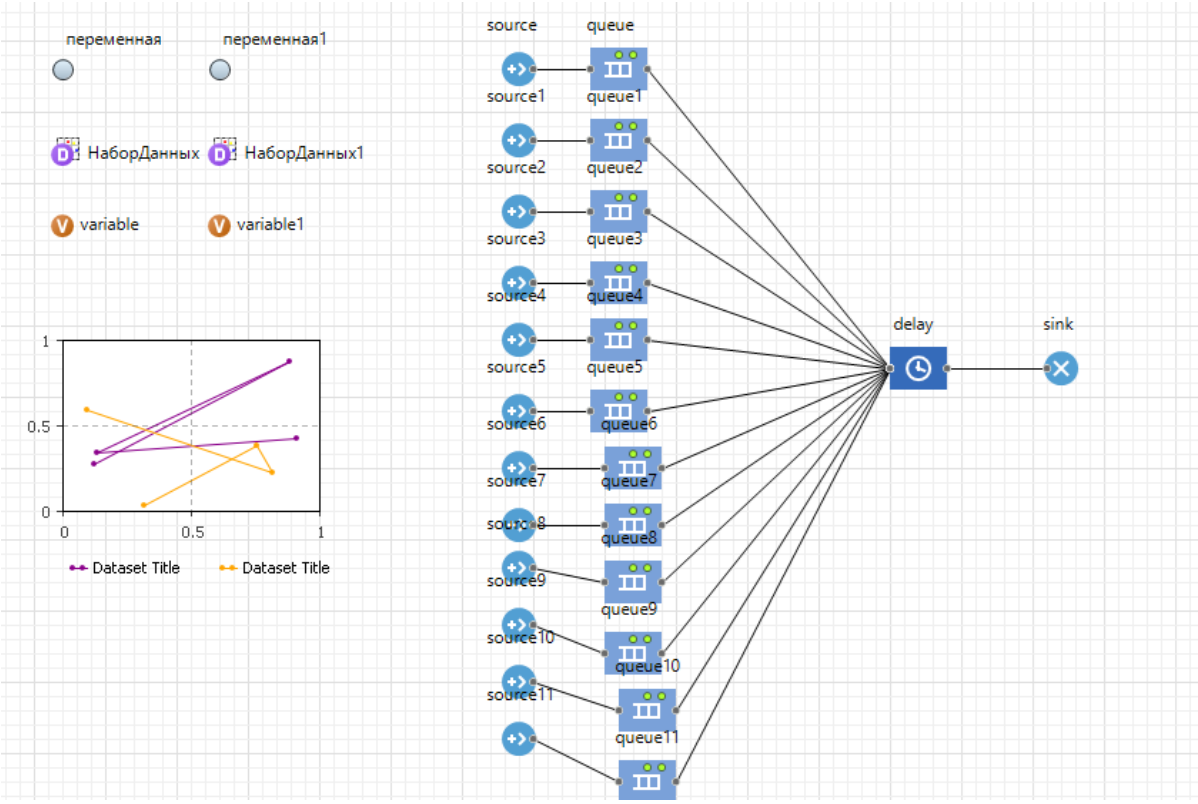


Рис. 2 Соединение элементов модели

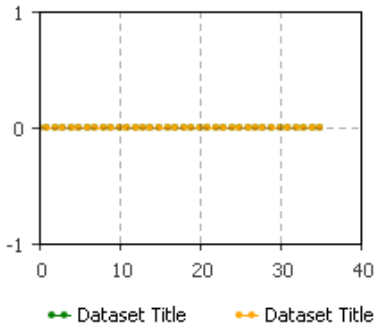


Рис. 3 Технологии Ethernet с длиной кадра 1000 байт.

2.2.Технико – экономическое обоснование выполнения работы

Таблица 6. Расчет стоимости развертывания ЛВС

Вид технологии ЛВС	Кабель			Разъёмы			Коммутационное оборудование или Хаб (для технологии Ethernet)			Сетевая карта			Общая стоимость, руб.
	Тип	Длина, м	Цена, руб.	Тип	Количество	Стоимость, руб.	Тип	Количество	Стоимость, руб.	Тип	Количество	Стоимость, руб.	
Ethernet	LAN UTP 3	24	1017,6	RJ-45	12	90	EFAH 05W	1	2935	Trendnet TEGPCITX R	6	3180	7222,6
Fast Ethernet	Витая пара UTP 5e	24	148,8	RJ-45	12	90	Trendnet NWay Switch TE100-S55E 5- port	1	1050	DGE530T	6	3888	5176,8
Gigabit Ethernet	Оптический 12 волокон 50/125 мкм	24	2380	RJ-45	12	90	TrendnetSG1005D 5-port	1	1350	Trendnet Ectx	6	23016	26836

В результате моделирования развертывать ЛВС (локально вычислительную сеть) рекомендуется по технологии Fast Ethernet, так как здесь достаточно высокий коэффициент использования канала,а также приемлемая стоимость аппаратных средств. Сеть Fast Ethernet получается более экономически выгодной, чем, к примеру, сеть Ethernet. Поэтому такая сеть

будет удовлетворять как техническим, так и экономическим требованиям задания. Таким образом, данная технология Fast Ethernet может использоваться для передачи данных.

3.Разработка клиент-серверного приложения

3.1. Разработка алгоритмов

3.1.1. Разработка и описание алгоритма клиентской части – схема связи классов

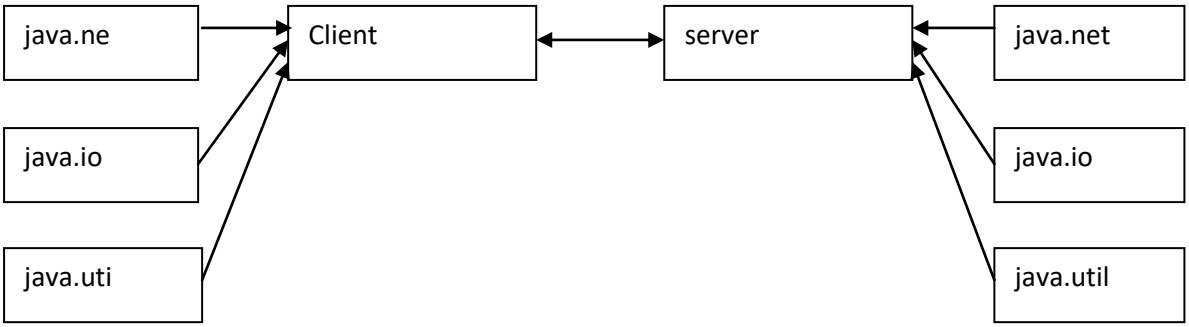


Рис. 6 Схема взаимосвязи классов

Сервер – это программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу клиента, предоставляя ему доступ к определённым ресурсам или услугам. В том числе, это специализированное аппаратное устройство, которое выполняет те или иные задачи, как удалённо (что бывает чаще всего), так и локально на месте. Сервер, как и компьютер, состоит из: процессора, материнской платы, оперативной памяти и жёсткого диска. Чаще всего для крупных серверов используют специально предназначенные для этого комплектующие, но, тем не менее, есть сервера из тех комплектующих, которые используют и простые пользователи.

Понятия *сервер* и *клиент* и закреплённые за ними роли образуют программную концепцию «*клиент-сервер*».

Для взаимодействия с клиентом (или клиентами, если поддерживается одновременная работа с несколькими клиентами) сервер выделяет необходимые ресурсы межпроцессного взаимодействия (разделяемая память, пайп, сокет и т. п.) и ожидает запросы на открытие соединения (или, собственно, запросы на предоставляемый сервис). В зависимости от типа такого ресурса, сервер может обслуживать процессы в пределах одной компьютерной системы или процессы на других машинах через каналы передачи данных (например, COM-порт) или сетевые соединения.

Формат запросов клиента и ответов сервера определяется протоколом. Спецификации открытых протоколов описываются открытыми стандартами, например, протоколы Интернета определяются в документах RFC.

В зависимости от выполняемых задач одни серверы, при отсутствии запросов на обслуживание, могут простаивать в ожидании. Другие могут выполнять какую-то работу (например, работу по сбору информации), у таких серверов работа с клиентами может быть второстепенной задачей.

«Клиент — сервер» (англ. *client–server*) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине.

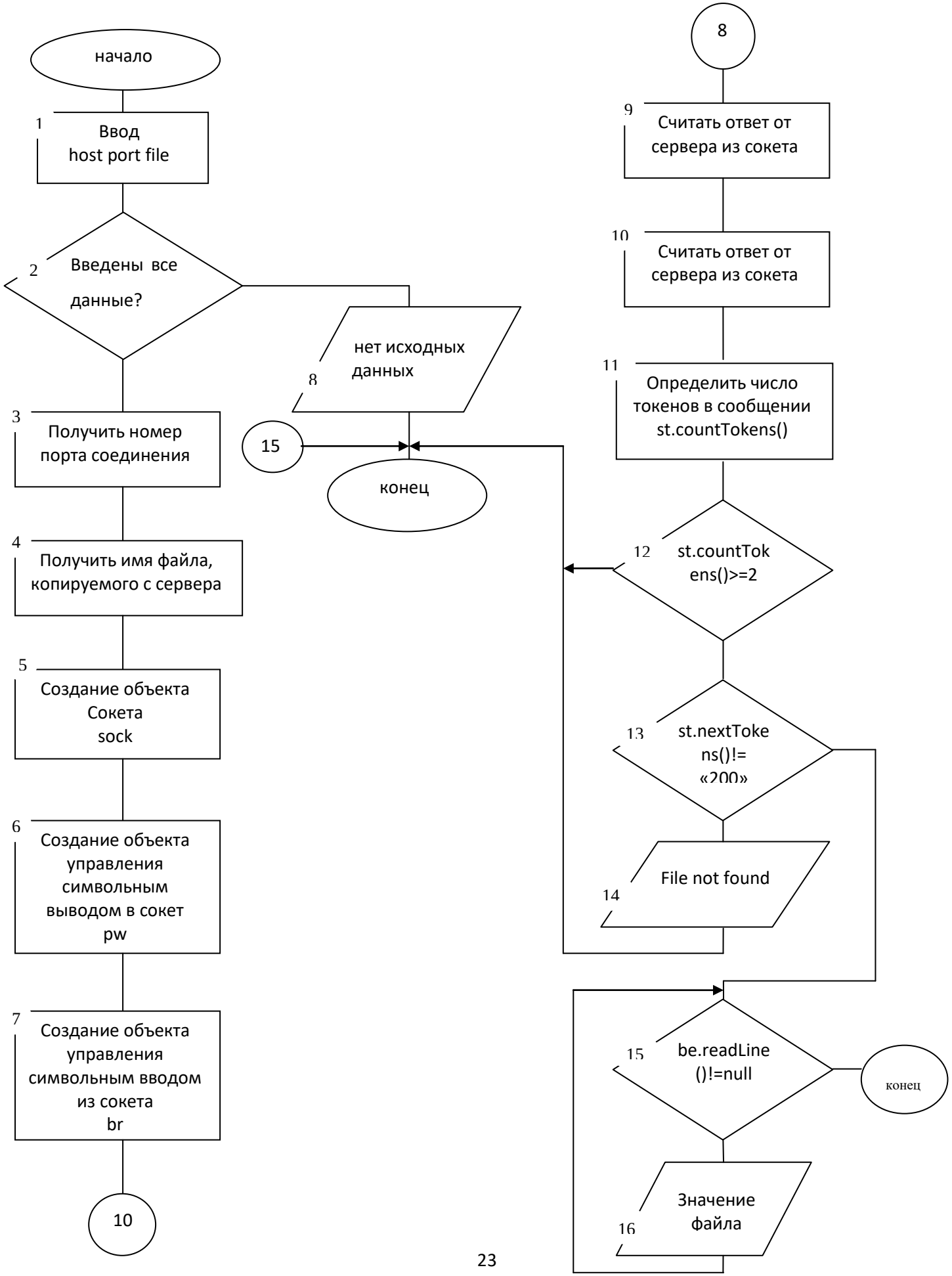

начало
1
Ввод
host port file
2
Введены все данные?
нет исходных данных
8
15
конец
3
Получить номер порта соединения
4
Получить имя файла, копируемого с сервера
5
Создание объекта Сокета
sock
6
Создание объекта управления символьным выводом в сокет
pw
7
Создание объекта управления символьным вводом из сокета
br
10
8
9
Считать ответ от сервера из сокета
10
Считать ответ от сервера из сокета
11
Определить число токенов в сообщении
st.countTokens()
12
st.countTok ens()>=2
13
st.nextToke ns()!= «200»
File not found
14
15
be.readLine ()!=null
конец
Значение файла
16

Рис. 7 Алгоритм функционирования приложения клиента

Описание алгоритма.

Начало алгоритма.

Шаг1.Выполняется действие«Ввод host port file».

Шаг 2.Проверяется логическое условие «Введены все данные?». Если условие выполняется,то идет переход к*Шаг 3*,если нет,то к *Шаг 8*.

Шаг 3. Выполняется действие «Получить номер порта соединения».

Шаг 4.Выполняется действие «Получить имя файла, копируемого с сервера».

Шаге 5.Выполняется действие «Создание объекта Сокета sock».

Шаг 6.Выполняется действие «Создание объекта управления символьным выводом в сокет Pw».

Шаг 7.Выполняется действие «Создание объекта управления символьным вводом из сокета br».

Шаг 8.Этот шаг ответвляется от этапа 2, осуществляется вывод «Нет исходных данных». Конец алгоритма.

Шаг 9.Выполняется действие «Считать ответ от сервера из сокета».

Шаг 10.Выполняется действие «Считать ответ от сервера из сокета».

Шаг 11.Выполняетсядействие «Определить число токенов в сообщении st.countTokens ()».

Шаг 12. Выполняется проверка логического условия «countTokens()>=2». Левая ветвь - конец алгоритма.

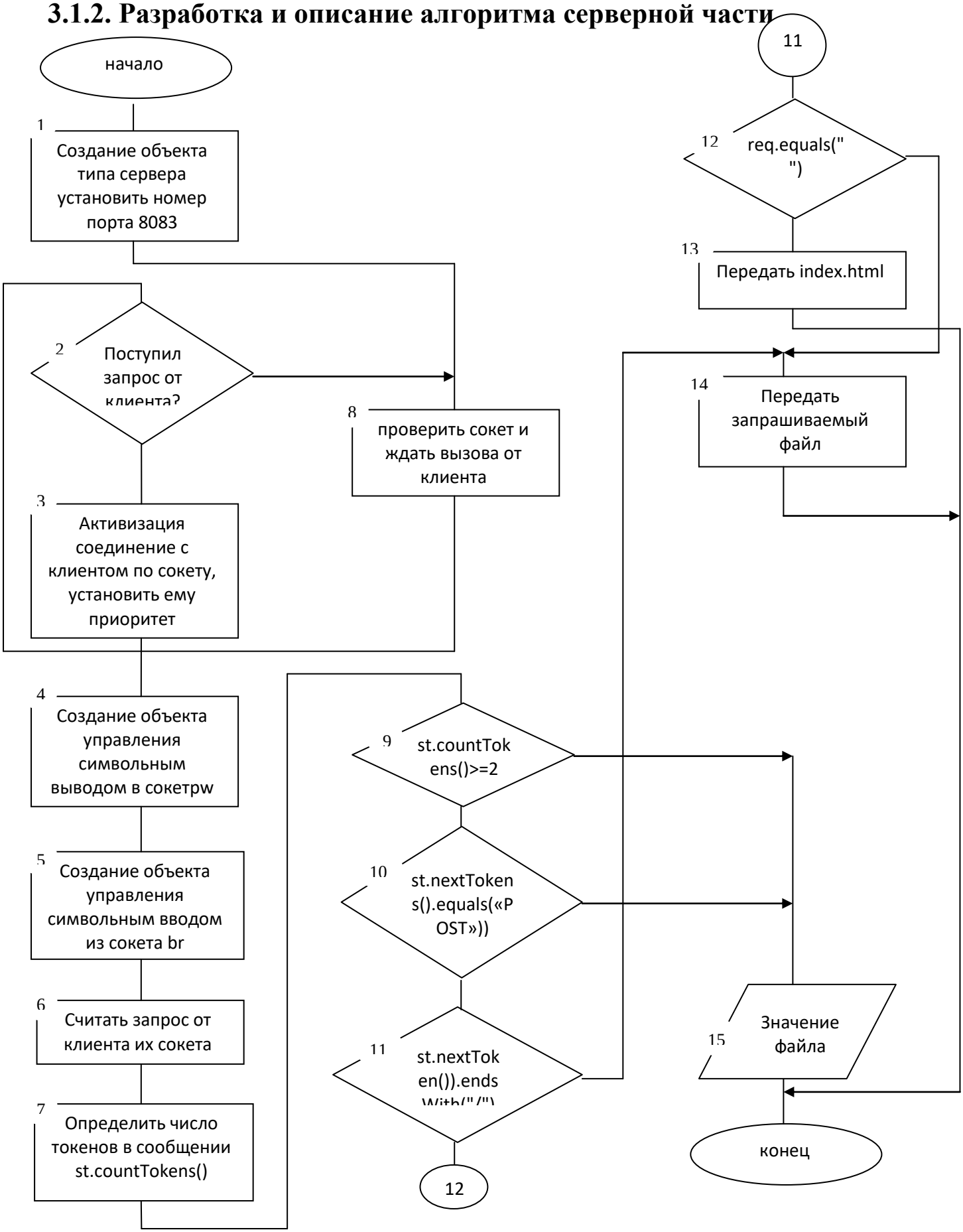
Шаг 13. Происходит проверка логического условия «st.nextTokens()!= “200”».

Шаг 14. Осуществляется вывод результатов «File not found».

Шаг 15.Выполняется проверка логического условия «br.readLine()!= null». Правая ветвь - конец алгоритма.

Шаг 16. Осуществляется вывод результатов «Значение файла».

3.1.2. Разработка и описание алгоритма серверной части



Начало алгоритма.

Шаг 1.Выполняется действие «Создание объекта типа сервера установить номер порта 8083».

Шаг 2.Проверяется логическое условие «Поступил запрос от клиента?».

Шаг 3.Выполняется действие «Активизация соединение с клиентом по сокету, установить ему приоритет».

Шаг4.Выполняется действие «Создание объекта управления символьным выводом в сокет pw».

Шаг5.Выполняется действие «Создание объекта управления символьным вводом из сокета br».

Шаг6.Выполняется действие «Считать запрос от клиента из сокета».

Шаг7.Выполняется действие «Определить число токенов в сообщении st.countTokens()».

Шаг8.Выполняется действие «проверить сокет и ждать вызова от клиента».***Шаг9.***Проверяется логическое условие «st.countTokens()>=2».

Шаг10.Проверяется логическое условие «st.nextToken().equals("POST")».

Шаг11.Проверяется логическое условие «st.nextToken().endsWith("/")».

Шаг12. Происходит проверка логического условия «req.equals("")».

Шаг13.Выполняется действие «Передать index.html».

Шаг14.Выполняется действие «Передать запрашиваемый файл».

Шаг15. Осуществляется вывод результатов «Значение файла».

Конец алгоритма.

4. Разработка программ

4.1. Разработка серверной части программы

Посылка дейтаграмм по протоколу UDP

package server_program; // объявление пакета, к которому привязываются разрабатываемые классы

import java.net.*; //подключения методов класса java.net и его дочерних классов

import java.io.*; // потоки и файлы произвольного доступа. Аналог библиотеки стандартного ввода-вывода

import java.util.*; // классы-контейнеры (Dictionary, HashTable, Stack) и некоторые другие утилиты. Кодирование и декодирование. Классы Date и Time.

public class Main { // Программная структура, содержит в себе основные элементы такие как методы и поля

public static void main(String[] args) // главный метод, начальные значения, передаются аргументы, описывает функцию

{

try // содержит один или более операторов, выдает исключения

{

ServerSocket ss = new ServerSocket(Integer.parseInt(args[0])); // объект класса предназначен для установки канала связи с клиентским приложением

while (true) //логическое выражение, оператор цикла

new HttpConnect(ss.accept()); // установка канала связи с клиентским приложением

}

catch(ArrayIndexOutOfBoundsException ae) // Перехват исключений в Java оформляется блоком "try-catch», сначала делается попытка выполнить фрагмент кода, и если генерируется исключение, то оно обрабатывается фрагментом catch

{

System.err.println("Usage: Server port"); // метод вывода системных сообщений

System.exit(0); // "нормальное" завершение процесса

}

catch(IOException e) // содержит операторы, которые специфицируют действия при вызове исключения в блоке try

{

System.out.println(e); // стандартный поток вывода

}

}

}

class HttpConnect extends Thread // представляет собой отдельный поток управления в пределах процесса

{

private Socket sock; // чтобы сгладить различия в реализациях разных серверов, между сервером и портом

HttpConnect(Socket s) // устанавливает соединение между локальной машиной и указанным портом узла Internet, имя которого было передано конструктору

{

sock = s; // присвоение значения s переменной sock

setPriority(NORM_PRIORITY - 1); // приоритет устанавливается на два уровня выше Thread

run(); // метод в котором задается последовательность действий, выполняемых в рамках потока

}

public void run() // может объявлять переменные, вызывать другие методы и использовать другие классы.

{

try // содержит один или более операторов, оператор вызывающий исключения

{

PrintWriter pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()), true); // для форматного вывода данных различных типов с целью их визуального представления в виде текстовой строки

BufferedReader br = new BufferedReader(new InputStreamReader(sock.getInputStream())); // классы которые организуют входные потоки, буферизированный ввод данных

String req = br.readLine(); // класс создающий объект готовый разбить строки на слова

System.out.println("Request: " + req); // стандартный поток вывода

StringTokenizer st = new StringTokenizer(req); // класс предназначен для выделения отдельных элементов из строк типа String

if ((st.countTokens() >= 2) && st.nextToken().equals("POST"))// оператор условия, операция возвращает в виде строки следующее слово

{

if ((req = st.nextToken()).ends With("/")|| req.equals(""))// оператор условия, операция возвращает в виде строки следующее слово

req += "index.html"; // Этот оператор используется для выполнения определённых операторов, если логическое условие true

try // содержит один или более операторов

{

File f = new File(req); // создание нового объекта с аргументом req

BufferedReader bfr = new BufferedReader(new FileReader(f)); // классы которые организуют входные потоки, буферизированный ввод данных

char[] data = new char[(int)f.length()]; // 16-ти разрядная переменная в виде символов, символы кодируются с помощью юникода

bfr.read(data); // буферизированный ввод данных

pw.println("HTTP/1.1 200 OK "); // выполнен принудительный переход на следующую строку

pw.write(data); // программа выполняет серию операций по выводу в поток данных различного типа

pw.flush();// программа выполняет серию операций по выводу в поток данных различного типа

}

catch(FileNotFoundException fe) // содержит операторы, которые специфицируют действия при вызове исключения в блоке try

{

pw.println("HTTP/1.1 404 Not FoundXn");

}

catch(IOException ioe) // содержит операторы, которые специфицируют действия при вызове исключения в блоке try

{

System.out.println(ioe); // стандартный поток вывода

}

}

else pw.println("HTTP/l.l 400 Bad RequestW"); // необязательный блок else для выполнения других операторов, если условие false

sock.close(); // Основной метод этого класса accept () ожидает поступления запроса. Когда запрос получен, метод устанавливает соединение с клиентом и возвращает объект класса socket, через который сервер будет обмениваться информацией с клиентом.

}

catch(IOException e) // содержит операторы, которые специфицируют действия при вызове исключения в блоке try

{

System.out.println(e); // стандартный поток вывода

}

} }

4.2 Разработка клиентской части программы

package javaapplication33; // объявление пакета, к которому привязываются разрабатываемые классы

import java.net.*; //подключения методов класса java.net и его дочерних классов

import java.io.*; // потоки и файлы произвольного доступа. Аналог библиотеки стандартного ввода-вывода

import java.util.*; // классы-контейнеры (Dictionary, HashTable, Stack) и некоторые другие утилиты. Кодирование и декодирование. Классы Date и Time.

import java.lang.Integer.*; // каждый модуль компиляции содержит неявное импортирование этого пакета

public class Main { // Программная структура, содержит в себе основные элементы такие как методы и поля

public static void main(String[] args) // главный метод, начальные значения, передаются аргументы, описывает функцию

{

String[] hostportfile={"127.0.0.1", "8080", "c:/Minori "}; // класс создающий объект готовый разбить строки на слова

if (hostportfile.length != 3) // оператор условия, операция возвращает в виде строки следующее слово

{

System.err.println("Usage: Clienthostportfile"); // методвыводасистемныхсообщений

System.exit(0); // "нормальное" завершение процесса

}

String host = hostportfile[0]; // класс создающий объект готовый разбить строки на слова

int port = Integer.parseInt(hostportfile[1]); // четырехбайтные целые числа

String file = hostportfile[2]; // класс создающий объект готовый разбить строки на слова

try // содержит один или более операторов, оператор вызывающий исключения

{

Socket sock = new Socket(host, port); // Основной метод этого класса accept () ожидает поступления запроса.

PrintWriter pw = new PrintWriter(new OutputStreamWriter(sock.getOutputStream()), true); // для форматного вывода данных различных типов с целью их визуального представления в виде текстовой строки

pw.println("POST " + file + " HTTP/1.l "); // выполнен принудительный переход на следующую строку

BufferedReader br = new BufferedReader(new InputStreamReader(sock.getInputStream()));// классы которые организуют входные потоки, буферизированный ввод данных

String line = null; // класс создающий объект готовый разбить строки на слова

line = br.readLine(); // присвоение значения переменной

StringTokenizer st = new StringTokenizer(line); // класс предназначен для выделения отдельных элементов из строк типа String

String code = null; // класс создающий объект готовый разбить строки на слова

if ((st.countTokens() >= 2) && st.nextToken().equals("POST")) // оператор условия, операция возвращает в виде строки следующее слово

{

if ((code = st.nextToken()) != "200") // оператор условия, операция возвращает в виде строки следующее слово

{

System.err.println("File not found, code = " + code); // метод вывода системных сообщений

System.exit (0); // "нормальное" завершение процесса

}

}

while ((line = br.readLine()) != null) //логическое выражение, оператор цикла

System.out.println(line); // стандартный поток вывода

sock.close();// Основной метод этого класса accept () ожидает поступления запроса. Когда запрос получен, метод устанавливает соединение с клиентом и возвращает объект класса socket, через который сервер будет обмениваться информацией с клиентом.

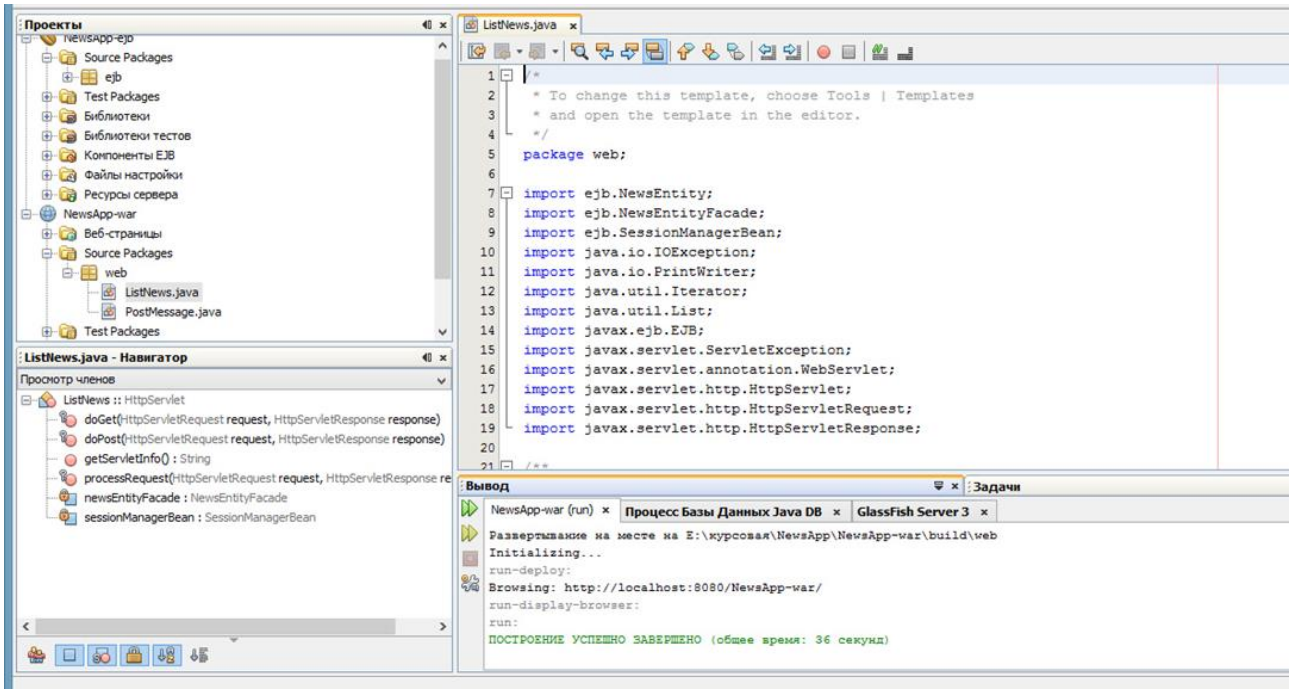
}catch(Exception e) // содержит операторы, которые специфицируют действия при вызове исключения в блоке try

{

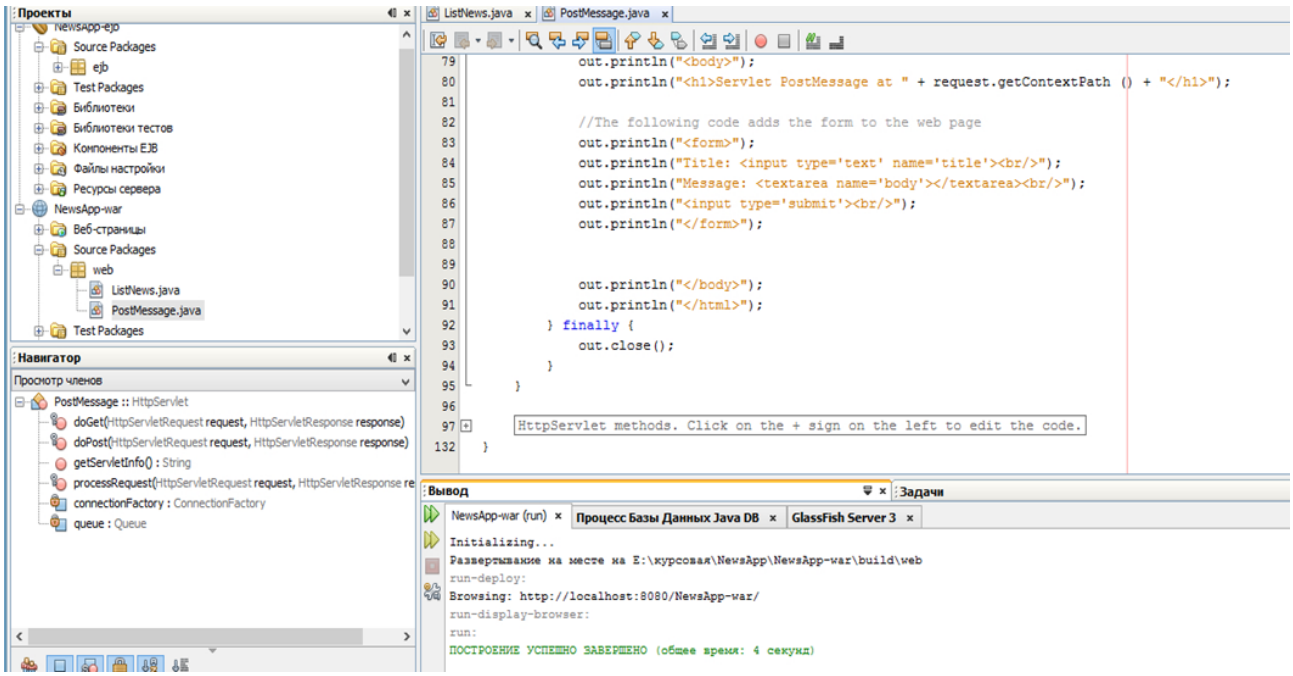
System.err.println(e); // метод вывода системных сообщений

}}}

4.3 Тесты. Результат тестирования



Клиентская часть программы



Серверная часть программы

Заключение

В данной курсовой работе изучены основные принципы построения компьютерных сетей, разработки сетевых распределенных приложений на основе клиент - серверной технологии. Рассчитаны первичные параметры сети, на основе которых осуществлена разработка имитационной модели функционирования локальной вычислительной сети. Построена модель в программной среде имитационного моделирования AnyLogic. Изучена и определена стоимость аппаратных средств, что позволяет оптимально подобрать технологию построения компьютерной сети, удовлетворяющей технико–экономическим требованиям пользователей. А также, разработаны алгоритмы и программы клиент-серверного приложения.

Список литературы

1. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. -4-е изд. –СПб.: Питер, 2015. – 944 с.
2. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов/ В.Г.Олифер, Н.А.Олифер. - 3-е изд. -СПб.: Питер, 2008. – 958 с.
3. Олифер, В.Г. Сетевые операционные системы: учебник для вузов / В.Г. Олифер,Н.А. Олифер 2-е изд. - СПб.: Питер, 2009. – 669 с.
4. Сети электронно-вычислительных машин и средства коммуникаций : метод.указания по выполнению курсовой работы / сост. В.Г.Брежнев. – Ульяновск : УВАУ ГА(И), 2013. -25 с.
5. Сети электронно-вычислительных машин и средства коммуникаций: методические указания по выполнению лабораторной работы «Создание приложения J2EE с помощью EJB 3.1 в редакторе Java IDE NetBeans» / сост. Е.В. Беляева, В.Г. Брежнев, В.В. Савин. – Ульяновск, УВАУ ГА (И), 2015. – 49 с.
6. Локальная вычислительная сеть ЭВМ под управлением операционной системы Windows 7 : учеб.пособие / сост. А.Н. Подъяченков, В.Г. Брежнев. – Ульяновск : УИ ГА, 2016. – 64 с.