Глава 1 . Взаимодействие процессов распределенного приложения

1.1. Предисловие к главе

Когда рассматривают принципы взаимодействия различных частей (процессов) распределенного приложения, то говорят о ***модели взаимодействия***, а когда рассматривают распределение ролей между различными частями (процессами) распределенного приложения, то говорят об ***архитектуре распределенного приложения***.

Для обсуждения принципов взаимодействия процессов распределенного приложения, как правило, применяется модель ***ISO/OSI*** (International Standards Organization/Open System Interconnection reference model), которая была разработана в 1980-х годах и регулируется стандартом ISO 7498. Официальное название модели ISO/OSI – ***сетевая эталонная модель взаимодействия открытых систем Международной организации по стандартизации***. Спецификации ISO/OSI используются производителями аппаратного и программного обеспечений

Наиболее популярной архитектурой для распределенного программного приложения является ***архитектура клиент-сервер***. Будем говорить, что распределенное приложение имеет архитектуру клиент-сервер, если все процессы распределенного приложения можно условно разбить на две группы. Одна группа процессов называется серверами другая – клиентами. Обмен данными осуществляется только между процессами-клиентами и процессами-серверами. Основное отличие процесса-клиента от процесса-сервера в том, что инициатором обмена данными всегда является процесс-клиент. Другими словами процесс-клиент обращается за услугой (сервисом) к процессу-серверу. Такая архитектура лежит в основе большинства современных информационных систем [1,2,3].

В этой главе рассматривается модель ISO/OSI и особенности архитектуры клиент-сервер.

1.2. Модель взаимодействия открытых систем

Функции, обеспечивающие взаимодействие открытых систем в модели ISO/OSI распределены по следующим семи уровням: 1) физический; 2) канальный; 3) сетевой; 4) транспортный; 5) сеансовый; 6) представительский; 7) прикладной. Задача каждого уровня – предоставление услуг вышестоящему уровню таким образом, чтобы детали реализации этих услуг были скрыты. Наборы правил и соглашений, описывающих процедуры взаимодействия каждого уровня модели с соседними уровнями называются ***протоколами***.

Опишем кратко назначение всех уровней модели OSI.

Физический уровень. Физический уровень определяет свойства среды передачи данных (коаксиальный кабель, витая пара, оптоволоконный канал и т.п.) и способы ее соединения с сетевыми адаптерами: технические характеристики кабелей (сопротивление, емкость, изоляция и т.д.), перечень допустимых разъемов, способы обработки сигнала и т.п.

Канальный уровень. На канальном уровне модели рассматривается два подуровня: подуровень управления доступом к среде передачи данных и подуровень управления логическим каналом. Управление доступом к среде передачи данных определяет методы совместного использования сетевыми адаптерами среды передачи данных. Подуровень управления логической связью определяет понятия канала между двумя сетевыми адаптерами, а также способы обнаружения и исправления ошибок передачи данных. Основное назначение процедур канального уровня подготовить блок данных (обычно называемый кадром) для следующего сетевого уровня.

Здесь следует отметить два момента: 1) начиная с подуровня управления логической связью и выше протоколы никак не зависят от среды передачи данных; 2) для организации локальной сети достаточно только физического и канального уровней, но такая сеть не будет масштабируемой (не сможет расширяться), т.к. имеет ограниченные возможности адресации и не имеет функций маршрутизации.

Сетевой уровень. Сетевой уровень определяет методы адресации и маршрутизации компьютеров в сети. В отличие от канального уровня сетевой уровень определяет единый метод адресации для всех компьютеров в сети не зависимого от способа передачи данных. На этом уровне определяются способы соединения компьютерных сетей. Результатом процедур сетевого уровня является пакет, который обрабатывается процедурами транспортного уровня.

Транспортный уровень. Основным назначением процедур транспортного уровня является подготовка и доставка пакетов данных между конечными точками без ошибок и в правильной последовательности. Процедуры транспортного уровня формируют файлы для сеансового уровня из пакетов, полученных от сетевого уровня.

Сеансовый уровень. Сеансовый уровень определят способы установки и разрыва соединений (называемых сеансами) двух приложений, работающих в сети.

Следует отметить, что сеансовый уровень - это точка взаимодействия программ и компьютерной сети.

Представительский уровень. На представительский уровне определяется формат данных, используемых приложениями. Процедуры этого уровня описывают способы шифрования, сжатия и преобразования наборов символов данных.

Прикладной уровень. Основное назначения уровня: определить способы взаимодействия пользователей с системой (определить интерфейс).

На рис. 1.2.1 изображена схема взаимодействия двух систем с точки зрения модели OSI. Толстой линией со стрелками на концах обозначается движение данных между системами. Данные проходят от прикладного уровня одной системы до прикладного уровня другой через все нижние уровни системы. Причем по мере своего движения от отправителя к получателю (из одной системы в другую) на каждом уровне данные подвергаются необходимому преобразованию (в соответствии с

протоколами модели): при движении от прикладного уровня к физическому данные преобразовываются в формат, позволяющий передать данные по физическому каналу; при движении от физического уровня до прикладного происходит обратное преобразование данных. При такой организации обмена данными фактически взаимодействие осуществляется между одноименными уровнями (на рисунке это взаимодействие обозначено линиями со стрелками между уровнями двух систем).

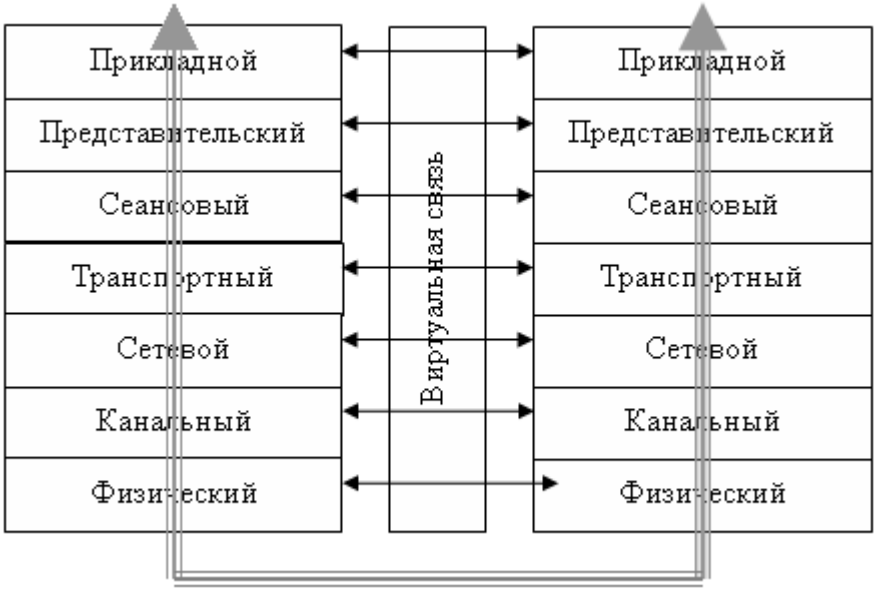

Рисунок 1.2.1. Схема взаимодействия открытых систем

Выше уже отмечалось, что точкой взаимодействия программ и компьютерной сети является сеансовый уровень. Рассмотрим этот момент более подробно для распределенного приложения состоящего из двух взаимодействующих процессов.

На рис. 1.2.2 изображены два процесса (с именами C и S), функционирующие на разных компьютерах в среде соответствующих операционных систем. В составе операционных систем имеются службы (специальные программы) обеспечивающие поддержку протоколов канального, сетевого и транспортного уровней. Протоколы физического уровня, как правило, обеспечиваются сетевыми адаптерами. На рисунке граница операционной системы условно проходит по канальному уровню. Действительно, часто (но не всегда) часть процедур канального уровня (обычно подуровня управления доступом к среде) обеспечивается аппаратно (сетевым адаптером), а другая часть процедур (обычно подуровня управления логическим каналом) реализована в виде драйвера, инсталлированного в состав операционной системы. Процессы, взаимодействуют со службами, обеспечивающими процедуры протоколов транспортного уровня с помощью набора специальных функций API (Application Program Interface), входящими в состав операционной системы. Следует отметить, что рисунок 1.2.2 носит чисто схематический характер и

служит, только для объяснения принципа взаимодействия процессов в распределенном приложении. В каждом конкретном случае распределение протокольных процедур разное и зависит от архитектуры компьютера, степени интеллектуальности сетевого адаптера, типа операционной системы и т.п. Заметим также, что функции сеансового, представительского и прикладного уровней обеспечивается самим распределенных приложением.

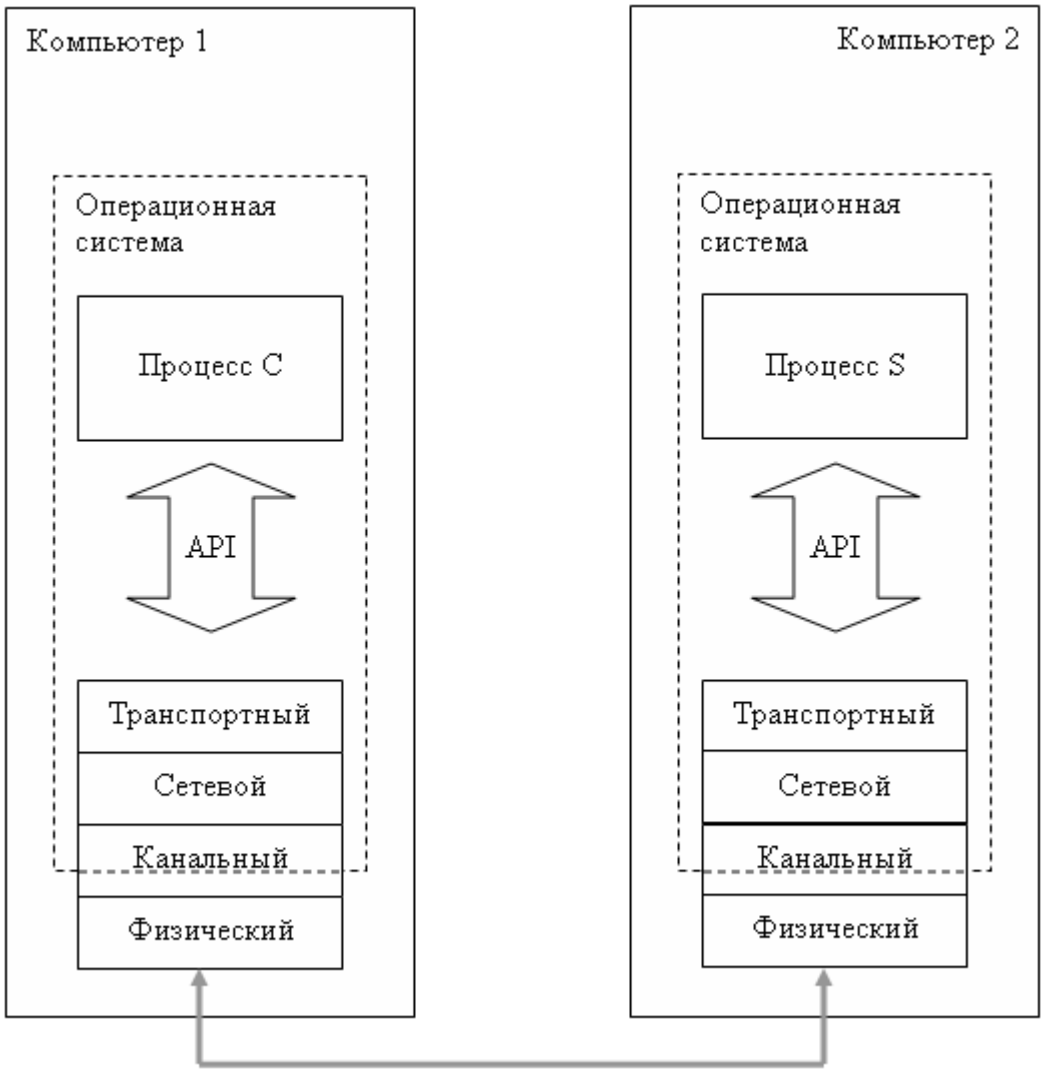

Рисунок 1.2.2. Схема взаимодействия процессов в распределенном приложении

1.3. Архитектура клиент-сервер

Распределенное приложение, имеющее архитектуру клиент-сервер, подразумевает наличие в своем составе два вида процессов: процессы-серверы и процессы-клиенты. Далее эти процессы будем назвать просто серверами и клиентами.

Следует отметить, что приведенное в предисловии определение архитектуры клиент-сервер несколько упрощено. Дело в том, что некоторые процессы распределенного приложения могут выступать клиентом для некоторых процессов-серверов и одновременно являться серверами других процессов-клиентов.

Инициатором обмена данными между клиентом и сервером всегда является клиент. Для этого клиент должен обладать информацией о месте нахождения сервера или иметь механизмы для его обнаружения. Способы связи между клиентом и сервером могут быть различными и в общем случае зависят от интерфейсов, поддерживаемых операционной средой, в которой работает распределенное приложение. Клиент должен быть тоже распознан сервером, для того, чтобы сервер, во-первых, мог его отличить от других клиентов, а во-вторых, чтобы смог обмениваться с клиентом данными. Если основная вычислительная нагрузка ложится на сервер, а клиент лишь обеспечивает интерфейсом пользователя с сервером, то такой клиент часто называют ***тонким***.

По методу облуживания серверы подразделяются на ***итеративные*** и ***параллельные*** серверы ***(iterative and concurrent servers)***. Принципиальная разница заключается в том, что параллельный сервер предназначен для обслуживания нескольких клиентов одновременно и поэтому использует специальные средства операционной системы позволяющие распараллеливать обработку нескольких клиентских запросов. Итеративный сервер, как правило, обслуживает запросы клиентов поочередно, заставляя клиентов ожидать своей очереди на обслуживание, или просто отказывает клиенту обслуживании. По всей видимости, можно говорить о итеративно-параллельных серверах, когда сервер имеет ограниченные возможности по распараллеливанию своей работы. В этом случае только часть клиентских запросов будет обслуживаться параллельно.

1.3. Итоги главы

1. Принцип взаимодействия процессов распределенного приложения следует рассматривать в рамках модели ISO/OSI.
2. Как правило, при разработке распределенного приложения, разработчику нет необходимости вникать в детали обмена данными между процессами этого приложения. В основном программист руководствуется API, предоставленный операционной системой для взаимодействия со специальными программами, обеспечивающими выполнение процедур протокола транспортного уровня.
3. Изменение в конфигурации компьютерной сети не приведут к необходимости перепрограммирования приложения, если эти изменения не касаются протокола транспортного уровня.
4. Процессы распределенного приложения могут работать на компьютерах разной архитектуры, в разных операционных средах (на разных платформах). В общем случае разработчику приходится

программировать приложение (или его части) для каждой конкретной платформы.

5. Архитектура клиент-сервер лежит в основе большинства современных информационных систем. Для разработки параллельного сервера, необходимо, чтобы операционная система предоставляла возможность распараллеливания процессов.
6. При разработке распределенных приложений могут быть использованы готовые решения: серверы систем управления базами данных (например, Microsoft SQL Server, Oracle Server), серверы приложений (Citrix Application Server), Web-серверы (Apache, Microsoft IIS, Apache Tomcat), браузеры (Microsoft Internet Explorer, Opera, Netscape Navigator) и т.д. Применение готовых решений значительно упрощает разработку распределенных приложений.