Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

В.Г. Резник

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

Учебное пособие

Тема 1. Введение в теорию вычислительных систем и сетей

Томск
2020

УДК 004.75

Резник, Виталий Григорьевич

Распределенные вычислительные сети. Учебное пособие. Тема 1. Введение в теорию вычислительных сетей / В.Г. Резник. – Томск : Томск. гос. ун-т систем упр. и радиоэлектроники, 2019. – 35 с.

В учебном пособии рассмотрены основные современные технологии организации распределенных вычислительных сетей, которые уже получили достаточно широкое распространение и подкреплены соответствующими инструментальными средствами реализации распределенных приложений. Представлены основные подходы к распределенной обработке информации. Проводится обзор организации распределенных вычислительных систем: методы удалённых вызовов процедур, многослойные клиент-серверные системы, технологии гетерогенных структур и одноранговых вычислений. Приводится описание концепции GRID-вычислений и сервис-ориентированный подход к построению распределенных вычислительных систем. Рассматриваемые технологии подкрепляется описанием инструментальных средств разработки программного обеспечения, реализованных на платформе языка Java.

Пособие предназначено для студентов бакалавриата по направлению 09.03.01 «Информатика и вычислительная техника» при изучении курсов «Вычислительные системы и сети» и «Распределенные вычислительные системы».

Одобрено на заседании каф. АСУ протокол №____ от _____

УДК 004.75

Оглавление

1 Тема 1. Введение в теорию вычислительных сетей

Терминологические разночтения в учебной литературе и научных работах, связанных с распределенной обработкой информации, затрудняют освоение соответствующих дисциплин. Назначение данной главы — уточнить значение используемых терминов, а также ограничить рамки изучаемого материала.

В фундаментальном курсе по архитектуре и структуре современных компьютерных средств, написанных С.А. Орловым и Б.Я. Цилькером [2], даются следующие определения понятиям «вычислительная машина» и «вычислительная система»: «... *Вычислительная машина* (ВМ) — совокупность технических средств, создающая возможность проведения обработки информации (данных) и получение результата в необходимой форме. Под техническими средствами понимают все оборудование, предназначенное для автоматизированной обработки данных. Как правило, в состав ВМ входит и системное программное обеспечение. ВМ, основные функциональные устройства которой выполнены на электронных компонентах, называют электронной вычислительной машиной (ЭВМ).

В свою очередь, *вычислительную систему* (ВС) стандарт ISO/IEC 2382/1-93 определяет как одну или несколько вычислительных машин, периферийное оборудование и программное обеспечение, которые выполняют обработку данных. ...».

Продолжая следовать рассуждениям учебника [2], мы приходим к понятию «Архитектура компьютера» (Computer architecture), а затем — к понятию уровни детализации структуры ВМ, например, так как показано на рисунке 1.1.

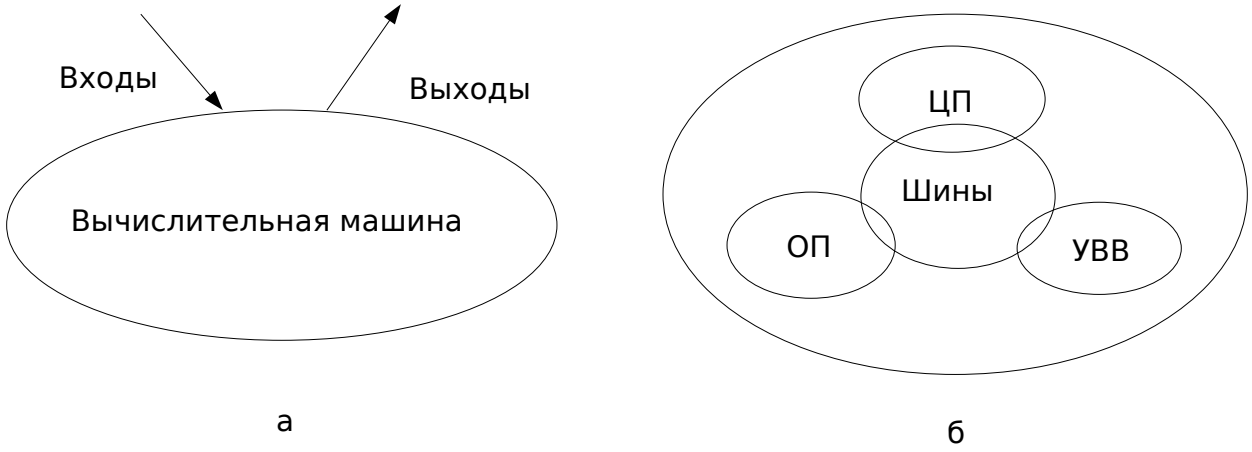

Рисунок 1.1 — Уровни детализации вычислительной машины [2]: а — уровень «чёрного ящика»; б — уровень общей архитектуры

Первоначально введённые термины и обозначения не вызывают особых вопросов. Например, в левой части рисунка «Входы» и «Выходы» обозначают связь отдельной вычислительной машины с другими ВМ посредством каналов связи или сетевых устройств, что позволяет создавать разнообразные вычислительные системы. Элементы правой части рисунка вводят обозначения связанной между собой совокупности технических средств ВМ. Все это соответствует общеприня-

той логике рассуждений и не вызывают гностических (познавательных) противоречий.

Терминологические сложности начинаются с момента содержательного анализа понятия «*Вычислительные системы*» (ВС). Здесь все уважаемые источники [1-5] дают различные толкования, которые, если и приемлемы для подготовленного специалиста, то — не допустимы для учебного пособия уровня бакалавриата. Особую проблему вызывает толкование ВС (как определение макроуровня), которое затем детализируется и раскрывается по мере описания решаемой задачи. В результате образуется комбинаторный набор понятий, которые разрушают иерархическую зависимость используемых определений и создают гносеологические трудности у обучающихся. Примером такого определения является базовое понятие «*Распределенные вычислительные системы*» используемое в названии учебного пособия [5].

Чтобы устранить указанные выше недостатки, в данном учебном пособии используется терминология ориентированная на последний выпуск государственного стандарта: «ГОСТ 33707-2016 (ISO/IEC 2382:2015). Информационные технологии (ИТ). Словарь» [9], где базовыми понятиями макроуровня являются: «... 4.1270 **система обработки данных**; СОД: Система, выполняющая автоматизированную обработку данных и включающая аппаратные средства, программное обеспечение и соответствующий персонал (data processing system, computer system, computing system).
4.1271 **система обработки информации**; СОИ: Совокупность технических средств и программного обеспечения, а также методов обработки информации и действий персонала, обеспечивающая выполнение автоматизированной обработки информации (information processing system). ...».

Существенная разница между СОД и СОИ, сложно воспринимаемая на слух, состоит в том, что:

- **СОД** — выполняет автоматизированную обработку данных;
- **СОИ** — только обеспечивает автоматизированную обработку информации.

Далее, в пределах изучаемого материала, системы не конкретизируются до различий между СОД и СОИ. Тем не менее, за основу классификации изучаемых систем выбрано понятие ***СОД***, которое подробно описано в учебнике [4]. Кроме того, для большего соответствия стандарту [9, раздел 2], используется следующий «Список сокращений и условных обозначений»:

ЭВМ — электронно-вычислительная машина;
СУБД — система управления базами данных;
ПК — персональный компьютер;
ПЗУ — постоянное запоминающее устройство;
ОС — операционная система;

ОЗУ — оперативное запоминающее устройство;

ЛВС — локальная вычислительная сеть;

ИС — 1. интегральная схема; 2. информационная система; 3. интеллектуальная собственность;

ИИ — искусственный интеллект;

ЗУ — запоминающее устройство;

ВС — вычислительная система;

БД — база данных.

Чтобы в должной степени раскрыть подробности изучаемой предметной области, выделим основные подразделы данного подраздела, где излагаются главные теоретические положения, касающиеся вычислительных сетей:

- ***Общая классификация систем обработки данных*** представлена мнением А.М. Ларионова, С.А. Майорова и Г.И. Новикова, изложенным в материалах первой главы ученика [4] и положенным в основу изучаемой дисциплины. Это мнение сравнивается с принципами и парадигмами фундаментального труда Э. Таненбаума [3].
- ***Сетевые объектные системы*** отражают начальную стадию теоретического построения распределенных систем, появившихся в эпоху бурного развития сетевых технологий. Различные технологические аспекты этого подраздела подробно излагаются в разделах 3 и 4 данного учебного пособия.
- ***Сервис-ориентированные технологии*** отражают текущие теоретические построения, положенные в основу теории и практики современных распределённых систем в качестве концепции SOA. Теоретические положения этой части кратко раскрываются в последнем подразделе 5.
- ***Виртуальные системы*** отражают общий методологический принцип построения современных сложных систем. В этом плане, виртуальные системы связывают общие вопросы проектирования различных технических систем с инструментальными средствами их разработки. Конечным пунктом такого видения являются языковые и инструментальные средства Java, базовые основы которого излагаются во втором разделе данного учебного пособия и последовательно распространяются на весь учебный материал последующих разделов.

1.1 Общая классификация систем обработки данных

Предметная область изучаемой дисциплины находится в пространстве понятий, которые определяются термином «*Системы обработки данных*» (СОД). Для содержательного раскрытия этого термина воспользуемся классификацией Ларионова (А.М. Ларионов, С.А. Майоров и Г.И. Новиков) [4], которая представлена в общем виде на рисунке 1.2.

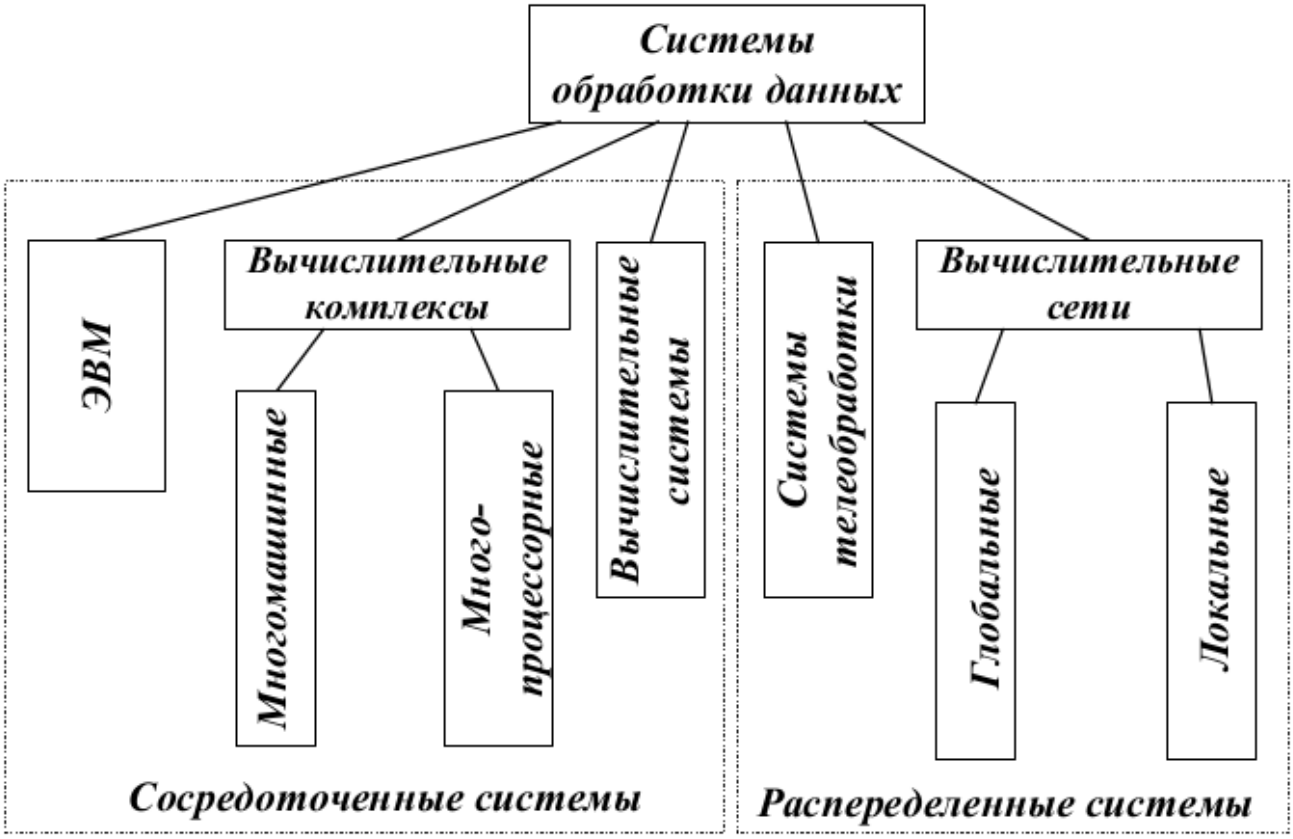

Рисунок 1.2 — Классификация СОД [4]

Представленная классификация снимает терминологические трудности определений, отмеченные во введении и начале данного раздела. Хорошо видна иерархическая структура предложенной классификации, которая выделяет две группы систем: «*Сосредоточенные системы*» и «*Распределенные системы*». Достаточно чётко обозначена предметная область изучаемой дисциплины: «*Вычислительные сети*», а сами авторы так характеризуют предложенную классификацию [4, стр. 10]: «... СОД, построенные на основе отдельных ЭВМ, вычислительных комплексов и систем, образуют класс *сосредоточенных (централизованных) систем*, в которых вся обработка реализуется ЭВМ, вычислительным комплексом или специализированной системой. Системы телеобработки и вычислительные сети относятся к классу *распределенных систем*, в которых процессы обработки

данных рассредоточены по многим компонентам. При этом системы телеобработки считаются распределенными в некоторой степени условно, поскольку основные функции обработки данных здесь реализуются централизованно – в одной ЭВМ или вычислительном комплексе. ...».

Особое внимание уделяется *интерфейсам*, которые являются элементами сопряжения и подразделяются на: *параллельные*, *последовательные* и *связанные* интерфейсы. Для уточнения отметим [9]: «... 4.447 **интерфейс**: Совместно используемая граница между двумя функциональными единицами, определяемая различными функциональными характеристиками, параметрами физического соединения, параметрами взаимосвязи при обмене сигналами, а также другими характеристиками в зависимости от задаваемых требований (interface). П р и м е ч а н и е - Примерами интерфейсов являются RS232, RS422, RS485 и радиоинтерфейс. ...».

Следует отметить, что авторы учебника [4] (Ларионов А.М.) являются специалистами очень высокого класса, участвовавшими в разработках большого количества отечественной цифровой вычислительной техники, включая многопроцессорный вычислительный комплекс (МПВК) «Эльбрус».

Прежде чем перейти к рассмотрению компонент модели СОД отметим, что ЭВМ и другие выделенные системы включают не только аппаратные средства, показанные на рисунке 1.1б, но и программное обеспечение ОС, базовый состав которой адекватно соответствует изучаемой компоненте. Более того, современные ЭВМ фактически не выпускаются без программного обеспечения, которое включает ПО BIOS (UEFI) и firmware (программные «прошивки» аппаратной части ЭВМ). Этот факт хорошо известен студентам уже изучившим курс «*Операционные системы*». Например, дистрибутив ОС Arch Linux, установленный в апреле 2019 года, содержал ПО драйверов — порядка 86 МБ, а ПО firmware — 376 МБ.

1.1.1 Сосредоточенные системы

Следуя рассуждениям авторов учебника [4], выделяются: *ЭВМ*, «*Вычислительные комплексы*» и «*Вычислительные системы*».

ЭВМ — одномашинная СОД с традиционной однопроцессорной архитектурой и программным обеспечением ОС, допускающим развитие такой системы в плане установки и эксплуатации инструментального и прикладного ПО. Такой тип СОД хорошо известен студентам в плане изучения различных курсов, включая дисциплину «ЭВМ и периферийные устройства».

Вычислительные комплексы — СОД построенные по одному из двух принципов: «*Многомашинные вычислительные комплексы*» или «*Многопроцессорные вычислительные комплексы*». Основное внимание в таких архитектурах уделяется аппаратным средствам, которые обеспечивают распараллеливание вычислений, что в конечном итоге ориентировано на увеличение скорости работы приложений.

Важной отличительной особенностью таких комплексов является наличие только базового программного обеспечения ОС, которое создаёт единую систему, скрывая детали аппаратной реализации.

Многомашинные вычислительные комплексы — разновидность систем, которые включают несколько ЭВМ и предназначены для повышения надёжности и производительности СОД. Они появились в начале 60-х годов и строились на основе высокоскоростных адаптеров, обеспечивая *прямую связь* между ЭВМ, или на основе внешних накопителей информации, обеспечивая *косвенную связь* между ЭВМ посредством магнитных дисков или магнитных лент.

Многопроцессорные вычислительные комплексы — разновидность систем, которые включают более одного процессора, но используют общую память СОД. Само взаимодействие процессоров и памяти осуществляется через специальную коммуникационную среду, определяющую различные качественные характеристики системы в целом.

Вычислительные системы — СОД настроенные на решение задач конкретной области применения. Другими словами, это — **Вычислительные комплексы**, дополненные прикладным программным обеспечением, специализированным для конкретной предметной области.

Таким образом, термин «*Вычислительные системы*» является эквивалентом термину «*Сосредоточенные системы*» и в таком виде используется не только в отдельных научных публикациях, но и в учебной литературе.

Чтобы обосновать заявленное утверждение, обратимся к другой классификации, известной как таксономия Флинна. Для этого процитируем, например, статью Википедии [11]: «... **Таксономия** (**Классификация**) **Флинна** (англ. *Flynn's taxonomy*) — общая общая классификация архитектур ЭВМ по признакам наличия параллелизма в потоках команд и данных. Была предложена Майклом Флинном в 1966 году и расширена в 1972 году. ... Все разнообразие архитектур ЭВМ в этой таксономии Флинна сводится к четырём классам:

- **ОКОД** — Вычислительная система с **о**диночным потоком **к**оманд и **о**диночным потоком **д**анных (SISD, **s**ingle **i**nstruction stream over a **s**ingle **d**ata stream).
- **ОКМД** — Вычислительная система с **о**диночным потоком **к**оманд и **м**ножественным потоком **д**анных (SIMD, **s**ingle **i**nstruction, **m**ultiple **d**ata).
- **МКОД** — Вычислительная система со **м**ножественным потоком **к**оманд и **о**диночным потоком **д**анных (MISD, **m**ultiple **i**nstruction, **s**ingle **d**ata).
- **МКМД** — Вычислительная система со **м**ножественным потоком **к**оманд и **м**ножественным потоком **д**анных (MIMD, **m**ultiple **i**nstruction, **m**ultiple **d**ata). ...».

Приведённый пример наглядно показывает неправомерное использование термина «*Вычислительные системы*» взамен гораздо более адекватного термина

«*Вычислительные комплексы*». В частности, отметим [11]: «... Архитектура *SISD* — это традиционный компьютер фон-Неймановской архитектуры с одним процессором, который выполняет последовательно одну инструкцию за другой, работая с одним потоком данных. В данном классе не используется параллелизм ни данных, ни инструкций, и, следовательно, SISD-машина не является параллельной. К этому классу также принято относить *конвейерные*, *суперскалярные* и *VLIW*-процессоры. ...».

Поскольку «*Сосредоточенные системы*» не являются предметом нашего изучения, то ограничим их рассмотрение приведённой выше детализацией.

1.1.2 Распределенные системы

На рисунке 1.2 «*Распределенные системы*» представлены как «*Системы телеобработки*» и «*Вычислительные сети*». Согласно [4, стр. 7]: «... **Системы телеобработки**. Уже первоначальное применение СОД для управления производством, транспортом и материально-техническим снабжением показало, что эффективность систем можно значительно повысить, если обеспечить ввод данных в систему непосредственно с мест их появления и выдачу результатов обработки к местам их использования. Для этого необходимо связать СОД и рабочие места пользователей с помощью каналов связи. Системы, предназначенные для обработки данных, передаваемых по каналам связи, называются *системами телеобработки данных*.

Состав технических средств системы телеобработки данных укрупненно представлен на рис. 1.3. Пользователи (абоненты) взаимодействуют с системой посредством терминалов (абонентских пунктов), подключаемых через каналы связи к средствам обработки данных – ЭВМ или вычислительному комплексу. Данные передаются по каналам связи в форме сообщений – блоков данных, несущих в себе кроме собственно данных служебную информацию, необходимую для управления процессами передачи и защиты данных от искажений. Программное обеспечение систем телеобработки содержит специальные средства, необходимые для управления техническими средствами, установления связи между ЭВМ и абонентами, передачи данных между ними и организации взаимодействия пользователей с программами обработки данных. ...».

В современных представлениях, благодаря сетевым технологиям, «*Системы телеобработки данных*», представленные в виде рисунка 1.3, потеряли свою актуальность и используются только в специализированных областях, например, системы дальней космической связи или в системах диспетчеризации (SCADA), обеспечивая удалённый (полевой) уровень сбора данных. На передний план вышли сетевые технологии, которые авторами учебника [4, стр. 8-9] характеризуются следующим образом: «… **Вычислительные сети**. С ростом масштабов применения электронной вычислительной техники в научных исследованиях, проектно-конструкторских работах, управлении производством и транспортом и

прочих областях стала очевидна необходимость объединения СОД, обслуживающих отдельные предприятия и коллективы. Объединение разрозненных СОД обеспечивает доступ к данным и процедурам их обработки для всех пользователей, связанных общей сферой деятельности.

В конце 60-х годов был предложен способ построения вычислительных сетей, объединяющих ЭВМ (вычислительные комплексы) с помощью базовой сети передачи данных. Структура вычислительной сети в общих чертах представлена на рис. 1.4. Ядром является *базовая сеть передачи данных* (СПД), которая состоит из каналов и *узлов связи* (УС). Узлы связи принимают данные и передают их в направлении, обеспечивающем доставку данных абоненту. ЭВМ подключаются к узлам базовой сети передачи данных, чем обеспечивается возможность обмена данными между любыми парами ЭВМ. Совокупность ЭВМ, объединённых сетью передачи данных, образует *сеть ЭВМ*. ...».

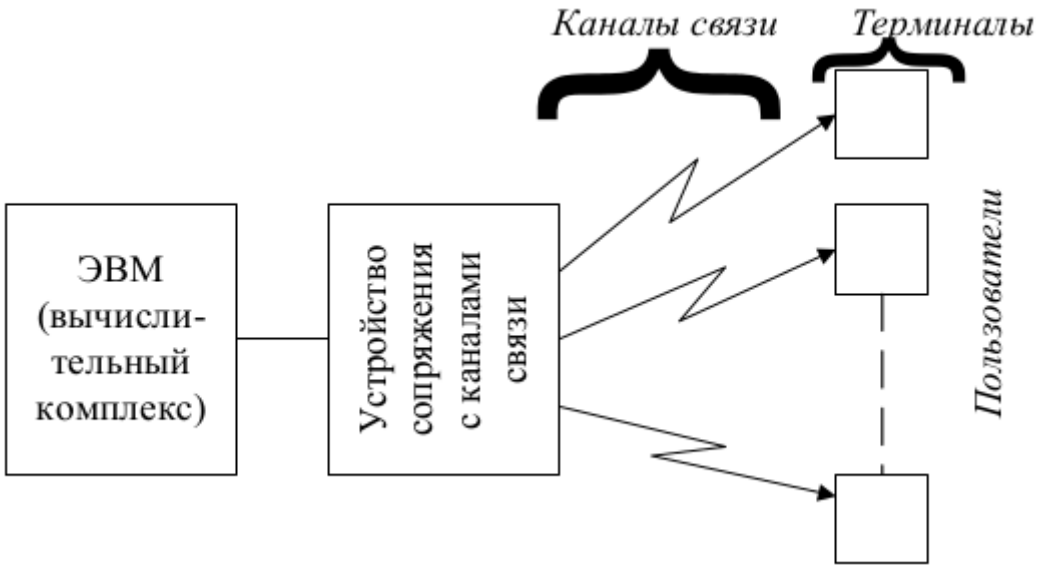

Рисунок 1.3 - Система телеобработки данных [4]

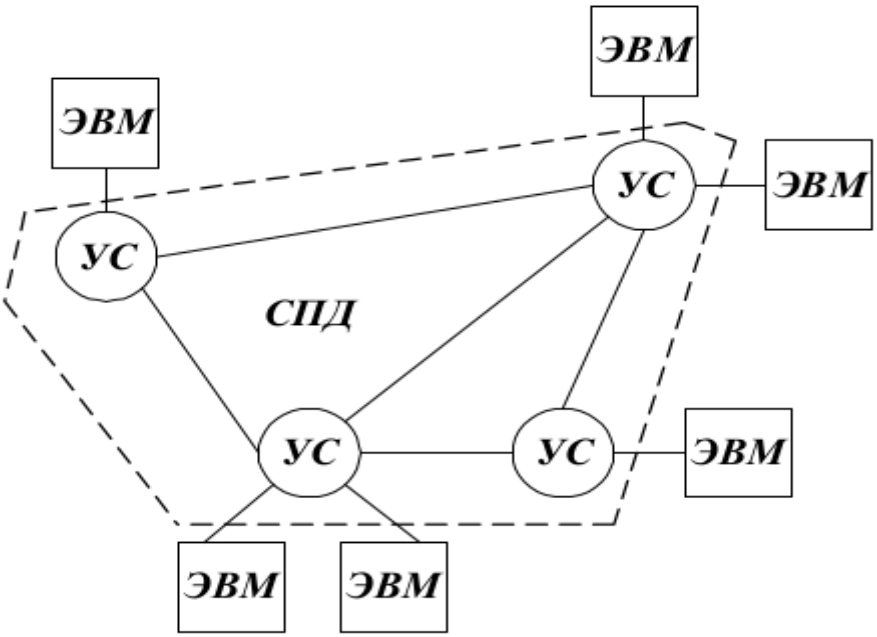

Рисунок 1.4 — Вычислительная сеть [4]

Такое представление термина «*Вычислительная сеть*» хорошо соответствует предметному материалу дисциплины «*Сети и телекоммуникации*», а также ГОСТ 33707-2016 [9]: «...4.223 **вычислительная сеть**: Сеть, узлы которой состоят из компьютеров и аппаратуры передачи данных, а ветви которой являются линиями передачи данных (computer network)». В любом случае, приведённые определения не раскрывают содержание термина «*Распределенные системы*», а стандарт [9] содержит только близкое по смыслу содержание: «4.1166 **распределенная обработка данных**; DDP: Обработка данных, при которой выполнение операций распределено по узлам вычислительной сети. П р и м е ч а н и е — Для распределенной обработки данных требуется коллективное сотрудничество, которое достигается путём обмена данными между узлами (distributed data processing, DDP). 4.1167 **распределенная система базы данных**: Совокупность данных, распределенных между двумя или более базами данных (distributed database). ...».

Чтобы раскрыть содержательную часть термина «*Распределенные системы*», обратимся к фундаментальной работе Эндрю Таненбаума «Распределенные системы. Принципы и парадигмы» [3]. В ней указывается, что распределенные системы реализуются в виде *системы (службы) промежуточного уровня* (middleware), которая показана на рисунке 1.5.

Рисунок 1.5 — Распределенная система [3]

Предполагается, что служба промежуточного уровня должна быть реализована в виде специального программного обеспечения, которое устанавливается на каждую ЭВМ, подключённую к сети. Перспективность такой модели Таненбаум обосновывает:

- значительным снижением цен на отдельные ЭВМ;
- широким распространением высокоскоростных *локальных сетей* (Local-Area Networks, LAN);
- значительным увеличением скорости обмена данными в *глобальных сетях* (Wide-Area Networks, WAN).

К сожалению работа Таненбаума [3] мало подходит в качестве базового учебного пособия по тематике изучаемой дисциплины. Действительно, как отмечено в ее предисловии: «Эта книга первоначально задумывалась как пересмотренный вариант книги Distributed Operating Systems, но вскоре мы поняли, что с 1995 года произошли такие изменения, что простым пересмотром здесь не обойтись. Нужна совершенно новая книга. Эта новая книга получила и новое название Distributed Systems: Principles and Paradigm». Возможно в будущем автор больше уделит внимания согласованию классификационных принципов построения распределенных систем. Чтобы обосновать это мнение, обратимся к рисунку 1.6, на котором Таненбаум выделяет группы систем с *шинной* и *коммутируемой* архитектурами.

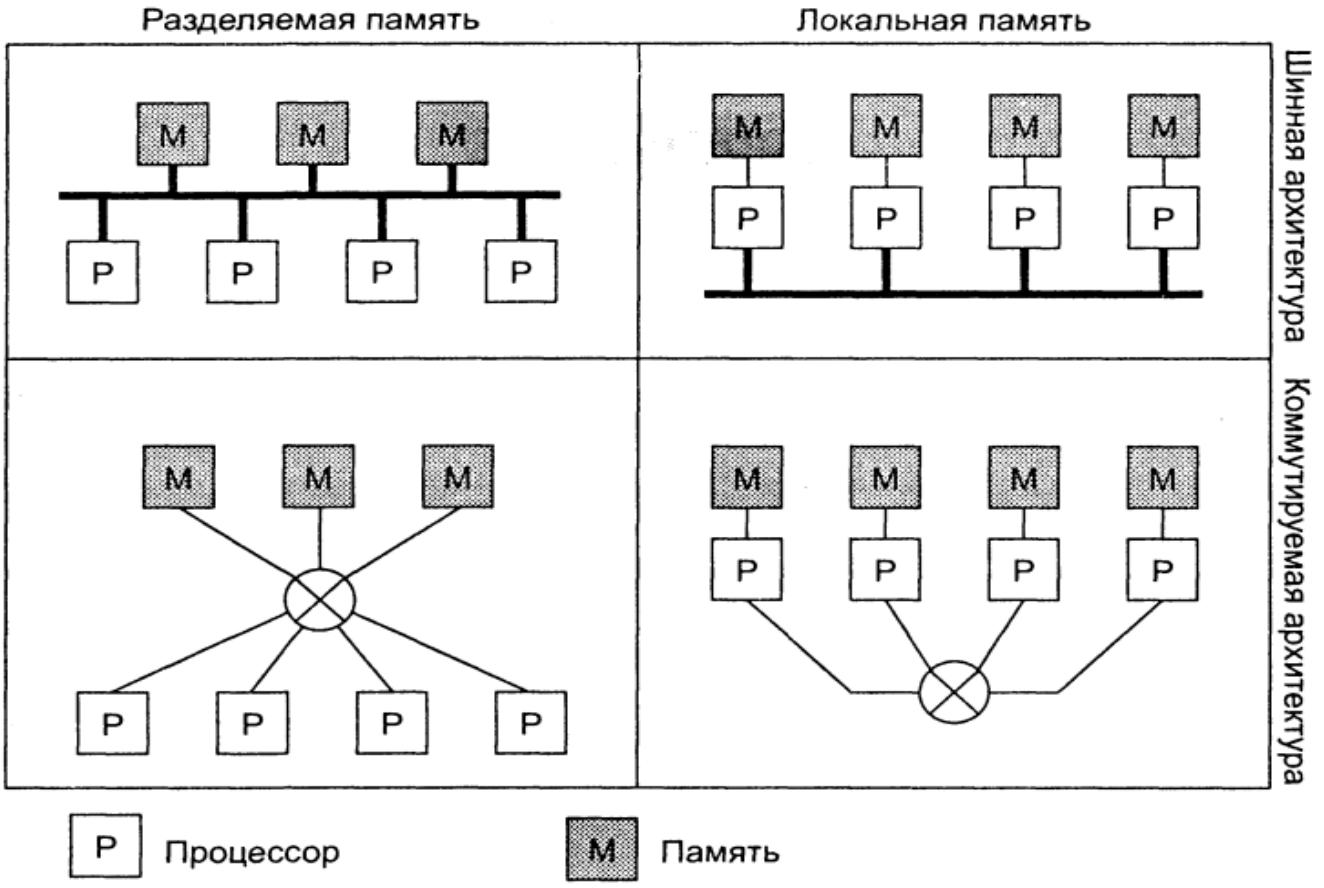

Рисунок 1.6 - Различные базовые архитектуры процессоров и памяти распределенных компьютерных систем [3]

Логически, показанные распределенные архитектуры можно рассматривать как в классе «*Вычислительные комплексы*», так и в классе «*Вычислительные сети*». Поэтому, чтобы избежать возможных последующих разночтений, введём более конкретный класс - «*Распределенные вычислительные сети*». Его и будем рассматривать как предметную область данного пособия.

1.1.3 Распределенные вычислительные сети

Определим предметную область «*Распределенных вычислительных сетей*» (РС-сетей) методом противопоставления понятий «*Распределенные системы*» и «*Сосредоточенные системы*».

Обратимся к публичному источнику Википедии, где прочитаем [12]: «... **Распределенная система** — система, для которой отношения местоположений элементов (или групп элементов) играют существенную роль с точки зрения функционирования системы, а, следовательно, и с точки зрения анализа и синтеза системы.

Для распределенных систем характерно распределение функций, ресурсов

между множеством элементов (узлов) и отсутствие единого управляющего центра, поэтому выход из строя одного из узлов не приводит к полной остановке всей системы. Типичной распределенной системой является Интернет.
Примеры распределенных систем:

- Распределенная система компьютеров — компьютерная сеть.
- Распределенная система управления — система управления технологическим процессом.
- Распределенная энергетика.
- Распределенная экономика.
- Распределенная файловая система — сетевые файловые системы.
- Распределенные операционные системы.
- Системы распределенных вычислений.
- Распределенные системы контроля версий.
- Распределенные базы данных
    - Система доменных имён (DNS) — распределенная система для получения информации о доменах».

Можно обсуждать являются ли «*Распределенные системы*» необходимым условием отсутствия единого управляющего центра, но наличие элементов (или групп элементов), которые являются «С*осредоточенными системами*», является обязательным. Например, DNS является распределенной системой, поскольку размещается на множестве DNS-серверов, которые обслуживают единую иерархическую структуру *доменных имён*. При этом, отдельный DNS-сервер представляет собой сосредоточенную систему и может рассматриваться как отдельная ЭВМ, комплекс или система. Другим примером являются «*Автоматизированные системы управления*» (АСУ, АСУП, АСУПП, АСУТП), практика создания которых хорошо согласуется с архитектурами рисунков 1.3 и 1.4, но для которых структуры рисунка 1.6 оказываются практически бессмысленными. В любом случае, обмен данными и управляющей информацией осуществляется между отдельными сосредоточенными системами.

Что касается современных «*Сосредоточенных систем*», то для них показанные на рисунке 1.6 архитектуры, действительно являются базовыми и исследуются в классе вычислительных комплексов. Тот факт, что Э. Таненбаум называет их распределенными, не обеспечивает полноценную реализацию распределенных приложений. В лучшем случае, такие приложения и на таких структурах могут только моделироваться или проектироваться, но не обеспечивать их нормальную целевую работу. Это подтверждается практикой создания различных АСУ, где обработка информации осуществляется параллельно и асинхронно на некотором наборе «*Автоматизированных рабочих мест*» (АРМ).

Чтобы более наглядно показать возможности «*Сосредоточенных систем*», вспомним их важнейшую характеристику — быстродействие вычислений. Для этого воспользуемся данными учебника [2] по иерархии запоминающих устройств

компьютеров. Сами данные представлены в виде таблицы 1.1 и отражают общепринятое название устройства, ёмкость памяти и время выборки данных.

Таблица 1.1 — Иерархия запоминающих устройств (по данным источника [2])

Название устройства	***Ёмкость памяти***	***Время выборки данных***
Регистры процессора	32 бита — 64 бита	Несколько системных циклов
Кэш-память: уровень L1 уровень L2 уровень L3	 32 КБ — 64 КБ 256 КБ — 2 МБ 4 МБ — 8 МБ	 Порядка 10 нс Порядка 25 нс Порядка 50 нс
Основная память	1 ГБ — 4 ГБ	50 — 100 нс
Твердотельные диски	256 ГБ — 1 ТБ	50 — 200 нс
Магнитные диски	512 ГБ — 6 ТБ	2.5 — 16 мс
Оптические диски CD DVD BD	 700 МБ 4.7 ГБ — 8.5 ГБ 25 ГБ — 128 ГБ	150 — 400 мс
Магнитные ленты	5 ТБ — 100 ТБ	500 — 1000 мс

Учитывая общие физические принципы распространения сигналов в электронных устройствах, можно утверждать, что основная тенденция развития сосредоточенных систем — концентрация всех их структур в минимальных физических объемах. Это позволяет создавать ВС различной мощности, включая суперкомпьютеры, но они остаются только элементами «*Распределенных систем*».

Из сказанного можно было бы сделать вывод, что термин «*Распределенные вычислительные системы*» (РВС) вполне подошёл бы для предметной области изучаемой дисциплины. Таким образом поступил и автор учебного пособия [5], тем самым игнорируя тот факт, что все приложения РВС явно взаимодействуют через систему, которая обозначена как СПД (система передачи данных). Чтобы более наглядно раскрыть эту ситуацию, кратко характеризуем классы сетевых объектных систем и сервис-ориентированные технологии.

1.2 Сетевые объектные системы

Любое взаимодействие приложений в сети описывается моделью OSI [13]: «... **Сетевая модель OSI** (англ. *open systems interconnection basic reference model* — **Б**азовая **Э**талонная **М**одель **В**заимодействия **О**ткрытых **С**истем (**ЭМВОС**)) — сетевая модель стека (магазина) сетевых протоколов OSI/ISO (ГОСТ Р ИСО/МЭК 7498-1-99). Посредством данной модели различные сетевые устройства могут взаимодействовать друг с другом. Модель определяет различные уровни взаимодействия систем. Каждый уровень выполняет определённые функции при таком взаимодействии. ...».

Сами уровни такой системы представлены на рисунке 1.7.

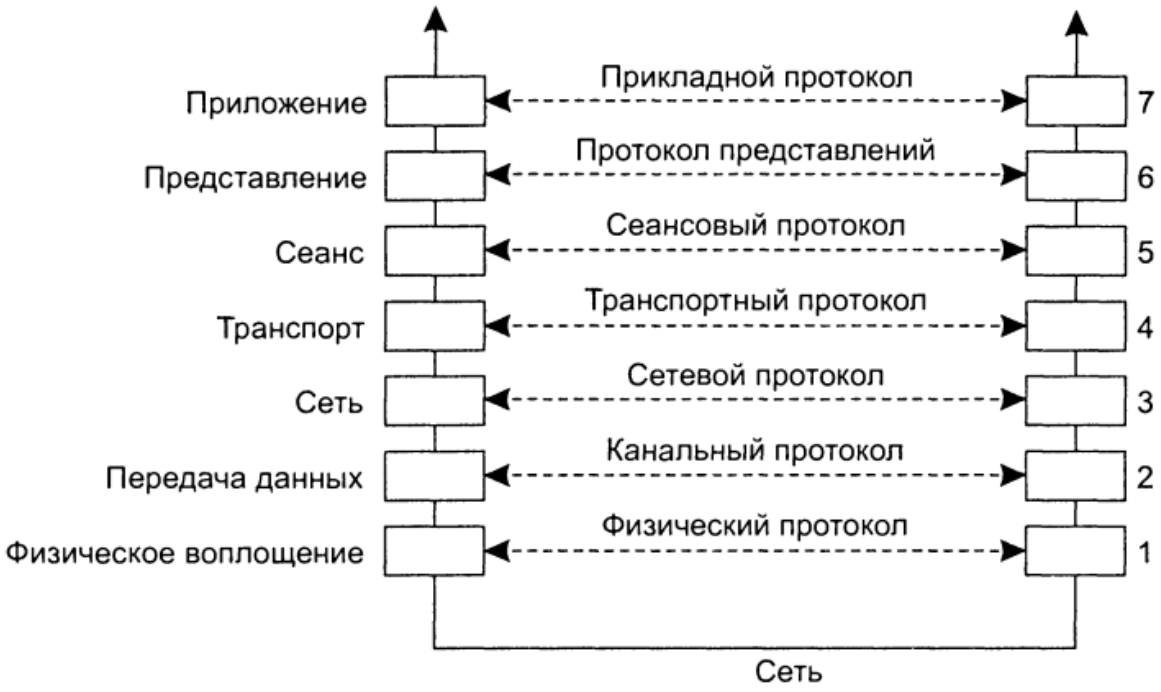

Рисунок 1.7 - Уровни, интерфейсы и протоколы модели OSI [3]

Студентам эта модель хорошо известна из курса «*Сети и телекоммуникации*», поэтому не будем описывать назначение каждого уровня. Для нас важно, что распределенные приложения обмениваются сообщениями, общая структура которых представлена на рисунке 1.8.

Технологии, основанные на передаче сообщений, породили различные подходы к реализации распределенных приложений. Эти подходы неуклонно смещались в сторону «*Распределенных объектных систем*», использующих:

- классические модели, основанные на модели ***OSI***;
- инструменты распределенной вычислительной среды (***DCE***);
- специальные сетевые системы: ***брокеры сообщений***;

- вызов методов удалённых объектов, например, ***RMI***.

Рисунок 1.8 — Типовое сетевое сообщение [3]

1.2.1 Классические приложения модели OSI

В модели OSI, «*Транспортный уровень*» отвечает за надёжную передачу массивов данных, поэтому все современные ОС содержат набор библиотек, обеспечивающих разработку сетевых приложений, классическими примерами которых являются:

- **Telnet** (**tele**type **net**work) — набор приложений и сетевой протокол для реализации текстового терминального интерфейса по сети; в современных реализациях, обеспечивающих защищённую работу в сети, заменён на приложения: **ssh** (***s**ecure **sh**ell* для ОС UNIX) и **PuTTY** (для ОС MS Windows);
- **FTP** (*File Transfer Protocol*) — набор приложений и сетевой протокол для передачи файлов по сети;
- **HTTP** (*HyperText Transfer Protocol*) — протокол передачи гипертекста, на основе которого создаются общеизвестные приложения: браузеры и web-сервера.

Все приведённые примеры имеют два главных общих признака:

- протоколы *Telnet*, *FTP* и *HTTP* являются оригинальными суммарными реализациями трёх протоколов верхнего уровня модели OSI: *сеансового*, *представления* и *прикладного*;
- все приложения реализованы по общей архитектуре [14]: «... **Клиент-сервер** (англ. *client–server*) — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами.

Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, ее размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики ее оборудования и программного обеспечения, ее также называют сервером, а машины, выполняющие клиентские программы, соответственно, - клиентами. ...».

Дополнительными признаками приведённых примеров являются:

- использование стека протоколов TCP/IP;
- первоначальная *реализация на языке программирования C* и последующий переход на объектно-ориентированные языки программирования (ООП), например, C++ и другие;
- использование *статической адресации* ЭВМ (*hosts*, *хостов*), требующих обращения к протоколам третьего (*сетевого*) уровня, а также фиксации точек доступа (*портов*) на четвёртом (*транспортном*) уровне;
- использование *централизованного управления сетевым взаимодействием* посредством DNS-серверов.

В общем случае, предметная область распределенных приложений не требует привязки к конкретным протоколам транспортного и более нижних уровней. Тем не менее, мы будем опираться на стек протоколов TCP/IP, поскольку на нём построены многие широкоизвестные технологи и имеется соответствующее программное обеспечение.

1.2.2 Распределенная вычислительная среда (DCE)

Естественным образом, расширяя технологическую модель основанную на архитектуре клиент-сервер, мы приходим к идее распределённой вычислительной среды, которая могла бы формироваться программным окружением ОС, например, так [15]: «... **Распределительная вычислительная среда** (англ. *Distributed Computing Environment*, сокр. DCE) — система программного обеспечения, разработанная в начале 1990-х годов в Open Software Foundation, который представлял со-

бой ассоциацию нескольких компаний: Apollo Computer, IBM, Digital Equipment Corporation и других. DCE предоставляет фреймворк и средства разработки клиент-серверных приложений. Фреймворк включает в себя:

- удалённый вызов процедур DCE/RPC;
- служба каталогов;
- служба связанная со временем;
- службу проверки подлинности;
- файловую систему DCE/DFS.

DCE стала важным шагом в направлении стандартизации архитектуры программного обеспечения, которые были зависимы от производителя. Как и сетевая модель OSI, DCE не получила большого успеха на практике, однако основные идеи (концепции) имели более существенное влияние, чем последующие исследования и разработки в этом направлении. ...».

Важнейшим технологическим инструментом DCE является удалённый вызов процедур [16]: «... **Удалённый вызов процедур**, реже **Вызов удалённых процедур** (от англ. *Remote Procedure Call, RPC*) — класс технологий, позволяющих компьютерным программам вызывать функции или процедуры в другом адресном пространстве (как правило, на удалённых компьютерах). Обычно реализация RPC-технологии включает в себя два компонента: сетевой протокол для обмена в режиме клиент-сервер и язык сериализации объектов (или структур, для необъектных RPC). ...».

Суть этой технологии отображена на рисунке 1.9.

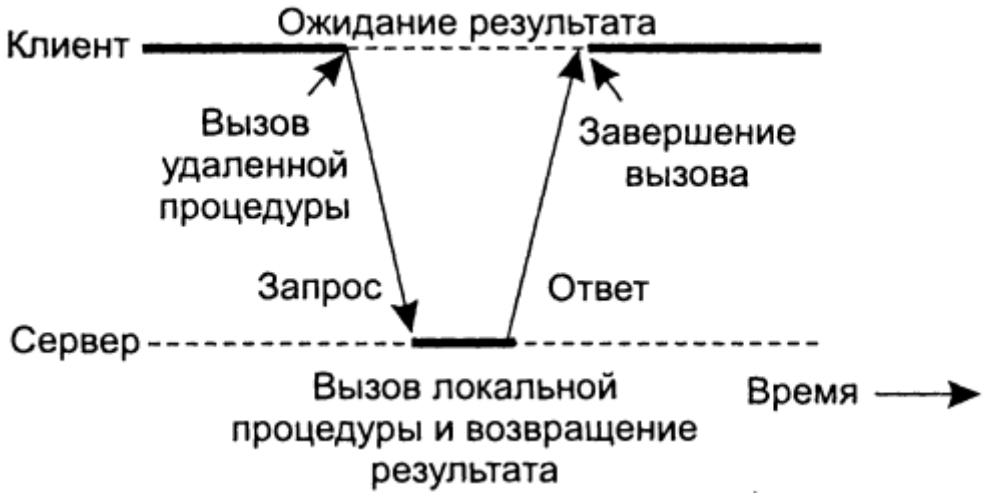

Рисунок 1.9 - Схема RPC между программами клиента и сервера [3]

Особенностью реализации технологии RPC является необходимость создания программных заглушек как на стороне клиента, так и на стороне сервера:

- **Client stub** — программная заглушка на стороне клиента;
- **Server stub** — программная заглушка на стороне сервера.

Поскольку вызов удалённых процедур осуществляется через заранее заданные интерфейсы, то, для упрощения разработки, интерфейсы часто описываются с использованием *языка определения интерфейсов* (***Interface Definition Language, IDL***). В последующем, такие описания компилируются специальными утилитами для создания заглушек на конкретных языках программирования.

В современных условиях, DCE как технология может претендовать только на создание ПО распределенных ОС, поскольку, находясь в «плену» низкоуровневых системных программных средств, не может обеспечивать необходимый уровень решений при создании сложных распределенных приложений.

1.2.3 Технология CORBA

Разработчикам распределенных систем требовались инструменты, которые бы манипулировали объектами на уровне сети ЭВМ.

Для этого, в 1989 году был создан консорциум [17]: «... **OMG** (сокр. от англ. *Object Management Group*, читается как [*о-эм-джи*]) — консорциум (рабочая группа), занимающийся разработкой и продвижением объектно-ориентированных технологий и стандартов. Это некоммерческое объединение, разрабатывающее стандарты для создания интероперабельных, то есть платформонезависимых, приложений на уровне предприятия. С консорциумом сотрудничает около 800 организаций — крупнейших производителей программного обеспечения. ...».

Основным достижением консорциума является брокерная архитектура взаимодействия ПО [18]: «... **CORBA** (обычно произносится [*ко́рба*], иногда жарг. [*ко́бра*]; англ. *Common Object Request Broker Architecture* — общая архитектура брокера объектных запросов; типовая архитектура опосредованных запросов к объектам) — технологический стандарт написания распределенных приложений, продвигаемый консорциумом (рабочей группой) OMG и соответствующая ему информационная технология. … Технология CORBA создана для поддержки разработки и развёртывания сложных объектно-ориентированных прикладных систем. CORBA является механизмом в программном обеспечении для осуществления интеграции изолированных систем, который даёт возможность программам, написанным на разных языках программирования, работающим в разных узлах сети, взаимодействовать друг с другом так же просто, как если бы они находились в ад-

ресном пространстве одного процесса. … GIOP (General Inter-ORB Protocol) — абстрактный протокол в стандарте CORBA, обеспечивающий интероперабельность брокеров. ... Архитектура GIOP включает несколько конкретных протоколов:

1. Internet InterORB Protocol (IIOP) (Межброкерный протокол для Интернет) — протокол для организации взаимодействия между различными брокерами, опубликованный консорциумом OMG. IIOP используется GIOP в среде интернет, и обеспечивает отображение сообщений между GIOP и слоем TCP/IP.
2. SSL InterORB Protocol (SSLIOP) — IIOP поверх SSL, поддерживаются шифрование и аутентификация.
3. HyperText InterORB Protocol (HTIOP) — IIOP поверх HTTP. ...».

Главные теоретические и практические достижения технологии CORBA достаточно подробно рассматриваются в разделе 3. Они наглядно показывают, что распределенные системы не могут полноценно строиться только на основе разнесения приложений по отдельным вычислительным системам подключённым к сети. «Службы промежуточного уровня» сами должны представлять собой систему, которая опирается на собственные протоколы, например, GIOP.

Этот вывод обосновывает заявленное название предметной области дисциплины как «Распределенные вычислительные сети».

1.2.4 Удалённый вызов методов

Как заявлено выше, удалённый вызов процедур — RPC является важнейшим технологическим инструментом DCE. Развитие этого инструмента применительно к взаимодействию объектов привело к созданию технологии **RMI** (*Remote Method Invocation*) — программному интерфейсу вызова удалённых методов в языке Java. Реализованный на основе протокола IIOP, RMI является простейшим инструментом реализации не очень сложных распределённых систем.

Технология RMI, как и технология CORBA, рассматриваются в третьем разделе данного пособия. Она имеет прежде всего практический интерес, раскрывающий методики программирования, заложенные технологиями DCE. Общая схема взаимодействия объектов RMI показана на рисунке 1.10.

Отметим, что развитие технологий RMI и CORBA, а также присущие им недостатки и ограничения, заложили основы более современных представлений, которые обобщенно называются - «***Сервис-ориентированные технологии***».

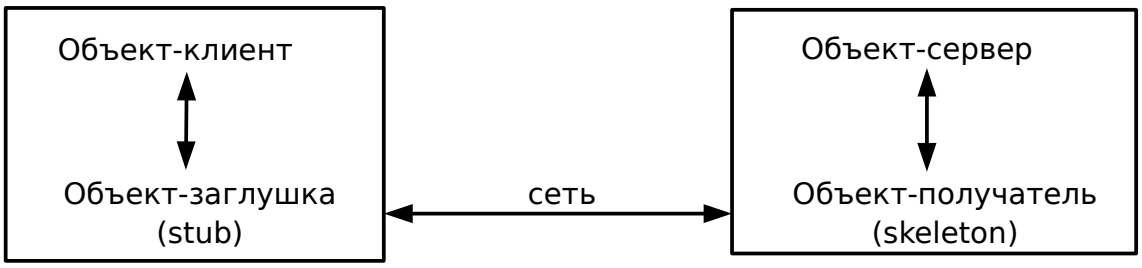

Рисунок 1.10 — Общая схема взаимодействия объектов RMI

1.3 Сервис-ориентированные технологии

Рассмотренные ранее технологии, в той или иной степени, были ориентированы на *технические аспекты* применения ЭВМ и сетей. Кроме того, все элементы вычислительной техники и системное программное обеспечение стоят немалых денег и, в процессе развития компьютерной индустрии, возникает понимание того, что на результатах работы ЭВМ тоже можно строить самостоятельный *бизнес*. Соответственно, в конце 90-х годов появляется термин «сервис» и формируется парадигма: «***всё есть сервис***». Окончательно, понимая, что этот термин охватывает слишком широкую сферу понятий, были сформированы более строгие трактовки:

- **Everything-as-a-Service** – всё как услуга.
- **Infrastructure-as-a-Service** (IaaS) – инфраструктура как услуга.
- **Platform-as-a-Service** (PaaS) – платформа как услуга.
- **Software-as-a-Service** (SaaS) — программное обеспечение как услуга.
- **Hardware-as-a-Service** (HaaS) — аппаратное обеспечение как услуга.
- **Workplace-as-a-Service** (WaaS) – рабочее место как услуга.
- **Data-as-a-Servise** (DaaS) – данные как услуга.
- **Secure-as-a-Service** — безопасность как сервис.

Хотя многие из перечисленных сервисов не получили должного развития или популярности, общая тенденция этого движения ведёт к *аутсорсингу*, трактовка которого даётся в следующих аспектах:

- **Аутсорсинг** - *outsourcing*: *outer-source-using* — использование внешнего источника/ресурса.
- **Аутсорсинг** — передача организацией на основании договора определённых *бизнес-процессов* или *производственных функций* на обслуживание другой компании, специализирующейся в соответствующей области.

Строго говоря, между терминами услуга сервиса и услуга аутсорсинга имеются следующие различия:

- ***услуга сервиса*** подразумевает разовый, эпизодический или случайный ха-

рактер и ограниченна началом и концом;

- ***услуга аутсорсинга*** обычно подразумевает функции по профессиональной поддержке *бесперебойной работоспособности* отдельных систем и инфраструктуры на основе длительного контракта (не менее одного года).

Краткое раскрытие тематики сервисно-ориентированных технологий приведено в пятом разделе данного пособия. Здесь мы отметим только:

- теоретические аспекты различия между *функциями* и *сервисами*;
- особое значение *middleware* — как «Службы промежуточного уровня»;
- важную потребность формирования «Сервисной шины предприятия».

1.3.1 Функции и сервисы

Общее понятие сервиса интерпретируется в рамках информационных технологий (***ИТ***) с помощью модели ***SOA***. Для этого, обратимся к публикации «SOA. Архитектурные особенности и практические аспекты» [19]: «... **Сервис-ориентированная архитектура** (service-oriented architecture, SOA) — подход к разработке программного обеспечения, в основе которого лежат сервисы со стандартизированными интерфейсами. … С помощью SOA реализуются три аспекта ИТ-сервисов, каждый из которых способствует получению максимальной отдачи от ИТ в бизнесе:

- **Сервисы бизнес-функций.** Суть этих сервисов заключается в автоматизации компонентов конкретных бизнес-функций, необходимых потребителю.
- **Сервисы инфраструктуры.** Данные сервисы выполняют проводящую функцию, посредством платформы, через которую поставляются сервисы бизнес-функций.
- **Сервисы жизненного цикла.** Эти сервисы являются своего рода «обёрткой», которая в большинстве случаев поставляет ИТ-пользователям «настоящие сервисы». Сервисы жизненного цикла отвечают за дизайн, внедрение, управление, изменение сервисов инфраструктуры и бизнес-функций. ...».

Связь между парадигмами объектно-ориентированного и сервис-ориентированного подходами показана на рисунке 1.11.

Рисунок 1.11 — Соотношение объектно-ориентированного и сервисного подходов в создании прикладных систем [19]

Из представленного рисунка хорошо видно, что сервис-ориентированный подход вводит следующие новые понятия:

- **BPM** (англ. *business process management*, **управление бизнес-процессами**) — концепция процессного управления организацией, рассматривающая бизнес-процессы как особые ресурсы, непрерывно адаптируемые к постоянным изменениям, и полагающаяся на такие принципы, как понятность и видимость бизнес-процессов в организации за счёт их моделирования с использованием формальных нотаций, использования программного обеспечения моделирования, симуляции, мониторинга и анализа бизнес-процессов [20].
- **EAI** (*Enterprise Application Integration*) — общее название сервиса интеграции прикладных систем предприятия.
- **AOP** (A*spect-Oriented Programming*) — аспектно-ориентированное программирование; парадигма программирования, основанная на идее разделения функциональности программы на модули.

Википедия даёт этому подходу следующую характеристику [21]: «... **Сервис-ориентированная архитектура** (***SOA***, англ. *service-oriented architecture*) — модульный подход к разработке программного обеспечения, основанный на использовании распределенных, слабо связанных (англ. *loose coupling*) заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам. Программные комплексы, разрабо-

танные в соответствии с сервис-ориентированной архитектурой, обычно реализуются как набор веб-служб, взаимодействующих по протоколу SOAP, но существуют и другие реализации (например, на базе *jini*, *CORBA*, на основе *REST*). ...».

1.3.2 Системы middleware

Концептуально, SOA реализуется на основе протокола SOAP [22]: «... **SOAP** (от англ. *Simple Object Access Protocol* — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределенной вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур. Официальная спецификация последней версии 1.2 протокола никак не расшифровывает название SOAP. SOAP является расширением протокола XML-RPC. SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS и др. Однако его взаимодействие с каждым из этих протоколов имеет свои особенности, которые должны быть определены отдельно. Чаще всего SOAP используется поверх HTTP. SOAP является одним из стандартов, на которых базируются технологии веб-служб. ...».

В процессе создания и эксплуатации систем, созданных на основе подхода SOA, стало ясно, что необходима промежуточная сетевая система. Стандартизацией подхода SOA занялся некоммерческий консорциум [23]: «... **OASIS** (англ. *Organization for the Advancement of Structured Information Standards*) — глобальный консорциум, который управляет разработкой, конвергенцией и принятием промышленных стандартов электронной коммерции в рамках международного информационного сообщества. Данный консорциум является лидером по количеству выпущенных стандартов, относящихся к Веб-службам. Кроме этого он занимается стандартизацией в области безопасности, электронной коммерции; также затрагивается общественный сектор и рынки узкоспециальной продукции. В OASIS входит свыше 5000 участников, представляющих более 600 различных организаций из 100 стран мира. Консорциум спонсируется ведущими корпорациями IT индустрии такими, как IBM, Novell, Oracle, Microsoft. ...».

Казалось бы такой представительный форум должен был решить все проблемные вопросы, но этого не произошло. Поэтому стали создаваться различные (нестандартизованные) варианты *middleware*-систем, реализующие потребности разработчиков SOA.

1.3.3 Сервисные шины предприятий

Поскольку термины *middleware* и «*Службы промежуточного уровня*» достаточно абстрактны, а консорциум OASIS не предлагал им достойного эквивалента,

то по аналогии с системной шиной ЭВМ был предложен термин [24]: «... **Сервисная шина предприятия** (англ. E*nterprise service bus*, *ESB*) — связующее программное обеспечение, обеспечивающее централизованный и унифицированный событийно-ориентированный обмен сообщениями между различными информационными системами на принципах сервис-ориентированной архитектуры. Понятие введено в начале 2000-х годов специалистами подразделения Progress Software — Sonic, разрабатывавшими MOM-продукт SonicMQ. … Основной принцип сервисной шины — концентрация обмена сообщениями между различными системами через единую точку, в которой, при необходимости, обеспечивается транзакционный контроль, преобразование данных, сохранность сообщений. … По состоянию на вторую половину 2011 года Forrester относит к «волне лидеров» следующие продукты со значительным присутствием на рынке: WebMethods ESB (Software AG, семейство продуктов WebMethods, поглощённой одноимённой компанией), ActiveMatrix Service Bus (Tibco), Oracle Service Bus (Oracle, семейство Fusion Midlleware), WebSphere Message Broker (IBM, семейство WebSphere). Среди продуктов с менее значительным присутствием на рынке упомянуты Sonic ESB (Progress Software), WebSphere ESB и ESBRE (IBM), FuseSource, с незначительным — MuleESB, WSO2, JBoss ESB (Red Hat). ...».

Таким образом, смысловое содержание понятия ***ESB*** является своеобразным «зонтичным брендом» специализированного программного обеспечения для создания *сетевых инфраструктур*, которые интегрируют *потребителей* и *поставщиков* различных сервисов. Такие инфраструктуры скрывают детали реализации сетей, определённых моделью OSI, обеспечивая доступность только протоколов прикладного уровня, таким как: SMTP, FTP, HTTP, HTTPS и другим.

1.4 Виртуальные системы

Понятие «*Виртуальные системы*», которое включает в себя *метасемантику*, присутствующую в изучаемой предметной области, но обычно употребляемую в виде прилагательного, например: [25] «... **Виртуальная машина** (VM, от англ. *Virtual Machine*) — программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы (target — целевая, или гостевая платформа) и исполняющая программы для target-платформы на host-платформе (host — хост-платформа, платформа-хозяин) или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы (см.: песочница); также спецификация некоторой вычислительной среды (например: «виртуальная машина языка программирования Си»). Виртуальная машина исполняет некоторый машиннонезависимый код (например, байт-код, шитый код, p-код) или машинный код реального процессора. Помимо процессора, ВМ может эмулировать работу как отдельных компонентов аппаратного обеспечения, так и целого реального компьютера (включая BIOS, оператив-

ную память, жёсткий диск и другие периферийные устройства). В последнем случае в ВМ, как и на реальный компьютер, можно устанавливать операционные системы (например, Windows можно запускать в виртуальной машине под Linux или наоборот). На одном компьютере может функционировать несколько виртуальных машин (это может использоваться для имитации нескольких серверов на одном реальном сервере с целью оптимизации использования ресурсов сервера)».

Следуя заявленной общей традиции, уточним применительно к тематике дисциплины метасемантику следующих концепций:

- виртуальные машины;
- виртуализация вычислительных комплексов на уровне ОС;
- виртуализация ПО на уровне языка (программирования);
- виртуальная машина языка Java.

1.4.1 Виртуальные машины

Словосочетание «*Виртуальная машина*» является наиболее широко используемым понятием. Слово «*машина*» подразумевает некоторый механизм, который обычно имеет аппаратное исполнение и эффективно выполняет некоторый набор действий, имеющих некоторый результат. Например, процессор выполняет последовательность команд над заданным набором данных и выдаёт результат произведённых вычислений. Соответственно, прилагательное «*виртуальная*» задаёт метасемантику подмены первоначального объекта «*машина*» на другой объект «*машина-2*», которая конкретизируется в двух противоположных аспектах:

- *негативный аспект* конкретизации связан не только с фактом подмены, что ассоциируется с фактом обмана, но и с более низким качеством объекта подмены; например, замена аппаратной реализации машины программой обычно *снижает скорость вычислений*;
- *позитивный аспект* конкретизации варьируется в более широких пределах; простейший вариант обоснования — отсутствие необходимых аппаратных средств, поэтому виртуализация посредством программной реализации позволяет получить необходимый результат; более сложные варианты обоснования могут быть построены на очень сомнительных аргументах, например, идея реализации вычислительных машин на основе громоздких и ненадёжных электронно-ламповых схем взамен компактных и надёжных электромеханических переключателей (реле).

Важнейшим событием для нашей предметной области стал переход от гарвардской архитектуры ЭВМ к архитектуре фон-Неймана, поскольку в ней появилось понятие адреса как элемента конструктива не только аппаратных средств, но и программного обеспечения. Память становится той универсальной средой, где

размещаются программы и данные, а местоположение программ и данных обеспечивается набором систем адресации, с помощью которых и организуется любой вычислительный процесс. Таким образом, согласно модели СОД мы приходим к модели ЭВМ как совокупности аппаратных средств и программного обеспечения ОС, реализующего конкретную виртуальную машину — ЭВМ. Теперь, рассматривая ЭВМ на уровне ОС (например, Linux), мы выделяем:

- *виртуальную машину ядра* (kernel), создающую для всего пользовательского пространства: набор виртуальных устройств, например, виртуальные терминалы; виртуальную память на уровне страниц и сегментов; виртуальную файловую систему;
- *виртуальное адресное пространство* процессов размером 4 ГБ, первые три ГБ которого принадлежат пользовательскому пространству, а 4-й ГБ - пространству ядра;
- *разделяемые между процессами библиотеки и память*, которые каждый процесс использует в своём (виртуальном) адресном пространстве.

Главная особенность такой предметной интерпретации - «Виртуальная машина» является целостной системой в плане нормального функционирования всех её компонент. Вся система ориентирована на организацию эффективного вычислительного процесса. Выход из строя любой компоненты существенно влияет на позитивное состояние системы, что банально устраняется ее ремонтом.

1.4.2 Виртуализация вычислительных комплексов на уровне ОС

По определению СОД, вычислительный комплекс является базовой основой вычислительной системы и содержит ОС, которая, реализуя виртуальную машину, позволяет рассматривать комплекс как отдельную ЭВМ. Поскольку комплексы создаются с целью повышения эффективности работы ВС, то соответствующие проектные решения реализуются посредством:

- *виртуализации* многоядерных процессоров в виде отдельных вычислителей, объединённых общей памятью и построением систем симметричного мультипроцессирования (SMP-систем);
- *виртуализации* многомашинных систем средствами программного обеспечения Message Passing Interface (MPI, Интерфейс передачи сообщений), что также обеспечивает построение систем средствами ПО ОС.

Указанные средства могут иметь различную вычислительную мощность и архитектуру различной сложности, например, построенные на гомогенных архитектурах описанных Эндрю Таненбаумом [3], они, по сути, остаются в классе сосредоточенных систем.

1.4.2 Виртуализация ПО на уровне языка

Использование ЭВМ, построенных на архитектуре фон-Неймана, породило большое количество систем адресации, которые необходимо использовать как при проектировании, так при создании и эксплуатации различных систем. Большинство возникающих при этом проблем более или менее успешно решаются с помощью языков программирования. Инструментом решения указанных проблем является ***именование*** и ***типизация имён*** языковых конструкций, например:

- языки *ассемблеров*, посредством мнемоник команд, скрывают многие особенности методов адресации данных и размеров самих команд, перекладывая эти проблемы на компиляторы и линковщики программ;
- язык *Fortran*, ориентированный исключительно на вычисления, полностью освобождается от прямой адресации, но вынужден поддерживать типизацию и работу с массивами данных;
- в языке *C* — наоборот, вводится тип указателя, необходимый для решения системных задач и написания программ максимально заменяющих языки ассемблеров;
- языки ООП маскируют адресацию в конструкции объекта, инкапсулируя данные и методы в одном типе;
- языки, поддерживающие *динамическую типизацию*, например, Python, PHP, Perl, JavaScript и другие максимально скрывают саму типизацию, перекладывая решение вопросов адресации на интерпретаторы этих языков.

Здесь следует высказать ряд важных замечаний.

Языки обеспечивающие создание виртуальных машин ориентированы на потребности в собственной системе адресации. Они не охватывают адресацию приложений, выходящих за рамки виртуальной машины ОС. Например, типичная реализация программы в архитектуре клиент-сервер на уровне стека протоколов TCP/IP требует от клиентской части программы знания адреса и порта сервера, а если адрес сервера задан его доменным именем, то необходимо указать и адрес сервера DNS.

1.4.3 Виртуальная машина языка Java

Самым ярким событием середины 90-х годов можно считать появление виртуальной машины Java [26]: «... **Java Virtual Machine** (сокращённо **Java VM**, **JVM**) — виртуальная машина Java — основная часть исполняющей системы Java, так называемой *Java Runtime Environment* (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM может также использоваться для выполнения программ, написанных на других языках программирования. Например, исходный код на языке Ada может быть откомпилирован в байт-код Java, который затем мо-

жет выполниться с помощью JVM. JVM является ключевым компонентом платформы Java. Так как виртуальные машины Java доступны для многих аппаратных и программных платформ, Java может рассматриваться и как связующее программное обеспечение, и как самостоятельная платформа. Использование одного байт-кода для многих платформ позволяет описать Java как «скомпилировано однажды, запускается везде» (compile once, run anywhere). Виртуальные машины Java обычно содержат Интерпретатор байт-кода, однако, для повышения производительности во многих машинах также применяется JIT-компиляция часто исполняемых фрагментов байт-кода в машинный код. ...».

Хотя сам язык Java является сильнотипизированным объектно-ориентированным языком программирования [27], именно благодаря его виртуальной машине (JVM) удалось в кратчайшие сроки реализовать многие современные технологии объектных сетевых систем, web-технологии и сервис-ориентированные технологии. Дело в том, что в отличие от других скриптовых языков, исходный код Java-программы подвергается компиляции и может быть использован без последующего синтаксического анализа. Это значительно ускоряет выполнение Java-программ и позволяет выполнять кэширование серверных приложений.

Уникальными особенностями обладает и байт-код Java. Благодаря разработчикам, в JVM обеспечивается [27] «... полная независимость байт-кода от ОС и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединение с другим компьютером), вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят снижение производительности. Ряд усовершенствований несколько увеличил скорость выполнения программ на языке Java:

- применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,
- обширное использование платформо-ориентированного кода (native-код) в стандартных библиотеках,
- аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами архитектуры ARM).

По данным сайта ***shootout.alioth.debian.org***, для семи разных задач время выполнения на Java составляет в среднем в полтора — два раза больше, чем для C/C++, в некоторых случаях Java быстрее, а в отдельных случаях в 7 раз медленнее. С другой стороны, для большинства из них потребление памяти Java-машиной было в 10—30 раз больше, чем программой на C/C++. Также примечательно исследование, проведённое компанией Google, согласно которому отмечается су-

щественно более низкая производительность и большее потребление памяти в тестовых примерах на Java в сравнении с аналогичными программами на C++.

Идеи, заложенные в концепцию и различные реализации среды виртуальной машины Java, вдохновили множество энтузиастов на расширение перечня языков, которые могли бы быть использованы для создания программ, исполняемых на виртуальной машине. Эти идеи нашли также выражение в спецификации общеязыковой инфраструктуры CLI, заложенной в основу платформы .NET компанией Microsoft. ...».

Наличие такого большого количества замечательных качеств виртуальной машины Java опирается на серьёзную интеллектуальную работу сотрудников американской компании Sun Microsystems, которая совместно с нынешней правопреемницей ее собственности — корпорацией Oracle (*Oracle Corporation*), обеспечивших грамотную пакетную организацию программного обеспечения этой платформы. Благодаря проведённой работе, пакетная организация ПО Java может служить образцом стандартизации программного обеспечения для любых систем и входить во все учебники по программированию.

Отдельной статьи заслуживает описание достижений платформы JVM применительно к web-технологиям. Особенно следует выделить технологию апплетов, которая появилась в первой версии языка Java уже в 1995 году. Именно апплеты обеспечили запуск полноценных приложений в среде web-браузеров. В дальнейшем, подобные технологии распространились и на web-сервера (сервлеты, 1997 год), появилась технология JSP-страниц, реализованная в сервере Apache Tomcat, и так далее.

Таким образом, практическое применение технологических достижений виртуальной машины Java является важной (вспомогательной) частью предметной области изучаемой дисциплины.

1.5 Итоги теоретических построений

Общим итогом теоретических построений данной главы стало определение границ и понятийного содержания предметной области дисциплины, обозначенной как «*Распределенные вычислительные сети*» (РВ-сети). Необходимость более чёткого выделения таких границ вызвана разнообразием различных понятийных трактовок, присутствующих не только в широких публичных изданиях, но и в учебной литературе [1-5]. Важность более строгого выделения определения понятийного содержания рассматриваемой предметной области обусловлена учебным характером данной работы. Опираясь на классификационную схему Ларионова, нами проведён теоретический анализ модели СОД и выполнено уточнение таких понятий как «*Сосредоточенные системы*» и «*Распределенные системы*».

Сосредоточенные системы группируют «*классическое направление*» развития цифровых технологий, основанных на вычислительной мощности программно-аппаратных средств и локализованных во вполне обозримых границах. Для таких систем подключение и взаимодействие с другими подобными системами является допустимым явлением, но не главной характеристикой.

Непосредственное последующее толкование таких систем будет рассматриваться в трёх аспектах:

- **ЭВМ** — согласованный в функциональном назначении аппаратный конструктив, дополненный базовым ПО ОС до конкретной виртуальной машины; в последующем, такая машина рассматривается как элемент более сложных систем;
- **Вычислительные системы** — все прикладные аспекты программного обеспечения, реализованные поверх виртуальной машины ОС; в последующем, такие системы также могут рассматривается как элементы более сложных систем;
- **Вычислительные комплексы** — системы аппаратных средств, ориентированные на повышение вычислительной мощности и надёжности всего конструктива, а также — дополненные соответствующей виртуальной машиной ОС; в последующем, такой комплекс также может рассматривается как отдельная ЭВМ или составляющая часть вычислительной системы.

Распределенные системы группируют «*современное направление*» развития цифровых технологий, основной характеристикой которых является интеграция *сосредоточенных систем* (ЭВМ, ВС) на основе сетевых технологий. Важной особенностью таких систем является их *свойство несводимости* к моделям сосредоточенных систем. Обычно, это определяется самой природой интегрируемых приложений, например, программное обеспечение клиента банка должно быть отделено от ПО самого банка в силу нормативных требований к проведению финансовых операций.

Что же касается декомпозиции распределенных систем, представленной на рисунке 1.2, то являясь по сути правильной, она морально устарела с точки зрения современных достижений. Действительно:

- ***Системы телеобработки***, детализированные на рисунке 1.3, с успехом и без ущерба для производительности и стоимости их эксплуатации, могут быть реализованы на основе современных сетевых технологий;
- ***Вычислительные сети***, детализированные на рисунке 1.4, также отражают устаревшие технологические представления, поскольку *базовая сеть передачи данных* (СПД), построенная на *узлах связи* (УС), охватывает только нижние три уровня модели OSI (см. рисунок 1.7); современные же сетевые системы, как это показано в пунктах 1.2 и 1.3, создаются на основе протоколов прикладного уровня, что охватывает все уровни модели OSI.

Таким образом, суммируя все перечисленные выше доводы, *распределенные системы* будем детализировать одной моделью *РВ-сетей*, показанной на рисунке 1.12.

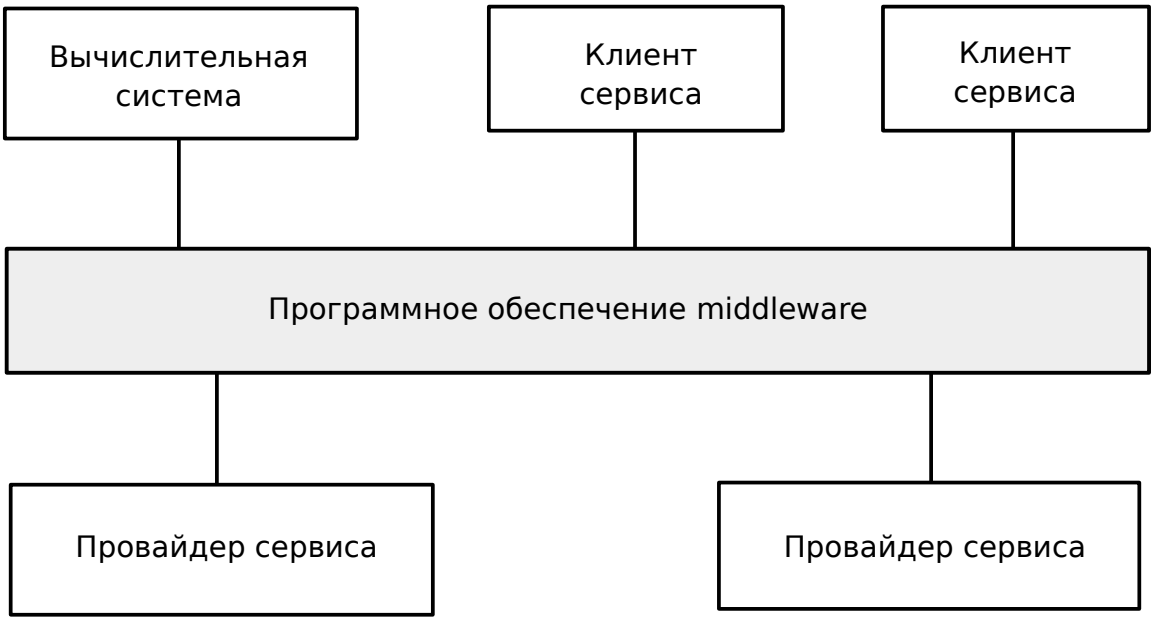

Рисунок 1.12 — Модель РВ-сети

Представленная модель РВ-сети будет конкретизироваться в зависимости от тематики рассматриваемых вопросов. Соответственно, «*Программное обеспечение middleware*» будет обозначать модель OSI (обычно стек протоколов TCP/IP), а окружающие прикладные объекты будут отображать *сосредоточенные системы*.

Вопросы для самопроверки

1. Какие уровни детализации допускает понятие «*Вычислительная машина*»?
2. На какие две группы делит системы классификация СОД?
3. Что подразумевает понятие «*Сосредоточенные системы*»?
4. Чем отличаются понятия «*Вычислительные системы*» и «*Вычислительные комплексы*»?
5. На какие четыре части делит архитектуры ЭВМ таксономия Флинна?
6. Что входит в состав классического понятия «*Распределенные системы*»?
7. Какие элементы включает в себя понятие «*Вычислительная сеть*»?
8. Что означает понятие «*Служба промежуточного уровня*»?
9. Что означает понятие «*Распределенная вычислительная среда*»?
10. Что означает понятие «*Remote Procedure Call*»?
11. Что такое - «*Client stub*» и «*Server stub*», а также в чем их различие?
12. Каково назначение и область применения аббревиатуры — IDL?
13. В чем состоит суть технологии CORBA?
14. В чем отличие технологии RMI от технологии CORBA?
15. На какие общие услуги подразделяется парадигма «*Сервис-ориентированные технологии*»?
16. В чем состоит отличие понятий «*Функции*» и «*Сервисы*»?
17. Какая практическая интерпретация применима к понятию «*Middleware*»?
18. Что означает понятие «*Виртуальная машина*» применительно к программным системам?
19. Что означает понятие «*Java Virtual Machine*»?
20. Что такое - «*Промышленная шина предприятия*»?