Лекция 12 Введение в распределенные системы

В настоящее время много внимания уделяется технологиям разработки распределенных приложений, охватывающих несколько независимых компьютеров. В течение последних десяти лет было создано большое число технологий и стандартов, использование которых должно было помочь разработчикам в создании распределенных приложений масштаба предприятия. Однако поддержка многих технологий была изначально достаточно трудоемкой и сложной для разработчиков прикладных программ, использовавших классические языки программирования, такие как C/C++.

Одной из задач, стоящих перед разработчиками Microsoft, создающими так называемую общеязыковую инфраструктуру (Common Language Infrastructure, CLI), так же известную как .NET, была наиболее полная поддержка средств разработки распределенных систем. Поэтому в платформе разработки приложений Microsoft .NET Framework имеется встроенная поддержка четырех взаимосвязанных технологий, предназначенных для использования в распределенных системах: очередей сообщений (messaging queues), объектов COM+, объектов .NET Remoting, веб служб (web services).

Каждая из данных технологий имеет свои достоинства, недостатки и особенности применения при построении распределенных приложений. Сделать осознанный выбор в пользу той или иной технологии при решении конкретных прикладных задач трудно без знакомства со всеми ними, также как и без базовых теоретических знаний о распределенных системах.

Понятие распределенной системы

По утверждению известного специалиста в области информатики Э. Таненбаума, не существует общепринятого и в то же время строгого определения распределенной системы. Некоторые остряки утверждают, что распределенной является такая вычислительная система, в которой неисправность компьютера, о существовании которого пользователи ранее даже не подозревали, приводит к остановке всей их работы. Значительная часть распределенных вычислительных систем, к сожалению, удовлетворяют такому определению, однако формально оно относится только к системам с уникальной точкой уязвимости (single point of failure).

Часто при определении распределенной системы во главу угла ставят разделение ее функций между несколькими компьютерами. При таком подходе распределенной является любая вычислительная система, где обработка данных разделена между двумя и более компьютерами. Основываясь на определении Э. Таненбаума, несколько более узко распределенную систему можно определить как набор соединенных каналами связи независимых компьютеров, которые с точки зрения пользователя некоторого программного обеспечения выглядят единым целым.

Такой подход к определению распределенной системы имеет свои недостатки. Например, все используемое в такой распределенной системе программное обеспечение могло бы работать и на одном единственном компьютере, однако с точки зрения приведенного выше определения такая система уже перестанет быть распределенной. Поэтому понятие распределенной системы, вероятно, должно основываться на анализе образующего такую систему программного обеспечения.

Как основу описания взаимодействия двух сущностей рассмотрим общую модель взаимодействия клиент-сервер, в которой одна из сторон (клиент) инициирует обмен

данными, посылая запрос другой стороне (серверу). Сервер обрабатывает запрос и при необходимости посылает ответ клиенту (рис. 1).

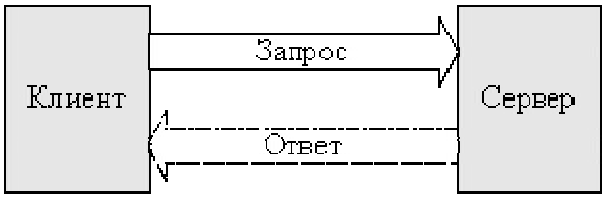

Рис. 1. Модель взаимодействия клиент сервер

Взаимодействие в рамках модели клиент сервер может быть как синхронным, когда клиент ожидает завершения обработки своего запроса сервером, так и асинхронным, при котором клиент посылает серверу запрос и продолжает свое выполнение без ожидания ответа сервера. Модель клиента и сервера может использоваться как основа описания различных взаимодействий. Для данного курса важно взаимодействие составных частей программного обеспечения, образующего распределенную систему.

Рис. 2. Логические уровни приложения

Рассмотрим некое типичное приложение, которое в соответствиис современными представлениями может быть разделено на следующие логические уровни (рис. 2): пользовательский интерфейс (ИП), логика приложения (ЛП) и доступ к данным (ДД), работающий с базой данных (БД). Пользователь системы взаимодействует с ней через интерфейс пользователя, база данных хранит данные, описывающие предметную область приложения, а уровень логики приложения реализует все алгоритмы, относящиеся к предметной области.

Поскольку на практике разных пользователей системы обычно интересует доступ к одним и тем же данным, наиболее простым разнесением функций такой системы между несколькими компьютерами будет разделение логических уровней приложения между одной серверной частью приложения, отвечающим за доступ к данным, и находящимися на нескольких компьютерах клиентскими частями, реализующими интерфейс пользователя. Логика приложения может быть отнесена к серверу, клиентам, или разделена между ними (рис. 3).

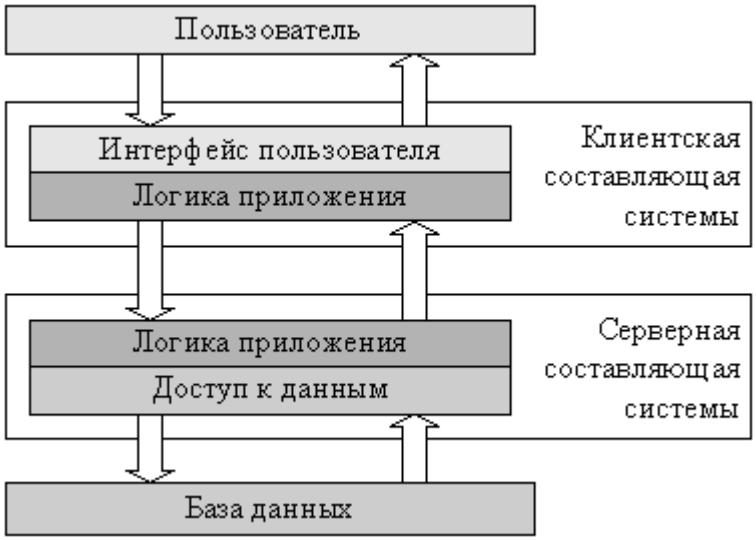

Рис. 3. Двухзвенная архитектура

Архитектуру построенных по такому принципу приложений называют клиент серверной или двухзвенной. На практике подобные системы часто не относят к классу распределенных, но формально они могут считаться простейшими представителями распределенных систем.

Развитием архитектуры клиент-сервер является трехзвенная архитектура, в которой интерфейс пользователя, логика приложения и доступ к данным выделены в самостоятельные составляющие системы, которые могут работать на независимых компьютерах (рис. 4).

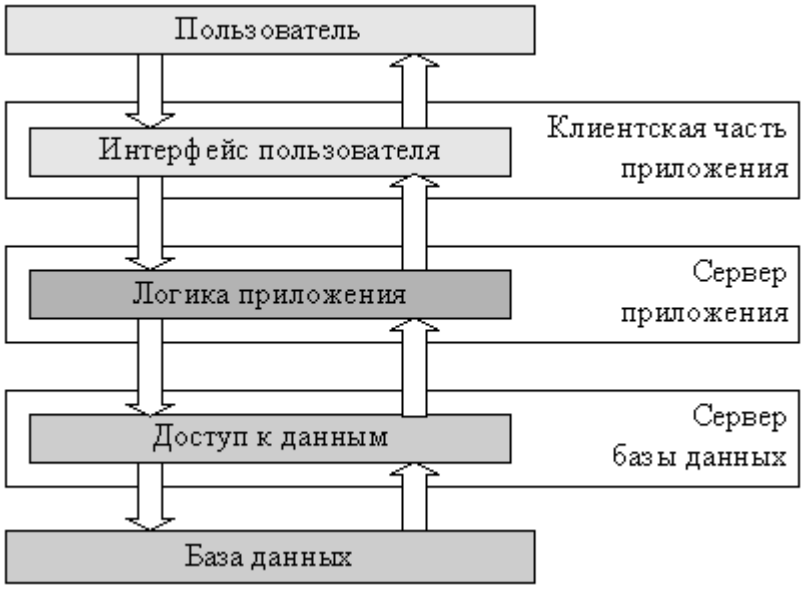

Рис. 4. Трехзвенная архитектура

Запрос пользователя в подобных системах последовательно обрабатывается клиентской частью системы, сервером логики приложения и сервером баз данных. Однако обычно под распределенной системой понимают системы с более сложной архитектурой, чем трехзвенная.

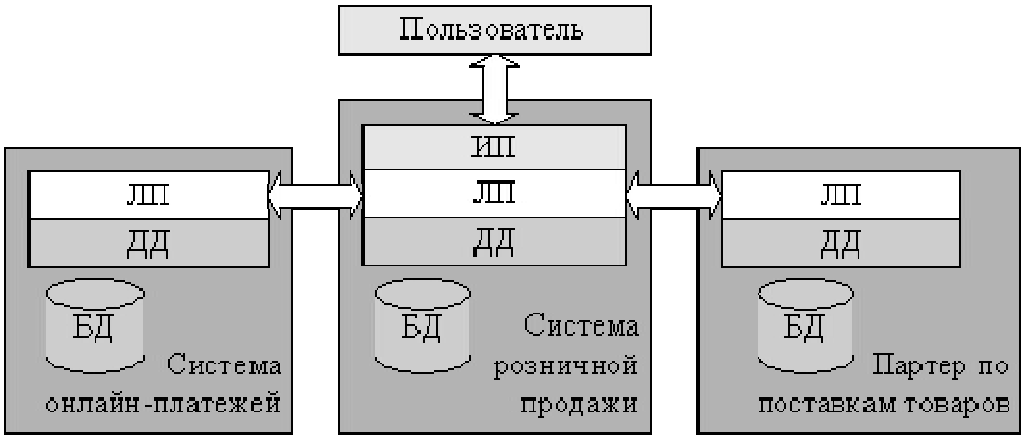

Рис. 5. Распределенная система розничных продаж

Применительно к приложениям автоматизации деятельности предприятия, распределенными обычно называют системы с логикой приложения, распределенной между несколькими компонентами системы, каждая из которых может выполняться на отдельном компьютере. Например, реализация логики приложения системы розничных продаж должна использовать запросы к логике приложения третьих фирм, таких как поставщики товаров, системы электронных платежей или банки, предоставляющие потребительские кредиты (рис. 5).

Таким образом, в обиходе под распределенной системой часто подразумевают рост многозвенной архитектуры "в ширину", когда запросы пользователя не проходят последовательно от интерфейса пользователя до единственного сервера баз данных.

В качестве другого примера распределенной системы можно привести сети прямого обмена данными между клиентами (peer-to-peer networks). Если предыдущий пример имел "древовидную" архитектуру, то сети прямого обмена организованы более сложным образом, рис. 6. Подобные системы являются в настоящий момент, вероятно, одними из крупнейших существующих распределенных систем, объединяющие миллионы компьютеров.

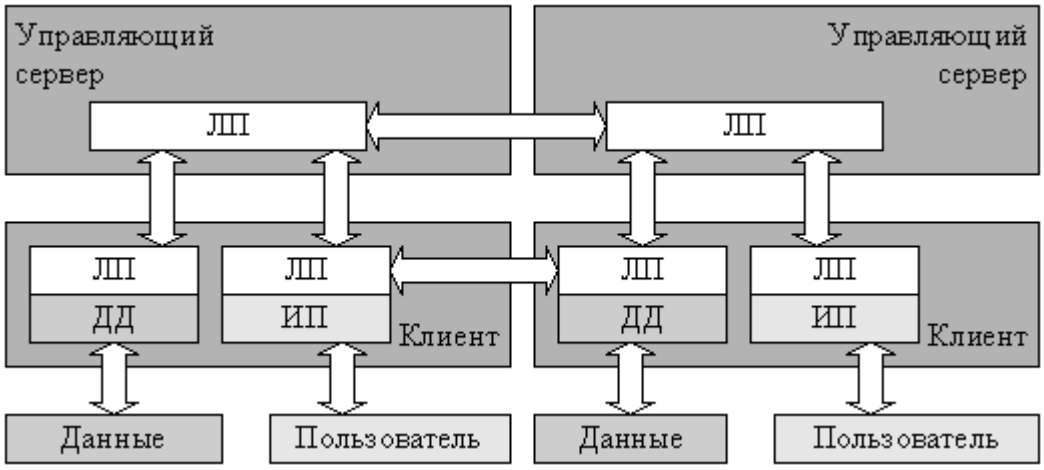

Рис. 6. Система прямого обмена данными между клиентами

Определение распределенной системы. Программные компоненты

В распределенных системах функции одного уровня приложения могут быть разнесены между несколькими компьютерами. С другой стороны, программное обеспечение, установленное на одном компьютере, может отвечать за выполнение функций, относящихся к разным уровням. Поэтому подход к определению распределенной системы, считающей ее совокупностью компьютеров, условен. Для описания и реализации распределенных систем было введено понятие программной компоненты.

Программная компонента – это единица программного обеспечения, исполняемая на одном компьютере в пределах одного процесса, и предоставляющая некоторый набор сервисов, которые используются через ее внешний интерфейс другими компонентами, как выполняющимися на этом же компьютере, так и на удаленных компьютерах. Ряд компонент пользовательского интерфейса предоставляют свой сервис конечному пользователю.

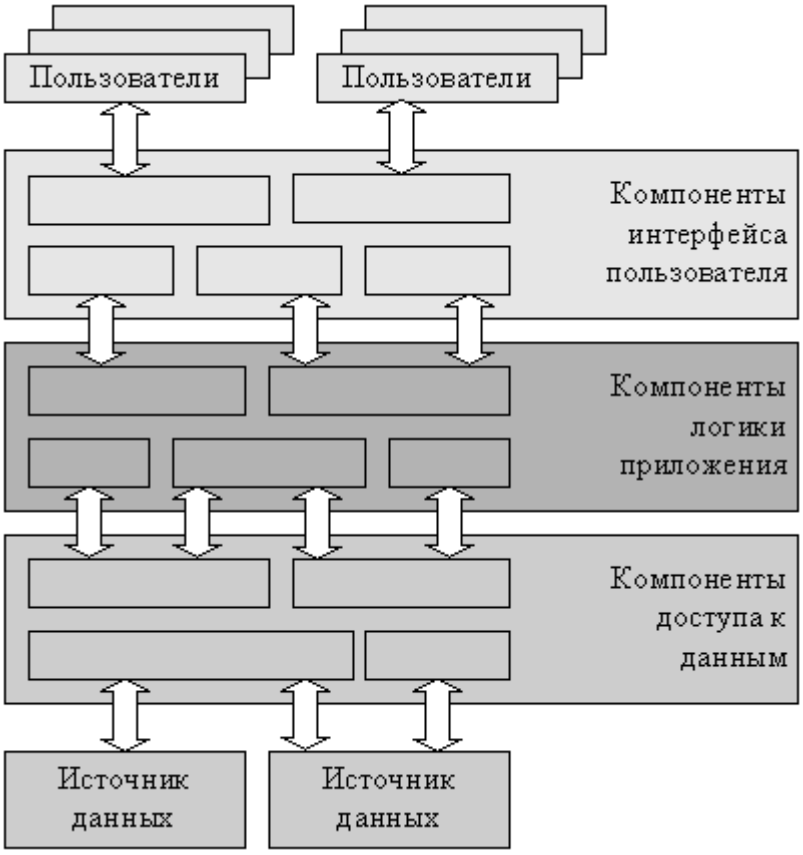

Рис. 7. Компоненты распределенной системы

Применительно к программам с использованием платформы CLI, под процессом в приведенном определении компоненты следует понимать домен приложения (application domain), который можно рассматривать как аналог процесса в управляемом коде.

Основываясь на определении программной компоненты, можно дать более точное определение распределенной системы. Согласно нему, распределенная система есть набор взаимодействующих программных компонент, выполняющихся на одном или нескольких связанных компьютерах и выглядящих с точки зрения пользователя системы как единое целое (рис. 7). Прозрачность является атрибутом распределенной системы. При исправном функционировании системы от конечного пользователя должно быть скрыто, где и как выполняются его запросы.

Программная компонента является минимальной единицей развертывания распределенной системы. В ходе модернизации системы одни компоненты могут быть обновлены независимо от прочих компонент.

В хорошо спроектированной системе функции каждой компоненты относятся только к одному уровню приложения. Однако разделение только на три уровня представляется недостаточным для классификации компонент. Например, часть компонент пользовательского интерфейса могут взаимодействовать с пользователем, а часть – предоставлять свои сервисы другим компонентам, но с пользователем не взаимодействовать. Классификации подобного рода существуют, однако они не являются общепринятыми и часто в значительной степени привязаны к приложениям автоматизации деятельности предприятия, что все таки не является синонимом распределенной системы.

Требования к распределенным системам

Чтобы достигнуть цели своего существования – улучшения выполнения запросов пользователя – распределенная система должна удовлетворять некоторым необходимым требованиям. Можно сформулировать следующий набор требований, которым в наилучшем случае должна удовлетворять распределенная вычислительная система.

Открытость. Все протоколы взаимодействия компонент внутри распределенной системы в идеальном случае должны быть основаны на общедоступных стандартах. Это позволяет использовать для создания компонент различные средства разработки и операционные системы. Каждая компонента должна иметь точную и полную спецификацию своих сервисов. В этом случае компоненты распределенной системы могут быть созданы независимыми разработчиками. При нарушении этого требования может исчезнуть возможность создания распределенной системы, охватывающей несколько независимых организаций.

Масштабируемость. Масштабируемость вычислительных систем имеет несколько аспектов. Наиболее важный из них для данного курса – возможность добавления в распределенную систему новых компьютеров для увеличения производительности системы, что связано с понятием балансировки нагрузки (load balancing) на серверы системы. К масштабированию относятся так же вопросы эффективного распределения ресурсов сервера, обслуживающего запросы клиентов.

Поддержание логической целостности данных. Запрос пользователя в распределенной системе должен либо корректно выполняться целиком, либо не выполняться вообще. Ситуация, когда часть компонент системы корректно обработали поступивший запрос, а часть – нет, является наихудшей.

Устойчивость. Под устойчивостью понимается возможность дублирования несколькими компьютерами одних и тех же функций или же возможность автоматического распределения функций внутри системы в случае выхода из строя одного из компьютеров. В идеальном случае это означает полное отсутствие уникальной точки сбоя, то есть выход из строя одного любого компьютера не приводит к невозможности обслужить запрос пользователя.

Безопасность. Каждый компонент, образующий распределенную систему, должен быть уверен, что его функции используются авторизированными на это компонентами или

пользователями. Данные, передаваемые между компонентами, должны быть защищены как от искажения, так и от просмотра третьими сторонами.

Эффективность. В узком смысле применительно к распределенным системам под эффективностью будет пониматься минимизация накладных расходов, связанных с распределенным характером системы. Поскольку эффективность в данном узком смысле может противоречить безопасности, открытости и надежности системы, следует отметить, что требование эффективности в данном контексте является наименее приоритетным. Например, на поддержку логической целостности данных в распределенной системе могут тратиться значительные ресурсы времени и памяти, однако система с недостоверными данными вряд ли нужна пользователям. Желательным свойством промежуточной среды является возможность организации эффективного обмена данными, если взаимодействующие программные компоненты находятся на одном компьютере. Эффективная промежуточная среда должна иметь возможность организации их взаимодействия без затрагивания стека TCP/IP. Для этого могут использоваться системные сокеты (unix sockets) в POSIX системах или именованные каналы (named pipes).

Устойчивость распределенной системы связана с понятием масштабируемости, но не эквивалентна ему. Допустим, система использует набор обрабатывающих запросы серверов и один диспетчер запросов, который распределяет запросы пользователей между серверами. Такая система может считаться достаточно хорошо масштабируемой, однако диспетчер является уязвимой точкой такой системы. С другой стороны, система с единственным сервером может быть устойчива, если существует механизм его автоматической замены в случае выхода его из строя, однако она вряд ли относится к классу хорошо масштабируемых систем. На практике достаточно часто встречаются распределенные системы, не удовлетворяющие данным требованиям: например, любая система с уникальным сервером БД, реализованным в виде единственного компьютера, имеет уникальную точку сбоя. Выполнение требований устойчивости и масштабируемости обычно связано с некоторыми дополнительным расходами, что на практике может быть не всегда целесообразно. Однако используемые при построении распределенных систем технологии должны допускать принципиальную возможность создания устойчивых и высоко масштабируемых систем.

Классическим примером системы, в значительной мере отвечающей всем представленным выше требованиям, является система преобразования символьных имен в сетевые IP-адреса (DNS). Система имен – организованная иерархически распределенная система, с дублированием всех функций между двумя и более серверами (рис. 8).

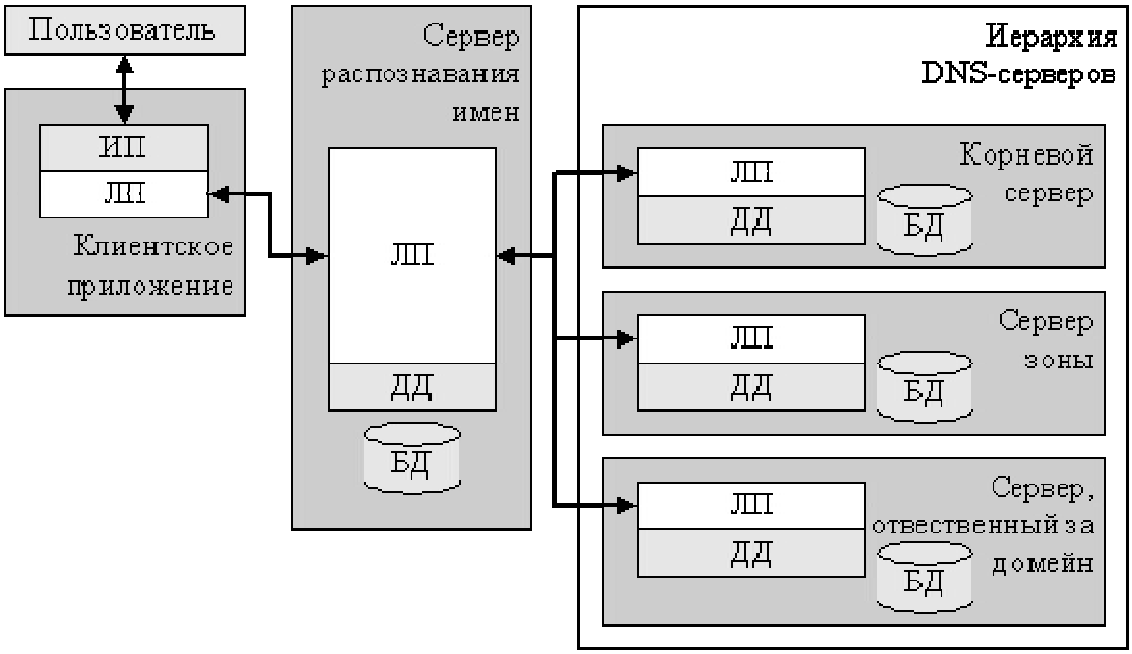

Рис. 8. Система DNS

Запрос пользователя на преобразование имени (например, w3c.org) в сетевой адрес передается серверу распознавания имен поставщика услуг интернета. Сервер распознавания имен по очереди опрашивает серверы из иерархии службы имен. Опрос начинается с корневых серверов, который возвращает адреса серверов, ответственных за зону домена. Затем опрашивается сервер, ответственный за зону (в данном случае – .org), возвращающий адреса серверов, ответственных за домен второго уровня, и так далее. Серверы имен кешируют информацию о соответствии имен и адресов для уменьшения нагрузки на систему. Программное обеспечение на компьютере пользователя обычно имеет возможность соединиться с как минимум двумя различными серверами распознавания имен.

Тем не менее, и в системе распознавания имен не все требования к распределенным системам выполнены. В частности, она не содержит каких-либо явных механизмов обеспечения безопасности. Это приводит к регулярным атакам на серверы имен в надежде вывести их из строя, например, большим количеством запросов.

Понятие промежуточной среды

С точки зрения одного из компьютеров распределенной системы, все другие входящие в нее машины являются удаленными вычислительными системами. Теоретической основой сетевого взаимодействия удаленных систем является общеизвестная модель взаимодействия открытых систем OSI/ISO, которая разделяет процесс взаимодействия двух сторон на семь уровней: физический, канальный, сетевой, транспортный, сеансовый, прикладной, представительский.

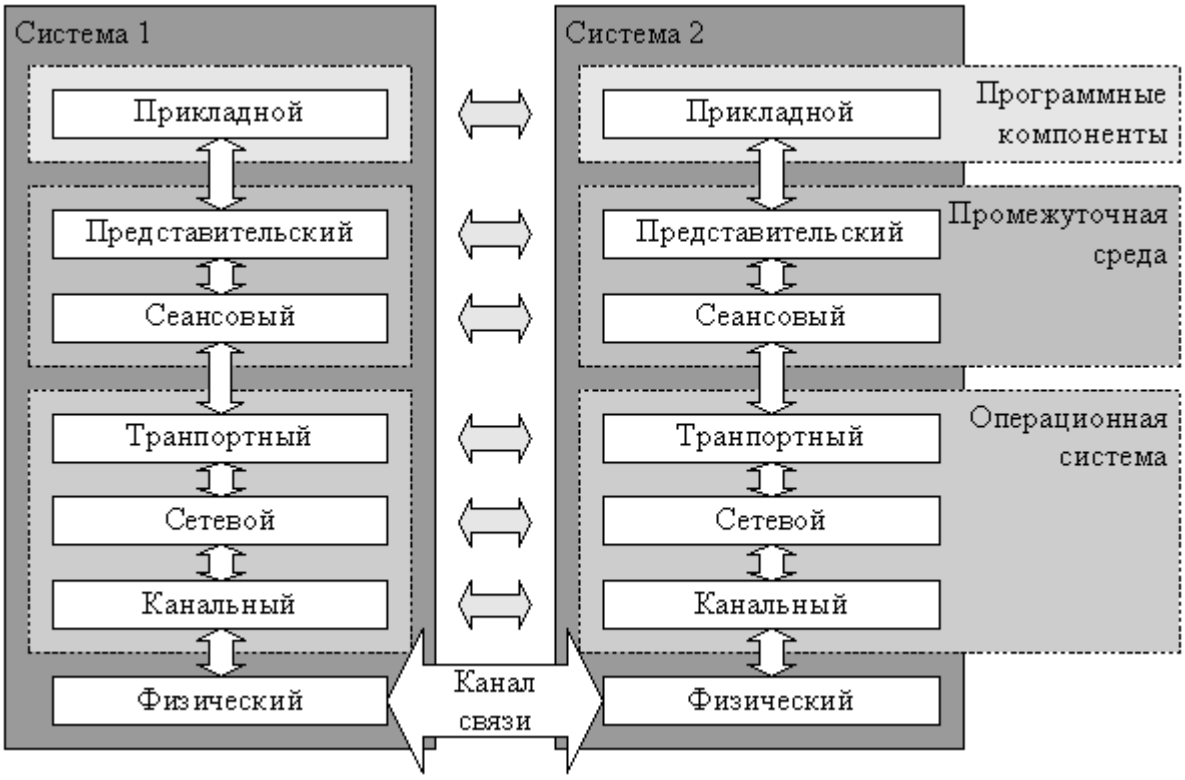

Рис. 9. Модель взаимодействия вычислительных систем

В сетях наиболее распространенного стека протоколов TCP/IP протокол TCP является протоколом транспортного, а протокол IP – протоколом сетевого уровня. Обеспечение интерфейса к транспортному уровню в настоящее время берет на себя сетевая компонента операционной системы, предоставляя обычно основанный на сокетах интерфейс для верхних уровней. Сокеты обеспечивают примитивы низкого уровня для непосредственного обмена потоком байт между двумя процессами. Стандартного представительского или сеансового уровня в стеке протоколов TCP/IP нет, иногда к ним относят защищенные протоколы SSL/TLS.

Использование протокола TCP/IP посредством сокетов предоставляет стандартный, межплатформенный, но низкоуровневый сервис для обмена данными между компонентами. Для выполнения сформулированных выше требований к распределенным системам функции сеансового и представительского уровня должна взять на себя некоторая промежуточная среда (middleware), называемая так же промежуточным программным обеспечением (рис. 9). Такая среда должна помочь создать разработчиками открытые, масштабируемые и устойчивые распределенные системы. Для достижения этой цели промежуточная среда должна обеспечить сервисы для взаимодействия компонент распределенной системы. К таким сервисам относятся:

- обеспечение единого и независимого от операционной системы механизма использования одними программными компонентами сервисов других компонент;
- обеспечение безопасности распределенной системы: аутентификация и авторизация всех пользователей сервисов компоненты и защита передаваемой между компонентами информации от искажения и чтения третьими сторонами;
- обеспечение целостности данных: управление транзакциями, распределенными между удаленными компонентами системами;
- балансировка нагрузки на серверы с программными компонентами;
- обнаружение удаленных компонент.

В пределах одной распределенной системы может использоваться несколько видов промежуточных сред (рис. 10). При хорошем подходе к проектированию системы каждая распределенная ее компонента предоставляет свои сервисы посредством единственной промежуточной среды, и использует сервисы других компонент посредством так же единственной промежуточной среды, однако эти среды могут быть различными.

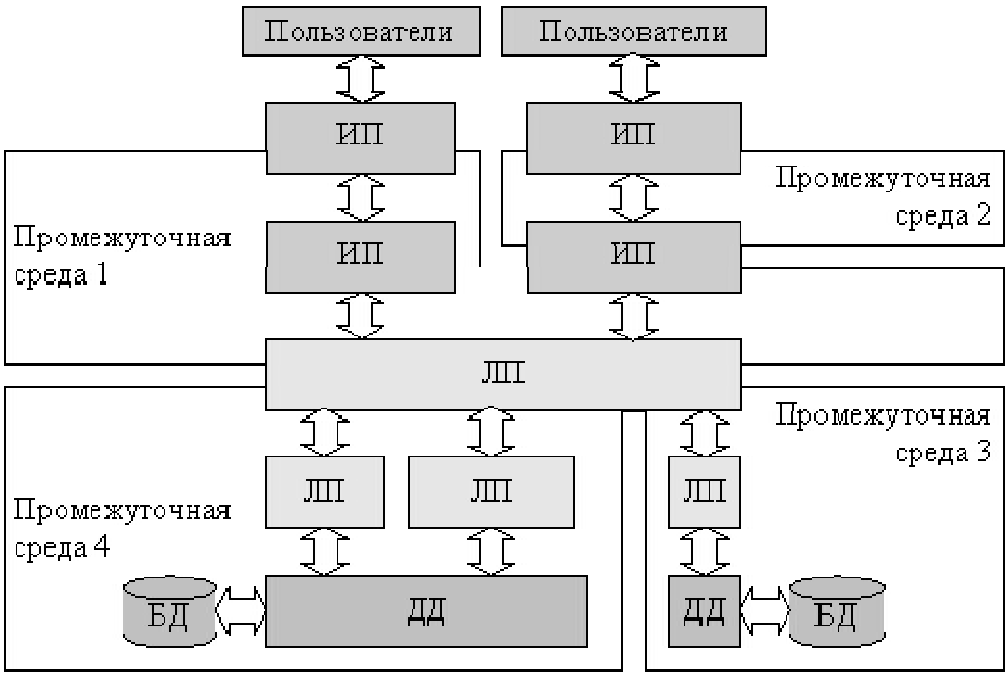

Рис. 10. Гетерогенная распределенная система

Распределенную систему, компоненты которой используют несколько промежуточных сред, можно называть гетерогенной, в противоположность гомогенной, использующей единственную промежуточную среду. Поскольку одна и та же промежуточная среда может быть реализована на различных аппаратных платформах и операционных системах, то оба класса распределенных систем могут включать в себя компьютеры под управлением как одной, так и различных операционных систем.

В настоящий момент нет универсально применимой промежуточной среды, хотя как будет показано в курсе, есть определенное движение в этом направлении. Основной причиной отсутствия такой среды являются отчасти противоречивые требования к распределенным системам, а так же различающийся характер сетевых соединений между компонентами системы: например взаимодействие компонент внутри одного предприятия, вероятно, может быть построено иначе, чем взаимодействие компонент двух различных предприятий, не полностью доверяющих друг другу.

Взаимодействие программных компонент в пределах одного и того же компьютера также происходит при помощи промежуточной среды, что при использовании некоторых промежуточных сред может быть как неудобно, так и неэффективно. В идеальном случае распределенная компонента должна быть реализована таким образом, чтобы переход с одной промежуточной среды на другую происходил путем изменения конфигурации программной компоненты, а не изменения исходного кода. На практике данное требование, к сожалению, может быть трудно осуществимо, однако необходимо хотя бы минимизировать возможные исправления программного кода при возможной смене промежуточной среды.