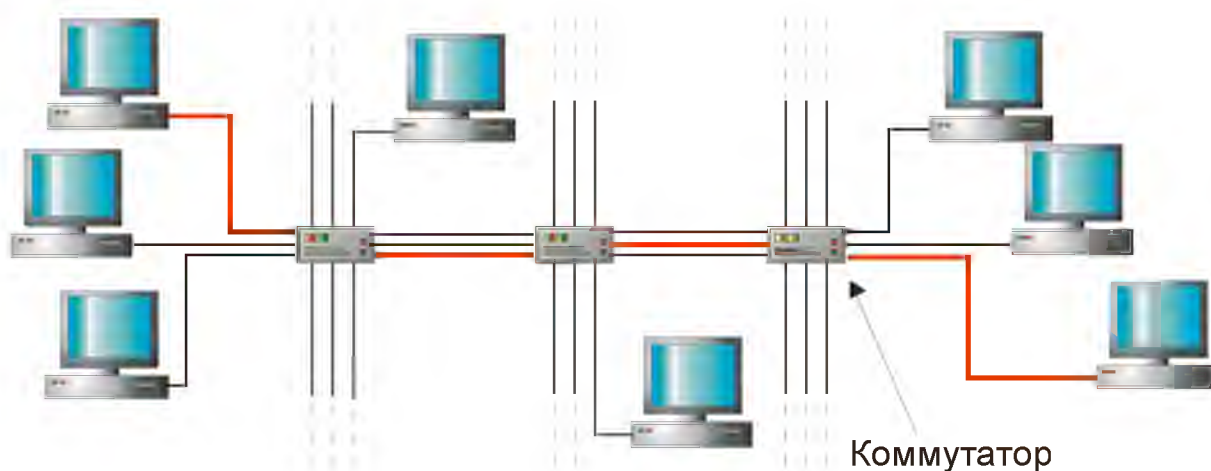


В.И. ЛОЙКО

Вычислительные системы, сети и телекоммуникации

Электронный учебник



В.И. ЛОЙКО

Вычислительные системы, сети и телекоммуникации

Электронный учебник

Рекомендован
методической комиссией
факультета прикладной информатики
Кубанского государственного
аграрного университета
для студентов специальности
«Прикладная информатика»

**Краснодар
2011**

УДК 681.5
Л - 46

Лойко В.И.

Вычислительные системы, сети и телекоммуникации: Электронный учебник. - Краснодар: КубГАУ, 2011. - 293 с.: ил.

Электронный учебник предназначен для студентов специальности «Прикладная информатика» по дисциплине «Вычислительные системы, сети и телекоммуникации» и для расширения и углубления знаний в области информационных систем и технологий студентов других экономических специальностей вузов. В нем освещены теоретические и практические вопросы современных нетрадиционных архитектур вычислительных систем, позволяющих эффективнее и производительнее реализовывать информационные процессы обработки и накопления данных. В главе, посвященной компьютерным сетям, основной акцент сделан на объяснение сути процесса обмена данными, подкрепленный теоретическим и справочным материалом.

Для лучшего усвоения материала студентами учебник содержит гипертекстовые ссылки на видеоматериалы, ответы на вопросы для самопроверки и т.п.

Рецензент:

Кафедра вычислительной техники и автоматизированных систем управления Кубанского государственного технологического университета (зав. кафедрой, д-р техн. наук, профессор, академик МА ПРЭ В.И. Ключко).

© КубГАУ, 2011

Оглавление

ПРЕДИСЛОВИЕ	8
ВВЕДЕНИЕ	10
ГЛАВА 1. КОМПЬЮТЕРНЫЕ СИСТЕМЫ, УПРАВЛЕНИЕ И ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ.....	13
1.1. ПОНЯТИЕ СИСТЕМ	13
1.2. УПРАВЛЕНИЕ В СИСТЕМАХ.....	14
1.3. БАЗОВАЯ ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ	19
1.4. КЛАССИФИКАЦИЯ КОМПЬЮТЕРОВ ПО ОБЛАСТЯМ ПРИМЕНЕНИЯ.....	24
1.5. ОБЩИЕ ТРЕБОВАНИЯ, ПРЕДЪЯВЛЯЕМЫЕ К СОВРЕМЕННЫМ КОМПЬЮТЕРАМ.....	42
1.6. ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ	48
ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ	62
ГЛАВА 2. НЕТРАДИЦИОННЫЕ АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ	63
2.1. ЧИСЛОВАЯ И НЕЧИСЛОВАЯ ОБРАБОТКА	64
2.2. ОГРАНИЧЕНИЯ ФОННЕЙМАНОВСКОЙ АРХИТЕКТУРЫ	68
2.2. ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА.....	70
2.3. КОНВЕЙЕРНАЯ ОБРАБОТКА.....	73
2.3.1. Последовательные конвейеры.....	74
2.3.2. Векторные конвейеры	77
2.4. КЛАССИФИКАЦИЯ АРХИТЕКТУР ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ	79
2.5. МУЛЬТИПРОЦЕССОРНЫЕ СИСТЕМЫ.....	84
2.5.1. Классификация систем параллельной обработки данных	84
2.5.2. Мультимикропроцессорные системы с общей памятью	90
2.5.2. Мультимикропроцессорные системы с локальной памятью и многомашинные системы.....	95

2.6.БАЗОВЫЕ АРХИТЕКТУРЫ СУПЕРКОМПЬЮТЕРОВ	99
2.6.1. Система Illiac 4.....	99
2.6.2. MPP - процессор фирмы Goodyear	101
2.6.3. Векторные конвейерные процессоры	102
2.7. АССОЦИАТИВНЫЙ ПРОЦЕССОР.....	112
2.8. КОНЦЕПЦИЯ ВС С УПРАВЛЕНИЕМ ПОТОКОМ ДАННЫХ	119
2.9. ЗАКОН АМДАЛА И ЕГО СЛЕДСТВИЯ	121
2.10. НАИБОЛЕЕ ИЗВЕСТНЫЕ СОВРЕМЕННЫЕ МНОГОПРОЦЕССОРНЫЕ КОМПЬЮТЕРЫ	122
ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ.....	138

ГЛАВА 3. ВВЕДЕНИЕ В ТЕОРИЮ МАССОВОГО ОБСЛУЖИВАНИЯ И УПРАВЛЕНИЯ РЕСУРСАМИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ 140

3.1. ПОНЯТИЕ МАРКОВСКОГО СЛУЧАЙНОГО ПРОЦЕССА	140
3.2. ПОТОКИ СОБЫТИЙ.....	141
3.3. УРАВНЕНИЯ КОЛМОГОРОВА.....	143
3.4. БАЗОВЫЕ СООТНОШЕНИЯ СИСТЕМ МАССОВОГО ОБСЛУЖИВАНИЯ.....	147
3.4.1. Схема гибели и размножения.....	147
3.4.2. Формула Литтла	148
3.5. ПРОСТЕЙШИЕ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ И ИХ ХАРАКТЕРИСТИКИ.....	151
3.5.1. Задача Эрланга.....	151
3.5.2. Одноканальная СМО с неограниченной очередью	155
3.5.3. Многоканальная СМО с неограниченной очередью	158
3.6. УПРАВЛЕНИЕ РЕСУРСАМИ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ	160
3.6.1. Управление ресурсами однопроцессорных систем оперативной обработки данных.....	161
3.6.1.1. Алгоритм SPT	161
3.6.1.2. Алгоритм RR	162
3.6.1.3. Алгоритм FB.....	167

3.6.2. <i>Планирование вычислительного процесса</i>	171
3.6.2.1. Методы управления ресурсами многопроцессорных систем при обработке пакетов задач с прерываниями	171
3.6.2.2. Управление ресурсами многопроцессорных систем при обработке пакетов независимых задач без прерываний	174
3.6.3. <i>Производительность мультипроцессорных систем с общей и индивидуальной памятью (режимы разделения нагрузки и разделения функций)</i>	175
3.6.3.1. МПС с общей памятью	176
3.6.3.1. Характеристики МПС с общей памятью	177
3.6.3.2. МПС с индивидуальной памятью	182
3.6.3.2. Характеристики МПС с индивидуальной памятью	183
ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ	185
ГЛАВА 4. КОМПЬЮТЕРНЫЕ СЕТИ.....	187
4.1. БАЗОВЫЕ ТОПОЛОГИИ ЛОКАЛЬНЫХ КОМПЬЮТЕРНЫХ СЕТЕЙ	189
4.2. ТОПОЛОГИЯ ГЛОБАЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ	193
4.3. СЕТЕВЫЕ ПРОТОКОЛЫ И УРОВНИ	195
4.4. ФИЗИЧЕСКИЙ И КАНАЛЬНЫЙ УРОВНИ	198
4.4.1. Модуляция и демодуляция.....	199
4.4.2. Емкость канала связи	202
4.4.3. Кодирование информации	206
4.4.4. Уплотнение информационных потоков	210
4.5. ПРОТОКОЛЫ КАНАЛЬНОГО УРОВНЯ.....	214
4.5.1. Схема организации фаз коммуникаций	215
4.5.2. Виды протоколов	218
4.5.3. Анализ производительности протоколов	220
4.5.4. Определение скорости передачи полезной информации и оптимальной длины кадра	222
4.5.5. Методы случайного доступа к сети	223
4.5.5.1. Методы Алоха	224
4.5.5.2. Случайный доступ типа МДПН/ОС (CSMA/CD) ..	229
4.5.6. Спецификации ETHERNET	236

4.5.6.1. Стандарты IEEE на 10 Мбит/с	238
4.6. СЕТЕВОЙ УРОВЕНЬ МОДЕЛИ OSI	245
4.6.1. Методы коммутации в компьютерных сетях	246
4.6.1.1. Сети с коммутацией каналов	246
4.6.1.2. Сети с коммутацией сообщений	248
4.6.1.3. Сеть с пакетной коммутацией	250
4.6.2. Управление потоком в сети	252
4.6.2.1. Метод скользящего окна	253
4.6.3. Выбор кратчайших путей	258
4.6.3.1. Алгоритм Дейкстры	258
4.6.3.2. Алгоритм Флойда	261
4.7. ГЛОБАЛЬНАЯ СЕТЬ INTERNET	263
4.7.1. Появление и развитие Internet	263
4.7.2. Структура Internet	264
4.7.3. Передача информации в Internet	267
4.7.4. Краткая характеристика ресурсов Internet	271
4.7.5. Удаленный доступ к ресурсам сети	277
4.7.6. Коммерческое применение Internet	278
ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ	290
ЛИТЕРАТУРА	292

Предисловие

Широчайшее распространение персональных компьютеров может создать у студентов впечатление, что других компьютерных систем не существует или, по крайней мере, они отживают свой век. Это, конечно же, не так. Интеллектуальные запросы человечества растут быстрее роста производительности и возможностей компьютерной техники, что стимулирует исследования и разработки новых вычислительных архитектур, новых подходов к организации вычислений в ЭВМ, использующих нетрадиционные методы и их реализацию.

Электронный учебник «Архитектура компьютерных систем и сетей» предназначен для студентов специальностей «Прикладная информатика». Тематика Учебника соответствует Стандартам указанной специальности в части вычислительных систем и сетей.

Учебник состоит из четырех глав, посвященных изложению основных теоретических и прикладных вопросов компьютерных систем и сетей.

В первой главе излагаются основные понятия систем управления и место в них информационных технологий. В свою очередь на основе архитектур ЭВМ и компьютерных сетей реализуются информационные процессы обработки данных и обмена данными в информационных технологиях. Этот важный момент подчеркивается в первой главе. Здесь же дана классификация компьютеров по областям применения и описаны основные параметры компьютеров.

Во второй главе изложены современные классификация и способы оценки компьютерных систем, а также рассматриваются мультипроцессорные системы. Изложены принципы параллельной и конвейерной обработки данных, приведены современные нетрадиционные архитектуры ЭВМ, включая ассоциативные и потоковые процессоры.

В анализе их производительности используется аппарат теории массового обслуживания. Поэтому авторы третью главу начали с введения в теорию массового обслуживания, где дана необходимая база для последующих теоретических исследований. Основная же часть третьей главы посвящена обоснованию методов управления производительностью мультипроцессорных систем; излагаются также основы теории управления ресурсами вычислительных систем.

В четвертой (самой большой по объему) главе рассматриваются теоретические и прикладные вопросы компьютерных сетей, включая глобальную сеть Internet.

В учебнике изложены общие подходы и принципы, на которых основано построение современных компьютерных систем и сетей. Для овладения практическими навыками работы в рамках конкретных вопросов архитектурного и алгоритмического содержания курса студенты проходят лабораторный практикум и выполняют курсовой проект.

Для лучшего усвоения материала студентами электронный учебник содержит гипертекстовые ссылки на видеоматериалы, ответы на вопросы для самопроверки и т.п.

Введение

Вычислительные системы различной архитектуры являются аппаратной частью информационной технологии, достигшей к концу XX века глобального характера и содержания. Мультипроцессорные системы, к которым относятся также компьютерные сети, позволяют за счет изменения их архитектуры оптимизировать параметры основных информационных процессов информационной технологии: обработки, накопления, передачи данных и представления знаний.

Под технологией в широком смысле понимают науку о производстве материальных благ, включающей три аспекта: информационный, инструментальный и социальный [2]. *Информационный* аспект включает описание принципов и методов производства; *инструментальный* - орудия труда, с помощью которых реализуется производство; *социальный* - кадры и их организацию. В более узком промышленном смысле технологию рассматривают как последовательность действий над предметом труда в целях получения конечного продукта. Например, технология получения стали из железной руды или технология производства сливочного масла из молока.

Понятие “информационная технология” возникло в последние десятилетия XX века в процессе становления информатики. Особенность информационной технологии в том, что в ней и предметом, и продуктом труда является информация, а орудиями труда - средства вычислительной техники и связи. Информационная технология как наука о производстве информации возникла именно потому, что информация стала рассматриваться как вполне реальный производственный ресурс наряду с другими материальными ресурсами. Причем производство информации и ее верхнего уровня - знаний оказывает решающее влияние на модификацию и создание новых промышленных технологий.

Потребность в передаче и обмене информацией человечество испытывало уже на ранних стадиях своего развития. Если сначала для ускорения передачи информации использовались костры, курьеры, потом - почта, семафорный телеграф и пр., то с изобретением электрического телеграфа и телефона принципиально изменились возможности передачи информации. Изобретение радио и телевидения, а затем компьютера, цифровых систем связи и вычислительных сетей, создание в 1978 году первого персонального компьютера и совершенно невероятное и исключительно быстрое его распространение и развитие именно в качестве инструментального средства накопления, преобразования и передачи информации позволили новым, автоматизированным информационным технологиям внедриться практически во все области человеческой деятельности. На рисунке 1 представлена схема формирования автоматизированных информационных технологий, являющихся следствием интеграции достижений человечества в области средств связи, обработки, накопления и отображения информации [2].



Рисунок 1-Основные технические достижения человечества, обусловившие появление АИТ

Основу автоматизированных информационных технологий составляют следующие технические достижения:

1) создание средств накопления больших объемов информации на машинных носителях, таких как магнитные и оптические диски;

2) создание различных средств связи, таких как радио - и телевизионная связь, телекс, телефакс, цифровые системы связи, компьютерные сети, космическая связь, позволяющих воспринимать, использовать и передавать информацию практически в любой точке земного шара;

3) создание компьютера, и особенно персонального компьютера, позволяющего по определенным алгоритмам обрабатывать и отображать информацию, накапливать и генерировать знания.

Наиболее полно достоинства информационной технологии проявляются при ее использовании в человеко-машинных системах управления (АСУ).

Глава 1. Компьютерные системы, управление и информационная технология

1.1. Понятие систем

По-гречески система (*systema*) - это целое, составленное из частей. Другими словами *система* - есть совокупность элементов, взаимосвязанных друг с другом и таким образом образующих определённую целостность.

Количество элементов, из которых состоит система, может быть любым, важно, чтобы они были между собой взаимосвязаны. Примеры систем: техническое устройство, состоящее из узлов и деталей; живой организм, состоящий из клеток; коллектив людей; предприятие; государство и т.д. Лекционная аудитория с лектором и студентами - система; каждый студент - тоже система; оборудование аудитории - система; отдельный стол - тоже система. А вот ножка стола - уже не система. Но это с точки зрения макропредставлений. Если же рассматривать ножку стола с точки зрения микропредставлений, то это тоже система, образуемая совокупностью молекул и атомов.

Из этих примеров ясно, что системы очень разнообразны, но все они имеют ряд общих свойств и понятий.

Элемент системы - часть системы выполняющая определённую функцию (лектор читает лекцию, студенты её слушают и конспектируют и т.д.). Элемент системы может быть сложным, состоящим из взаимосвязанных частей, то есть то-

же представлять собой систему. Такой сложный элемент часто называют *подсистемой*.

Организация системы - внутренняя упорядоченность и согласованность взаимодействия элементов системы. Организация системы проявляется, например, в ограничении разнообразия состояний элементов в рамках системы (во время лекции не играют в волейбол).

Структура системы - совокупность внутренних устойчивых связей между элементами системы, определяющая её основные свойства. Например, в иерархической структуре отдельные элементы образуют соподчиненные уровни и внутренние связи образованы между этими уровнями.

Целостность системы - принципиальная несводимость свойств системы к сумме свойств её элементов. В то же время свойства каждого элемента зависят от его места и функции в системе. Так, если вернуться к примеру с лекцией, то рассматривая отдельно свойства лектора, студентов, предметов оборудования аудитории и т.д., нельзя однозначно определить свойства системы, где эти элементы будут совместно использоваться.

1.2. Управление в системах

Совокупность объекта управления (ОУ), управляющего органа (УО) и исполнительного органа (ИО) образует систему управления, в которой выделяются две подсистемы: управляющая подсистема (УО и ИО) и управляемая подсистема (ОУ) [2]. На рисунке 1.1 представлена укрупнённая структурная схема системы управления, на которой выделены входящие в неё подсистемы.

В процессе функционирования этой системы управляющий орган (УО) получает информацию $I_{ос}$ о текущем состоянии объекта управления (ОУ) и информацию $I_{вх}$ о том, в каком состоянии *должен* находиться объект управления. Отклонения объекта управления от заданного состояния

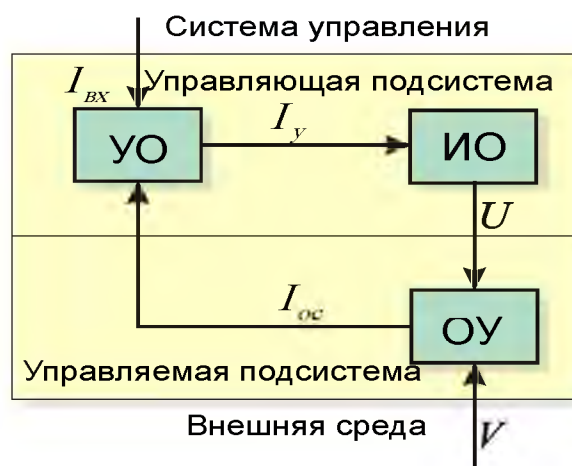


Рисунок 1.1-Укрупнённая структурная схема системы управления

происходят под воздействием внешних возмущений (v). Результатом сравнения информации I_{vx} и I_{oc} в управляющем органе является возникновение управляющей информации I_y , которая воздействует на исполнительный орган (ИО). На основе информации I_y исполнительный орган вырабатывает управляющее воздействие (U), которое ликвидирует отклонение в объекте управления.

Наиболее сложным звеном в системе управления считается управляющий орган. Здесь степень сложности определяется количеством выполняемых функций, то есть управляющий орган должен уметь производить наибольшее разнообразие действий. Это естественно, так как на любое состояние объекта управления управляющий орган должен отреагировать соответствующим образом, своевременно обработав поступившую в него информацию и выработав управляющую информацию.

Как видно из структурной схемы управления, для её функционирования необходима *информация*. На приведённой схеме изображены три её потока: I_{vx} , I_{oc} и I_y . Информация I_{vx} сообщает управляющему органу о множестве возможных состояний объекта управления и управляющего органа, а также о том, в каком из состояний должен находиться объект управления при заданных внешних условиях. Информация

I_{oc} - это информация *обратной связи*. Понятие обратной связи является фундаментальным в теории управления. В общем случае под обратной связью понимают передачу воздействия с выхода какой-либо системы обратно на её вход. В системах управления обратная связь является информационной, и с её помощью в управляющую подсистему поступает информация о текущем состоянии управляемой подсистемы. Третий информационный поток *I_y* - это информация, возникшая в результате обработки в управляющем органе информации *I_{вх}* и *I_{oc}* и управляющая работой исполнительного органа (ИО).

Очень важной компонентой входной информации *I_{вх}* является информация о *цели управления*, ибо управление *бесмысленно*, если не направлено на достижение определённой *цели*. Если управление наилучшим образом соответствует поставленной цели, то такое управление называется *оптимальным*. Критерием оптимальности управления считается некоторая количественно измеряемая величина, отражающая цель управления. Математическая запись критерия оптимальности носит название *целевой функции*. При *оптимальном управлении* значение целевой функции достигает экстремума (максимума или минимума в зависимости от критерия оптимальности).

Ярко выраженный целевой информационный характер управления подтверждается кибернетическим его определением: *управление есть процесс целенаправленной переработки информации*.

В зависимости от того, в какой системе (простой, сложной, большой) производится управление, различают системы автоматического управления (САУ) и автоматизированные системы управления (АСУ).

Автоматическое управление осуществляется, как правило, в простых системах, в которых заранее известны описание объекта управления и алгоритм управления им. По принципу управления системы автоматического управления могут быть *разомкнутыми* и *замкнутыми*. В *разомкнутых* системах из-

меряется возмущение, отклоняющее объект от заданного состояния, и вырабатывается воздействие, компенсирующее возникшее возмущение. Такая система неспособна длительное время управлять неустойчивым объектом. В *замкнутых системах* реализуется идея обратной связи, благодаря которой информация об отклонении управляемого объекта от заданного состояния позволяет выработать воздействие, возвращающее объект в это состояние.

Благодаря тому, что поведение объекта и алгоритм управления строго заданы, системы автоматического управления могут работать автономно, без участия человека (хотя, конечно, их создание и наблюдение за их функционированием невозможно без человека). На рисунке 1.2. приведена упрощенная структурная схема замкнутой системы автоматического управления.

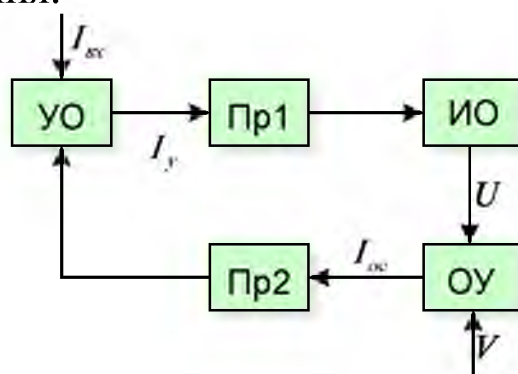


Рисунок 1.2-Упрощённая структурная схема замкнутой САУ

Как правило, САУ используются в технических системах и в качестве управляющего органа (УО) используется компьютер, который с помощью программы (для него это $I_{вх}$) выдаёт результат обработки информации, обычно физический сигнал. Это - сигнал управления (I_y), который через преобразователь (Пр1) приводит в действие исполнительный орган (ИО), возвращающий объект управления (ОУ) в заданное программой компьютера состояние. Состояние ОУ, меняющееся под воздействием внешних возмущений V , определяет значение сигнала обратной связи ($I_{ос}$), которое через преоб-

разователь (Пр2) поступает в компьютер (УО). Преобразователи необходимы для изменения уровней или природы проходящих через них сигналов, так как элементы системы могут быть различны по своей физической сути.

С ростом и усложнением производства объекты управления приобретают характер сложных и больших систем, имеющих большое число элементов и подсистем, связи между которыми не всегда ясны, а критерии функционирования не обладают достаточной чёткостью. В этих условиях использовать результаты теории автоматического управления в полной мере не удаётся, и в контур управления, помимо человека - оператора ЭВМ, действующего по заданным алгоритмам, включается *лицо, принимающее решения* (ЛПР). Наличие ЛПР в контуре управления является отличительной чертой *автоматизированных систем управления* (рисунок 1.3). Автоматизированное управление применяется в том случае, если нет возможности реализовывать автоматическое управление.

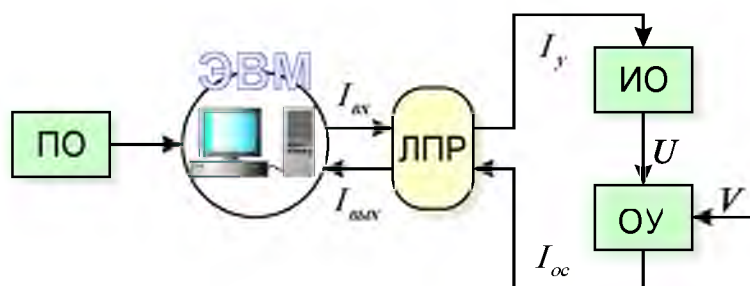


Рисунок 1.3-Структурная схема АСУ

Как видно из рисунка 1.3., ЛПР, получив информацию обратной связи I_{oc} , освещающую его о состоянии объекта управления (ОУ), обращается к ЭВМ (поток $I_{вх}$), имеющей определённое программное обеспечение (ПО) и вырабатывающей рекомендации к принятию решения (поток $I_{вых}$). На основе анализа предложенных ЭВМ альтернатив ЛПР принимает решение, которое в виде управляющей информации (I_y) поступает в исполнительный орган (ИО), переводя его в

необходимое состояние. Например, министр (это - ЛПР), получив информацию о состоянии отрасли (это - ОУ), после обработки всей нужной информации на ЭВМ и просчета наборов вариантов поведения в сложившейся ситуации, принимает решение, которое реализуется аппаратом министерства (это - ИО) в управляемой отрасли производства.

1.3.Базовая информационная технология

При производстве информационного продукта исходный информационный ресурс в соответствии с поставленной задачей подвергается в определенной последовательности различным преобразованиям. Динамика этих преобразований отображается в протекающих при этом информационных процессах. Таким образом, информационный процесс - это процесс преобразования информации. В результате него информация может изменить и содержание, и форму представления как в пространстве, так и во времени.

Фазы преобразования информации в информационной технологии достаточно многочисленны, и простое их перечисление может привести к потере ощущения целостности технологической системы (за деревьями не увидеть леса). Однако, если провести структуризацию технологии, выделив такие крупные структуры, как процессы и процедуры, то концептуальная модель базовой информационной технологии может быть представлена схемой рисунка 1.4 [2].

На этой схеме в левой части изображены блоки информационных процессов, в правой - блоки процедур. Блок в виде прямоугольника изображает процесс или процедуру, в которых преобладают ручные или традиционные операции. Овальная форма блоков соответствует автоматическим операциям, производимым с помощью технических средств (ЭВМ и средств передачи данных). В верхней части схемы информационные процессы и процедуры осуществляют преобразование информации, имеющей человеческую форму

представления, то есть ярко выраженное смысловое содержание. Синтаксический аспект информации находится здесь на втором плане. В этом случае говорят о преобразовании собственно *информации*. В нижней части схемы производится преобразование *данных*, то есть информации, представленной в машинном виде. И на этом уровне представления преобладает синтаксический аспект информации.

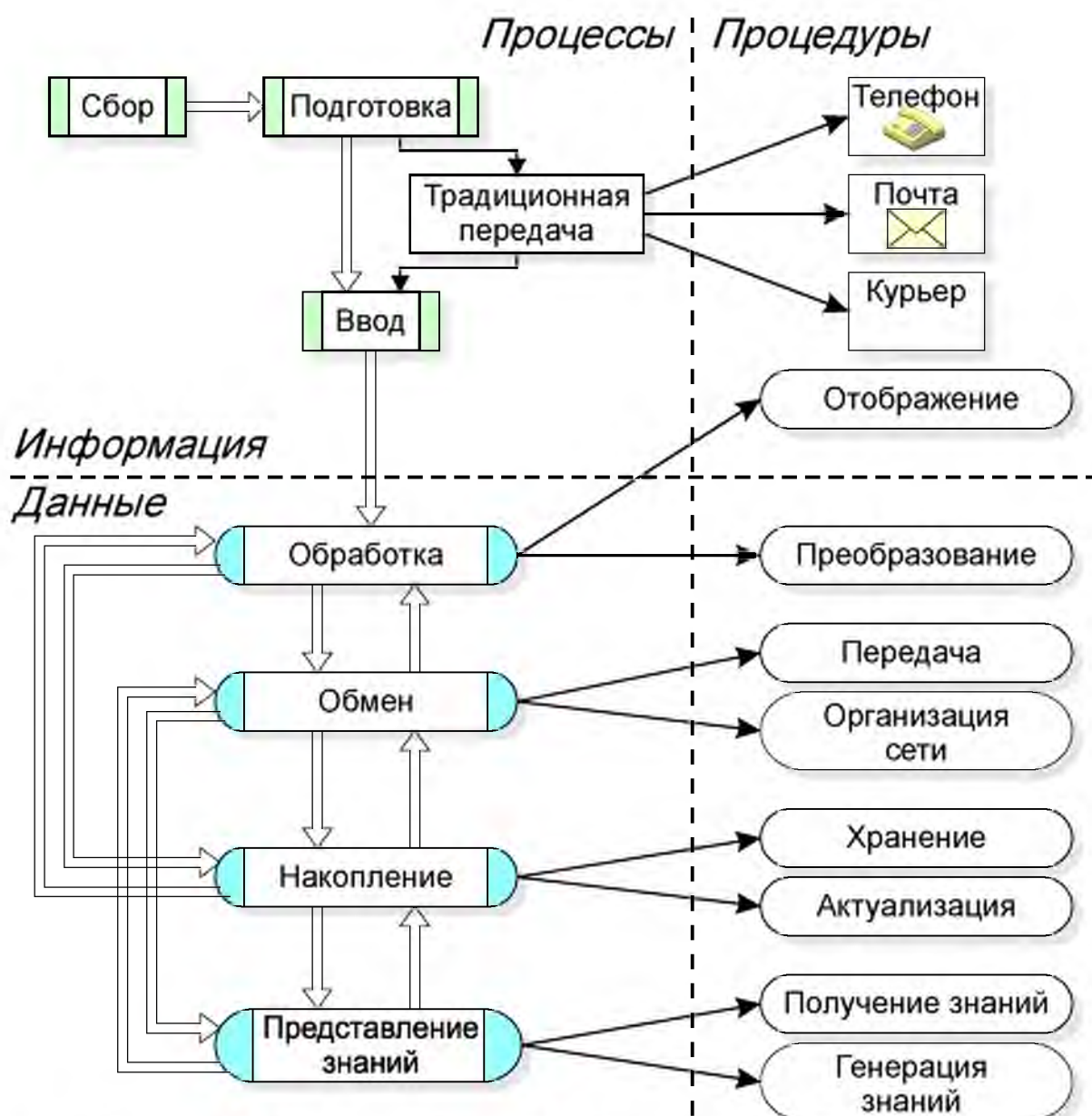


Рисунок 1.4-Концептуальная модель базовой информационной технологии

Технология переработки информации начинается с формирования информационного ресурса, который после определенных целенаправленных преобразований должен превратиться в информационный продукт. Формирование информационного ресурса (получение исходной информации) начинается с процесса *сбора* информации, которая должна в информационном плане отразить предметную область, то есть объект управления или исследования (его характеристики, параметры, состояние и т.п.). Собранная информация для ее оценки (по полноте, непротиворечивости, достоверности и т.д.) и последующих преобразований должна быть соответствующим образом *подготовлена* (осмыслена и структурирована, например, в виде таблиц). После подготовки информация может быть передана для дальнейшего преобразования традиционными способами (с помощью телефона, почты, курьера и т.п.), а может быть подвергнута сразу процессу преобразования в машинные данные, то есть процессу *ввода*. Процессы сбора, подготовки и ввода в информационной технологии организационно-экономических систем по своей реализации являются в основном ручными (кроме процесса подготовки, который частично может быть автоматизированным). Процесс ввода преобразует информацию в данные, имеющие форму цифровых кодов, реализуемых на физическом уровне с помощью различных физических представлений (электрических, магнитных, оптических, механических и т.д.).

Следующие за вводом информационные процессы уже производят *преобразование данных* в соответствии с поставленной задачей. Эти процессы протекают в ЭВМ (или организуются ЭВМ) под управлением различных программ, которые и позволяют так организовать данные, что после вывода из ЭВМ результат обработки представляет собой наполненную смыслом информацию о результате решения поставленной задачи. При преобразованиях данных можно выделить четыре основных информационных процесса и соответствую-

ющих им процедур. Это *процессы обработки, обмена, накопления данных и представление знаний*.

Процесс *обработки* данных связан с *преобразованием* значений и структур данных, а также их преобразованием в форму, удобную для человеческого восприятия, то есть *отображением*. Отображенные данные - это уже информация. Процедуры преобразования данных осуществляются по определенным алгоритмам и реализуются в ЭВМ с помощью набора машинных операций. Процедуры отображения переводят данные из цифровых кодов в изображение (текстовое или графическое) или звук.

Информационный процесс *обмена* предполагает обмен данными между процессами информационной технологии. Из схемы рисунка 1.4 видно, что процесс обмена связан взаимными потоками данных со всеми информационными процессами на уровне переработки данных. При обмене данными можно выделить два основных типа процедур. Это процедуры *передачи* данных по каналам связи и сетевые процедуры, позволяющие осуществить *организацию* вычислительной *сети*. Процедуры передачи данных реализуются с помощью операции кодирования-декодирования, модуляции-демодуляции, согласования и усиления сигналов. Процедуры организации сети включают в себя в качестве основных операции по коммутации и маршрутизации потоков данных (трафика) в вычислительной сети. Процесс обмена позволяет с одной стороны передавать данные между источником и получателем информации, а с другой - объединять информацию многих ее источников.

Процесс *накопления* позволяет так преобразовать информацию в форме данных, что удастся ее длительное время хранить, постоянно обновляя, и при необходимости оперативно извлекать в заданном объеме и по заданным признакам. Процедуры процесса накопления, таким образом, состоят в организации *хранения* и *актуализации* данных. *Хранение* предполагает создание такой структуры расположения дан-

ных в памяти ЭВМ, которая позволила бы быстро и не избыточно накапливать данные по заданным признакам и не менее быстро осуществлять их поиск. В настоящее время ЭВМ имеет два основных вида запоминающих устройств оперативные (электронные) и внешние (на магнитных и оптических дисках). Их физическая природа и устройство различны, поэтому различаются и возможности по организации структур хранения данных. Можно выделить *операции* по организации хранения и поиска данных в *оперативной* и *внешней* памяти ЭВМ. В процессе накопления данных важной процедурой является их *актуализация*. Под актуализацией понимается поддержание хранимых данных на уровне, соответствующем информационным потребностям решаемых задач в системе, где организована информационная технология. Актуализация данных осуществляется с помощью операций добавления новых данных к уже хранимым, корректировки (изменения значений или элементов структур) данных и их уничтожения, если данные устарели и уже не могут быть использованы при решении функциональных задач системы. Наконец, информационный процесс *представления знаний* включен в базовую информационную технологию как один из основных информационных процессов, поскольку высшим продуктом информационной технологии является знание. Формирование знания как высшего информационного продукта до недавнего времени было (да в основе своей является и сейчас) прерогативой человека. Однако оказать помощь человеку при решении не формализуемых или трудно формализуемых задач может автоматизированный процесс представления знаний. В этом процессе объединяются процедуры формализации знаний, их накопления в формализованном виде и формальной генерации (вывода) новых знаний на основе накопленных в соответствии с поставленной задачей. Вывод нового знания - это эквивалент решения задачи, которую не удастся представить в формальном виде. Таким образом, *процесс представления знаний* - это процесс, состоящий из процедур получения

формализованных знаний и процедур *генерации* (вывода) новых знаний из полученных. К сожалению, практическая реализация процесса представления знаний с помощью ЭВМ еще не достигла достаточно широкого применения в информационных технологиях. Это связано как с продолжающимися поисками форм представления знаний в теории искусственного интеллекта, так и практическими трудностями при создании баз знаний. Тем не менее, развитие теории искусственного интеллекта продолжается и в новом веке процесс представления знаний займет ключевое место в информационных технологиях.

В зависимости от решаемых информационной технологией задач удельный вес и взаимосвязь информационных процессов различны.

Современные, или новые архитектуры вычислительных систем, отличающиеся от традиционной (фон Неймана), призваны решать усложняющиеся задачи информационной технологии, связанные с обработкой, отображением, накоплением данных, обменом данными, формализацией и генерацией знаний.

1.4.Классификация компьютеров по областям применения

По областям применения компьютеров может быть дана следующая классификация [10].

- Персональные компьютеры и рабочие станции
- X-терминалы
- Серверы
- Мейнфреймы
- Кластерные архитектуры
- Суперкомпьютеры

Персональные компьютеры и рабочие станции

Персональные компьютеры (ПК) появились в результате эволюции миниперсональных компьютеров при переходе элементной базы машин с малой и средней степенью интеграции на большие и

сверхбольшие интегральные схемы. ПК, благодаря своей низкой стоимости, очень быстро завоевали хорошие позиции на компьютерном рынке и создали предпосылки для разработки новых программных средств, ориентированных на конечного пользователя. Это прежде всего - "дружественные пользовательские интерфейсы", а также проблемно-ориентированные среды и инструментальные средства для автоматизации разработки прикладных программ.

Миникомпьютеры стали прародителями и другого направления развития современных систем - 32-разрядных машин. Создание RISC-процессоров и микросхем памяти емкостью более 1 Мбит привело к окончательному оформлению настольных систем высокой производительности, которые сегодня известны как рабочие станции. Первоначальная ориентация рабочих станций на профессиональных пользователей (в отличие от ПК, которые в начале ориентировались на самого широкого потребителя непрофессионала) привела к тому, что рабочие станции - это хорошо сбалансированные системы, в которых высокое быстродействие сочетается с большим объемом оперативной и внешней памяти, высокопроизводительными внутренними магистралями, высококачественной и быстродействующей графической подсистемой и разнообразными устройствами ввода/вывода. Это свойство выгодно отличает рабочие станции среднего и высокого класса от ПК и сегодня. Даже наиболее мощные IBM PC совместимые ПК не в состоянии удовлетворить возрастающие потребности систем обработки из-за наличия в их архитектуре ряда "узких мест".

Тем не менее быстрый рост производительности ПК на базе новейших микропроцессоров Intel в сочетании с резким снижением цен на эти изделия и развитием технологии локальных шин (VESA и PCI), позволяющей устранить многие "узкие места" в архитектуре ПК, делают современные персональные компьютеры весьма привлекательной альтернативой рабочим станциям. В свою очередь производители рабочих

станций создали изделия так называемого "начального уровня", которые по стоимостным характеристикам близки к высокопроизводительным ПК, но все еще сохраняют лидерство по производительности и возможностям наращивания. Насколько успешно удастся ПК на базе процессоров 486 и Pentium бороться против рабочих станций UNIX, покажет будущее, но уже в настоящее время появилось понятие "персональной рабочей станции", которое объединяет оба направления.

Современный рынок "персональных рабочих станций" не просто определить. По сути он представляет собой совокупность архитектурных платформ персональных компьютеров и рабочих станций, которые появились в настоящее время, поскольку поставщики компьютерного оборудования уделяют все большее внимание рынку продуктов для коммерции и бизнеса. Этот рынок традиционно считался вотчиной миникомпьютеров и мейнфреймов, которые поддерживали работу настольных терминалов с ограниченным интеллектом. В прошлом персональные компьютеры не были достаточно мощными и не располагали достаточными функциональными возможностями, чтобы служить адекватной заменой подключенных к главной машине терминалов. С другой стороны, рабочие станции на платформе UNIX были очень сильны в научном, техническом и инженерном секторах и были почти также неудобны, как и ПК для того чтобы выполнять серьезные офисные приложения. С тех пор ситуация изменилась коренным образом. Персональные компьютеры в настоящее время имеют достаточную производительность, а рабочие станции на базе UNIX имеют программное обеспечение, способное выполнять большинство функций, которые стали ассоциироваться с понятием "персональной рабочей станции". Вероятно оба этих направления могут серьезно рассматриваться в качестве сетевого ресурса для систем масштаба предприятия. В результате этих изменений практически ушли со сцены старомодные миникомпьютеры с их па-

тентованной архитектурой и использованием присоединяемых к главной машине терминалов. По мере продолжения процесса разукрупнения (downsizing) и увеличения производительности платформы Intel наиболее мощные ПК (но все же чаще открытые системы на базе UNIX) стали использоваться в качестве серверов, постепенно заменяя миникомпьютеры.

Среди других факторов, способствующих этому процессу, следует выделить:

Применение ПК стало более разнообразным. Помимо обычных для этого класса систем текстовых процессоров, даже средний пользователь ПК может теперь работать сразу с несколькими прикладными пакетами, включая электронные таблицы, базы данных и высококачественную графику.

Адаптация графических пользовательских интерфейсов существенно увеличила требования пользователей ПК к соотношению производительность/стоимость. И хотя оболочка MS Windows может работать на моделях ПК 386SX с 2 Мбайтами оперативной памяти, реальные пользователи хотели бы использовать все преимущества подобных систем, включая возможность комбинирования и эффективного использования различных пакетов.

Широкое распространение систем мультимедиа прямо зависит от возможности использования высокопроизводительных ПК и рабочих станций с адекватными аудио- и графическими средствами, и объемами оперативной и внешней памяти.

Слишком высокая стоимость мейнфреймов и даже систем среднего класса помогла сместить многие разработки в область распределенных систем и систем клиент-сервер, которые многим представляются вполне оправданной по экономическим соображениям альтернативой. Эти системы прямо базируются на высоконадежных и мощных рабочих станциях и серверах.

В начале представлялось, что необходимость сосредоточения высокой мощности на каждом рабочем месте приведет к переходу многих потребителей ПК на UNIX-станции. Это определялось частично тем, что RISC-процессоры, использовавшиеся в рабочих станциях на базе UNIX, были намного более производительными по сравнению с CISC-процессорами, применявшимися в ПК, а частично мощностью системы UNIX по сравнению с MS-DOS и даже OS/2.

Производители рабочих станций быстро отреагировали на потребность в низкостоимостных моделях для рынка коммерческих приложений. Потребность в высокой мощности на рабочем столе, объединенная с желанием поставщиков UNIX-систем продавать как можно больше своих изделий, привела такие компании как Sun Microsystems и Hewlett Packard на рынок рабочих станций для коммерческих приложений. И хотя значительная часть систем этих фирм все еще ориентирована на технические приложения, наблюдается беспрецедентный рост продаж продукции этих компаний для работы с коммерческими приложениями, требующими все большей и большей мощности для реализации сложных, сетевых прикладных систем, включая системы мультимедиа.

Это привело к временному отступлению производителей ПК на базе микропроцессоров Intel. Острая конкуренция со стороны производителей UNIX-систем и потребности в повышении производительности огромной уже инсталлированной базы ПК, заставили компанию Intel форсировать разработку высокопроизводительных процессоров семейства 486 и Pentium. Процессоры 486 и Pentium, при разработке которых были использованы многие подходы, применявшиеся ранее только в RISC-процессорах, а также использование других технологических усовершенствований, таких как архитектура локальной шины, позволили снабдить ПК достаточной мощностью, чтобы составить конкуренцию рабочим станциям во многих направлениях рынка коммерческих приложений. Правда для многих других приложений, в частности, в обла-

сти сложного графического моделирования, ПК все еще сильно отстают.

Х-терминалы

Х-терминалы представляют собой комбинацию бездисковых рабочих станций и стандартных ASCII-терминалов. Бездисковые рабочие станции часто применялись в качестве дорогих дисплеев и в этом случае не полностью использовали локальную вычислительную мощь. Одновременно многие пользователи ASCII-терминалов хотели улучшить их характеристики, чтобы получить возможность работы в многооконной системе и графические возможности. Совсем недавно, как только стали доступными очень мощные графические рабочие станции, появилась тенденция применения "подчиненных" Х-терминалов, которые используют рабочую станцию в качестве локального сервера.

На компьютерном рынке Х-терминалы занимают промежуточное положение между персональными компьютерами и рабочими станциями. Поставщики Х-терминалов заявляют, что их изделия более эффективны в стоимостном выражении, чем рабочие станции высокого ценового класса, и предлагают увеличенный уровень производительности по сравнению с персональными компьютерами. Быстрое снижение цен, прогнозируемое иногда в секторе Х-терминалов, в настоящее время идет очевидно благодаря обострившейся конкуренции в этом секторе рынка. Многие компании начали активно конкурировать за распределение рынка, а быстрый рост объемных поставок создал предпосылки для создания такого рынка. В настоящее время уже достигнута цена в \$1000 для Х-терминалов начального уровня, что делает эту технологию доступной для широкой пользовательской базы.

Как правило, стоимость Х-терминалов составляет около половины стоимости сравнимой по конфигурации бездисковой машины и примерно четверть стоимости полностью оснащенной рабочей станции.

Что такое Х-терминал?

Типовой X-терминал включает следующие элементы:

Экран высокого разрешения - обычно размером от 14 до 21 дюйма по диагонали;

Микропроцессор на базе Motorola 68xxx или RISC-процессор типа Intel i960, MIPS R3000 или AMD29000;

Отдельный графический сопроцессор в дополнение к основному процессору, поддерживающий двухпроцессорную архитектуру, которая обеспечивает более быстрое рисование на экране и прокручивание экрана;

Базовые системные программы, на которых работает система X-Windows и выполняются сетевые протоколы;

Программное обеспечение сервера X11;

Переменный объем локальной памяти (от 2 до 8 Мбайт) для дисплея, сетевого интерфейса, поддерживающего TCP/IP и другие сетевые протоколы.

Порты для подключения клавиатуры и мыши.

X-терминалы отличаются от ПК и рабочих станций не только тем, что не выполняет функции обычной локальной обработки. Работа X-терминалов зависит от главной (хост) системы, к которой они подключены посредством сети. Для того, чтобы X-терминал мог работать, пользователи должны установить программное обеспечение многооконного сервера X11 на главном процессоре, выполняющим прикладную задачу (наиболее известная версия X11 Release 5). X-терминалы отличаются также от стандартных алфавитно-цифровых ASCII и традиционных графических дисплейных терминалов тем, что они могут быть подключены к любой главной системе, которая поддерживает стандарт X-Windows. Более того, локальная вычислительная мощность X-терминала обычно используется для обработки отображения, а не обработки приложений (называемых клиентами), которые выполняются удаленно на главном компьютере (сервере). Вывод такого удаленного приложения просто отображается на экране X-терминала.

Минимальный объем требуемой для работы памяти X-терминала составляет 1 Мбайт, но чаще 2 Мбайта. В зависимости от функциональных возможностей изделия оперативная память может расширяться до 32 Мбайт и более.

Оснащенный стандартной системой X-Windows, X-терминал может отображать на одном и том же экране множество приложений одновременно. Каждое приложение может выполняться в своем окне и пользователь может изменять размеры окон, их месторасположение и манипулировать ими в любом месте экрана.

X-Windows - результат совместной работы Масачусетского технологического института (MIT) и корпорации DEC. Система X-Windows (известная также под именем X) в настоящее время является открытым де-факто стандартом для доступа к множеству одновременно выполняющихся приложений с возможностями многооконного режима и графикой высокого разрешения на интеллектуальных терминалах, персональных компьютерах, рабочих станциях и X-терминалах. Она стала стандартом для обеспечения интероперабельности (переносимости) продуктов многих поставщиков и для организации доступа к множеству приложений. В настоящее время X-Windows является стандартом для разработки пользовательского интерфейса. Более 90% поставщиков UNIX-рабочих станций и многие поставщики персональных компьютеров адаптировали систему X-Windows и применяют в качестве стандарта.

Серверы

Прикладные многопользовательские коммерческие и бизнес-системы, включающие системы управления базами данных и обработки транзакций, крупные издательские системы, сетевые приложения и системы обслуживания коммуникаций, разработку программного обеспечения и обработку изображений все более настойчиво требуют перехода к модели вычислений "клиент-сервер" и распределенной обработке. В распределенной (сетевой) модели "клиент-сервер" часть

работы выполняет сервер, а часть пользовательский компьютер (в общем случае клиентская и пользовательская части могут работать и на одном компьютере). Существует несколько типов серверов, ориентированных на разные применения: файл-сервер, сервер базы данных, принт-сервер, вычислительный сервер, сервер приложений. Таким образом, тип сервера определяется видом ресурса, которым он владеет (файловая система, база данных, принтеры, процессоры или прикладные пакеты программ).

С другой стороны существует классификация серверов, определяющаяся масштабом сети, в которой они используются: сервер рабочей группы, сервер отдела или сервер масштаба предприятия (корпоративный сервер). Эта классификация весьма условна. Например, размер группы может меняться в диапазоне от нескольких человек до нескольких сотен человек, а сервер отдела обслуживать от 20 до 150 пользователей. Очевидно в зависимости от числа пользователей и характера решаемых ими задач требования к составу оборудования и программного обеспечения сервера, к его надежности и производительности сильно варьируются.

Файловые серверы небольших рабочих групп (не более 20-30 человек) проще всего реализуются на платформе персональных компьютеров и программном обеспечении Novell NetWare. Файл-сервер, в данном случае, выполняет роль центрального хранилища данных. Серверы прикладных систем и высокопроизводительные машины для среды "клиент-сервер" значительно отличаются требованиями к аппаратным и программным средствам.

Типичными для небольших файл-серверов являются: процессор 486DX2/66 или более быстродействующий, 32-Мбайт ОЗУ, 2 Гбайт дискового пространства и один адаптер Ethernet 10BaseT, имеющий быстродействие 10 Мбит/с. В состав таких серверов часто включаются флоппи-дисковод и дисковод компакт-дисков. Графика для большинства сервер-

ров несущественна, поэтому достаточно иметь обычный монокромный монитор с разрешением VGA.

Скорость процессора для серверов с интенсивным вводом/выводом не критична. Они должны быть оснащены достаточно мощными блоками питания для возможности установки дополнительных плат расширения и дисковых накопителей. Желательно применение устройства бесперебойного питания. Оперативная память обычно имеет объем не менее 32 Мбайт, что позволит операционной системе (например, NetWare) использовать большие дисковые кэши и увеличить производительность сервера. Как правило, для работы с многозадачными операционными системами такие серверы оснащаются интерфейсом SCSI (или Fast SCSI). Распределение данных по нескольким жестким дискам может значительно повысить производительность.

При наличии одного сегмента сети и 10-20 рабочих станций пиковая пропускная способность сервера ограничивается максимальной пропускной способностью сети. В этом случае замена процессоров или дисковых подсистем более мощными не увеличивают производительность, так как узким местом является сама сеть. Поэтому важно использовать хорошую плату сетевого интерфейса.

Хотя влияние более быстрого процессора явно на производительности не сказывается, оно заметно снижает коэффициент использования ЦП. Во многих серверах этого класса используются процессоры 486DX2/66, Pentium с тактовой частотой 60 и 90 МГц, microSPARC-II и PowerPC. Аналогично процессорам влияние типа системной шины (EISA со скоростью 33 Мбит/с или PCI со скоростью 132 Мбит/с) также минимально при таком режиме использования.

Однако для файл-серверов общего доступа, с которыми одновременно могут работать несколько десятков, а то и сотен человек, простой однопроцессорной платформы и программного обеспечения Novell может оказаться недостаточно. В этом случае используются мощные многопроцессорные

серверы с возможностями наращивания оперативной памяти до нескольких гигабайт, дискового пространства до сотен гигабайт, быстрыми интерфейсами дискового обмена (типа Fast SCSI-2, Fast&Wide SCSI-2 и Fiber Channel) и несколькими сетевыми интерфейсами. Эти серверы используют операционную систему UNIX, сетевые протоколы TCP/IP и NFS. На базе многопроцессорных UNIX-серверов обычно строятся также серверы баз данных крупных информационных систем, так как на них ложится основная нагрузка по обработке информационных запросов. Подобного рода серверы получили название суперсерверов.

По уровню общесистемной производительности, функциональным возможностям отдельных компонентов, отказоустойчивости, а также в поддержке многопроцессорной обработки, системного администрирования и дисковых массивов большой емкости суперсерверы вышли в настоящее время на один уровень с мейнфреймами и мощными миникомпьютерами. Современные суперсерверы характеризуются:

- наличием двух или более центральных процессоров RISC, либо Pentium;

- многоуровневой шинной архитектурой, в которой запатентованная высокоскоростная системная шина связывает между собой несколько процессоров и оперативную память, а также множество стандартных шин ввода/вывода, размещенных в том же корпусе;

- поддержкой технологии дисковых массивов RAID;

- поддержкой режима симметричной многопроцессорной обработки, которая позволяет распределять задания по нескольким центральным процессорам или режима асимметричной многопроцессорной обработки, которая допускает выделение процессоров для выполнения конкретных задач.

Как правило, суперсерверы работают под управлением операционных систем UNIX, а в последнее время и Windows NT (на Digital 2100 Server Model A500MP), которые обеспечивают многопотокową многопроцессорную и многозадач-

ную обработку. Суперсерверы должны иметь достаточные возможности наращивания дискового пространства и вычислительной мощности, средства обеспечения надежности хранения данных и защиты от несанкционированного доступа. Кроме того, в условиях быстро растущей организации, важным условием является возможность наращивания и расширения уже существующей системы.

Мейнфреймы

Мейнфрейм - это синоним понятия "большая универсальная ЭВМ". Мейнфреймы и до сегодняшнего дня остаются наиболее мощными (не считая суперкомпьютеров) вычислительными системами общего назначения, обеспечивающими непрерывный круглосуточный режим эксплуатации. Они могут включать один или несколько процессоров, каждый из которых, в свою очередь, может оснащаться векторными сопроцессорами (ускорителями операций с суперкомпьютерной производительностью). В нашем сознании мейнфреймы все еще ассоциируются с большими по габаритам машинами, требующими специально оборудованных помещений с системами водяного охлаждения и кондиционирования. Однако это не совсем так. Прогресс в области элементно-конструкторской базы позволил существенно сократить габариты основных устройств. Наряду со сверхмощными мейнфреймами, требующими организации двухконтурной водяной системы охлаждения, имеются менее мощные модели, для охлаждения которых достаточно принудительной воздушной вентиляции, и модели, построенные по блочно-модульному принципу и не требующие специальных помещений и кондиционеров.

Основными поставщиками мейнфреймов являются известные компьютерные компании IBM, Amdahl, ICL, Siemens Nixdorf и некоторые другие, но ведущая роль принадлежит безусловно компании IBM. Именно архитектура системы IBM/360, выпущенной в 1964 году, и ее последующие поколения стали образцом для подражания. В нашей стране в те-

чение многих лет выпускались машины ряда ЕС ЭВМ, являвшиеся отечественным аналогом этой системы.

В архитектурном плане мейнфреймы представляют собой многопроцессорные системы, содержащие один или несколько центральных и периферийных процессоров с общей памятью, связанных между собой высокоскоростными магистралями передачи данных. При этом основная вычислительная нагрузка ложится на центральные процессоры, а периферийные процессоры (в терминологии IBM - селекторные, блок-мультиплексные, мультиплексные каналы и процессоры телеобработки) обеспечивают работу с широкой номенклатурой периферийных устройств.

Первоначально мейнфреймы ориентировались на централизованную модель вычислений, работали под управлением патентованных операционных систем и имели ограниченные возможности для объединения в единую систему оборудования различных фирм-поставщиков. Однако повышенный интерес потребителей к открытым системам, построенным на базе международных стандартов и позволяющим достаточно эффективно использовать все преимущества такого подхода, заставил поставщиков мейнфреймов существенно расширить возможности своих операционных систем в направлении совместимости. В настоящее время они демонстрируют свою "открытость", обеспечивая соответствие со спецификациями POSIX 1003.3, возможность использования протоколов межсоединений OSI и TCP/IP или предоставляя возможность работы на своих компьютерах под управлением операционной системы UNIX собственной разработки.

Стремительный рост производительности персональных компьютеров, рабочих станций и серверов создал тенденцию перехода с мейнфреймов на компьютеры менее дорогих классов: миникомпьютеры и многопроцессорные серверы. Эта тенденция получила название "разукрупнение" (downsizing). Однако этот процесс в самое последнее время несколько замедлился. Основной причиной возрождения интереса к мей-

нфреймам эксперты считают сложность перехода к распределенной архитектуре клиент-сервер, которая оказалась выше, чем предполагалось. Кроме того, многие пользователи считают, что распределенная среда не обладает достаточной надежностью для наиболее ответственных приложений, которой обладают мейнфреймы.

Очевидно выбор центральной машины (сервера) для построения информационной системы предприятия возможен только после глубокого анализа проблем, условий и требований конкретного заказчика и долгосрочного прогнозирования развития этой системы.

Главным недостатком мейнфреймов в настоящее время остается относительно низкое соотношение производительность/стоимость. Однако фирмами-поставщиками мейнфреймов предпринимаются значительные усилия по улучшению этого показателя.

Следует также помнить, что в мире существует огромная инсталлированная база мейнфреймов, на которой работают десятки тысяч прикладных программных систем. Отказаться от годами наработанного программного обеспечения просто не разумно. Поэтому в настоящее время ожидается рост продаж мейнфреймов по крайней мере до конца этого столетия. Эти системы, с одной стороны, позволяют модернизировать существующие системы, обеспечив сокращение эксплуатационных расходов, с другой стороны, создадут новую базу для наиболее ответственных приложений.

Кластерные архитектуры

Двумя основными проблемами построения вычислительных систем для критически важных приложений, связанных с обработкой транзакций, управлением базами данных и обслуживанием телекоммуникаций, являются обеспечение высокой производительности и продолжительного функционирования систем. Наиболее эффективный способ достижения заданного уровня производительности - применение параллельных масштабируемых архитектур. Задача обеспече-

ния продолжительного функционирования системы имеет три составляющих: надежность, готовность и удобство обслуживания. Все эти три составляющих предполагают, в первую очередь, борьбу с неисправностями системы, порождаемыми отказами и сбоями в ее работе. Эта борьба ведется по всем трем направлениям, которые взаимосвязаны и применяются совместно.

Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры. Повышение уровня готовности предполагает подавление в определенных пределах влияния отказов и сбоев на работу системы с помощью средств контроля и коррекции ошибок, а также средств автоматического восстановления вычислительного процесса после проявления неисправности, включая аппаратную и программную избыточность, на основе которой реализуются различные варианты отказоустойчивых архитектур. Повышение готовности есть способ борьбы за снижение времени простоя системы. Основные эксплуатационные характеристики системы существенно зависят от удобства ее обслуживания, в частности от ремонтпригодности, контролепригодности и т.д.

В последние годы в литературе по вычислительной технике все чаще употребляется термин "системы высокой готовности" (High Availability Systems). Все типы систем высокой готовности имеют общую цель - минимизацию времени простоя. Имеется два типа времени простоя компьютера: плановое и неплановое. Минимизация каждого из них требует различной стратегии и технологии. Плановое время простоя обычно включает время, принятое руководством, для проведения работ по модернизации системы и для ее обслу-

живания. Неплановое время простоя является результатом отказа системы или компонента. Хотя системы высокой готовности возможно больше ассоциируются с минимизацией неплановых простоев, они оказываются также полезными для уменьшения планового времени простоя.

Существует несколько типов систем высокой готовности, отличающиеся своими функциональными возможностями и стоимостью. Следует отметить, что высокая готовность не дается бесплатно. Стоимость систем высокой готовности на много превышает стоимость обычных систем. Вероятно, наибольшее распространение в мире получили кластерные системы потому, что они обеспечивают достаточно высокий уровень готовности систем при относительно низких затратах. Термин "кластеризация" на сегодня в компьютерной промышленности имеет много различных значений. Строгое определение могло бы звучать так: "реализация объединения машин, представляющегося единым целым для операционной системы, системного программного обеспечения, прикладных программ и пользователей". Машины, кластеризованные вместе таким способом могут при отказе одного процессора очень быстро перераспределить работу на другие процессоры внутри кластера. Это, возможно, наиболее важная задача многих поставщиков систем высокой готовности.

Первой концепцию кластерной системы анонсировала компания DEC, определив ее как группу объединенных между собой вычислительных машин, представляющих собой единый узел обработки информации. По существу VAX-кластер представляет собой слабосвязанную многомашинную систему с общей внешней памятью, обеспечивающую единый механизм управления и администрирования. В настоящее время на смену VAX-кластерам приходят UNIX-кластеры. При этом VAX-кластеры предлагают проверенный набор решений, который устанавливает критерии для оценки подобных систем.

VAX-кластер обладает следующими свойствами:

Разделение ресурсов. Компьютеры VAX в кластере могут разделять доступ к общим ленточным и дисковым накопителям. Все компьютеры VAX в кластере могут обращаться к отдельным файлам данных как к локальным.

Высокая готовность. Если происходит отказ одного из VAX-компьютеров, задания его пользователей автоматически могут быть перенесены на другой компьютер кластера. Если в системе имеется несколько контроллеров внешних накопителей и один из них отказывает, другие контроллеры автоматически подхватывают его работу.

Высокая пропускная способность. Ряд прикладных систем могут пользоваться возможностью параллельного выполнения заданий на нескольких компьютерах кластера.

Удобство обслуживания системы. Общие базы данных могут обслуживаться с единственного места. Прикладные программы могут устанавливаться только однажды на общих дисках кластера и разделяться между всеми компьютерами кластера.

Расширяемость. Увеличение вычислительной мощности кластера достигается подключением к нему дополнительных VAX-компьютеров. Дополнительные накопители на магнитных дисках и магнитных лентах становятся доступными для всех компьютеров, входящих в кластер.

Работа любой кластерной системы определяется двумя главными компонентами: высокоскоростным механизмом связи процессоров между собой и системным программным обеспечением, которое обеспечивает клиентам прозрачный доступ к системному сервису.

В настоящее время широкое распространение получила также технология параллельных баз данных. Эта технология позволяет множеству процессоров разделять доступ к единственной базе данных. Распределение заданий по множеству процессорных ресурсов и параллельное их выполнение позволяет достичь более высокого уровня пропускной способности транзакций, поддерживать большее число одновременно

работающих пользователей и ускорить выполнение сложных запросов. Существуют три различных типа архитектуры, которые поддерживают параллельные базы данных:

Симметричная многопроцессорная архитектура с общей памятью (Shared Memory SMP Architecture). Эта архитектура поддерживает единую базу данных, работающую на многопроцессорном сервере под управлением одной операционной системы. Увеличение производительности таких систем обеспечивается наращиванием числа процессоров, устройств оперативной и внешней памяти.

Архитектура с общими (разделяемыми) дисками (Shared Disk Architecture). Это типичный случай построения кластерной системы. Эта архитектура поддерживает единую базу данных при работе с несколькими компьютерами, объединенными в кластер (обычно такие компьютеры называются узлами кластера), каждый из которых работает под управлением своей копии операционной системы. В таких системах все узлы разделяют доступ к общим дискам, на которых собственно и располагается единая база данных. Производительность таких систем может увеличиваться как путем наращивания числа процессоров и объемов оперативной памяти в каждом узле кластера, так и посредством увеличения количества самих узлов.

Архитектура без разделения ресурсов (Shared Nothing Architecture). Как и в архитектуре с общими дисками, в этой архитектуре поддерживается единый образ базы данных при работе с несколькими компьютерами, работающими под управлением своих копий операционной системы. Однако в этой архитектуре каждый узел системы имеет собственную оперативную память и собственные диски, которые не разделяются между отдельными узлами системы. Практически в таких системах разделяется только общий коммуникационный канал между узлами системы. Производительность таких систем может увеличиваться путем добавления процессоров,

объемов оперативной и внешней (дисковой) памяти в каждом узле, а также путем наращивания количества таких узлов.

Таким образом, среда для работы параллельной базы данных обладает двумя важными свойствами: высокой готовностью и высокой производительностью. В случае кластерной организации несколько компьютеров или узлов кластера работают с единой базой данных. В случае отказа одного из таких узлов, оставшиеся узлы могут взять на себя задания, выполнявшиеся на отказавшем узле, не останавливая общий процесс работы с базой данных. Поскольку логически в каждом узле системы имеется образ базы данных, доступ к базе данных будет обеспечиваться до тех пор, пока в системе имеется по крайней мере один исправный узел. Производительность системы легко масштабируется, т.е. добавление дополнительных процессоров, объемов оперативной и дисковой памяти, и новых узлов в систему может выполняться в любое время, когда это действительно требуется.

Параллельные базы данных находят широкое применение в системах обработки транзакций в режиме on-line, системах поддержки принятия решений и часто используются при работе с критически важными для работы предприятий и организаций приложениями, которые эксплуатируются по 24 часа в сутки.

1.5. Общие требования, предъявляемые к современным компьютерам

В настоящее время к компьютерам предъявляются требования по следующим общим показателям [10].

- Отношение стоимость/производительность
- Надежность и отказоустойчивость
- Масштабируемость
- Совместимость и мобильность программного обеспечения

Отношение стоимость/производительность

Появление любого нового направления в вычислительной технике определяется требованиями компьютерного рынка. Поэтому у разработчиков компьютеров нет одной единственной цели. Большая универсальная вычислительная машина (мейнфрейм) или суперкомпьютер стоят дорого. Для достижения поставленных целей при проектировании высокопроизводительных конструкций приходится игнорировать стоимостные характеристики. Суперкомпьютеры фирмы Cray Research и высокопроизводительные мейнфреймы компании IBM относятся именно к этой категории компьютеров. Другим крайним примером может служить низкостоймостная конструкция, где производительность принесена в жертву для достижения низкой стоимости. К этому направлению относятся персональные компьютеры различных клонов IBM PC. Между этими двумя крайними направлениями находятся конструкции, основанные на отношении стоимость/ производительность, в которых разработчики находят баланс между стоимостными параметрами и производительностью. Типичными примерами такого рода компьютеров являются мини-компьютеры и рабочие станции.

Для сравнения различных компьютеров между собой обычно используются стандартные методики измерения производительности. Эти методики позволяют разработчикам и пользователям использовать полученные в результате испытаний количественные показатели для оценки тех или иных технических решений, и в конце концов именно производительность и стоимость дают пользователю рациональную основу для решения вопроса, какой компьютер выбрать.

Надежность и отказоустойчивость

Важнейшей характеристикой вычислительных систем является *надежность*. Повышение надежности основано на принципе предотвращения неисправностей путем снижения интенсивности отказов и сбоев за счет применения электронных схем и компонентов с высокой и сверхвысокой степенью

интеграции, снижения уровня помех, облегченных режимов работы схем, обеспечение тепловых режимов их работы, а также за счет совершенствования методов сборки аппаратуры.

Отказоустойчивость - это такое свойство вычислительной системы, которое обеспечивает ей, как логической машине, возможность продолжения действий, заданных программой, после возникновения неисправностей. Введение отказоустойчивости требует избыточного аппаратного и программного обеспечения. Направления, связанные с предотвращением неисправностей и с отказоустойчивостью, - основные в проблеме надежности. Концепции параллельности и отказоустойчивости вычислительных систем естественным образом связаны между собой, поскольку в обоих случаях требуются дополнительные функциональные компоненты. Поэтому, собственно, на параллельных вычислительных системах достигается как наиболее высокая производительность, так и, во многих случаях, очень высокая надежность. Имеющиеся ресурсы избыточности в параллельных системах могут гибко использоваться как для повышения производительности, так и для повышения надежности. Структура многопроцессорных и многомашинных систем приспособлена к автоматической реконфигурации и обеспечивает возможность продолжения работы системы после возникновения неисправностей.

Следует помнить, что понятие надежности включает не только аппаратные средства, но и программное обеспечение. Главной целью повышения надежности систем является целостность хранимых в них данных.

Масштабируемость

Масштабируемость представляет собой возможность наращивания числа и мощности процессоров, объемов оперативной и внешней памяти и других ресурсов вычислительной системы. Масштабируемость должна обеспечиваться архи-

тектурой и конструкцией компьютера, а также соответствующими средствами программного обеспечения.

Добавление каждого нового процессора в действительно масштабируемой системе должно давать прогнозируемое увеличение производительности и пропускной способности при приемлемых затратах. Одной из основных задач при построении масштабируемых систем является минимизация стоимости расширения компьютера и упрощение планирования. В идеале добавление процессоров к системе должно приводить к линейному росту ее производительности. Однако это не всегда так. Потери производительности могут возникать, например, при недостаточной пропускной способности шин из-за возрастания трафика между процессорами и основной памятью, а также между памятью и устройствами ввода/вывода. В действительности реальное увеличение производительности трудно оценить заранее, поскольку оно в значительной степени зависит от динамики поведения прикладных задач.

Возможность масштабирования системы определяется не только архитектурой аппаратных средств, но зависит от заложенных свойств программного обеспечения. Масштабируемость программного обеспечения затрагивает все его уровни от простых механизмов передачи сообщений до работы с такими сложными объектами как мониторы транзакций и вся среда прикладной системы. В частности, программное обеспечение должно минимизировать трафик межпроцессорного обмена, который может препятствовать линейному росту производительности системы. Аппаратные средства (процессоры, шины и устройства ввода/вывода) являются только частью масштабируемой архитектуры, на которой программное обеспечение может обеспечить предсказуемый рост производительности. Важно понимать, что простой переход, например, на более мощный процессор может привести к перегрузке других компонентов системы. Это означает, что

действительно масштабируемая система должна быть сбалансирована по всем параметрам.

Совместимость и мобильность программного обеспечения

Концепция программной совместимости впервые в широких масштабах была применена разработчиками системы IBM/360. Основная задача при проектировании всего ряда моделей этой системы заключалась в создании такой архитектуры, которая была бы одинаковой с точки зрения пользователя для всех моделей системы независимо от цены и производительности каждой из них. Огромные преимущества такого подхода, позволяющего сохранять существующий задел программного обеспечения при переходе на новые (как правило, более производительные) модели были быстро оценены как производителями компьютеров, так и пользователями и начиная с этого времени практически все фирмы-поставщики компьютерного оборудования взяли на вооружение эти принципы, поставляя серии совместимых компьютеров. Следует заметить однако, что со временем даже самая передовая архитектура неизбежно устаревает и возникает потребность внесения радикальных изменений в архитектуру и способы организации вычислительных систем.

В настоящее время одним из наиболее важных факторов, определяющих современные тенденции в развитии информационных технологий, является ориентация компаний-поставщиков компьютерного оборудования на рынок прикладных программных средств. Это объясняется прежде всего тем, что для конечного пользователя в конце концов важно программное обеспечение, позволяющее решить его задачи, а не выбор той или иной аппаратной платформы. Переход от однородных сетей программно совместимых компьютеров к построению неоднородных сетей, включающих компьютеры разных фирм-производителей, в корне изменил и точку зрения на саму сеть: из сравнительно простого средства обмена

информацией она превратилась в средство интеграции отдельных ресурсов - мощную распределенную вычислительную систему, каждый элемент которой (сервер или рабочая станция) лучше всего соответствует требованиям конкретной прикладной задачи.

Этот переход выдвинул ряд новых требований. Прежде всего такая вычислительная среда должна позволять гибко менять количество и состав аппаратных средств и программного обеспечения в соответствии с меняющимися требованиями решаемых задач. Во-вторых, она должна обеспечивать возможность запуска одних и тех же программных систем на различных аппаратных платформах, т.е. обеспечивать мобильность программного обеспечения. В третьих, эта среда должна гарантировать возможность применения одних и тех же человеко-машинных интерфейсов на всех компьютерах, входящих в неоднородную сеть. В условиях жесткой конкуренции производителей аппаратных платформ и программного обеспечения сформировалась концепция открытых систем, представляющая собой совокупность стандартов на различные компоненты вычислительной среды, предназначенных для обеспечения мобильности программных средств в рамках неоднородной, распределенной вычислительной системы.

Одним из вариантов моделей открытой среды является модель OSE (Open System Environment), предложенная комитетом IEEE POSIX. На основе этой модели национальный институт стандартов и технологии США выпустил документ "Application Portability Profile (APP). The U.S. Government's Open System Environment Profile OSE/1 Version 2.0", который определяет рекомендуемые для федеральных учреждений США спецификации в области информационных технологий, обеспечивающие мобильность системного и прикладного программного обеспечения. Все ведущие производители компьютеров и программного обеспечения в США в настоящее время придерживаются требований этого документа

1.6.Оценка производительности вычислительных систем

Общие замечания

Основу для сравнения различных типов компьютеров между собой дают стандартные методики измерения производительности [11]. В процессе развития вычислительной техники появилось несколько таких стандартных методик. Они позволяют разработчикам и пользователям осуществлять выбор между альтернативами на основе количественных показателей, что дает возможность постоянного прогресса в данной области.

Единицей измерения производительности компьютера является время: компьютер, выполняющий тот же объем работы за меньшее время является более быстрым. Время выполнения любой программы измеряется в секундах. Часто производительность измеряется как скорость появления некоторого числа событий в секунду, так что меньшее время подразумевает большую производительность.

Однако в зависимости от того, что мы считаем, время может быть определено различными способами. Наиболее простой способ определения времени называется астрономическим временем, временем ответа (response time), временем выполнения(execution time) или прошедшим временем(elapsed time). Это задержка выполнения задания, включающая буквально все: работу процессора, обращения к диску, обращения к памяти, ввод/вывод и накладные расходы операционной системы. Однако при работе в мультипрограммном режиме во время ожидания ввода/вывода для одной программы, процессор может выполнять другую программу, и система не обязательно будет минимизировать время выполнения данной конкретной программы.

Для измерения времени работы процессора на данной программе используется специальный параметр - время ЦП (CPU time), которое не включает время ожидания ввода/вывода или время выполнения другой программы. Оче-

видно, что время ответа, видимое пользователем, является полным временем выполнения программы, а не временем ЦП. Время ЦП может далее делиться на время, потраченное ЦП непосредственно на выполнение программы пользователя и называемое пользовательским временем ЦП, и время ЦП, затраченное операционной системой на выполнение заданий, затребованных программой, и называемое системным временем ЦП.

В ряде случаев системное время ЦП игнорируется из-за возможной неточности измерений, выполняемых самой операционной системой, а также из-за проблем, связанных со сравнением производительности машин с разными операционными системами. С другой стороны, системный код на некоторых машинах является пользовательским кодом на других и, кроме того, практически никакая программа не может работать без некоторой операционной системы. Поэтому при измерениях производительности процессора часто используется сумма пользовательского и системного времени ЦП.

В большинстве современных процессоров скорость протекания процессов взаимодействия внутренних функциональных устройств определяется не естественными задержками в этих устройствах, а задается единой системой синхросигналов, вырабатываемых некоторым генератором тактовых импульсов, как правило, работающим с постоянной скоростью. Дискретные временные события называются тактами синхронизации (clock ticks), просто тактами (ticks), периодами синхронизации (clock periods), циклами (cycles) или циклами синхронизации (clock cycles). Разработчики компьютеров обычно говорят о периоде синхронизации, который определяется либо своей длительностью (например, 10 наносекунд), либо частотой (например, 100 МГц). Длительность периода синхронизации есть величина, обратная к частоте синхронизации.

Таким образом, время ЦП для некоторой программы может быть выражено двумя способами: количеством тактов

синхронизации для данной программы, умноженным на длительность такта синхронизации, либо количеством тактов синхронизации для данной программы, деленным на частоту синхронизации.

Важной характеристикой, часто публикуемой в отчетах по процессорам, является среднее количество тактов синхронизации на одну команду - CPI (clock cycles per instruction). При известном количестве выполняемых команд в программе этот параметр позволяет быстро оценить время ЦП для данной программы.

Таким образом, производительность ЦП зависит от трех параметров: такта (или частоты) синхронизации, среднего количества тактов на команду и количества выполняемых команд. Невозможно изменить ни один из указанных параметров изолированно от другого, поскольку базовые технологии, используемые для изменения каждого из этих параметров, взаимосвязаны: частота синхронизации определяется технологией аппаратных средств и функциональной организацией процессора; среднее количество тактов на команду зависит от функциональной организации и архитектуры системы команд; а количество выполняемых в программе команд определяется архитектурой системы команд и технологией компиляторов. Когда сравниваются две машины, необходимо рассматривать все три компонента, чтобы понять относительную производительность.

В процессе поиска стандартной единицы измерения производительности компьютеров было принято несколько популярных единиц измерения, вследствие чего несколько безвредных терминов были искусственно вырваны из их хорошо определенного контекста и использованы там, для чего они никогда не предназначались. В действительности единственной подходящей и надежной единицей измерения производительности является время выполнения реальных программ, и все предлагаемые замены этого времени в качестве единицы измерения или замены реальных программ в каче-

стве объектов измерения на синтетические программы только вводят в заблуждение.

MIPS

Одной из альтернативных единиц измерения производительности процессора (по отношению к времени выполнения) является MIPS - (миллион команд в секунду). Имеется несколько различных вариантов интерпретации определения MIPS.

В общем случае MIPS есть скорость операций в единицу времени, т.е. для любой данной программы MIPS есть просто отношение количества команд в программе к времени ее выполнения. Таким образом, производительность может быть определена как обратная к времени выполнения величина, причем более быстрые машины при этом будут иметь более высокий рейтинг MIPS.

Положительными сторонами MIPS является то, что эту характеристику легко понять, особенно покупателю, и что более быстрая машина характеризуется большим числом MIPS, что соответствует нашим интуитивным представлениям. Однако использование MIPS в качестве метрики для сравнения наталкивается на три проблемы. Во-первых, MIPS зависит от набора команд процессора, что затрудняет сравнение по MIPS компьютеров, имеющих разные системы команд. Во-вторых, MIPS даже на одном и том же компьютере меняется от программы к программе. В-третьих, MIPS может меняться по отношению к производительности в противоположенную сторону.

Классическим примером для последнего случая является рейтинг MIPS для машины, в состав которой входит сопроцессор плавающей точки. Поскольку в общем случае на каждую команду с плавающей точкой требуется большее количество тактов синхронизации, чем на целочисленную команду, то программы, используя сопроцессор плавающей точки вместо соответствующих подпрограмм из состава программного

обеспечения, выполняются за меньшее время, но имеют меньший рейтинг MIPS. При отсутствии сопроцессора операции над числами с плавающей точкой реализуются с помощью подпрограмм, использующих более простые команды целочисленной арифметики и, как следствие, такие машины имеют более высокий рейтинг MIPS, но выполняют настолько большее количество команд, что общее время выполнения значительно увеличивается. Подобные аномалии наблюдаются и при использовании оптимизирующих компиляторов, когда в результате оптимизации сокращается количество выполняемых в программе команд, рейтинг MIPS уменьшается, а производительность увеличивается.

Другое определение MIPS связано с очень популярным когда-то компьютером VAX 11/780 компании DEC. Именно этот компьютер был принят в качестве эталона для сравнения производительности различных машин. Считалось, что производительность VAX 11/780 равна 1MIPS (одному миллиону команд в секунду).

В то время широкое распространение получил синтетический тест Dhrystone, который позволял оценивать эффективность процессоров и компиляторов с языка C для программ нечисловой обработки. Он представлял собой тестовую смесь, 53% которой составляли операторы присваивания, 32% - операторы управления и 15% - вызовы функций. Это был очень короткий тест: общее число команд равнялось 100. Скорость выполнения программы из этих 100 команд измерялась в Dhrystone в секунду. Быстродействие VAX 11/780 на этом синтетическом тесте составляло 1757Dhrystone в секунду. Таким образом 1MIPS равен 1757 Dhrystone в секунду.

Следует отметить, что в настоящее время тест Dhrystone практически не применяется. Малый объем позволяет разместить все команды теста в кэш-памяти первого уровня современного микропроцессора и он не позволяет даже оценить эффект наличия кэш-памяти второго уровня, хотя может хорошо отражать эффект увеличения тактовой частоты.

Третье определение MIPS связано с IBM RS/6000 MIPS. Дело в том, что ряд производителей и пользователей (последователей фирмы IBM) предпочитают сравнивать производительность своих компьютеров с производительностью современных компьютеров IBM, а не со старой машиной компании DEC. Соотношение между VAX MIPS и RS/6000 MIPS никогда широко не публиковались, но 1 RS/6000 MIPS примерно равен 1.6 VAX 11/780 MIPS.

MFLOPS

Измерение производительности компьютеров при решении научно-технических задач, в которых существенно используется арифметика с плавающей точкой, всегда вызывало особый интерес. Именно для таких вычислений впервые встал вопрос об измерении производительности, а по достигнутым показателям часто делались выводы об общем уровне разработок компьютеров. Обычно для научно-технических задач производительность процессора оценивается в MFLOPS (миллионах чисел-результатов вычислений с плавающей точкой в секунду, или миллионах элементарных арифметических операций над числами с плавающей точкой, выполненных в секунду).

Как единица измерения, MFLOPS, предназначена для оценки производительности только операций с плавающей точкой, и поэтому не применима вне этой ограниченной области. Например, программы компиляторов имеют рейтинг MFLOPS близкий к нулю вне зависимости от того, насколько быстра машина, поскольку компиляторы редко используют арифметику с плавающей точкой.

Ясно, что рейтинг MFLOPS зависит от машины и от программы. Этот термин менее безобидный, чем MIPS. Он базируется на количестве выполняемых операций, а не на количестве выполняемых команд. По мнению многих программистов, одна и та же программа, работающая на различных компьютерах, будет выполнять различное количество ко-

манд, но одно и то же количество операций с плавающей точкой. Именно поэтому рейтинг MFLOPS предназначался для справедливого сравнения различных машин между собой.

Однако и с MFLOPS не все обстоит так безоблачно. Прежде всего, это связано с тем, что наборы операций с плавающей точкой не совместимы на различных компьютерах. Например, в суперкомпьютерах фирмы Cray Research отсутствует команда деления (имеется, правда, операция вычисления обратной величины числа с плавающей точкой, а операция деления может быть реализована с помощью умножения делимого на обратную величину делителя). В то же время многие современные микропроцессоры имеют команды деления, вычисления квадратного корня, синуса и косинуса.

Другая, осознаваемая всеми, проблема заключается в том, что рейтинг MFLOPS меняется не только на смеси целочисленных операций и операций с плавающей точкой, но и на смеси быстрых и медленных операций с плавающей точкой. Например, программа со 100% операций сложения будет иметь более высокий рейтинг, чем программа со 100% операций деления.

Решение обеих проблем заключается в том, чтобы взять "каноническое" или "нормализованное" число операций с плавающей точкой из исходного текста программы и затем поделить его на время выполнения. На рис. 3.1 показано, каким образом авторы тестового пакета "Ливерморские циклы", о котором речь пойдет ниже, вычисляют для программы количество нормализованных операций с плавающей точкой в соответствии с операциями, действительно находящимися в ее исходном тексте. Таким образом, рейтинг реальных MFLOPS отличается от рейтинга нормализованных MFLOPS, который часто приводится в литературе по суперкомпьютерам.

Таблица 1. Соотношение между реальными и нормализованными операциями с плавающей точкой, которым пользуются авторы "ливерморских циклов" для вычисления рейтинга MFLOPS

Реальные операции с плавающей точкой	Нормализованные операции с плавающей точкой
Сложение, вычитание, сравнение, умножение	1
Деление, квадратный корень	4
Экспонента, синус, ...	8

Наиболее часто MFLOPS, как единица измерения производительности, используется при проведении контрольных испытаний на тестовых пакетах "Ливерморские циклы" и

LINPACK

Ливерморские циклы - это набор фрагментов фортран-программ, каждый из которых взят из реальных программных систем, эксплуатируемых в Ливерморской национальной лаборатории им. Лоуренса (США). Обычно при проведении испытаний используется либо малый набор из 14 циклов, либо большой набор из 24 циклов.

Пакет Ливерморских циклов используется для оценки производительности вычислительных машин с середины 60-х годов. Ливерморские циклы считаются типичными фрагментами программ численных задач. Появление новых типов машин, в том числе векторных и параллельных, не уменьшило важности Ливерморских циклов, однако изменились значения производительности и величины разброса между разными циклами.

На векторной машине производительность зависит не только от элементной базы, но и от характера самого алгоритма, т.е. коэффициента векторизуемости. Среди Ливермор-

ских циклов коэффициент векторизуемости колеблется от 0 до 100%, что еще раз подтверждает их ценность для оценки производительности векторных архитектур. Кроме характера алгоритма, на коэффициент векторизуемости влияет и качество векторизатора, встроенного в компилятор.

На параллельной машине производительность существенно зависит от соответствия между структурой аппаратных связей вычислительных элементов и структурой вычислений в алгоритме. Важно, чтобы тестовый пакет представлял алгоритмы различных структур. В Ливерморских циклах встречаются последовательные, сеточные, конвейерные, волновые вычислительные алгоритмы, что подтверждает их пригодность и для параллельных машин. Однако обобщение результатов измерения производительности, полученных для одной параллельной машины, на другие параллельные машины или хотя бы некоторый подкласс параллельных машин, может дать неверный результат, ибо структуры аппаратных связей в таких машинах гораздо более разнообразны, чем, скажем, в векторных машинах.

LINPACK - это пакет фортран-программ для решения систем линейных алгебраических уравнений. Целью создания LINPACK отнюдь не было измерение производительности. Алгоритмы линейной алгебры весьма широко используются в самых разных задачах, и поэтому измерение производительности на LINPACK представляют интерес для многих пользователей. Сведения о производительности различных машин на пакете LINPACK публикуются сотрудником Аргоннской национальной лаборатории (США) Дж. Донгаррой и периодически обновляются.

В основе алгоритмов действующего варианта LINPACK лежит метод декомпозиции. Исходная матрица размером 100×100 элементов (в последнем варианте размером 1000×1000) сначала представляется в виде произведения двух матриц стандартной структуры, над которыми затем выполняется собственно алгоритм нахождения решения. Подпро-

граммы, входящие в LINPACK, структурированы. В стандартном варианте LINPACK выделен внутренний уровень базовых подпрограмм, каждая из которых выполняет элементарную операцию над векторами. Набор базовых подпрограмм называется BLAS (Basic Linear Algebra Subprograms). Например, в BLAS входят две простые подпрограммы SAXPY (умножение вектора на скаляр и сложение векторов) и SDOT (скалярное произведение векторов). Все операции выполняются над числами с плавающей точкой, представленными с двойной точностью. Результат измеряется в MFLOPS.

Использование результатов работы тестового пакета LINPACK с двойной точностью как основы для демонстрации рейтинга MFLOPS стало общепринятой практикой в компьютерной промышленности. При этом следует помнить, что при использовании исходной матрицы размером 100×100 , она полностью может размещаться в кэш-памяти емкостью, например, 1 Мбайт. Если при проведении испытаний используется матрица размером 1000×1000 , то емкости такого кэша уже недостаточно и некоторые обращения к памяти будут ускоряться благодаря наличию такого кэша, другие же будут приводить к промахам и потребуют большего времени на обработку обращений к памяти. Для многопроцессорных систем также имеются параллельные версии LINPACK и такие системы часто показывают линейное увеличение производительности с ростом числа процессоров.

Однако, как и любая другая единица измерения, рейтинг MFLOPS для отдельной программы не может быть обобщен на все случаи жизни, чтобы представлять единственную единицу измерения производительности компьютера, хотя очень соблазнительно характеризовать машину единственным рейтингом MIPS или MFLOPS без указания программы.

SPECint92, SPECfp92

Важность создания пакетов тестов, базирующихся на реальных прикладных программах широкого круга пользова-

лей и обеспечивающих эффективную оценку производительности процессоров, была осознана большинством крупнейших производителей компьютерного оборудования, которые в 1988 году учредили бесприбыльную корпорацию SPEC (Standard Performance Evaluation Corporation). Основной целью этой организации является разработка и поддержка стандартизованного набора специально подобранных тестовых программ для оценки производительности новейших поколений высокопроизводительных компьютеров. Членом SPEC может стать любая организация, уплатившая вступительный взнос.

Главными видами деятельности SPEC являются:

Разработка и публикация наборов тестов, предназначенных для измерения производительности компьютеров. Перед публикацией объектные коды этих наборов вместе с исходными текстами и инструментальными средствами интенсивно проверяются на предмет возможности импортирования на разные платформы. Они доступны для широкого круга пользователей за плату, покрывающую расходы на разработку и административные издержки. Специальное лицензионное соглашение регулирует вопросы выполнения тестирования и публикации результатов в соответствии с документацией на каждый тестовый набор.

SPEC публикует ежеквартальный отчет о новостях SPEC и результатах тестирования: "The SPEC Newsletter", что обеспечивает централизованный источник информации для результатов тестирования на тестах SPEC.

Основным результатом работы SPEC являются наборы тестов. Эти наборы разрабатываются SPEC с использованием кодов, поступающих из разных источников. SPEC работает над импортированием этих кодов на разные платформы, а также создает инструментальные средства для формирования из кодов, выбранных в качестве тестов, осмысленных рабочих нагрузок. Поэтому тесты SPEC отличаются от свободно распространяемых программ. Хотя они могут существовать

под похожими или теми же самыми именами, время их выполнения в общем случае будет отличаться.

В настоящее время имеется два базовых набора тестов SPEC, ориентированных на интенсивные расчеты и измеряющих производительность процессора, системы памяти, а также эффективность генерации кода компилятором. Как правило, эти тесты ориентированы на операционную систему UNIX, но они также импортированы и на другие платформы. Процент времени, расходуемого на работу операционной системы и функции ввода/вывода, в общем случае ничтожно мал.

Набор тестов CINT92, измеряющий производительность процессора при обработке целых чисел, состоит из шести программ, написанных на языке Си и выбранных из различных прикладных областей: теория цепей, интерпретатор языка Лисп, разработка логических схем, упаковка текстовых файлов, электронные таблицы и компиляция программ.

Набор тестов CFP92, измеряющий производительность процессора при обработке чисел с плавающей точкой, состоит из 14 программ, также выбранных из различных прикладных областей: разработка аналоговых схем, моделирование методом Монте-Карло, квантовая химия, оптика, робототехника, квантовая физика, астрофизика, прогноз погоды и другие научные и инженерные задачи. Две программы из этого набора написаны на языке Си, а остальные 12 - на Фортране. В пяти программах используется одинарная, а в остальных - двойная точность.

Результаты прогона каждого индивидуального теста из этих двух наборов выражаются отношением времени выполнения одной копии теста на тестируемой машине к времени ее выполнения на эталонной машине. В качестве эталонной машины используется VAX 11/780. SPEC публикует результаты прогона каждого отдельного теста, а также две составные оценки: SPECint92 - среднее геометрическое 6 результатов индивидуальных тестов из набора CINT92 и SPECfp92 -

среднее геометрическое 14 результатов индивидуальных тестов из набора CFP92.

Следует отметить, что результаты тестирования на наборах CINT92 и CFT92 сильно зависят от качества применяемых оптимизирующих компиляторов. Для более точного выяснения возможностей аппаратных средств с середины 1994 года SPEC ввел две дополнительные составные оценки: SPECbase_int92 и SPECbase_fp92, которые накладывает определенные ограничения на используемые компиляторы поставщиками компьютеров при проведении испытаний.

TPC-A, TPC-B, TPC-C

По мере расширения использования компьютеров при обработке транзакций в сфере бизнеса все более важной становится возможность справедливого сравнения систем между собой. С этой целью в 1988 году был создан Совет по оценке производительности обработки транзакций (TPC - Transaction Processing Performance Council), который представляет собой неприбыльную организацию. Любая компания или организация может стать членом TPC после уплаты соответствующего взноса. На сегодня членами TPC являются практически все крупнейшие производители аппаратных платформ и программного обеспечения для автоматизации коммерческой деятельности. К настоящему времени TPC создал три тестовых пакета для обеспечения объективного сравнения различных систем обработки транзакций и планирует создать новые оценочные тесты.

В компьютерной индустрии термин транзакция (transaction) может означать почти любой вид взаимодействия или обмена информацией. Однако в мире бизнеса "транзакция" имеет вполне определенный смысл: коммерческий обмен товарами, услугами или деньгами. В настоящее время практически все бизнес-транзакции выполняются с помощью компьютеров. Наиболее характерными примерами систем обработки транзакций являются системы управления учетом, системы резервирования авиабилетов и банковские системы.

Таким образом, необходимость стандартов и тестовых пакетов для оценки таких систем все больше усиливается.

До 1988 года отсутствовало общее согласие относительно методики оценки систем обработки транзакций. Широко использовались два тестовых пакета: Дебет/Кредит и TP1. Однако эти пакеты не позволяли осуществлять адекватную оценку систем: они не имели полных, основательных спецификаций; не давали объективных, проверяемых результатов; не содержали полного описания конфигурации системы, ее стоимости и методологии тестирования; не обеспечивали объективного, беспристрастного сравнения одной системы с другой.

Чтобы решить эти проблемы, и была создана организация TRC, основной задачей которой является точное определение тестовых пакетов для оценки систем обработки транзакций и баз данных, а также для распространения объективных, проверяемых данных в промышленности.

TRC публикует спецификации тестовых пакетов, которые регулируют вопросы, связанные с работой тестов. Эти спецификации гарантируют, что покупатели имеют объективные значения данных для сравнения производительности различных вычислительных систем. Хотя реализация спецификаций оценочных тестов оставлена на усмотрение индивидуальных спонсоров тестов, сами спонсоры, объявляя результаты TRC, должны представить TRC детальные отчеты, документирующие соответствие всем спецификациям. Эти отчеты, в частности, включают конфигурацию системы, методику калькуляции цены, диаграммы значений производительности и документацию, показывающую, что тест соответствует требованиям атомарности, согласованности, изолированности и долговечности (ACID - atomicity, consistency, isolation, and durability), которые гарантируют, что все транзакции из оценочного теста обрабатываются должным образом.

Работой TRC руководит Совет Полного Состава (Full Council), который принимает все решения; каждая компания-

участник имеет один голос, а для того, чтобы провести какое-либо решение требуется две трети голосов. Управляющий Комитет (Steering Committee), состоящий из пяти представителей и избираемый ежегодно, надзирает за работой администрации ТРС, поддерживает и обеспечивает все направления и рекомендации для членов Совета Полного Состава и Управляющего Комитета.

В составе ТРС имеются два типа подкомитетов: постоянные подкомитеты, которые управляют администрацией ТРС, осуществляют связи с общественностью и обеспечивают выпуск документации; и технические подкомитеты, которые формируются для разработки предложений по оценочным тестам и распускаются после того, как их работа по разработке завершена.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Дайте определение понятию „система“ и объясните ее свойства.
2. Нарисуйте и объясните укрупненную структурную схему системы управления.
3. Почему самым сложным элементом системы управления является управляющий орган?
4. Что такое обратная связь?
5. В чем заключается особенность автоматизированных систем управления?
6. Объясните суть базовой информационных технологий.
7. Проведите классификацию компьютеров по областям применения.
8. Назовите общие требования, предъявляемые к современным компьютерам.
9. Каковы достоинства и недостатки существующих методов оценки производительности вычислительных систем?

Глава 2. Нетрадиционные архитектуры вычислительных систем

Термин "архитектура системы" [11,10] часто употребляется как в узком, так и в широком смысле этого слова. В узком смысле под архитектурой понимается архитектура набора команд. Архитектура набора команд служит границей между аппаратурой и программным обеспечением и представляет ту часть системы, которая видна программисту или разработчику компиляторов. Следует отметить, что это наиболее частое употребление этого термина. В широком смысле архитектура охватывает понятие организации системы, включающее такие высокоуровневые аспекты разработки компьютера, как систему памяти и процессоров, структуру системной шины, организацию ввода/вывода и т.п.

Применительно к вычислительным системам термин "архитектура" может быть определен как распределение функций, реализуемых системой, между ее уровнями, точнее как определение границ между этими уровнями. Таким образом, архитектура вычислительной системы предполагает многоуровневую организацию. Архитектура первого уровня определяет, какие функции по обработке данных выполняются системой в целом, а какие возлагаются на внешний мир (пользователей, операторов, администраторов баз данных и т.д.). Система взаимодействует с внешним миром через набор интерфейсов: языки (язык оператора, языки программирования, языки описания и манипулирования базой данных, язык управления заданиями) и системные программы (программы-утилиты, программы редактирования, сортировки, сохранения и восстановления информации).

Представляя себе компьютер, мы обычно имеем в виду несколько его функций: вычисления в центральном процессоре, обмен данными между центральным процессором и внешними устройствами. Мы также вспоминаем об операци-

онной системе, утилитах, сортировке, слиянии и т. д. Если «вычисления» понимаются обычно как арифметические и логические операции над данными, хранящимися в основной памяти, то термином *обработка* мы будем обозначать всю последовательность действий (например: ввод, вычисления, сортировка, вывод), необходимых для выполнения некоторого задания.

2.1. Числовая и нечисловая обработка

Известно, что компьютеры были созданы для удовлетворения нужд крупных учреждений при реализации большого объема вычислений, для которых существенными были точность и время выполнения. Как правило, эти вычисления представляют собой длинные цепочки итераций и требуют сохранения высокой точности (примерами таких вычислений могут служить решение линейных и дифференциальных уравнений, преобразование Лапласа, операции с матрицами, векторами и т. д.). Такие вычисления характерны для числовой обработки.

По мере распространения компьютеров и создания, в частности, персональных ЭВМ возникли другие области их применения, отличные от вычислений. Так, появилась необходимость в обработке экономической информации, в создании информационных систем для различных организаций, автоматизации работ в учреждениях и т. д. Все эти применения требуют различных баз данных, которые могут хранить миллионы и миллиарды отдельных записей. В отличие от числовой обработки в данном случае не требуются высокая точность и большой объем вычислений. Обычно достаточно выполнить одно сложение или умножение (например, при вычислении налога). Однако объем обрабатываемых данных велик (например, нужно вычислить налог с каждого вида товара или с каждого клиента), поэтому и базы данных, содержащие требуемую информацию, огромны.

Кроме того, нужно предварительно найти требуемую запись, обработать ее и определить форму вывода обработанных данных. Для этого требуются такие операции, как поиск и сортировка. Процесс, который только что описан, характеризует нечисловую обработку данных. Легко убедиться в том, что на нечисловую обработку в настоящее время тратится машинного времени значительно больше, чем на числовую [6].

Данные — это цифровые и графические сведения об объектах окружающего мира. В понятие «данные» при числовой и нечисловой обработке вкладывается различное содержание. При числовой обработке используются такие объекты, как *переменные, векторы, матрицы, многомерные массивы, константы* и т.д. При нечисловой обработке объектами могут быть *файлы, записи, поля, иерархии, сети, отношения* и т. д.

Любые данные в той или иной степени обладают некоторым содержанием. При числовой обработке мы хотим, например, вычислить сумму элементов массива или перемножить две переменные. При этом содержание элементов массива мало о чем нам говорит. Для суммирования достаточно указать адрес начального элемента массива, а затем накапливать сумму элементов в некотором регистре, индексируя соответствующим образом адреса. В случае умножения двух переменных нас не особенно интересует способ, каким это вычисление запрограммировано, или текущее значение перемножаемых переменных, но мы должны проследить за тем, чтобы предварительно был очищен накопитель и были проверены некоторые переменные, влияющие на ход выполнения программы. Даже в условном операторе мы обращаемся к элементу данных не по содержанию, а по имени (например, элемент матрицы $A(I,J)$). Таким образом, при числовой обработке содержание данных не имеет большого значения.

При нечисловой обработке, наоборот, нас интересуют непосредственные сведения об объектах (конкретный служа-

щий или группа служащих), а не файл служащих как таковой. Мы не индексируем файл служащих для выбора конкретного человека (по крайней мере, в нашей программе); нас больше интересует содержание искомой записи. Мы можем запросить сведения о служащем, указав его имя или номер страхового свидетельства, или узнать имена всех служащих моложе 35 лет, месячный оклад которых составляет 3 000 руб.

Различия между логическим представлением данных и их конкретным содержанием отразились на принципах, которые с самого начала легли в основу организации ЭВМ (ЭВМ, работающих на других принципах, не существует, по крайней мере среди ЭВМ, выпускаемых серийно). Другими словами, способы построения запоминающих устройств и способы обращения к ним центрального процессора ориентированы на числовую обработку.

ЭВМ классической (фоннеймановской) архитектуры состоит из пяти основных функциональных блоков: запоминающего устройства, устройства управления, арифметико-логического устройства (два последних устройства обычно рассматриваются вместе и называются *центральным процессором*), устройства ввода и устройства вывода. Такова архитектура всех современных ЭВМ, даже если они используются, для нечисловой обработки данных.

В ЭВМ фоннеймановской архитектуры обращение к данным организовано так, что для выборки объекта из памяти нужно сначала указать начальный адрес. Иными словами, адрес элемента данных определяется указанием начального значения (начало массива или блока памяти) и смещения конкретного элемента относительно начального адреса. Эти два значения складываются и их сумма обрабатывается некоторым способом, зависящим от механизма доступа.

Если же имена служащих выбираются из файла не по адресу, а по содержимому полей ВОЗРАСТ и ЗАРПЛАТА, то этот способ адресации отличается от способа обращения к

элементам массивов. Такой способ адресации называется *ассоциативным обращением или ассоциативной адресацией*.

Не останавливаясь пока на принципах построения запоминающих устройств, реализующих этот способ адресации, отметим, что независимо от типа используемой памяти адресация по содержанию в программах — реальный факт. Если не учитывать первые программы по обработке файлов, в которых ассоциативный способ адресации проявлялся неявно в логике обработки данных, то можно утверждать, что большинство программ использует ассоциативный способ адресации и зависят от него. Это вполне естественно как с семантической (выбор по содержанию), так и с физической точки зрения (доступ к каждому элементу данных).

Выше было введено понятие ассоциативной адресации. Рассмотрим еще некоторые особенности нечисловой обработки. Как отмечалось, нечисловой обработке обычно подвергаются огромные объемы информации. В различных приложениях над этими данными можно выполнять, например, такие операции:

- а) повысить зарплату всем служащим предприятия;
- б) вычислить банковский процент по счетам всех клиентов;
- в) внести изменения в список товаров, имеющихся на складе;
- г) найти требуемый реферат из всех текстов, хранимых в библиотеке или в библиографической информационно-поисковой системе;
- д) найти описание нужного контракта в файле, содержащем юридические документы;
- е) просмотреть файлы, содержащие описания патентов, и найти патент (если он есть), аналогичный предлагаемому вновь.

В качестве примеров можно привести и другие виды обработки: операции с матрицами, векторами, решение систем уравнений. Это — примеры числовой обработки; в них при-

существует строгая упорядоченность и повторяемость действий. Легко убедиться в том, что основные временные затраты как при числовой, так и при нечисловой обработке легко снизить путем распараллеливания операций в том смысле, что сходные действия над группами данных выполняются одновременно одинаковыми процессорами (по одному или несколько процессоров на группу). Параллелизм — это естественное решение проблемы обработки больших наборов данных с повторяющейся структурой. До сих пор пытались отразить ассоциативную адресацию и параллельную обработку на ЭВМ классической архитектуры, в которой один процессор обращается к памяти по адресу.

Рассмотрим ограничения этой архитектуры.

2.2. Ограничения фоннеймановской архитектуры ПРОЦЕССОР

С точки зрения логики, все запоминающие устройства (ЗУ) хранят данные, подготовленные к обработке. Суть обработки сводится к передаче данных через канал в центральный процессор (ЦП) с одновременной проверкой. Если данные требуемые, они обрабатываются и результат возвращается через канал либо на устройство вывода, либо в соответствующее запоминающее устройство.

В фоннеймановской архитектуре для обработки огромного объема информации мы располагаем всего лишь одним процессором. При этом возникает ситуация, когда миллиарды байтов (символов) информации находятся в состоянии ожидания передачи через канал и обработки на устройстве весьма ограниченной мощности. Такая ситуация для процессора является тупиковой. Для выхода из тупика необходимо внести на этом уровне два изменения в архитектуру:

- а) использовать параллельные процессоры;
- б) приблизить процессоры к данным, а не наоборот, чтобы устранить постоянную передачу данных по каналу.

Таким образом, решениями являются:

- а) параллелизм обработки;
- б) распределенная логика.

ЗАПОМИНАЮЩИЕ УСТРОЙСТВА

В ЗУ фоннеймановской архитектуры обращение происходит по адресу. Для числовой обработки это приемлемо. Но при нечисловой обработке обращение должно осуществляться по содержанию. Поскольку для нечисловой обработки используется та же архитектура, что и для числовой, нужно найти способ организации ассоциативного доступа. Таким способом является эмуляция ассоциативной адресации с помощью основного адресного доступа. При этом создаются специальные таблицы (справочники) для перевода ассоциативного запроса в соответствующий адрес. Таблицы называются *списками ссылок, или индексами*. Они помогают осуществить доступ к конкретной ячейке памяти. Учитывая миллионы и миллиарды байтов, которые нужно обработать, легко представить себе, с какими издержками связаны списки ссылок.

Иными словами, чтобы построить список ссылок, нужно продублировать часть данных и присвоить им соответствующие адреса. Поскольку для очень больших объемов информации ссылки объединяются в достаточно большие файлы, приходится громоздить иерархически связанную систему файлов ссылок. Размер файла ссылок, содержащегося на вершине иерархии, доступен для обработки. И хотя с помощью системы ссылок уменьшается время поиска по сравнению с простым просмотром всех данных, мы проигрываем в другом. В системах, где приходится часто обновлять информацию, необходимо также постоянно реорганизовывать всю систему ссылок.

По мере того как происходит изменение, добавление или удаление записей, ссылки перестают указывать на нужные адреса в памяти, следовательно, их нужно изменять. А ведь нечисловая обработка — это не только просмотр, но и обнов-

ление данных. В результате полная производительность, которая складывается из производительности по выборке и обновлению, резко уменьшается. Итак, для сохранения высокой производительности нужны новые идеи и решения.

2.2. Параллельная обработка

Необходимость параллельной обработки может возникнуть по следующим причинам [6]:

1. Велико время решения данной задачи.
2. Мала пропускная способность системы.
3. Необходимо улучшение использования системы.

Для распараллеливания необходимо соответствующим образом организовать вычисления. Сюда входят:

- составление параллельных программ, то есть отображение в явной форме параллельной обработки с помощью надлежащих конструкций языка, ориентированного на параллельные вычисления;
- автоматическое обнаружение параллелизма. Последовательная программа может быть автоматически проанализирована и выявлена явная или скрытая параллельная обработка. Последняя должна быть преобразована в явную.

Рассмотрим граф, описывающий последовательность процессов большой программы (рисунок 2.1).

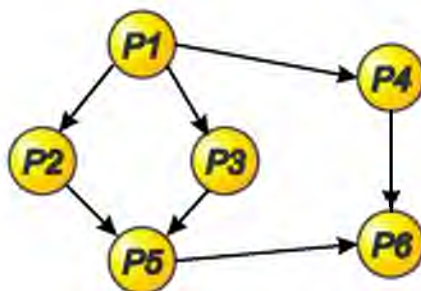


Рисунок 2.1- Граф процессов программы

Из рисунка 2.1 видно, что выполнение процесса P5 не может начаться до завершения процессов P2 и P3 и в свою

очередь выполнение процессов P2 и P3 не может начаться до завершения процесса P1.

В данном случае для выполнения программы достаточно трех процессоров.

Ускорение обработки на многопроцессорной системе определяется отношением времени однопроцессорной обработки к времени многопроцессорной обработки, то есть

$$U = \frac{T_s}{T_m} \quad (2.1)$$

При автоматическом обнаружении параллельных вычислений в последовательной программе выявляют явную и скрытую параллельную обработку. Хотя в обоих случаях требуется анализ программы, различие между этими видами обработки состоит в том, что скрытая параллельная обработка требует некоторой процедуры преобразования последовательной программы, чтобы сделать возможным ее параллельное выполнение. При анализе программы строится граф потока данных. Чтобы обнаружить явную параллельность процессов, анализируются множества входных (считываемых) переменных R (Read) и выходных (записываемых) переменных W (Write) каждого процесса. Два процесса i, j ($i \neq j$) могут выполняться параллельно при следующих условиях:

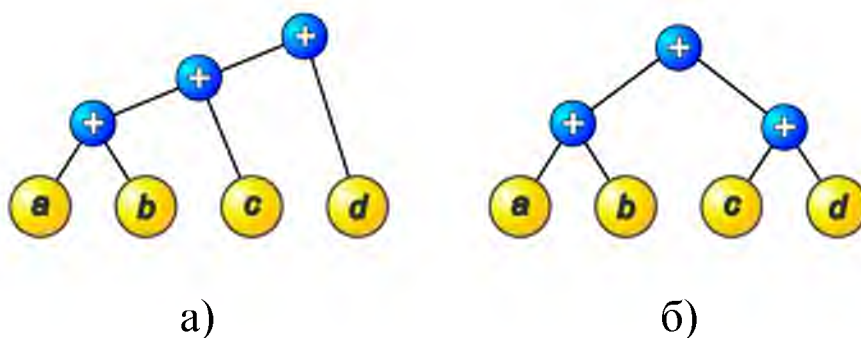
$$\begin{aligned} R_i \cap W_j &= \emptyset \\ W_i \cap R_j &= \emptyset \\ W_i \cap W_j &= \emptyset \end{aligned} \quad (2.2)$$

где \emptyset - пустое множество

Это означает, что входные данные одного процесса не должны модифицироваться другим процессом и никакие два процесса не должны модифицировать общие переменные. Явная параллельная обработка может быть обнаружена среди процессов, удовлетворяющих этим условиям. Для использования скрытой параллельной обработки требуются преобразования программных конструкций, таких как:

- уменьшение высоты деревьев арифметических выражений;
- преобразование линейных рекуррентных соотношений;
- замена операторов;
- преобразование блоков IF и DO к каноническому виду;
- распределение циклов.

Уменьшения высоты дерева арифметического выражения



Замена операторов

Рассмотрим следующий блок операторов присваивания:

$$X = BCD + E$$

$$Y = AX$$

$$Z = X + FG$$

При использовании одного процессора этот блок может быть вычислен за шесть шагов, если один временной шаг отводится для каждой арифметической операции. Путем замены операторов можно получить следующий блок:

$$X = BCD + E$$

$$Y = ABCD + AE$$

$$Z = BCD + E + FG$$

Этот блок может быть вычислен параллельно за три шага с ускорением обработки $U = 6/3 = 2$ при использовании 5 процессоров.

Приведенные примеры преобразований показывают, какое ускорение можно получить, используя параллельную обработку и оптимизацию выражений.

2.3. Конвейерная обработка

Конвейерная обработка улучшает использование ресурсов для заданного набора процессов, каждый из которых применяет эти ресурсы заранее предусмотренным способом. Хорошим примером конвейерной организации является сборочный транспортер на производстве, на котором изделие последовательно проходит все стадии вплоть до готового продукта. Преимущество этого способа состоит в том, что все изделия вдоль своего пути используют один и тот же набор ресурсов, и как только некоторый ресурс освобождается данным изделием, он сразу же может быть использован следующим изделием, не ожидая, пока предыдущее изделие достигнет конца сборочной линии. Если транспортер несет аналогичные, но не тождественные изделия, то это — последовательный конвейер; если же все изделия одинаковы, то это — векторный конвейер.

2.3.1. Последовательные конвейеры

Разработчики компьютеров издавна прибегали к методам проектирования, известным под общим названием "совмещение операций", при котором аппаратура компьютера в любой момент времени выполняет одновременно более одной базовой операции. Этот общий метод включает два понятия: параллелизм и конвейеризацию. При параллелизме совмещение операций достигается путем воспроизведения в нескольких копиях аппаратной структуры. Высокая производительность достигается за счет одновременной работы всех элементов структур, осуществляющих решение различных частей задачи.

Конвейеризация (или конвейерная обработка) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Так обработку любой машинной команды можно разделить на несколько этапов (несколько ступеней), организовав передачу данных от одного этапа к следующему. При этом конвейерную обработку можно использовать для совмещения этапов выполнения разных команд. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

На рисунке 2.3 показано устройство обработки команд простейшего процессора, которое включает четыре ступени:

1. выборка команд из памяти;
2. декодирование;
3. определение адреса и выборка операнда;
4. исполнение.

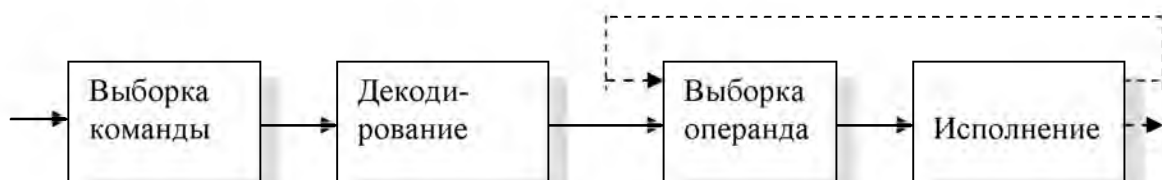


Рисунок 2.3 – Последовательный конвейер

Конвейеризация увеличивает пропускную способность процессора (количество команд, завершающихся в единицу времени), но она не сокращает время выполнения отдельной команды. В действительности, она даже несколько увеличивает время выполнения каждой команды из-за накладных расходов, связанных с хранением промежуточных результатов. Однако увеличение пропускной способности означает, что программа будет выполняться быстрее по сравнению с простой неконвейерной схемой.

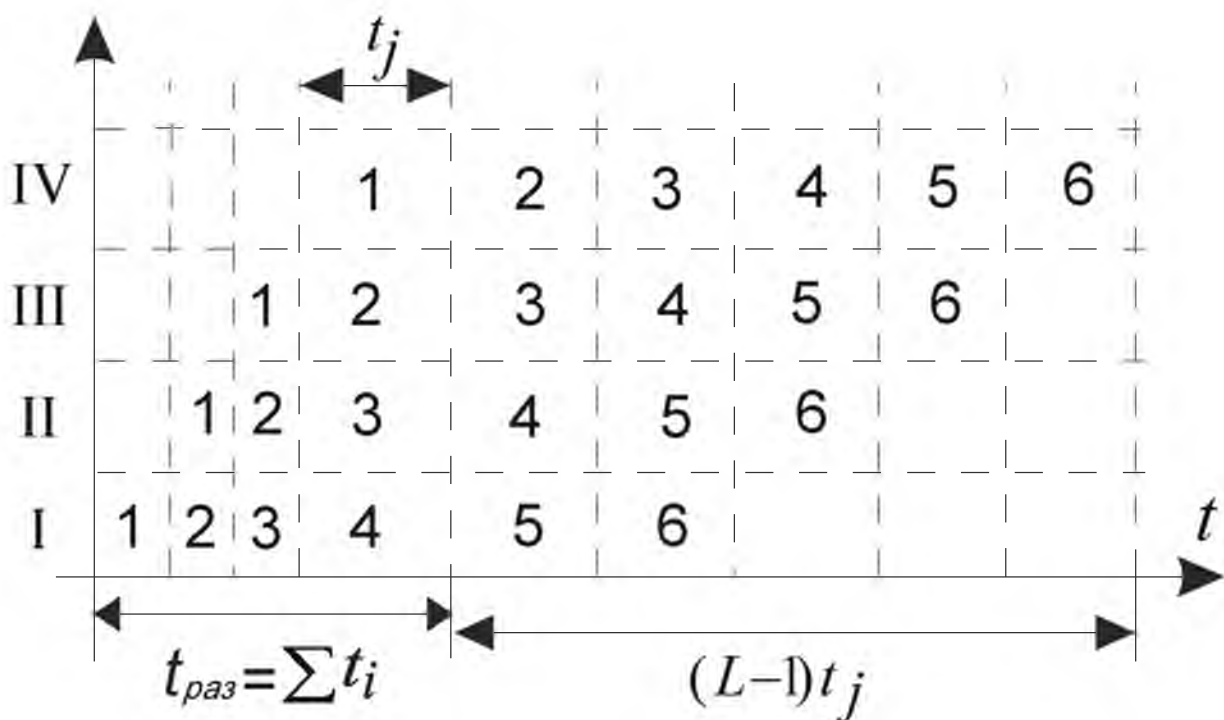


Рисунок 2.4 – Диаграмма работы последовательного конвейера

Ускорение обработки в данном устройстве измеряется отношением времени T_s , необходимом для последовательного выполнения L заданий (без конвейеризации), ко времени T_p выполнения той же обработки на конвейере. Обозначим

через t_i время обработки на i -той ступени, а через t_j - соответствующее время для самой медленной ступени ($j \in i$). Тогда, если L заданий (команд) проходят через конвейер с n ступенями (для рисунков 2.3 и 2.4 – четыре ступени), эффективность конвейера определяется следующим выражением:

$$\frac{T_s}{T_p} = \frac{L \cdot \sum_{i=1}^n t_i}{\sum_{i=1}^n t_i + (L-1) \cdot t_j}$$

для

$$t_i = t_j \rightarrow \frac{n \cdot L}{n + L - 1} \quad (4.3)$$

Как видно из этого анализа, первый результат на выходе конвейера появляется спустя время $t_{раз} = \sum_{i=1}^n t_i$, называемое временем разгона конвейера, а последующие – с интервалами t_j .

Конвейеризация эффективна только тогда, когда загрузка конвейера близка к полной, а скорость подачи новых команд и операндов соответствует максимальной производительности конвейера. Если произойдет задержка, то параллельно будет выполняться меньше операций и суммарная производительность снизится. Такие задержки могут возникать в результате появления конфликтных ситуаций.

Конфликты снижают реальную производительность конвейера, которая могла бы быть достигнута в идеальном случае. Существуют три класса конфликтов:

1. Структурные конфликты, которые возникают из-за конфликтов по ресурсам, когда аппаратные средства не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением.

2. Конфликты по данным, возникающие в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды.

3. Конфликты по управлению, которые возникают при конвейеризации команд переходов и других команд, которые изменяют значение счетчика команд.

Конфликты в конвейере приводят к необходимости приостановки выполнения команд (pipeline stall). Обычно в простейших конвейерах, если приостанавливается какая-либо команда, то все следующие за ней команды также приостанавливаются. Команды, предшествующие приостановленной, могут продолжать выполняться, но во время приостановки не выбирается ни одна новая команда.

2.3.2. Векторные конвейеры

В векторных машинах создается множество функциональных элементов, каждый из которых выполняет определенную операцию с парой операндов, принадлежащих двум разным векторам. Эти пары подаются на функциональное устройство (включающее множество одинаковых функциональных элементов) одновременно и со всеми парами элементов векторов проводят одновременно функциональные преобразования. Для предварительной подготовки преобразуемых векторов используются векторные регистры, на которые собираются подлежащие обработке вектора.

Типичное использование векторного конвейера – это процесс, вырабатывающий по двум исходным векторам **A** и **B** результирующий вектор **C** для арифметической операции $C = A + B$. В этом случае на конвейер поступает множество одинаковых команд.

Векторная команда в этом примере реализуется с помощью специального управляющего вектора. Если n -й разряд управляющего вектора установлен в 1, то операция $C_n = A_n + B_n$ выполняется и C_n записывается в результирующий вектор. Как видно из примера, по мере вычисления адресов пары операндов могут непрерывно вводиться в арифметическое устройство. В такой конвейерной архитектуре требуются регистры или управляющие векторы, хранящие необходимую информацию до тех пор, пока можно начать выполнение ко-

манды. Подготовка данных для векторной обработки может потребовать выполнения нескольких операций загрузки регистров. Необходимое для этого время называют *временем подготовки* t_s . Время от момента декодирования векторной команды до появления на выходе конвейера первого элемента результирующего вектора называют *временем разгона* конвейера t_f . Если длина обрабатываемого векторного поля равна l , а время обработки на самой медленной ступени равно t_b , то общее время выполнения на конвейере векторной команды составляет

$$t_{vp} = t_s + t_f + (l - 1) t_b$$

Для того чтобы выполнить ту же обработку на последовательном конвейере, потребовалось бы использовать его l раз. По сравнению с векторными конвейерами выборка операндов на последовательных конвейерах реализуется менее эффективно. Эквивалентом оценки t_{vp} в последовательном конвейере является величина

$$t_{sq} = t_{fs} + (l - 1) t_{bs}$$

где t_{sq} , t_{fs} и t_{bs} — соответственно время обработки на последовательном конвейере, время его разгона и время обработки на самой медленной его ступени. Сравнивая t_{vp} и t_{sq} , получаем

$$t_s + t_f + (l - 1) t_b \leq t_{fs} + (l - 1) t_{bs} ,$$

если справедливо соотношение

$$l \geq 1 + (t_s + t_f - t_{fs}) / (t_{bs} - t_b)$$

Приведенные характеристики показывают, в каких случаях векторный конвейер имеет преимущества по сравнению с последовательным. В работе [6] установлено, что знаменатель, как правило, составляет около одной десятой числителя, что соответствует значению $l \geq 10$.

2.4. Классификация архитектур вычислительных систем

Многопроцессорные системы (МПС), ориентированные на достижение сверхбольших скоростей работы, содержат десятки или сотни сравнительно простых процессоров с упрощенными блоками управления. Отказ от универсальности применения таких ВС и специализация их на определенном круге задач, допускающих эффективное распараллеливание вычислений, позволяют строить их с регулярной структурой связей между процессорами.

Наиболее удачной считается классификация *Флинна*, которая строится по признаку одинарности или множественности потоков команд и данных.

Однопроцессорная ЭВМ. Структура обыкновенной однопроцессорной ЭВМ (рисунок 2.5) содержит одинарный поток команд и одинарный поток данных (структура ОКОД или SISD).

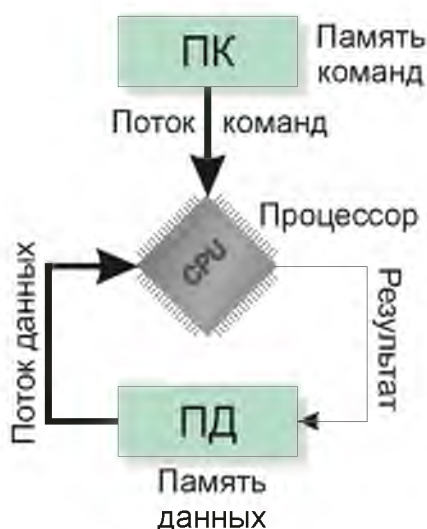


Рисунок 2.5 - Структура SISD

Мультипроцессорная ЭВМ типа ОКМД (или SIMD). В системе образуется несколько потоков данных и один общий поток команд (рис. 2.6).

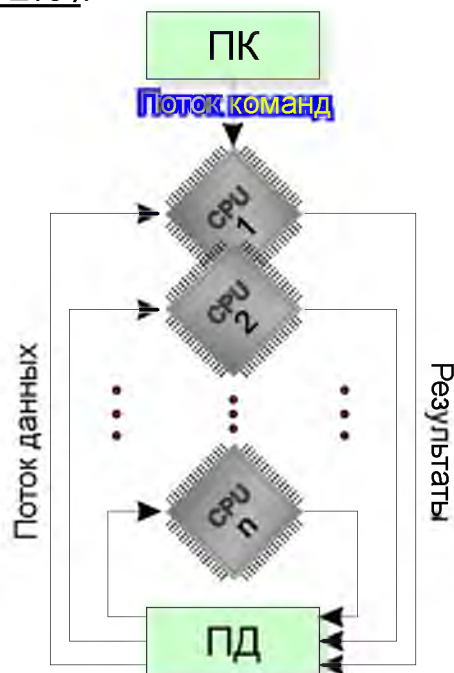


Рисунок 2.6 - Структура типа SIMD

К типу SIMD принадлежит обширный класс новых архитектур. Основная структура машин этого класса состоит из одного (центрального) контроллера, управляющего рядом одинаковых процессоров. Если эти процессоры организованы так, что при выполнении заданных вычислений, инициированных контроллером, они работают параллельно, то система называется *векторным процессором*. Если соединить каждый процессор непосредственно с его памятью и работать в режиме поиска по всему массиву, то получим *ассоциативный процессор*. Параллельная обработка является общим признаком всех машин класса SIMD.

Конвейерная МПС – структура типа МКОД (или MISD) (рисунок 2.7). Система имеет регулярную структуру в виде

цепочки последовательно соединенных процессоров или специальных вычислительных блоков (СВБ), так что информация на выходе одного процессора является входной информацией для следующего в конвейерной цепочке. Процессоры (СВБ) образуют конвейер, на вход которого одинарный поток данных доставляет операнды из памяти. Каждый процессор обрабатывает соответствующую часть задачи, передавая результаты соответствующему процессору, который использует их в качестве исходных данных. Таким образом, решение задач для некоторых исходных данных разворачивается последовательно в конвейерной цепочке. Это обеспечивает подведение к каждому процессору (СВБ) своего потока команд, то есть имеется множественный поток команд.

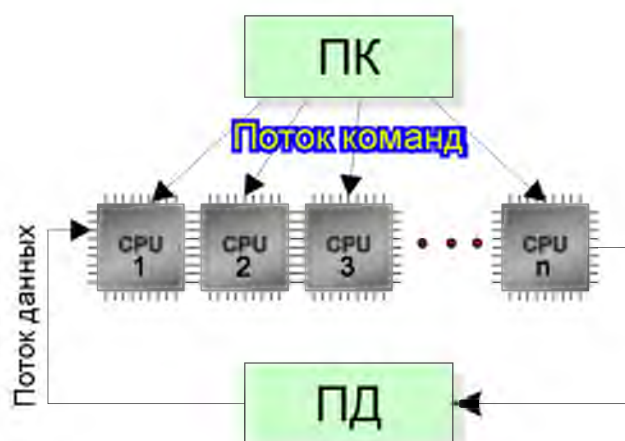


Рисунок 2.7 - Структура типа MISD

Общий случай МПС – структура типа МКМД (или MIMD). На рисунке 2.8 представлен общий случай структуры МПС, в которой существуют несколько потоков данных и несколько потоков команд.

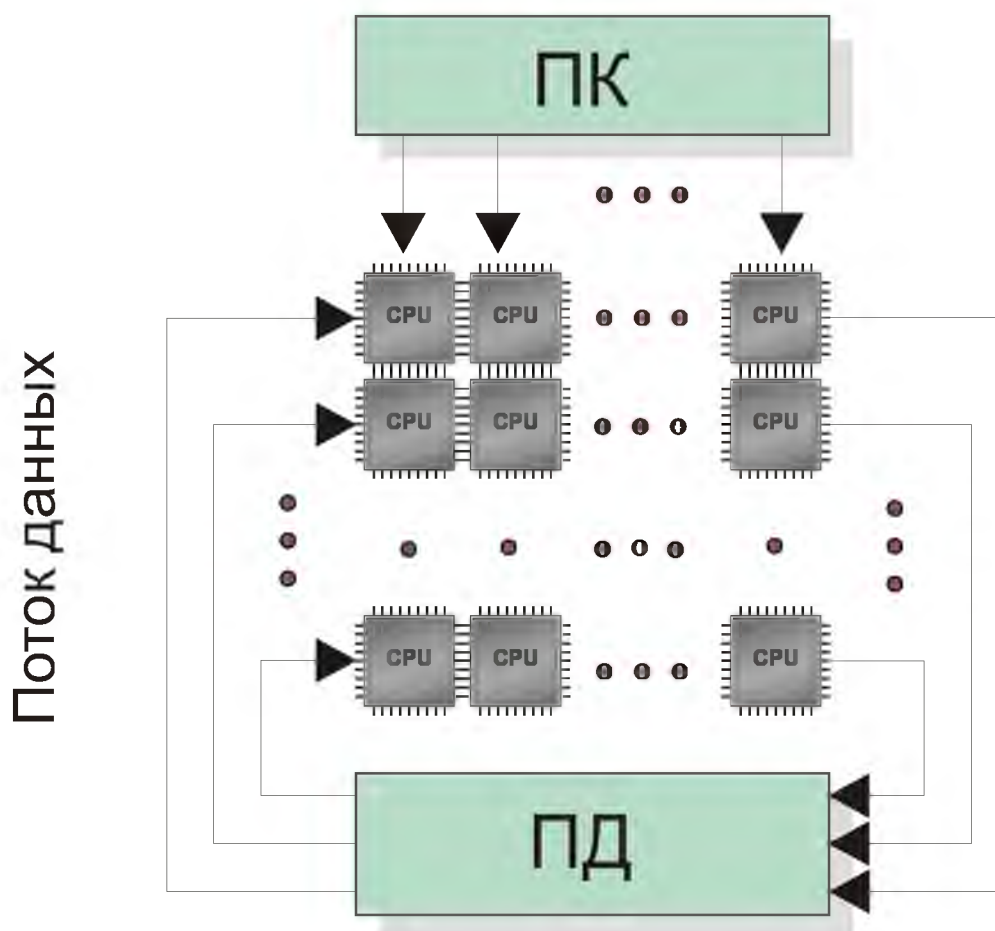


Рисунок 2.8 – Структура типа MIMD

Система содержит некоторое число одинаковых сравнительно простых быстродействующих процессоров, соединенных друг с другом и с памятью данных регулярным образом так, что образуется сетка (матрица), в узлах которой размещаются процессоры (см. рисунок 2.8). Здесь возникает сложная задача распараллеливания алгоритмов решаемых задач для обеспечения загрузки процессоров.

Для мультипроцессорных архитектур требуются соединения двух типов:

- 1) между самими процессорными элементами;
- 2) между процессорными элементами и запоминающими устройствами.

Для соединительных сетей, обеспечивающих связи между процессорными элементами, в литературе установилось название *маршрутных* (routing), а для сетей, связывающих

процессорные элементы с запоминающими устройствами, — название *выравнивающих* (alignment).

В зависимости от возможностей контроллера и процессорных элементов, числа процессоров, организации режима поиска и характеристик маршрутных и выравнивающих сетей архитектуры типа MIMD подразделяются на два типа:

- а) матричные процессоры;
- в) процессорные ансамбли;

Матричные процессоры используются для решения векторных и матричных (сеточных) задач и в общем случае для реализации алгоритмов, соответствующих характеристикам архитектуры MIMD. К этим задачам относятся в основном задачи числовой обработки. Для решения нечисловых задач используются ассоциативные процессоры (архитектура типа SIMD). Процессорные ансамбли применимы как для числовой, так и нечисловой обработки. Конвейерные и векторные процессоры могут быть тоже реализованы на архитектуре типа MIMD, то есть эта архитектура является для мультипроцессорных систем универсальной.

К классу MIMD могут быть отнесены, хотя и не всегда однозначно, следующие конфигурации:

- а) мультипроцессорные системы;
- б) системы с мультиобработкой;
- в) многомашинные системы;
- г) компьютерные сети.

В каждой из перечисленных систем имеется ряд процессоров. Однако в отличие от параллельных матричных систем число процессоров здесь невелико. Более того, вместо процессорных элементов матрицы здесь используется дублирование на уровне систем или подсистем компьютеров. Конфигурация г) предполагает гибкое взаимодействие вычислительных систем, входящих в сеть, а в конфигурациях б) и в) несколько процессоров одновременно выполняют различные фрагменты большой задачи. Одновременность означает здесь распределение функций, т. е., каждый процессор или подси-

система выполняет над каждым из поступающих заданий определенную операцию (некоторое подобие конвейера). В качестве примера можно привести систему обработки данных, состоящую из процессора сортировки, процессора слияния и процессора ввода-вывода и замещения страниц. Каждая из этих подсистем выполняет одни и те же специальные функции для всех поступающих заданий. Здесь используется термин *мультиобработка* в широком смысле для обозначения функционально распределенной одновременной обработки.

Мультипроцессорные системы определяются формальным образом как системы, имеющие два или более процессоров с общим управлением. Это определение можно обобщить, включив также совместное использование ресурсов и взаимодействие между отдельными компонентами системы. Мультипроцессорная система в целом управляется одной операционной системой, и взаимодействие между процессорами возможно на самом низком уровне (вплоть до отдельных единиц данных). В следующем разделе мультипроцессорная организация рассмотрена подробнее.

2.5. Мультипроцессорные системы

2.5.1. Классификация систем параллельной обработки данных

На протяжении всей истории развития вычислительной техники делались попытки найти какую-то общую классификацию, под которую подпадали бы все возможные направления развития компьютерных архитектур. Ни одна из таких классификаций не могла охватить все разнообразие разрабатываемых архитектурных решений и не выдерживала испытания временем. Тем не менее в научный оборот попали и широко используются ряд терминов, которые полезно знать не только разработчикам, но и пользователям компьютеров [11].

Любая вычислительная система (будь то супер-ЭВМ или персональный компьютер) достигает своей наивысшей про-

производительности благодаря использованию высокоскоростных элементов и параллельному выполнению большого числа операций. Именно возможность параллельной работы различных устройств системы (работы с перекрытием) является основой ускорения основных операций.

Параллельные ЭВМ, как указывалось, часто подразделяются по классификации Флинна на машины типа SIMD (Single Instruction Multiple Data - с одним потоком команд при множественном потоке данных) и MIMD (Multiple Instruction Multiple Data - с множественным потоком команд при множественном потоке данных). Как и любая другая, приведенная выше классификация несовершенна: существуют машины прямо в нее не попадающие, имеются также важные признаки, которые в этой классификации не учтены. В частности, к машинам типа SIMD часто относят векторные процессоры, хотя их высокая производительность зависит от другой формы параллелизма - конвейерной организации машины. Многопроцессорные векторные системы, типа Cray Y-MP, состоят из нескольких векторных процессоров и поэтому могут быть названы MSIMD (Multiple SIMD).

Классификация Флинна не делает различия по другим важным для вычислительных моделей характеристикам, например, по уровню "зернистости" параллельных вычислений и методам синхронизации.

Можно выделить четыре основных типа архитектуры систем параллельной обработки:

- 1) Конвейерная и векторная обработка.

Основу конвейерной обработки составляет отдельное выполнение некоторой операции в несколько этапов (за несколько ступеней) с передачей данных одного этапа следующему. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько операций. Конвейеризация эффективна только тогда, когда загрузка конвейера близка к полной, а скорость подачи новых операндов соответствует мак-

симальной производительности конвейера. Если происходит задержка, то параллельно будет выполняться меньше операций и суммарная производительность снизится. Векторные операции обеспечивают идеальную возможность полной загрузки вычислительного конвейера.

При выполнении векторной команды одна и та же операция применяется ко всем элементам вектора (или чаще всего к соответствующим элементам пары векторов). Для настройки конвейера на выполнение конкретной операции может потребоваться некоторое установочное время, однако затем операнды могут поступать в конвейер с максимальной скоростью, допускаемой возможностями памяти. При этом не возникает пауз ни в связи с выборкой новой команды, ни в связи с определением ветви вычислений при условном переходе. Таким образом, главный принцип вычислений на векторной машине состоит в выполнении некоторой элементарной операции или комбинации из нескольких элементарных операций, которые должны повторно применяться к некоторому блоку данных. Таким операциям в исходной программе соответствуют небольшие компактные циклы.

2) Машины типа SIMD. Машины типа SIMD состоят из большого числа идентичных процессорных элементов, имеющих собственную память. Все процессорные элементы в такой машине выполняют одну и ту же программу. Очевидно, что такая машина, составленная из большого числа процессоров, может обеспечить очень высокую производительность только на тех задачах, при решении которых все процессоры могут делать одну и ту же работу. Модель вычислений для машины SIMD очень похожа на модель вычислений для векторного процессора: одиночная операция выполняется над большим блоком данных.

В отличие от ограниченного конвейерного функционирования векторного процессора, матричный процессор (синоним для большинства SIMD-машин) может быть значительно более гибким. Обработываемые элементы таких процессоров

- это универсальные программируемые ЭВМ, так что задача, решаемая параллельно, может быть достаточно сложной и содержать ветвления. Обычное проявление этой вычислительной модели в исходной программе примерно такое же, как и в случае векторных операций: циклы на элементах массива, в которых значения, вырабатываемые на одной итерации цикла, не используются на другой итерации цикла.

Модели вычислений на векторных и матричных ЭВМ настолько схожи, что эти ЭВМ часто обсуждаются как эквивалентные.

3) Машины типа MIMD. Термин "мультипроцессор" покрывает большинство машин типа MIMD и (подобно тому, как термин "матричный процессор" применяется к машинам типа SIMD) часто используется в качестве синонима для машин типа MIMD. В мультипроцессорной системе каждый процессорный элемент (ПЭ) выполняет свою программу достаточно независимо от других процессорных элементов. Процессорные элементы, конечно, должны как-то связываться друг с другом, что делает необходимым более подробную классификацию машин типа MIMD. В мультипроцессорах с общей памятью (сильносвязанных мультипроцессорах) имеется память данных и команд, доступная всем ПЭ. С общей памятью ПЭ связываются с помощью общей шины или сети обмена. В противоположность этому варианту в слабосвязанных многопроцессорных системах (машинах с локальной памятью) вся память делится между процессорными элементами и каждый блок памяти доступен только связанному с ним процессору. Сеть обмена связывает процессорные элементы друг с другом.

Базовой моделью вычислений на MIMD-мультипроцессоре является совокупность независимых процессов, эпизодически обращающихся к разделяемым данным. Существует большое количество вариантов этой модели. На одном конце спектра - модель распределенных вычислений, в которой программа делится на довольно большое число па-

раллельных задач, состоящих из множества подпрограмм. На другом конце спектра - модель потоковых вычислений, в которых каждая операция в программе может рассматриваться как отдельный процесс. Такая операция ждет своих входных данных (операндов), которые должны быть переданы ей другими процессами. По их получении операция выполняется, и полученное значение передается тем процессам, которые в нем нуждаются. В потоковых моделях вычислений с большим и средним уровнем гранулярности, процессы содержат большое число операций и выполняются в потоковой манере.

4) Многопроцессорные машины с SIMD-процессорами.

Многие современные супер-ЭВМ представляют собой многопроцессорные системы, в которых в качестве процессоров используются векторные процессоры или процессоры типа SIMD. Такие машины относятся к машинам класса MSIMD.

Языки программирования и соответствующие компиляторы для машин типа MSIMD обычно обеспечивают языковые конструкции, которые позволяют программисту описывать "крупнозернистый" параллелизм. В пределах каждой задачи компилятор автоматически векторизует подходящие циклы. Машины типа MSIMD, как можно себе представить, дают возможность использовать лучший из этих двух принципов декомпозиции: векторные операции ("мелкозернистый" параллелизм) для тех частей программы, которые подходят для этого, и гибкие возможности MIMD-архитектуры для других частей программы.

Многопроцессорные системы за годы развития вычислительной техники претерпели ряд этапов своего развития. Исторически первой стала осваиваться технология SIMD. Однако в настоящее время наметился устойчивый интерес к архитектурам MIMD. Этот интерес главным образом определяется двумя факторами:

Архитектура MIMD дает большую гибкость: при наличии адекватной поддержки со стороны аппаратных средств и

программного обеспечения MIMD может работать как однопользовательская система, обеспечивая высокопроизводительную обработку данных для одной прикладной задачи, как многопрограммная машина, выполняющая множество задач параллельно, и как некоторая комбинация этих возможностей.

Архитектура MIMD может использовать все преимущества современной микропроцессорной технологии на основе строгого учета соотношения стоимость/производительность. В действительности практически все современные многопроцессорные системы строятся на тех же микропроцессорах, которые можно найти в персональных компьютерах, рабочих станциях и небольших однопроцессорных серверах.

Одной из отличительных особенностей многопроцессорной вычислительной системы является сеть обмена, с помощью которой процессоры соединяются друг с другом или с памятью. Модель обмена настолько важна для многопроцессорной системы, что многие характеристики производительности и другие оценки выражаются отношением времени обработки к времени обмена, соответствующим решаемым задачам. Существуют две основные модели межпроцессорного обмена: одна основана на передаче сообщений, другая - на использовании общей памяти. В многопроцессорной системе с общей памятью один процессор осуществляет запись в конкретную ячейку, а другой процессор производит считывание из этой ячейки памяти. Чтобы обеспечить согласованность данных и синхронизацию процессов, обмен часто реализуется по принципу взаимно исключаящего доступа к общей памяти методом "почтового ящика".

В архитектурах с локальной памятью непосредственное разделение памяти невозможно. Вместо этого процессоры получают доступ к совместно используемым данным посредством передачи сообщений по сети обмена. Эффективность схемы коммуникаций зависит от протоколов обмена, основ-

ных сетей обмена и пропускной способности памяти и каналов обмена.

Часто, и притом необоснованно, в машинах с общей памятью и векторных машинах затраты на обмен не учитываются, так как проблемы обмена в значительной степени скрыты от программиста. Однако накладные расходы на обмен в этих машинах имеются и определяются конфликтами шин, памяти и процессоров. Чем больше процессоров добавляется в систему, тем больше процессов соперничают при использовании одних и тех же данных и шины, что приводит к состоянию насыщения. Модель системы с общей памятью очень удобна для программирования и иногда рассматривается как высокоуровневое средство оценки влияния обмена на работу системы, даже если основная система в действительности реализована с применением локальной памяти и принципа передачи сообщений.

В сетях с коммутацией каналов и в сетях с коммутацией пакетов по мере возрастания требований к обмену следует учитывать возможность перегрузки сети. Здесь межпроцессорный обмен связывает сетевые ресурсы: каналы, процессоры, буферы сообщений. Объем передаваемой информации может быть сокращен за счет тщательной функциональной декомпозиции задачи и тщательного диспетчерирования выполняемых функций.

Таким образом, существующие MIMD-машины распадаются на два основных класса в зависимости от количества объединяемых процессоров, которое определяет и способ организации памяти и методику их межсоединений.

2.5.2. Мультипроцессорные системы с общей памятью

К первой группе относятся машины с общей (разделяемой) основной памятью, объединяющие до нескольких десятков (обычно менее 32) процессоров. Сравнительно небольшое количество процессоров в таких машинах позволяет иметь одну централизованную общую память и объединить процессоры и память с помощью одной шины. При наличии у про-

цессоров кэш-памяти достаточного объема высокопроизводительная шина и общая память могут удовлетворить обращения к памяти, поступающие от нескольких процессоров. Поскольку имеется единственная память с одним и тем же временем доступа, эти машины иногда называются UMA (Uniform Memory Access). Такой способ организации со сравнительно небольшой разделяемой памятью в настоящее время является наиболее популярным. Структура подобной системы представлена на рис. 2.9.



Рисунок 2.9 - Типовая архитектура мультипроцессорной системы с общей памятью

Требования, предъявляемые современными процессорами к полосе пропускания памяти можно существенно сократить путем применения больших многоуровневых кэшей. Тогда, если эти требования снижаются, то несколько процессоров смогут разделять доступ к одной и той же памяти. Начиная с 1980 года, эта идея, подкрепленная широким распространением микропроцессоров, стимулировала многих разработчиков на создание небольших мультипроцессоров, в которых несколько процессоров разделяют одну физическую память, соединенную с ними с помощью разделяемой шины.

Из-за малого размера процессоров и заметного сокращения требуемой полосы пропускания шины, достигнутого за счет возможности реализации достаточно большой кэш-памяти, такие машины стали исключительно эффективными по стоимости. В первых разработках подобного рода машин удавалось разместить весь процессор и кэш на одной плате, которая затем вставлялась в заднюю панель, с помощью которой реализовывалась шинная архитектура. Современные конструкции позволяют разместить до четырех процессоров на одной плате. На рис. 2.9 показана схема именно такой машины.

В этой машине кэши могут содержать как разделяемые, так и частные данные. Частные данные - это данные, которые используются одним процессором, в то время как разделяемые данные используются многими процессорами, по существу обеспечивая обмен между ними. Когда кэшируется элемент частных данных, их значение переносится в кэш для сокращения среднего времени доступа, а также требуемой полосы пропускания. Поскольку никакой другой процессор не использует эти данные, этот процесс идентичен процессу для однопроцессорной машины с кэш-памятью. Если кэшируются разделяемые данные, то разделяемое значение реплицируется и может содержаться в нескольких кэшах. Кроме сокращения задержки доступа и требуемой полосы пропускания такая репликация данных способствует также общему сокращению количества обменов. Однако кэширование разделяемых данных вызывает новую проблему: когерентность (согласованность по данным) кэш-памяти.

Мультипроцессорная когерентность кэш-памяти

Проблема, о которой идет речь, возникает из-за того, что значение элемента данных в памяти, хранящееся в двух разных процессорах, доступно этим процессорам только через их индивидуальные кэши.

Проблема когерентности памяти для мультипроцессоров и устройств ввода-вывода имеет много аспектов. Обычно в

малых мультипроцессорах используется аппаратный механизм, называемый протоколом, позволяющий решить эту проблему. Такие протоколы называются протоколами когерентности кэш-памяти. Существуют два класса таких протоколов:

протоколы на основе справочника (directory based). Информация о состоянии блока физической памяти содержится только в одном месте, называемом справочником (физически справочник может быть распределен по узлам системы);

протоколы наблюдения (snooping). Каждый кэш, который содержит копию данных некоторого блока физической памяти, имеет также соответствующую копию служебной информации о его состоянии. Централизованная система записей отсутствует. Обычно кэши расположены на общей (разделяемой) шине и контроллеры всех кэшей наблюдают за шиной (просматривают ее) для определения того, не содержат ли она копию соответствующего блока.

В мультипроцессорных системах, использующих микропроцессоры с кэш-памятью, подсоединенные к централизованной общей памяти, протоколы наблюдения приобрели популярность, поскольку для опроса состояния кэшей они могут использовать заранее существующее физическое соединение - шину памяти.

Неформально, проблема когерентности памяти состоит в необходимости гарантировать, что любое считывание элемента данных возвращает последнее по времени записанное в него значение. Это определение не совсем корректно, поскольку невозможно требовать, чтобы операция считывания мгновенно видела значение, записанное в этот элемент данных некоторым другим процессором. Если, например, операция записи на одном процессоре предшествует операции чтения той же ячейки на другом процессоре в пределах очень короткого интервала времени, то невозможно гарантировать, что чтение вернет записанное значение данных, поскольку в этот момент времени записываемые данные могут даже не

покинуть модифицировавший их процессор. Вопрос о том, когда точно записываемое значение должно быть доступно процессору, выполняющему чтение, определяется выбранной моделью согласованного (непротиворечивого) состояния памяти и связан с реализацией синхронизации параллельных вычислений. Поэтому с целью упрощения предположим, что мы требуем только, чтобы записанное операцией записи значение было доступно операции чтения, возникшей немного позже записи и что операции записи данного процессора всегда видны в порядке их выполнения.

С этим простым определением согласованного состояния памяти мы можем гарантировать когерентность путем обеспечения двух свойств:

операция чтения ячейки памяти одним процессором, которая следует за операцией записи в ту же ячейку памяти другим процессором получит записанное значение, если операции чтения и записи достаточно отделены друг от друга по времени;

операции записи в одну и ту же ячейку памяти выполняются строго последовательно (иногда говорят, что они сериализованы): это означает, что две подряд идущие операции записи в одну и ту же ячейку памяти будут наблюдаться другими процессорами именно в том порядке, в котором они появляются в программе процессора, выполняющего эти операции записи.

Первое свойство очевидно связано с определением когерентного (согласованного) состояния памяти: если бы процессор всегда бы считывал только старое значение данных, мы сказали бы, что память некогерентна.

Необходимость строго последовательного выполнения операций записи является более тонким, но также очень важным свойством. Представим себе, что строго последовательное выполнение операций записи не соблюдается. Тогда процессор P1 может записать данные в ячейку, а затем в эту ячейку выполнит запись процессор P2. Строго последова-

тельное выполнение операций записи гарантирует два важных следствия для этой последовательности операций записи. Во-первых, оно гарантирует, что каждый процессор в машине в некоторый момент времени будет наблюдать запись, выполняемую процессором P2. Если последовательность операций записи не соблюдается, то может возникнуть ситуация, когда какой-нибудь процессор будет наблюдать сначала операцию записи процессора P2, а затем операцию записи процессора P1, и будет хранить это записанное P1 значение неограниченно долго. Более тонкая проблема возникает с поддержанием разумной модели порядка выполнения программ и когерентности памяти для пользователя: представьте, что третий процессор постоянно читает ту же самую ячейку памяти, в которую записывают процессоры P1 и P2; он должен наблюдать сначала значение, записанное P1, а затем значение, записанное P2. Возможно он никогда не сможет увидеть значения, записанного P1, поскольку запись от P2 возникла раньше чтения. Если он даже видит значение, записанное P1, он должен видеть значение, записанное P2, при последующем чтении. Подобным образом любой другой процессор, который может наблюдать за значениями, записываемыми как P1, так и P2, должен наблюдать идентичное поведение. Простейший способ добиться таких свойств заключается в строгом соблюдении порядка операций записи, чтобы все записи в одну и ту же ячейку могли наблюдаться в том же самом порядке. Это свойство называется последовательным выполнением (сериализацией) операций записи (write serialization). Вопрос о том, когда процессор должен увидеть значение, записанное другим процессором достаточно сложен и имеет заметное воздействие на производительность, особенно в больших машинах.

2.5.2. Мультипроцессорные системы с локальной памятью и многомашинные системы

Вторую группу машин составляют крупномасштабные системы с распределенной памятью. Для того чтобы поддер-

живать большое количество процессоров приходится распределять основную память между ними, в противном случае полосы пропускания памяти просто может не хватить для удовлетворения запросов, поступающих от очень большого числа процессоров. Естественно при таком подходе также требуется реализовать связь процессоров между собой. На рис. 2.10 показана структура такой системы.

С ростом числа процессоров просто невозможно обойти необходимость реализации модели распределенной памяти с высокоскоростной сетью для связи процессоров. С быстрым ростом производительности процессоров и связанным с этим ужесточением требования увеличения полосы пропускания памяти, масштаб систем (т.е. число процессоров в системе), для которых требуется организация распределенной памяти, уменьшается, так же как и уменьшается число процессоров, которые удастся поддерживать на одной разделяемой шине и общей памяти.

Распределение памяти между отдельными узлами системы имеет два главных преимущества. Во-первых, это эффективный с точки зрения стоимости способ увеличения полосы пропускания памяти, поскольку большинство обращений могут выполняться параллельно к локальной памяти в каждом узле. Во-вторых, это уменьшает задержку обращения (время доступа) к локальной памяти. Эти два преимущества еще больше сокращают количество систем, для которых архитектура с распределенной памятью имеет смысл.

Обычно устройства ввода/вывода, также как и память, распределяются по узлам и в действительности узлы могут состоять из небольшого числа (2-8) процессоров, соединенных между собой другим способом. Хотя такая кластеризация нескольких процессоров с памятью и сетевой интерфейс могут быть достаточно полезными с точки зрения эффективности в стоимостном выражении, это не очень существенно для понимания того, как такая машина работает, поэтому мы пока остановимся на системах с одним процессором на узел. Ос-

новная разница в архитектуре, которую следует выделить в машинах с распределенной памятью заключается в том, как осуществляется связь и какова логическая модель памяти.

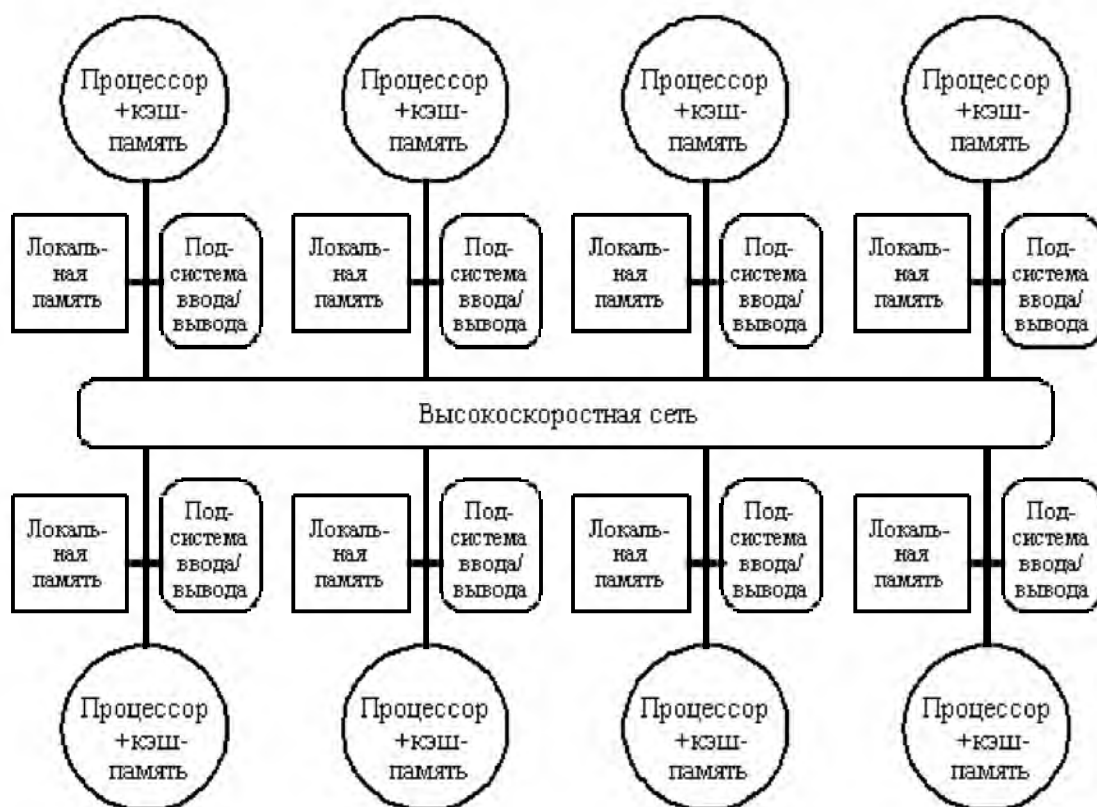


Рисунок 2.10 - Типовая архитектура мультимикропроцессорной системы с распределенной памятью.

Существуют два различных способа построения крупномасштабных систем с распределенной памятью. Простейший способ заключается в том, чтобы исключить аппаратные механизмы, обеспечивающие когерентность кэш-памяти, и сосредоточить внимание на создании масштабируемой системы памяти. Несколько компаний разработали такого типа машины. Наиболее известным примером такой системы является компьютер T3D компании Cray Research. В этих машинах память распределяется между узлами (процессорными элементами) и все узлы соединяются между собой посредством того или иного типа сети. Доступ к памяти может быть локальным или удаленным. Специальные контроллеры, размещаемые в узлах сети, могут на основе анализа адреса об-

ращения принять решение о том, находятся ли требуемые данные в локальной памяти данного узла, или размещаются в памяти удаленного узла. В последнем случае контроллеру удаленной памяти посылается сообщение для обращения к требуемым данным.

Чтобы обойти проблемы когерентности, разделяемые (общие) данные не кэшируются. Конечно, с помощью программного обеспечения можно реализовать некоторую схему кэширования разделяемых данных путем их копирования из общего адресного пространства в локальную память конкретного узла. В этом случае когерентностью памяти также будет управлять программное обеспечение. Преимуществом такого подхода является практически минимальная необходимая поддержка со стороны аппаратуры, хотя наличие, например, таких возможностей как блочное (групповое) копирование данных было бы весьма полезным. Недостатком такой организации является то, что механизмы программной поддержки когерентности подобного рода кэш-памяти компилятором весьма ограничены. Существующая в настоящее время методика в основном подходит для программ с хорошо структурированным параллелизмом на уровне программного цикла.

Машины с архитектурой, подобной Cray T3D, называют процессорами (машинами) с массовым параллелизмом (MPP - Massively Parallel Processor). К машинам с массовым параллелизмом предъявляются взаимно исключающие требования. Чем больше объем устройства, тем большее число процессоров можно расположить в нем, тем длиннее каналы передачи управления и данных, а значит и меньше тактовая частота. Происшедшее возрастание нормы массовости для больших машин до 512 и даже 64К процессоров обусловлено не ростом размеров машины, а повышением степени интеграции схем, позволившей за последние годы резко повысить плотность размещения элементов в устройствах. Топология сети обмена между процессорами в такого рода системах может быть различной.

Для построения крупномасштабных систем альтернативой рассмотренному в предыдущем разделе протоколу наблюдения может служить протокол на основе справочника, который отслеживает состояние кэш-кэшей. Такой подход предполагает, что логически единый справочник хранит состояние каждого блока памяти, который может кэшироваться. В справочнике обычно содержится информация о том, в каких кэшах имеются копии данного блока, модифицировался ли данный блок и т.д. В существующих реализациях этого направления справочник размещается рядом с памятью. Имеются также протоколы, в которых часть информации размещается в кэш-памяти. Положительной стороной хранения всей информации в едином справочнике является простота протокола, связанная с тем, что вся необходимая информация сосредоточена в одном месте. Недостатком такого рода справочников является его размер, который пропорционален общему объему памяти, а не размеру кэш-памяти. Это не составляет проблемы для машин, состоящих, например, из нескольких сотен процессоров, поскольку связанные с реализацией такого справочника накладные расходы можно преодолеть. Но для машин большего размера необходима методика, позволяющая эффективно масштабировать структуру справочника.

2.6. Базовые архитектуры суперкомпьютеров

2.6.1. Система Illiac 4

Эта система создана по совместному проекту Иллинойского университета и корпорации “Берроуз”. В конструкции системы Illiac 4 нашли отражение идеи системы Solomon с некоторыми небольшими изменениями. 256 процессорных элементов (ПЭ) системы Illiac 4 были подразделены на четыре квадранта по 64 ПЭ, причем каждый квадрант должен был управляться собственным устройством управления (УУ). Весь комплекс из четырех квадрантов должен был управлять-

ся универсальным компьютером (УК). Фактически был построен лишь один квадрант [4], [6].

Как видно из рисунка 2.11, матричная система Illiac 4 работает как специализированная приставка к универсальному компьютеру Burroughs 6500, который осуществляет интерфейс с пользователями, вычислительной сетью и архивной лазерной памятью. В системе В 6500, с которой связаны пользователи, содержатся основная операционная система, компиляторы, служебные и прикладные программы, подсистема ввода-вывода. Программы пользователей компилируются в код ассемблера матричной системы и пересылаются с необходимыми данными в ПЭ.

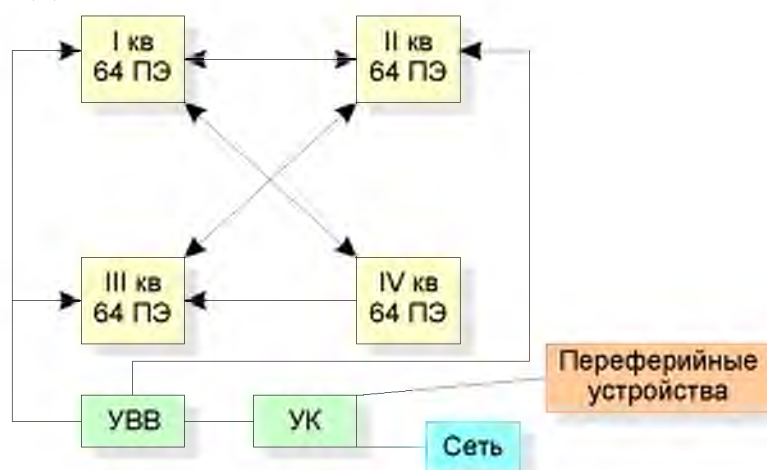


Рисунок 2.11

Программы ПЭ выполняются параллельно по всей матрице. Работа матрицы осуществляется под управлением УУ, а результаты передаются пользователям через компьютер В6500.

Составляющие квадрант 64 ПЭ физически разбиваются на восемь групп по восемь ПЭ в каждой. Каждая группа связана с файловыми дисками двумя 64-разрядными каналами, так что общее число каналов по всей матрице равно 16. Этим объясняется параллельная связь шириной 1024 бит. Емкость диска эквивалентна 30 полным загрузкам процессорной матрицы Illiac 4.

Система Illiac 4 является быстродействующей и ее связь с универсальным компьютером осуществляется быстрее, чем в обычном последовательном канале интерфейса.

2.6.2. MPP - процессор фирмы Goodyear

Подобные процессоры называют машинами с массовым параллелизмом (Massively Parallel Processor).

Он был создан фирмой Goodyear Aerospace по контракту с NASA для обработки изображений, полученных со спутников. Структура этого процессора изображена на рисунке 2.12 [2].

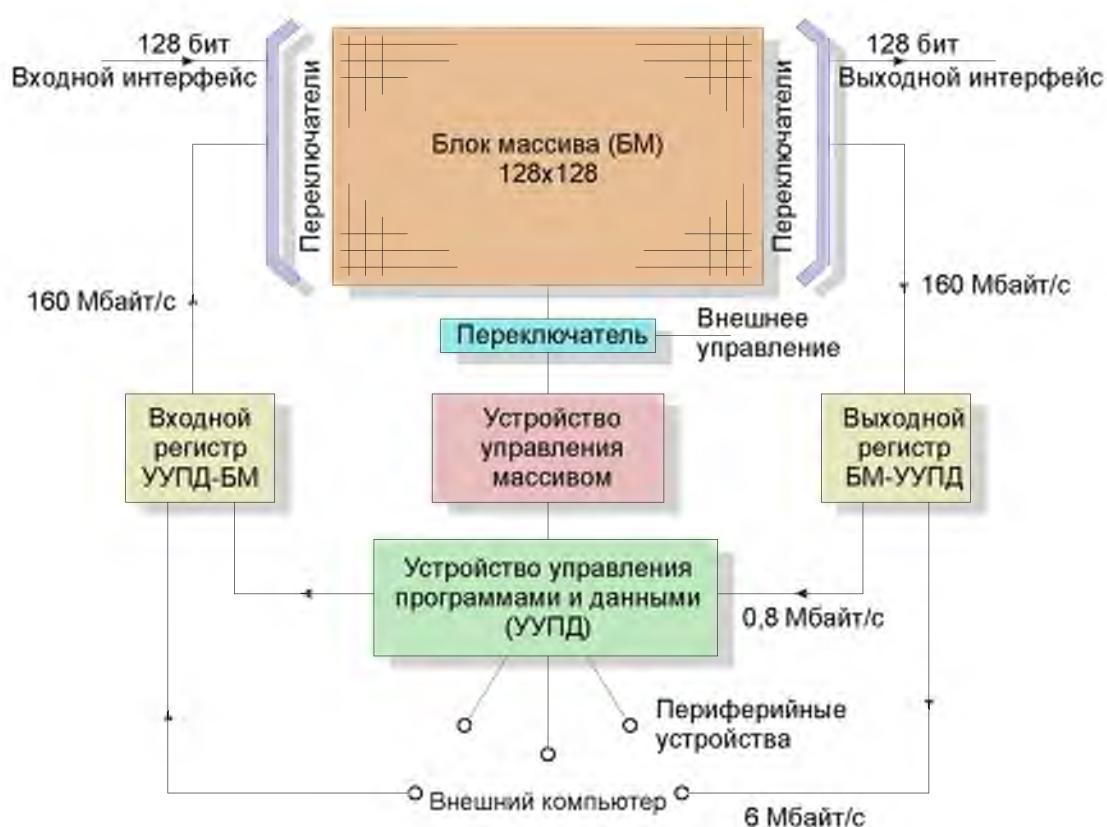


Рисунок 2.12-Структура процессора с массовым параллелизмом (MPP)

Весь блок массива состоит из 128x128 ПЭ с четырьмя избыточными столбцами, введенными для увеличения надежности. Общий массив делится на 33 подмассива по 128

х 4 ПЭ. Каждая группа снабжена каналом включения-выключения, ведущим к управляющему устройству массива УУМ.

Производительность процессора может составлять 10^9 - 10^{10} операций в секунду. Это возможно из-за большого числа ПЭ.

2.6.3. Векторные конвейерные процессоры

Архитектура машины Cray-1

На рисунке 2.13 приведена блок-схема машины Cray-1.

Всего имеется 12 функциональных устройств [2, 11]. Они разделены на группы в зависимости от типа выполняемых ими операций и адресуемых регистров. Выполняют вычисление адресов, логические, скалярные и векторные операции над целыми числами, операции с плавающей запятой над скалярами и векторами. Большинство простых операций центрального процессора выполняется за один такт, который составляет 12.5 нс.

Производительность однопроцессорной машины Cray-1 составляет примерно $100 \cdot 10^6$ операций с плавающей точкой в секунду (100 Мфлопс). В более новых и более мощных моделях машинах системы Cray, выпускаемых фирмой Silicon Grafics, использовались от 2 до 8 процессоров и производительность составляла от 1 до 2,5 Гфлопс.

С 1998 года фирма Silicon Grafics приступила к выпуску векторной масштабируемой (то есть с изменяемой конфигурацией) суперЭВМ серии Cray SV1. В суперкомпьютере используются векторные процессоры, пиковая производительность которых достигает 4 Гфлопс. Пиковая производительность заключенного в один корпус узла составляет 32 Гфлопс. Общее число процессоров системы может быть больше 1000.

Согласно планам, объявленным компанией, вслед за первым поколением Cray SV1 появится векторный масшта-

бируемый суперкомпьютер второго поколения, пиковая производительность которого составит десятки терафлопс.

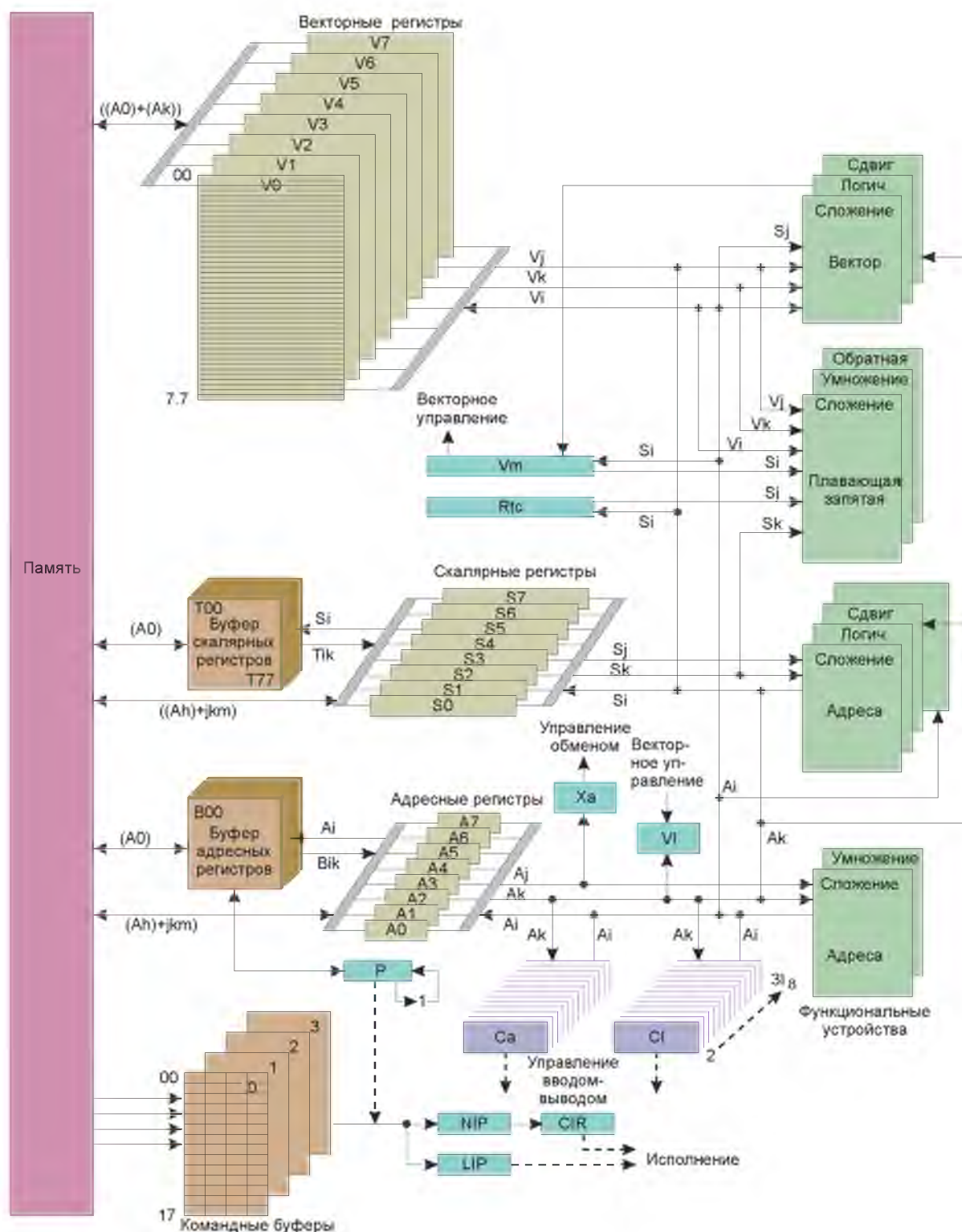


Рисунок 2.13-Функциональные блоки и регистры машины CRAY-1

Многопроцессорные векторные суперкомпьютеры Cray SV1

Рынок современных многопроцессорных векторных (векторно-параллельных) суперкомпьютеров достаточно узок. На этом поле «играют» только избранные — прежде всего это американская компания SGI, производящая компьютеры с маркой Cray, и японские NEC и Fujitsu [11]. Все прогнозы об уходе компьютеров этой архитектуры с рынка пока не оправдались. Векторные процессоры NEC SX-5 и Cray SV-1 по-прежнему значительно опережают по производительности вычислений с плавающей запятой самые быстродействующие микропроцессоры. И все же рискну предположить, что доля векторных систем в парке установленных суперкомпьютеров будет постепенно уменьшаться. В то же время сохраняются серьезные задачи, в которых применение векторных систем весьма эффективно.

Cray SV1, последнее поколение векторных систем SGI/Cray, о котором было объявлено еще в 1998 году, можно считать наследником мини-суперкомпьютеров Cray J90, представленных в 1994 году.



Рисунок 2.14 – Внешний вид суперЭВМ Cray SV1

Внешний вид суперЭВМ Cray SV1 представлен на рисунке 2.14, а общая архитектура - на рисунке 2.15. Ее анализ стоит начать с главного козыря SV1 — ее центрального про-

цессора. «Базовые» процессоры SV1 имеют по два векторных конвейера, каждый из которых может выполнять две операции с плавающей запятой за один такт. При тактовой частоте в 250 МГц это дает производительность 1 GFLOPS, что в пять раз выше, чем у 100-мегагерцевого Cray J90, но по-прежнему уступает RISC-микропроцессорам. В 1999 году появилась версия Cray SV1 с 300-мегагерцевым процессором с производительностью уже 1,2 GFLOPS, но и это ниже, чем у некоторых RISC-микропроцессоров.

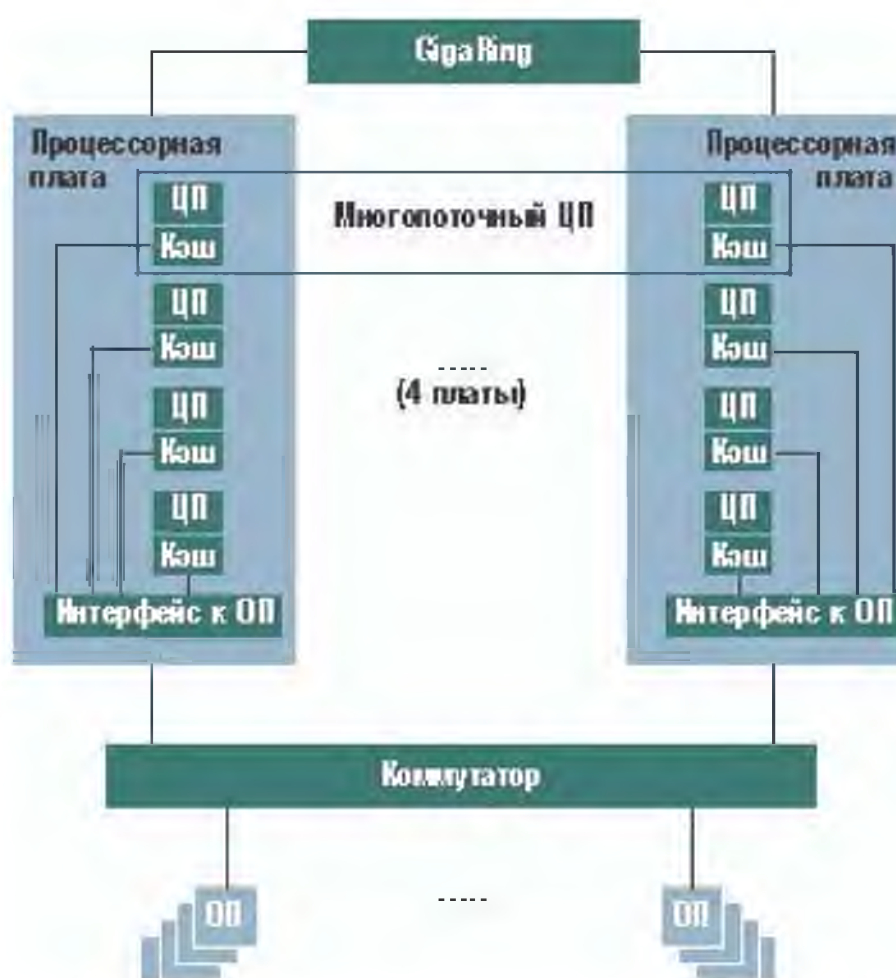


Рисунок 2.15 – Архитектура Cray SV1-1A с коммутатором 4x4

Однако, из обычных процессоров Cray SV1 можно сконфигурировать так называемые многопоточные процессо-

ры путем объединения четырех стандартных двухконвейерных процессоров в один. При этом все векторные регистры отдельных процессоров становятся общими, число конвейеров становится равным восьми, а пиковая производительность — 4/4,8 GFLOPS для процессоров на 250/300 МГц соответственно.

Вот это уже превышает возможности RISC-микропроцессоров, не перешагнувших границу 2 GFLOPS. Многопоточные процессоры побеждают по производительности также и Cray T90 (1,8 GFLOPS), хотя сильно отстают от японских конкурентов — NEC SX-5 (8 GFLOPS, см. таблицу) и Fujitsu VPP5000 (9,6 GFLOPS). Оценка производительности процессора SV1 на тестах Linpack (536 MFLOPS при $N = 100$, 996 MFLOPS при $N = 1000$) подтверждает это соотношение. Не стоит забывать и об отношении стоимость/производительность, которое для Cray SV1 очень хорошее.

Известно, что одним из основных преимуществ векторно-параллельных суперкомпьютеров перед многопроцессорными системами на базе «серийных» RISC-микропроцессоров является гораздо более высокая пропускная способность памяти. Резкое повышение производительности центрального процессора при сохранении пропускной способности оперативной памяти в Cray SV1 делает эту характеристику потенциально узким местом.

Известно, что одним из основных преимуществ векторно-параллельных суперкомпьютеров перед многопроцессорными системами на базе «серийных» RISC-микропроцессоров является гораздо более высокая пропускная способность памяти. Резкое повышение производительности центрального процессора при сохранении пропускной способности оперативной памяти в Cray SV1 делает эту характеристику потенциально узким местом. IDC в своем отчете отмечает, что SGI следует работать над увеличением пропускной способности памяти.

Длина строки кэша равна 8 байт, что эквивалентно одному элементу векторного регистра. Если в Cray J90 восемь векторных регистров по 64 элемента в каждом, то есть всего 512 элементов, то емкость векторного кэша SV1 — уже 32К элементов. Однако для оптимизации использования векторного кэша существующие векторные приложения для компьютеров Cray предыдущих поколений должны быть переработаны.

На каждой процессорной плате размещается по четыре процессора, которые разделяют общий интерфейс к оперативной памяти. На процессорной плате имеется также интерфейс к каналу ввода/вывода GigaRing с пропускной способностью 1 Гбайт/с. Шкаф SV1 содержит систему с симметрично-многопроцессорной архитектурой, основанной на коммутаторе 8x8 с конструктивом backplane. Его восемь портов используются для подсоединения восьми процессорных плат, а другие восемь портов — для подсоединения оперативной памяти. Соответственно SMP-система SV1 может масштабироваться до 32 двухконвейерных центральных процессоров. Общее число каналов GigaRing на систему может при этом достигать восьми.

При конфигурировании многопоточных центральных процессоров каждый такой восьмиконвейерный процессор объединяет в себе четыре двухконвейерных — по одному из четырех разных процессорных плат (см. рисунок). Поэтому многопоточный ЦП обменивается данными с ОП сразу через четыре интерфейса ОП, и ПС ОП для такого ЦП составляет 25,6 Гбайт/с.

В обычном шкафу SV1 можно сконфигурировать до шести многопоточных процессоров плюс восемь обычных двухконвейерных. Возможность создания различных смешанных конфигураций, отличающихся числом процессоров разных типов, позволяет точно подстроить конфигурацию SV1 под конкретные особенности задач пользователя. Однако суммарная пиковая производительность SV1 при такой перекон-

фигурации не меняется (до 32/38 GFLOPS при 250/300 МГц соответственно).

Оперативная память SV1 построена по технологии DRAM и имеет емкость от 2 до 32 Гбайт. Она может включать 256, 512 и 1024 банка; соответственно максимальный уровень расслоения (чередования адресов) оперативной памяти, используемого для повышения ее пропускной способности, равен 1024.

Кроме SMP-систем SV1, SGI предлагает также кластеры на их основе. Основным «строительным блоком» кластера являются четырехузловые системы. До восьми таких блоков можно объединить, доведя общее число процессоров до 1024 (192 многопоточных). Это позволяет иметь суперкомпьютерную систему с емкостью оперативной памятью свыше 1 Тбайт и производительностью свыше 1 TFLOPS (отметим, что самые мощные массивно-параллельные системы, в том числе от самой SGI, этот рубеж превзошли еще раньше). Такие высокие характеристики масштабирования векторных многопроцессорных систем — одна из наиболее привлекательных черт Cray SV1, на что указывает и сама аббревиатура SV (Scalable Vector).

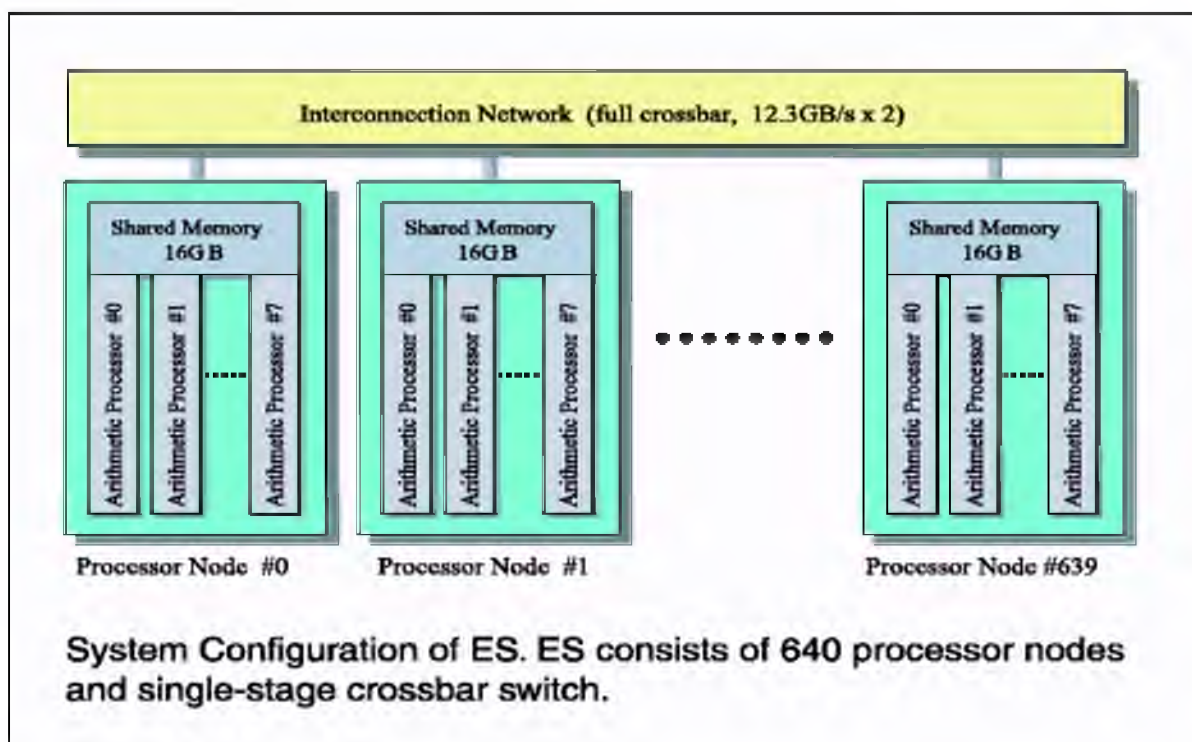
Топология кластера может быть различной, но SGI предполагает, что для «маленьких» конфигураций будут чаще использоваться соединения «точка-точка», а для больших — двухмерный тор. Вообще-то Cray SV1 использует воздушное охлаждение, но в кластерных конфигурациях возможно и водяное охлаждение.

Стоимость минимальной конфигурации SV1-1A (восемь двухконвейерных центральных процессоров) составляет 500 тыс. долл., а такой же, но с возможностью расширения SV1-1 — 1 млн. долл. Очевидно, что соотношение стоимость/производительность для таких систем выглядит весьма привлекательно.

Суперкомпьютер Earth Simulator (ES)

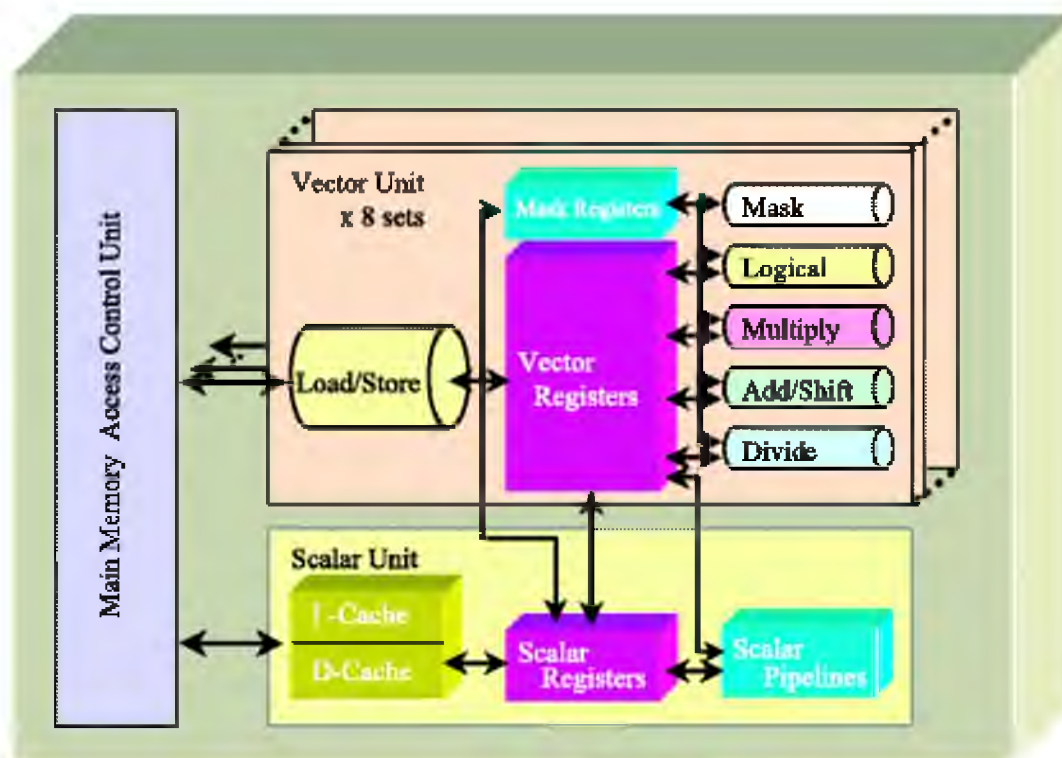
Earth Simulator (ES) - мультипроцессорная компьютерная

система с распределенной памятью, состоящая из 640 процессорных узлов (PNs), связанных через 640x640 одноступенчатых перекрестных переключателей, образующих внутреннюю высокоскоростную коммутирующую сеть. Каждый PN - система с общей памятью, состоящая из 8 арифметических процессоров векторного типа (APs), оперативной памяти на 16 Гбайт (MS), блока управления удаленного доступа (RCU) и процессора ввода - вывода. Пиковая производительность каждого арифметического процессора (AP) – 8 Gflops. ES в целом, таким образом, состоит из 5120 APs с 10 ТВ оперативной памяти и пиковой производительностью - 40 Tflor/s. На конец 2002 года это - самая высокая производительность в мире.



Каждый арифметический процессор (AP) состоит из суперскалярного модуля (SU), векторного модуля (VU) и модуля управления доступом к оперативной памяти, выполненной на отдельной большой интегральной схеме. Арифметический процессор работает на тактовой частоте 500 MHz. Каждый суперскалярный модуль снабжен суперскалярным процессором с кэшем команд на 64 КБ, кэшем данных на 64 КБ и 128 уни-

версальными скалярными регистрами. Здесь выполняются такие операции, как организация ветвлений, предвыборка данных и восстановление сбойных команд. Векторный модуль состоит из 8 наборов, содержащих по 72 векторных регистра, каждый из которых может иметь до 256 векторных элементов, и по шести типов векторных конвейеров:



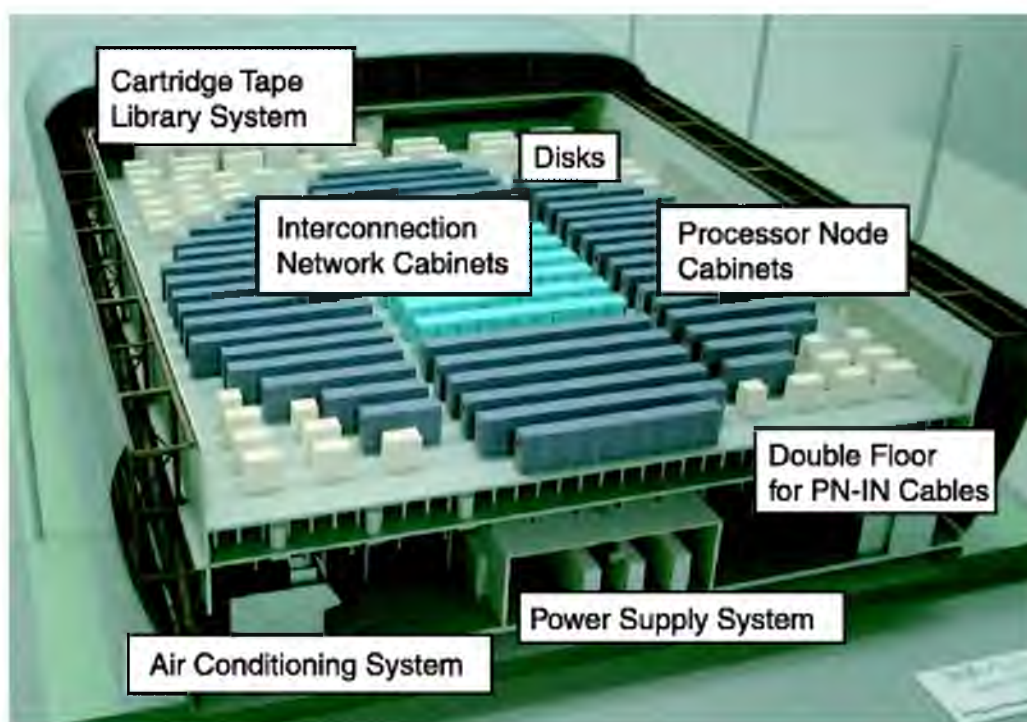
The construction of arithmetic processor(AP). One LSI Chip implementation for these functional units including Vector Processing with 8GFlop/s peak performance.

сложение/сдвиг, умножение, деление, логические операции, маскировка и загрузка/сохранение. VU и SU поддерживают формат данных с плавающей точкой стандарта IEEE 754.

Блок управления удаленного доступа (RCU) непосредственно связан с внутренней высокоскоростной коммутирующей сетью и управляет двунаправленной передачей данных между узлами со скоростью 12.3GB/s в каждом направлении. Таким образом, полная полоса пропускания сети – 8 TB/s. Не-

сколько режимов передачи данных, включая доступ к трехмерным подмассивам и косвенные режимы доступа, реализованы аппаратными средствами. В операции доступа к данным подмассива, данные перемещаются от одного PN к другому в единой аппаратной операции за относительно короткое время.

Система оперативной памяти (MS) разделена на 2048 банков, и последовательность номеров банков соответствует увеличению адресов слотов в памяти. Поэтому пиковая производительность была получена при обращении к непрерывным данным, расположенным в памяти в порядке возрастания адресов.



A model of the ES system in the gym-like building. The building is 50m x 65m x 17m and has two stories; it includes a seismic isolation system.

Earth Simulator (ES) был разработан как национальный проект Национальным агентством космического развития Японии, Научно-исследовательским институтом атомной энергии Японии и Центром морской науки и технологии Японии. Установ-

лен и введен в эксплуатацию к концу февраля 2002 в Центре моделирования Земли в Йокогаме.

Суперкомпьютер размещен на площади 50x65 m.

2.7. Ассоциативный процессор

Отсутствие ассоциативных возможностей в фоннеймановской архитектуре приводит к семантическому разрыву между ассоциативной моделью обращения пользователя и обращением по адресу на физическом уровне. Другими словами, необходимы дополнительные средства (т. е. система ссылок — индексов) для определения местонахождения искомых данных [2].

Рассмотрим пример из реальной жизни. Предположим, что в некотором кинотеатре нужно «присматриваться к зрителям». Для этого кассир должен вписать в план зрительного зала (который представляет собой ряды квадратов, соответствующие местам) имя владельца билета (а не только лица, приобретшего билет). При такой системе учета в кинотеатре появится возможность в случае острой необходимости обнаружить требуемого человека во время просмотра фильма. Достаточно обратиться к администратору, сообщить ему имя нужного человека и в ответ получить номер ряда и места, на котором сидит разыскиваемый зритель. Если это так, то можно (даже в темноте) добраться до указанного места и привлечь внимание нужного зрителя. А что делать, если зритель поменялся с соседом или сел на другое свободное место? Если некоторые зрители опоздали к началу сеанса и сели на первые попавшиеся свободные места? Информация, которую имеет администратор, окажется неверной, что приведет к ошибкам поиска. +Администратор располагает системой адресуемой памяти. Если же у нас была бы система ассоциативного доступа, мы просто объявили бы по радио имя разыскиваемого зрителя.

Может случиться, что в зале находятся несколько человек с нужным именем (либо один или даже никого); аналогичная ситуация может иметь место и при поиске ячеек памяти, содержимое которых соответствует поисковому признаку. Следовательно, требуется уметь обрабатывать множественные отклики (многозначный ответ) системы.

Этот пример хорошо поясняет принцип работы ассоциативной памяти, но с точки зрения затрат на реализацию эта аналогия не годится. Установка системы с громкоговорителем стоит, вероятно, не больше чем годовой оклад администратора. А ассоциативная память стоит гораздо больше, чем адресуемая. Тем не менее в обоих случаях устранение ссылок и средств их поддержания значительно упрощает систему.

Таким образом, в ассоциативной памяти параллельный поиск идет сразу по большой группе ячеек и в итоге поисковому признаку может удовлетворять содержимое нескольких ячеек. Возможности выполнения различных видов поиска и разнообразие структур ассоциативной памяти объясняют, почему для обозначения этого устройства существует так много синонимов: *память с параллельным поиском, запоминающее устройство с многозначным ответом, память с распределенной логикой, логико-запоминающее устройство* и т.д. Очевидно, что параллельный поиск и другие свойства данной системы делают ее «интеллектуальной» по сравнению с адресной памятью.

Базовая структура пословно организованного ассоциативного процессора

Для того чтобы понять основное различие между ассоциативным доступом и доступом по адресу, попробуем при адресном доступе найти нужную информацию, отказавшись от словаря ссылок. В простейшем случае придется просмотреть всю память, т. е. ассоциативность в этом случае обеспечивается полным перебором. Для того чтобы выполнить эту работу за приемлемое время, ее обязательно нужно распараллелить. Мы можем эмулировать этот процесс на машине

фоннеймановской архитектуры с помощью цикла, обращаясь последовательно к ячейкам памяти и сравнивая их содержимое. Не будь ограничений по времени, можно было бы обойтись без ассоциативной памяти. Если в примере с кинотеатром строго придерживаться инструкции и всякий раз обращаться к таблице поиска, то не останется времени смотреть фильм. Этот пример объясняет, почему в современных системах обработки данных все операции по обновлению информации осуществляются ночью, а днем пользуются неизменными данными: иначе времени на обработку совсем не оставалось бы.

Итак, параллельность, одновременность в работе — неотъемлемое свойство ассоциативной памяти, что можно пояснить следующим образом:

- имеется большое число элементов памяти (в примере с кинотеатром — большое число мест);
- поиск идет по всем элементам сразу (в примере — путем объявления по радио).

Ассоциативным процессором (АП) называют ассоциативное запоминающее устройство, дополненное логикой и микропрограммным управлением [2].

На рисунке 2.15 изображена структура пословно организованного АП. В основе архитектур ассоциативных процессоров с пословной организацией лежит параллелизм на уровне слов, и в большинстве конфигураций обработка слов выполняется последовательно по разрядам. Множество слов образует ассоциативный массив или ассоциативное запоминающее устройство (АЗУ) пословно организованного ассоциативного процессора. Соответственно имеется по одному процессорному элементу (ПЭ) на каждое слово, так что весь разрядный срез может обрабатываться параллельно.

Базовая структура пословно организованного АП содержит следующие подсистемы:

- массив ассоциативной памяти;

- регистр компаранда (компаранд – признак, по которому ведется поиск в АЗУ; он записывается в регистр компарандов, затем пересылается в регистр маски);
- регистр маски (выделяет указанный признак, остальные маскирует);
- регистры хранения ответов;
- регистры (буфер) ввода-вывода АЗУ;
- маска (буфер) вывода слов (последовательно по словам выводит содержимое АЗУ);
- соединительная сеть (используется для логической комбинации полей в данной физической среде);
- контроллер (управляет рассмотренными подсистемами).

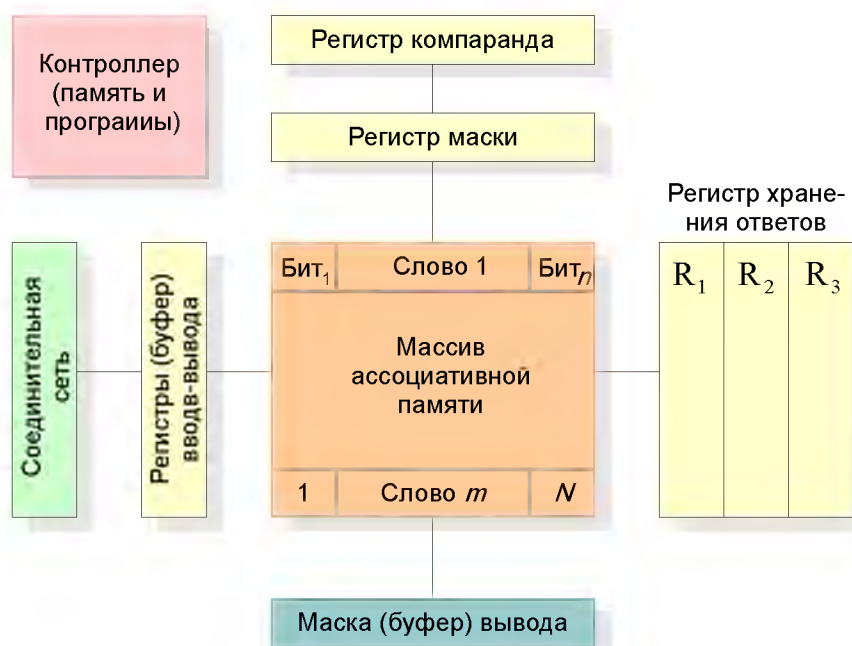


Рисунок 2.16 - Базовая структура пословно организованного ассоциативного процессора

Массив ассоциативной памяти. Этот массив содержит m n -разрядных слов. При нечисловой обработке такая запоминающая матрица размером $m \times n$ является удобной средой для отображения двумерных логических структур, таких, как матрицы, файлы и отношения. Каждая строка матрицы, каждая запись файла или каждый кортеж отношения соответ-

ствуется слову ассоциативного массива. Хотя в таких структурах и допускается переполнение с переносом в следующие слова, типичная характеристика этих систем состоит в том, что каждое слово массива предназначается для одной строки, записи или кортежа соответствующих структур. Это объясняется несколькими причинами. Одной из них являются физические ограничения реализации, связанные с длиной слов. Начиная с некоторой длины слова, в аппаратуре возникает проблема шумов. Другая причина состоит в том, что периферийная аппаратура поиска массива ассоциативной памяти может выполнять одновременно лишь один поиск, если не вводить дорогостоящего дублирующего оборудования.

Другая характеристика этих процессоров состоит в том, что, поскольку операции выполняются в них пословно и поразрядно, они требуют высокой скорости чтения и записи. Это в свою очередь обуславливает выбор в качестве памяти статических запоминающих устройств вместо более медленных, хотя и более емких динамических. С этим типом памяти совместимы слова длиной 256, 1024 и даже 4096 бит (одна ячейка машины баз данных RAR.3 содержит основную память емкостью 2 Мбайт; ячейка в данном случае соответствует отдельному слову АП). При горизонтальных размерах такого порядка требуется вертикальный размер (или параллелизм по словам), соответствующий кардинальному числу (т.е. числу записей файла или кортежей отношения) хранимой в них структуры. Чтобы выполнить соединение двух отношений, каждое из которых содержит 100000 кортежей, потребовался бы 218 АП, имеющий АЗУ емкостью 200000 слов. Но до сих пор ни разу не строились и даже не предлагались машины такого размера [6].

Регистр компаранда. В массиве АЗУ могут выполняться различные комбинации операций поиска. Это означает, что можно сравнивать некоторое внешнее значение со всеми словами, хранимыми в АЗУ; можно сравнивать друг с другом два поля во всех словах; наконец, можно сравнивать некото-

рое поле выбранного слова с полями всех других слов. Но во всех этих случаях компаранд должен быть сначала введен в регистр компаранда, чтобы выполнить массовое сравнение (или сравнение типа SIMD), в котором одно значение сравнивается параллельно со множеством значений. При параллельном сравнении двух полей во всех словах имеет место параллелизм другого типа, и самый эффективный способ этого сравнения состоит в выполнении логической операции или операции вычитания между двумя полями в каждом слове, так что конечный результат является искомым результатом сравнения. Таким образом, хотя данные, т.е. компаранды, в каждом слове различны, операция выполняется параллельно по словам и последовательно по разрядным срезам. Длина регистра компаранда соответствует длине слова АЗУ. В типичном режиме сравнения позиции разрядов компаранда, маски и слов АЗУ соответствуют друг другу. В общем случае в таком соответствии нет необходимости, поскольку можно направить некоторое поле компаранда для сравнения с полем, расположенным в другой области АЗУ..

Регистр маски. Регистр маски можно рассматривать как простой фильтр или селектор разрядов регистра компаранда. С помощью регистра маски указывается поле регистра компаранда, которое служит действительным компарандом. Все разряды, не входящие в указанный компаранд, маскируются (т.е. соответствующие разряды регистра маски устанавливаются в 0), в то время как разряды, соответствующие полю компаранда, устанавливаются в 1.

Регистры хранения ответов. Обычно каждый регистр хранения ответов представляет собой множество триггеров, образующее одноразрядный двоичный вектор вдоль всего массива АЗУ. Один из этих регистров называется *регистром меток (тегов)* или *отклика* и служит для индикации или хранения результатов операций над массивом. Выделенное слово АЗУ, т.е. слово, отмеченное единицей в регистре меток, обычно индицирует успешное окончание поиска или хранит

результат логической операции (например, перенос в текущем шаге операции сложения). Второй регистр памяти ответов используется как временная рабочая область (или сверхоперативная память) при обработке данных или реализации логических функций. Третий регистр памяти ответов служит для выбора слов. Он указывает для каждого слова АЗУ, участвует ли оно в подлежащей выполнению операции. Если, например, в массиве АЗУ хранятся файлы А и В и мы хотим выполнить некоторую операцию над одним из этих файлов, то необходимо запретить обработку всех слов другого файла путем установки в 0 всех разрядов регистра выбора слов, соответствующих второму файлу.

Регистры (или буфер) ввода-вывода АЗУ. Эти регистры играют роль буфера при передаче данных в АЗУ или из него. Манипулирование данными может относиться к одному слову или распространяться на весь массив. Последнее происходит обычно при загрузке и разгрузке АЗУ. В этом случае самое важное — выполнить эту операцию как можно быстрее. В самом деле, поскольку объем АЗУ относительно небольшой по сравнению с объемом подлежащих обработке данных, такие операции потребуется выполнять очень часто, и при этом массив АЗУ не должен долго бездействовать в ожидании ввода и вывода данных. Поэтому используются специальные интерфейсы ввода-вывода, такие, как диски с фиксированными головками и барабаны, в которых каждая головка (т.е. тракт) предназначена для одного слова АЗУ. Если массив АЗУ состоит из 256 слов, такой интерфейс обеспечивает параллельный канал шириной 256 бит, имеющий полосу пропускания в $256/8 = 32$ раза большую, чем полоса стандартного последовательного по байтам канала ввода-вывода.

Маска (буфер) вывода слов. Этот буфер используется для последовательного по словам вывода содержимого АЗУ. Возможность маскирования позволяет выбирать отдельные поля по тому же принципу, что и в регистре маски.

Соединительная сеть. В рассматриваемом классе ассоциативных процессоров соединительная сеть используется для логических комбинаций полей в данной физической среде, чтобы можно было выполнить требуемые операции. Иначе говоря, соединительная сеть эмулирует на двумерном массиве АЗУ более сложные виды межпроцессорных соединений. Например, при обработке изображений или геометрических фигур возможна работа с сеточными структурами, где необходимо соединение каждого элемента с четырьмя ближайшими соседями. Может потребоваться также сеть «идеальной тасовки» для сортировки, вычисления полиномов и т. д.

Контроллер. Массив АЗУ и все рассмотренные выше схемы управляются контроллером. Запрос пользователя транслируется в основной машине в базовые операции АП и посылается в контроллер. Контроллер имеет память для программ, реализующих базовые операции, и различные регистры, логику управления шиной, механизмы прерываний и арифметико-логическое устройство для вычисления адресов и сдвига информации. Каждая конкретная система имеет собственную конструкцию контроллера.

2.8. Концепция ВС с управлением потоком данных

Существуют трудности, связанные с решением проблемы автоматизации параллельного программирования, необходимой в целях эффективного использования для широкого круга задач матричных ВС. Поэтому актуальны исследования новых методов построения высокопроизводительных ВС, одними из которых являются ВС с управлением потоком данных, или потоковые ВС [11].

В системах с управлением потоками данных предполагается наличие большого числа специализированных операционных блоков для определения видов операций (сложения, умножения и т.п., отдельных для разных типов данных). Данные снабжаются указателями типа данных (тегами), на осно-

вании которых по мере готовности данных к обработке они загружаются в соответствующие свободные операционные блоки. При достаточном количестве операционных блоков может быть достигнут высокий уровень распараллеливания вычислительного процесса.

Во всех традиционных машинах и вычислительных системах порядок выполнения операций над данными при решении задачи строго детерминирован, он однозначно определяется последовательностью команд программы.

Принципиальное отличие потоковых машин состоит в том, что команды выполняются не в порядке следования команд в тексте программы, а по мере готовности их операндов. Как только будут вычислены операнды команды, она может захватывать свободное операционное устройство и выполнять предписанную ей операцию. В этом случае последовательность, в которой выполняются команды, уже не является детерминированной.

Идея процессора, управляемого потоком данных, отображена на рисунке 2.17.

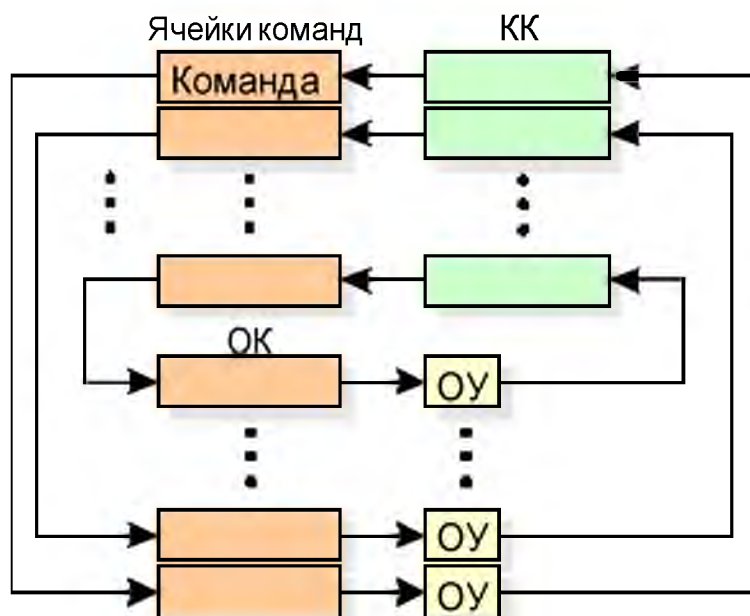


Рисунок 2.17 - Процессор с управлением потоком данных

«Потоковая программа» размещается в массиве ячеек команд. Команда наряду с кодом операции содержит поля, куда заносятся готовые операнды, и поле, содержащее адреса команд, в которые должен быть направлен в качестве операнда результат операции. Кроме того, каждой команде поставлен в соответствие двухразрядный тег (располагаемый в управляющем устройстве), разряды которого устанавливаются в “1” при занесении в тело команды соответствующих операндов. В состоянии тега “11” (оба операнда готовы) инициируется запрос к операционному коммутатору (ОК) на передачу готовой команды в соответствующее коду операции операционное устройство. Результат выполнения команды над ее непосредственно адресуемыми операндами направляется через командный коммутатор (КК) согласно указанным в команде адресам в ячейки команд и помещается в их поля операндов. Далее указанная процедура циклически повторяется, причем управление этим процессом полностью децентрализовано и не нуждается в счетчике команд.

2.9.Закон Амдала и его следствия

Предположим, что в программе доля операций, которые нужно выполнять последовательно, равна f , где $0 \leq f \leq 1$ (при этом доля понимается не по статическому числу строк кода, а по числу операций в процессе выполнения). Крайние случаи в значениях f соответствуют полностью параллельным ($f=0$) и полностью последовательным ($f=1$) программам. Для того, чтобы оценить, какое ускорение S может быть получено на компьютере из p процессоров при данном значении f , можно воспользоваться законом Амдала [11]:

$$S \leq 1/(f+(1-f)/p)$$

Если 9/10 программы исполняется параллельно, а 1/10 по-прежнему последовательно, то ускорения более, чем в 10 раз получить в принципе невозможно вне зависимости от ка-

чества реализации параллельной части кода и числа используемых процессоров (ясно, что 10 получается только в том случае, когда время исполнения параллельной части равно 0).

Посмотрим на проблему с другой стороны: а какую же часть кода надо ускорить (а значит и предварительно исследовать), чтобы получить заданное ускорение? Ответ можно найти в следствии из закона Амдала: для того чтобы ускорить выполнение программы в q раз необходимо ускорить не менее, чем в q раз не менее, чем $(1-1/q)$ -ю часть программы. Следовательно, если есть желание ускорить программу в 100 раз по сравнению с ее последовательным вариантом, то необходимо получить не меньшее ускорение не менее, чем на 99% кода, что почти всегда составляет значительную часть программы!

Отсюда первый вывод - прежде, чем основательно переделывать код для перехода на параллельный компьютер (а любой суперкомпьютер, в частности, является таковым) надо основательно подумать. Если, после оценки заложенного в программу алгоритма, выяснилось, что доля последовательных операций велика, то на значительное ускорение рассчитывать явно не приходится и нужно думать о замене отдельных компонент алгоритма.

Словом, заставить параллельную вычислительную систему или супер-ЭВМ работать с максимальной эффективностью на конкретной программе - задача не из простых, поскольку *необходимо тщательное согласование структуры программ и алгоритмов с особенностями архитектуры параллельных вычислительных систем.*

2.10. Наиболее известные современные многопроцессорные компьютеры

Согласно данным [11], в настоящее время к наиболее известным мультипроцессорным системам относятся нижеследующие.

IBM RS/6000 SP

Производитель	International Business Machines (IBM), подразделение RS/6000.
Класс архитектуры	Масштабируемая массивно-параллельная вычислительная система (MPP).
Узлы	Узлы имеют архитектуру рабочих станций RS/6000. Существуют несколько типов SP-узлов, которые комплектуются различными процессорами: PowerPC 604e/332MHz, POWER3/200 и 222 MHz (более ранние системы комплектовались процессорами POWER2). High-узлы на базе POWER3 включают до 8 процессоров и до 16 GB памяти.
Масштабируемость	До 512 узлов. Возможно совмещение узлов различных типов. Узлы устанавливаются в стойки (до 16 узлов в каждой).
Коммутатор	Узлы связаны между собой высокопроизводительных коммутатором (IBM high-performance switch), который имеет многостадийную структуру и работает с коммутацией пакетов.
Системное ПО	ОС AIX (устанавливается на каждом узле), система пакетной обработки LoadLeveler, параллельная файловая система GPFS, параллельная СУБД INFORMIX-OnLine XPS. Параллельные приложения исполняются под управлением Parallel Operating Environment (POE).
Средства программирования	Оптимизированная реализация интерфейса MPI, библиотеки параллельных математических подпрограмм - ESSL, OSL.

HP 9000 (Exemplar)

Производитель	Hewlett-Packard, подразделение высокопроизводительных систем.
Класс	Многопроцессорные сервера с общей памятью (SMP).
Предшественники	SMP/NUMA-системы Convex SPP-1200, SPP-1600, SPP-2000.
Модификации	В настоящее время доступны несколько "классов" систем семейства HP 9000: сервера начального уровня (D,K-class), среднего уровня (N-class) и наиболее мощные системы (V-class).
Процессоры	64-битные процессоры с архитектурой PA-RISC 2.0 (PA-8200, PA-8500).
Число процессоров	N-class - до 8 процессоров. V-class - до 32 процессоров. В дальнейшем ожидается увеличение числа процессоров до 64, а затем до 128.
Масштабируемость	SCA-конфигурации (Scalable Computing Architecture) - до 4 узлов V-class, т.е. до 128 процессоров.
Системное ПО	Устанавливается операционная система HP-UX (совместима на уровне двоичного кода с ОС SPP-UX компьютеров Convex SPP).
Средства программирования	HP MPI - реализация MPI 1.2, оптимизированная к архитектуре Exemplar. Распараллеливающие компиляторы Fortran/C, математическая библиотека HP MLIB. CXperf - средство анализа производительности программ.

Cray T3E

Производитель	Silicon Graphics
Класс архитектуры	Масштабируемая массивно-параллельная система, состоит из процессорных элементов (PE).

Предшественники	Cray T3D
Модификации	В настоящее время существуют две модификации: T3E-900 и T3E-1200.
Процессорный элемент	PE состоит из процессора, блока памяти и устройства сопряжения с сетью. Используются процессоры Alpha 21164 (EV5) с тактовой частотой 450 MHz (T3E-900) и 600 MHz (T3E-1000), пиковая производительность которых составляет 900 и 1200 MFLOP/sec соответственно. Процессорный элемент располагает своей локальной памятью (DRAM) объемом от 256MB до 2GB.
Число процессоров	Системы T3E масштабируются до 2048 PE.
Коммутатор	Процессорные элементы связаны высокопроизводительной сетью с топологией трехмерного тора и двунаправленными каналами. Скорость обменов по сети достигает 480MB/sec в каждом направлении.
Системное ПО	Используется операционная система UNICOS/mk.
Средства программирования	Поддерживается явное параллельное программирование с помощью пакета Message Passing Toolkit (MPT) - реализации интерфейсов передачи сообщений MPI, MPI-2 и PVM, библиотека Shmem. Для Фортран-программ возможно также неявное распараллеливание в моделях CRAFT и HPF. Среда разработки включает также набор визуальных средств для анализа и отладки параллельных программ.

Cray T90

Производитель	Silicon Graphics, Cray Research.
Класс архитектуры	Многопроцессорная векторная система (несколько векторных процессоров работают на общей памяти).
Предшественники	CRAY Y-MP C90, CRAY X-MP.
Модели	Серия T90 включает модели T94, T916 и T932.
Процессор	Системы серии T90 базируются на векторно-конвейерном процессоре Cray Research с пиковой производительностью 2GFlop/s.
Число процессоров	Система T932 может включать до 32 векторных процессоров (до 4-х в модели T94, до 16 модели T916), обеспечивая пиковую производительность более 60GFlop/s.
Масштабируемость	Возможно объединение нескольких T90 в MPP-системы.
Память	Система T932 содержит от 1GB до 8GB (до 1 GB в модели T94 и до 4GB в модели T916) оперативной памяти и обеспечивает скорость обменов с памятью до 800MB/sec.
Системное ПО	Используется операционная система UNICOS.

Cray SV1

Производитель	Silicon Graphics
Класс архитектуры	Масштабируемый векторный суперкомпьютер.
Процессор	Используются 8-конвейерные векторные процессоры MSP (Multi-Streaming Processor) с пиковой производительностью 4.8 GFLOP/sec; каждый MSP может быть подразделен на 4 стандартных 2-конвейерных процессора с пиковой производительностью 1.2 GFLOP/sec. Тактовая частота процессоров - 250MHz.
Число процессоров	Процессоры объединяются в SMP-узлы, каждый из которых может содержать 6 MSP и 8 стандартных процессоров. Система (кластер) может содержать до 32 таких узлов.
Память	SMP-узел может содержать от 2 до 16GB памяти. Система может содержать до 1TB памяти. Вся память глобально адресуема (архитектура DSM).
Системное ПО	Используется операционная система UNICOS.
Средства программирования	Поставляется векторизирующий и распараллеливающий компилятор CF90. Поддерживается также явное параллельное программирование с использованием интерфейсов MPI, OpenMP или Shmem.

Cray Origin2000

Производитель	Silicon Graphics
Класс архитектуры	Модульная система с общей памятью (cc-NUMA).

Процессор	64-разрядные RISC-процессоры MIPS R10000, R12000/300MHz
Модуль	Основной компонент системы - модуль Origin, включающий от 2 до 8 процессоров MIPS R10000 и до 16GB оперативной памяти.
Масштабируемость	Поставляются системы Origin2000, содержащие до 256 процессоров (т.е. до 512 модулей). Вся память системы (до 256GB) глобально адресуема, аппаратно поддерживается когерентность кэшей.
Коммутатор	Модули системы соединены с помощью сети CrayLink, построенной на маршрутизаторах MetaRouter.
Системное ПО	Используется операционная система SGI IRIX.
Средства программирования	Поставляется распараллеливающий компилятор Cray Fortran 90. Поддерживается стандарт OpenMP.

Onyx2 InfiniteReality2

Производитель	Silicon Graphics
Класс архитектуры	Многопроцессорная система визуализации; по аппаратной архитектуре очень похожа на Origin2000.
Число процессоров	Система может включать до 128 процессоров MIPS R10000.
Визуализация	Графические возможности системы обеспечивают специальные устройства трех типов: геометрические (векторные) процессоры, растровые процессоры, генераторы аналоговых сигналов. Система может быть оборудована 16 независимыми каналами графического вывода (visualization pipelines). На аппаратном уровне поддерживается графический интерфейс OpenGL.
Системное ПО	Используется операционная система SGI IRIX.

Sun HPC 10000 (StarFire)

Производитель	Sun Microsystems, серия Sun HPC.
Класс архитектуры	Многопроцессорный SMP-сервер.
Процессор	UltraSPARC II/336MHz
Число процессоров	Система StarFire объединяет от 16 до 64 процессоров.
Память	Система включает от 2GB до 64GB памяти.
Системное ПО	ОС Solaris, ПО распределения ресурсов Load Sharing Facility (LSF).
Средства разработки	Поставляется пакет поддержки параллельных приложений Sun HPC 2.0, включающий такие средства как HPF, MPI, PVM, PFS (параллельная файловая система), Prism (визуальная среда разработки), S ³ L (библиотека математических подпрограмм), и др.

Sun Fire 15K

Производитель	Sun Microsystems.
Класс архитектуры	Многопроцессорный SMP-сервер.
Процессор	UltraSPARC III/900MHz
Число процессоров	Система Sun Fire 15K объединяет до 106 процессоров.
Память	Система включает до 576GB памяти.
Системное ПО	ОС Solaris 8.

NEC SX-5

Производитель	NEC, серия SX.
Класс архитектуры	Параллельный векторный суперкомпьютер (PVP).
Предшественники	NEC SX-4.
Узел	Каждый узел системы является векторно-конвейерным SMP-суперкомпьютером, объединяющим до 16 индивидуальных векторных процессоров (каждый с пиковой векторной производительностью 8 Gflop/s и скалярной производительностью 500 MFlop/s).
Память	Объем памяти каждого узла - до 128GB, производительность обменов с памятью достигает 1TB/sec.
Масштабируемость	Система может включать до 32 узлов, обеспечивая совокупную пиковую производительность до 4 TFlop/s.
Коммутатор	Для связи узлов используется высокоскоростной коммутатор (IXS Internode Crossbar Switch).
Системное ПО	Используется операционная система SUPER-UX.
Средства программирования	поставляются компилятор языка HPF, реализация интерфейса MPI, компиляторы Фортран 77/90 с автоматической векторизацией и поддержкой OpenMP 1.1, а также интегрированная среда разработки и оптимизации PSUITE.

NEC SX-6

Производитель	NEC, серия SX.
Класс архитектуры	Параллельный векторный суперкомпьютер

	ютер (PVP).
Предшественники	NEC SX-5.
Узел	Каждый узел системы является векторно-конвейерным SMP-суперкомпьютером, объединяющим от 2 до 8 индивидуальных векторных процессоров (каждый с пиковой векторной производительностью 8 Gflop/s и скалярной производительностью 500 MFlop/s).
Память	Объем памяти каждого узла - до 64GB, производительность обменов с памятью достигает 1TB/sec.
Масштабируемость	Система может включать до 128 узлов, обеспечивая совокупную пиковую производительность до 8 TFlop/s.
Коммутатор	Для связи узлов используется высокоскоростной коммутатор (IXS Internode Crossbar Switch).
Системное ПО	Используется операционная система SUPER-UX с улучшенной поддержкой SSI (Single System Image).
Средства программирования	поставляются компилятор языка HPF 2.0, реализация интерфейса MPI, компиляторы Фортран 77/90 с автоматической векторизацией, интегрированная среда разработки и оптимизации PSUITE, поддерживается OpenMP 1.1 (в конце 2002 года предполагается поддержка OpenMP 2.0).

Fujitsu VPP

Производитель	Fujitsu
Класс архитектуры	Параллельный векторный суперкомпьютер

	(PVP).
Модификации	VPP300, VPP700, VPP5000
Процессорный элемент	<p>Каждый процессорный элемент (PE) системы VPP700E состоит скалярного устройства (SU), векторного устройства (VU), блока памяти и устройства сопряжения.</p> <p>Для VPP700: VU состоит из 7 конвейеров и обеспечивает пиковую производительность до 2.4 GFLOP/sec. Объем памяти - до 2GB.</p> <p>Для VPP5000: VU состоит из 4 конвейеров, пиковая производительность - 9.6 GFLOP/sec. Объем памяти - до 16GB.</p>
Масштабируемость	<p>Для VPP700: система может включать от 8 до 256 PE, суммарная пиковая производительность до 14.4 GFLOP/sec</p> <p>Для VPP5000: до 512 PE, суммарная пиковая производительность до 4.9 TFLOP/sec.</p>
Коммутатор	Процессорные элементы связаны коммутатором (crossbar network), который производить двухсторонние обмены, не прерывая вычислений. Пропускная способность каналов коммутатора: для VPP700 - 615MB/sec, для VPP5000 - 1.6GB/sec.
Системное ПО	Используется операционная система UXP/V, основанная на UNIX System VR4.
Средства программирования	Среди средств разработки поставляются: распараллеливающий и векторизующий компилятор Fortran90/VPP, оптимизированная для VPP библиотека математических подпрограмм SSLII/VPP, библиотеки передачи сообщений MPI-2 и PVM 3.3.

AlphaServer

Производитель	Compaq (Digital).
Класс архитектуры.	AlphaServer GS/ES - высокопроизводительный SMP-сервер, AlphaServer SC - массивно-параллельная система, AlphaServer HPC - кластерные системы.
Модификации	GS320, GS160, HPC320, HPC160, GS140, GS60, ES40, DS20 и др.
Процессор	Alpha 21264, 21264A (тактовая частота до 731 MHz в новых моделях)
Число процессоров	до 32 (модель GS320)
Память	до 256 GB (модель GS320)
Масштабируемость	Системы HPC320 включают до 4-х узлов AlphaServer ES40, т.е. до 16 процессоров. Системы AlphaServer SC могут объединять до 128 узлов AlphaServer ES40, т.е. до 512 процессоров. Также Compaq предлагает разнообразные кластерные решения на базе своих серверов.
Системное ПО	На платформе AlphaServer поддерживаются операционные системы Tru64 UNIX (это новое имя Digital UNIX), OpenVMS и Linux. Поставляется ПО кластеризации TruCluster Software.
Средства программирования	Поддерживается параллельное программирование в стандартах OpenMP и MPI.

RM600 E

Производитель	Siemens Computer Systems (SNI), серия RM Servers.
Класс архитектуры	Многопроцессорная система с общей памятью (cc-NUMA).
Модификации	E60, E20
Процессор	Используются процессоры MIPS R10000 (200MHz).
Число процессоров	Процессорные платы с SMP-архитектурой объединяют до 4-х процессоров. Система включает в общей сложности до 24 процессоров в модели E60 и до 8 в модели E20.
Память	Общий объем оперативной памяти систем E60 - до 4GB, а систем E20 - до 2GB. Архитектура памяти системы - NUMA.
Масштабируемость	Несколько систем RM600 E могут объединяться в кластерную (MPP) систему - Reliant Cluster Server.
Системное ПО	Устанавливается операционная система Reliant UNIX.

QM-1

Производитель	Quadrics Supercomputers World Ltd. (QSW).
Класс архитектуры	Кластерная система
Процессор	UltraSPARC II/250 или 300 MHz с пиковой производительностью 500 MFLOP/sec (600 MFLOP/sec). На каждом процессоре находятся кэши данных и команд первого уровня по 16К и внешняя кэш-память объ-

	емом 1 или 2 МВ.
Узел	Каждый узел системы содержит от 1 до 4 процессоров над общей памятью, внешние устройства и коммуникационный процессор (Elan3), осуществляющий доступ в память других узлов.
Масштабируемость	В максимальной конфигурации может быть более 4000 узлов.
Память	Максимальный объем памяти для узла составляет 2GB. Скорость обменов с памятью внутри узла составляет 1.78 GB/sec.
Сеть	Заявленная скорость обменов с другими узлами составляет 250 MB/sec одновременно в двух направлениях, а латентность сети - 2 мкс.
Средства программирования	Реализованы средства параллельного программирования HPF, PVM, PARMACS, NX/2 и MPI.

NUMA-Q 2000

Производитель	IBM (ранее - Sequent)
Класс архитектуры	Многопроцессорная система с общей памятью (cc-NUMA) Используется для организации сложных информационных систем.
Модификации	Model E410/E330/E320/E300/E200
Процессоры	Intel Pentium III Xeon (700 MHz в модели E410)
Узел	от 4 до 64 процессоров, до 64 GB оперативной памяти; узел состоит из базовых плат по 4 процессора (quads), соединенных между собой коммутатором IQ-Link.

Масштабируемость	Возможна организация кластеров, включающих до 4 узлов
Системное ПО	Используется операционная система DYNIX/ptx - версия UNIX от Sequent. Внутри одной системы могут одновременно исполняться UNIX и Windows NT.

AViiON

Производитель	Data General (подразделение EMC)
Класс архитектуры	Многопроцессорная система с общей памятью (cc-NUMA). По аппаратной архитектуре похожа на сервера Sequent NUMA-Q. Используется для организации сложных информационных систем.
Модификации	AV25000, AV20000, AV10000 (корпоративные сервера)
Процессоры	Pentium II Xeon, Pentium III Xeon
Узел	От 4 до 64 процессоров, до 64 GB оперативной памяти. Сервер включает до 16 блоков (SBB, Scalable Building Block) на основе материнских плат SHV от Intel, в каждом из которых по 4 процессора и до 4GB памяти.
Масштабируемость	Возможна организация 2-серверных кластеров QuickClusters.
Системное ПО	Используются ОС DG/UX (версия UNIX от Data General) и Windows NT.

WorldMark

Производитель	NCR.
Класс архитектуры	Массивно-параллельная система на базе SMP-узлов. Используется для построения параллельных СУБД,

	масштабируемых систем Data Warehousing.
Модификации	WorldMark 5200/5150/5100M/5100C/5100S/4800/4700.
Процессоры	Intel Pentium II Xeon, Pentium Pro, Pentium.
Узел	Для WorldMark 5100M в качестве узла используется SMP-сервер 5100S - до 32 процессоров, до 4 GB оперативной памяти.
Масштабируемость	До 128 узлов, 4096 процессоров (для WorldMark 5100M).
Коммуникационная сеть	Узлы объединяются с помощью сети BYNET V2 (120 MB/sec).
Системное ПО	Операционная система NCR UNIX SVR4 MP-RAS. Параллельные СУБД: NCR Teradata, Oracle Parallel Server и др.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. В чем отличие числовой и нечисловой обработки данных?
2. Перечислите ограничения фон-неймановской архитектуры.

3. Поясните суть параллельной обработки данных.
4. Поясните конвейерную обработку данных.
5. Что такое «время разгона» конвейера?
6. Поясните суть организации векторных конвейеров.
7. Проведите классификацию архитектур вычислительных систем.
8. Расскажите о мультипроцессорных системах с общей памятью.
9. Расскажите о мультипроцессорных системах с локальной памятью.
10. Перечислите базовые архитектуры суперкомпьютеров.
11. Расскажите об архитектуре современных векторных суперкомпьютеров.
12. Что такое ассоциативная память и ассоциативный процессор?
13. Изложите концепцию вычислительных систем с управлением потоком данных.
14. Поясните закон Амдала и его следствия.

Глава 3.

Введение в теорию массового обслуживания и управления ресурсами ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

3.1.Понятие Марковского случайного процесса

Случайный процесс, протекающий в системе, называется марковским, если для любого момента времени t_0 вероятностные характеристики процесса в будущем зависят только от его состояния в данный момент t_0 и не зависят от того, когда и как система пришла в это состояние [4].

Пусть в настоящий момент t_0 система находится в определенном состоянии S_0 (см. рисунок 3.1)

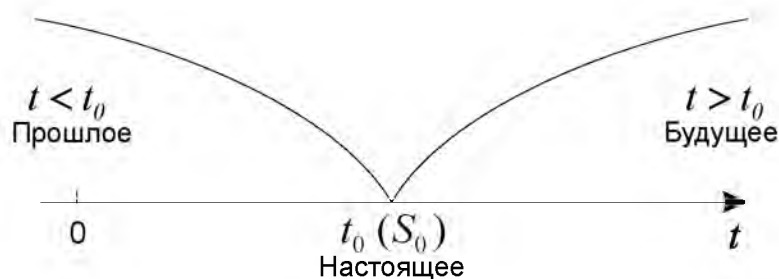


Рисунок 3.1 – К понятию Марковского случайного процесса

Мы наблюдаем процесс со стороны и в момент t_0 знаем состояние системы S_0 и всю предысторию процесса, все, что было при $t < t_0$. Нас интересует будущее ($t > t_0$). В точности предсказать его мы не можем, так как процесс случайный. Но мы можем найти некоторые вероятностные характеристики процесса в будущем, например, вероятность того, что через некоторое время t система S окажется в состоянии S_1 или сохранит состояние S_0 , и т.п. Таким образом, если процесс марковский, то предсказывать можно, только учитывая настоя-

щее состояние системы S_0 и забыв о его предыстории. Само состояние S_0 зависит от прошлого, но как только оно достигнуто, о прошлом можно забыть. Иначе формулируя, в марковском процессе «будущее зависит от прошлого только через настоящее».

На практике марковские процессы в чистом виде обычно не встречаются, но нередко приходится иметь дело с процессами, для которых влиянием предыстории можно пренебречь. При их изучении можно с успехом применять марковские модели.

В исследовании операций большое значение имеют так называемые *марковские процессы с дискретным состоянием и непрерывным временем*. Процесс называется процессом с непрерывным временем, если моменты возможных переходов из состояния в состояние не фиксированы заранее, а неопределенны, случайны, если переход может осуществиться, в принципе, в любой момент.

3.2.Потоки событий

Потоком событий называется последовательность однородных событий, следующих одно за другим в какие-то случайные моменты времени. Например, поток вызовов на телефонной станции; поток сбоев ЭВМ.

Поток событий можно наглядно изобразить рядом точек на оси времени $0t$, причем положение их случайно (см. рисунок 3.2).

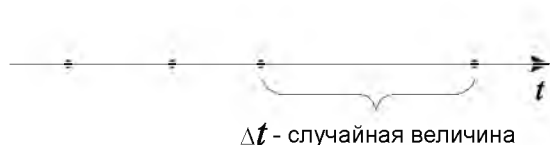


Рисунок 3.2 - Поток событий

Важной характеристикой потока событий является его интенсивность λ — среднее число событий, приходящееся на единицу времени. Интенсивность может быть как постоянной, так и переменной, зависящей от времени t .

Поток событий называется простейшим (или стационарным пуассоновским), если он обладает сразу тремя свойствами: стационарен, ординарен и без последствия. Название простейший связано с тем, что процессы, связанные с простейшими потоками, имеют наиболее простое математическое описание.

Для простейшего потока с интенсивностью λ интервал T между событиями имеет так называемое показательное распределение [4]:

$$P(t) = 1 - e^{-\lambda t}$$

с плотностью

$$\frac{dP}{dt} = f(\tau) = \lambda e^{-\lambda \tau} \quad (\tau > 0) \quad (3.1)$$

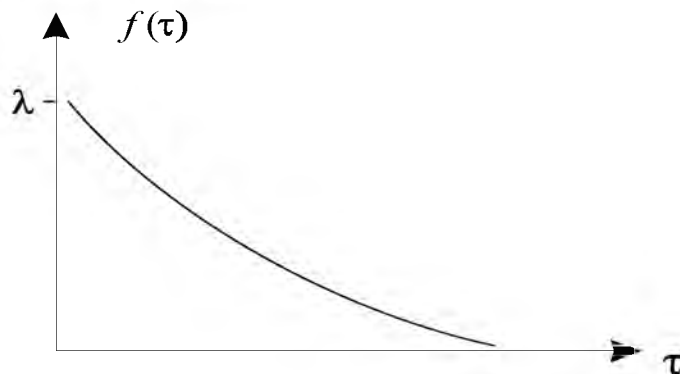


Рисунок 3.3 - Показательное распределение

Величина λ в формуле (3.1) называется параметром показательного закона. Если интервал T между событиями

$$T \leq 1/\lambda, \quad (3.2)$$

то такой интервал называют коротким.

Для простейшего потока характерно, что короткие интервалы между событиями более вероятны, чем длинные; $\approx 63\%$ промежутков времени между событиями имеют длину меньше средней, равной $1/\lambda$. Таким образом, предположение о действии на вычислительную систему простейшего потока заявок создает более тяжелые условия для ее работы, чем при других потоках, что позволяет считать результаты анализа ВС для простейших потоков заявок более надежными.

В расчетах, связанных с потоками событий, очень удобно пользоваться понятием «элемента вероятности». Рассмотрим на оси Ot простейший поток с интенсивностью λ и произвольно расположенный элементарный участок времени dt . Элементом вероятности называется вероятность попадания на этот участок хотя бы одного события потока.

$$P_{\Delta t} = \lambda \Delta t \quad (3.3)$$

то есть для простейшего потока элемент вероятности равен интенсивности потока, умноженной на длину элементарного участка. Элемент вероятности из-за отсутствия последствия, совершенно не зависит от того, сколько событий и когда появлялись ранее.

3.3. Уравнения Колмогорова

Пусть техническое устройство S состоит из двух узлов, каждый из которых в случайный момент времени может выйти из строя, после чего мгновенно начинается ремонт узла, тоже продолжающийся заранее неизвестное, случайное время.

Возможные состояния системы можно перечислить:

S_0 – оба узла исправны;

S_1 - первый узел ремонтируется, второй исправен;

S_2 - второй узел ремонтируется, первый исправен;

S_3 – оба узла ремонтируются.

Переходы системы из состояния в состояние происходят практически мгновенно, в случайные моменты выхода из строя того или другого узла или окончания ремонта.

При анализе случайных процессов с дискретными состояниями удобно пользоваться графом состояний, в котором состояния системы изображаются окружностями, а возможные переходы из состояния в состояние – стрелками, соединяющими состояния.

Граф состояний для рассмотренного выше примера изображен на рисунке 3.4

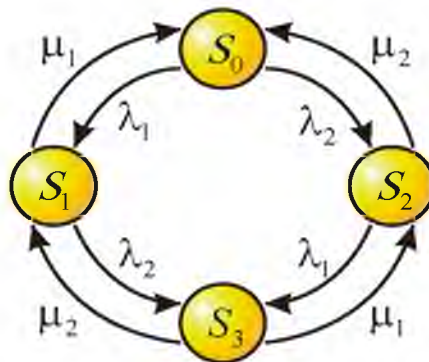


Рисунок 3.4 - Граф состояний

Стрелка, направленная из S_0 в S_1 , означает переход в момент отказа первого узла; стрелка, направленная обратно, из S_1 в S_0 - переход в момент окончания ремонта этого узла. Остальные стрелки объясняются аналогично.

Буквой μ_i обозначена интенсивность ремонта i -го узла.

Вероятностью i -го состояния называется вероятность $P_i(t)$ того, что в момент t система будет находиться в состоянии S_i . Очевидно, что для любого момента сумма всех вероятностей состояний равна единице:

$$\sum_{i=0}^n P_i(t) = 1 \quad (3.4)$$

Имея в своем распоряжении размеченный граф состояний, можно найти все вероятности состояний $P_i(t)$ как функции времени. Для этого составляются и решаются так называемые уравнения Колмогорова - особого вида дифференциальные уравнения, в которых неизвестными функциями являются вероятности состояний.

Составим эти уравнения для заданной системы. Рассмотрим одну из вероятностей состояний, например $P_0(t)$. Это вероятность того, что в момент t система будет в состоянии S_0 . Придадим t малое приращение Δt и найдем $P_0(t+\Delta t)$ – вероятность того, что в момент $(t+\Delta t)$ система будет в состоянии S_0 .

Произойти это может тремя способами: либо в момент t система уже была в состоянии S_0 , а за время Δt не вышла из него; либо в момент t система была в состоянии S_1 , а за время Δt перешла из него в S_0 ; либо в момент t система была в состоянии S_2 , а за время Δt перешла из него в S_0 .

Найдем вероятность первого варианта. Вероятность того, что в момент t система была в состоянии S_0 , равна $P_0(t)$. Эту вероятность нужно умножить на вероятность того, что, находясь в момент t в состоянии S_0 , система за время Δt не перейдет из него в другое состояние. Суммарный поток событий, выводящий систему из состояния S_0 , будет простейший, с интенсивностью $(\lambda_1 + \lambda_2)$.

Значит, вероятность того, что за время Δt система выйдет из состояния S_0 , равна $(\lambda_1 + \lambda_2) \cdot \Delta t$; вероятность того, что не выйдет:

$(1 - (\lambda_1 + \lambda_2) \cdot \Delta t)$. Отсюда вероятность первого варианта равна $P_0(t) \cdot (1 - (\lambda_1 + \lambda_2) \cdot \Delta t)$.

Вероятность второго варианта равна $(\mu_1 \cdot \Delta t \cdot P_1(t))$; третьего $(\mu_2 \cdot \Delta t \cdot P_2(t))$.

Складывая вероятности всех вариантов, получим:

$$P_0(t + \Delta t) = P_0(t) \cdot [1 - (\lambda_1 + \lambda_2) \cdot \Delta t] + \mu_1 \cdot \Delta t \cdot P_1(t) + \mu_2 \cdot \Delta t \cdot P_2(t) \quad (3.5)$$

Раскроем квадратные скобки, перенесем $P_0(t)$ в левую часть и разделим обе части на Δt :

$$\frac{(P_0(t + \Delta t) - P_0(t))}{\Delta t} = -P_0(t) \cdot (\lambda_1 + \lambda_2) + P_1(t) \cdot \mu_1 + P_2(t) \cdot \mu_2 \quad (3.6)$$

Устремив Δt к нулю, получим:

$$\frac{dP_0}{dt} = P_1 \cdot \mu_1 + P_2 \cdot \mu_2 - P_0 \cdot (\lambda_1 + \lambda_2) \quad (3.7)$$

Таким образом, мы получили первое уравнение Колмогорова. Аналогично составляются и следующие уравнения.

Сформулируем теперь общее правило составления уравнений Колмогорова. В левой части каждого из них стоит производная вероятности данного состояния. В правой части – сумма произведений вероятностей всех состояний, из которых идут стрелки в данное состояние, на интенсивности соответствующих потоков событий, минус суммарная интенсивность всех потоков, выводящих систему из данного состояния, умноженная на вероятность данного состояния.

Пользуясь этим правилом, получим уравнения Колмогорова для рассмотренной системы S:

$$\begin{aligned}\frac{dP_0}{dt} &= P_1 \cdot \mu_1 + P_2 \cdot \mu_2 - P_0 \cdot (\lambda_1 + \lambda_2) \\ \frac{dP_1}{dt} &= P_0 \cdot \lambda_1 + P_3 \cdot \mu_2 - P_1 \cdot (\mu_1 + \lambda_2) \\ \frac{dP_2}{dt} &= P_0 \cdot \lambda_2 + P_3 \cdot \mu_1 - P_2 \cdot (\lambda_1 + \mu_2) \\ \frac{dP_3}{dt} &= P_1 \cdot \lambda_2 + P_2 \cdot \lambda_1 - P_3 \cdot (\mu_1 + \mu_2).\end{aligned}\tag{3.8}$$

Чтобы решить эти уравнения и найти вероятности состояний, необходимо задать начальные условия.

Правомерен вопрос: что будет происходить с вероятностями состояний при t , стремящемся к бесконечности? Будут ли $P_i(t)$ стремиться к каким-то пределам? Если эти пределы существуют и не зависят от начального состояния системы, то они называются финальными вероятностями состояний.

Если вероятности $P_i(t)$ постоянны, то их производные равны нулю. Значит, чтобы найти финальные вероятности, нужно все левые части в уравнениях Колмогорова положить равными нулю и решить полученную систему уже не дифференциальных, а линейных алгебраических уравнений. Для нашей системы они будут выглядеть следующим образом:

$$\begin{aligned}P_0 * (\lambda_1 + \lambda_2) &= P_1 * \mu_1 + P_2 * \mu_2 \\ P_1 * (\mu_1 + \lambda_2) &= P_0 * \lambda_1 + P_3 * \mu_2\end{aligned}\tag{3.9}$$

$$P_2 * (\lambda_1 + \mu_2) = P_0 * \lambda_2 + P_3 * \mu_1$$

$$P_3 * (\mu_1 + \mu_2) = P_1 * \lambda_2 + P_2 * \lambda_1$$

Для решения этой системы необходимо воспользоваться нормировочным условием $P_0 + P_1 + P_2 + P_3 = 1$.

3.4. Базовые соотношения систем массового обслуживания

3.4.1. Схема гибели и размножения

Граф состояний для схемы гибели и размножения имеет вид, показан на рисунке 3.5.

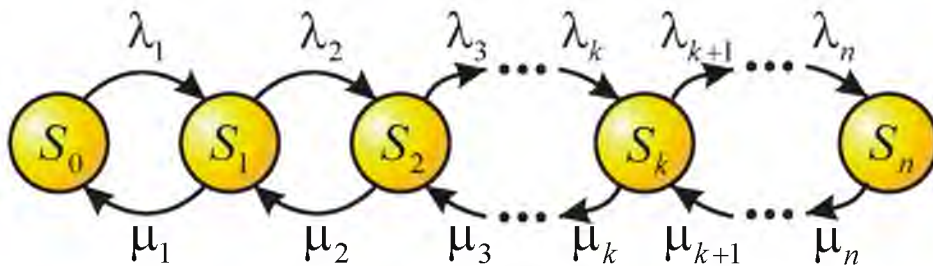


Рисунок 3.5 - Схема гибели и размножения

Особенность этого графа в том, что все состояния системы можно вытянуть в одну цепочку, в которой каждое из средних состояний (S_1, S_2, \dots, S_{n-1}) связано прямой и обратной стрелкой с каждым из соседних состояний – правым и левым, а крайние состояния (S_0, S_n) – только с одним соседним состоянием. Термин «схема гибели и размножения» ведет начало от биологических задач, где подобной схемой описывается изменение численности популяции.

Пользуясь графом рисунка 3.5, составим и решим алгебраические уравнения Колмогорова для финальных вероятностей состояний.

Для состояния S_0 :

$$P_0 \lambda_1 = P_1 \mu_1 \tag{3.10}$$

Для состояния S_1 :

$$P_1 (\mu_1 + \lambda_2) = P_0 \lambda_1 + P_2 \mu_2$$

Или, с учетом уравнения (3.10), для состояния S_1 окончательно:

$$P_1 \lambda_2 = P_2 \mu_2$$

Очевидно, для состояния S_2 получим:

$$P_2 \lambda_3 = P_3 \mu_3$$

Из уравнения для состояния S_0 выразим P_1 через P_0 :

$$P_1 = P_0 \cdot \frac{\lambda_1}{\mu_1} \quad (3.11)$$

С учетом (3.11), получим для вероятности P_2 :

$$P_2 = P_1 \cdot \frac{\lambda_2}{\mu_2} = \frac{\lambda_1 \cdot \lambda_2}{\mu_1 \cdot \mu_2} \cdot P_0 \quad (3.12)$$

С учетом (3.12), для P_3 :

$$P_3 = \frac{\lambda_1 \cdot \lambda_2 \cdot \lambda_3}{\mu_1 \cdot \mu_2 \mu_3} \cdot P_0 \quad (3.13)$$

И вообще, для любого k (от 1 до n)

$$P_k = \frac{\lambda_k \cdot \lambda_{k-1} \cdot \dots \cdot \lambda_1}{\mu_k \cdot \mu_{k-1} \cdot \dots \cdot \mu_1} \cdot P_0 \quad (3.14)$$

Таким образом, все вероятности состояний выражаются через P_0 .

Подставив эти выражения в нормировочное условие (3.4) и вынеся P_0 за скобку, получим:

$$P_0 = \left(1 + \frac{\lambda_1}{\mu_1} + \frac{\lambda_1 \cdot \lambda_2}{\mu_1 \cdot \mu_2} + \dots + \frac{\lambda_n \cdot \lambda_{n-1} \cdot \dots \cdot \lambda_1}{\mu_n \cdot \mu_{n-1} \cdot \dots \cdot \mu_1} \right)^{-1} \quad (3.15)$$

Полученные формулы очень полезны при решении простейших задач теории массового обслуживания.

3.4.2. Формула Литтла

Теперь выведем одну важную формулу, связывающую среднее число заявок $L_{\text{сист}}$, находящихся в системе массового обслуживания, и среднее время пребывания заявки в системе $W_{\text{сист}}$.

Рассмотрим любую СМО и связанные с ней два потока событий:

$X(t)$ – число заявок, прибывших в СМО до момента t ;

$Y(t)$ – число заявок, покинувших СМО до момента t .

И та, и другая функции являются случайными и меняются скачком (увеличиваются на единицу в моменты приходов и уходов заявок). Вид функции $X(t)$ и $Y(t)$ показан на рисунке 3.6; обе линии - ступенчатые, верхняя – $X(t)$, нижняя – $Y(t)$. Очевидно, что для любого момента t их разность $Z(t)=X(t)-Y(t)$ есть не что иное, как число заявок, находящихся в СМО. Когда линии $X(t)$ и $Y(t)$ сливаются, в системе нет заявок.

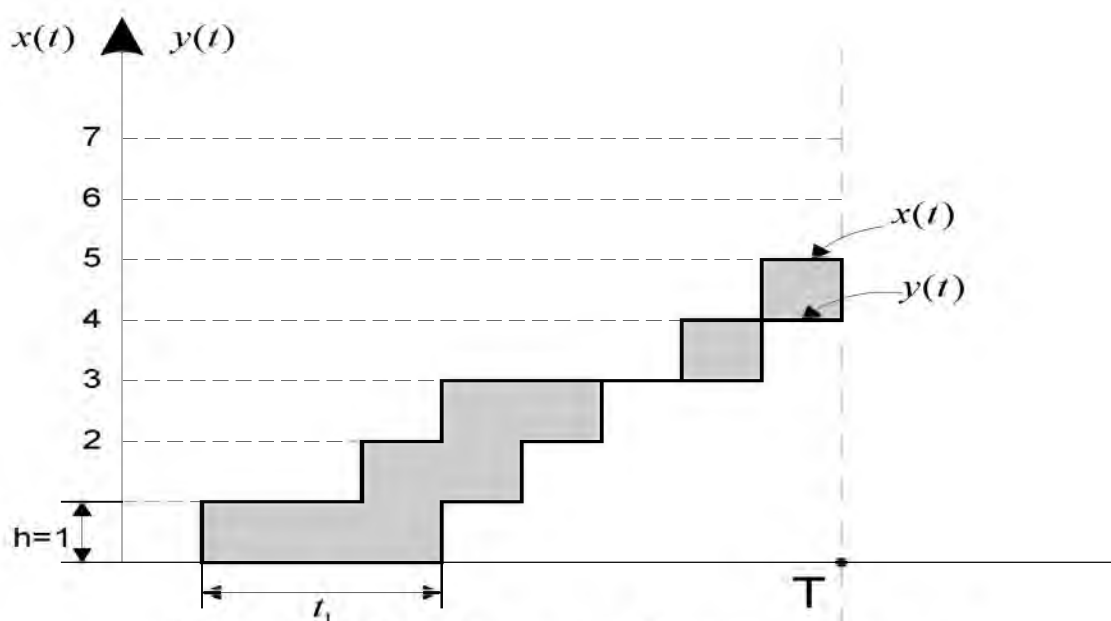


Рисунок 3.6 – К выводу формулы Литтла

Рассмотрим очень большой промежуток времени T и вычислим для него среднее число заявок, находящихся в СМО. Оно будет равно интегралу от функции $Z(t)$ на этом промежутке, деленному на длину интервала T :

$$L_{\text{сист}} = \frac{1}{T} \int_0^T Z(t) dt \quad (3.16)$$

Интеграл $\int_0^T Z(t) dt$ представляет собой не что иное, как площадь фигуры, заштрихованной на рисунке 3.6, состоящей из прямоугольников высотой, равной единице и основанием t_i . Поэтому, можно считать, что

$$\int_0^T Z(t)dt = \sum_i t_i, \quad (3.17)$$

где сумма распространяется на все заявки, пришедшие за время T .

Разделим правую и левую часть (3.17) на длину интервала T , получим с учетом (3.16):

$$L_{\text{сист}} = \frac{1}{T} \sum_i t_i \quad (3.18)$$

Разделим и умножим правую часть (3.18) на интенсивность λ :

$$L_{\text{сист}} = \frac{1}{T\lambda} \sum_i t_i \cdot \lambda \quad (3.19)$$

Но величина $T\lambda$ есть не что иное, как среднее число заявок, пришедших за время T . Если мы разделим сумму всех времен t_i на среднее число заявок, то получим среднее время пребывания заявки в системе $W_{\text{сист}}$. Итак,

$$L_{\text{сист}} = \lambda \cdot W_{\text{сист}},$$

откуда

$$W_{\text{сист}} = \frac{1}{\lambda} \cdot L_{\text{сист}} \quad (3.20)$$

Это и есть замечательная формула Литтла: для любой СМО, при любом характере потока заявок, при любом распределении времени обслуживания, при любой дисциплине обслуживания среднее время пребывания заявки в системе равно среднему числу заявок в системе, деленному на интенсивность потока заявок.

Точно таким же образом выводим вторую формулу Литтла, связывающую среднее время пребывания заявки в очереди $W_{\text{оч}}$ и среднее число заявок в очереди $L_{\text{оч}}$:

$$W_{\text{оч}} = \frac{1}{\lambda} \cdot L_{\text{оч}} \quad (3.21)$$

3.5. Простейшие системы массового обслуживания и их характеристики

Все потоки событий, переводящие систему массового обслуживания (СМО) из состояния в состояние, будем считать простейшими. Поток обслуживаний - поток заявок, обслуживаемых одним непрерывно занятым каналом. В этом потоке интервал между событиями, как и всегда в простейшем потоке, имеет показательное распределение.

3.5.1. Задача Эрланга

Рассмотрим одну из первых по времени «классических» задач теории массового обслуживания; эта задача возникла из практических нужд телефонии и была решена в начале нашего века датским математиком Эрлангом. Задача ставится так: имеется n каналов, на которые поступает поток заявок с интенсивностью λ . Поток обслуживания одним каналом имеет интенсивность μ (величина, обратная среднему времени обслуживания t_{ob}). Найти финальные вероятности состояний СМО, а также характеристики ее эффективности:

A – абсолютную пропускную способность, то есть среднее число заявок, обслуживаемых в единицу времени;

Q – относительную пропускную способность, то есть среднюю долю пришедших заявок, обслуживаемых системой;

$P_{отк}$ - вероятность отказа, то есть того, что заявка покинет СМО необслуженной;

\bar{k} – среднее число занятых каналов.

Состояние системы массового обслуживания S будем нумеровать по числу заявок, находящихся в системе (в данном случае оно совпадает с числом занятых каналов):

S_0 – в СМО нет ни одной заявки;

S_1 - в СМО находится одна заявка (один канал занят, остальные свободны);

.....

S_k - в СМО находится k заявок (k каналов заняты, остальные свободны)

S_n - в СМО находятся n заявок (все n каналов заняты).

Граф состояний СМО выглядит следующим образом (рисунок 3.7):

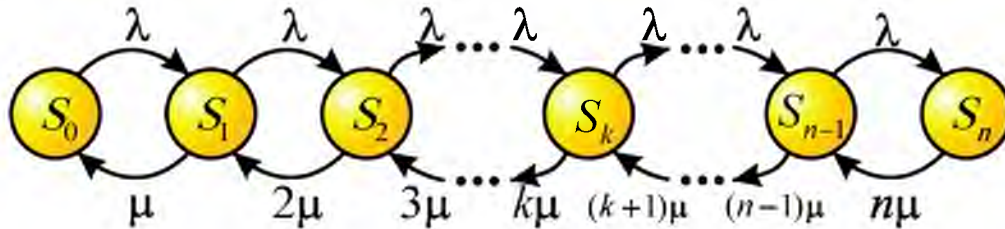


Рисунок 3.7 – Граф состояний многоканальной СМО с отказами

Разметим этот граф – проставим у стрелок интенсивности потоков событий. Интенсивности потоков λ все одинаковы, что естественно, так как на СМО действует один и тот же внешний поток заявок на обслуживание. Интенсивности же потоков обслуживания увеличиваются с каждым занятым каналом. А теперь, зная все интенсивности, воспользуемся уже готовыми формулами для финальных вероятностей в системе гибели-размножения. Получим:

$$P_0 = \left(1 + \frac{\lambda}{\mu} + \frac{\lambda^2}{2! \mu^2} + \frac{\lambda^3}{3! \mu^3} + \dots + \frac{\lambda^k}{k! \mu^k} + \dots + \frac{\lambda^n}{n! \mu^n} \right)^{-1} \quad (3.22)$$

где ! обозначает факториал.

Члены разложения $\frac{\lambda}{\mu}, \dots, \frac{\lambda^n}{n! \mu^n}$ будут представлять собой коэффициенты при P_0 в выражениях для P_1, P_2, \dots, P_n :

$$\begin{aligned}
P_1 &= \frac{\lambda}{\mu} \cdot P_0 \\
P_2 &= \frac{\lambda^2}{2\mu^2} \cdot P_0 \\
&\dots \\
P_k &= \frac{\lambda^k}{k! \mu^k} \cdot P_0 \\
P_h &= \frac{\lambda^n}{n! \mu^n} \cdot P_0
\end{aligned} \tag{3.23}$$

Заметим, что в формулы (3.22), (3.23) интенсивности λ и μ входят не по отдельности, а только в виде отношения λ/μ . Обозначим

$$\frac{\lambda}{\mu} = \rho \tag{3.24}$$

и будем называть величину ρ приведенной интенсивностью потока заявок. Ее смысл - среднее число заявок, приходящее за среднее время обслуживания одной заявки. Пользуясь этим обозначением, перепишем формулы (3.22), (3.23) в виде:

$$P_0 = \left(1 + \rho + \frac{\rho^2}{2!} + \dots + \frac{\rho^k}{k!} + \dots + \frac{\rho^n}{n!} \right)^{-1}$$

$$P_1 = \rho \cdot P_0$$

$$P_2 = \frac{\rho^2}{2!} \cdot P_0$$

...

$$P_k = \frac{\rho^k}{k!} \cdot P_0$$

...

$$P_n = \frac{\rho^n}{n!} \cdot P_0$$

(3.25)

Формулы (3.25) для финальных вероятностей состояний называются формулами Эрланга - в честь основателя теории массового обслуживания.

По финальным вероятностям можно вычислить характеристики эффективности СМО. Сначала найдем $P_{отк}$ - вероятность того, что пришедшая заявка получит отказ. Для этого нужно, чтобы все n каналов были заняты, значит,

$$P_{отк} = P_n = \frac{\rho^n}{n!} \cdot P_0 \quad (3.26)$$

Отсюда находим относительную пропускную способность - вероятность того, что заявка будет обслужена:

$$Q = 1 - P_{отк} = 1 - \frac{\rho^n}{n!} \cdot P_0 \quad (3.27)$$

Абсолютную пропускную способность получим, умножая интенсивность потока заявок λ на Q :

$$A = \lambda \cdot Q = \lambda \cdot \left(1 - \frac{\rho^n}{n!} \cdot P_0 \right) \quad (3.28)$$

Таким образом, мы нашли интенсивность потока обслуженных системой заявок. Так как каждый занятый канал в единицу времени обслуживает в среднем μ заявок, то среднее число занятых каналов равно

$$\bar{k} = \frac{A}{\mu} = \rho \left(1 - \frac{\rho^n}{n!} \cdot P_0 \right) \quad (3.29)$$

3.5.2. Одноканальная СМО с неограниченной очередью

На практике довольно часто встречаются одноканальные СМО с очередью (врач, обслуживающий пациентов; телефон-автомат с одной будкой; ЭВМ, выполняющая заказы пользователей). Пусть имеется одноканальная СМО с очередью, на которую не наложено никаких ограничений (ни по длине очереди, ни по времени ожидания). На эту СМО поступает поток заявок с интенсивностью λ ; поток обслуживаний имеет интенсивность μ , обратную среднему времени обслуживания заявки $t_{об}$. Требуется найти финальные вероятности состояний СМО, а также характеристики ее эффективности:

$L_{сист}$ - среднее число заявок в системе;

$W_{сист}$ - среднее время пребывания заявки в системе;

$L_{оч}$ - среднее число заявок в очереди;

$W_{оч}$ - среднее время пребывания заявки в очереди;

$P_{зан}$ - вероятность того, что канал занят (степень загрузки канала).

Что касается абсолютной пропускной способности A и относительной Q , то вычислять их нет надобности: в силу того, что очередь не ограничена, каждая заявка рано или поздно будет обслужена, поэтому $A = \lambda$, по той же причине $Q=1$.

Решение. Состояния системы будем нумеровать по числу заявок, находящихся в СМО:

S_0 - канал свободен;

S_1 - канал занят, очереди нет;

S_2 - канал занят, одна заявка стоит в очереди; и т.д.

Теоретически число состояний ничем не ограничено. Граф состояний имеет вид (рисунок 3.8).

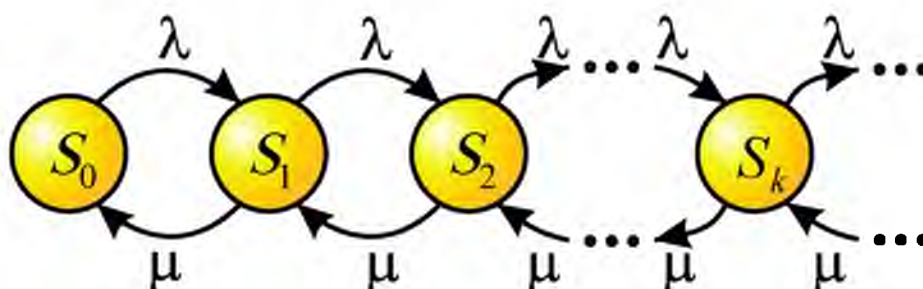


Рисунок 3.8 – Граф состояний одноканальной СМО с неограниченной очередью

Это - схема гибели и размножения, но с бесконечным числом состояний. По всем стрелкам поток заявок с интенсивностью λ переводит систему слева направо, а справа налево - поток обслуживаний с интенсивностью μ . Если $\lambda > \mu$, то канал с заявками не справляется, очередь растет до бесконечности. Если $\lambda \leq \mu$, то задача вполне разрешима. Воспользуемся формулами для финальных вероятностей из схемы гибели и размножения и для бесконечного числа состояний. Подсчитаем финальные вероятности :

$$P_0 = (1 + \rho + \rho^2 + \dots + \rho^k + \dots)^{-1} \quad (3.30)$$

Ряд в формуле (3.30) представляет собой геометрическую прогрессию. Известно, что при $\rho < 1$ ряд сходится; при $\rho \geq 1$ ряд расходится. Теперь предположим, что это условие выполнено, и $\rho < 1$. Суммируя прогрессию в (3.30), получаем

$$1 + \rho + \rho^2 + \dots = \frac{1}{1 - \rho}$$

$$\text{откуда } p_0 = 1 - \rho. \quad (3.31)$$

Вероятности $p_1, p_2, \dots, p_k, \dots$ найдутся по формулам:

$$p_1 = \rho \cdot p_0, \quad p_2 = \rho^2 \cdot p_0, \quad \dots, \quad p_k = \rho^k \cdot p_0, \quad \dots,$$

откуда с учетом (3.31) найдем окончательно:

$$p_1 = \rho \cdot (1 - \rho), p_2 = \rho^2 \cdot (1 - \rho), \dots, p_k = \rho^k \cdot (1 - \rho). (3.32)$$

Как видно, вероятности p_0, p_1, \dots образуют геометрическую прогрессию со знаменателем ρ . Как ни странно, максимальная из них p_0 - вероятность того, что канал будет вообще свободен. Как бы ни была загружена система с очередью, если только она вообще справляется с потоком заявок, самое вероятное число заявок в системе будет равно нулю.

Найдем среднее число заявок в СМО $L_{сис}$. Случайная величина Z – число заявок в системе – имеет возможные значения $0, 1, 2, \dots, k, \dots$ с вероятностями $p_0, p_1, \dots, p_k, \dots$. Ее математическое ожидание равно

$$L_{сис} = 0 \cdot p_0 + 1 \cdot p_1 + 2 \cdot p_2 + \dots + k \cdot p_k + \dots = \sum_{k=1}^{\infty} k \cdot p_k \quad (3.33)$$

Подставим в (3.33) выражение для p_k из (3.32):

$$L_{сис} = \sum_{k=1}^{\infty} k \cdot \rho^k \cdot (1 - \rho) = \rho \cdot (1 - \rho) \cdot \sum_{k=1}^{\infty} k \cdot \rho^{k-1}$$

Произведение $k \cdot \rho^{k-1}$ есть ни что иное, как производная по ρ от выражения ρ^k ; значит,

$$L_{сис} = \rho \cdot (1 - \rho) \cdot \sum_{k=1}^{\infty} \frac{d}{d\rho} \cdot \rho^k = \rho \cdot (1 - \rho) \cdot \frac{d}{d\rho} \cdot \sum_{k=1}^{\infty} \rho^k \quad (3.34)$$

Но сумма в (3.34) есть не что иное, как сумма бесконечно убывающей геометрической прогрессии с первым членом ρ и знаменателем ρ ; эта сумма равна $\frac{\rho}{1 - \rho}$, а ее производная

$\frac{1}{(1 - \rho)^2}$. Подставляя это выражение в (3.34), получим:

$$L_{сис} = \frac{\rho}{1 - \rho}. \quad (3.35)$$

Теперь применим формулу Литтла и найдем среднее время пребывания заявки в системе:

$$W_{сис} = \frac{\rho}{\lambda \cdot (1 - \rho)} \quad (3.36)$$

Найдем среднее число заявок в очереди $L_{оч}$. Будем рассуждать так: число заявок в очереди равно числу заявок в си-

стеме минус число заявок, находящихся под обслуживанием. Значит (по правилу сложения математических ожиданий), среднее число заявок в очереди $L_{оч}$ равно среднему числу заявок в системе $L_{сист}$ минус среднее число заявок под обслуживанием. Число заявок под обслуживанием может быть либо нулем (канал свободен), либо единицей (канал занят). Математическое ожидание такой случайной величины равно вероятности того, что канал занят ($P_{зан}$). Очевидно, $P_{зан}$ равно 1 минус вероятность p_0 того, что канал свободен:

$$P_{зан} = 1 - p_0 = \rho. \quad (3.37)$$

Следовательно, среднее число заявок под обслуживанием равно

$$L_{оч} = \rho \quad (3.38)$$

отсюда

$$L_{оч} = L_{сист} - \rho = \frac{\rho}{1 - \rho} - \rho$$

и окончательно

$$L_{оч} = \frac{\rho^2}{1 - \rho} \quad (3.39).$$

По формуле Литтла найдем среднее время пребывания заявки в очереди:

$$W_{оч} = \frac{\rho^2}{\lambda \cdot (1 - \rho)} \quad (3.40)$$

Таким образом, все характеристики эффективности СМО найдены.

3.5.3. Многоканальная СМО с неограниченной очередью

Аналогично одноканальной СМО решается задача о многоканальной СМО с неограниченной очередью. Нумерация каналов - опять по числу заявок, находящихся в очереди:

S_0 - все каналы свободны;

S_1 - один канал занят, очереди нет;

S_2 - занято два канала;

.....

S_n - занято n каналов;

S_{n+1} - заняты все n каналов, одна заявка стоит в очереди;

.....

S_{n+r} - заняты все n каналов, r заявок стоит в очереди;

.....

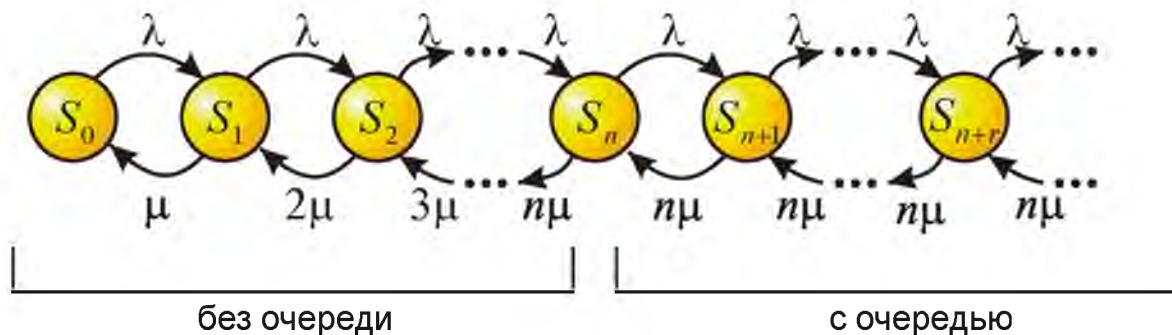


Рисунок 3.9 – Граф состояний многоканальной СМО с неограниченной очередью

Граф состояний показан на рисунке 3.9. Граф есть схема гибели и размножения, но с бесконечным числом состояний. Естественное условие существования финальных вероятностей- $\frac{\rho}{n} < 1$. Если $\frac{\rho}{n} \geq 1$, очередь растет до бесконечности.

Предположим, что условие $\frac{\rho}{n} < 1$ выполнено, и финальные вероятности существуют. Применяя формулы для схемы гибели и размножения, найдем эти финальные вероятности. В выражении для P_0 будет стоять ряд членов, содержащих факториалы, плюс сумма бесконечно убывающей геометрической прогрессии со знаменателем $\frac{\rho}{n}$. Суммируя ее, найдем

$$\begin{cases} P_0 = (1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \dots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n! \cdot (n - \rho)})^{-1} \\ p_1 = \frac{\rho}{1!} \cdot p_0, \dots, p_k = \frac{\rho^k}{k!} \cdot p_0, \dots, p_n = \frac{\rho^n}{n!} \cdot p_0, \\ p_{n+1} = \frac{\rho^{n+1}}{n \cdot n!} \cdot p_0, \dots, p_{n+r} = \frac{\rho^{n+r}}{n^r \cdot n!} \cdot p_0, \dots \end{cases}$$

(3.41)

Теперь найдем характеристики эффективности СМО. Из них легче всего находится среднее число занятых каналов $\bar{k} = \frac{\lambda}{\mu} = \rho$ (это вообще справедливо для любой СМО с неограниченной очередью). Найдем среднее число заявок в системе $L_{сист}$ и среднее число заявок в очереди $L_{оч}$. Из них легче вычислить второе по формуле $L_{оч} = \sum_{r=1}^{\infty} r \cdot p_{n+r}$; выполняя соответствующие преобразования по образцу одноканальной СМО с неограниченной очередью, получим:

$$L_{оч} = \frac{\rho^{n+1} \cdot p_0}{n \cdot n! \cdot (1 - \rho/n)^2} \quad (3.42)$$

Прибавляя к нему среднее число заявок под обслуживанием (оно же - среднее число занятых каналов) $\bar{k} = \rho$, получаем:

$$L_{сист} = L_{оч} + \rho \quad (3.43)$$

Деля выражение для $L_{сист}$ и $L_{оч}$ на λ , по формуле Литтла получим средние времена пребывания заявки в очереди и в системе:

$$W_{оч} = \frac{1}{\lambda} \cdot L_{оч}, \quad W_{сист} = \frac{1}{\lambda} \cdot L_{сист} \quad (3.44)$$

3.6. Управление ресурсами вычислительных систем

3.6.1. Управление ресурсами однопроцессорных систем оперативной обработки данных

3.6.1.1. Алгоритм SPT

В системах оперативной обработки в качестве основного критерия эффективности используется среднее время обслуживания заявок. Нетрудно видеть, что в случае, когда времена решения задач априори известны, минимальное среднее время ответа дает алгоритм SPT (Shortest-processing-task-first), назначающий задачи на решение в порядке убывания времени решения t_i , то есть $t_1 \leq t_2 \leq \dots \leq t_L$. При этом время ответа u_i для задачи z_i есть $u_i = \sum t_j$, где $\sum t_j$ - время ожидания, t_i - собственно время решения и среднее время ответа есть

$$u^* = \frac{1}{L} \sum_{i=1}^L \sum_{j=1}^i t_j.$$

Покажем, что u^* действительно минимальное значение среднего времени обслуживания. Для того чтобы показать, что u^* действительно минимально среди u для всех перестановок, достаточно показать, что применение к произвольной перестановке $(\alpha_1, \dots, \alpha_L)$ любой парной транспозиции, меняющей местами элементы α_k и α_l , где $t_{\alpha_l} \leq t_{\alpha_k}$ и $l > k$, может лишь уменьшить исходное значение u , соответствующее перестановке $(\alpha_1, \dots, \alpha_L)$, где α_i - номер задачи, назначаемой на решение i -й по порядку, $i=1, L$. Действительно, пусть задачи с номерами α_k и α_l поменялись местами. Тогда для полученной перестановки среднее время обслуживания равно

$$\begin{aligned} u' = & \sum_{j=1}^{k-1} \sum_{i=1}^j t_{\alpha_i} + \left(\sum_{i=1}^{k-1} t_{\alpha_i} + t_{\alpha_l} \right) + \sum_{j=k+1}^{l-1} \left(\sum_{i=1}^{k-1} t_{\alpha_i} + t_{\alpha_l} + \sum_{r=k+1}^j t_{\alpha_r} \right) + \left(\sum_{r=1}^{k-1} t_{\alpha_r} + t_{\alpha_l} + \sum_{r=k+1}^{l-1} t_{\alpha_r} + t_{\alpha_k} \right) + \\ & + \sum_{j=l+1}^L \left(\sum_{r=1}^{k-1} t_{\alpha_r} + t_{\alpha_l} + \sum_{r=k+1}^{l-1} t_{\alpha_r} + t_{\alpha_k} + \sum_{s=l+1}^j t_{\alpha_s} \right) = \\ & \left[\sum_{j=1}^{k-1} \sum_{i=1}^j t_{\alpha_i} + \sum_{i=1}^{k-1} t_{\alpha_i} + \sum_{j=k+1}^{l-1} \left(\sum_{i=1}^{k-1} t_{\alpha_i} + \sum_{r=k+1}^j t_{\alpha_r} \right) + \sum_{i=1}^{k-1} t_{\alpha_i} + \sum_{r=k+1}^{l-1} t_{\alpha_r} + \sum_{j=l+1}^L \left(\sum_{i=1}^{k-1} t_{\alpha_i} + \sum_{r=k+1}^{l-1} t_{\alpha_r} + \sum_{s=l+1}^j t_{\alpha_s} \right) \right] + \\ & t_{\alpha_l} + (l-l-k) * t_{\alpha_l} + t_{\alpha_l} + t_{\alpha_k} + (t_{\alpha_l} + t_{\alpha_k}) * (L-l) = C + t_{\alpha_l} * (L-k+1) + t_{\alpha_k} * (L-l+1). \end{aligned}$$

Нетрудно видеть, что

$$u = C + t_{\alpha_k}(L - k + 1) + t_{\alpha_l}(L - l + 1), \text{ а потому}$$

$$u' - u = t_{\alpha_l}(l - k) + t_{\alpha_k}(k - l) = (l - k)(t_{\alpha_l} - t_{\alpha_k}) \leq 0,$$

так как $l > k$, а $t_{\alpha_l} \leq t_{\alpha_k}$. Следовательно, перемещение вперед задачи с меньшим временем решения приводит к уменьшению среднего времени обслуживания. В перестановке $(1, \dots, L)$ при условии, что $t_1 \leq \dots \leq t_L$, нельзя сделать ни одной такой улучшающей транспозиции, а потому u^* есть минимальное среднее время обслуживания и алгоритм SPT дает оптимальное решение рассматриваемой задачи.

3.6.1.2. Алгоритм RR

В реальных системах оперативной обработки априорная информация о временах решения задач, как правило, отсутствует. Чтобы воспользоваться принципами планирования на основе алгоритма SPT, в систему вводятся средства, обеспечивающие выявление коротких и длинных работ непосредственно в ходе вычислительного процесса.

Простейшее правило планирования работ, обеспечивающее выполнение указанного требования, задается алгоритмом циклического обслуживания, или, иначе, алгоритмом RR (Round–Robin). Работа алгоритма иллюстрируется рисунком 3.10.



Рисунок 3.10 - Циклическое обслуживание заданий

Заявки на выполнение работ поступают с интенсивностью λ в очередь O , откуда они выбираются и исполняются процессором $Пр$. Для обслуживания отдельной заявки отво-

дится постоянный квант времени q , достаточный для выполнения нескольких тысяч операций. Если работа была выполнена за время q , она покидает систему. В противном случае она вновь поступает в конец очереди и ожидает предоставления ей очередного кванта процессорного времени.

Оценим среднее время ожидания и пребывания работ в системе с циклическим планированием. Воспользуемся для этого аппаратом теории массового обслуживания. Для упрощения последующих выкладок предположим, что длительность кванта - не постоянная величина, а случайная, распределенная по экспоненциальному закону с тем же средним значением q . Примем также, что на вход системы поступает простейший поток с интенсивностью λ работ в единицу времени, и с вероятностями σ или $(1-\sigma)$ работа не будет или соответственно будет завершена в текущем кванте. Из последнего предположения следует, что вероятность того, что работа будет выполнена точно за k квантов (не завершена за первые $k-1$ квантов и завершена в k -том варианте), описывается геометрическим распределением

$$p_k = \sigma^{k-1} (1-\sigma), \quad k=1, 2, \dots$$

со средним

$$\bar{n} = \sum_{k=1}^{\infty} k \sigma^{k-1} (1-\sigma) = (1-\sigma) \left(\sum_{k=0}^{\infty} \sigma^k \right)' = (1-\sigma) \left(\frac{1}{1-\sigma} \right)' = \frac{(1-\sigma)}{(1-\sigma)^2} = \frac{1}{1-\sigma}$$

Таким образом, если известны средняя трудоемкость работ Θ и длительность q кванта, то среднее количество \bar{n} квантов, с одной стороны, равно Θ/q , а с другой – полученному выше выражению, то есть

$$\frac{1}{1-\sigma} = \frac{\theta}{q}, \quad \text{откуда} \quad \sigma = 1 - \frac{q}{\theta}.$$

Определим среднее время ответа для работы J , требующей ровно τ единиц времени обработки. Пусть m – наименьшее целое, при котором $m_q \geq \tau$ (то есть m – число квантов, достаточное для обслуживания заявки). Рассмотрим состояние системы на момент поступления работы J . При поступлении

работы J в системе в среднем находится W_1 других работ. Значение N_1 определяется как среднее число заявок в системе с беспriorитетным обслуживанием, на вход которой заявки поступают с интенсивностью

$$\Lambda = \lambda + \lambda\sigma + \lambda\sigma^2 + \dots + \lambda\sigma^n + \dots = \lambda/(1-\sigma)$$

(с учетом интенсивности поступления заявок на дообслуживание в последующих тактах) и обслуживаются по экспоненциальному закону со средним q . Для определения W_1 требуется найти распределение вероятностей $\{p_k\}$ того, что в очереди будет ровно k заявок. Тогда

$$W_1 = \sum k p_k.$$

Для определения p_k составим систему дифференциальных уравнений Колмогорова с помощью графа переходов (рисунок 3.11).

$$(\mu = 1/q)$$

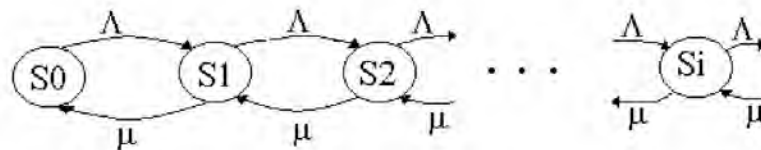


Рисунок 3.11 – Граф состояний (переходов)

Здесь S_i – состояние системы с i заявками в очереди (одна из них обслуживается). Тогда система уравнений имеет вид

$$p'_i = \Lambda p_{i-1} - \mu p_i - \Lambda p_i + \mu p_{i+1},$$

или

(

$$\left\{ \begin{array}{l} p'_i = \Lambda p_{i-1} - (\mu + \Lambda) p_i + \mu p_{i+1}, \quad i=1,2,\dots \\ p'_0 = -\Lambda p_0 + \mu p_1. \end{array} \right.$$

$$\left\{ \begin{array}{l} p'_0 = -\Lambda p_0 + \mu p_1. \end{array} \right.$$

Кроме того, требуется соблюдение условия нормировки

$$\sum_{i=0}^{\infty} p_i = 1.$$

Рассмотрим установившийся режим, то есть считаем, что $t \rightarrow \infty$. Тогда вместо дифференциальных уравнений получаем алгебраические

$$\Lambda p_{i-1} - (\mu + \Lambda) p_i + \mu p_{i+1} = 0, \quad i=1, 2, \dots$$

$$\Lambda p_0 + \mu p_1 = 0,$$

$$p_0 + p_1 + \dots + p_i + \dots = 1.$$

Из второго уравнения выразим p_1 через p_0 :

$$p_1 = \Lambda / \mu * p_0 ;$$

подставим это значение в первое уравнение с $i=1$. Тогда

$$\mu * p_2 = \frac{(\Lambda + \mu)\Lambda}{\mu} p_0 - \Lambda p_0 \Rightarrow p_2 = \frac{\Lambda^2}{\mu^2} p_0$$

Делаем индуктивное предположение: $p_k = \left(\frac{\Lambda}{\mu}\right)^k p_0$. Тогда из k -го уравнения получаем

$$\begin{aligned} \mu * p_{k+1} &= (\Lambda + \mu) * \left(\frac{\Lambda}{\mu}\right)^k p_0 - \Lambda * \left(\frac{\Lambda}{\mu}\right)^{k-1} p_0 = \frac{\Lambda^{k+1}}{\mu^k} p_0 + \frac{\Lambda^k}{\mu^{k-1}} p_0 - \frac{\Lambda^k}{\mu^{k-1}} p_0 = \\ &= \frac{\Lambda^{k+1}}{\mu^k} p_0, \end{aligned}$$

откуда $p_{k+1} = \left(\frac{\Lambda}{\mu}\right)^{k+1} p_0$, что подтверждает индуктивное предположение. Поэтому в соответствии с условием нормировки получим

$$\sum_{k=0}^{\infty} \left(\frac{\Lambda}{\mu}\right)^k p_0 = 1 \Rightarrow p_0 \frac{1}{1 - \rho} = 1 \Rightarrow p_0 = 1 - \rho \Rightarrow p_k = (1 - \rho) \rho^k,$$

где

$$\rho = \frac{\Lambda}{\mu} = \Lambda * q = \frac{\lambda * q}{1 - \sigma}.$$

На основании полученного выражения

$$W_1 = \sum_{k=1}^{\infty} k p_k = \sum_{k=1}^{\infty} k (1 - \rho) \rho^k = \rho (1 - \rho) \sum_{k=1}^{\infty} k \rho^{k-1} = \frac{\rho (1 - \rho)}{(1 - \rho)^2} = \frac{\rho}{1 - \rho}.$$

С учетом допущения об экспоненциальности распределения кванта q время дообслуживания работы J не зависит от этого момента и в среднем равно q . Таким образом, работа будет ожидать W_{1q} единиц времени до получения первого кванта. За время W_{1q} и первый квант выполнения работы J в систему поступит λ новых работ. Кроме того, σW_1 работ из их общего числа W_1 вернутся обратно в очередь на довыполнение. Поэтому в следующем цикле работа J застанет в системе W_2 работ:

$$W_2 = \lambda W_{1q} + \lambda q + \sigma W_1.$$

Подставляя в последнее выражение ранее полученное значение W_1 , получаем

$$\begin{aligned} W_2 &= \frac{\lambda q \rho}{(1-\rho)} + \lambda q + \frac{\sigma \rho}{(1-\rho)} = \frac{\lambda q \rho + \lambda q - \lambda q \rho + \sigma \rho}{1-\rho} = \frac{\lambda q + \sigma \rho}{1-\rho} = \\ &= \frac{\rho - \rho \sigma + \rho \sigma}{1-\rho} = \frac{\rho}{1-\rho}, \end{aligned}$$

$$\rho = \frac{\lambda q}{1-\sigma}$$

В общем случае аналогично получаем

$$W_i = \lambda W_{i-1} q + \lambda q + \sigma W_{i-1} = \rho / (1-\rho).$$

Среднее время ожидания для работы J , время выполнения которой составляет m квантов, равно

$$\omega_m = q \sum_{i=1}^m W_i = m q \rho / (1-\rho).$$

Здесь среднее время ожидания ω_m определяется как время, необходимое для обслуживания всех W_i работ, стоящих впереди работы J в каждом из m циклов обслуживания. Среднее время ответа для работы J

$$U_m = \omega_m + m q = m q / (1-\rho),$$

где $\rho = \lambda q / (1-\sigma) = \lambda q / (q / \Theta) = \lambda \Theta$ – загрузка системы. Из выражения для времени ожидания ω_m видно, что время ожидания обслуживания возрастает с увеличением трудоемкости $\tau = m q$ задачи. В то же время при обслуживании задач в поряд-

ке поступления без прерываний среднее время ожидания ω не зависит от трудоемкости и составляет

$$\omega = \theta \frac{\rho'}{1 - \rho'} = \frac{\theta * \lambda \theta}{1 - \lambda \theta} = \frac{\lambda \theta^2}{1 - \lambda \theta},$$

где $\rho' = \frac{\lambda}{(1/\theta)} = \lambda \theta$, θ – средняя трудоемкость.

Сравним ω и ω_m :

$$\omega_m - \omega = \frac{mq\rho}{1 - \rho} - \frac{\theta\rho'}{1 - \rho'} = \frac{mq\lambda\theta}{1 - \lambda\theta} - \frac{\theta\lambda\theta}{1 - \lambda\theta} = \frac{\lambda\theta}{1 - \lambda\theta}(mq - \theta).$$

Из последнего выражения видно, что время ожидания длинных задач ($mq > \theta$) больше, чем при обслуживании в порядке поступления, а время ожидания коротких задач ($mq < \theta$) – меньше времени ожидания в режиме без прерываний.

3.6.1.3. Алгоритм FB

Для обеспечения еще более быстрой реакции системы на короткие работы в системах оперативной обработки используются алгоритмы многоуровневого циклического планирования. Одним из таких алгоритмов является алгоритм FB (foreground-background). Принцип его работы можно проиллюстрировать следующей схемой (3.12):

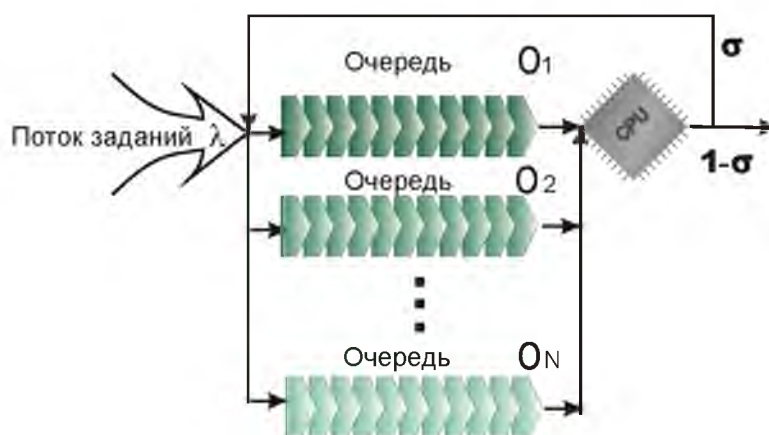


Рисунок 3.12 - Многоуровневая циклическая обработка заданий

Заявки на выполнение работ поступают в очередь O_1 . Работы, стоящие в очереди O_1 , получают квант процессорного времени q . Если за это время работа была выполнена, то она покидает систему. В противном случае заявка на работу переносится в очередь O_2 , откуда она может быть занесена в очереди O_3, O_4, \dots, O_n . Очереди обслуживаются в следующем порядке. Если имеется хотя бы одна заявка в очереди O_1 , то эта заявка непременно обслуживается. Заявки из очереди O_2 обслуживаются при условии, что нет заявок в очереди O_1 . Аналогично заявки из очереди O_m обслуживаются только в том случае, если все очереди O_1, \dots, O_{m-1} пусты. Заявка, достигшая последней очереди O_n , остается в ней до полного завершения работы.

Применяются модификации алгоритма FВ, различающиеся по величине квантов времени, предоставляемых заявкам из разных очередей. Возможно планирование на основе постоянной величины кванта или с использованием квантов переменной длительности, которая возрастает по мере увеличения номера очереди. Одна из таких модификаций – алгоритм планирования FВ с учетом приоритетов работ. Работы, поступающие в систему, разделяются в зависимости от приоритетов $1, \dots, n$ на n потоков $\lambda_1, \dots, \lambda_n$. Приоритеты задач относительны, то есть поступление в систему заявки более высокого приоритета не прерывает процесс обработки менее приоритетных заявок, но при освобождении ресурса более приоритетные заявки будут назначены в первую очередь. Работы с высшим приоритетом поступают в очередь O_1 , а работы с низким приоритетом – в очередь O_n . Работам, выбираемым на обслуживание из разных очередей, выделяются кванты времени различной длительности, причем заявкам из очереди O_m выделяется больший по продолжительности квант времени, чем заявкам из очереди O_{m-1} , $m = 2, n$.

Приоритеты работам могут назначаться исходя из трудоемкости последних. Если трудоемкости работ известны хо-

тя бы приближенно, то работам с большой трудоемкостью присваиваются низкие приоритеты и они сразу же поступают в очереди соответствующего приоритета, в которых получают большие кванты времени. В результате этого трудоемкие работы не будут задерживать процесс выполнения менее трудоемких работ. Если трудоемкость работы была занижена, то есть ее приоритет оказался завышен, то после окончания выделенного для нее кванта времени работа переместится в очередь следующего, более низкого приоритета.

Алгоритм планирования с учетом приоритетов очень эффективен для систем с ограниченной емкостью оперативной памяти, не позволяющей разместить в ней программы всех работ, выполняемых системой. В таком случае в оперативной памяти размещается только небольшая часть программ, а остальные программы хранятся во внешней памяти — на магнитном диске. Все программы циклически обслуживаются предоставлением им кванта процессорного времени, поэтому они вызываются в оперативную память поочередно, а получив квант обслуживания, удаляются из нее во внешнюю память. Процесс циклического завершения программ в оперативной памяти называется *свопингом*. Если система работает со свопингом и все без исключения работы поступают в первую очередь, причем всем очередям выделяются одинаковые кванты времени, то затраты ресурсов системы на свопинг крайне большие. Для уменьшения непроизводительных затрат целесообразно трудоемкие работы сразу же размещать в очередях с низкими приоритетами и выделять им большие по длительности кванты времени.

Например, операционная система для малых ЭВМ ОСРВ обеспечивает две процедуры разделения времени — циклическое планирование и вытеснение на диск. Процедура циклического планирования через заданный интервал времени циклически перемещает указатель задач в таблице задач системы STD (System Task Directory) и объявляет значимое событие, в результате обработки которого происходит перепланирова-

ние задач. Планировщик выбирает для выполнения первую задачу из STD. Обычно интервал времени циклического планирования устанавливается равным 0,1с.

Процедура вытеснения на диск перемещает временно на диск часть задач из основной памяти, освобождая тем самым место для более приоритетных задач. Необходимые условия для вытеснения задачи:

- задача должна быть установлена как вытесняемая;
- на диске есть более приоритетная задача, для которой нет места в основной памяти;
- задача не имеет незавершенных запросов ввода-вывода (кроме запроса ввода с терминала).

При выполнении процедур вытеснения на диск записываются область занимаемой задачей основной памяти и информация о текущем состоянии задачи, необходимая для продолжения ее работы. Разделение ресурсов задачами базируется на периодическом уменьшении приоритетов задач, находящихся в основной памяти, и как только приоритет задачи в основной памяти становится меньше приоритета задачи на диске, выполняется процедура вытеснения.

Приоритетность программ для систем со свопингом может назначаться в соответствии с алгоритмом Корбатто. Здесь априорно принимается следующее предположение: программы с большей длиной – более трудоемкие.

Исходя из этого предположения приоритеты программам присваиваются на основе формулы

$$p = \lfloor \log_2 (L_n / L_q + 1) \rfloor ,$$

где $[x]$ – целая часть X ; L_n – длина программы в байтах;

L_q – число байтов, передаваемых между оперативной и внешней памятью за время q , равное минимальной длительности кванта.

Отношение L_n / L_q определяет число квантов времени, необходимых для загрузки программы в оперативную память и для вывода ее из оперативной памяти. Работа с приорите-

том p заносится в очередь O_p . Очередям с номерами $p = 1, \dots, n$ выделяются кванты времени длительности

$$q_p = 2^{p-1} q,$$

где q – квант времени, выделяемый для работ из очереди O_1 .

Таким образом, очередям $O_1, O_2, O_3, O_4, \dots$ соответствуют кванты времени $q, 2q, 4q, 8q, \dots$. Увеличение кванта времени на выполнение работ с большой трудоемкостью способствует сокращению числа прерываний работы процессора и числа пересылок программ между оперативной и внешней памятью.

3.6.2. Планирование вычислительного процесса

3.6.2.1. Методы управления ресурсами многопроцессорных систем при обработке пакетов задач с прерываниями

Алгоритм Макнотона

Рассмотрим систему с n идентичными процессорами, на которой необходимо решить L независимых задач; каждая задача решается одним процессором в течение времени t_i , $i = 1, \dots, L$. Требуется найти алгоритм, который для каждого данного пакета (набора) строил бы расписание решения задач на процессорах системы, обеспечивающее минимально возможное время решения. При этом достигается максимально возможная производительность системы. Например, в случае 2-процессорной системы и набора задач с временами (3,3,2,2,2) возможны различные варианты назначения задач на решение. Приведем некоторые из них (рисунок 3.13).

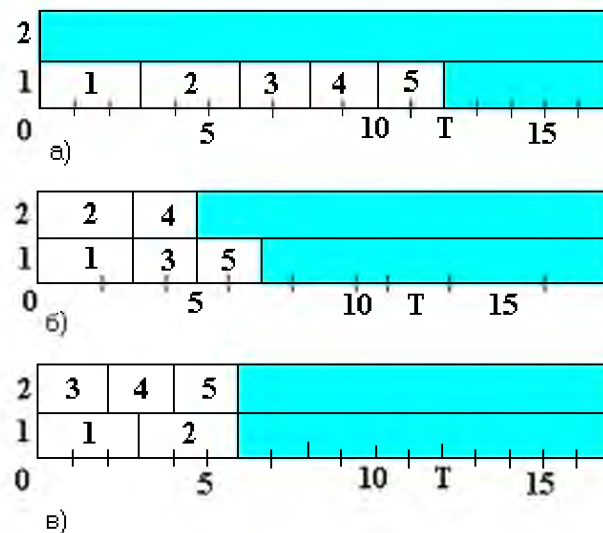


Рисунок 3.13 – Варианты расписаний решения задач

Очевидно, что наилучший в смысле минимизации общего времени решения задач - вариант в), для которого время T решения пакета задач совпадает с соответствующим оптимальным значением T_0 и в данном случае равно величине

$$\theta = \max \{ \max t_i, 1/n * \sum t_i \}$$

Величина θ является нижней границей для оптимального значения T_0 . Действительно, длина T любого расписания не может быть меньше ни $\max t_i$ - максимального из времен решения задач пакета Π , ни величины $(1/n * \sum t_i)$, дающей длину расписания в том случае, когда до момента завершения решения последней из задач пакета ни один процессор не простаивает, то есть все процессоры имеют 100% загрузки.

В общем случае даже при $n = 2$ задача поиска оптимального значения T при условии решения задач, является NP-трудной, то есть все известные алгоритмы ее решения имеют трудоемкость, экспоненциально зависящую от L . Однако, если допустить возможность прерывания решения задач пакета до завершения их обслуживания, то могут быть предложены полиномиально-трудоемкие алгоритмы, приводящие к расписанию оптимальной длины T_0 . При этом считается, что после прерывания решение задачи может быть возобновлено с точки прерывания на любом процессоре, не обязательно на том,

на котором она первоначально решалась. Число прерываний должно быть по возможности меньшим, так как с каждым актом прерывания связаны потери машинного времени на загрузку-выгрузку задач из оперативной памяти.

Рассмотрим предложенный Макнотом в 1959 г. алгоритм построения оптимального по длине расписания с не более чем $n-1$ прерываниями.

Алгоритм Макнотона заключается в предварительном упорядочении задач по убыванию времени решения и назначении задач последовательно по порядку номеров одну за другой на процессоры системы справа налево от уровня θ .

Пример

$n=2, L=4$, времена решения задач: $(5, 4, 3, 2)$. Тогда

$$\theta = \max \{ 5, 1/2 * (5+4+3+2) \} = 7.$$

И расписание, полученное в соответствии с алгоритмом Макнотона, имеет следующий вид (рисунок 3.14):

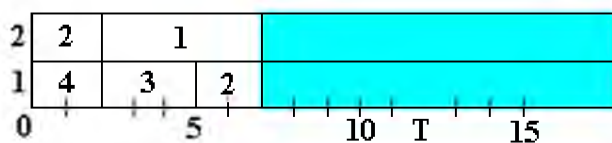


Рисунок 3.14

В данном случае число прерываний равно единице.

Покажем, что $n - 1$ (максимальное число прерываний для расписания, полученного в соответствии с алгоритмом Макнотона) является достижимой границей числа прерываний.

Для доказательства этого построим такой пример пакета задач, для которого алгоритм Макнотона дает расписание с числом прерываний $n-1$.

Пусть $L=n+1$ и $t_i = n, i = 1, \dots, n+1$.

Тогда $\theta = \max \{ n, 1/n * (n+1)*n \} = n+1$, а расписание, получаемое в соответствии с алгоритмом Макнотона, имеет вид (рисунок 3.15):

n	2	1
n-1	3	2
n-2	4	3
	⋮	
2	n	n-1
1	n-1	n

Рисунок 3.15 – Расписание по алгоритму Макнотона

$$T=n+1=\theta$$

Число прерываний в этом случае, как видно, равно $n-1$, что и требовалось показать. Покажем теперь, что любое оптимальное расписание для этого пакета задач также имеет не менее $n-1$ прерываний. Очевидно, что в любом оптимальном расписании ни один процессор не простаивает на интервале $[0, n+1]$. Предположим, что существует некоторое оптимальное расписание с числом прерываний, меньшим $n-1$. Тогда по крайней мере 2 процессора (предположим для определенности P_k и P_l) обслуживают заявки без прерываний. Очевидно эти процессоры обслуживают некоторые задачи Z_{ik} и Z_{il} в интервале $[0, n]$ без прерываний (если решение этих задач начинается позже момента времени $t=0$, значит до этого момента на этих процессорах решались некоторые другие задачи, решение которых прерывается в моменты начала решения задач Z_{ik} и Z_{il}). Найдутся моменты времени t, t' , такие, что $n \leq t < t' \leq n+1$, и в интервале $[t, t']$ хотя бы один процессор простаивает, а потому рассматриваемое решение не может быть оптимальным.

Так как мы пришли к противоречию, делаем вывод о том, что предположение о числе прерываний, меньшем $n-1$, в оптимальном расписании ложно.

3.6.2.2. Управление ресурсами многопроцессорных систем при обработке пакетов независимых задач без прерываний

Алгоритм LPT

Рассмотрим систему, содержащую n идентичных процессоров, на которой необходимо решить без прерываний набор из L независимых задач с временами решения $t_{ij=1,...,L}$. Получение расписания с минимальным временем решения и в этом случае является NP-трудной задачей. Один из наиболее эффективных и нетрудоемких алгоритмов организации вычислений в этом случае – **алгоритм LPT (*longest-processing task first*)** - самая длинная задача решается первой), являющийся частным случаем алгоритма критического пути для независимых задач. Суть этого алгоритма заключается в назначении задач в порядке убывания времени решения на освобождающиеся процессоры. Сотрудником фирмы Bell Laboratories США, Грэхемом в 1967г. был получен следующий результат.

При использовании алгоритма LPT для распределения любого пакета $\Pi = \{Z_i\}$ независимых задач без прерываний в системе с n идентичными процессорами справедливо:

$$T \leq (4/3 - 1/3n) * T_0,$$

где T - время решения пакета Π при распределении задач алгоритмом LPT,

T_0 - длина соответствующего оптимального расписания.

$$\text{Очевидно, } T_0 \geq \frac{1}{n} * \sum_{i=1}^l t_i.$$

Приведенная оценка является наилучшей.

3.6.3.Производительность мультипроцессорных систем с общей и индивидуальной памятью (режимы разделения нагрузки и разделения функций)

Для увеличения производительности в состав ВС может вводиться несколько процессоров, способных функционировать параллельно во времени и независимо друг от друга и наряду с тем взаимодействовать между собой и с другим оборудованием системы. ВС, содержащие несколько процессоров, связанных между собой и с общим для них комплек-

том внешних устройств, называются мультипроцессорными системами (МПС).

Производительность МПС увеличивается по сравнению с однопроцессорной системой в результате того, что мультипроцессорная организация создает возможность для одновременной обработки нескольких задач или параллельной обработки различных частей одной задачи.

В ряде случаев требуется обеспечить непрерывность функционирования системы во времени. Это означает, что отказ в любом устройстве ВС, в том числе и в процессоре, не должен приводить к катастрофическим последствиям, то есть система должна сохранять работоспособность и после отказа. В таком случае все устройства ВС должны быть по крайней мере задублированы и система должна содержать не менее двух процессоров, то есть строиться, как МПС.

Наиболее существенен в структурной организации МПС способ связи между процессорами и памятью системы. В этом аспекте МПС разделяются на МПС с памятью общей (полнодоступной) и индивидуальной (раздельной).

3.6.3.1. МПС с общей памятью

В МПС с общей памятью каждый из процессоров имеет доступ к любому модулю памяти, которые могут функционировать независимо друг от друга и в каждый момент времени могут выполняться одновременные обращения для записи или чтения слова информации, число которых определяется числом модулей. Конфликтные ситуации (обращение к одному и тому же модулю памяти) разрешаются коммутатором, начинающим обслуживать первым устройство с наибольшим приоритетом, например процессор с наименьшим номером. Каждый из процессоров может инициировать работу любого канала ввода/вывода.

Структура МПС с общей памятью наиболее универсальна: любая информация, хранимая в памяти системы, в равной степени доступна любому процессору и каналу ввода/вывода. Отрицательное свойство МПС с общей памятью – большие

затраты оборудования в коммутаторах (эти затраты пропорциональны произведению числа устройств, подключенных к памяти, и числа модулей памяти).

3.6.3.1. Характеристики МПС с общей памятью

Будем рассматривать мультипроцессорную систему (МПС) с общей памятью, в которой размещаются все программы и данные, используемые в процессе функционирования системы. Такая организация типична для управляющих систем, жесткие ограничения на время реакции которых исключают возможность размещения информации во внешней памяти. Будем считать, что в МПС используются одинаковые процессоры, то есть МПС – однородная система. Наличие общей оперативной памяти, в которой размещается вся необходимая информация, и однородность системы позволяют выполнять любую программу на любом процессоре, то есть любой процессор может принять на обслуживание любую заявку. Режим работы МПС, при котором каждый из процессоров может обслуживать любую заявку, называется *режимом разделения нагрузки*. При этом режиме каждый из N процессоров принимает на обслуживание N -ю часть заявок, то есть N -ю часть общей нагрузки.

Модель МПС с общей памятью. Процесс обслуживания заявок в режиме разделения нагрузки можно рассматривать как процесс функционирования одной многоканальной системы массового обслуживания (рисунок 3.16) с интенсивностью λ входящего потока, общей очередью O , заявки из которой выбираются в порядке поступления их в систему, и средней длительностью обслуживания заявки каждым из процессоров Pr_1, \dots, Pr_N равной ϑ . Заявка, поступающая в систему, содержащую N процессоров, при наличии хотя бы одного свободного процессора немедленно принимается последним на обслуживание. Если все процессоры заняты обслуживанием ранее поступивших заявок, поступающая заявка размещается в очереди.

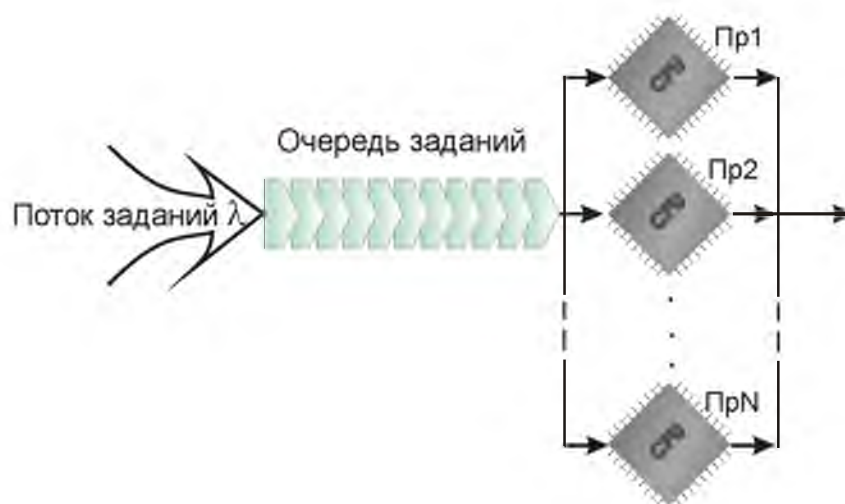


Рисунок 3.16 - МПС с общей памятью

Определим характеристики МПС на основе модели рис. 3.16.

Пусть в МПС поступает M потоков с интенсивностями $\lambda_1, \dots, \lambda_M$. Обслуживание заявок сводится к выполнению соответствующих программ, средние трудоемкости которых равны $\Theta_1, \dots, \Theta_M$ операций в расчете на один прогон программы. Примем, что обслуживание заявок выполняется на основе дисциплины FIFO. В таком случае можно считать, что система обслуживает однородный поток заявок, поступающих с интенсивностью

$$\Lambda = \sum_{i=1}^M \lambda_i \quad (3.45)$$

Для обслуживания любой заявки из суммарного потока требуется в среднем

$$\Theta = \sum_{i=1}^M (\lambda_i / \Lambda) * \Theta_i \quad (3.46)$$

процессорных операций.

Примем, что заявка, поступившая на обслуживание, захватывает процессор до полного завершения обслуживания. В таком случае средняя длительность обслуживания заявки процессором с быстродействием V равна $\vartheta = \Theta / V$ и интенсивность обслуживания заявок одним процессором $\mu = 1 / \vartheta$.

Параметры системы Λ , N и $\vartheta = \Theta / V$ должны отвечать условию существования стационарного режима, при котором

в очереди пребывает конечное число заявок и, следовательно, конечны времена ожидания и пребывания заявок. На каждый из процессоров поступает N -я доля заявок и, следовательно, отдельный процессор обслуживает поток с интенсивностью Λ/N . Загрузка процессора

$$\rho = (\Lambda/N) * \vartheta = \Lambda/(N * \mu) = \Lambda/\mu_{\Sigma} \quad (3.47)$$

где $\mu_{\Sigma} = N * \mu$ - суммарная интенсивность обслуживания заявок N -процессорной системой.

Стационарный режим существует, если $\rho < 1$. Следовательно, параметры МПС должны отвечать соотношению $\lambda\vartheta/(NB) < 1$.

Характеристики системы можно получить в явной аналитической форме, если принять предположение о том, что входящий поток заявок – пуассоновский и длительность обслуживания распределена по экспоненциальному закону со средним ϑ .

В теории массового обслуживания доказывается, что при указанных предположениях вероятность пребывания в системе $N=0,1,2,\dots$ заявок, обслуживаемых процессорами и стоящих в очереди

$$p_n = \begin{cases} p_0 \frac{R^n}{n!} & \text{при } 0 \leq n \leq N; \\ p_0 \frac{R^n}{N!N^{n-N}} & \text{при } n > N, \end{cases} \quad (3.48)$$

где

$$p_0 = \left[\frac{R^N}{(N-1)!(N-R)} + \sum_{n=0}^{N-1} \frac{R^n}{n!} \right]^{-1} \quad (3.49)$$

- вероятность того, что в системе нет ни одной заявки, то есть все N процессоров простаивают; R – суммарная нагрузка N -канальной системы, равная

$$R = \Lambda / \mu = N \Lambda / N \mu = N \rho. \quad (3.50)$$

Суммарная нагрузка R в отношении N -канальной системы массового обслуживания определяет среднее число каналов, которые заняты обслуживанием заявок. Для стационарного режима $R < N$. С учетом (3.50) выражения (3.48) и (3.49) можно представить в виде:

$$p_n = \begin{cases} p_0 \frac{N^n}{n!} \rho^n & \text{при } 0 \leq n \leq N; \\ p_0 \frac{N^N}{N!} \rho^n & \text{при } n > N, \end{cases} \quad (3.51)$$

$$p_0 = \left[\frac{N^{N-1} * \rho^N}{(N-1)!(1-\rho)} + \sum_{n=0}^{N-1} \frac{N^n \rho^n}{n!} \right]^{-1} \quad (3.52)$$

где $\rho = \Lambda / (N \mu)$ – нагрузка процессора N -процессорной системы.

Характер изменения вероятностей P_n при изменении суммарной нагрузки четырехпроцессорной системы представлен на рисунке 3.17.

Распределение числа заявок в системе носит унимодальный характер, причем с увеличением нагрузки максимальное значение P_n сдвигается в сторону больших N . Распределение (3.51) содержит всю информацию, необходимую для определения характеристик МПС. Среднюю длину очереди заявок, ожидающих обслуживания в N -процессорной системе, находим исходя из (3.51) как математическое ожидание случайной величины $i = n - N > 0$, равной числу заявок в очереди:

$$\begin{aligned}
l &= \sum_{i=1}^{\infty} i * p_{N+i} = \sum_{i=1}^{\infty} i * p_0 \frac{N^N}{N!} \rho^{N+i} = p_0 \frac{N^N}{N!} \rho^N \sum_{i=1}^{\infty} i * \rho^i = \\
&= p_0 \frac{N^N}{N!} \rho^N \sum_{i=1}^{\infty} \sum_{j=i}^{\infty} \rho^j = \frac{N^{N-1} * \rho^{N+1}}{(N-1)!(1-\rho)^2} p_0, \quad (3.53)
\end{aligned}$$

где p_0 определяется (3.52).

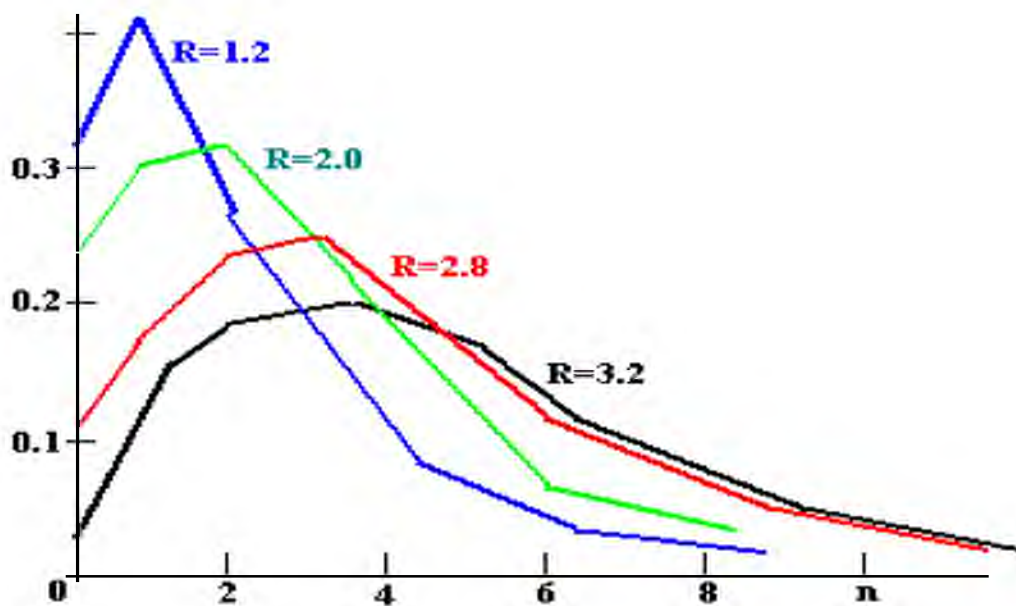


Рисунок 3.17 - Характер изменения вероятностей в зависимости от суммарной загрузки R

Среднее число заявок, пребывающих в системе
 $m=1+R,$ (3.54)

где 1 – среднее число заявок, находящихся в очереди и определяемое (3.53); R – суммарная нагрузка МПС, определяемая из формулы (3.50).

Для систем без потерь заявок среднее время ожидания и среднее время пребывания заявок в системе равны соответственно $w=1/\lambda$ и $u=m/\lambda$. Подставляя в эти соотношения выражения (3.53) и (3.54), получим:

$$w = \frac{N^{N-1} \rho^N}{\mu N! (1 - \rho)^2} P_0; \quad (3.55)$$

$$u = w + \mathcal{G} = w + 1 / \mu \quad (3.56)$$

или с использованием (3.50)

$$w = \frac{R^N}{\mu (N - 1)! (N - R)^2} P_0 \quad (3.57)$$

Одна из важных характеристик системы – вероятность ненулевого ожидания заявок $P_r (w > 0)$, то есть вероятность того, что в момент поступления очередной заявки все N процессоров заняты обслуживанием. Эта вероятность

$$P_r (w > 0) = P_r (n \geq N) = \sum_{n=N}^{\infty} p_n = p_0 \frac{N^N}{N!} \sum_{n=N}^{\infty} \rho^n = \frac{N^N \rho^N}{N! (1 - \rho)} P_0. \quad (3.58)$$

Из сравнения (3.55) и (3.58) вытекает следующее выражение для среднего времени ожидания заявок:

$$w = P_r (w > 0) / [\mu N (1 - \rho)]. \quad (3.59)$$

В свою очередь, вероятность нулевого ожидания заявок, то есть вероятность того, что в момент поступления заявки хотя бы один процессор свободен, равна

$$P_r (w = 0) = 1 - P_r (w > 0).$$

3.6.3.2. МПС с индивидуальной памятью

В МПС с индивидуальной памятью каждый из процессоров обращается в основном к своему модулю памяти. Для обмена данными между подсистемами «процессор – модуль памяти» в процессорах предусмотрены блоки обмена, обеспечивающие передачу сегментов информации между общей памятью и модулем памяти. При этом блок обмена может работать как селекторный канал: операция обмена инициируется процессором и передача данных выполняется с параллельной работой последнего. Принцип индивидуальной памяти позволяет исключить коммутаторы в интенсивно используемом канале «процессор – модуль памяти», вследствие чего увеличивается номинальное быстродействие процессоров и

уменьшаются затраты оборудования по сравнению с общей памятью. Отрицательным последствием разделения памяти между процессорами является потеря ресурсов быстродействия в процессе обмена информацией между модулями памяти и общей памятью системы. Потери возникают, во-первых, из-за возможных приостановок работы процессоров для ожидания моментов окончания обмена данными с общей памятью и, во-вторых, из-за дополнительной загрузки модулей памяти операциями обмена.

Если класс задач, решение которых возлагается на МПС, таков, что работа каждого процессора связана с использованием в основном ограниченного подмножества данных и обращение к остальным данным происходит сравнительно редко, то индивидуализация памяти приводит к экономии оборудования и обеспечивает высокое номинальное быстродействие процессоров в системе. В противном случае, когда каждый из процессоров почти равновероятно обращается к любому сегменту данных, МПС должна строиться по схеме с общей памятью, исключающей необходимость в обмене информацией между модулями памяти.

3.6.3.2. Характеристики МПС с индивидуальной памятью

В МПС с индивидуальной памятью множество программ обслуживания и связанных с ними данных $P = \{P_1, \dots, P_M\}$ разделяется на подмножества Q_1, \dots, Q_N , $Q_i = \{P_{\alpha_1}, \dots, P_{\alpha_{n_i}}\} \subseteq P$ ($i = 1, \dots, N$), размещаемые в памяти соответствующих процессоров Pr_1, \dots, Pr_N . В результате этого каждый из процессоров ориентируется на обслуживание заявок определенных типов, а именно тех, программы обслуживания которых размещены в памяти процессора. Режим работы МПС, при котором каждый из процессоров обслуживает заявки определенных типов и не может обслуживать заявки других типов, называется *режимом разделения функций*.

Модель МПС с индивидуальной памятью.

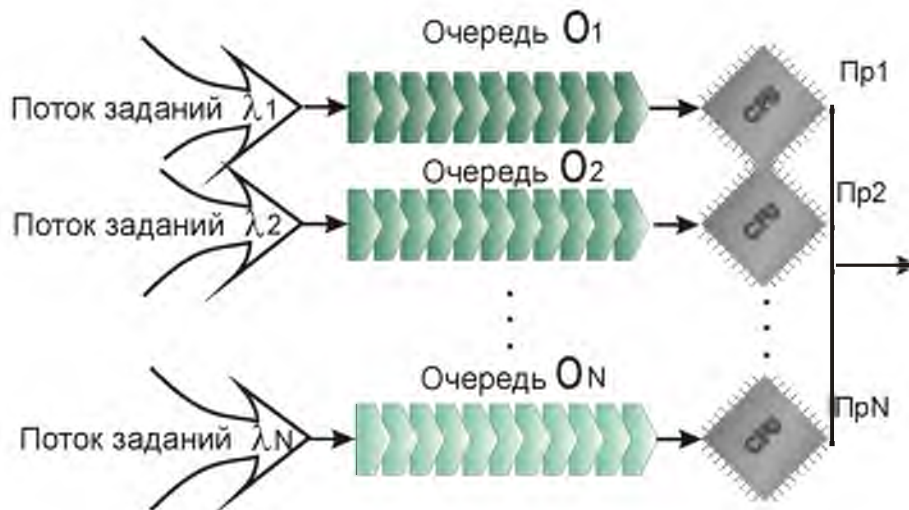


Рисунок 3.18 - МПС с индивидуальной памятью

В наиболее простом случае процессоры обмениваются информацией с общей памятью или количество информации, передаваемой при обменах, может быть столь незначительно, что допустимо пренебречь влиянием процессов обмена на процесс обслуживания заявок. В таком случае можно считать, что процессоры функционируют независимо и работу N-процессорной системы в режиме разделения функций можно рассматривать как процесс функционирования N одноканальных систем массового обслуживания (рисунок 3.18). Каждая из систем массового обслуживания состоит из потока заявок, поступающих с интенсивностью λ_i , очереди O_i и процессора Pr_i .

Для этой модели характеристики обслуживания заявок каждого типа могут быть вычислены в предположении, что входящие потоки – пуассоновские, при произвольном распределении длительностей обслуживания и различных дисциплинах обслуживания заявок. В частности, при экспоненциальном распределении длительности обслуживания и дисциплине FIFO среднее время ожидания заявок в системе с номером $i = 1, \dots, N$ и загрузкой $\rho_i = \lambda_i / \mu_i < 1$ равно

$$w_i = [\rho_i / (1 - \rho_i)] g_i, \quad (3.60)$$

среднее время пребывания заявок

$$u_i = w_i + g_i = [1/(1 - \rho_i)]g_i, \quad (3.61)$$

среднее число заявок в очереди $l_i = w_i \cdot \lambda_i = \rho_i^2 / (1 - \rho_i)$
и среднее число заявок в системе $m_i = u_i \cdot \lambda_i = \rho_i / (1 - \rho_i)$.

МПС как целый объект обслуживает суммарный поток заявок, поступающий на вход системы с интенсивностью

$$\Lambda = \sum_{i=1}^N \lambda_i$$

Заявка из суммарного потока с вероятностью λ_i / Λ будет ожидать обслуживания в среднем W_i единиц времени. С учетом этого среднее время ожидания заявки из суммарного потока определяется выражением

$$W = \sum_{i=1}^N (\lambda_i / \Lambda) w_i \quad (3.62)$$

Аналогично среднее время пребывания заявки в системе

$$U = \sum_{i=1}^N (\lambda_i / \Lambda) u_i \quad (3.63)$$

Рассмотрим случай, когда каждый из процессоров обслуживает точно N -ю часть суммарного потока заявок и средняя длительность обслуживания одинакова для всех процессоров и равна g . В таком случае $\lambda_1 = \dots = \lambda_N = \Lambda / n$ и $\rho_1 = \dots = \rho_N = \rho$. При равномерном распределении нагрузки из (3.62) и (3.60), а также из (3.63) и (3.61) следует, что средние времена ожидания и пребывания заявок равны соответственно

$$W = [\rho / (1 - \rho)]g; \quad (3.64)$$

$$U = [1 / (1 - \rho)]g; \quad (3.65)$$

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Что такое Марковский случайный процесс?
2. Что называют потоками случайных событий?

3. Какие функциональные зависимости описывают уравнения Колмогорова?
4. Нарисуйте граф схемы «гибели и размножения».
5. Напишите и объясните формулу Литтла.
6. В чем состоит задача Эрланга? Нарисуйте граф СМО по задаче Эрланга.
7. Нарисуйте и объясните граф одноканальной СМО с неограниченной очередью.
8. Нарисуйте и объясните граф многоканальной СМО с неограниченной очередью.
9. Что используется в системах оперативной обработки в качестве критерия эффективности?
10. Как назначаются задачи на решение в алгоритме SPT?
11. В чем суть алгоритма RR (round-robin)?
12. В чем суть алгоритма FB (foreground-background)?
13. В чем заключается алгоритм Макнотона?
14. Как вычисляется оптимальное время θ решения задач в алгоритме Макнотона?
15. В чем суть алгоритма LPT?
16. Как строятся мультипроцессорные системы с общей памятью?
17. Как строятся мультипроцессорные системы с индивидуальной памятью?
18. Какие преимущества имеет структура МПС с общей памятью перед МПС с индивидуальной памятью?

Глава 4.

Компьютерные сети

Распределенные вычислительные системы (вычислительные сети) создаются с целью объединения информационных ресурсов нескольких компьютеров (под словом „несколько“ понимается от двух до нескольких миллионов компьютеров). Ресурсы компьютера - это, прежде всего память, в которой хранится информация, и производительность процессора (процессоров), определяющая скорость обработки данных. Поэтому в распределенных системах общая память и производительность системы как бы *распределены* между входящими в нее ЭВМ. Совместное использование общих ресурсов сети породило такие понятия и методы как распределенные базы и банки данных, распределенная обработка данных. В концептуальном плане вычислительные сети, как и отдельные компьютеры, являются средством осуществления информационных технологий и ее процессов.

Вычислительные сети принято подразделять на два класса: локальные вычислительные сети (ЛВС) и глобальные вычислительные сети (ГВС) [2].

Под локальной вычислительной сетью понимают распределенную вычислительную систему, в которой передача данных между компьютерами не требует специальных устройств, а достаточно электрического соединения компьютеров с помощью кабелей и разъемов. Так как электрический сигнал ослабевает (уменьшается его мощность) при передаче по кабелю и тем сильнее, чем протяженнее кабель, то, естественно, длина проводов, соединяющих компьютеры, ограничена. Поэтому ЛВС объединяют компьютеры, локализованные на весьма ограниченном пространстве. Обычно длина кабеля, по которому передаются данные между компьютерами, не должна превышать в лучшем случае 1 километра. Ука-

занные ограничения обусловили расположение компьютеров ЛВС в одном здании или в рядом стоящих зданиях. Обычно службы управления предприятий так и расположены, что и определило широкое использование в них для реализации процессов обмена локальных вычислительных сетей.

Глобальные сети объединяют ресурсы компьютеров, расположенных на значительном удалении, таком, что простым кабельным соединением не обойтись и приходится добавлять в междокомпьютерные соединения специальные устройства, позволяющие передать данные без искажения и по назначению. Эти устройства коммутируют (соединяют, переключают) между собой компьютеры сети и в зависимости от ее конфигурации могут быть как пассивными коммутаторами, соединяющими кабели, так и достаточно мощными ЭВМ, выполняющие логические функции выбора наименьших маршрутов передачи данных. В глобальных вычислительных сетях, помимо кабельных линий, применяют и другие среды передачи данных. Большие расстояния, через которые передаются данные в глобальных сетях, требуют особого внимания к процедуре передачи цифровой информации, с тем, чтобы посланные в сети данные дошли до компьютера-получателя в полном и не искаженном виде. В глобальных сетях компьютеры отдалены друг от друга на расстояние не менее одного километра и объединяют ресурсные возможности компьютеров в рамках района (округа) города или сельской местности, региона, страны и т.д.

Отдельные локальные и глобальные вычислительные сети могут объединяться, и тогда возникает сложная сеть, которую называют *распределенной сетью*.

Таким образом, в общем виде вычислительные сети представляют собой систему компьютеров, объединенных линиями связи и специальными устройствами, позволяющими передавать без искажения и переключать между компьютерами потоки данных. Линии связи вместе с устройствами передачи и приема данных называют *каналами связи*, а

устройства, производящие переключения потоков данных в сети можно определить одним общим названием - *узлы коммутации*.

4.1. Базовые топологии локальных компьютерных сетей

Термин *топология сетей* характеризует физическое расположение компьютеров, узлов коммутации и каналов связи в сети.

Проблема синтеза структуры (топологии) сети является одной из важнейших, но до конца не решенной, в связи с чем, при решении задач определения числа и взаимосвязи компонентов сети используются приближенные, эмпирические методы.

Все сети строятся на основе трех базовых топологий [1]:

звезда (star);

кольцо (ring);

шина (bus).

Топология „звезда“ характерна тем, что в ней все узлы соединены с одним центральным узлом (рисунок 4.1). Данные от передающего компьютера передаются *всем* компьютерам сети, однако, воспринимаются только тем компьютером, адрес которого указан в передаваемом сообщении.

Подобная структура экономична и удобна с точки зрения организации управления взаимодействия компьютеров (абонентов). Звездообразную сеть легко расширить, поскольку для добавления нового компьютера нужен только один новый канал связи. Существенным недостатком звездообразной топологии является низкая надежность: при отказе центрального узла выходит из строя вся сеть.

В топологии „кольцо“ компьютеры подключаются к повторителям (репитерам) сигналов, связанных в однонаправленное кольцо (рисунок 4.2). Кольцо из репитеров обычно реализуется на активном коммутаторе (habe).



Рисунок 4.1-Звёздообразная топология сети

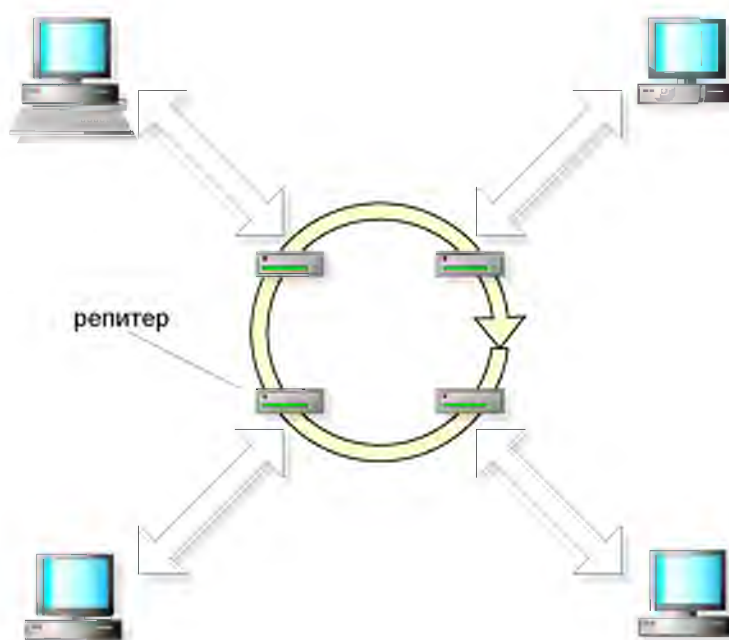


Рисунок 4.2-Кольцевая топология сети

По методу доступа к каналу связи (среде передачи данных) различают два основных типа кольцевых сетей: маркерное и тактированное кольца.

В *маркерных* кольцевых сетях по кольцу передается специальный управляющий **маркер** (метка), разрешающий передачу сообщений из компьютера, который им „владеет“. Если компьютер получил маркер и у него есть сообщение для передачи, то он „захватывает“ маркер и передает сообщение в кольцо. Данные проходят через все повторители, пока не окажутся на том повторителе, к которому подключен компьютер с адресом, указанным в данных. Получив подтверждение, передающий компьютер создает новый маркер и возвращает его в сеть. При отсутствии у компьютера сообщения для передачи он пропускает движущийся по кольцу **маркер**.

В *тактированном* кольце по сети непрерывно вращается замкнутая последовательность тактов - специально закодированных интервалов фиксированной длины. В каждом такте имеется бит-указатель занятости. Свободные такты могут заполняться передаваемыми сообщениями по мере необходимости, либо за каждым узлом могут закрепляться определенные **такты**.

Достоинством кольцевых сетей считаются равенство компьютеров по доступу к сети и высокая расширяемость. К недостаткам можно отнести выход из строя всей сети при выходе из строя одного повторителя и остановку работы сети при изменении ее конфигурации.

В топологии „*шина*“, широко применяемой в локальных сетях, все компьютеры подключены к единому каналу связи с помощью трансиверов (приёмо-передатчиков) (рисунок 4.3).



Рисунок 4.3-Шинная топология сети

Канал оканчивается с двух сторон пассивными терминаторами, поглощающими передаваемые сигналы. Без терминаторов возникают отражения от концов шины, что приводит к искажениям передаваемых по шине сигналов. Данные от передающего компьютера передаются всем компьютерам сети, однако, воспринимаются только тем компьютером, адрес которого указан в передаваемом сообщении. Причем, в каждый момент только один компьютер может вести передачу. Шина - пассивная топология. Это означает, что компьютеры только „слушают“ передаваемые по сети данные, но не перемещают их от отправителя к получателю. Поэтому, если один компьютер выйдет из строя, это не скажется на работе остальных, что является достоинством шинной топологии. В активных топологиях компьютеры регенерируют сигналы и передают их по сети (как повторители компьютеров в кольцевой топологии). Другие достоинства „шины“ - высокая расширяемость и экономичность в организации каналов связи. К недостаткам шинной организации сети относится уменьшение пропускной способности сети при значительных трафиках (трафик - это объем данных).

В настоящее время часто используются топологии, комбинирующие базовые: звезда-шина, звезда-кольцо.

Топология *звезда-шина* чаще всего выглядит как объединение с помощью магистральной „шины“ нескольких звездообразных сетей (рисунок 4.4).

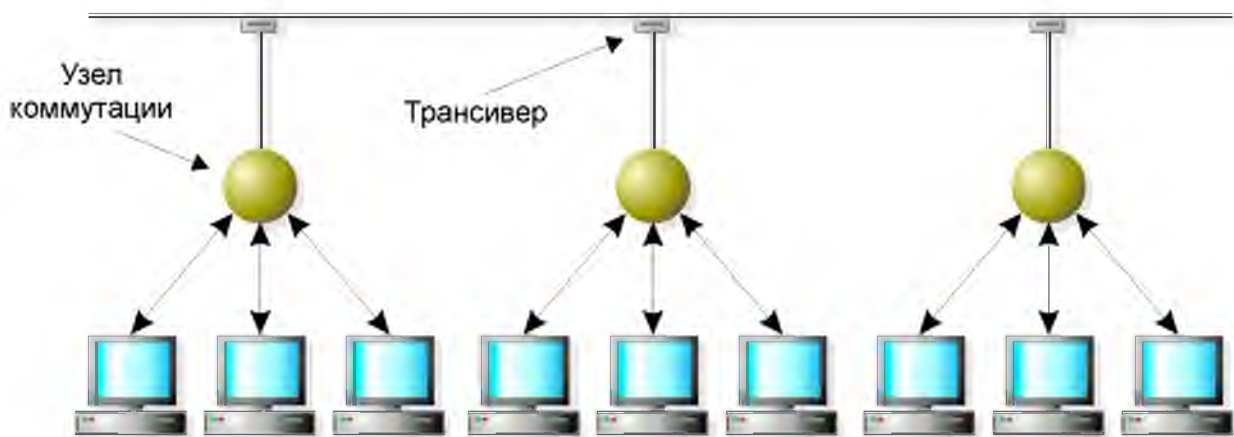


Рисунок 4.4-Топология „звезда-шина“

При топологии „звезда - кольцо“ несколько звездообразных сетей соединяются своими центральными узлами коммутации в кольцо (рисунок 4.5).

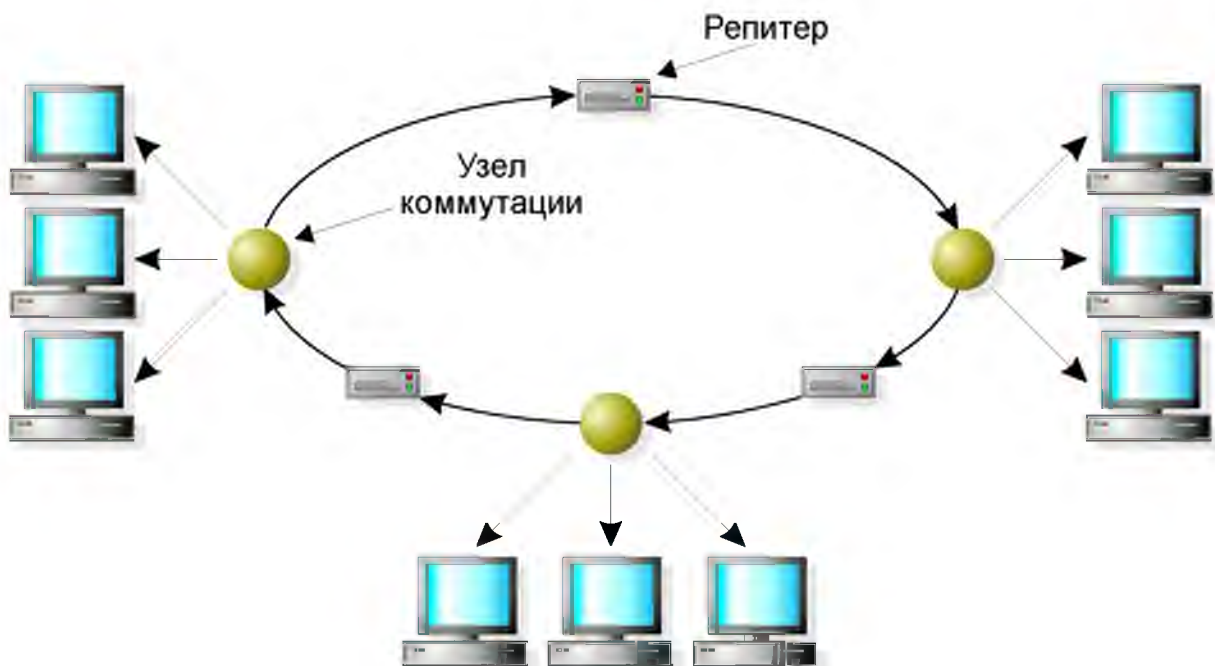


Рисунок 4.5-Топология „звезда-кольцо“

4.2.Топология глобальной вычислительной сети

Расширение локальных сетей, как базовых, так и комбинированных топологий, из-за удлинения линий связи приво-

дит к необходимости их расчленения и создания *распределенных сетей*, в которых компонентами служат не отдельные компьютеры, а отдельные локальные сети, иногда называемыми „сегментами“ [6]. Узлами коммутации таких сетей являются активные концентраторы (К) и мосты - устройства, коммутирующие линии связи (в том числе разного типа) и одновременно усиливающие проходящие через них сигналы. Мосты кроме этого еще и управляют потоками данных между сегментами сети. При соединении компьютеров или сетей (локальных или распределенных), удаленных на большие расстояния, используются каналы связи и устройства коммутации, называемые маршрутизаторами (М) и шлюзами (Ш). Маршрутизаторы взаимодействуют друг с другом и соединяются между собой каналами связи, образуя распределенный магистральный канал связи. Для согласования параметров данных (форматов, уровней сигналов, протоколов и т.п.), передаваемых по магистральному каналу связи, между маршрутизаторами и терминальными компонентами включаются устройства сопряжения (УС). При подключении к магистральному каналу вычислительных сетей или устройств (например, мейнфреймов), которых невозможно согласовать с помощью стандартных устройств сопряжения, используются стандартные средства, называемые шлюзами. Терминальными абонентами называют отдельные компьютеры, локальные или распределенные сети, подключенные через УС к магистральному каналу. Таким образом, возникает глобальная вычислительная сеть. Глобальные сети могут объединяться между собой путем соединения через маршрутизаторы магистральных каналов, что в конечном итоге приводит к созданию мировой (действительно глобальной) информационно - вычислительной сети.

Типовая топология описанной выше глобальной вычислительной сети приведена на рисунке 4.6.

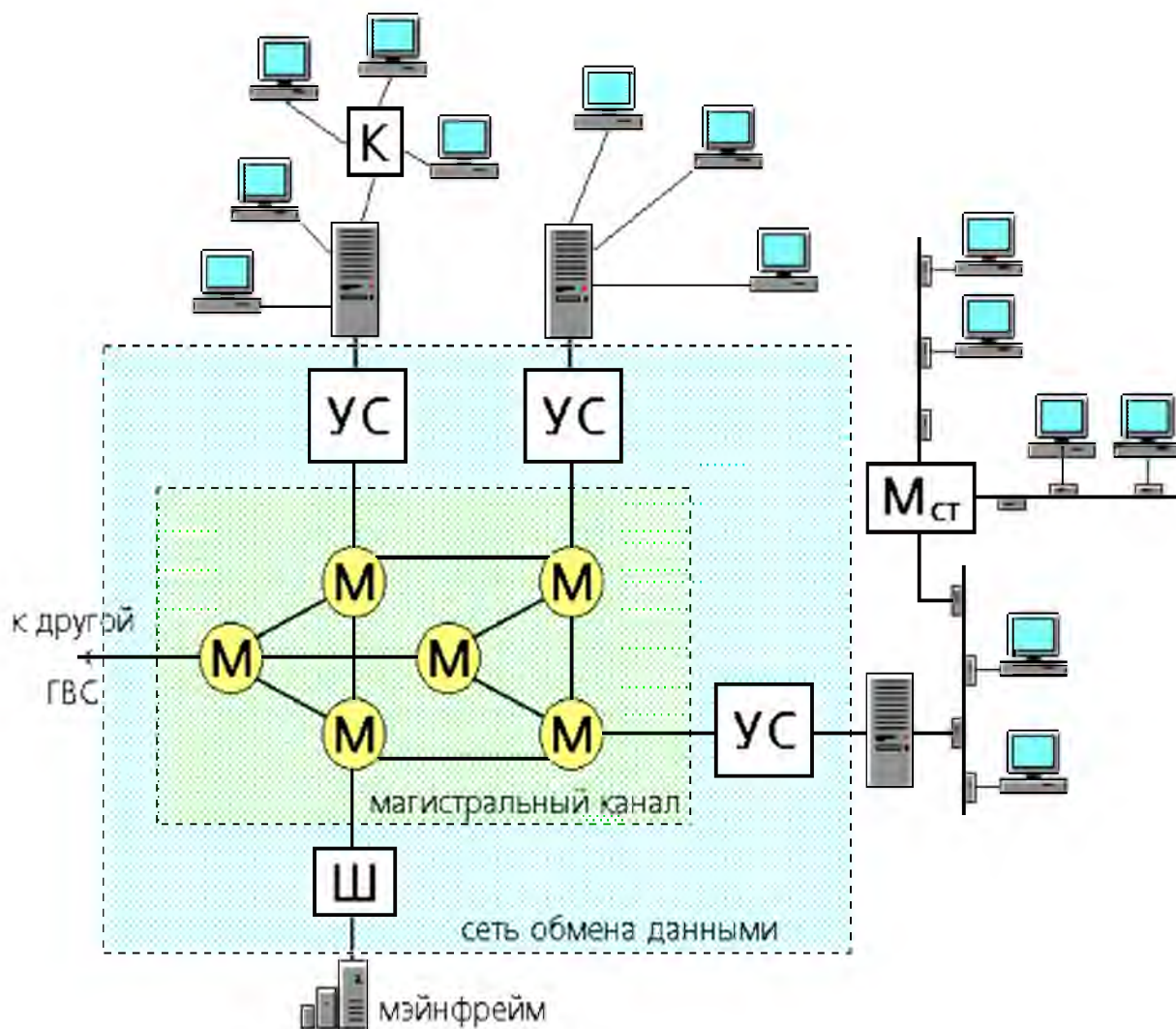


Рисунок 4.6 - Типовая топология глобальной информационно-вычислительной сети (ГВС)

4.3.Сетевые протоколы и уровни

Увеличение разнообразия различных архитектур связи побудило Международную организацию по стандартизации (МОС) направить значительные усилия на разработку стандарта архитектуры связи, который позволил бы системам открыто связываться между собой [11]. В 1983 году эти усилия увенчались успехом и была предложена базовая эталонная модель взаимодействия открытых систем (ВОС). Она состоит из семи уровней, показанных на рисунке 4.7. Три нижних уровня представляют сетевые услуги. Протоколы, реализую-

щие эти уровни должны быть предусмотрены в каждом узле сети. Четыре верхних уровня предоставляют услуги самим конечным пользователям и таким образом связаны с ними, а не с сетью.

Уровень канала передачи данных и находящийся под ним физический уровень обеспечивают канал безошибочной передачи между двумя узлами в сети. Функция физического уровня заключается в гарантии того, что символы, поступающие в физическую среду передачи на одном конце канала, достигнут другого конца. При использовании этой нижестоящей услуги по транспортировке символов задача протокола канала состоит в обеспечении надежной передачи блоков данных по каналу.

Функция сетевого уровня состоит в том, чтобы установить канал для передачи данных по сети от узла передачи до узла назначения. Этот уровень предусматривает также управление потоком или перегрузками в целях предотвращения переполнения сетевых ресурсов, которое может привести к прекращению работы.

Транспортный уровень обеспечивает надежный, последовательный обмен данными между двумя конечными пользователями. Для этой цели на транспортном уровне используется услуга сетевого уровня. Он также управляет потоком, чтобы гарантировать правильный прием блоков данных.

Существование сеанса между двумя пользователями означает необходимость установления и прекращения его, что делается на уровне сеанса. Этот уровень при необходимости управляет переговорами, чтобы гарантировать правильный обмен данными.

Уровень представления управляет и преобразует синтаксис блоков данных, которыми обмениваются конечные пользователи, а протоколы прикладного уровня придают соответствующий смысл обмениваемой информации.

В сети с коммутацией пакетов блоками данных, передаваемых по сетевому маршруту от одного конца к другому,

являются пакеты. Блоки или кадры данных, передаваемые по каналу связи через сеть, состоят из пакетов плюс управляющей информации в виде заголовков и окончаний, добавляемых к пакету непосредственно перед его отправлением из узла. В каждом принимающем узле управляющая информация отделяется от остальной части пакета, а затем вновь добавляется, когда этот узел в свою очередь передает пакет по каналу в следующий соседний узел. Этот принцип добавления управляющей информации к данным в архитектуре ВОС расширен и включает возможность добавления управляющей информации на каждом уровне архитектуры. Как это происходит, показано на рисунке 4.7.

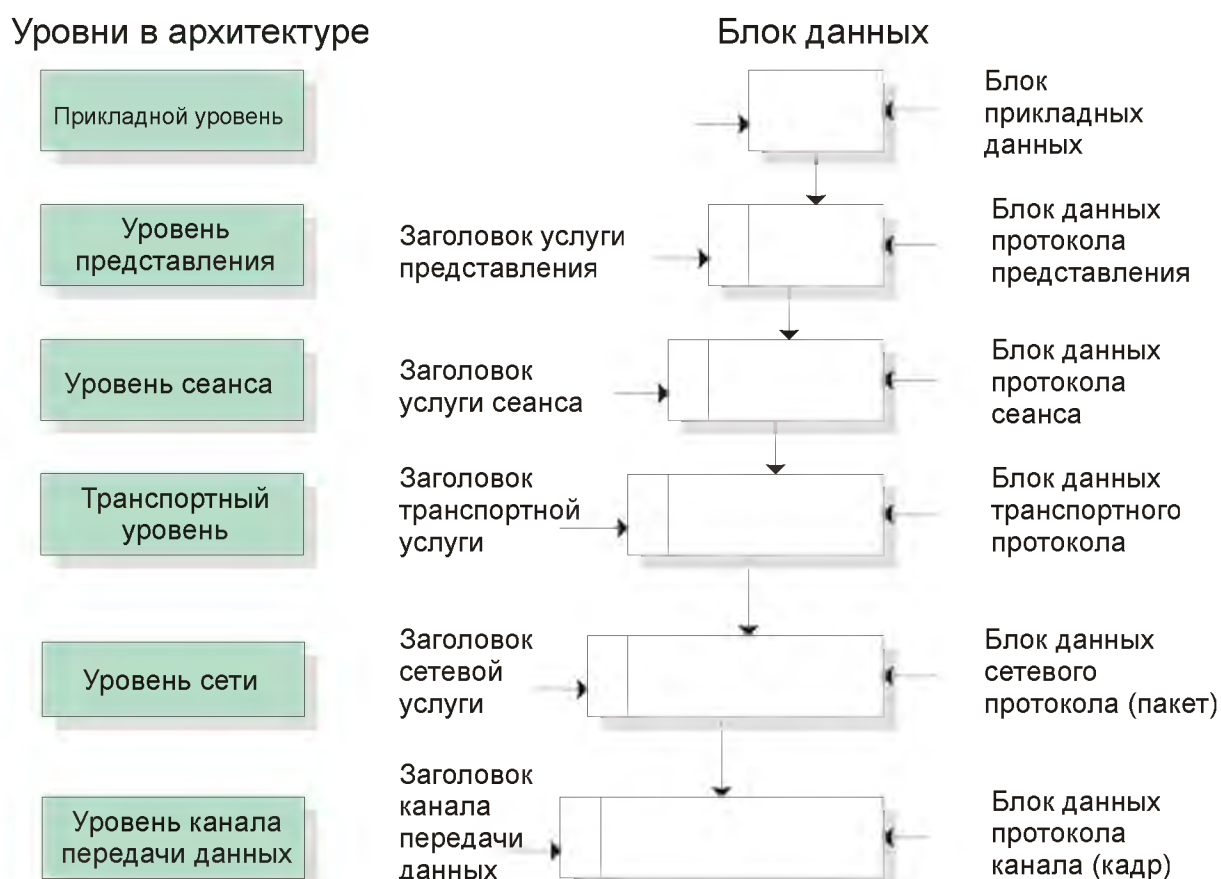


Рисунок 4.7 - Блоки данных, применяемые в структуре сети ВОС

На каждом уровне блок данных принимается от вышестоящего уровня, к данным добавляется управляющая ин-

формация, и блок передается нижестоящему уровню. Данный уровень не просматривает блок данных, который он получает от вышестоящего уровня. Следовательно, уровни самостоятельны и изолированы друг от друга. На принимающем компьютере производятся обратные операции.

На рисунке 4.8 показан пример конкретной многоуровневой архитектуры связи. Между источником и получателем информации включен промежуточный узел. Пакет, поступающий по физической среде, связывающей исходящий узел с промежуточным, направляется на сетевой уровень этого узла, на котором определяется следующая часть пути в составе маршрута через сеть.

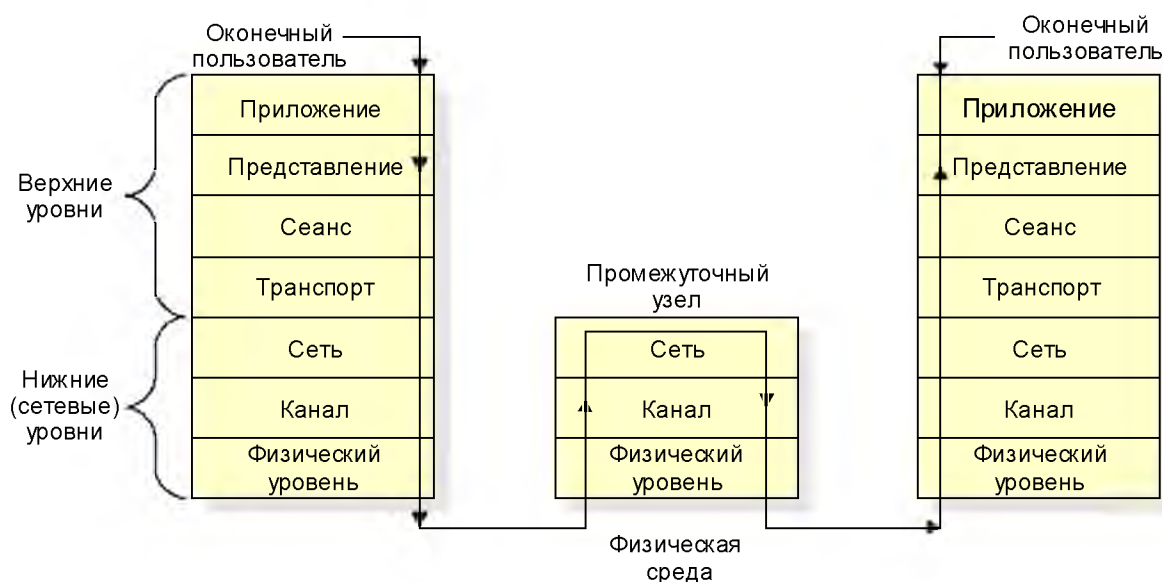


Рисунок 4.8 - Семиуровневая архитектура ВОС

4.4. Физический и канальный уровни

Современные системы связи способны передавать сообщения в любой форме: телеграфные, телефонные, телевизионные, массивы данных, печатные материалы, фотографии и др. [10].

С точки зрения эталонной модели открытых систем, процедуры передачи данных действуют на физическом и канальном уровнях.

В соответствии со спецификой передаваемых сообщений организуется канал, представляющий собой комплекс технических средств, обеспечивающих передачу сигналов от источника к потребителю. К основным параметрам, характеризующим канал связи, относятся ширина полосы пропускания, допустимый динамический диапазон изменений амплитуды сигнала, а также уровень помех.

Передача больших информационных потоков на значительные расстояния осуществляется с помощью кабельных, радиорелейных и спутниковых линий связи. В ближайшие годы можно ожидать широкого применения оптической связи по оптическим кабелям.

Рассмотрим основные принципы передачи информации с помощью электрических сигналов. Эти принципы, многие из которых носят фундаментальный характер, прочно вошли в практику не только систем электросвязи, но и вычислительной техники и, конечно, информационных технологий.

4.4.1. Модуляция и демодуляция

Сообщение для передачи с помощью средств электросвязи (так у нас принято называть то, что на Западе называют telecommunication) должно быть предварительно преобразовано в сигнал, под которым понимается изменяющаяся физическая величина, адекватная сообщению. Процесс преобразования сообщения в сигнал называется *кодированием*.

По физическим законам излучение электромагнитных волн эффективно, если размеры излучателя соизмеримы с длиной излучаемой волны, поэтому передача сигналов по радиоканалам, кабелям, микроволновым линиям производится на высоких частотах (т. е. на весьма коротких волнах). Сигнал передается на „несущей« частоте. Процесс изменения параметров несущей в соответствии с сигналом, передаваемым

на этой несущей, называют *модуляцией*. Модуляция - основной процесс или функция передатчика.

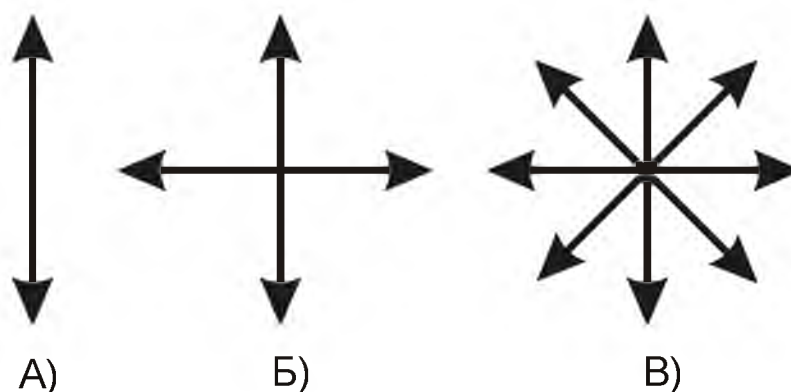


Рисунок 4.9-Фазовые диаграммы 2-кратной (а), 4-кратной (б), 8-кратной (в) фазовой манипуляции

Гармоническая (синусоидальная) несущая имеет три информационных параметра, которые можно модулировать: амплитуду, частоту и фазу

$$u = U \cos(\omega_0 t + \varphi_0) ,$$

где ω_0 - частота несущей;

φ_0 - начальная фаза;

U - амплитуда гармонического колебания.

Соответственно при передаче сигналов используют амплитудную, частотную и фазовую модуляцию, которая в случае дискретных сигналов называется *манипуляцией*.

Наиболее помехоустойчивой, т. е. невосприимчивой к помехам, является *фазовая* модуляция или манипуляция (ФМн), что объясняется «амплитудным» характером воздействующих помех. Такой параметр, как фаза несущей, менее других параметров подвергается губительному воздействию помех. Фазоманипулированный сигнал представляет собой отрезок гармонического колебания с изменяющейся на 180° фазой. В векторной форме это можно изобразить так, как показано на рисунке 4.9 а.

При векторном изображении сигналов помехи также можно рассматривать как случайные вектора со случайной амплитудой и фазой. Такое геометрическое представление сигналов и помех позволяет легко понять, почему ФМн сигнал с двумя значениями фазы оказывается наиболее помехоустойчивым. Дело в том, что приёмник при приёме сигналов решает задачу: в какой из областей решения находится сигнал (верхней или нижней, рисунок 4.9а). В том случае, когда область принятия решения состоит только из двух частей, вероятность ошибки наименьшая. Однако, если 2ФМн сигнал переносит один сигнал, то 4ФМн переносит сразу два сигнала (рисунок 4.9б), 8ФМн — четыре сигнала (рисунок 4.9в).

Прохождение сигналов по каналу связи всегда сопровождается искажениями и воздействием помех. Поэтому основной функцией приемника является распознавание в принимаемых колебаниях переданного сигнала. Такую операцию приёмник производит в процессе *демодуляции*, т. е. в процессе выделения передаваемого сигнала, после чего он преобразовывается в сообщение (рисунок 4.10).

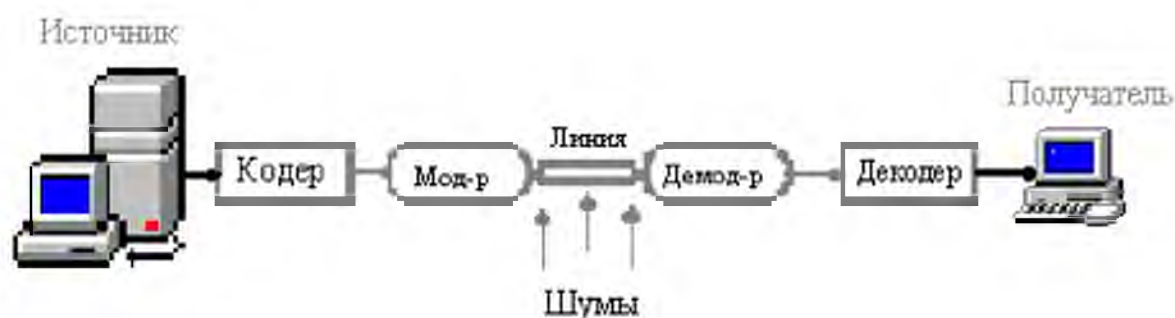


Рисунок 4.10-Передача информации по каналу связи

Каналом передачи информации (каналом связи) называют совокупность технических средств, обеспечивающую передачу электрических сигналов от одного пункта к другому. Непременной составной частью любого канала является линия связи - проводная, кабельная, радио, микроволновая, оптическая, спутниковая.

В современных цифровых системах связи основные функции передатчика и приёмника выполняет устройство, называемое модемом. Он представляет собой совокупность передатчика и приемника в одном корпусе для осуществления проводной дуплексной связи. Если терминал находится на значительном расстоянии от компьютера, например в соседнем здании или другом городе, или связь пользователя с компьютером происходит через обычную телефонную сеть, необходимы приёмопередатчики на оконечных пунктах линии и их функции выполняет модем.

Выпускаемые в настоящее время модемы различны по конструкции, но, как правило, состоят из интерфейсной части для соединения с компьютером, кодера и декодера, модулятора и демодулятора. Часто в состав модема входят шифрующее и дешифрующее устройства, обеспечивающие секретность передаваемой информации. Имеются также способы, обеспечивающие скрытность передачи. В зависимости от типа модема он производит амплитудную, частотную или фазовую модуляцию. В целях уплотнения полосы канала чаще всего используют многократную фазовую манипуляцию (см. рисунок 4.9). Типовые скорости передачи у модемов 2400, 4800, 9600, 14400, 19200, 28800, 33600, 57600 бит/с.

4.4.2. Емкость канала связи

Скорость передачи информации, а ее предельно допустимое значение для данного канала называют *емкостью* канала, относится к фундаментальным понятиям теории связи. Она служит одной из главных характеристик канала передачи информации. Оценка скорости передачи информации и предельных возможностей канала связи представляет большой практический и теоретический интерес.

Рассматривая процесс передачи информации в общих чертах, можно предположить, что основными факторами, ограничивающими скорость передачи информации, считаются полоса пропускания F и уровень помех.

Существует фундаментальная теорема о «выборках», которая доказывает, что сигнал, не содержащий в своем спектре частот выше F , может представляться $2F$ независимыми значениями в секунду, и совокупность значений, отстоящих друг от друга на T секунд, определяет непрерывный сигнал полностью. Заметим, что «выборкой» является отсчет амплитуды сигнала в определенный момент (на рисунке 4.11 а можно увидеть эти выборки).

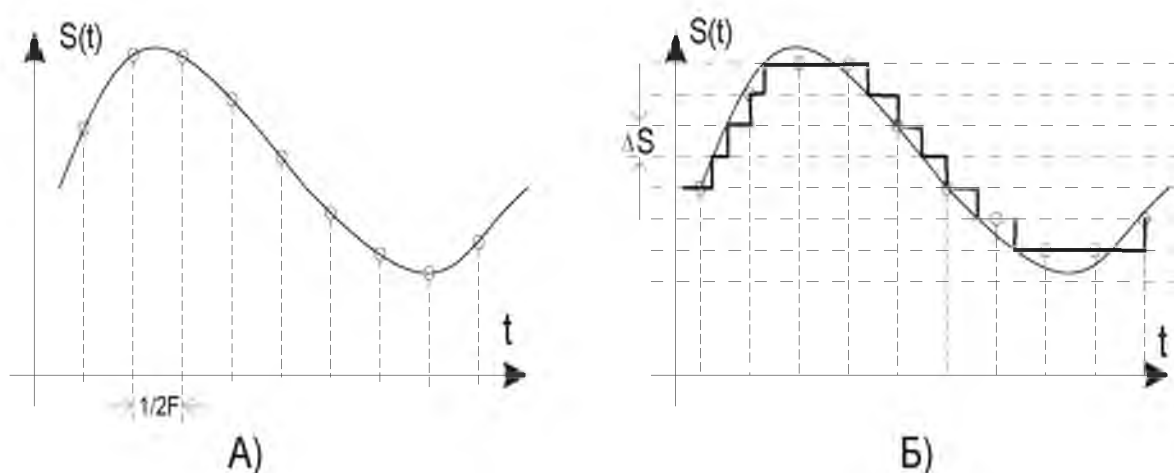


Рисунок 4.11-Представление непрерывного сигнала в виде дискретных отсчетов (выборок);

А — отсчеты сигнала, взятые через интервал $1/(2F)$; Б — отсчеты сигнала, квантованные по амплитуде.

Термин «выборки» взят от английского *samples*, теорему о выборках называют также теоремой отсчетов. Она позволяет на интервале T заменить непрерывный сигнал с ограниченным спектром последовательностью его дискретных значений, причем их нужно не бесконечное число, а вполне определенное, равное $2FT$. Уровень шумов (помех) не позволяет точно определить амплитуду сигнала и в этом смысле вносит некоторую неопределенность в значение отсчетов сигнала.

Максимально возможная скорость передачи информации по каналу связи при фиксированных ограничениях назы-

вается емкостью канала, обозначается через C и имеет размерность *бит/с*.

Рассмотрим соотношение для емкости канала связи, являющееся фундаментальным соотношением в теории связи. Оно позволяет понять некоторые принципиальные зависимости при передаче информации вообще. Напомним, что количество информации I , снимающее неопределенность о состоянии объекта с L равновероятными состояниями,

$$I = \log L.$$

Основание логарифма здесь не имеет значения. Если основание равно двум, то единицей измерения количества информации оказывается *бит*.

Определим количество различных сообщений, которое можно составить из n элементов, принимающих любые из m различных фиксированных состояний. Из ансамбля n элементов, каждый из которых может находиться в одном из m фиксированных состояний, можно составить m^n различных комбинаций, т. е. $L = m^n$. Тогда

$$I = \log m^n = n \log m.$$

При полосе F наибольшее число отсчетов сигнала равно $2F$ в единицу времени или $2FT$ за время T , т.е. $n = 2FT$.

Если бы шума не существовало, то число дискретных уровней сигнала было бы бесконечным. При наличии шума определяется степень различимости отдельных уровней амплитуды сигнала. Так как мощность является усредненной характеристикой амплитуды, число различимых уровней сигнала по мощности равно $(P_c + P_{ш})/P_{ш}$, а по амплитуде соответственно $m = \sqrt{(P_c + P_{ш})/P_{ш}}$.

Тогда ёмкость канала

$$C = \frac{I}{T} = \frac{1}{T} 2FT \log_2 \sqrt{\frac{P_c + P_{ш}}{P_{ш}}} = F \log_2 \left(1 + \frac{P_c}{P_{ш}} \right).$$

Итак, емкость канала ограничивается двумя величинами: шириной полосы канала и шумом. Приведенное соотношение

известно как формула Хартли-Шеннона и считается основной в теории информации.

Полоса частот и мощность сигнала входят в формулу таким образом, что для $C = \text{const}$ при сужении полосы необходимо увеличивать мощность сигнала, и наоборот.

Ёмкость канала называют максимальной величиной скорости. Чтобы достигнуть такой скорости передачи, информация должна быть закодирована наиболее эффективным образом. Утверждение, что такое кодирование возможно, является важнейшим результатом созданной Шенноном теории информации. Шеннон доказал принципиальную возможность такого эффективного кодирования, не определив, однако, конкретных путей его реализации. (Отметим, что на практике инженеры часто говорят о ёмкости канала, подразумевая под этим реальную, а не потенциальную скорость передачи).

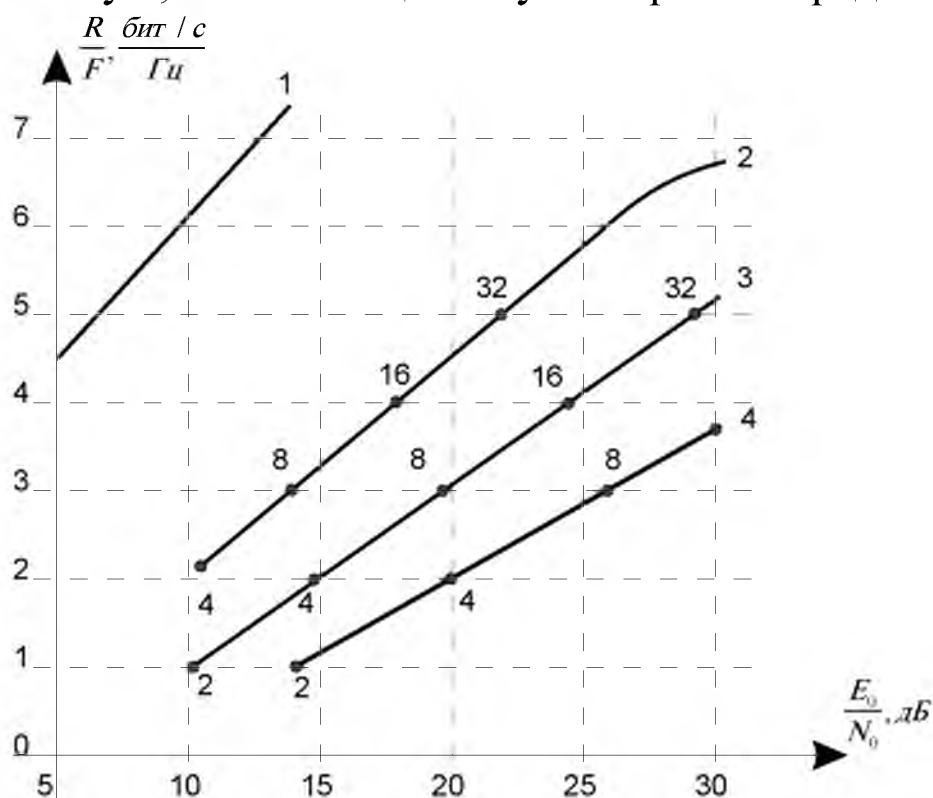


Рисунок 4.12-Эффективность цифровых систем связи: 1- граница Шеннона; 2- М-ичная ФМ; 3- М-ичная АМ; 4- М-ичная ЧМ.

Эффективность систем связи характеризуется параметром, равным скорости передачи информации R на единицу ширины полосы F , т. е. R/F . Для иллюстрации существующих возможностей по созданию эффективных систем связи на рисунке 4.12 приведены графики зависимости эффективности передачи информации при различных видах M -ичной дискретной амплитудной, частотной и фазовой модуляции (кроме бинарной модуляции используется также модуляция с 4-, 8-, 16- и даже с 32-мя положениями модулируемого параметра) от отношения энергии одного бита к спектральной плотности мощности шума (E_o/N_o). Для сравнения показана также граница Шеннона.

Сравнение кривых показывает, в частности, что при неизменном отношении сигнал-шум наиболее популярный вид модуляции 4ФМн в три раза хуже потенциально достижимого. Из сравнения кривых можно сделать более общие выводы: наиболее эффективной оказывается передача с фазовой дискретной модуляцией; современные методы кодирования и модуляции еще весьма далеки от совершенства.

4.4.3. Кодирование информации

Кодированием называется сопоставление алфавитов, а правило, по которому оно производится, - *кодом*. Иными словами, кодирование можно определить как представление сообщений в форме, удобной для передачи по данному каналу. Электрический ток в телефонных проводах - это кодированная речь, а звуковые волны речи - это кодированные колебания голосовых связок.

В рассматриваемом нами конкретном случае кодирование есть представление по определенным правилам дискретных сообщений в некоторые комбинации, составленные из определенного числа элементов - символов. Эти элементы называются элементами кода, а число различных элементов, из которых слагаются комбинации, - основанием кода. Элементы кода образуют кодовые комбинации. Например, если

составляем комбинации из различных сочетаний 0 и 1, то это код с основанием два, или двоичный код. Если все комбинации имеют одинаковое число знаков, код называется равномерным. Широко известный код Морзе - неравномерный код. Правило кодирования обычно выражается кодовой таблицей, в которой каждому символу сообщения ставится в соответствие определенная кодовая комбинация.

Кодовое представление дискретных значений сигнала осуществляется с помощью цифр, но не обязательно десятичных. Напомним, что в десятичной системе, называя число, мы указываем, сколько единиц от нуля до девяти имеется в разряде единиц, в разряде десятков, сотен, тысяч и т. д. То же происходит в любой другой системе счисления с другим основанием. В десятичной системе мы пользуемся десятью цифрами: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. В двоичной системе счисления в нашем распоряжении только две цифры: 0 и 1

Если пронумеровать все буквы алфавита и необходимые специальные символы и выразить каждую цифру в двоичной системе счисления, получится натуральный двоичный код данного алфавита. Очевидно, число разрядов в двоичной системе больше, чем в десятичной, так как основание системы счисления меньше.

Число кодовых комбинаций определяется числом дискретных значений сигнала. Например, если в языке 32 буквы (или букв и знаков), то для передачи сообщений на этом языке необходимо иметь 32 различные кодовые комбинации. В десятичной системе это означало бы передачу 32 цифр от 0 до 31. В двоичной системе необходимо составить отличающиеся друг от друга 32 кодовые комбинации, и так как $32 = 2^5$, эти комбинации должны быть из 5 элементов, например 01010, 11111, 11001 и т. д. Число возможных кодовых комбинаций для представления 32 букв колоссально: $32!$ Один из этих вариантов есть натуральный пятизначный двоичный код, используемый для передачи букв латинского и русского алфавитов. При цифровом кодировании речевых сигналов ис-

ходят из практического наблюдения: искажения сигнала невелики, если его изменения представлять 128-ю амплитудными значениями, т. е. для его передачи необходимо 128 кодовых комбинаций. Для двоичного кода из соотношения $2^n = 128$ определяем, что длина кодовой комбинации $n=7$. Таким образом, для передачи речевых сигналов нужен код с семи-элементными кодовыми комбинациями.

Обычно речевой сигнал по спектру ограничен частотой 4000 Гц. В этом случае речь в цифровой форме необходимо передавать со скоростью (вспомним теорему о выборках!) $4000 \cdot 2 \cdot 7 = 56$ кбит/с. Заметим, что обычно в комбинацию добавляют один служебный символ и тогда комбинация становится 8-элементной, а необходимая скорость передачи увеличивается до 64 кбит/с.

Остановимся также на принципах помехоустойчивого кодирования, имеющего чрезвычайно важную роль в развитии средств передачи информации. Отметим, что теория помехоустойчивого кодирования является достаточно сложной, и наши рассуждения носят весьма упрощенный характер.

Основным условием обнаружения и исправления ошибок в принимаемых кодовых комбинациях является избыточность. Поясним это на примере.

Условимся, что необходимо передавать только четыре сообщения: А, Б, В и Г. Можно составить четыре двухэлементные комбинации для передачи этих сообщений:

А	Б	В	Г
00	01	10	11

Пусть помехи воздействуют на комбинацию таким образом, что изменяют только один из ее элементов. Если помехе подверглась комбинация 00 и она вследствие этого превратилась в комбинацию 01, то мы не обнаружим ошибку, а будем просто считать, что вместо А передатчик послал Б. И так будет со всеми четырьмя комбинациями.

Теперь введем избыточность. Используем для передачи А, Б, В, Г трехэлементные кодовые комбинации, которых, кстати, может быть всего восемь. Выберем из восьми возможных комбинаций 000, 001, 010, 100, 110, 011, 101, 111 (других комбинаций быть не может) только четыре, но так, чтобы они максимально отличались друг от друга: 000, 011, 101, 110.

Пусть в результате действия помехи изменится один из элементов в любой из выбранных комбинаций. Такая комбинация не будет идентичной ни одной из наших комбинаций, и сразу станет ясно, что она ошибочна. Таким образом, для передачи сообщений А, Б, В, Г код 00, 01, 10, 11 годится, но он не помехоустойчив, код же 000, 011, 101, 110 является помехоустойчивым. При этом следует оговориться, что он помехоустойчив только к таким помехам, которые могут привести лишь к однократной ошибке в комбинации. При двукратной ошибке код не помехоустойчив. Для защиты от таких помех ансамбля А, Б, В и Г пришлось бы допустить еще большую избыточность, используя четырехэлементные кодовые комбинации, т. е. выбрав 4 комбинации из 16 возможных.

Таким образом, обнаружить ошибку невозможно, если любой принятый символ служит сообщением. Ошибки можно обнаружить только в том случае, если на возможные сообщения наложены некоторые ограничения.

Одним из основных достоинств передачи информации в цифровой форме является возможность использования кодированных сигналов и оптимального в заданных условиях способа их приема. Важно, что при цифровой передаче все типы сигналов, такие как речь, музыка, телевидение, данные, могут объединяться в один общий поток информации, передача которого формализована. Кроме того, уплотнение при одновременном использовании компьютера позволяет эффективнее использовать спектр и время, защитить канал от несанкционированного доступа, объединить процесс передачи цифро-

вой информации и цифровую коммутацию каналов и сообщений.

4.4.4. Уплотнение информационных потоков

На практике часто требуется осуществить одновременную передачу информации от многих источников по одному каналу ко многим получателям, т. е. осуществить многоканальную передачу. Следует сказать, что современные системы передачи информации практически всегда многоканальные [6].

Способ объединения отдельных сообщений в один групповой сигнал с последующим разделением сообщений на индивидуальные называется *уплотнением* или *мультиплексированием*. К классическим методам уплотнения относятся частотное, временное и кодовое.

Современная техника связи позволяет организовывать широкополосные каналы, поэтому целесообразно использовать методы, позволяющие передавать наибольшее число телеграфных, телефонных, телевизионных и других сообщений на одной несущей или в отведенном интервале частот.

Сущность методов мультиплексирования состоит в том, что сообщения от нескольких источников определенным образом комбинируются в групповой сигнал и принимаются с помощью одного приемопередатчика. Поскольку современная система связи обычно является многоканальной, необходимой частью любой системы передачи информации служит мультиплексор (рисунок 4.13).

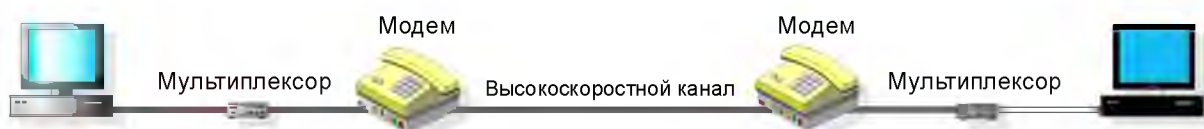


Рисунок 4.13-Цифровая система связи

Наиболее известен способ частотного мультиплексирования, когда в полосе пропускания канала размещается мно-

жество каналов, разделенных с помощью фильтрации по частоте (рисунок 4.14а). Каждый частотный канал представлен своим спектром. Его временная структура может быть различной — это может быть последовательность импульсов или телефонное сообщение. Соответствующая настройка разделительных фильтров приемника позволяет разделить принимаемый групповой сигнал на отдельные.

При временном мультиплексировании в условном временном интервале размещают последовательно отрезки сообщений, например кодовые последовательности каждого частного канала (рисунок 4.14б). Если при частотном мультиплексировании сообщения от разных абонентов передаются одновременно по общему каналу, при временном мультиплексировании передача осуществляется строго по очереди, т. е. полоса пропускания канала предоставляется полностью на определенный интервал времени каждому абоненту. На практике обычно группы каналов объединяются в супергруппы и при каждом иерархическом объединении может применяться разный способ модуляции несущей.

Аналоговый сигнал, например в телефонном канале, преобразуется в цифровой с помощью импульсно-кодовой модуляции (ИКМ) и передается в каналах с временным мультиплексированием. Передача организуется так: выборки каждого непрерывного сигнала сдвигаются на интервал, достаточный для передачи соответствующей кодовой комбинации.

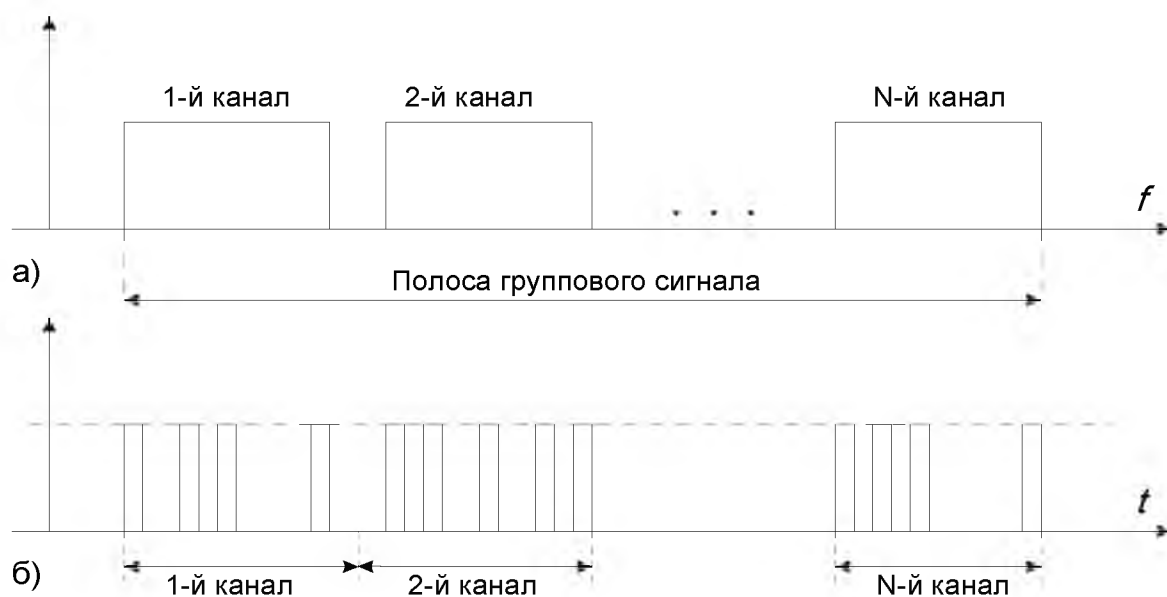


Рисунок 4.14-Частотное (а) и временное (б) уплотнение

При передаче n непрерывных сигналов в стандартном интервале времени размещают n кодовых комбинаций, по одной на каждую выборку каждого сигнала. При этом полоса частот группового сигнала увеличивается примерно в n раз. Так, 24-канальная система для передачи речи работает со скоростью 1544 кбит/с (скорость одного канала 64 кбит/с).

Международный консультативный комитет по телефонии и телеграфии разработал стандарты образования многоканальных сообщений при временном мультиплексировании. Прежде всего предложен восьмиразрядный равномерный код для указания значений уровней квантования сигнала и закон квантования, названный „ $A=87,6$ “. На рисунке 4.11 использован линейный закон квантования, когда интервалы квантования одинаковы. Закон квантования $A=87,6$ является нелинейным, он лучше учитывает природу восприятия человеком речевых сигналов. Частота дискретизации телефонного сообщения принята равной 8 кГц. При этом скорость передачи одного телефонного сообщения оказывается равной 64 кбит/с.

Так как принципиальной основой многоканальной цифровой системы передачи информации является временная

шкала, определяющая расстановку информационных и служебных сигналов, соединение цифровых систем различной емкости в единую сеть возможно лишь при условии кратного соответствия временных шкал различных систем и стандартизации групповых сигналов и способов синхронизации. С этой целью разрабатывается иерархия (соподчиненность) цифровых систем.

Под уровнем цифровой системы понимается число каналов или скорость передачи. Иерархия предусматривает возможность образования цифровыми системами низшего порядка системы более высокого порядка. На одном уровне объединяется фиксированное число цифровых сигналов системы более низкого уровня для образования суммарного цифрового сигнала более высокого уровня.

Например, первый уровень соответствует многоканальной передаче 30 телефонных сообщений в цифровой форме. Для этого необходима суммарная скорость передачи 2048 кбит/с. Второй уровень образован из четырех систем первого уровня с учетом необходимой служебной информации. Он имеет суммарную скорость 8448 кбит/с. Система второго уровня способна передавать 120 телефонных каналов или один видеотелефонный. Третий и четвертый уровни по рекомендации МККТТ соответствуют скоростям 34,368 и 139,264 Мбит/с.

Некоторые фирмы или страны работают по своим стандартам. В таблице 2 приведены сведения об иерархии уровней цифровых систем (скоростей передачи).

Иерархия скоростей цифровых систем является важной эксплуатационной характеристикой. Она предусматривает адаптивность систем к любым цифровым каналам (от обычных телефонных до волоконно-оптических) и всем информационным сигналам (от речевых до сигналов цветного телевидения).

Таблица 4.2

Иерархия цифровых систем

Уровень иерархии	Скорость передачи, (Мбит/с)/число каналов		
	МККТТ	США, Канада	Япония
1	2,048/30	1,544/24	1,544/24
2	8,448/120	6,312/96	6,312/96
3	34,368/480	44,736/672	32,064/480
4	139,264/1920	274,176/4032	97.728/1440
5	565,148/7680		397,2/5760

Существует много причин, вызывающих необходимость стандартизации скоростей передачи цифровой информации. К ним относятся требования потребителей каналов к универсальности передающей аппаратуры по отношению к различным источникам информации, необходимости планирования развития сетей передачи данных с учетом старой и новой аппаратуры при гармоническом сочетании систем, надежности и гибкости сети передачи данных. Благодаря соблюдению стандартов иерархии можно осуществлять передачу цифровой информации по комбинированным системам с использованием кабельных, радио, спутниковых, волоконно-оптических и других каналов.

4.5. Протоколы канального уровня

В последнее время международным стандартом становится протокол ВУК (высокоуровневое управление каналом передачи данных, **HDLC**) [11]. Стандартный формат кадра ВУК изображен на рисунке 4.15.



Рисунок 4.15-Формат кадра ВУК

В начале и в конце кадра для установления и поддержания синхронизации применяется восьмиразрядная последовательность 01111110, называемая флагом, или меткой. Поскольку в начале и в конце кадра применяются флаги, в установке структуры информационного поля нет необходимости. Пакет, поступающий от вышестоящего сетевого уровня, может занимать любое желаемое число разрядов. Проверочное поле занимает 16 разрядов, поля адреса, контроля и управления – по 8 разрядов.

Протокол канального уровня реализует следующие функции:

1. Реализация соединения между концами каналов.
2. Организация передачи данных по каналу.
3. Разъединение каналов.

Следуя концепции многоуровневой архитектуры, МОС стандартизировала применение на каждом уровне архитектуры четырех основных примитивов услуг, чтобы предусмотреть взаимодействие между пользователями услуг на одном уровне и поставщиками услуг на нижестоящем уровне. Эти примитивы являются основными элементами определения обмена между пользователями услуг. К ним относятся:

- запрос (request);
- признак (indication);
- ответ (response);
- подтверждение (confirm).

При работе примитивов два соседних уровня взаимодействуют между собой. Нижние являются поставщиками услуг, верхние – потребителями.

4.5.1.Схема организации фаз коммуникаций

Как видно из рисунка 4.16, запрос подается пользователем услуги данного (N+1)-го уровня системы А, чтобы обратиться к процедуре протокола поставщика услуги нижестоя-

щего N-го уровня [11]. Это приводит к посылке сообщения N-го уровня БДП-N в систему В (БДП – блок данных протокола). Получение блока БДП-N в системе В вызывает затем появление примитива признак, выпускаемого поставщиком услуги на этом уровне. Примитив ответ выпускается поставщиком услуги на уровне (N+1) в системе В в ответ на признак, он является директивой протоколу уровня N завершить процедуру обращения примитива признак. Протокол на уровне N генерирует БДП, который передается по сети и повторяется на уровне N системы А, что вызывает, в свою очередь, посылку примитива подтверждение, который выпускается поставщиком услуги в системе А. В результате процедура, начатая запросом в точке доступа к услуге между уровнями N и (N+1) в системе А в этой же точке завершается.

В качестве конкретного применения этих примитивов рассмотрим уровень канала передачи данных. Его задачей является предоставление обслуживания сетевому уровню. Услуги делятся на три фазы: установление соединения, передачу и разъединение.

На рисунке 4.16 предполагается, что рассматриваемый канал передачи данных в данный момент не используется в сети для передачи.

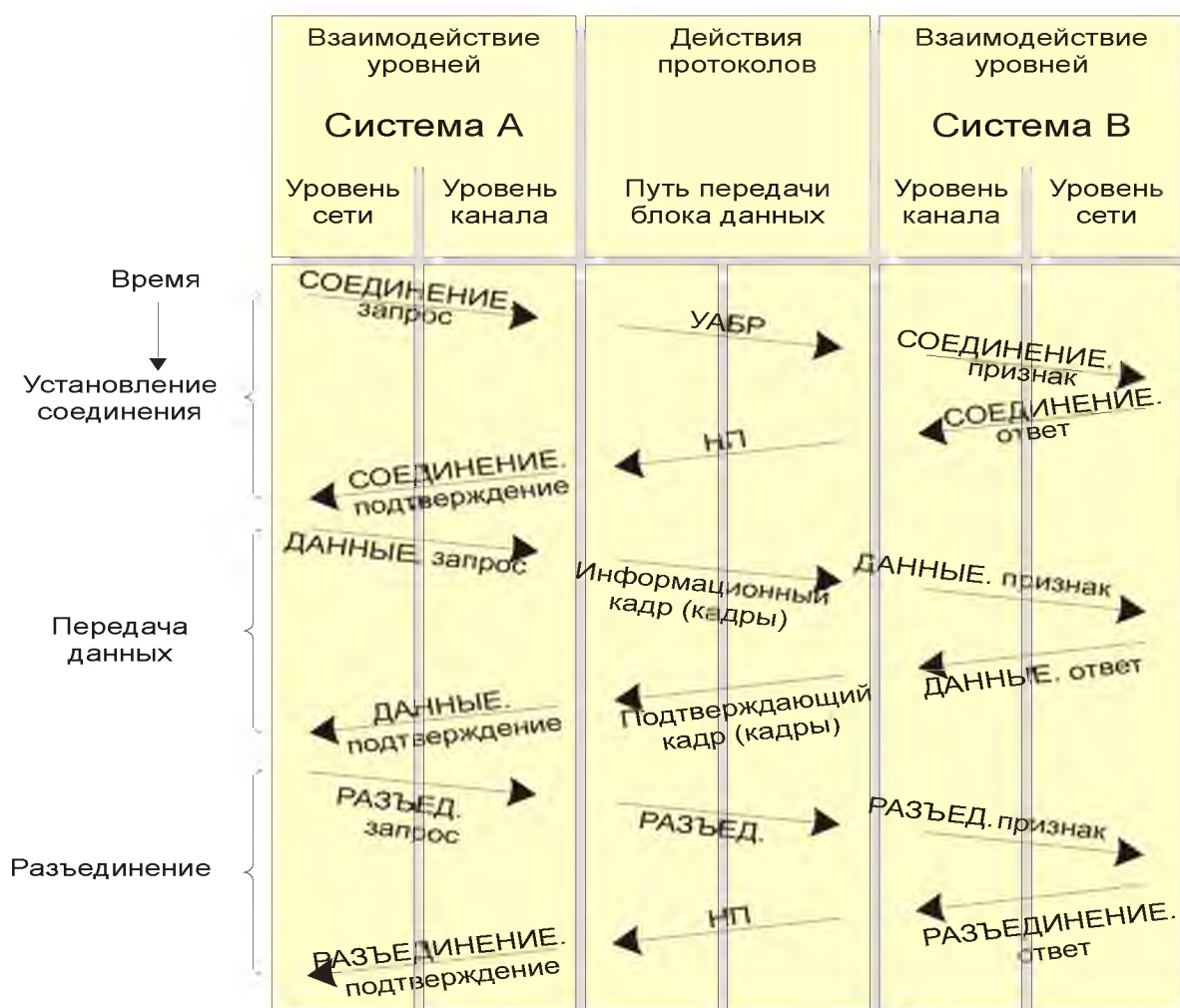


Рисунок 4.16-Применение примитивов протокола ВУК

Поэтому перед началом передачи должно быть установлено логическое соединение. Для этого система А выпускает примитив СОЕДИНЕНИЕ.запрос в свой уровень канала передачи данных, являющийся поставщиком услуги. После приема и соответствующей обработки этого примитива в систему В передается блок установки асинхронного балансного режима (УАБР), который приводит в действие объект уровня канала и выдается примитив СОЕДИНЕНИЕ.признак. В знак согласия на запрос об установлении связи объект уровня сети отвечает примитивом СОЕДИНЕНИЕ.ответ. Это вызывает посылку со стороны протокола канала системы В объекту уровня канала системы А блока нумерованного подтверждения НП. В системе А выдается примитив СОЕДИНЕ-

ННЕ.подтверждение, указывающий на завершение процесса установления подтверждения. Теперь на обоих концах сетевой уровень может начать передачу данных. Она будет происходить аналогично процедуре, рассмотренной в начале этого параграфа.

4.5.2.Виды протоколов

Различают три вида протоколов канального уровня [2]:

- протокол с остановками и ожиданием;
- протокол с N- возвратами (с непрерывной передачей);
- протокол с выборочной или селективной передачей.

1. Протокол с остановками и ожиданием

При этой процедуре одновременно может передаваться только один кадр. После этого передающая сторона ждет подтверждения. Если поступит отрицательное подтверждение или произойдет просрочка времени ожидания ответа, кадр передается повторно. Пакет сбрасывается из накопителя передающей стороны лишь после получения положительного подтверждения. Связь с остановками и ожиданием между двумя пунктами показана на рисунке 4.17.

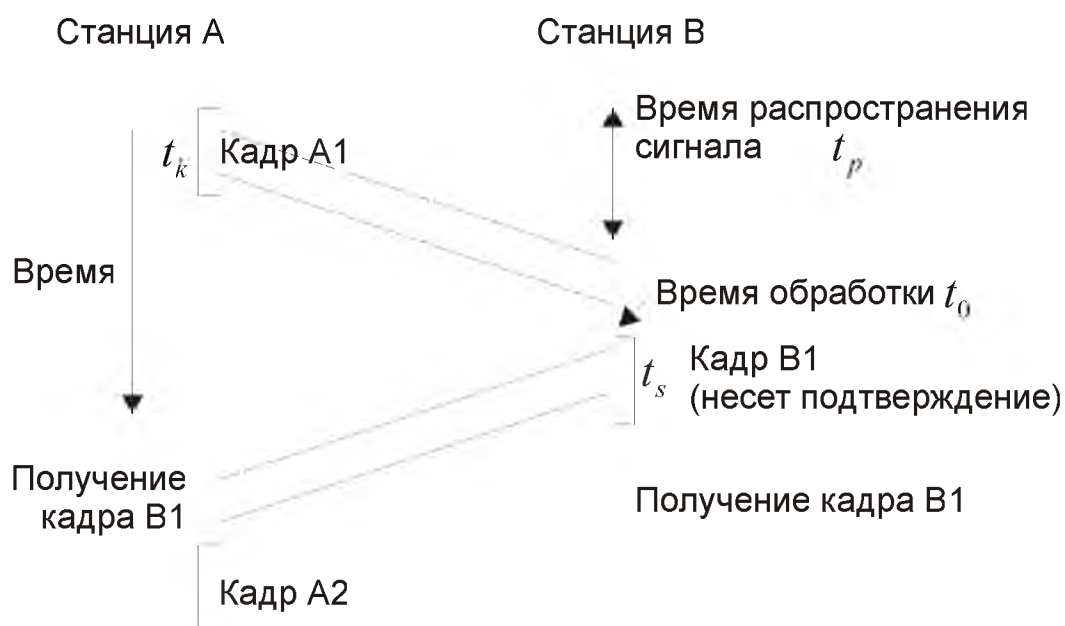


Рисунок 4.17-Протокол с остановками и ожиданием

Введем следующие обозначения:

t_n – перерыв в передаче кадров;

t_T – период передачи кадров.

Тогда, $t_n = 2t_p + t_0 + t_s$,

$$t_T = t_k + t_n,$$

где t_k - длительность передаваемого кадра;

t_s – длительность подтверждающего кадра.

Этот протокол подходит для полудуплексной передачи, при которой передача сторон чередуется.

2. Протокол с N -возвращениями, или непрерывная передача.

Здесь кадры передаются непрерывно без ожидания подтверждения (ПТВ). При получении отрицательного ("-" на рисунке) подтверждения или истечения установленного времени ожидания неподтвержденный кадр и все последующие кадры передаются вновь. Пример такой передачи представлен на рисунке 4.18.

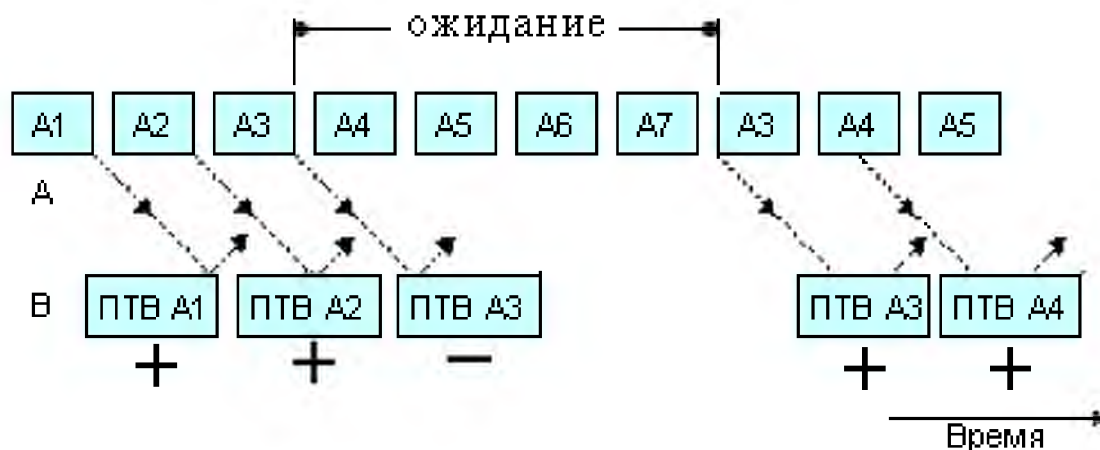


Рисунок 4.18-Протокол с N - возвращением

Этот протокол более производительный и предполагает использование дуплексной связи.

3. Протокол с выборочным повторением

В этом случае повторная передача требуется только для кадра, о котором поступило отрицательное подтверждение

или для которого истекло установленное время ожидания. Однако на приемном конце требуется накопитель с перестроениями, так как в этом случае кадры могут повторно передаваться и приниматься не по порядку.

Из-за увеличения стоимости реализации протокол выборочного повторения не нашел коммерческой реализации.

4.5.3. Анализ производительности протоколов

1. Протокол с остановками и ожиданием.

Рассмотрим следующую систему (рисунок 4.19).

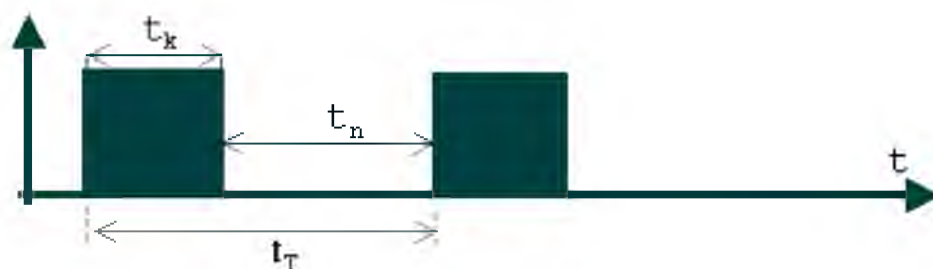


Рисунок 4.19-Передача кадра при анализе производительности протокола с остановками и ожиданием.

Предположим, что вероятность ошибочного приема в пункте В равна p . Тогда очевидно, что при отсутствии ограничений на число повторных передач среднее время правильной передачи найдется в виде:

$$t_v = t_T + (1-p) \cdot \sum_{i=1}^{\infty} i p^i t_T = \frac{t_T}{1-p}. \quad (4.1)$$

Это выражение показывает, что для i -го повторения кадр должен быть доставлен с ошибкой i раз. Вероятность правильного приема при i -м повторении равна $(1-p)$.

В случае насыщения величина t_v представляет собой среднее время между правильно переданными кадрами. Максимальная производительность в доставленных пакетах является обратной величиной t_v , или

$$\lambda_{\max} = \frac{1}{t_v} = \frac{1-p}{at_k} \quad (4.2)$$

где параметр $a = t_T/t_k \geq 1$ вводится, чтобы связать производительность с длиной кадра данных.

Если теперь принять λ равной практической интенсивности поступления кадров в передатчик, мы получим нормированную производительность для протокола с остановками и ожиданием в виде:

$$\rho = \lambda \cdot t_k \leq \frac{1-p}{a} < 1 \quad (4.3)$$

2. Протокол с N -возвращениями (см. рисунок 4.19).

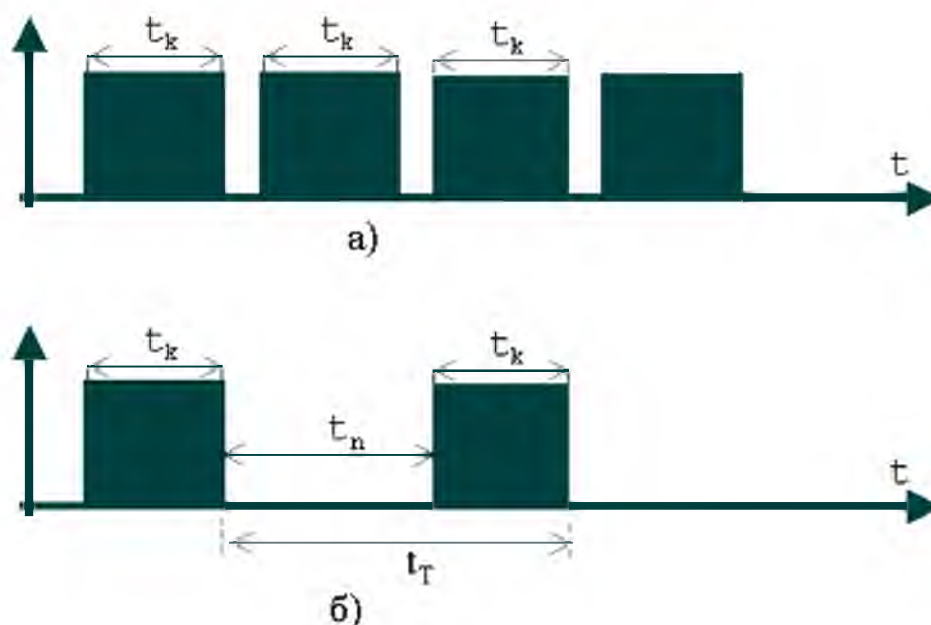


Рисунок 4.20-Анализ системы с N -возвращениями: (а) без-ошибочная передача, (б) случай появления ошибки (сбоя)

Поскольку по схеме с N -возвращениями кадры могут передаваться непрерывно один за другим, минимальное время между передачами равно t_k , то есть времени передачи кадра. Соответственно увеличивается максимальная производительность. Среднее время передачи кадра равно:

$$t_v = t_k + (1-p) \cdot \sum_{i=1}^{\infty} i p^i t_T = t_k + \frac{p \cdot t_T}{1-p} = t_k \cdot \left(\frac{1 + (a-1) \cdot p}{1-p} \right) \quad (4.4)$$

Максимально возможная производительность получается в виде:

$$\lambda_{\max} = \frac{1-p}{t_k \cdot (1+(a-1) \cdot p)}. \quad (4.5)$$

Нормированная производительность для любой интенсивности поступления кадров ограничивается значением

$$\rho = \lambda \cdot t_k = \frac{1-p}{1+(a-1) \cdot p}. \quad (4.6)$$

Пример:

Пусть $a = 4$, $p = 0.01$. Тогда для протокола с остановками и ожиданием нормированная производительность ограничивается значением $0.99/4$, тогда как соответствующая граница для схемы с N-возвращениями находится около $0.99/(1+0.01 \cdot 3) = 0.96$. То есть схема с N-возвращениями превосходит схему с остановками и ожиданием в 4 раза. При $a < 1$ предпочтительнее стратегия непрерывной передачи.

4.5.4. Определение скорости передачи полезной информации и оптимальной длины кадра

Проведенный простой анализ производительности дает также оценку наиболее предпочтительной длины пакетов (информационного поля в кадре данных), применяемых в сети. В большинстве сетей она выбирается равной около 1000 бит.

Обозначим длину пакета в кадре, выражаемую числом разрядов, через l , а число остальных разрядов, используемых в других полях через l' (см. рисунок 4.21) [11].



Рисунок 4.21 - Пакет

Пусть передающая станция А находится в состоянии насыщения и передает λ_{\max} кадров/с. Средняя скорость данных в бит/с, поступающих на принимающую станцию В, на основании выведенных в предыдущем пункте формул, равна

$$D = \lambda_{\max} \cdot l = \frac{(1-p) \cdot l}{t_k \cdot [1 + (a-1)p]} \quad (4.7)$$

Принимая $t_k = (1 + l')/C$, где C - пропускная способность канала в бит/с, для нормированной скорости поступления данных получим

$$V = \frac{D}{C} = \left(\frac{l}{l+l'} \right) \cdot \left[\frac{1-p}{1 + (a-1) \cdot p} \right]. \quad (4.8)$$

Это выражение ясно показывает влияние на скорость передачи данных длины пакета l , длины управляющего поля l' и вероятности ошибки. Длительность перерыва описывается нормированным параметром a .

Пусть p_B - вероятность искажения бита. Тогда вероятность ошибки в кадре определяется равенством

$$p = (l + l') \cdot p_B \quad \text{при } a=1. \quad (4.9)$$

Подставим полученное выражение в (4.2) и продифференцируем dv/dl . В результате оптимальная длина пакета равна

$$l_{opt} = \sqrt{\frac{l'}{p_B}} - l' \quad (4.10)$$

и не зависит от a .

4.5.5. Методы случайного доступа к сети

Двумя основными способами доступа к общей среде передачи являются управляемый доступ с применением опроса и случайный доступ. В свою очередь существуют различные типы стратегий случайного доступа.

Методы случайного доступа полностью децентрализованы. Пользователь может передавать когда угодно, лишь с незначительными ограничениями, зависящими от метода доступа.

Из-за случайности моментов времени, в которые пользователи могут решить начать передачу, независимо от метода не исключена возможность того, что два или несколько пользователей могут выйти на связь в пересекающиеся промежутки времени. Это приводит к столкновениям (*коллизиям*), которые сначала должны быть распознаны, а затем разрешены. При увеличении нагрузки увеличивается и вероятность коллизий, что приводит к возможной неустойчивости работы рассматриваемых механизмов.

В результате производительность ограничивается некоторым максимальным значением, меньшим пропускной способности канала, и это значение в каждом случае зависит от первоначального механизма доступа и алгоритма разрешения коллизий.

4.5.5.1. Методы Алоха

Сначала методы случайного доступа были предложены для случаев, когда многие пользователи пытаются довольно редко передать пакеты сообщений или когда друг с другом связываются небольшое число ЭВМ. Но применительно к производственным процессам, которые требуют строгого управления задержкой доступа, более предпочтителен управляемый доступ. Рассмотрим два простейших типа стратегии случайного доступа: *чистую Алоху* и *синхронную Алоху* [2].

Чистая Алоха

Эта схема сначала была применена для доступа к общему каналу сотрудниками Гавайского университета в начале 1970-х годов. По этой схеме пользователь, желающий передать сообщение, делает это когда угодно. В результате могут наложиться во времени два или несколько сообщений, вызвав столкновение (коллизию).

Распознавание коллизий и сообщение о них пострадавшим пользователям в первоначальной системе Алоха направлялись по радио на центральный пункт. Также это могло осуществляться применением положительных подтвержде-

ний в сочетании с перерывом. При обнаружении столкновения пострадавшие станции предпринимают попытки повторной передачи потерянного сообщения, но они должны распределять время попыток случайным образом, следуя некоторому алгоритму, уменьшающему возможность возникновения нового конфликта.

Стратегия доступа типа Чистая Алоха позволяет добиться производительности самое большее $1/2e \approx 0,18$ пропускной способности канала. Введем сначала некоторые определения. За доступ к каналу состязаются N станций. Станция передает, в среднем, λ пакетов в секунду (интенсивность обращений к сети). Величина $1/m$ представляет собой пропускную способность канала (μ) в передаваемых пакетах в секунду. Рассмотрим частный случай, при котором все передаваемые сообщения (пакеты) имеют среднюю длину, соответствующую m единицам времени передачи. Будем считать, что интенсивность нагрузки S (эквивалентно ρ - нормированной по μ нагрузке) характеризует использование канала вновь поступающими пакетами

$$S \equiv \rho = N\lambda m$$

Величина $1/m$, которая обозначается μ , представляет собой пропускную способность канала в передаваемых пакетах в секунду. Таким образом, $N\lambda/\mu = N\lambda m$ - относительное использование канала, или производительность, нормированная относительно μ . Общая интенсивность пакетов, передаваемых в канал, включая вновь генерируемые и передаваемые повторно, имеет некоторое значение $\lambda' > \lambda$ (из-за коллизий от каждого компьютера будет передаваться больше сообщений из-за необходимости возобновлять поток). Тогда фактическая интенсивность нагрузки, или использование канала, является параметром G , который равен

$$G = N\lambda' m.$$

Рассмотрим типичное сообщение длительностью m , показанное на рисунке 4.21.

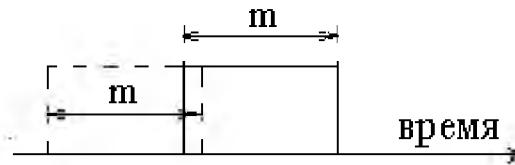


Рисунок 4.22-Столкновение двух сообщений

Оно подвергается столкновению с другим сообщением, если эти два сообщения будут наложены одно на другое в любой точке. Легко заметить, передвигая пунктирное сообщение во времени, что столкновение может произойти в промежутке времени продолжительностью $2m$ с. Вероятность того, что в промежутке $2m$ с не произойдет столкновения, равна $e^{-2N\lambda'm} = e^{-2G}$.

Отношение S/G представляет долю сообщений из числа передаваемых в канал, которые проходят успешно. Это число должно быть равно вероятности отсутствия столкновений. Таким образом уравнение производительности для чистой Алохи:

$$S = Ge^{-2G}. \quad (4.11).$$

Здесь S - нормированная производительность (средняя скорость поступления пакетов, деленная на максимальную производительность $1/m$), а G - нормированная пропущенная нагрузка. Таким образом, S – независимая переменная, а G - ее функция. График зависимости G от S имеет вид двузначной кривой (рисунок 4.23).

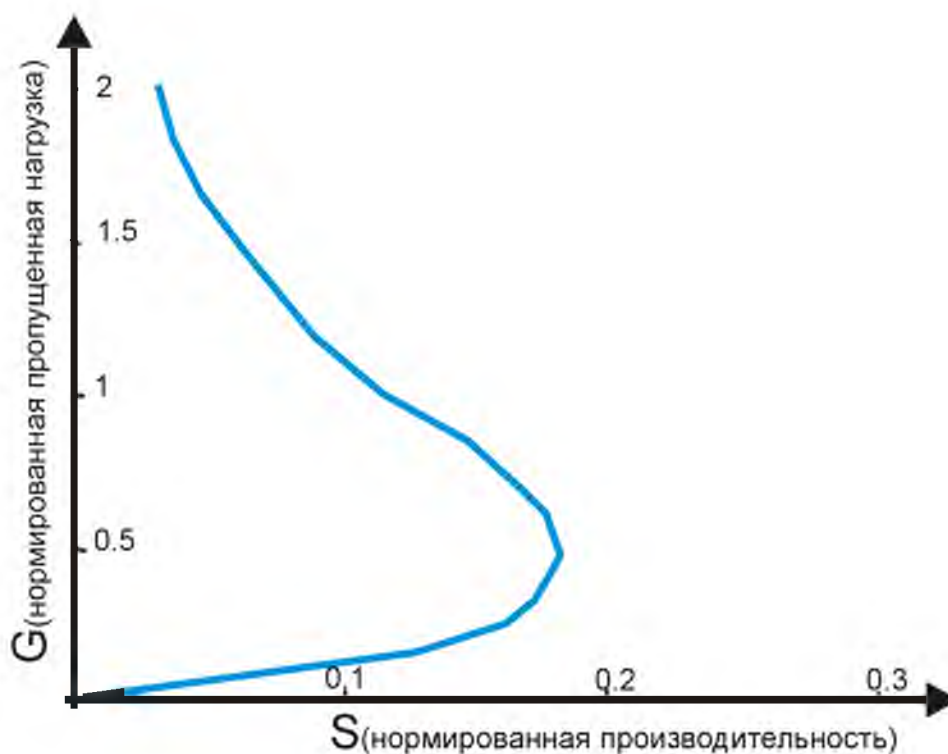


Рисунок 4.23-Характеристика производительности;
Чистая Алоха

Отметим, что S имеет максимум $S = 0,5e^{-1} \approx 0,18$ при $G = 0,5$. Судя по формуле (4.11) или кривой при малой поступающей нагрузке S столкновения происходят редко и $G \approx S$. Когда S начинает расти, приближаясь к максимальному значению 0,8, число столкновений быстро увеличивается, что ведет в свою очередь к росту вероятности столкновения. Система теряет устойчивость, S падает, а G увеличивается до больших значений.

Синхронная Алоха

Максимально возможная производительность схемы чистой Алохи может быть удвоена с помощью простого приема пазметки шкалы времени и разрешения пользователям начинать попытки передачи сообщений только в начале каждого временного интервала m (равного длительности сообщения). Эта схема требует, чтобы работа всех пользователей системы была синхронизирована во времени. Пример работы такой

системы показан на рисунке 4.24, на котором одно сообщение передано успешно, а с другим произошло столкновение.

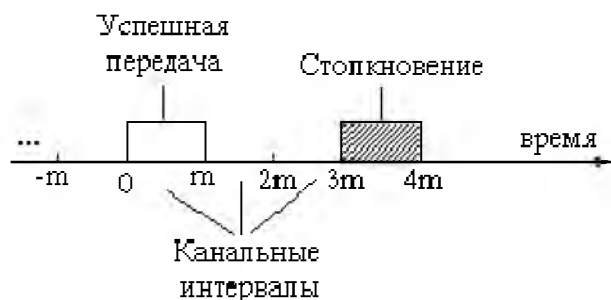


Рисунок 4.24-Передача при синхронной Алохе

Поскольку сообщения могут быть переданы только в размеченные промежутки времени, столкновения происходят лишь когда одна или несколько попыток передачи совершаются в том же промежутке.

Вероятность успешной передачи задается в виде e^{-G} , а производительность для синхронной Алохи имеет вид

$$S = Ge^{-G}.$$

Нормированная производительность S достигает максимального значения $1/e \approx 0,368$ при $G = 1$. Зависимость пропущенной нагрузки от производительности для синхронной Алохи показана на рисунке 4.25, где она сравнивается с соответствующей зависимостью для чистой Алохи.

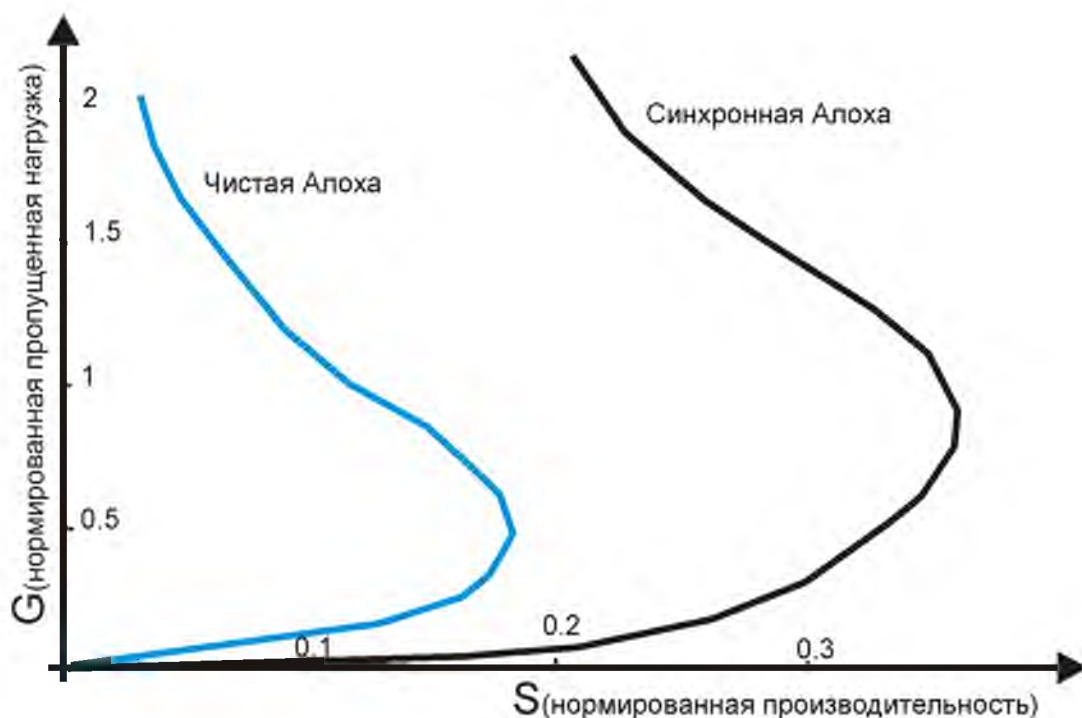


Рисунок 4.25-Характеристика производительности; Синхронная Алоха

Из приведенной характеристики очевидно, что ввиду двух возможных значений G при заданной производительности S для этой системы доступа также характерна неустойчивость.

4.5.5.2.Случайный доступ типа МДПН/ОС (CSMA/CD)

Протокол МДПН/ОС основан на методе чистой Алохи и позволяет улучшить ее характеристики. Метод МДПН/ОС входит в протокол сети Ethernet и принят, как один из стандартных методов в локальных сетях. Реализация локальных сетей по образцу сети Ethernet распространена весьма широко.

Основная концепция протокола МДПН/ОС очень проста [11]. Все станции *прослушивают* передачу по линии. Станция, желающая передать сообщение, выходит на связь только после обнаружения свободного состояния канала. Эта процедура называется *проверкой несущей*, а стратегия, основанная

на такой проверке, схемой многостанционного доступа с проверкой несущей (МДПН). Очевидно, столкновения все же могут возникнуть, поскольку станции физически разнесены одна от другой и две или несколько станций могут обнаружить свободное состояние канала и начать передачу, что и вызовет столкновение. Если станции обнаруживают столкновение (*обнаружение столкновения* - ОС), они передают всем остальным станциям специальный сигнал о помехе и отменяют свои передачи. Возможность проверки несущей позволяет увеличить производительность канала по сравнению с чистой Алохой, а обнаружение столкновения с прекращением передачи вместо его завершения дает еще большее повышение производительности.

Был предложен и проанализирован ряд методов МДПН. Они различаются тем, как происходит управление передачей, если канал оказался занятым. Например, в схеме с p -настойчивостью станция, обнаружившая занятый канал, осуществляет передачу после того, как канал станет свободным, с вероятностью p . С вероятностью $(1-p)$ передача откладывается на промежуток времени τ распространения сигнала. При схеме с 1-настойчивостью станция осуществляет попытку передачи, как только канал окажется свободным. При ненастойчивой схеме станция переносит передачу на другое время в соответствии с предписанным распределением задержек передачи, проверяет несущую в это время и продолжает процесс.

Эти схемы применимы прежде всего в локальных сетях или в более крупных сетях, работающих со сравнительно небольшими скоростями передачи.

Протокол МДПН/ОС, работающий по правилу 1-настойчивости с добавлением возможности обнаружения столкновений в целях дальнейшего улучшения характеристик принят в качестве протокола в схеме Ethernet. Если обнаруживается столкновение и передача прекращается, попытка повторной передачи принимается через случайный промежу-

ток времени, как и в схемах Алоха. Этот случайный промежуток времени удваивается каждый раз после обнаружения нового столкновения до некоторой максимальной величины, при которой станция выходит из строя и извещают вышестоящие уровни о нарушении связи. Удвоение промежутка называется процедурой двоичного замедления и может улучшить характеристику системы.

Проведем анализ системы с дискретным временем, чтобы выявить характеристики задержек протокола МДПН/ОС.

Рассмотрим шинную структуру.

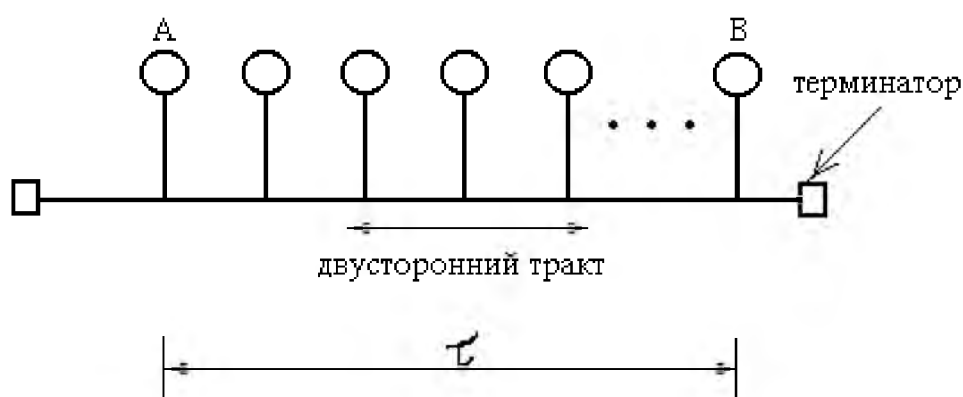


Рисунок 4.26-Модель МДПН/ОС

Станции подключаются через пассивные ответвления к двусторонней шине. Сосредоточим внимание на двух наиболее удаленных станциях (А и В). Рассчитаем среднее время, требуемое для успешного запуска сообщения в шину. Обратная величина такого времени и будет максимальной производительностью. Назовем время до успешного завершения передачи сообщения виртуальным временем передачи t_v . Это время имеет три составляющие(см. рисунок 4.27). Оно включает время t , требуемое для передачи сообщения, время τ , требуемое для проверки завершения передачи, и время, кратное 2τ единицам, для разрешения столкновений, если они обнаруживаются.

Пусть возникло столкновение между сигналами, передаваемыми станциями А и В. В худшем случае обнаружение

столкновения займет на станциях А и В время в 2τ с, после чего передача будет немедленно выключена. Это показано на рисунке 4.27 : станция А начинает передачу в некоторый момент времени. Перед тем как сообщение станции А поступит на станцию В, последняя решает начать передачу. Станция В проверяет канал, находит его свободным и начинает собственную передачу. Это очевидное столкновение, которое обнаружится только через τ с.

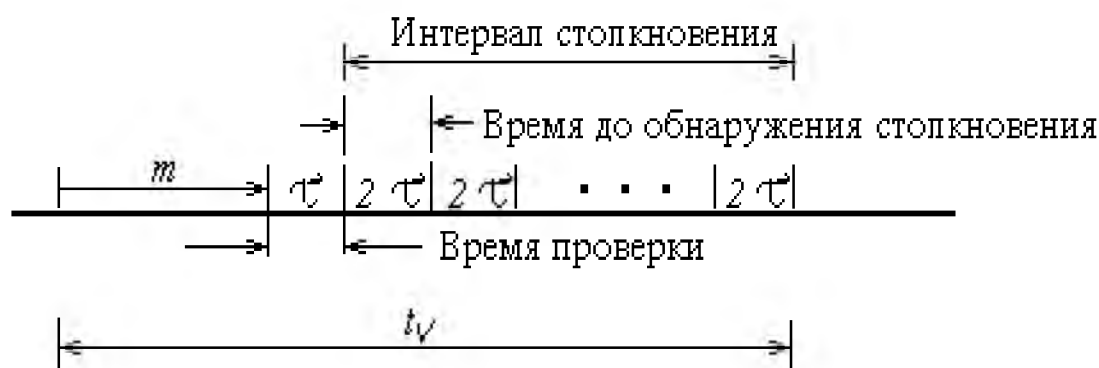


Рисунок 4.27-Расчет виртуального времени передачи
МДПН/ОС

Общее время до обнаружения столкновения составит 2τ единиц времени, что и показано на рисунке.

Если произошло столкновение, предположим, что для его разрешения потребуется $2\tau J$ единиц времени. J представляет собой среднее число повторных передач после того, как произошло столкновение. Оно сравнимо с параметром $E=G/S-1$, который был введен при изучении системы Алоха. Тогда виртуальное время передачи имеет вид:

$$t_v = m + \tau + 2\tau J = m [1 + a(1 + 2J)], \quad a \equiv \tau/m. \quad (4.11)$$

Далее найдем величину J . Она зависит от стратегии повторной передачи.

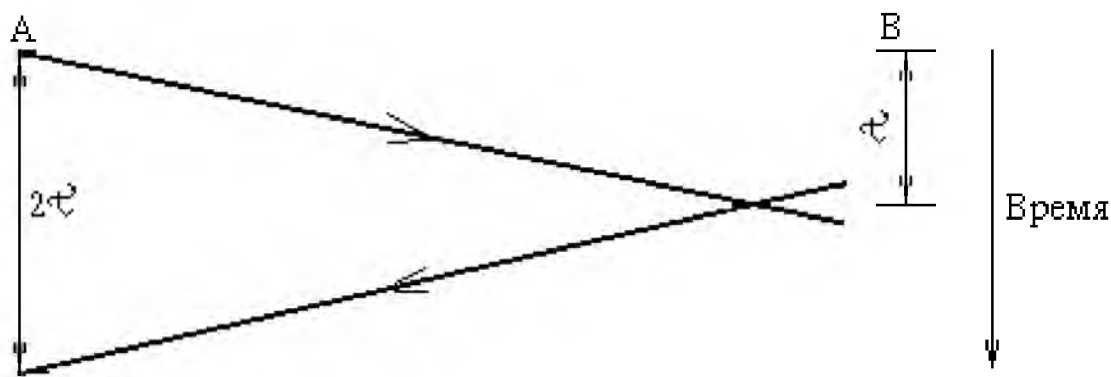


Рисунок 4.28-Наихудший случай обнаружения столкновения

Предположим, что длительность интервала столкновения (рисунок 4.27) описывается геометрическим распределением единиц 2τ с параметром ν . В частности, интервал равен одной единице (2τ) с вероятностью ν , двум единицам с вероятностью $\nu \cdot (1-\nu)$ и т.д. Таким образом, ν является вероятностью успеха в конце интервала, а $(1-\nu)$ - вероятностью столкновения. Среднее число повторных передач $J=1/\nu$, а именно

$$J = \sum_{k=1}^{\infty} J = k\nu (1-\nu)^{k-1} = 1/\nu \quad (4.12)$$

Это рассуждение переносит тяжесть нахождения J на ν .

Найдем вероятность ν .

Пусть в возможных передачах участвуют n станций ($n \gg 1$). Пусть вероятность того, что одна станция намеревается передавать в промежутке времени 2τ с, равна p . Тогда вероятность того, что передает одна станция, (а эта передача успешна)

$$\nu = n p (1-p)^{n-1}. \quad (4.13)$$

Используя величину $p=1/n$ и учитывая, как предполагалось, что

$n \gg 1$, в пределе получим, что

$$v_{\max} = (1-1/n)^{n-1} \rightarrow e^{-1}, \quad n \rightarrow \infty \quad (4.14)$$

Таким образом, величина v , которую нужно подставить в (4.12), равна e^{-1} . И равенство (4.11) принимает вид:

$$t_v = m [1+a(1+2e)], \quad a \equiv \tau / m. \quad (4.11a)$$

Заметим, что эта модель повторной передачи напоминает синхронную Алоху и фактически приводит к величине производительности синхронной Алохи e^{-1} .

Максимальная производительность λ_{\max} в числе сообщений за единицу времени равна $1/t_v$. Обозначая через λ среднее число сообщений, передаваемых по каналу за единицу времени от всех пользователей, и нормируя его относительно пропускной способности $1/m$ (в сообщениях за единицу времени), из (4.11a) находим

$$\rho \equiv \lambda m < \frac{1}{1+a(1+2e)} = \frac{1}{1+6.44a}, \quad a \equiv \tau / m.$$

Пусть в качестве примера $a=0,1$. Это значит, что длительность сообщений в десять раз больше времени распространения сигнала из конца в конец. Для этого значения a имеем

$$\rho < 0,6$$

Очевидно, это существенное улучшение по сравнению с эффективностью 0,18 для чистой Алохи и 0,368 для синхронной Алохи. Если значение a уменьшить еще больше (путем сокращения длины кабеля или уменьшения пропускной способности в бит/с в целях увеличения m), соответственно уве-

личится ρ_{max} , приближаясь к максимально возможному значению, равному 1.

Манчестерский код

Кроме проверки двух сигналов - обнаружения столкновения и проверки несущей, - блоки доступа к каналу передают символы в коаксиальный кабель и принимают их из кабеля. Блок кодирования передаваемых данных физического уровня кодирует символы в двоичные сигналы с помощью манчестерского кода. Пример такого кода показан на рисунке 4.29.

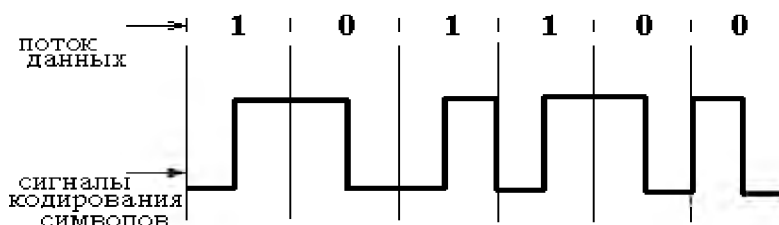


Рисунок 4.29-Манчестерский код

При этой схеме половина символьного интервала применяется для передачи логического дополнения к разряду этого интервала; в течение второй половины передается исходное значение этого разряда. Таким образом, единицы передаются положительным переходом сигнала, а нули - отрицательным переходом (рисунок 4.29). Функции кодирования/декодирования манчестерского кода выполняются передающим блоком кодирования и приемным блоком декодирования физического уровня. Эти блоки также генерируют и удаляют 64-разрядные серии, называемые преамбулами, которые предшествуют фактически передаваемому кадру и применяются для синхронизации.

Процедура кодирования, определенная стандартом для кольца с передачей метки, предусматривает применение

дифференциального манчестерского кода. Этот метод отличается от применения описанного выше манчестерского кода.

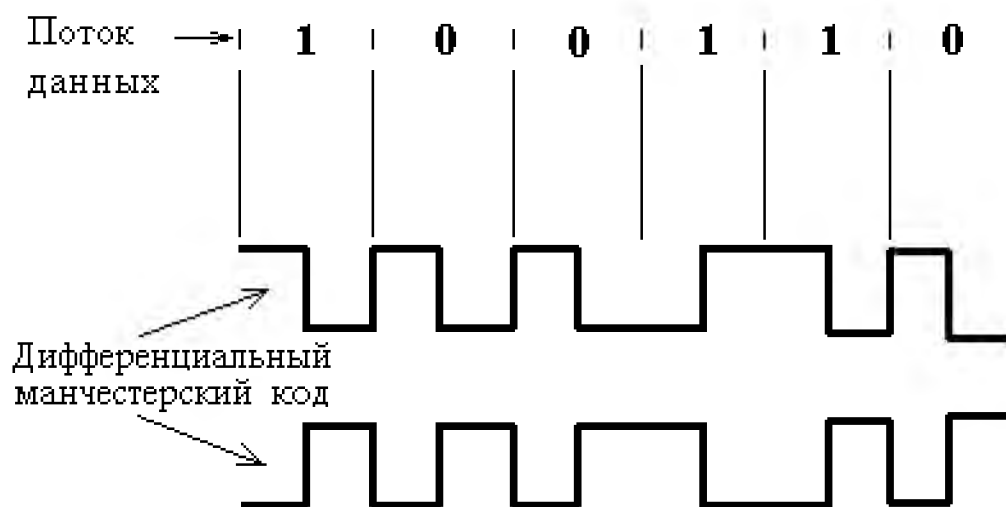


Рисунок 4.30-Дифференциальный манчестерский код

В дифференциальном манчестерском коде для переноса двоичной информации применяются две полярности и переходы происходят в середине двоичного интервала. Однако, в случае разряда 1 первая половина двоичного интервала несет ту же полярность, что и вторая половина предыдущего интервала. В случае же разряда 0 переход происходит как в начале, так и в середине двоичного интервала. Пример на рисунке 4.30 показывает, что при этой процедуре возникают две возможности в зависимости от полярности в конце интервала, предшествующего первому интервалу.

4.5.6.Спецификации ETHERNET

Ethernet - самая популярная в настоящее время сетевая архитектура. Она использует узкополосную передачу со скоростью 10 Мбит/с, топологию "шина", а для регулирования трафика в основном сегменте кабеля - CSMA/CD (МДПН/ОС) [1].

Среда (кабель) Ethernet является пассивной, то есть получает питание от компьютера. Следовательно, она прекратит работу из-за физического повреждения или неправильного подключения терминатора.

Сеть Ethernet имеет следующие характеристики:

- традиционная топология - линейная шина
- другие топологии - звезда-шина
- тип передачи - узкополосная
- метод доступа - CSMA/CD
- скорость передачи данных - 10 и 100 Мбит/с
- кабельная система - тонкий и толстый коаксиальный, UTP.

Формат кадра

Ethernet разбивает данные на пакеты (кадры), формат которых отличается от формата пакетов, используемого в других сетях. Кадры представляют собой блоки информации, передаваемые как единое целое. Кадр Ethernet может иметь длину от 64 до 1518 байтов, но сама структура кадра Ethernet использует по крайней мере 18 байтов, поэтому размер блока данных в Ethernet - от 46 до 1500 байтов. Каждый кадр содержит управляющую информацию и имеет общую с другими кадрами организацию.

Например, передаваемый по сети кадр Ethernet II используется для протокола TCP/IP. Кадр состоит из частей, которые перечислены в таблице.

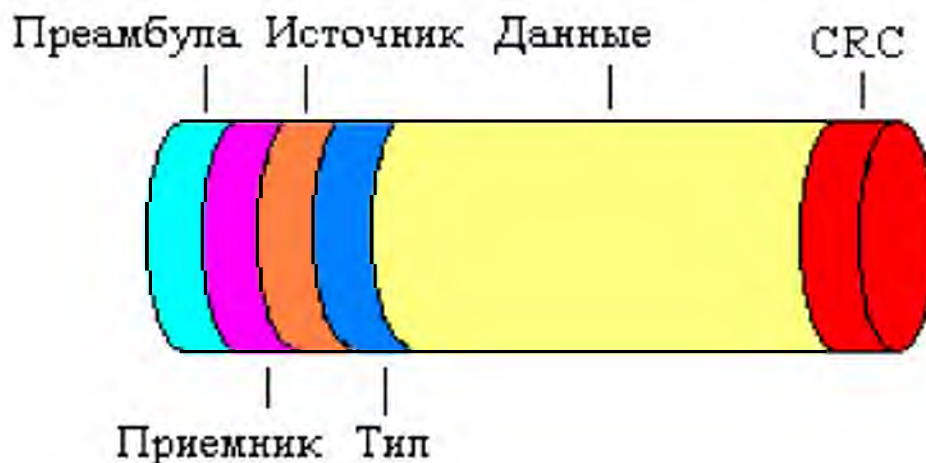


Рисунок 4.31-Кадр данных Ethernet II

Поле кадра	Описание
Преамбула	Отмечает начало кадра
Местоназначение и источник	Указывает адрес источника и адрес приемника
Тип	Используется для идентификации протокола Сетевого уровня
Циклический избыточный код	Поле информации для проверки ошибок

Сети Ethernet используют различные варианты кабелей и топологий. Далее будут представлены варианты, основанные на спецификации IEEE.

3.5.6.1. Стандарты IEEE на 10 Мбит/с

Ниже будут рассмотрены четыре топологии Ethernet со скоростью передачи данных 10 Мбит/с:

- 10 Base T

- 10 Base 2
- 10 Base 5
- 10 Base FL

10 Base T

В 1990 году IEEE опубликовал спецификацию 802.3 для построения сети Ethernet на основе витой пары. 10BaseT(10 - скорость передачи в Мбит/с, Base - узкополосная, T - витая пара) - сеть Ethernet, которая для соединения компьютеров обычно использует неэкранированную витую пару (UTP). Тем не менее и экранированная витая пара (STP) также может применяться в топологии 10BaseT без изменения каких-либо ее параметров.

Большинство сетей этого типа строятся в виде звезды, но по системе передачи сигналов представляют собой шину, как и другие конфигурации Ethernet. Обычно концентратор сети 10BaseT выступает как многопортовый репитер и часто располагается в распределительной стойке здания. Каждый компьютер подключается к другому концу кабеля, соединенного с концентратором, и использует две пары проводов: одну - для приема, другую - для передачи.

Максимальная длина сегмента 10BaseT - 100 м. (рис. 4.32). Минимальная длина кабеля - 2.5 м. ЛВС 10BaseT может обслуживать до 1024 компьютеров.

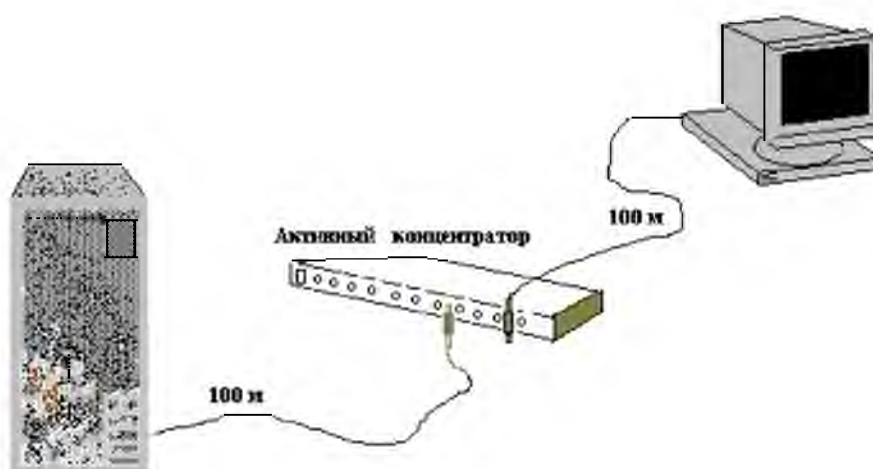


Рисунок 4.32-Для увеличения длины кабеля используется активный концентратор

На рисунке 4.33 показано, как сеть 10BaseT реализует преимущества топологии «звезда». Кабель UTP обеспечивает скорость передачи данных 10 Мбит/с. Изменение конфигурации производится на коммутационных панелях – простым переключением шнура из одного гнезда в другое. Эти изменения не затрагивают другие сетевые устройства.

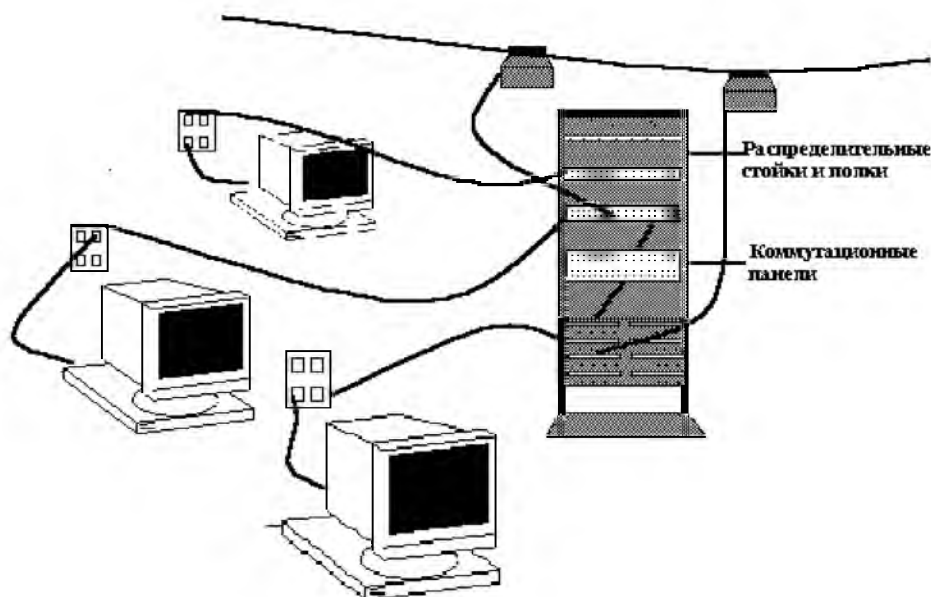


Рисунок 4.33-Коммутационные панели упрощают изменение
в конфигурации сети

10 Base 2

В соответствии со спецификацией IEEE 802.3 эта топология называется 10Base2[2- передача на расстояние, примерно в два раза превышающее 100м].

Сеть такого типа ориентирована на тонкий коаксиальный кабель, или тонкий Ethernet , с максимальной длиной сегмента 185 м. Минимальная длина кабеля 0.5 м. Кроме того, существует ограничение на максимальное количество компьютеров, которое может быть размещено на 185 - метровом сегменте кабеля, - 30 штук.

Компоненты кабеля "Тонкий Ethernet":

- BNC - баррел-коннекторы;

- BNC T - коннекторы;
- BNC - терминаторы.

Сети на тонком Ethernet обычно имеют топологию "шина". Стандарты IEEE для тонкого Ethernet не предусматривают использования кабеля трансивера между T-коннектором и компьютером. Вместо этого T-коннектор располагают непосредственно на плате сетевого адаптера.

BNC баррел-коннектор, соединяя сегменты кабеля, позволяет увеличить его общую длину. Например, вам нужен кабель длиной 30 м, а у вас есть сегменты тонкого кабеля по 20 м и 5 м. Соедините двумя баррел-коннекторами эти сегменты, чтобы получить кабель нужной длины. Однако использование баррел-коннекторов желательно свести к минимуму, поскольку они ухудшают качество сигнала.

Сеть на тонком Ethernet - экономичный способ реализации сетей для небольших отделений и рабочих групп. Используемый в такого типа сетях кабель:

- относительно недорогой
- прост в установке
- легко конфигурируется

По спецификации IEEE 802.3 сеть на тонком Ethernet может поддерживать до 30 узлов (компьютеров и репитеров) на один кабельный сегмент.

Правило 5-4-3 для 10 Base 2

Сеть на тонком Ethernet может состоять максимум из 5 сегментов кабеля, соединенных 4 репитерами, но только к трем сегментам могут быть подключены рабочие станции. Таким образом, 2 сегмента остаются зарезервированными для репитеров, их называют межрепитерными связями. Такая конфигурация известна как правило 5-4-3.

На рисунке 4.34 вы видите пять магистральных сегментов, четыре репитера. К магистральным сегментам 1,2,5 подключены компьютеры. Магистральные сегменты 3,4 предназначены только для увеличения общей длины сети.

Поскольку для сетей на тонком Ethernet ограничения слишком жесткие, большие предприятия, чтобы соединить сегменты и увеличить общую длину сети до 925 метров, используют репитеры.

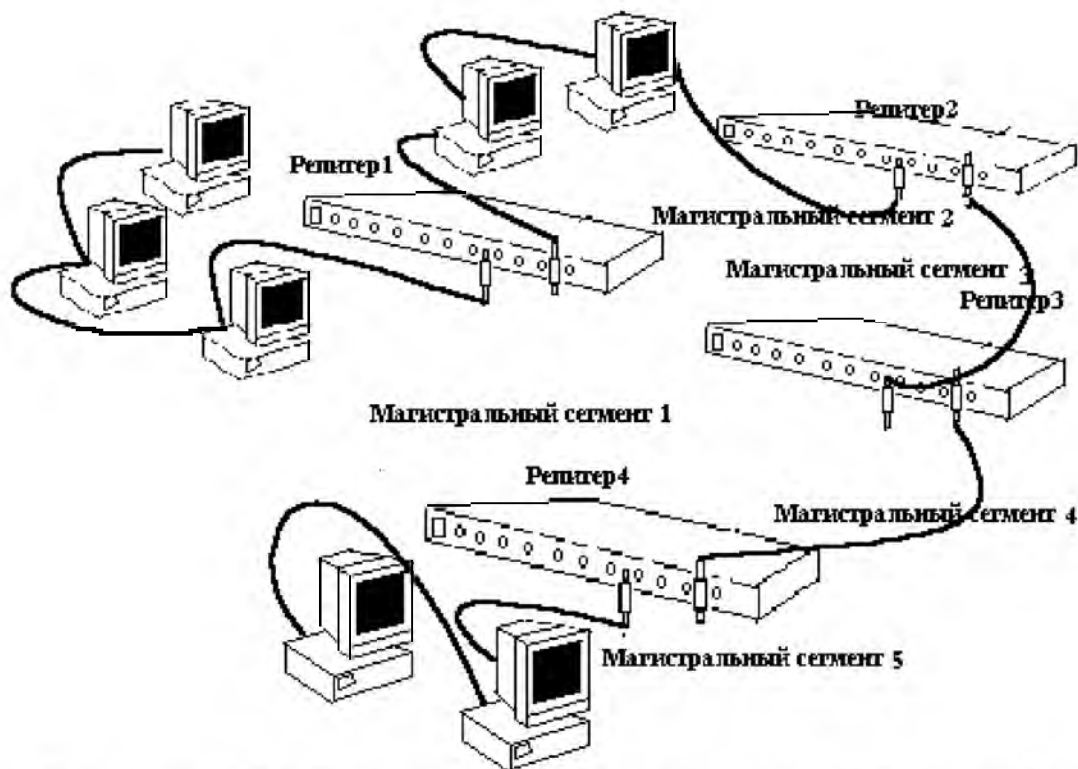


Рисунок 4.34-Правило 5-4-3 для тонкого Ethernet: 5 сегментов,
4 репитера, 3 сегмента для подключения станций

10 Base 5

В соответствии со спецификацией IEEE эта топология называется 10Base5. Известно и другое ее название - стандартный Ethernet.

Сети на толстом коаксиальном кабеле (толстый Ethernet) обычно используют топологию "шина". Толстый Ethernet может поддерживать до 100 узлов на магистральный сегмент. Магистраль - главный кабель, к которому присоединяются трансиверы с подключенными к ним рабочими станциями и репитерами. Сегмент толстого Ethernet может иметь длину 500 м при общей длине сети 2500 м.

Расстояния и допуски для толстого Ethernet больше, чем для тонкого Ethernet.

Компоненты кабельной системы:

- Трансиверы

Трансиверы, обеспечивая связь между компьютером и главным кабелем ЛВС, совмещены с "зубом вампира", соединенным с кабелем.

- Кабели трансиверов.

Кабель трансивера (ответвляющийся кабель) соединяет трансивер с платой сетевого адаптера.

- DIX - коннектор, или AUI - коннектор.

Расположен на кабеле трансивера.

- Коннекторы N - серии (в том числе баррел - коннекторы) и терминаторы N - серии.

Компоненты толстого Ethernet работают также, как компоненты тонкого Ethernet

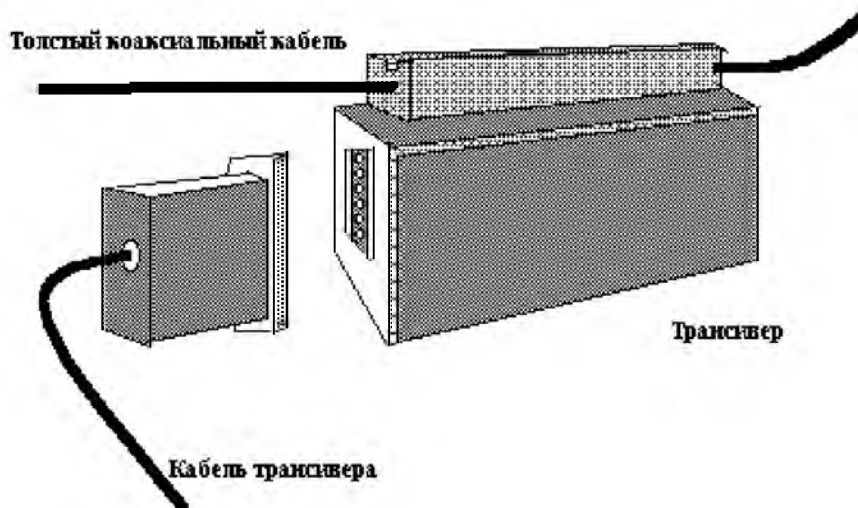


Рисунок 4.35-Толстый Ethernet с подключенным трансивером

Правило 5-4-3 для 10 Base 5

Сеть на толстом Ethernet может состоять максимум из пяти магистральных сегментов, соединенных репитерами, но только к трем сегментам при этом могут быть подключены компьютеры. При вычислении общей длины кабеля "толстый

Ethernet" длина кабеля трансивера не учитывается, то есть в расчет принимают только длину сегмента кабеля "толстый Ethernet".

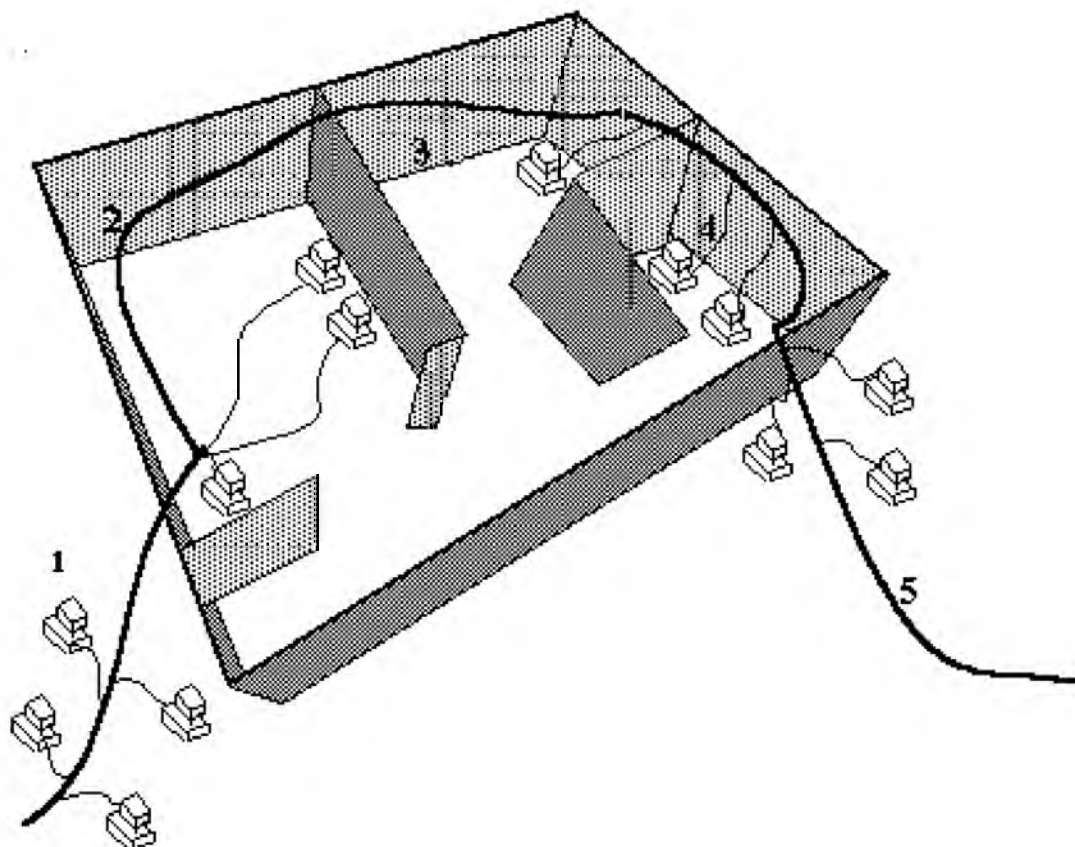


Рисунок 4.36-Правило 5-4-3 для толстого Ethernet:
5 сегментов, 4 репитера, 3 сегмента для подключения компьютеров

Минимальное расстояние между соседними подключениями - 2.5 м. В это расстояние не входит длина кабеля трансивера. Толстый Ethernet был разработан для построения ЛВС в рамках большого отдела или всего здания.

Комбинирование толстого и тонкого Ethernet

Обычно в крупных сетях используют толстый и тонкий Ethernet. Толстый Ethernet хорошо подходит в качестве магистрали, а для ответвляющихся сегментов применяют тонкий Ethernet. Трансивер соединяется с кабелем "Толстый

Ethernet", AUI - коннектор кабеля трансивера включается в репитер. Отвечающие сегменты тонкого Ethernet соединяются с репитером, а к ним уже подключаются компьютеры.

10 Base FL

10BaseFL представляет собой сеть Ethernet, в которой компьютеры и репитеры соединены оптоволоконным кабелем.

Основная причина популярности 10BaseFL - возможность прокладывать кабель между репитерами на большие расстояния (например, между зданиями). Максимальная длина сегмента 10BaseFL - 2000 м.

3.5.6.2. Стандарты IEEE на 100 Мбит/с

Новые стандарты Ethernet позволяют преодолеть скорость передачи в 10 Мбит/с. Эти новые возможности разрабатываются для таких приложений, порождающих интенсивный трафик, как:

- CAD - (системы автоматического проектирования);
- CAM - (системы автоматического производства);
- видео;
- отображение и хранение документов.

Известно два стандарта Ethernet, которые могут удовлетворить возросшие требования:

- 100BaseVG-AnyLAN Ethernet;
- 100BaseX Ethernet (Fast Ethernet).

И Fast Ethernet и 100BaseVG - AnyLan работают примерно в пять-десять раз быстрее, чем стандартный Ethernet. Кроме того, они совместимы с существующей кабельной системой 10BaseT. Это означает, что перейти от нее к этим стандартам достаточно просто и быстро.

4.6.Сетевой уровень модели OSI

Функции сетевого уровня:

- управление потоком и предотвращение блокировок передач;
- маршрутизация в сети.

4.6.1. Методы коммутации в компьютерных сетях

Магистральный канал передачи данных состоит из отдельных линий связи и узлов коммутации, которые обеспечивают соединение территориально удаленных абонентов между собой. Установление соединения (физического или виртуального) осуществляется с помощью того или иного метода коммутации. В зависимости от методов установления соединения и способов передачи данных от одного узла и другому узлу различают сети с коммутацией каналов, коммутацией сообщений и коммутацией пакетов [2].

4.6.1.1. Сети с коммутацией каналов

В сетях с коммутацией каналов между вызывающей и вызываемой оконечными установками в течение всего времени передачи имеется сквозное соединение (рисунок 4.37).

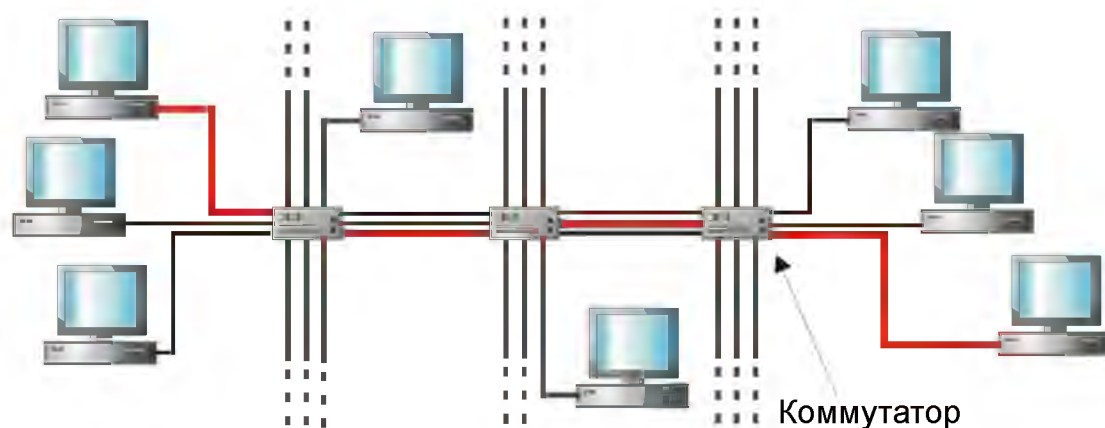


Рисунок 4.37-Сеть с коммутацией каналов

Соединительный тракт состоит из ряда участков, которые в процессе установления соединения включаются последовательно друг за другом. Он является «прозрачным» в от-

ношении кодов и методов управления. Время распространения сигнала данных по соединительному тракту постоянно.

В сеансе связи различают три фазы: установление соединения, передачу данных и разъединение соединения (рисунок 4.38).

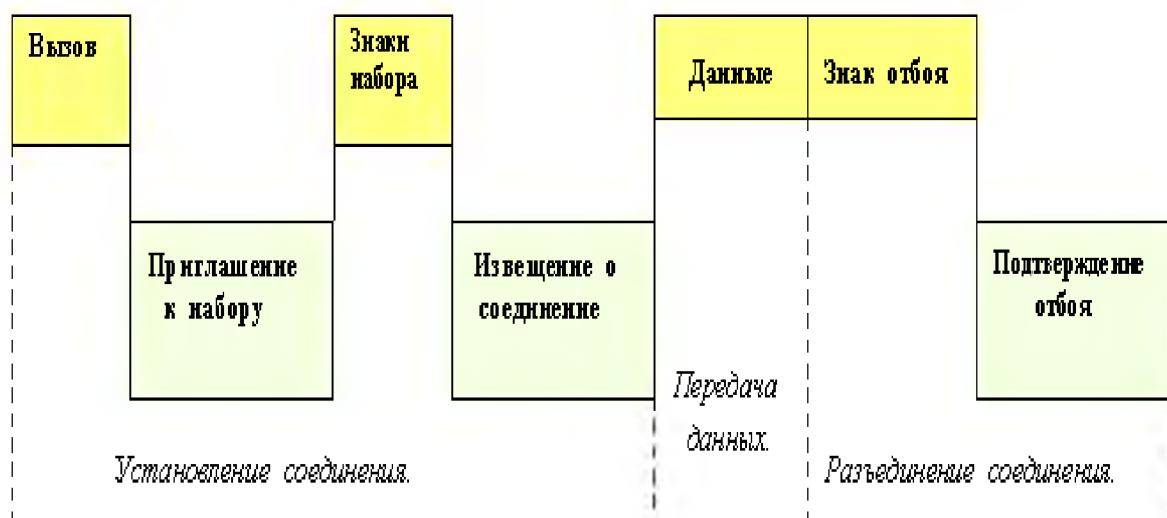


Рисунок 4.38-Фазы сеанса связи

Процессом установления соединения управляет источник, который посылает сигнал вызова, получает ответный сигнал (приглашение к набору номера) и вслед за этим передаёт адресную информацию (знаки набора номера). Коммутационный узел обрабатывает эту информацию, занимает один из каналов в пучке, ведущем к следующему коммутационному узлу, и передает последнему знаки набора, необходимые для дальнейшего установления соединения. Таким образом, постепенно, по участкам, вплоть до вызываемого абонента образуется соединительный тракт. После завершения этого процесса от сети на вызывающую и вызываемую оконечные установки поступают сигналы, извещающие о том, что соединение готово к передаче данных.

В течение фазы передачи данных управление осуществляется оконечной установкой. В оконечной установке принимается решение о мерах, которые необходимо принять для обнаружения и исправления ошибок передачи.

Разъединение может быть начато любой из двух связанных между собой оконечных установок с помощью сигнала отбоя. По этому сигналу все коммутационные узлы, участвующие в образовании соединительного тракта, отключают соединения. Среди сетей передачи данных с коммутацией каналов различают два типа: синхронные и асинхронные сети.

В асинхронных сетях общая синхронизация по элементам отсутствует и для сети не задаются единые «такты». Отдельные аппаратура передачи данных и коммутационные устройства имеют самостоятельные, независимые друг от друга тактовые генераторы.

В синхронной сети с коммутацией каналов ход во времени всех процессов передачи и коммутации определяется единым тактовым синхросигналом. Он подводится ко всей аппаратуре и оборудованию сети, задаёт для всей сети жесткий временной растр и обеспечивает синхронизм всех процессов.

4.6.1.2. Сети с коммутацией сообщений

В сети с коммутацией сообщений между оконечными установками, обменивающимися информацией, нет сквозного соединения. В коммутационных узлах сообщения заносятся в память и передаются далее по участкам переприёма от узла к узлу.

На рисунке 4.39 показана схема сети с коммутацией сообщений.

На оконечной установке, от которой необходимо передать сообщение, оно снабжается заголовком, содержащим адрес желаемого абонента, и по абонентской линии *a* передается на ближайший коммутационный узел **A**. В нем сообщение записывается, обрабатывается его заголовок, определяется в ка-

кую из исходящих линий далее его нужно направить и, наконец, передаётся на следующий коммутационный узел **В**.

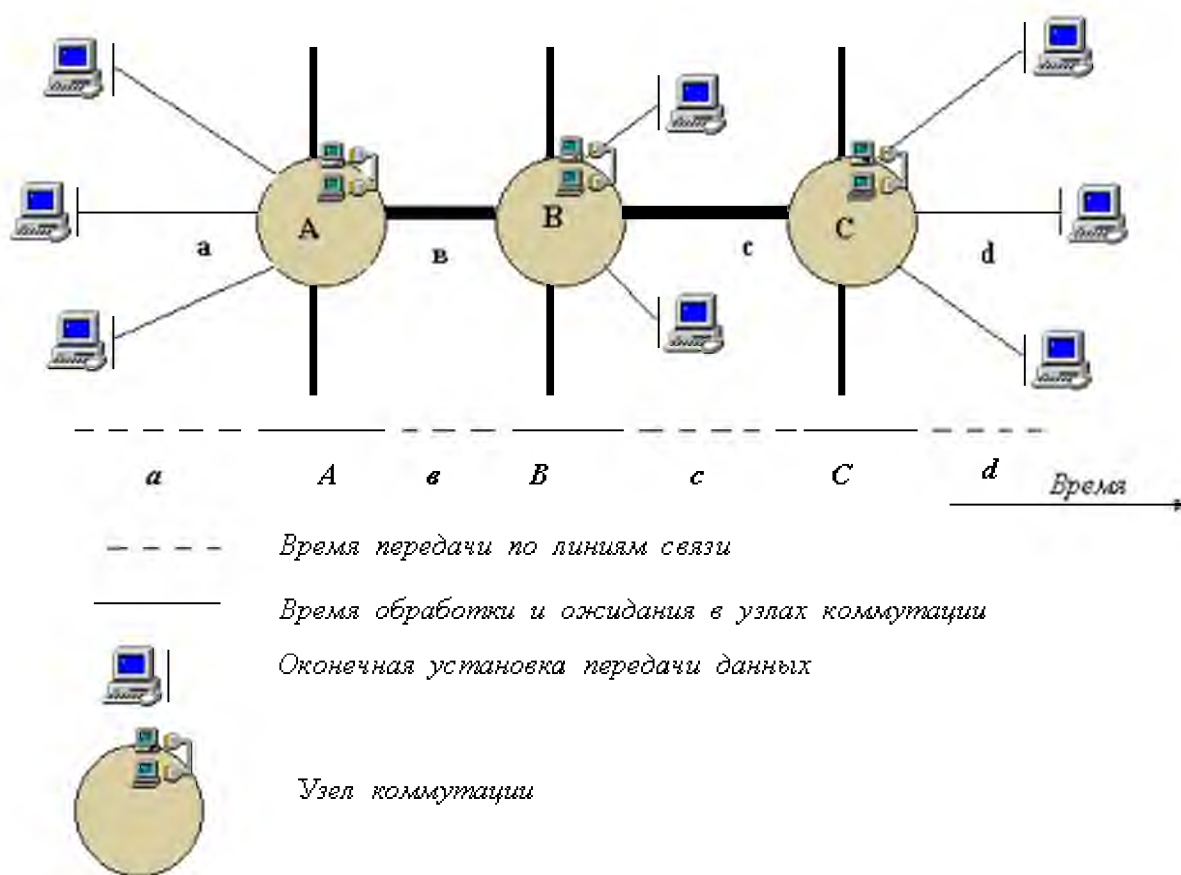


Рисунок 4.39-Сеть с коммутацией сообщений

Если линия *б*, по которой нужно передать сообщение далее, занята, то оно остаётся в запоминающем устройстве (буферном накопителе узла) до тех пор, пока не будут переданы все находящиеся перед ним в очереди другие сообщения. Поскольку на участке в магистральном канале скорость передачи обычно выше, чем в абонентской линии *а*, длительность передачи по этому участку на временной диаграмме (см. рисунок 4.38) показана более короткой. Узел коммутации **В**, в котором сообщение обрабатывается так же, как и в узле **А**, передает сообщение по линии *с* на узел **С**. От последнего оно передается по абонентской линии *д* на принимающий модуль абонента.

Время ожидания, в течение которого сообщение хранится в узле коммутации, зависит от длины очередей на линии связи, поэтому общее время прохождения сообщения между двумя оконечными установками в сети может быть различным. Запись сообщений в память упрощает трансформацию скоростей различного оборудования данных, осуществляемую в коммутационных узлах. Использование на межузловых участках дуплексных высокоскоростных линий связи позволяет более эффективно, чем в сетях с коммутацией каналов, передавать требуемый объём информации и использовать ресурсы сети. Однако экономию линий связи необходимо сопоставлять с затратами, которых требуют запоминание и обработка сообщений в узлах коммутации.

4.6.1.3. Сеть с пакетной коммутацией

Коммутация пакетов является развитием метода коммутации сообщений. Она позволяет добиться дальнейшего увеличения пропускной способности сети, скорости и надёжности передачи данных. В сети с коммутацией пакетов сообщения разделяются на отдельные части, называемые пакетами (см. рисунок 4.39). Каждый пакет имеет, как правило, фиксированную длину и снабжается заголовком, указывающим адрес пункта отправления, адрес пункта назначения и номер пакета в сообщении. Максимальная длина пакета лежит в пределах от 10^4 до $2 \cdot 10^4$ бит. Разложение сообщения на пакеты и восстановление его после передачи осуществляются оконечным оборудованием источника и адресата.

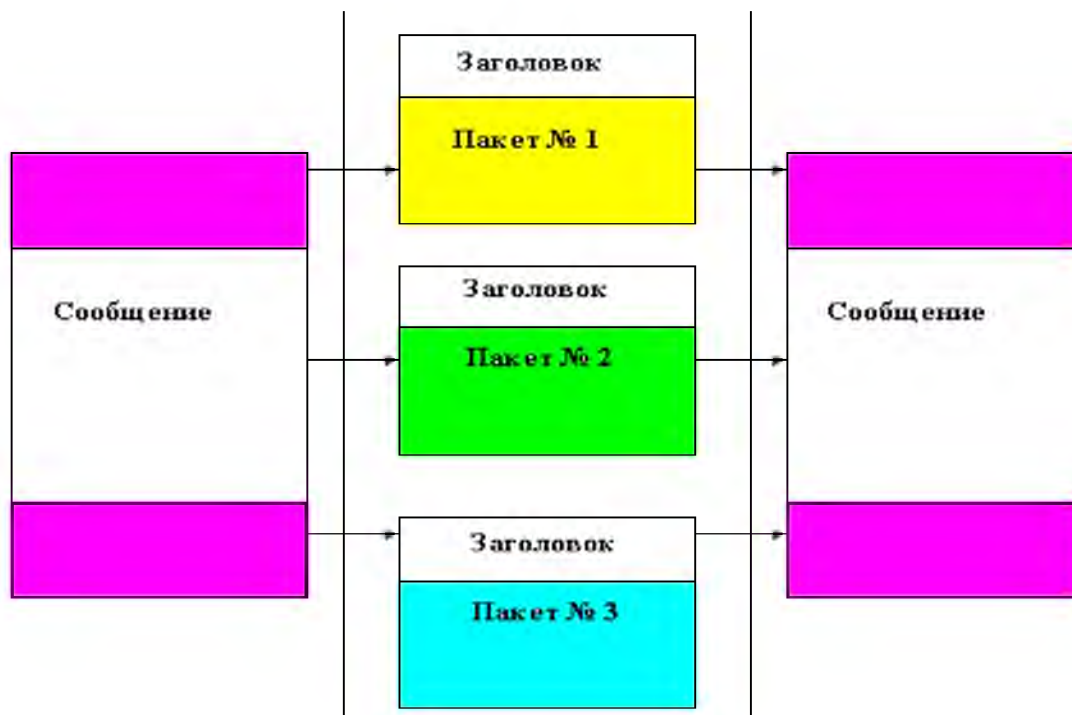


Рисунок 4.40-Пакетирование сообщений

В принимающем коммутационном узле каждый пакет проверяется на наличие ошибок. На пакеты, принятые без ошибок, в ответ направляется подтверждение их приёма (положительная квитанция ДА). Если же в пакете обнаружены ошибки, то посылается запрос на его повторную передачу (отрицательная квитанция НЕТ).

Для передачи отдельных пакетов каждого сообщения по сети могут выбираться различные пути (рисунок 4.41), что обеспечивает более гибкое и оперативное приспособление к состояниям занятости тех или иных линий связи и узлов коммутации. При этом могут возникнуть ситуации, при которых пакеты могут поступать в адрес получателя в неверной последовательности.

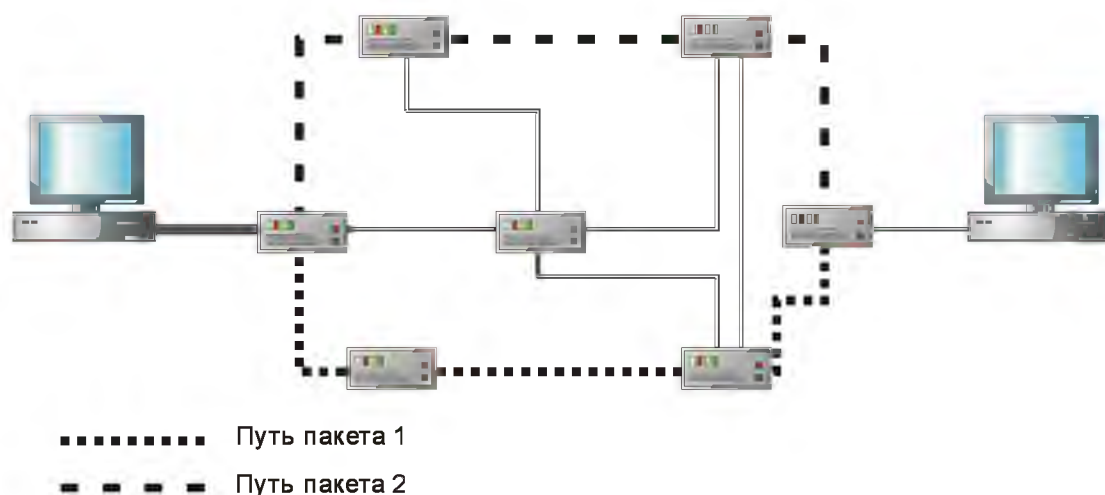


Рисунок 4.41-Различные пути передачи пакетов

Известны два способа передачи пакетов в сети передачи данных: виртуальный и датаграммный. При работе в режиме виртуального канала коммутационный узел пункта назначения сортирует поступившие на него пакеты перед их дальнейшей передачей на конечную станцию. В датаграммном режиме передачи сортировка возлагается на принимающую конечную станцию.

Метод коммутации пакетов по сравнению с другими методами обеспечивает наименьшую задержку при передаче данных и наибольшую пропускную способность сети.

4.6.2. Управление потоком в сети

На сетевом уровне, в отличие от канального, в котором управление идет между двумя точками, должно быть организовано управление потоком по всему виртуальному каналу от источника до получателя. При этом предполагается, что канальный уровень справляется со своими задачами и ошибок в передаче между узлами нет.

Если в сеть поступает большое число пакетов, то заметно растут задержки; производительность, измеряемая числом пакетов, доставляемых по назначению в единицу времени, начинает снижаться. Если поступающая нагрузка достаточно высока, может даже возникнуть тупиковая ситуация, когда

все накопители оказываются переполнены, поступление нагрузки прекращается и производительность падает до нуля.

Для предотвращения возникновения тупиковых ситуаций существуют специальные методы. Наиболее распространенным является метод скользящего окна [11].

4.6.2.1. Метод скользящего окна

Окно – это неподтвержденные пакеты, находящиеся в виртуальном канале.

Рассмотрим виртуальный канал (ВК), проходящий на пути от источника к получателю по сети с коммутацией пакетов M узлов с накоплением и воспроизведением информации. Он представлен в виде разомкнутой сети массового обслуживания (см. рисунок 4.42).

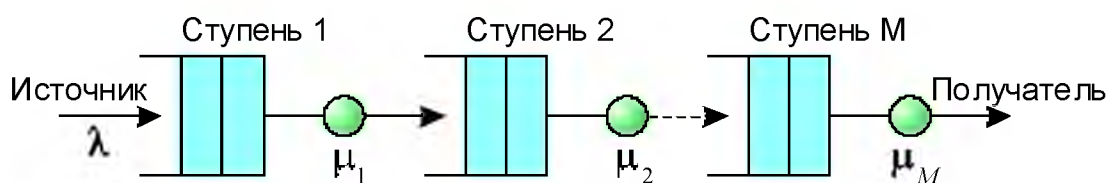


Рисунок 4.42-Модель массового обслуживания для виртуального канала

Если предположить, что каждая система независима, то можно применить аппарат теории массового обслуживания.

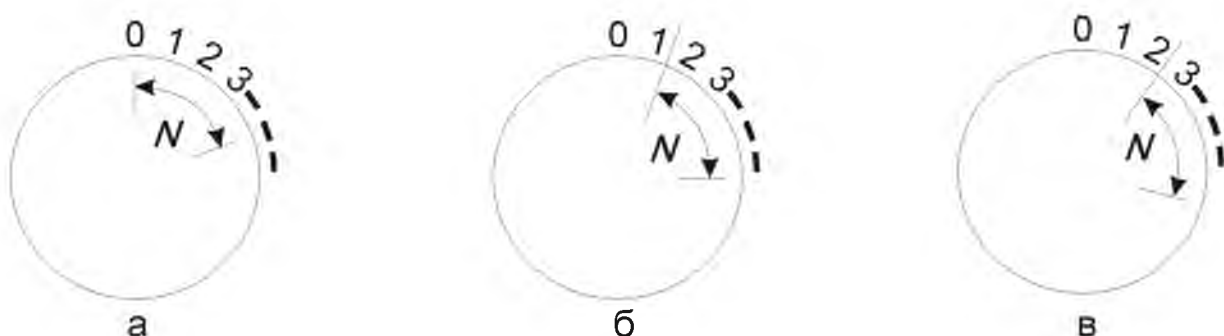


Рисунок 4.43-Управление с помощью скользящего окна: (а) начальное состояние, (б) пакет 1 подтвержден, (в) следующий пакет (2) подтвержден

Разомкнутая сеть может быть отнесена к составным системам массового обслуживания. Она имеет показатели, определенные в форме произведения (то есть вероятности состояний всей сети выражаются произведениями вероятностей состояний каждой системы обслуживания). Наложим на эту модель скользящее окно (рисунок 4.44).

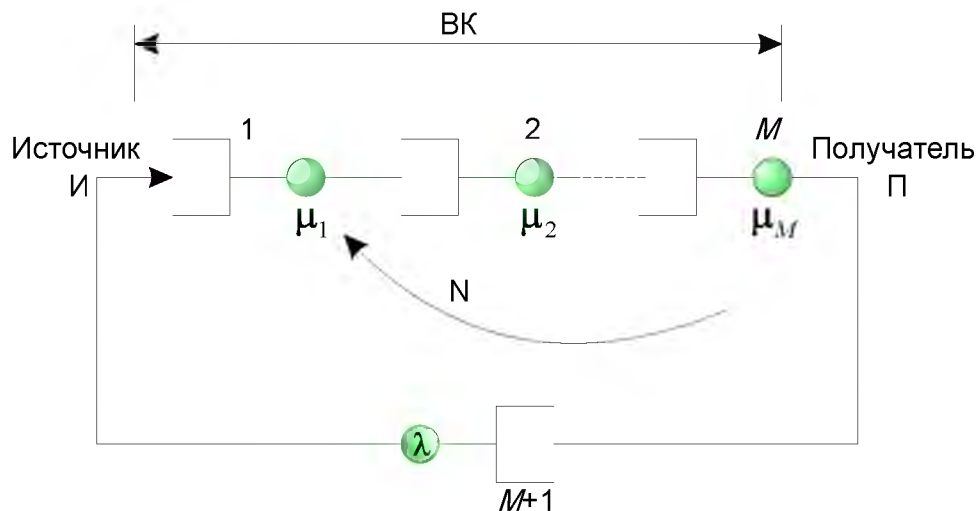


Рисунок 4.44-Модель управления с помощью скользящего окна

В данном случае – частный случай замкнутой сети массового обслуживания. Здесь источник и получатель связаны дополнительной искусственной системой обслуживания, обозначаемой $M+1$, интенсивность обслуживания которой равна λ , что соответствует интенсивности входящего потока в ВК. По замкнутой системе циркулирует фиксированное число N пакетов. Отметим теперь, как эта модель замкнутой системы охватывает механизм скользящего окна. Если в ВК (верхние M систем обслуживания) находятся N пакетов, нижняя система обслуживания $M+1$ пуста и обслуживать не может. Это моделирует состояние блокировки, которое пакеты встречают при поступлении в момент исчерпанного окна. В момент, когда один из N пакетов в ВК поступает к получателю, он появляется в системе обслуживания $M+1$ и источник теперь может доставить пакеты с пуассоновской интенсивностью λ .

Это имеет место всегда, если в ВК находится менее N пакетов; тогда остальные накапливаются в искусственной системе обслуживания $M+1$.

Теорема Нортонa: в сети массового обслуживания любой участок сети, содержащий несколько узлов, может быть заменен одним эквивалентным узлом, производительность которого зависит от состояния преобразуемого участка сети.

Теорема устанавливает, что M последовательных систем обслуживания между точками И и П могут быть заменены одной системой обслуживания, зависящей от состояния, без изменения статистических характеристик между точками И и П.

На рисунке 4.45 показана замкнутая сеть массового обслуживания с одной системой обслуживания СО, вынесенной за ее пределы. Предположим, что требуется найти вероятности состояний только этой системы. Теорема Нортонa устанавливает, что если сеть массового обслуживания имеет решение в форме произведения, то отдельная зависящая от состояния система массового обслуживания, входящая в эту сеть, может быть выделена путем короткого замыкания точек А и В, как показано на рисунке 4.45б, позволяя n пакетам циркулировать по вновь образованной замкнутой сети. Значительно более простой эквивалентной моделью для решения этой задачи является модель, показанная на рисунке 4.45в. В ней эквивалентная система обслуживания Нортонa с интенсивностью обслуживания $\mu(n)$, где n – состояние СО, заменяет всю сеть между точками А и В на рисунке 4.45а.

Эквивалентная замена для рассматриваемой сети приводит к зависящей от состояния характеристике обслуживания

$$u(n) = \frac{n \cdot \mu}{n + (M - 1)}. \quad (4.17)$$

Производительность $u(n)$ в случае, когда среди M показанных СО распределены n пакетов, всегда меньше или равна μ . Она должна удовлетворять условию

$$u(n) = \mu \cdot p_n \quad \{ \text{система не пуста} \}. \quad (4.18)$$

Производительность γ ВК, управляемого с помощью окна, получается усреднением интенсивностей обслуживания по всем N :

$$\gamma = \sum_{n=1}^N u(n) p_n \quad \gamma = \sum_{n=1}^N u(n) \cdot p_n. \quad (4.19)$$

На основании формулы Литтла задержка $E(T)$ ВК из конца в конец равна отношению среднего числа пакетов в ВК к γ :

$$E(T) = \sum_{n=1}^N n \cdot p_n / \gamma. \quad (4.20)$$

Чтобы увидеть, как хорошо работает управление с помощью окна и определить значение N , предлагаемое для соответствующей характеристики управления, рассмотрим область очень большой нагрузки. Предположим, что $\lambda \rightarrow \infty$. Тогда очевидно, что как только пакет поступит к получателю и к источнику вернется подтверждение, в ВК немедленно вводится другой пакет. Таким образом, в этом предельном случае ВК всегда находится в состоянии N , $E(n)=N$ и производительность γ равна

$$\gamma = u(n) = \frac{N \cdot \mu}{N + (M - 1)}, \quad \lambda \rightarrow \infty. \quad (4.21)$$

На основании формулы Литтла непосредственно получим

$$E(T) = \frac{N}{\gamma} = \frac{M - 1 + N}{\mu}, \quad \lambda \rightarrow \infty. \quad (4.22)$$

Из этого равенства видно, что минимальная задержка $E(T) = \frac{M}{\mu}$ имеет место при $N=1$. Это соответствует ожиданию.

При увеличении размера окна N задержка растет пропорционально, а производительность сначала линейно растет с N , но затем выравнивается, приближаясь к максимальному значению μ все более и более медленно с ростом N . Объединив равенства (4.21) и (4.22), получим простое выражение для характеристики обмена между задержкой и производительностью

$$\mu \cdot E(T) = (M - 1) / (1 - \frac{\gamma}{\mu}), \quad \lambda \rightarrow \infty. \quad (4.23)$$

Равенство справедливо только для целых значений N .

Какое значение N должно применяться? Из (4.21) и (4.22) видно, что оптимальное $N = M-1$. Увеличение N за пределы этой точки приводит к устойчивому росту задержки.

4.6.3. Выбор кратчайших путей

4.6.3.1. Алгоритм Дейкстры

Рассмотрим сеть на рисунке 4.46. Числа, проставленные у каждой линии, указывают ее стоимость. Целью является нахождение кратчайшего пути от узла 1, являющегося источником, ко всем остальным узлам сети.

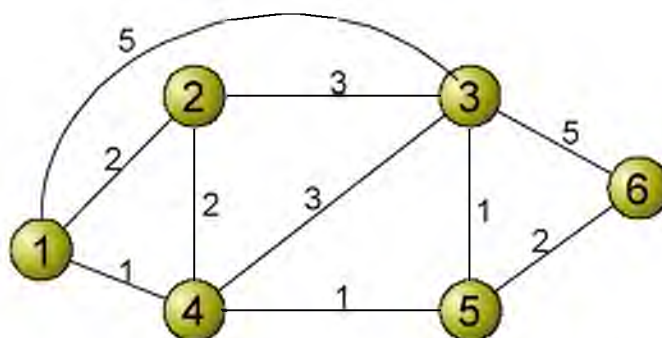


Рисунок 4.46-Пример сети

Обозначим через $D(v)$ расстояние от источника 1 до узла v . Пусть $L(i, j)$ - заданная стоимость пути между узлами i и j . Алгоритм состоит из двух частей: начального шага и итераций, повторяющихся до завершения алгоритма. В множество N будем заносить кратчайшие пути к узлам.

1. Начальный шаг

Устанавливаем $N = \{1\}$. Для каждого узла v , не принадлежащего множеству N , устанавливаем $D(v) = L(1, v)$. (Расстояние до узлов, не соединенных с узлом 1, принимаем равным бесконечности).

2. Каждый последующий шаг.

Находим не принадлежащий множеству N узел w , для которого $D(w)$ минимально и включаем w в множество N . Затем обновляем значения $D(v)$ для всех остальных узлов, не принадлежащих N путем вычисления

$$D(v) \leftarrow \min[D(v), D(w) + L(w, v)] /$$

(4.24)

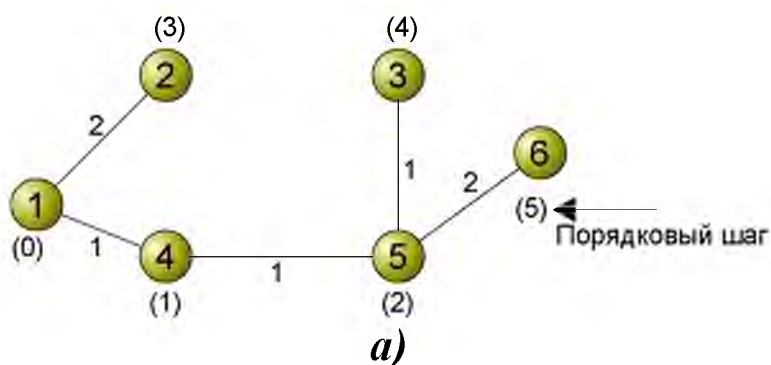
Шаг 2 повторяется, пока в множество N не войдут все узлы.

Применение этого алгоритма к сети иллюстрируется последовательными шагами, указанными в таблице 4.3. Полужирным курсивом в столбцах обозначены минимальные значения $D(w)$ на каждом шаге (при равных расстояниях выбор производится случайным образом). После каждого шага соответствующий узел w добавляется к N . Затем величины $D(v)$ обновляются. Например, после начального шага во время шага 1 к множеству N добавляется узел 4, имеющий минимальное значение $D(4)=1$. На шаге 2 в N включается узел 5 при $D(5)=2$, и т.д. После шага 5 все узлы оказываются включенными в множество N и алгоритм завершается.

Таблица 4.3 Применение алгоритма Дейкстры к сети рисунка 4.45

Шаг	N	$D(2)$	$D(3)$	$D(4)$	$D(5)$	$D(6)$
Началь- ный	{1}	2	5	1	∞	∞
1	{1,4}	2	4	1	2	∞
2	{1,4,5}	2	3	1	2	4
3	{1,2,4,5}	2	3	1	2	4
4	{1,2,3,4,5}	2	3	1	2	4
5	{1,2,3,4,5, 6}	2	3	1	2	4

По мере работы алгоритма, приводящей к результатам, показанным в таблице, в то же самое время строится дерево кратчайших путей с корнем в узле 1: при включении узла во множество N он соединяется с соответствующим узлом, уже принадлежащим N. Результирующее дерево для сети показано на рисунке 4.46. Цифры в скобках у каждого узла указывают шаг, на котором этот узел был включен в дерево. С помощью дерева кратчайших путей для узла 1 можно получить таблицу маршрутов для узла 1, показывающую исходящий путь, по которому нужно направлять пакеты к узлу назначения. Подобная таблица маршрутов может быть составлена для каждого узла, являющегося источником сообщений.



Получатель	Следующий шаг
2	2
3	4
4	4
5	4
6	4

б)

Рисунок 4.47-Применение алгоритма Дейкстры к рисунку 4.45.

Источник - узел 1: а) - построение дерева, б) - таблица маршрутов.

4.6.3.2.Алгоритм Флойда

Он также состоит из двух частей: начального шага и расчетов кратчайших расстояний, которые повторяются до завершения алгоритма. Здесь кратчайшее расстояние представляет собой расстояние до данного узла от узла 1, например, рассматриваемого как узел *назначения*. Алгоритм заканчивается, когда для всех узлов фиксируется их расстояние до узла 1 (узла получателя) и отмечаются следующие узлы по направлению к узлу назначения по кратчайшему пути. Построение таблицы маршрутов требует повторного или параллельного применения этого алгоритма для каждого узла назначения, что дает в результате *набор меток* для каждого узла, причем каждая метка дает информацию о маршруте и расстоянии до конкретного узла назначения.

Рассмотрим опять рисунок 4.45. Обращаясь ко второй части алгоритма и применяя ее в циклическом порядке для узлов 2-6, находим, что алгоритм завершается после двух циклов. Эти циклы и полученные в результате метки каждого цикла, приводятся в таблице 4.4 (любой другой порядок обхода узлов приводит к тому же самому окончательному результату).

Таблица 4.4

Применение алгоритма Флойда к сети рисунка 4.45

Цикл	Метки, узел→	2	3	4	5	6
Начальный		(•, ∞)	(•, ∞)	(•, ∞)	(•, ∞)	(•, ∞)
1		(1,2)	(1,5)	(1,1)	(4,2)	(5,4)
2		(1,2)	(5,3)	(1,1)	(4,2)	(5,4)

Таким образом, в цикле 1 нужно отметить, что узел 2 является «ближайшим» к его соседу 1. Его результирующая новая стоимость равна $D(v)=D(1)+L(1,2)=2$. Следовательно, метка, как это и указано, имеет вид (1,2). Переходя к узлу 3, приходим к необходимости выбора стоимостей между $D(2)+L(2,3)=5$ или $D(1)+L(1,3)=5$. Выбирая произвольно $D(1)+L(1,3)$, получим (1,5), что показано в таблице (при другом выборе получится то же самое). Аналогично получаются другие результаты. В цикле 2 узел 3 имеет теперь пять соседних узлов с конечными значениями $D(w)$, между которыми нужно произвести выбор. Минимальное значение $D(w)+L(w,3)$ равно $D(5)+L(5,3)$, и метка узла 3, как показано, меняется на (5,3). Другие узлы не меняют своих меток, и алгоритм завершается. Опять же дерево кратчайших путей с корнем в узле 1, как показано на рисунке 2а, может быть получено путем обхода меток каждого узла. В результате, узел 2 соединяется с узлом 1, узел 3 - с узлом 5, узел 4 - с узлом 1, узел 5 - с узлом 4 и узел 6 - с узлом 5. Иначе, таблица маршрутов или значение следующего узла в каждом узле по направлению к узлу 1 в точности является первым числом каждой двузначной метки, что и указывалось выше. Для получения полной таблицы маршрутов в каждом узле алгоритм должен быть повторен для каждого узла, принимаемого в качестве узла назначения.

4.7. Глобальная сеть INTERNET

В настоящее время существует два созвучных термина – internet и Internet. Под internet понимают технологию обмена данными, основанную на использовании протоколов TCP/IP, а под Internet – глобальное сообщество мировых сетей, которые используют internet для обмена данными. Internet начинался, аналогично большинству современных технологий, как военная программа, направленная на повышение устойчивости системы обороны США.

4.7.1. Появление и развитие Internet

2 января 1969 г. управление перспективных исследований (ARPA), являющееся одним из подразделений министерства обороны США, начало работу над проектом связи компьютеров оборонных организаций. В результате была создана сеть ARPANET, в основе функционирования которой лежали принципы, на которых позже был построен Internet. ARPANET должна была обеспечить сохранение коммуникаций в случае ядерной атаки противника, с другой стороны – облегчить сотрудничество различных исследовательских учреждений. ARPANET обеспечивала связь между университетами, военными учреждениями и предприятиями оборонной промышленности. В случае разрушения одной или нескольких линий связи, система должна была уметь переключаться на другие линии. Спустя некоторое время в систему были встроены программы перемещения файлов и электронная почта.

Следующим этапом в развитии Internet было создание сети научного фонда США (NSF). Сеть NSFNET объединяла научные центры США. Основой сети стали 5 суперкомпьютеров, соединенных между собой высокоскоростными линиями связи. Все остальные пользователи могли подключаться к сети и использовать возможности этих суперкомпьютеров.

В 1987 г. был создан хребет сети NSFNET, состоящий из 13 центров, соединенных высокоскоростными линиями связи. Центры располагались в разных частях США. Сеть NSFNET быстро заняла место ARPANET и последняя была ликвидирована в 1990 г. Таким образом, появилась сеть Internet в США.

Одновременно были созданы национальные сети в др. странах. Они стали объединяться и в 90 – х годах появился Internet в его нынешнем виде. Сейчас Internet объединяет тысячи разных сетей, расположенных по всему миру. К Internet`у имеют доступ десятки миллионов пользователей.

В России Internet появился недавно (сначала электронная почта). Бурный рост пользователей в России начался в 1996 году.

Internet скоро станет основным средством связи. Не только компьютеры, но телефоны, телевизоры, видеокамеры и другие устройства будут подключаться напрямую к Internet`у. Умение работать в Internet`е станет обязательным условием для достижения успехов практически в любой области деятельности.

4.7.2. Структура Internet

Отличительной особенностью Internet является высокая надежность. При выходе из строя части компьютеров и линий связи сеть будет продолжать функционировать. Такая надежность обеспечивается тем, что в Internet`е **нет единого центра управления**. Если выходят из строя некоторые линии связи и компьютеры, то сообщения м.б. переданы по другим линиям связи. Как и любая другая компьютерная сеть, Internet состоит из множества компьютеров, соединенных между собой линиями связи, и установленных на этих компьютерах программ. Internet обеспечивает обмен информацией между всеми компьютерами, которые входят в сети подключенные к ней. Тип компьютера и используемая ими операционная система значения не имеют.

Основные ячейки Internet – локальные вычислительные сети (ЛВС). Если ЛВС подключена к Internet, то и каждая рабочая станция этой сети также может подключиться к Internet. Существуют также компьютеры самостоятельно подключенные к Internet (host- хозяина) хост - компьютеры.

«Центральная жила» Internet – оптоволоконный кабель с очень высокой пропускающей способностью. Информацию можно переносить и с помощью спутниковых систем связи. Спутники позволяют передавать информацию между континентами через космическое пространство.

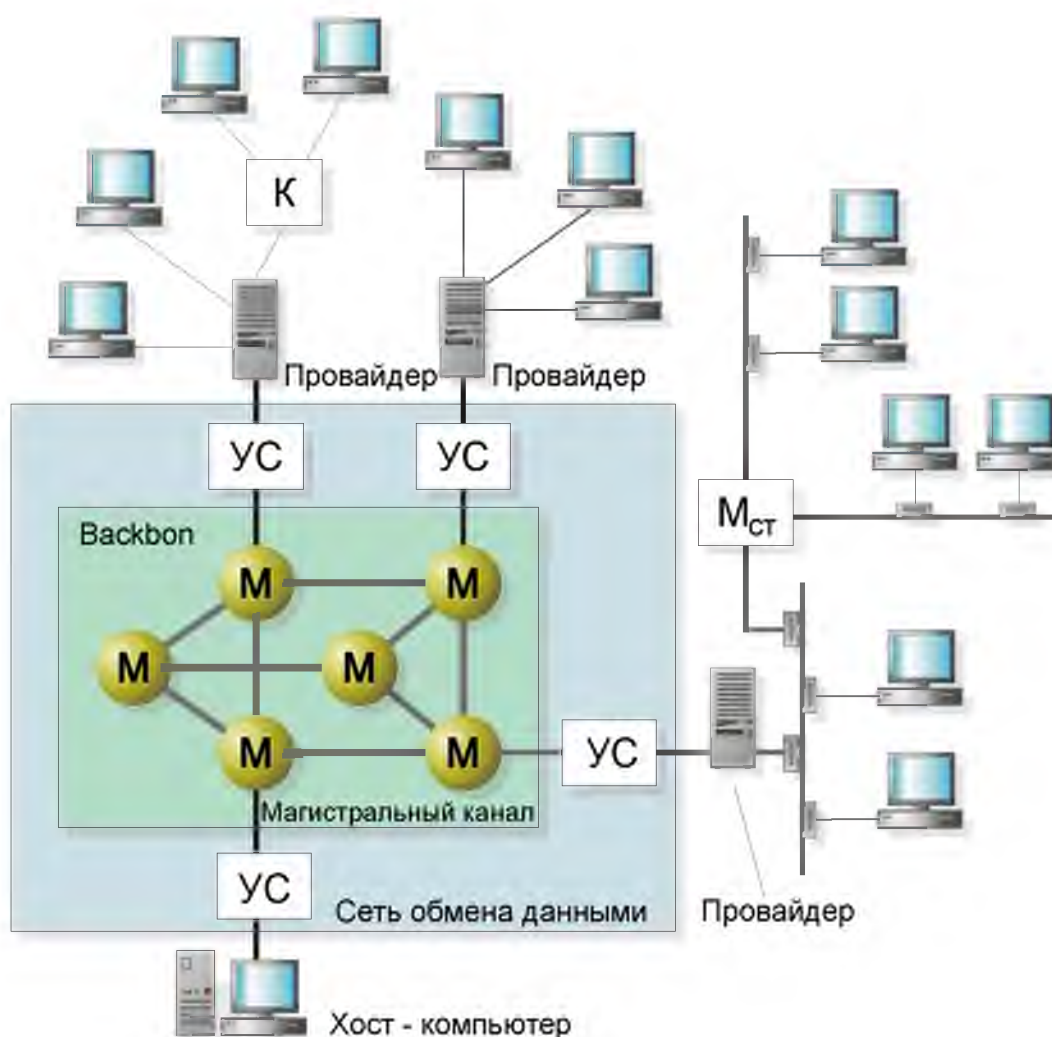


Рисунок 4.48-Схема соединения компьютеров в Internet

Internet представляет собой совокупность физически взаимосвязанных хост - компьютеров. Каждый подключен-

ный к сети компьютер имеет свой адрес, по которому его может найти абонент из любой точки мира.

Пользователи Interneta подключаются к сети через компьютеры специальных организаций, которые называются поставщиками услуг Interneta (провайдерами – provider). Провайдеры имеют множество линий для подключений пользователей и высокоскоростные линии связи для подключения к остальной части Internet. Мелкие поставщики подключены к более крупным и т.д. Все организации, соединенные между собой высокоскоростными линиями связи, обрабатывают базовую часть сети или хребет (Бэкбон - Backbon) Interneta. Если поставщик подключен непосредственно к хребту, то скорость передачи информации будет максимальной.

Однако и одиночный пользователь и ЛВС могут подключаться высокоскоростной линией к хребту Interneta и стать провайдерами.

Компьютеры, подключенные к Internetу, часто называются узлами Interneta или сайтами (Site – место). Узлы, установленные у провайдеров, обеспечивают доступ пользователей к Internetу.

Многие фирмы создают узлы в Internetе, с помощью которых они распространяют информацию о своих товарах и услугах (Web – узлы).

Подключение к Internetу с помощью провайдера означает, что вы с помощью своего модема устанавливаете соединение с компьютером поставщика, который связывает вас с Internetом. В настоящее время используется 4 различных варианта подключения к Internetу.

- Постоянное подключение (24 часа в сутки), ЛВС подсоединяются с помощью выделенной линии связи, которая обеспечивает высокую скорость передачи информации. Используется средними и крупными фирмами. Дорогой вариант.

- Работа с помощью электронной почты. Дешевый метод.

■ Коммутируемое соединение с помощью эмуляции терминала. (Ваш ПК – удаленный терминал поставщика – использует систему поставщика, сейчас этот способ используют в основном профессионалы, чтобы добиться некоторых нестандартных результатов – а раньше пользовались многие).

■ Коммутируемое IP – соединение. Через обычную телефонную линию ваш модем связывается с модемом провайдера. Сеансовое соединение так как во время сеанса - вы полноправный пользователь Interneta, но по окончании сеанса связь с Internetом разрывается.

4.7.3. Передача информации в Internet

В Internet используют 2 основных понятия **адрес** и **протокол**.

Свой уникальный **адрес** имеет любой компьютер, подключенный к Internetу. Даже при временном соединении по коммутируемому каналу компьютеру выделяется уникальный адрес. Адрес 1) должен иметь формат позволяющий вести его обработку автоматически и 2) должен нести некоторую информацию о своем владельце.

С этой целью для каждого компьютера устанавливается 2 адреса: цифровой IP – адрес (Internetwork Protocol – межсетевой протокол) и доменный адрес.

Цифровой адрес удобен для обработки на компьютере, а доменный адрес – для восприятия пользователем.

Цифровой адрес имеет длину 32 бита. Для удобства он разделен на 4 блока по 8 бит, которые можно записать в десятичном виде.

Адрес сети (192.45); адрес подсети (9); адрес компьютера – (150).

192.45.9.150

Доменная адресация. Числовая адресация удобна для машинной обработки таблиц маршрутов, но совершенно неприемлема для использования ее человеком. Запомнить набо-

ры цифр гораздо труднее, чем мнемонические осмысленные имена. Для облегчения взаимодействия в Сети сначала стали использовать таблицы соответствия числовых адресов именам машин. Эти таблицы сохранились до сих пор и используются многими прикладными программами. Это файлы с именем `hosts`. Если речь идет о системе типа Unix, то этот файл расположен в директории `/etc` и имеет следующий вид:

	имя машины	синонимы
127.0.0.1	localhost	localhost
144.206.160.32	Polyn	Polyn
144.206.160.40	Apollo	www

Последний столбец в этой таблице является необязательным. Пользователь для обращения к машине может использовать как IP-адрес машины, так и ее имя или синоним (*alias*). Обращения, приведенные ниже, приводят к одному и тому же результату – инициированию сеанса telnet с машиной Apollo:

```
telnet 144.206.160.40
```

или

```
telnet Apollo
```

или

```
telnet www
```

Однако такой способ присвоения символьных имен был хорош до тех пор, пока Internet был маленьким. По мере роста сети стало затруднительным держать большие списки имен на каждом компьютере. Для того чтобы решить эту проблему, были придуманы DNS (Domain Name System).

Любая DNS является прикладным процессом, который работает над стеком TCP/IP. Таким образом, базовым элементом адресации является IP-адрес, а доменная адресация выполняет роль сервиса.

Система доменных адресов строится по иерархическому принципу. Однако иерархия эта не строгая. Фактически нет единого корня всех доменов. В 80-е годы были определены первые домены верхнего уровня: gov, mil, edu, com, net. Позднее, когда сеть перешагнула национальные границы США, появились национальные домены типа uk, jp, au, ch и т.п. Для СССР также был выделен домен (ru). После 1991 года, когда республики союза стали суверенными, многие из них получили свои собственные домены. Однако домен СССР остался, ибо просто так выбросить домен из сервера имен нельзя, на основе доменных имен строятся адреса электронной почты и доступ ко многим другим информационным ресурсам Internet. Поэтому гораздо проще оказалось ввести новый домен к существующему, чем заменить его. Таким образом, в Москве существуют организации с доменными именами, оканчивающимися на su (например kiae.su) и оканчивающимися. Вслед за доменами верхнего уровня следуют домены, определяющие либо регионы, либо организации. Далее идут следующие уровни иерархии, которые могут быть закреплены либо за небольшими организациями, либо за подразделениями больших организаций.

Всю систему доменной организации можно представить следующим образом:



Рисунок 4.49-Система доменной адресации

Для пользователей Internet адресами могут быть их регистрационные номера на компьютере, подключенном к сети. За именем следует знак @, например:

BaranovskayaTatyana@agrU.Krasnodar.ru

Наиболее популярной программой поддержки DNS является `named`, которая реализует Berkeley Internet Name Domain (BIND).

Сетевой **протокол** предписывает правила работы компьютерам, которые подключены к сети. Стандартные протоколы заставляют разные компьютеры говорить на одном языке. Таким образом осуществляется возможность подключения к Internetу разнотипных компьютеров, работающих под управлением различных операционных систем.

На нижних (2-м и 3-м) уровнях используются 2 основных протокола:

IP – (протокол Interneta) и TCP – (протокол управления передачей).

Так как эти 2 протокола тесно взаимосвязаны, то часто их объединяют и говорят, что в Internetе базовым протоколом является TCP/IP. Все остальные протоколы строятся на их основе.

Конечными пользователями глобальной сети являются **host** (хозяин) - **компьютеры** (или устройства), имеющие 32 битный адрес, разбитый на 4 байта и представленный в десятичном формате (256.256.256.256), так как в двоичном виде он плохо воспринимается людьми.

Именно на их основе и функционирует Internet.

Протокол TCP разбивает информацию на порции, нумерует все порции, чтобы при получении можно было правильно собрать информацию. Каждый пакет получает заголовок TCP, где кроме адреса получателя содержится информация об исправлении ошибок и о последовательности передачи пакетов.

Затем пакеты TCP разделяются на еще более мелкие пакеты IP.

Пакеты состоят из 3 –х различных уровней, каждый из которых содержит:

- 1) Данные приложения
- 2) Информацию TCP

3) Информацию IP

Перед отправкой пакета протокол TCP вычисляет контрольную сумму. При поступлении снова рассчитывается контрольная сумма, если пакет поврежден, то запрашивается повторная передача.

Затем принимающая программа объединяет пакеты IP в пакеты TCP, из которых реконструируются исходные данные.

Протоколы TCP/IP обеспечивают передачу информации между компьютерами. Все остальные протоколы с их помощью реализуют самые разные услуги Interneta.

4.7.4. Краткая характеристика ресурсов Internet

Информационные ресурсы Internet – это вся совокупность информационных технологий и баз данных, которые доступны при помощи этих технологий. К их числу относятся, например:

- электронная почта;
- система телеконференций Usenet;
- система файловых архивов FTP;
- информационная сеть WWW;
- информационная система Gopher;
- информационная система WAIS;
- информационные ресурсы LISTSERV;
- справочные книги X.500;
- справочная служба WHOIS;
- информационные ресурсы Mailbase и TRICKLE

Главный режим доступа к информационным ресурсам Internet – это on-line. Даже серверы электронной почты обмениваются информацией друг с другом в интерактивном режиме по протоколу SMTP.

В отечественных условиях, несмотря на бурное развитие телекоммуникаций, основным средством доступа к Internet является электронная почта. К счастью, многие информационные ресурсы Сети имеют программы-роботы, которые спо-

собны общаться с почтовыми клиентами по схеме, представленной ниже.

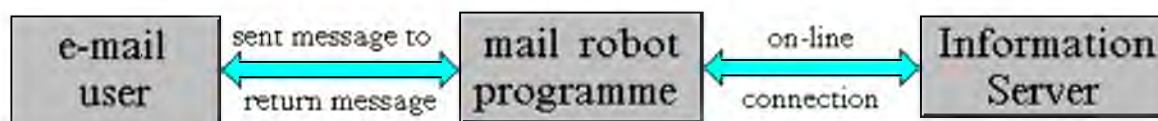


Рисунок 4.50- Схема взаимодействия с почтовыми клиентами через почтовые робот-программы

Электронная почта – один из важнейших информационных ресурсов Internet. Она является самым массовым средством электронных коммуникаций. Любой из пользователей Internet имеет свой почтовый ящик в сети. Если учесть, что через Internet можно принять или послать сообщения еще в два десятка международных компьютерных сетей, некоторые из которых не имеют on-line сервиса вовсе, то становится понятным, что почта предоставляет возможности в некотором смысле даже более широкие, чем просто информационный сервис Internet.

Электронная почта во многом похожа на обычную почтовую службу. Корреспонденция подготавливается пользователем на своем рабочем месте либо программой подготовки почты, либо просто обычным текстовым редактором. Обычно программа подготовки почты вызывает текстовый редактор, который пользователь предпочитает всем остальным программам этого типа. Затем пользователь должен вызвать программу отправки почты (программа подготовки почты вызывает программу отправки автоматически). Стандартной программой отправки является программа sendmail. Sendmail работает как почтовый курьер, который доставляет обычную почту в отделение связи для дальнейшей рассылки. В Unix-системах sendmail сама является отделением связи. Она сортирует почту и рассылает ее адресатам. Для пользователей персональных компьютеров, имеющих почтовые ящики на своих машинах и работающих с почтовыми серверами через

коммутируемые телефонные линии, могут потребоваться дополнительные действия. Так, например, пользователи почтовой службы Relcom должны запускать программу UUCP, которая осуществляет доставку почты на почтовый сервер.

Для работы электронной почты в Internet разработан специальный протокол Simple Mail Transfer Protocol (SMTP), который является протоколом прикладного уровня и использует транспортный протокол TCP. Однако совместно с этим протоколом используется и Unix-Unix-CoPy (UUCP) протокол. UUCP хорошо подходит для использования телефонных линий связи. Большинство пользователей электронной почты Relcom реально пользуются для доставки почты на узел именно этим протоколом. Разница между SMTP и UUCP заключается в том, что при использовании первого протокола sendmail пытается найти машину-получателя почты и установить с ней взаимодействие в режиме on-line для того, чтобы передать почту в ее почтовый ящик. В случае использования SMTP почта достигает почтового ящика получателя за считанные минуты и время получения сообщения зависит только от того, как часто получатель просматривает свой почтовый ящик.

USENET – это система телеконференций Internet. Система построена по принципу электронных досок объявлений, когда любой пользователь может поместить свою информацию в одну из групп новостей Usenet, и эта информация станет доступной другим пользователям, которые на данную группу новостей подписаны. Именно этим способом распространяется большинство сообщений Internet, например, списки наиболее часто задаваемых вопросов (FAQ) или реклама программных продуктов. По Usenet можно получить и вирус, если заказывать и распаковывать все подряд, что приходит на ваш почтовый адрес. Usenet – хорошее место для объявления международных конференций и семинаров.

Система файловых архивов FTP – это огромное распределенное хранилище всевозможной информации, накоп-

ленной за последние 10-15 лет в Сети. Любой пользователь может воспользоваться услугами анонимного доступа к этому хранилищу и скопировать интересующие его материалы. Объем программного обеспечения в архивах FTP составляет терабайты информации, и ни один пользователь или администратор сети не может просто физически обозреть эту информацию. Кроме программ в FTP-архивах можно найти стандарты Internet – RFC, пресс-релизы, книги по различным отраслям знаний, главным образом по компьютерной проблематике, и многое другое. Практически любой архив строится как иерархия директорий. Многие архивы дублируют информацию из других архивов (так называемые «зеркала» – mirrors). Для того чтобы получить нужную информацию, вовсе не обязательно ждать, пока информация будет передана из Австралии или Южной Африки, можно поискать «зеркало» где-нибудь ближе, например, в Финляндии или Швеции. Для этой цели существует специальная программа *Archie*, которая позволяет просканировать FTP-архивы и найти тот, который устраивает пользователя по составу программного обеспечения и коммуникационным условиям.

Распределенная гипертекстовая информационная система World Wide Web – это последний хит Internet. Темпы развития этого сервиса столь стремительны, что за последний год число серверов WWW увеличилось в три раза. World Wide Web предоставляет удобный доступ к большинству информационных архивов Internet. Особенностью системы является механизм гипертекстовых ссылок, который позволяет просматривать материалы в порядке выбора этих ссылок пользователем. Многие интерфейсы данной технологии позволяют выбирать интересующий материал простым нажатием кнопки «мыши» на нужном слове или поле графической картинке. Система универсальных адресов позволяет проадресовать практически все информационные ресурсы Internet. Многие издательства взяли WWW на вооружение для электронных версий своих журналов. В WWW существует большое

количество различного рода каталогов, которые позволяют ориентироваться в сети, кроме этого, пользователи могут выполнить даже удаленные программы или смотреть фильмы по сети. Такой сервис не обеспечивается другими информационными системами Internet.

GOPHER – это еще одна распределенная информационная система Internet. В основу ее интерфейсов положена идея иерархических каталогов. Внешне Gopher выглядит как огромная файловая система, которая расположена на машинах сети. Первоначально Gopher задумывался как информационная система университета с информационными ресурсами факультетов, кафедр, общежитий и т.п. До сих пор основные информационные ресурсы Gopher сосредоточены в университетах. Gopher считается простой системой, легкой в установке и администрировании, достаточно надежной и защищенной. Количество серверов Gopher на 1994 год превышало число серверов WWW в 1,5 раза, и до 1995 года темпы роста установок серверов Gopher опережали все остальные ресурсы Сети. В России Gopher-серверы не так распространены, как во всем мире – профессионалам больше нравится World Wide Web.

WAIS – это распределенная информационно-поисковая система Internet. Родилась WAIS как перспективная разработка четырех ведущих американских компаний и первое время была коммерческим продуктом, пока не появилась свободно распространяемая версия *freeWAIS*. В основу системы положен принцип поиска информации с использованием логических запросов, основанных на применении ключевых слов. Клиент «обшаривает» все серверы WAIS на предмет наличия на них документов, удовлетворяющих запросу. WAIS широко применяется как поисковая машина в других информационных сервисах Internet, например в WWW и Gopher. Наиболее известным проектом, где была применена WAIS, является электронная версия энциклопедии «Британика».

LISTSERV – это, строго говоря, не сервис Internet, а система почтовых списков BITNET. Однако, это очень популярный ресурс в глобальных компьютерных сетях, и в Internet существуют шлюзы для доступа к нему. **LISTSERV** специально ориентирован на применение в качестве транспорта электронной почты. Доступ к нему в интерактивном режиме затруднен. В мире насчитывается многие сотни списков **LISTSERV**, которые организованы по группам интересов, например, существуют группы разработчиков программ ядерно-физических расчетов EGS-4 или группы любителей научной фантастики. **LISTSERV** сильно пересекается с Usenet, однако это не мешает существованию как одной, так и другой системы.

X.500 – является европейским стандартом для компьютерных справочных служб. Базы данных **X.500** содержат информацию о пользователях сети, их электронные и обычные адреса, идентификаторы и реальные имена, должности и места службы. **X.500** хранит информацию не только о физических лицах, но и об организациях. В последнем случае дается краткое описание основных направлений их деятельности.

WHOIS – служба, аналогичная по назначению системе **X.500**, но являющаяся детищем Internet. Работа с системой **WHOIS** несколько отличается от работы с **X.500** в силу ее организации. **WHOIS** распределенная система – это значит, что запросы отправляются по всему множеству серверов **WHOIS** в Internet, если только не указан адрес конкретного сервера.

MAILBASE – это система, во многом повторяющая описанный выше **LISTSERV**.

TRICKLE – это доступ по почте к архивам RTF, который организован через специальный шлюз. Этот шлюз имеет специальные навигационные средства для поиска нужной информации в Сети, пользователь может вести с ним своеобразный диалог по почте, выбирая нужную информацию путем ввода специальных команд **TRICKLE**.

Вообще говоря, существуют и другие ресурсы, к которым можно получить доступ по почте.

4.7.5. Удаленный доступ к ресурсам сети

Telnet – это одна из самых старых информационных технологий Internet. Она входит в число стандартов, которых насчитывается три десятка на полторы тысячи рекомендуемых официальных материалов сети, называемых RFC (Request For Comments).

Под telnet понимают триаду, состоящую из:

- telnet-интерфейса пользователя;
- telnet-процесса;
- telnet-протокола.

Эта триада обеспечивает описание и реализацию сетевого терминала для доступа к ресурсам удаленного компьютера.

В настоящее время существует достаточно большое количество программ – от Kermit до различного рода BBS (Bel-luten Board System), которые позволяют работать в режиме удаленного терминала, но ни одна из них не может сравниться с telnet по степени проработанности деталей и концепции реализации.

Telnet как протокол описан в RFC-854 (май, 1983 год). Его авторы J.Postel и J.Reynolds во введении к документу определили назначение telnet так:

«Назначение TELNET - протокола – дать общее описание, насколько это только возможно, двунаправленного, восьмибитового взаимодействия, главной целью которого является обеспечение стандартного метода взаимодействия терминального устройства и терминал -ориентированного процесса. При этом этот протокол может быть использован и для организации взаимодействий «терминал- терминал» (связь) и «процесс- процесс» (распределенные вычисления).»

Telnet строится как протокол приложения над транспортным протоколом TCP. В основу telnet положены три фундаментальные идеи:

- концепция сетевого виртуального терминала (Network Virtual Terminal) или NTV;
- принцип договорных опций (согласование параметров взаимодействия);
- симметрия связи «терминал- процесс».

При установке telnet - соединения программа, работающая с реальным терминальным устройством, и процесс обслуживания этой программы используют для обмена информацией спецификацию представления правил функционирования терминального устройства, или Сетевой Виртуальный Терминал (Network Virtual Terminal). Для краткости эту спецификацию обозначают NVT. NVT – это стандартное описание наиболее широко используемых возможностей реальных физических терминальных устройств. NTV позволяет описать и преобразовать в стандартную форму способы отображения и ввода информации.

4.7.6. Коммерческое применение Internet

До недавнего времени Сеть рассматривалась исключительно как глобальное средство информации. В последние 2—3 года возможности Internet, связанные прежде всего с появлением сервиса WWW, создали благоприятную почву для ведения бизнеса в Сети. Реальность такова, что любая компания или бизнесмен могут рассматривать Сеть в качестве средства для реализации коммерческих целей. Организация виртуального (электронного) магазина для проведения коммерции через Internet становится все более насущной потребностью для большинства фирм.

Internet открывает доселе никому неизвестные возможности ведения бизнеса. Любой деловой человек должен понять, что только тот, кто может наиболее полно и качественно ис-

пользовать потенциал Internet, имеет шанс значительного улучшения своих конкурентных позиций.

Уже сейчас Сеть активно используется многими компаниями как оперативное средство связи. Причем речь идет как о коммутации внутри одной корпорации (допустим, между разными службами или отделами), так и об обмене информацией между разными фирмами, связанными партнерскими отношениями, выступая в роли заказчика и поставщика. Подобное применение Internet, чаще всего в этом случае речь идет об использовании электронной почты, позволяет оптимизировать информационные потоки и непосредственно ускорить и сделать более качественным процесс ведения самого бизнеса. Другая активно применяемая модель бизнеса в Сети связана с использованием Internet как средства массовой информации для распространения сведений как о самой фирме, так и о ее продукции и услугах, или, проще говоря, для рекламы, а также в качестве инструмента маркетингового исследования.

Однако наиболее перспективным вариантом ведения сетевого бизнеса следует считать электронную коммерцию, которая подразумевает создание **виртуального магазина**, позволяющего организовать торговлю своей продукцией в Сети.

Количество компаний в мире, занимающихся электронной коммерцией, в 1996 году составило 111 тысяч и, по прогнозам, достигнет к 2000 году 435 тысяч (данные Voipe, Wetty&Co.), при этом суммарный оборот Internet возрастет с 9,5 млрд. дол. до 196 млрд. дол. (данные Forrester Research, Inc.). Розничные продажи через Internet вырастут с 500 млн дол. в 1996 году до 7 млрд дол.. Еще более внушительны цифры, касающиеся сектора расчета между компаниями: соответственно 600 млн дол. и 66 млрд дол. Подобные внушительные темпы развития не могут не затронуть российской части Internet, которая также активно внедряется в электронную коммерцию. О перспективах введения последней говорит хотя бы тот факт, что 25 процентов пользователей российской части

Сети — руководители различного ранга.

Виртуальный магазин по сути дела — реализованное в Сети представительство путем создания Web-сервера. Основная цель организации последнего заключается в продаже товаров и услуг другим пользователям Internet. Заглянувший к вам на сервер посетитель может посредством гипертекстовых ссылок и используя многочисленные мультимедийные возможности получить в полном объеме интересующую его информацию о продукции и в конечном итоге сделать заказ. Виртуальный магазин должен иметь доменный адрес и, как и любой другой Web-сервер, он состоит из целого ряда гипертекстовых страниц, зачастую с мультимедийными элементами.

Варианты реализации виртуального магазина могут быть различны. Говоря об электронной коммерции, в одних случаях можно подразумевать лишь **сопровождение** при помощи Internet **сделок** и поставок: выбор товара, заказ, в некоторых случаях даже и оплата. Однако непосредственно поставка товара производится обычным путем (автомобили, компьютеры, бытовая техника и т.д.). В других случаях применение Сети предполагает **возможность поставок продукта** по Internet. Речь идет о продаже информации. Торговля информационной продукцией в Сети имеет свои особенности и поэтому должна рассматриваться отдельно.

Виртуальные магазины во многом схожи с обыкновенными торговыми центрами, однако при этом имеют ряд неоспоримых преимуществ. Как в любом магазине, здесь обязательно должен присутствовать торговый зал, где покупатель (он же посетитель вашего сервера) может спокойно «походить», щелкая мышкой, с одной страницы на другую и узнать всю интересующую его информацию о продуктах. В случае большого ассортимента товаров имеет смысл разложить их по полкам (отдельным страницам), чтобы пользователям было легче ориентироваться. Пользователь, заинтересовавшись каким-либо продуктом, должен иметь возможность узнать о

последнем все его интересующее — это основной принцип из которого из которого следует исходить при организации виртуального магазина. Некоторые из них организованы таким образом, что покупатель, прежде чем принять окончательное решение о покупке, может осмотреть товар со всех сторон, узнать все возможные параметры. Например, при покупке автомобиля в виртуальном магазине можно даже послушать, как у него работает мотор (не за горами время, когда без особых хлопот можно будет даже уловить запах машинного масла). Таким образом, продавец имеет возможность наилучшим образом описать и продемонстрировать качество товара, а покупатель, в свою очередь, не выходя из дома, получить всю необходимую информацию.

Выбрав в виртуальном магазине товар и узнав его стоимость, покупатель может, перейдя по ссылке на другую страницу, заказать его и получить на него счет. При оплате заказанного товара можно воспользоваться кредитной карточкой. Однако существуют определенного рода опасения, что информация о номерах и персональных кодах кредитных карточек может стать добычей хакеров. Проблема безопасности Сети, тормозящая развитие всего сетевого бизнеса, уже сейчас с успехом решается при помощи применения различных способов и схем шифрования информации, передаваемой по Internet. Вполне вероятно, что когда эта задача будет окончательно решена, то мы станем свидетелями новой волны роста электронной коммерции.

С целью полной реализации идеи виртуального магазина следует также отдельно разместить информацию о вашей фирме, что немаловажно для любого посетителя. Даже если он ничего не купит, он уже будет знать о вас. Имеет смысл посвятить один из разделов сервера партнерам, как существующим, так и потенциальным, где вы смогли бы размещать всю самую необходимую и оперативную информацию для налаживания качественных связей. Создав подобные постоянные информационные потоки, можно рассчитывать на

стабильность бизнеса.

Виртуальная коммерция через Internet при помощи новейших технологий означает для фирмы прежде всего улучшение конкурентных позиций. В любом случае на этом рынке вы можете чувствовать себя, без преувеличения, на равных с крупнейшими мировыми корпорациями, поскольку имеете реальную возможность создать электронный магазин как минимум не хуже, чем у других. Существенным моментом при этом является тот факт, что в сферу вашей деятельности попадет территория всего земного шара. Указанные тенденции определяют высокую значимость этого рынка.

Другим немаловажным фактором, способным существенно и благоприятно воздействовать на ваше конкурентное положение, является оперативность. Ваш магазин, работающий 24 часа в сутки, способен к тому же быстро и адекватно реагировать на запросы пользователей, у которых поиск нужной информации занимает всего несколько секунд. Необходимо также отметить значимость Сети как эффективного маркетингового инструмента. Любой посетитель, даже если он не оказался покупателем, может заполнить предложенную ему анкету. Таким образом можно без особых затрат изучить потенциального покупателя, круг его интересов и в дальнейшем учесть полученные результаты при осуществлении как реального, так и виртуального бизнеса.

И, наконец, одним из наиболее значимых факторов следует считать относительно низкие издержки. Это касается, во-первых, процесса организации самого виртуального магазина или Web - сервера, что в принципе оказывается более дешевым, чем организация простой торговой точки. При этом ваш виртуальный магазин может обслуживать, по сути дела, покупателей со всего земного шара; во-вторых, снижаются затраты на продвижение и торговлю товарами и услугами. Так, взять хотя бы расходы на обслуживающий персонал: для нормального функционирования Web-магазина необходимо существенно меньше работников, поскольку большинство их

функций (прежде всего речь идет об обслуживании клиента) берет на себя непосредственно виртуальный магазин.

Оптимизация контакта с поставщиками и партнерами по бизнесу также в конечном итоге положительно влияет на затраты, а соответственно приводит и к формированию более конкурентной цены. Ну и конечно же реклама. Распространяя информацию о своей компании, продукции и т. д., вы привлекаете все большее число сетевых покупателей, что, помимо всего прочего, позволяет улучшить ваш имидж и отношения с потенциальными покупателями. Все вышеперечисленное и определяет в конечном итоге высокую **эффективность** проведения электронной коммерции при помощи виртуального магазина.

Уже сегодня фактически любая организация может получать прибыль от функционирования виртуального магазина. Здесь может быть продан любой продукт: от канцелярского стэплера до объектов недвижимости. Безусловно, пока американские on-line магазины предлагают более широкий ассортимент, чем отечественные Internet-магазины, однако и у нас их число уже достаточно велико и выбор товаров богат. Многие крупные компьютерные фирмы открыли или открывают свои виртуальные магазины в Сети, по Internet можно купить компакт-диск, видеокассету, встречаются предложения о продаже недвижимости, автомобилей, можно приобрести туристическую путевку. В случае, если вы хотите ознакомиться или посетить какой-либо виртуальный магазин, то можно посоветовать посетить поисковый сервер-рубрикатор Yahoo и заглянуть в раздел Business: Electronic Commerce по адресу: [www. yahoo. com/Business/ Electronic Commerce](http://www.yahoo.com/Business/ElectronicCommerce), где имеется обширный список ссылок на фирмы, занимающиеся электронной коммерцией.

Тысячи компаний ежемесячно впервые открывают двери своих виртуальных магазинов. Поэтому не стоит медлить с началом ведения сетевого бизнеса, а активно использовать имеющиеся возможности. Сейчас во многих регионах нашей

страны в результате информационного вакуума некоторым компаниям становится все труднее находить потребителя. Подобные проблемы можно решить через электронную коммерцию, где не имеет значения, откуда ты, из какой страны, из какого региона. Однако сегодня наиболее ходовым товаром для российской части Internet является все-таки информация.

Торговля информацией в Internet — одна из самых старейших форм коммерции в Сети. 30—40 процентов всей информации посвящено бизнесу и финансам. При рассмотрении вариантов предоставления информационного сервиса по бизнесу и финансам можно выделить несколько вариантов.

Во-первых, следует отметить существование каталогов и справочных систем по ресурсам в Internet. Они созданы специально с целью облегчить работу пользователей в Сети. Интерфейс подобных систем позволяет организовать поиск данных по определенному ключу. Однако предоставление подобного сервиса не следует считать электронной коммерцией, поскольку пользование услугами происходит на бесплатной основе. Среди российских сервисов подобного рода можно отметить российскую поисковую систему Rambler, а также новый проект «Российский Internet» — рубрикатор ресурсов. Такие проекты чаще всего позволяют получать прибыль за счет большого объема рекламы, размещаемого на их страницах, поскольку являются одними из самых посещаемых пользователей серверов.

Другая форма информационного бизнеса связана с массовым приходом различных печатных изданий в Internet. При этом компания-издатель организует Web-сервер, на котором размещает материалы печатного издания либо его электронную версию. Основная цель — увеличение числа читателей издания. Для решения данной проблемы следует применять комбинированный подход. Один из наиболее распространенных вариантов — размещение на сервере дайджеста из информации, опубликованной в печатном издании, который был бы

интересен, но в то же время не являлся полной версией материалов и приглашал ознакомиться с ней в печатном издании. Пользователь должен иметь возможность подписаться на издание, перечислив на счет издательства его стоимость.

Другой вариант заключается в подписке на электронную версию издания. В этом случае после перечисления необходимой суммы денег на счет издательства пользователь получает определенное имя и пароль, которые необходимо вводить для доступа к информации. Однако следует иметь в виду, что зачастую попытки ввести оплату за использование информации приводили к тому, что клиенты переставали пользоваться данной услугой (обращаться к вашему Web-серверу) и переходили к другому поставщику аналогичной информации, которая предоставляется на бесплатной основе. В связи с этим необходимо найти золотую середину: в частности, возможным вариантом является предоставление наиболее свежей и оперативной информации на платной основе, в то время как архив выпусков печатного издания делается доступным для любого пользователя.

Практически все российские информационные агентства имеют свое представительство в Сети: ИТАР-ТАСС, РИА-Новости, Национальная служба новостей и т. д. Многие газеты размещают электронные версии своих изданий в Сети. В последнее время также стали появляться принципиально новые средства массовой информации. Первым среди них стал общедоступный Internet - сервер АКДИ «Экономика и жизнь» (www.akdi.ru), который зарегистрирован в Госкомитете РФ по печати и специализируется на предоставлении информации и консультаций в Сети по экономическим, финансовым, правовым вопросам.

Наиболее фундаментальными электронными изданиями в Сети являются реализованные в технологии WWW аналоги крупных печатных изданий, другими словами, **гипертекстовые книги, энциклопедии**. В качестве примера можно привести реализацию в виде гипертекстовой мультимедийной

энциклопедии одной из старейших энциклопедий мира «Британика». Доступ к ней платный, однако предоставляется возможность недельного бесплатного пользования для ознакомления с работой системы.

Другой вариант информационной коммерции в Сети — **предоставление бизнес - информации**. Это могут быть котировки ценных бумаг, курсы валют, цены на биржах, оперативные новости. В последнее время организуются специальные информационные системы или бизнес – службы. В качестве примера из российской практики можно обратиться к проекту «ГКО-Инвест».

Система позволяет в режиме реального времени передавать инвесторам, пользователям динамику торговой сессии на ММВБ по ГКО и ОФЗ. Плата за пользование подобными услугами зависит от ваших потребностей, допустим, в последнем случае, если вы хотите наблюдать за торгами с задержкой порядка 25 секунд, то придется потратиться на 400 дол. за программное обеспечение и еще на 250 дол. ежемесячной платы.

Создание виртуального магазина не требует особых усилий либо материальных затрат, в то же время электронная коммерция уже сейчас начинает активно внедряться в бизнес. Для того чтобы не остаться на обочине конкурентной гонки, каждой организации и каждому бизнесмену стоит подумать о вариантах организации такой новой торговой точки.

Платежные средства в сетях Internet

Постоянно растущая армия пользователей Internet проводит значительное количество времени, сканируя "киберпространство" сети в поисках информации, нужной им для работы или учебы, либо просто развлекаясь. Соответственно маркетинговый потенциал сети растет с увеличением количества пользователей WWW, с одной стороны, и организаций, заинтересованных в размещении коммерческой рекламы в Internet, с другой.

Начиная со второй половины 1994 г., коммерческая ре-

клама стала составлять заметную долю гипертекстовой информации, доступной в WWW. Однако возможности делового использования глобальных цифровых коммуникаций не ограничиваются размещением рекламы, какие бы изощренные ее формы не применялись. Товар или услуга должны быть представлены потенциальному покупателю, но у него также должна быть возможность их приобретения "не отходя от витрины" (в данном случае — виртуальной). С того же 1994 г. возможность расчетов и платежей с использованием компьютерных сетей публичного доступа перешла из разряда теоретических проблем в разряд практических задач.

В основе всех предлагаемых сегодня систем расчетов и платежей с использованием Internet лежат самые продвинутые криптографические технологии обеспечения конфиденциальности информации и аутентичности коммуникантов.

Средства электронных расчетов в Internet

Предложенные на сегодня системы (их чуть больше десятка) можно разбить на три категории. Это суррогатные расчетные средства, расширения существующих внесчетных расчетных систем, таких, как чеки и пластиковые карточки, и электронная наличность.

Суррогатные расчетные средства

Цифровые купоны и жетоны - суррогатные средства расчетов в сети - предлагаются сегодня несколькими компаниями, из которых наиболее известны First Virtual Holdings и Software Agents (знакомая более по торговой марке NetBaiik). Клиент за наличный или безналичный расчет приобретает у "банка" на некоторую сумму последовательности символов (для них "банк" гарантирует нетривиальность алгоритма генерации и уникальность каждого экземпляра), которыми расплачивается с торговцем. Торговец возвращает их в "банк" в обмен на ту же сумму, за вычетом комиссионных. При этом на "банке" лежит обязанность контролировать валидность поступающих жетонов (проверяя их наличие в регистре исходящих) и их единичность (проверяя отсутствие в регистре

входящих). Стороны могут использовать криптографические средства защиты информации с открытыми ключами, чтобы избежать перехвата жетонов.

Такая схема проста в реализации и эксплуатации. Однако правовой статус сделок с использованием таких суррогатов остается расплывчатым, равно как и фискальные обязанности клиентов, приобретающих товары и услуги у торговцев, находящихся под другой юрисдикцией.

Расширения несетевых расчетных систем

По другому пути пошла компания CyberCash, первой предложившая технологию, позволяющую использовать пластиковые карточки для расчетов в сети.

Предлагаемое этой компанией программное обеспечение использует криптозащиту с открытым ключом для конфиденциальной передачи данных о пластиковой карточке от покупателя к торговцу

Сетевые электронные наличные

Дэвид Чом (David Chaum), известный ученый-криптолог и бизнесмен, а также ряд его коллег, пришли к идее электронной (или цифровой) наличности - платежного средства, которое объединит удобство электронных расчетов с конфиденциальностью наличных денег. В Internet представлены две технологии, реализующие эту идею. Компания Mondex, возглавляемая Тимоти Джонсом (Timothy Jones), предлагает сетевую версию электронного кошелька, реализованную в виде аппаратно-программного комплекса. Компания DigiCash под руководством Д. Чома представила технологию сетевых электронных денег ecash в чисто программном варианте. Рассмотрим это решение.

В ядре технологии лежит все тот же прием криптозащиты с открытыми ключами. Эмитент электронной наличности (банк) имеет, кроме обычной пары ключей, аутентифицирующей его, еще и последовательность пар ключей, в соответствие которым ставятся номиналы "цифровых монет".

Снятие наличных со счета производится следующим об-

разом. В ходе сеанса связи клиент и банк (точнее, их программы - представители) аутентифицируют друг друга. Затем клиент генерирует уникальную последовательность символов, преобразует ее путем "умножения" на случайный множитель (blin-ding factor), "закрывает" результат открытым ключом банка и отправляет "монету" в банк. Банк "раскрывает" "монету", используя свой секретный ключ, "заверяет" ее электронной подписью, соответствующей номиналу "монеты", "закрывает" ее открытым ключом клиента и возвращает ее клиенту, одновременно списывая соответствующую сумму с его счета. Клиент, получив "монету", "открывает" ее с помощью своего секретного ключа, затем "делит" ее символьное представление на запомненный случайный множитель и сохраняет результат в "кошельке". Транзакция завершена. Теперь банк готов принять эту монету, от кого бы она ни поступила (разумеется, лишь один раз).

Использование blin-ding factor и составляет суть приема "слепой подписи", предложенного Чомом в дополнение к обычному методу криптозащиты с открытыми ключами. Благодаря использованию "слепой подписи" банк не в состоянии накапливать информацию о плательщиках, в то же время сохраняя возможность следить за однократным использованием каждой "монеты" данным клиентом и идентифицировать получателя каждого платежа. Чом называет такую логику взаимодействия сторон "односторонней безусловной непрослеживаемостью" платежей. Покупатель не может быть идентифицирован даже при сговоре продавца с банком. В то же время, покупатель при желании может идентифицировать себя сам и доказать факт осуществления сделки, апеллируя к банку. Такая логика призвана воспрепятствовать криминальному использованию электронной наличности.

Для вложения наличности клиент просто связывается с банком и отправляет ему полученную "монету", закрыв ее открытым ключом банка. Банк проверяет, не была ли она уже использована, заносит номер в регистр входящих и зачисляет

соответствующую сумму на счет клиента.

Сделка между двумя клиентами предполагает лишь передачу "монеты" от покупателя к продавцу, который может либо сразу попытаться внести ее в банк, либо принять ее на свой страх и риск без проверки. Вместе с "монетой" передается некоторая дополнительная информация, которая сама по себе не может помочь идентификации плательщика, но в случае попытки дважды использовать одну и ту же монету позволяет раскрыть его личность.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. В каких вычислительных системах процесс обмена данными проявляется наиболее ярко?
2. Какова классификация вычислительных сетей?
3. В чем отличие локальных и глобальных вычислительных сетей?
4. Перечислите и поясните базовые топологии вычислительных сетей.
5. Расскажите о маркерных и тактированных кольцевых сетях.
6. Нарисуйте схемы комбинированных топологий компьютерных сетей.
7. Нарисуйте и поясните типовую топологию глобальной вычислительной сети.
8. Расскажите о методах коммутаций в компьютерных сетях, сделайте их сравнительную оценку.
9. В чем суть базовой эталонной модели открытых систем и функций каждого из семи уровней?
10. Что такое протокол обмена данными в компьютерной сети? Какова иерархия протоколов эталонной модели открытых систем?
11. Какие функции выполняет процедура передачи данных? На каком уровне эталонной модели она реализуется?

12. В чем преимущество фазовой модуляции перед другими видами модуляции?
13. Расскажите об устройстве, назначении и характеристиках модемов.
14. Для чего выполняется операция кодирования сообщений при передаче? Объясните принципы кодирования.
15. Что такое емкость канала связи? Каким фактором она определяется?
16. Что такое уплотнение канала связи? Какие существуют виды уплотнения?
17. Какие приняты стандарты скорости передачи данных по каналам связи?
18. Сколько основных способов доступа к каналу вы знаете? Кратко опишите их суть.
19. Что представляет собой коллизия?
20. Какой максимальной производительности позволяет добиться стратегия доступа типа Чистой Алохи?
21. Каково максимальное значение нормированной производительности S при $G = 1$ для синхронной Алохи?
22. Что такое «проверка несущей»?
23. Расшифруйте аббревиатуру МДПН/ОС
24. Что подразумевает процедура двоичного замедления?
25. Какова эффективность доступа к каналу типа МДПН/ОС по сравнению с Алохой?
26. Какому уровню принадлежит функция маршрутизации пакетов, передаваемых через сеть?
27. Поясните алгоритм Дейкстры.
28. Поясните алгоритм Флойда?
29. Что такое Internet?
30. Какова система адресации в Internet?
31. Какие сетевые протоколы применяются в Internet?
32. Расскажите о ресурсах Internet.
33. В чем особенность системы World Wide Web (WWW)?
34. Расскажите о коммерческом использовании Internet.
35. Как производятся в Internet финансовые расчеты?

Литература

1. *Архитектура компьютерных систем и сетей: Учебное пособие.* / **Т.П. Барановская, В.И. Лойко, М.И. Семенов, А.И. Трубилин.** Под ред. **В.И. Лойко.** - М: Финансы и статистика, 2003. - 291 с.: ил.
2. *Информационные системы и технологии в экономике: Учебник* / **Т.П. Барановская, В.И. Лойко, М.И. Семенов, А.И. Трубилин.** Под ред. **В.И. Лойко.** - М: Финансы и статистика, 2006. - 426 с.: ил.
3. *Microsoft Corporation.* Компьютерные сети. Учебный курс/Пер. с англ. - М.: Издательский отдел "Русская редакция" ТОО "Channel Trading Ltd.".2002.
4. **Вентцель Е.С.** Исследование операций: задачи, принципы, методология .— М.: Наука, 2002.
5. *Методы управления ресурсами вычислительных систем:* Учебное пособие/**П.П. Кравченко, А.Г. Чефранов;** Таганрог. радиотехн. ин-т. Таганрог, 2001.
6. **Семенов М.И., Трубилин И.Т., Лойко В.И., Барановская Т.П.** Автоматизированные информационные технологии в экономике: Учебник для вузов. - Москва: Финансы и статистика, 2002. - 416 с.: ил.
7. **Советов Б.Я.** Информационная технология: Учеб. для вузов по спец. "Автоматизир. системы обработки информ. и упр.". - М.: Высш. шк.,1994.
8. **Шварц М.** Сети связи: протоколы, моделирование и анализ: В 2-х ч.: Пер. с англ. - М.: Наука, 1992.
9. **Якубайтис Э.А.** Информационные сети и системы. - М.: Финансы и статистика, 2003.
10. **Шнитман В.** Современные высокопроизводительные компьютеры. Центр Информационных Технологий, 2006. <http://www.citmgu.ru..>
11. **Воеводин В.В.** Параллельная обработка данных. Курс лекций, 2007. <http://www.citforum.ru/>; <http://www.parallel.ru>

Лойко Валерий Иванович

Вычислительные системы, сети и телекоммуникации

Электронный учебник

350044, Краснодар, Калинина, 13
Кубанский государственный аграрный университет