Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

В.Г. Резник

РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

Учебное пособие

Тема 5. Сервис-ориентированные архитектуры

Томск
2020

УДК 004.75

Резник, Виталий Григорьевич

Распределенные вычислительные сети. Учебное пособие. Тема 5. Сервис-ориентированные архитектуры / В.Г. Резник. – Томск : Томск. гос. ун-т систем упр. и радиоэлектроники, 2019. – 16 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

5 Тема 5. Сервис-ориентированные архитектуры

В подразделе 1.3 первой главы, уже была дана достаточно ёмкая характеристика понятию «*Сервис-ориентированные технологии*». В частности, на рисунке 1.11 была показана обобщённая схема соотношения *объектно-ориентированного* и *сервисного* подходов в создании прикладных систем. Дополнительно, было дано разъяснение понятию «*Сервис-ориентированная архитектура*» и была отмечена значимость систем типа ***Middleware***, использующих модель ***SOA*** и обеспечивающих взаимодействие распределенных приложений на основе протокола ***SOAP***.

В общем случае, тема сервисно-ориентированных технологий, далее — **СОТ**, требует отдельного учебного пособия, раскрывающего все многообразие подходов и методов реализованных в них. В данной главе мы отметим, что СОТ является дальнейшим продолжением и развитием ранее изученной концепции «*Объектные распределенные системы*», а также основным трендом современного развития компьютерных технологий.

Учебный материал данной главы ограничен теоретическим описанием двух аспектов СОТ: общей концепции SOA и частных подходов к реализации сервисных технологий.

Общая концепция SOA охватывает теоретические аспекты рассматриваемой технологии, которые детализируются в трёх направлениях:

- в первом направлении рассматриваются *вопросы связывания* распределённых программных систем;
- во втором направлении обсуждаются Web-сервисы, которые условно подразделяются на первое и второе поколения;
- в третьем направлении рассматриваются три брокерные архитектуры, которые в настоящее время используются Web-сервисами.

В подразделе, посвящённом частным подходам к реализации сервисных технологий, основное внимание уделяется следующим направлениям:

- технология одноранговых сетей становится все более популярной в различных общественных сервисах ориентированных на культуру, личные контакты и индустрию развлечений; первичным здесь является отказ от централизованных средств учёта и управления адресами ресурсов Интернет;
- технология GRID, которая ориентирована на создание некоторого большого «*виртуального суперкомпьютера*», обслуживающего множество организаций и научных коллективов;
- технология облачных вычислений, ориентированных на учёт используемых ресурсов ЭВМ и обеспечивающих построение как коммерческих, так и общественных распределенных сервисных ресурсов.

В целом, обсуждаемые технологии рассматриваются в теоретическом плане и не привязываются к конкретным языкам программирования или конкретным фреймворкам.

5.1 Концепция SOA

Суммируя уже изученный материал подраздела 1.3, можно утверждать, что ***SOA*** или «Сервис-ориентированная архитектура» - это парадигма организации и использования распределенных возможностей приложений, которые принадлежат различным владельцам сервисов.

SOA — это парадигма, которая расширяет архитектуру объектных распределённых систем, выделяя модель независимого компонента. Этот компонент не обязан присутствовать на каждой стороне участников распределенного взаимодействия, как это должен делать класс, реализующий используемый удалённый объект.

Базовыми составляющими SOA, как модели, являются: *сервисные компоненты* (сервисы), *интерфейсы сервисов*, *соединители сервисов* и *механизмы обнаружения сервисов*.

Сервисные компоненты (или сервисы) описываются программными компонентами, которые обеспечивают прозрачную сетевую адресацию.

Интерфейс сервиса обеспечивает описание возможностей и качества предоставляемых сервисом услуг. В таком описании определяется формат сообщений, используемых для обмена информацией, а также входные и выходные параметры методов, поддерживаемых сервисным компонентом.

Соединитель сервисов — это транспорт, обеспечивающий обмен информацией между отдельными сервисными компонентами.

Механизмы обнаружения сервисов предназначены для поиска сервисных компонентов, обеспечивающих требуемую функциональность сервиса. Среди всего множества вариантов, обеспечивающих обнаружения сервисов, выделяются две основные категории: системы *динамического* и *статического* обнаружения.

Статические системы обнаружения сервисов, например UDDI, ориентированы на хранение информации о сервисах в редко изменяющихся системах.

Динамические системы обнаружения сервисов ориентированы на системы, в которых допустимо частое появление и исчезновение сервисных компонентов.

Исторически, взаимодействие между поставщиками и потребителями сервисов основывалось на протоколе **SOAP** [22], который, в свою очередь, опирается на язык **XML** [50]. Сам протокол SOAP считается независимым от языка и платформы, хотя само взаимодействие проходит путь ***XML->HTTP/HTML->TCP/IP***. Такая ситуация привела к понятию «*Веб-сервисы*», которое и рассматривается в большинстве литературных источников. Подобную интерпретацию будем использовать и мы, хотя сама теория не ограничивает базу реализации сервисов, допуская применение моделей CORBA, RMI и другие технологии сетевого взаимодействия. В частности, большинство современных веб-сервисов для передачи своих сообщений применяют протокол HTTP, поскольку он экономит трафик передачи данных по сравнению с простой передачей текста на языке XML.

5.1.1 Связывание распределенных программных систем

Одной их характеристик РВ-сетей является ***связанность*** их сервисов. По этому признаку распределенные программные системы подразделяются на два типа: «*Сильносвязанные системы*» (Strong coupling) и «*Слабосвязанные системы*» (Loose coupling).

Сильная связанность возникает при использовании объектных распределенных сервисов, когда зависимый класс содержит ссылку на конкретный класс, предоставляющий сервис.

Слабая связанность возникает, когда зависимый класс содержит ссылку на интерфейс, который может быть реализован одним или многими различными классами.

Очевидно, что использование концепции слабосвязанных программных систем уменьшает количество зависимостей между сервисными компонентами. Это, в свою очередь, уменьшает объем возможных последствий, порождённых сбоями или модификациями распределенных систем. В таблице 5.1, заимствованной из источника [5], приведено сравнение ряда общих характеристик слабосвязанных и сильносвязанных систем.

Таблица 5.1 — Сравнение слабосвязанных и сильносвязанных систем

	Сильносвязанные системы	***Слабосвязанные системы***
Физические соединения	Точка-точка	Через посредника
Стиль взаимодействий	Синхронные	Асинхронные
Модель данных	Общие сложные типы	Простые типы
Связывание	Статическое	Динамическое
Платформа	Сильная зависимость от базовой платформы платформы	Независимость от базовой платформы
Развертывание	Одновременное	Постепенное

Общий вывод, следующий из анализа связанности, проектируемых систем, состоит в том, что:

- традиционный подход разработки распределенных приложений обычно значительно усложняет создание и сопровождение РВ-сетей;
- Веб-сервисы, используя слабую связанность сервисов, позволяют значительно упростить координацию распределенных систем и их реконфигурацию.

5.1.2 Web-сервисы первого и второго поколений

Первое поколение веб-сервисов (до 2007 года) опиралось на парадигму XML веб-служб, позволяющих создавать независимые масштабируемые слабосвязанные приложения. Для этого использовалось три основных стандарта: WSDL, UDDI и SOAP, образующие так называемый «*Треугольник SOA*», показанный на рисунке 5.1.

Рисунок 5.1 - Взаимодействие между клиентом и поставщиком веб-сервиса (заимствовано из источника [5])

Согласно общим представлениям [59]: «... **UDDI** (англ. *Universal Description Discovery & Integration*, ...) — инструмент для расположения описаний веб-сервисов (WSDL) для последующего их поиска другими организациями и интеграции в свои системы. UDDI это кроссплатформенное программное обеспечение, основанное на XML. UDDI является открытым проектом, спонсируемым OASIS, который позволяет организациям публиковать описания веб-сервисов (WSDL) для последующего их поиска другими организациями и интеграции в свои системы, а также определять, как сервисы или приложения взаимодействуют через Internet. UDDI был первоначально предложен в качестве основного веб-сервис стандарта. Он предназначен для опроса SOAP сообщениями и для обеспечения доступа к ... документам, описывающим привязки протоколов и форматов сообщений, необходимых для взаимодействия с веб-услугами, перечисленными в его каталоге. ...».

Соответственно [60]: «... **WSDL** (англ. ***W**eb **S**ervices **D**escription **L**anguage*) — язык описания веб-сервисов и доступа к ним, основанный на языке XML. Последняя официальная спецификация на момент написания статьи версия 2.0 (WSDL Version 2.0 от 26 июня 2007 года), которая имеет статус рекомендации, и версия 1.1 (WSDL Version 1.1 от 15 марта 2001 года), которая имеет статус заметки (note). ...».

Протокол SOAP [22] обеспечивает непосредственную передачу сообщений между клиентом сервиса поставщиком сервиса. Сообщения передаются XML-конвертами (***Envelope***), а общая структура которых показана на рисунке 5.2, где:

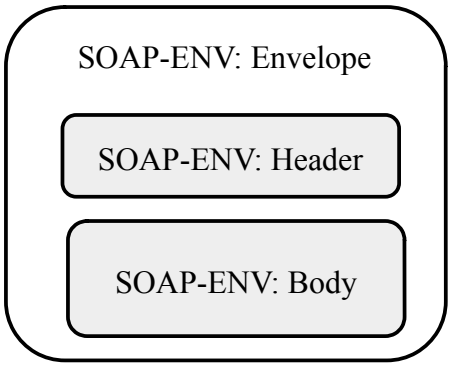

Рисунок 5.2 — Структура сообщения протокола SOAP

- **Envelope** – Корневой элемент, определяющий сообщение и пространство имен, использованное в документе.
- **Header** – Содержит атрибуты сообщения, такие как: информация о безопас-ности или о сетевой маршрутизации.
- **Body** – Содержит сообщение, которым обмениваются приложения.
- **Fault** – Необязательный элемент, который предоставляет информацию об ошибках, произошедших при обработке сообщений.

Качественные характеристики веб-сервисов первого поколения состоят из следующих положений:

- контент сервисов *формируется поставщиками сервисов*;
- *отсутствуют единые стандарты* протоколов авторизации и аутентифика-ции пользователей, что требовало от поставщиков сервиса проводить самос-тоятельные разработки;
- *отсутствует понятие состояния* сервиса и после обращения клиента к серверу, его состояние на сервере не сохраняется.

Начиная с 2004 года начинают публиковаться и внедряться различные стан-дарты, повышение качество работы веб-сервисов. К ним относятся:

- ***WS-Security*** – обеспечение безопасности веб-сервисов;
- ***WS-Addressing*** – маршрутизация и адресация SOAP-сообщений;
- ***WSRF, WS-Notification*** – работа с состоянием веб-сервисов.

Внедрение перечисленных стандартов позволило многим разработчикам говорить о «**Втором поколении веб-сервисов**», поскольку теперь:

- контент сервисов *формируется пользователями сервисов*;
- *используются единые стандарты* протоколов авторизации и аутентификации пользователей;
- в разработках сервисов начинает использоваться «*Состояние сервиса*».

5.1.3 Брокерные архитектуры Web-сервисов

Излишне напоминать, что «*Треугольник SOA*», показанный на рисунке 5.1, демонстрирует взаимодействие клиента и сервера с помощью посредника (брокера). В целом, выделяется три разновидности такого взаимодействия: *прямой вызов* через UDDI, *синхронный вызов* через посредника и а*синхронный вызов* через посредника. Рассмотрим каждый из них.

Прямой вызов через UDDI предполагает развёртывание «*Службы адресов*» по хорошо известному потребителю адресу. Потребитель сервиса может использовать UDDI для поиска нужных ему Web-служб (*Провайдеров сервиса*), демонстрируется рисунком 5.3.

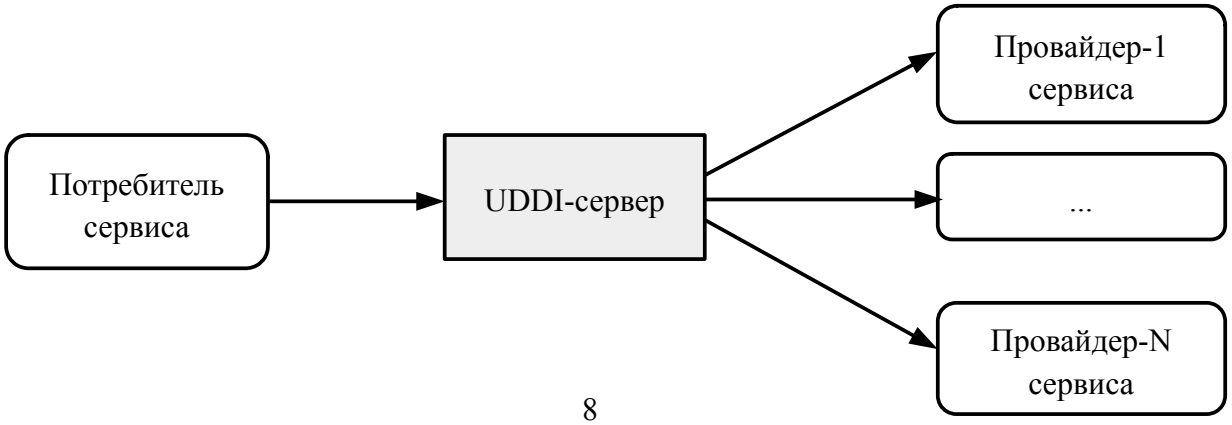

Рисунок 5.3 — Прямой вызов через UDDI

Получив адрес, потребитель самостоятельно обращается к провайдеру сервиса и ведёт с ним собственный диалог, а такой подход имеет ряд недостатков, которые можно сформулировать следующим образом:

- Потребитель *должен знать* ***URI оконечной точки*** провайдера для вызова службы. Он использует UDDI как каталог для поиска этого URI.
- Если имеется несколько провайдеров, UDDI содержит несколько URI, и потребитель должен выбрать один из них.
- Если провайдер сервиса меняет URI оконечной точки, он должен повторно зарегистрироваться на сервере UDDI для того, чтобы UDDI хранил новый URI. Потребитель должен повторно запросить UDDI для получения нового URI.

В сущности, использование прямого вызова UDDI означает, что каждый раз, когда потребитель хочет вызвать службу, он должен запросить URI оконечных точек в службе UDDI. Такой подход также вынуждает потребителя самостоятельно оценивать качество провайдера каким-либо способом.

Синхронный вызов через посредника предполагает использование «*интеллектуального брокера*», который способен самостоятельно проводить оценку провайдеров сервиса на предмет их присутствия в сети, качества обслуживания пользователей и загруженности запросами.

Потребитель вызывает прокси-службу такого брокера, который, в свою очередь, оценивает загруженность провайдеров, выбирает одного из них и передаёт UDDI. UDDI возвращает только один URI, и потребитель не должен делать выбор.

Потребитель даже может и не знать, что оконечная точка является прокси-службой. Он знает только о том, что может использовать этот URI для вызова Web-службы.

Взаимодействие потребителя сервиса, UDDI-службы и провайдеров такого сервиса показано на рисунке 5.4.

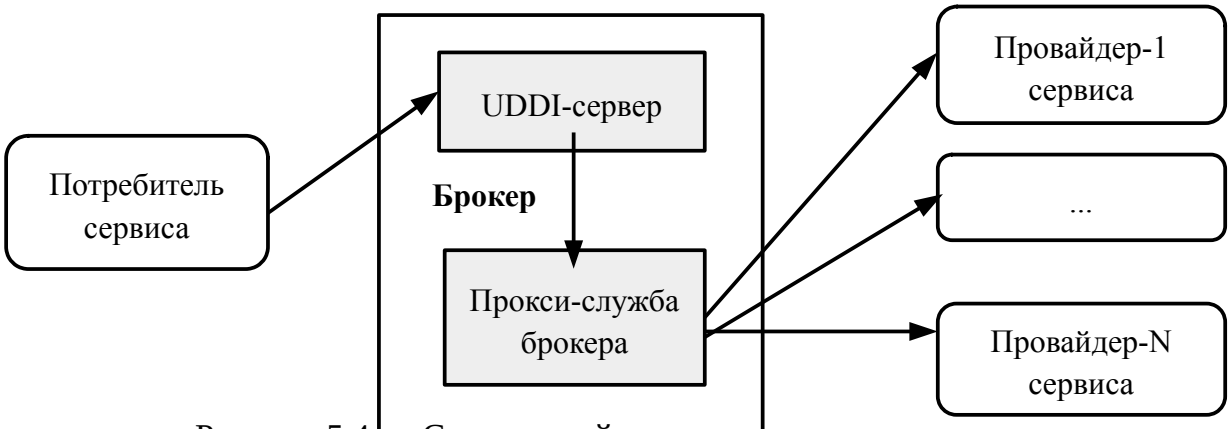

Рисунок 5.4 — Синхронный вызов через посредника

В РВ-сети слабосвязанных приложений использование синхронного вызова, также имеет ряд недостатков:

- Потребитель должен ждать окончания выполнения службы провайдера. На это время поток исполнения клиента должен быть заблокирован.
- Если служба провайдера выполняется длительное время, то потребитель может прекратить ожидание ответа.
- Имеются случаи, когда потребитель выполняет запрос, но не может ждать ответ провайдера сервиса.
- Если у потребителя возникает аварийная ситуация во время блокировки его работы, то ответ будет потерян и вызов сервиса нужно будет повторить.

Асинхронный вызов через посредника решает многие проблемы, связанные как с реализацией брокера, так и с организацией взаимодействия потребителя (клиентское приложение) провайдера (поставщика сервиса).

Брокер, используя свои оценки провайдеров сервиса, разрешает или ограничивает им доступ к двум очередям запросов и ответов. Поэтому, провайдеры сервисов получают доступ к запросам клиентов, конкурируя между собой и с учётом своих возможностей обслуживать клиента, что и показано на рисунке 5.5.

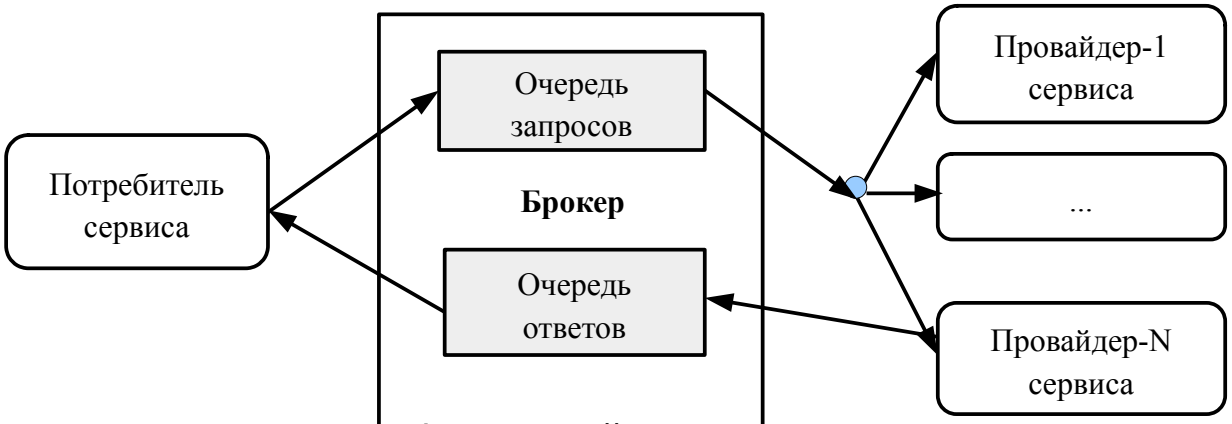

Рисунок 5.5 — Асинхронный вызов через посредника

При таком подходе, потребитель использует *два асинхронных канала* связи с брокером. По одному каналу передаёт запрос брокеру, который ставит его в свою очередь запросов. По другому каналу, потребитель асинхронно забирает ответ из очереди ответов брокера.

Преимущества асинхронного подхода является наиболее предпочтительным для создания РВ-сетей:

- Потребитель *не должен блокировать* свою работу при ожидании ответа и может в это время выполнять другую работу, поэтому потребитель намного менее чувствителен к продолжительности работы службы у провайдера.
- Брокер, предоставляющий возможность потребителю вызывать Web-службу асинхронно, реализуется при помощи технологии хорошо развитых систем обмена сообщениями.
- Пара очередей сообщений выступает как один адрес, использующийся любым потребителем сервиса независимо от количества обслуживающих их провайдеров.

Завершая краткий теоретический обзор концепции SOA, отметим, что «С*ервис-ориентированные технологии*» не ограничиваются только веб-службами. Просто, веб-службы являются наиболее «*модным*» современным направлением, который пока ещё не достиг пика своих возможностей. Тем не менее, мы должны помнить, что слабо связанные приложения и сервисы, не

сохраняющие состояние запросов клиентов, больше соответствуют публичной сфере использования РВ-сетей. Большинство же приложений требуют учёта поставщиками сервисов своих состояний, что значительно усложняет как их разработку, так и сопровождение.

5.2 Частные подходы к реализации сервисных технологий

В этом подразделе приведён краткий обзор трёх технологических направлений, которые напрямую не ассоциируют себя с веб-сервисами, но в практическом плане являются реализациями сервисных технологий. К ним относятся: *технологии одноранговых сетей*, *технологии Grid* и *облачные вычисления*. Более подробный обзор можно найти, например, в источнике [5].

5.2.1 Технология одноранговых сетей

Технология одноранговых или P2P сетей обеспечивает взаимодействие приложений РВ-сетей на базе принципа децентрализации, когда разделение вычислительных ресурсов и сервисов производится напрямую посредством прямого взаимодействия между участниками сети друг с другом. В этом отношении «*Одноранговые вычислительные сети*» являются противоположностью «*строгим*» клиент-серверным архитектурам, таким как CORBA, RMI, RPC и другими. Согласно [61]: «... **Одноранговая, децентрализованная,** или **пиринговая** (англ. *peer-to-peer, P2P* — равный к равному) **сеть** — оверлейная компьютерная сеть, основанная на равноправии участников. Часто в такой сети отсутствуют выделенные серверы, а каждый узел (peer) является как клиентом, так и выполняет функции сервера. В отличие от архитектуры клиент-сервера, такая организация позволяет сохранять работоспособность сети при любом количестве и любом сочетании доступных узлов. Участниками сети являются все пиры. ...». Считается, что положительные качества P2P-сетей определяются свойствами:

- отсутствия зависимости от централизованных сервисов и ресурсов;
- возможностью системы пережить серьёзное изменение в структуре сети;
- высокой масштабируемостью модели одноранговых вычислений.

Основным элементом является **Пир** (***Peer***), который считается фундаментальным составляющим блоком, который может быть двух типов:

- ***простым пиром***, обеспечивающим работу конечного пользователя, предоставляя ему сервисы других ***пиров*** и обеспечивая предоставление ресурсов другим участникам сети;
- ***роутером***, которой реализует механизм взаимодействия между ***пирами***, отделёнными от сети брандмауэрами или NAT-системами.

Сам пир:

- имеет уникальный идентификатор;
- принадлежит одной или нескольким группам;
- может взаимодействовать пирами как в своей, так и в других группах.

Согласно источнику [5]: «... **Группа пиров** – это набор пиров, сформированный для решения общей задачи или достижения общей цели. Группы пиров могут предоставлять членам своей группы такие наборы сервисов, которые недоступны пирам, входящим в другие группы. **Сервисы** – это функциональные возможности, которые может привлекать отдельный пир для полноценной работы с удалёнными пирами. В качестве примера сервисов, которые может предоставлять отдельный пир можно указать сервисы передачи файлов, предоставления информации о статусе, проведения вычислений и др. **Сервисы пира** – это такие сервисы, которые может предоставить конкретный узел P2P. Каждый узел в сети P2P предоставляет определённые функциональные возможности, которыми могут воспользоваться другие узлы. Эти возможности зависят от конкретного узла и доступны только тогда, когда узел подключён к сети. Как только узел отключается, его сервисы становятся недоступны. **Сервисы группы** – это функциональ-

ные возможности, предоставляемые группой входящим в неё узлам. Возможности могут предоставляться несколькими узлами в группе, для обеспечения избыточного доступа к этим возможностям. Как только к группе подключается узел, обеспечивающий необходимый сервис, он становиться доступным для всей группы. ...».

P2P-сети имеют свои протоколы. Один из таких протоколов реализовала компания Sun Microsystems в рамках проекта JXTA [62]: «... **JXTA** (**Juxtapose**) — спецификации протоколов по обслуживанию P2P-сетей для обмена данными различного типа. Проект был запущен корпорацией Sun Microsystems в 2001 для решения проблем, стоящих на пути развития пиринговых сетей. JXTA использует открытые протоколы XML и может быть реализован на любом современном языке программирования. В настоящий момент JXTA реализован на J2SE, J2ME и Си/Си++/Си#. JXTA распространяется под лицензией, производной от Apache License. ...».

Несмотря на значительные преимущества децентрализованного взаимодействия участников РВ-сетей, выделяются также недостатки технологии одноранговых сетей [5]: «...

- в одноранговых сетях не может быть обеспечено гарантированное качество обслуживания: любой узел, предоставляющий те или иные сервисы, может быть отключён от сети в любой момент;
- индивидуальные технические характеристики узла могут не позволить полностью использовать ресурсы P2P сети (каждый из узлов обладает индивидуальными техническими характеристиками что, возможно, будет ограничивать его роль в P2P-сети и не позволять полностью использовать ее ресурсы: низкий рейтинг в torrent-сетях, LowID в eDonkey могут значительно ограничить ресурсы сети, доступные пользователю);
- при работе того или иного узла через брандмауэр может быть значительно снижена пропускная способность передачи данных в связи с необходимостью использования специальных механизмов обхода;
- участниками одноранговых сетей в основном являются индивидуальные пользователи, а не организации, в связи с чем возникают вопросы безопасности предоставления ресурсов: владельцы узлов P2P-сети, скорее всего, не знакомы друг с другом лично, предоставление ресурсов происходит без предварительной договорённости;
- при увеличении числа участников P2P сети может возникнуть ситуация значительного возрастания нагрузки на сеть (как с централизованной, так и с децентрализованной структурой);
- в случае применения сети типа P2P приходится направлять значительные усилия на поддержку стабильного уровня ее производительности, резервное копирование данных, антивирусную защиту, защиту от информационного шума и других злонамеренных действий пользователей. ...».

5.2.2 Технологии GRID

Другим подходом, относящимся к РВ-сетям, являются Grid-вычисления появившиеся в начале 1990-х годов, как метафора, демонстрирующая возможность простого доступа к вычислительным ресурсам. Согласно [63]: «... **Грид-вычисления** (англ. *grid* — решётка, сеть) — это форма распределенных вычислений, в которой «виртуальный суперкомпьютер» представлен в виде кластеров, соединённых с помощью сети, слабосвязанных гетерогенных компьютеров, работающих вместе для выполнения огромного количества заданий (операций, работ). Эта технология применяется для решения научных, математических задач, требующих значительных вычислительных ресурсов. Грид-вычисления используются также в коммерческой инфраструктуре для решения таких трудоемких задач, как экономическое

прогнозирование, сейсмоанализ, разработка и изучение свойств новых лекарств. Грид с точки зрения сетевой организации представляет собой согласованную, открытую и стандартизованную среду, которая обеспечивает гибкое, безопасное, скоординированное разделение вычислительных ресурсов и ресурсов хранения информации, которые являются частью этой среды, в рамках одной виртуальной организации. ...».

Не вдаваясь в детали реализации виртуальных суперкомпьютеров и виртуальных организаций, можно выделить следующие уровни архитектуры ***Grid***:

1. **Базовый уровень** (Fabric) – содержит различные ресурсы, такие как компьютеры, устройства хранения, сети, сенсоры и другие.
2. **Связывающий уровень** (Connectivity) – определяет коммуникационные протоколы и протоколы аутентификации.
3. **Ресурсный уровень** (Resource) – реализует протоколы взаимодействия с ресурсами РВС и их управления.
4. **Коллективный уровень** (Collective) – управление каталогами ресурсов, диагностика, мониторинг.
5. **Прикладной уровень** (Applications) – инструментарий для работы с ***Grid*** и пользовательские приложения.

С 2001 года, в качестве базы для создания стандарта архитектуры ***Grid*** была выбрана технология веб-сервисов. Этот выбор был обусловлен двумя основными причинами:

- язык описания интерфейсов веб-сервисов WSDL (Web Service Definition Language) поддерживает стандартные механизмы для определения интерфейсов отдельно от их реализации, что в совокупности со специальными механизмами связывания обеспечивает возможность динамического поиска и компоновки сервисов в гетерогенных средах;
- широко распространённая адаптация механизмов веб-сервисов означает, что инфраструктура, построенная на базе веб-сервисов, может использовать различные утилиты и другие существующие сервисы, такие как различные процессоры WSDL, системы планирования потоков задач и среды для размещения веб-сервисов.

5.2.3 Облачные вычисления и «виртуализация»

Концепция облачных вычислений является одной из новомодных современных концепций компьютерных технологий, так называемый - C*loud computing* [64].

Облачные вычисления (*cloud computing*) — технология распределенной обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как Интернет-сервис. Сам термин *«Облако»* используется как метафора, основанная на изображении *сложной инфраструктуры*, за которой скрываются все технические детали.

Многие авторы, на период 2008 года, указывали, что «Облачная обработка данных — *это парадигма*, в рамках которой информация постоянно хранится на серверах в Интернет и временно кэшируется на клиентской стороне, например, на персональных компьютерах, игровых приставках, ноутбуках, смартфонах и тому подобных устройствах». Для обеспечения согласованной работы ЭВМ, которые предоставляют услугу облачных вычислений, используется специализированное ПО, обобщенно называющееся «*Middleware control*». Это ПО обеспечивает:

- *мониторинг* состояния оборудования,

- *балансировку* нагрузки,
- *обеспечение ресурсов* для решения задачи.

Для облачных вычислений основным предположением является неравномерность запроса ресурсов со стороны клиентов. Для сглаживания этой неравномерности для предоставления сервиса между реальным железом и ***Middleware*** помещается ещё один слой - *виртуализация серверов*. Серверы, выполняющие приложения, виртуализируются и балансировка нагрузки осуществляется как средствами ПО, так и средствами распределения виртуальных серверов по реальным серверам, что соответственно порождает различные модели развёртывания. Приведём конкретные примеры таких моделей.

Частное облако (*private cloud)* — инфраструктура, предназначенная для использования одной организацией, включающей несколько потребителей, например, подразделений клиентами или подрядчиками данной организации. Частное облако может находиться в собственности, управлении и эксплуатации как самой организации, так и третьей стороны или какой-либо их комбинации, а также может физически существовать как внутри, так и вне юрисдикции владельца.

Публичное облако (*public cloud*) — инфраструктура, предназначенная для свободного использования широкой публикой. Публичное облако может находиться в собственности, управлении и эксплуатации коммерческих, научных и правительственных организаций или какой-либо их комбинации. Публичное облако физически существует в юрисдикции владельца - поставщика услуг.

Гибридное облако (*hybrid cloud*) — это комбинация из двух или более различных облачных инфраструктур - частных, публичных или коммунальных, остающихся уникальными объектами, но связанных между собой стандартизованными или частными технологиями переносимости данных и приложений, например, кратковременное использование ресурсов публичных облаков для балансировки нагрузки между облаками.

Общественное облако (*community cloud*) — вид инфраструктуры, предназначенный для использования конкретным сообществом потребителей из организаций, имеющих общие задачи, например, миссии, требования безопасности, политики, и соответствия различным требованиям. Общественное облако может находиться в кооперативной (совместной) собственности, управлении и эксплуатации одной или более из организаций сообщества или третьей стороны или какой-либо их комбинации, и может физически существовать как внутри так и вне юрисдикции владельца.

Завершая данный подраздел и главу в целом, отметим, что проведённый краткий обзор теоретических концепций СОТ, а также ряда их практических реализаций не обеспечивает студента полным набором знаний, необходимых для реализации таких систем. Главное, на что следует обратить внимание, это — интенсивное развитие указанного направления, которое в будущем должно обязательно прийти к своему насыщению и естественному теоретическому завершению рассматриваемой тематики.

Вопросы для самопроверки

1. В чем состоит основное отличие модели SOA от известных технологий CORBA и RMI?
2. Какое значение имеет ПО Middleware для систем построенных по моделям SOA?
3. Что такое - «*Интерфейс сервиса*» и где он применяется?
4. Что такое — UDDI и какое отношение к нему имеет понятие брокера?
5. На чем основан протокол SOAP и какие языки его поддерживают?
6. В чем отличие сильносвязанных программных систем от слабосвязанных?
7. В чем состоит парадигма первого поколения веб-сервисов?
8. Чем отличается второе поколение веб-сервисов от первого?
9. Соотношение каких схематических компонент предполагает «*Треугольник SOA*»?
10. Что означает аббревиатура WSDL и каково ее соотношение с архитектурами объектных распределенных систем?
11. Что подразумевается под сокращением РВ-сети?
12. Использование каких брокерных архитектур подразумевают Веб-сервисы?
13. Какую схему подразумевает асинхронное взаимодействие через посредника?
14. В чем состоят недостатки синхронных методов взаимодействия программных систем?
15. Что такое — P2P-сети?
16. В чем состоят преимущества децентрализованных пиринговых сетей?
17. Для чего предназначена технология GRID?
18. Какие уровни архитектуры выделяются в технологии GRID?
19. Что такое - «Виртуальный суперкомпьютер»?
20. Перечислите виды «*Облачных*» инфраструктур?