КРОСС-ПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ

УДАЛЕННЫЙ ВЫЗОВ ПРОЦЕДУР
(Remote procedure call)

Удаленный вызов процедур

Концепция **вызова удаленных процедур** (*remote procedure call — RPC*) была разработана и реализована в компанией XEROX в начале 80-х годов XX века. Общий смысл RPC можно представить так: программа может выполнять не только собственные (скомпилированные) процедуры и функции, но и обращаться к процедурам удаленного сервера.

Наибольшая эффективность использования RPC достигается в тех приложениях, в которых существует интерактивная связь между удаленными компонентами с небольшим временем ответов и относительно малым количеством передаваемых данных. Такие приложения называются **RPC-ориентированными**.

Характерными чертами вызова локальных процедур являются:

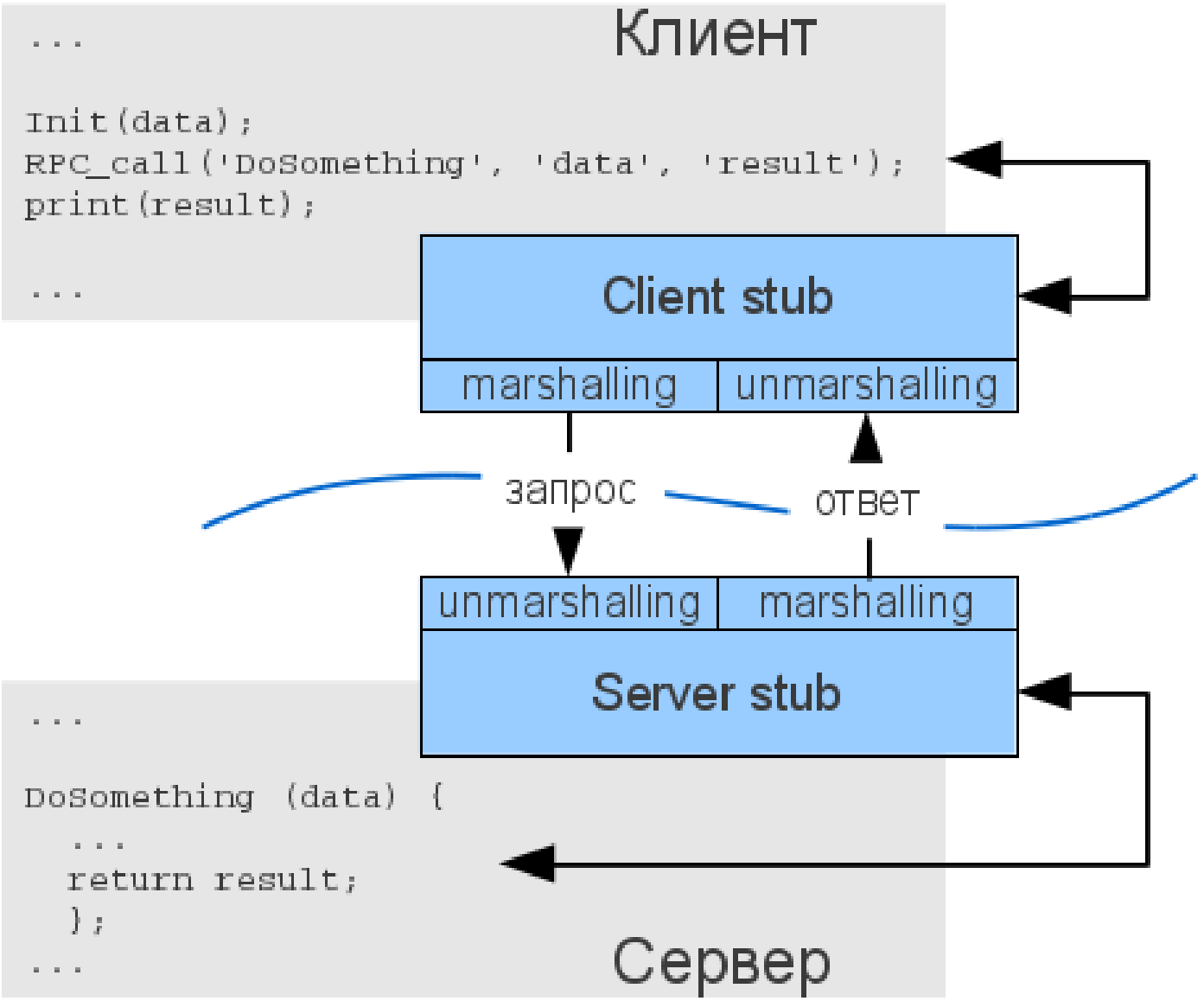
- Асимметричность, то есть одна из взаимодействующих сторон является инициатором;
- Синхронность, то есть выполнение вызывающей процедуры при останавливается с момента выдачи запроса и возобновляется только после возврата из вызываемой процедуры.

Удаленный вызов процедур

Реализация удаленных вызовов существенно сложнее реализации вызовов локальных процедур.

- Поскольку вызывающая и вызываемая процедуры выполняются на разных машинах, то они имеют разные адресные пространства, и это создает проблемы при передаче параметров и результатов, особенно если машины не идентичны.
- RPC обязательно использует нижележащую систему связи, однако это не должно быть явно видно ни в определении процедур, ни в самих процедурах.
- Выполнение вызывающей программы и вызываемой локальной процедуры в одной машине реализуется в рамках единого процесса. Но в реализации RPC участвуют как минимум два процесса – по одному в каждой машине. В случае, если только один из них аварийно завершится, могут возникнуть сложные проблемы.
- Проблемы, связанные с неоднородностью языков программирования и операционных сред: структуры данных и структуры вызова процедур, поддерживаемые в каком-либо одном языке программирования, не поддерживаются точно так же во всех других языках.

Общая схема удаленного вызова процедур



Для установки связи, передачи вызова и возврата результата клиентский и серверный процессы обращаются к специальным компонентам – «заглушкам», соответственно клиентской и серверной (client stub и server stub).

Эти stub'ы не реализуют никакой прикладной логики и предназначены только для организации взаимодействия удаленных (в общем случае) приложений.

Все RPC-обращения клиента (имена функций и списки параметров) упаковываются клиентской заглушкой в сетевые сообщения (этот процесс называется *marshaling*) и ей же передаются серверной заглушке.

Та, в свою очередь, распаковывает полученные данные (*unmarshaling*), передает их реальной функции сервера, а полученные результаты упаковывает в обратном порядке.

Клиентская заглушка принимает ответ, распаковывает его и возвращает в приложение.

Таким образом, процедуры-заглушки изолируют прикладные модули клиента и сервера от уровня сетевых коммуникаций.

Для определения правил, задающих отношения между клиентом и сервером RPC, используется язык описания интерфейсов (Interface Definition Language, IDL).

Интерфейс содержит определение имени функции и полное описание передаваемых параметров и результатов выполнения.

Правила IDL обеспечивают независимость механизма RPC от языков программирования: обращаясь к удаленной процедуре, клиент использует свои языковые конструкции, а IDL-компилятор преобразует их в унифицированные описания, понятные серверу.

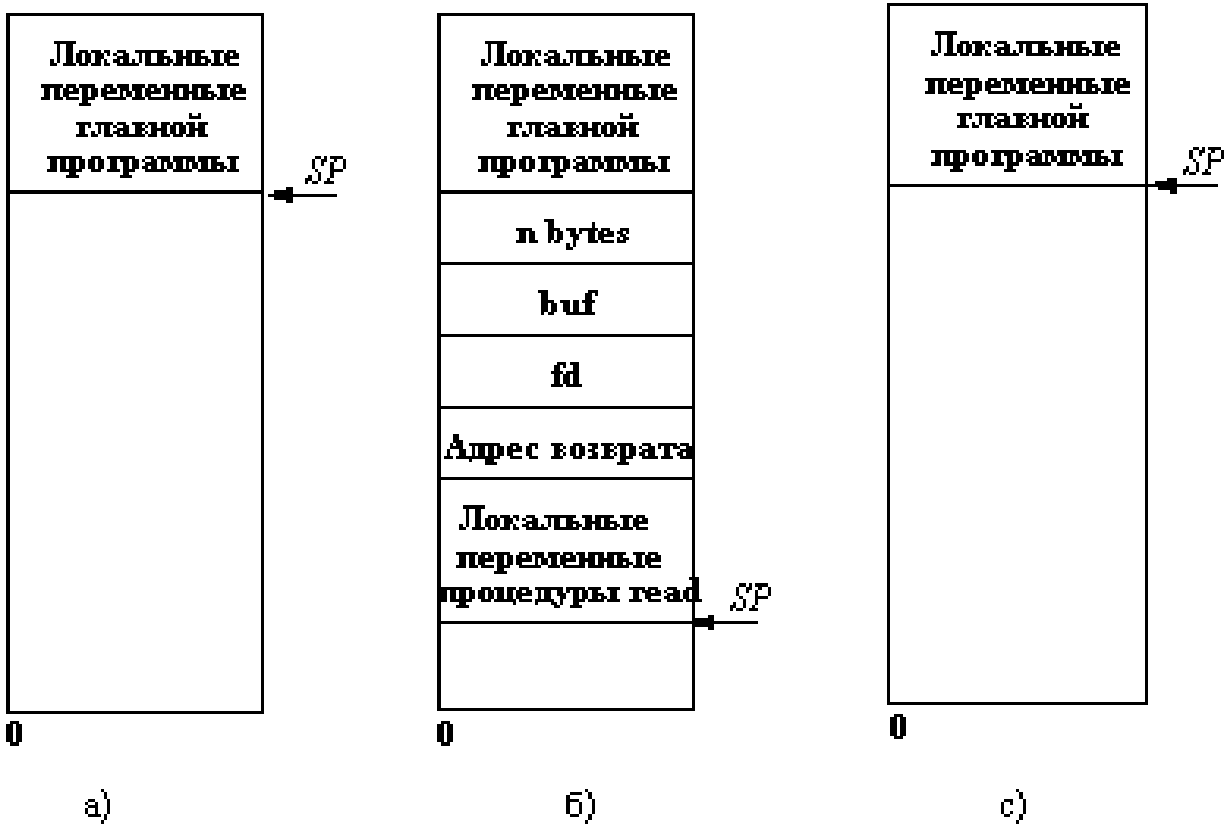
На сервере IDL-описания преобразуются в конструкции языка программирования, на котором реализован серверный процесс.

Базовые операции RPC

Пусть есть системный вызов:

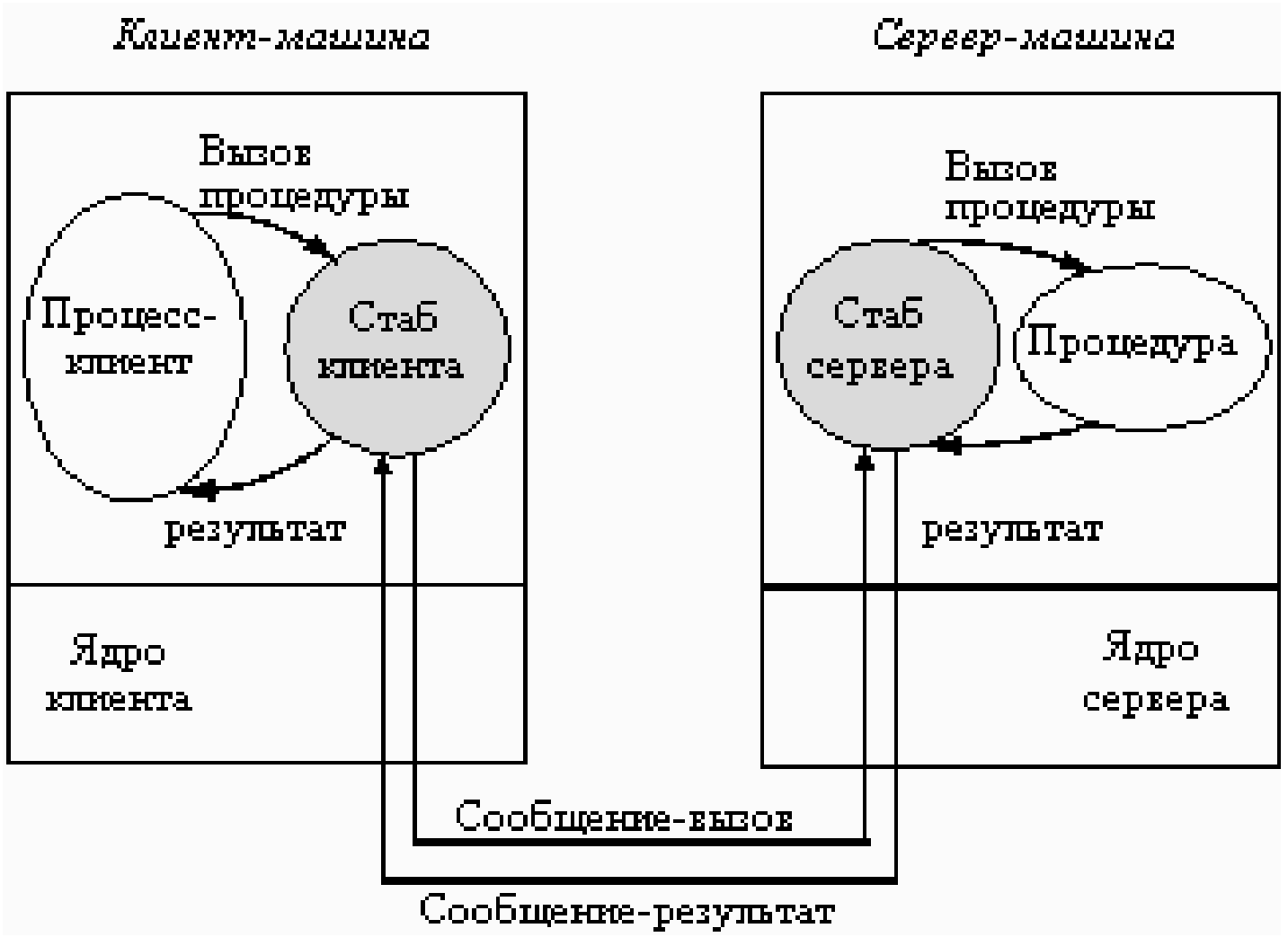
count=read (fd,buf,nbytes);

где fd - целое число, buf - массив символов, nbytes - целое число.

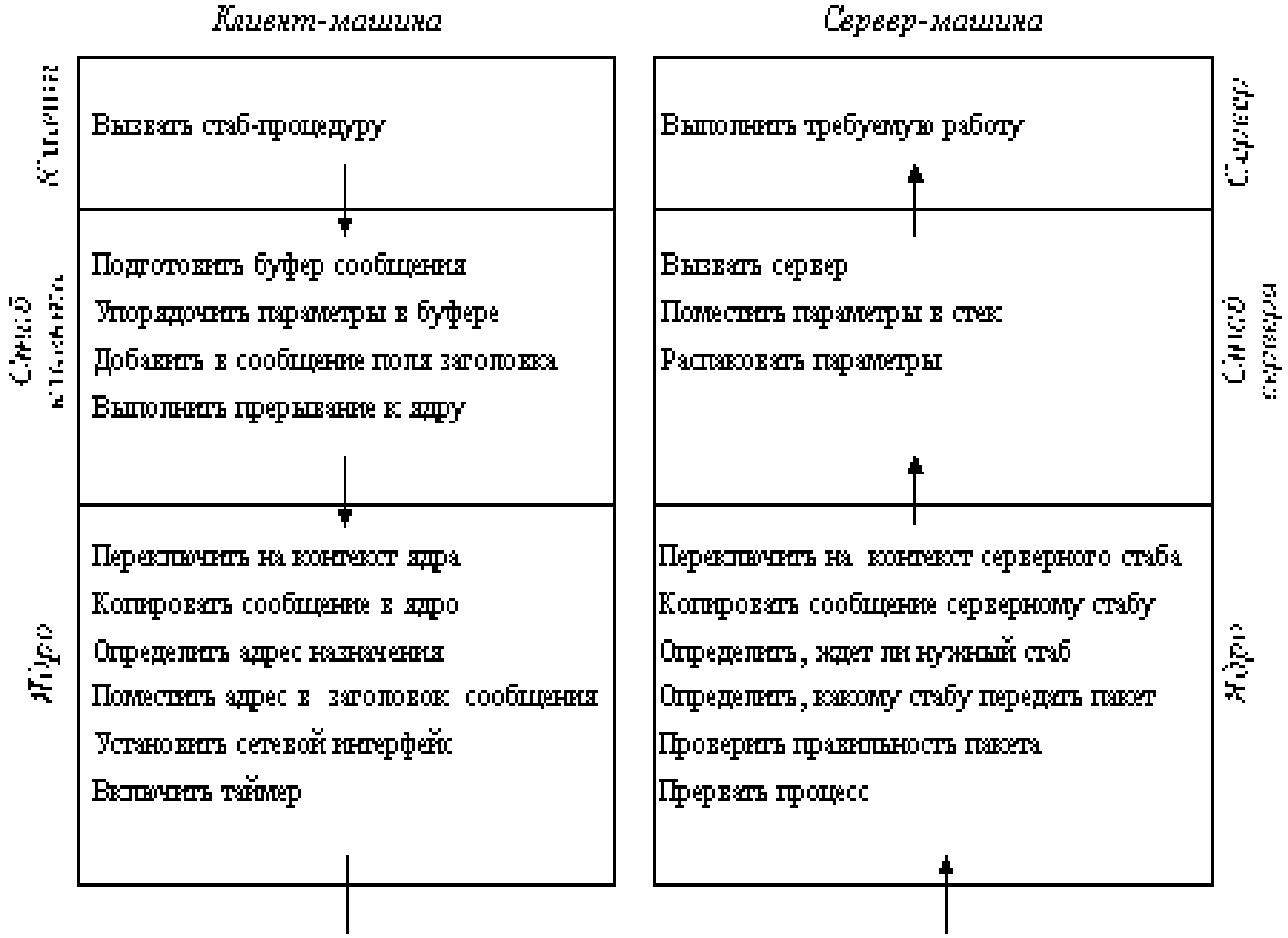


а) Стек до выполнения вызова read; б) Стек во время выполнения процедуры; в) Стек после возврата в вызывающую программу

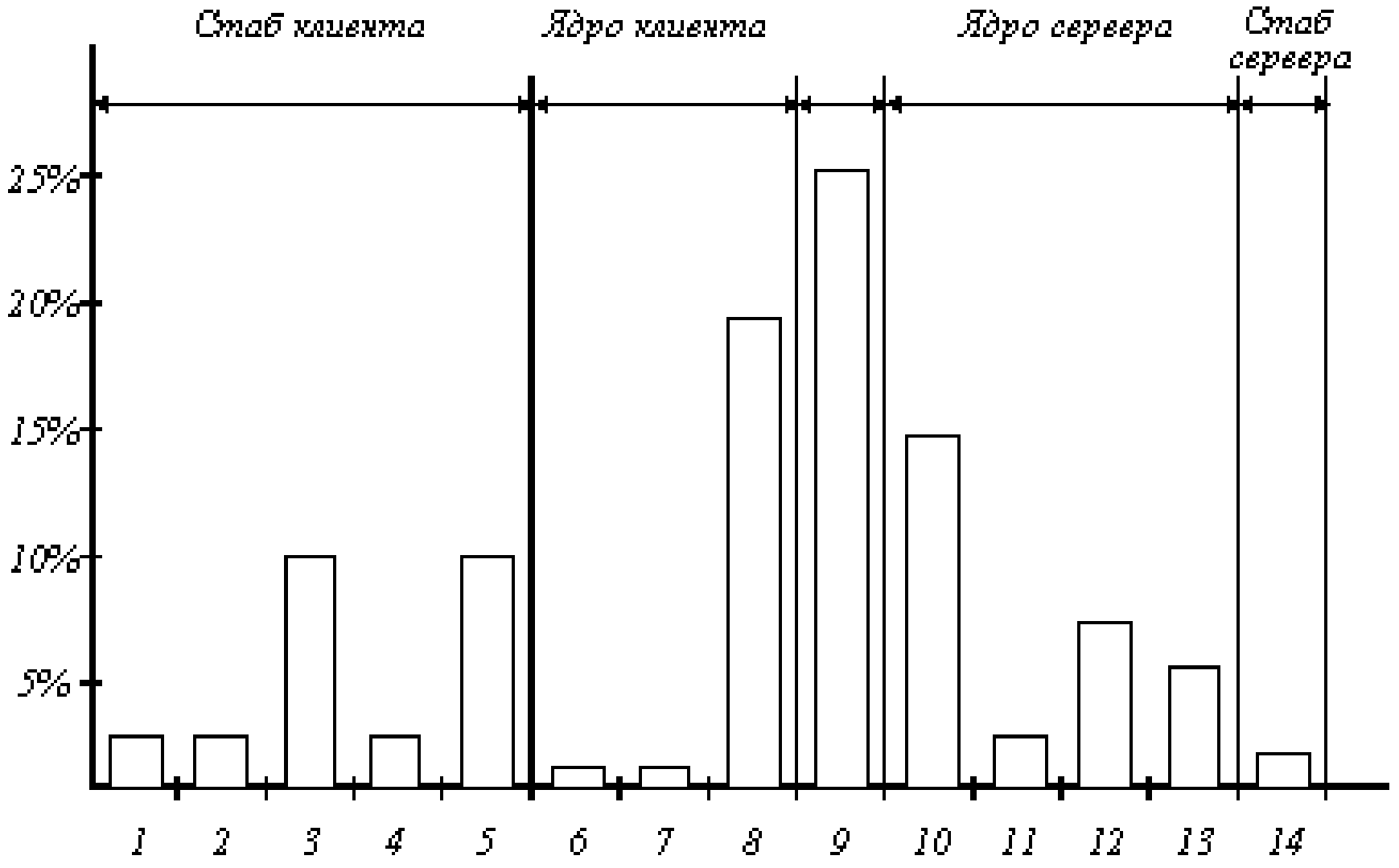
Этапы выполнения RPC



Этапы выполнения процедуры RPC



Распределение времени между 14 этапами выполнения RPC



Удаленный вызов процедур

1. ***Вызов стаба***
2. ***Подготовить буфер***
3. ***Упаковать параметры***
4. ***Заполнить поле заголовка***
5. ***Вычислить контрольную сумму в сообщении***
6. ***Прерывание к ядру***
7. ***Очередь пакета на выполнение***
8. ***Передача сообщения контроллеру по шине QBUS***
9. ***Время передачи по сети Ethernet***
10. ***Получить пакет от контроллера***
11. ***Процедура обработки прерывания***
12. ***Вычисление контрольной суммы***
13. ***Переключение контекста в пространство пользователя***
14. ***Выполнение серверного стаба***

Семантика RPC в случае отказов

Классы отказов:

1. Клиент не может определить местонахождения сервера, например, в случае отказа нужного сервера, или из-за того, что программа клиента была скомпилирована давно и использовала старую версию интерфейса сервера.
2. Потерян запрос от клиента к серверу.
3. Потеряно ответное сообщение от сервера клиенту.
4. Сервер потерпел аварию после получения запроса.
5. Клиент потерпел аварию после отсылки запроса.

У RPC как средства организации сетевого взаимодействия есть ряд минусов, кроющихся в самой парадигме структурного программирования:

- Синхронные запросы — RPC приостанавливает выполнение приложения до тех пор, пока сервер не вернет требуемые данные.
- Статические отношения между компонентами — привязка клиентского процесса к серверным заглушкам происходит на этапе компиляции и не может быть изменена во время выполнения.

Для устранения этих недостатков были разработаны более совершенные механизмы реализации RPC:

- асинхронные RPC, позволяющие избежать приостановки выполнения клиентского приложения посредством функций обратного вызова (call-back functions);
- код заглушек вынесен в динамически подключаемые библиотеки.

Сервисы обработки сообщений

Сервисы обработки сообщений (*MOM — Message-oriented middleware*) — это системы, как правило ***асинхронные*** (в отличие от RPC), в которых взаимодействие между клиентом и сервером основано на обмене сообщениями.

Сообщения — это текстовые блоки, состоящие из управляющих команд и передаваемых данных. Для передачи сообщений используются байт-ориентированные протоколы, такие как HTTP, POP/SMTP и т.п.

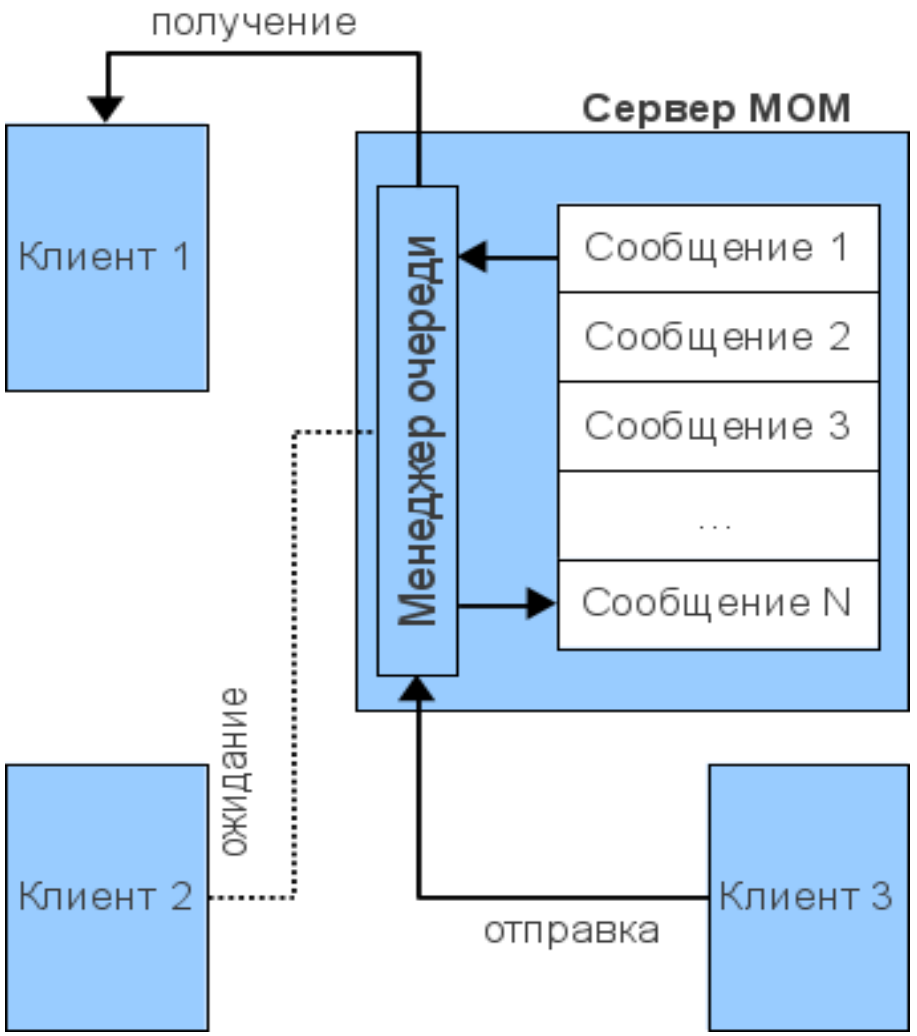
Обмен сообщениями реализуется через API системы MOM.

Запросы сервисов ставятся в ***очередь сообщений*** и обрабатываются в соответствии с приоритетами и доступностью ресурсов.

Приоритеты сообщений позволяют обеспечить первоочередную доставку ***важных сообщений***, а ***отложенная доставка*** осуществляется либо по расписанию, либо при появлении адресата в сети.

Ответы сервера содержат информацию об успешном или неуспешном выполнении операции.

Сервисы обработки сообщений



Промежуточное ПО, ориентированное на обработку сообщений (МОМ)

Сервисы обработки сообщений

Сервисы МОМуспешно используются в сильно распределенных приложениях, используемых в гетерогенной сети с медленными и ненадежными соединениями. Это, во-многом, достигается благодаря поддержке уровней «качества обслуживания» (Quality of service, QoS):

- **надежная доставка сообщений** (reliable message delivery) — система МОМ гарантирует, что в процессе обмена ни одно сообщение не будет утеряно;
- **гарантированная доставка сообщений** (guaranteed message delivery) — сообщение доставляется адресату немедленно или через заданный промежуток времени, не превышающий определенного значения (в случае, если сеть в данный момент не доступна);
- **застрахованная доставка сообщений** (assured message delivery) — каждое сообщение доставляется только один раз.

Сервисы обработки сообщений

Помимо классической схемы МОМ с очередями, используются сервисы МОМ с ***непосредственной передачей сообщений*** и на основе ***подписки***.

Системы с непосредственной передачей сообщений (message passing) используют логическое сетевое соединение для обмена сообщениями между взаимодействующими приложениями.

Сервисы МОМ, обслуживающие клиентов по подписке/публикации (publish&subscribe) работают по принципу, напоминающему почтовую рассылку: одно приложение публикует информацию в сети, а другие подписываются на эту публикацию для получения необходимых данных.

Очереди сообщений представляют собой мощный, гибкий и в то же время простой механизм межпрограммного взаимодействия.

Мониторы обработки транзакций

Мониторы обработки транзакций

Мониторы обработки транзакций (*Transaction Processing monitors*, TP-monitors) — это промежуточное программное обеспечение, обеспечивающее контроль передачи данных от клиента при работе с распределенными базами данных.

Монитор транзакций обеспечивает целостность данных, следя за тем, чтобы не было потерянных или незавершенных транзакций.

Транзакция – это законченный блок обращений к базе данных, для которого гарантируется выполнение требований атомарности (Atomicity), согласованности (Consistency), изолированности (Isolation) и долговременности (Durability).

Монитор транзакций обеспечивает контроль над выполнением этих условий, выполняя функции концентрации и перенаправления запросов к БД в распределенной среде с множеством баз данных от различных поставщиков

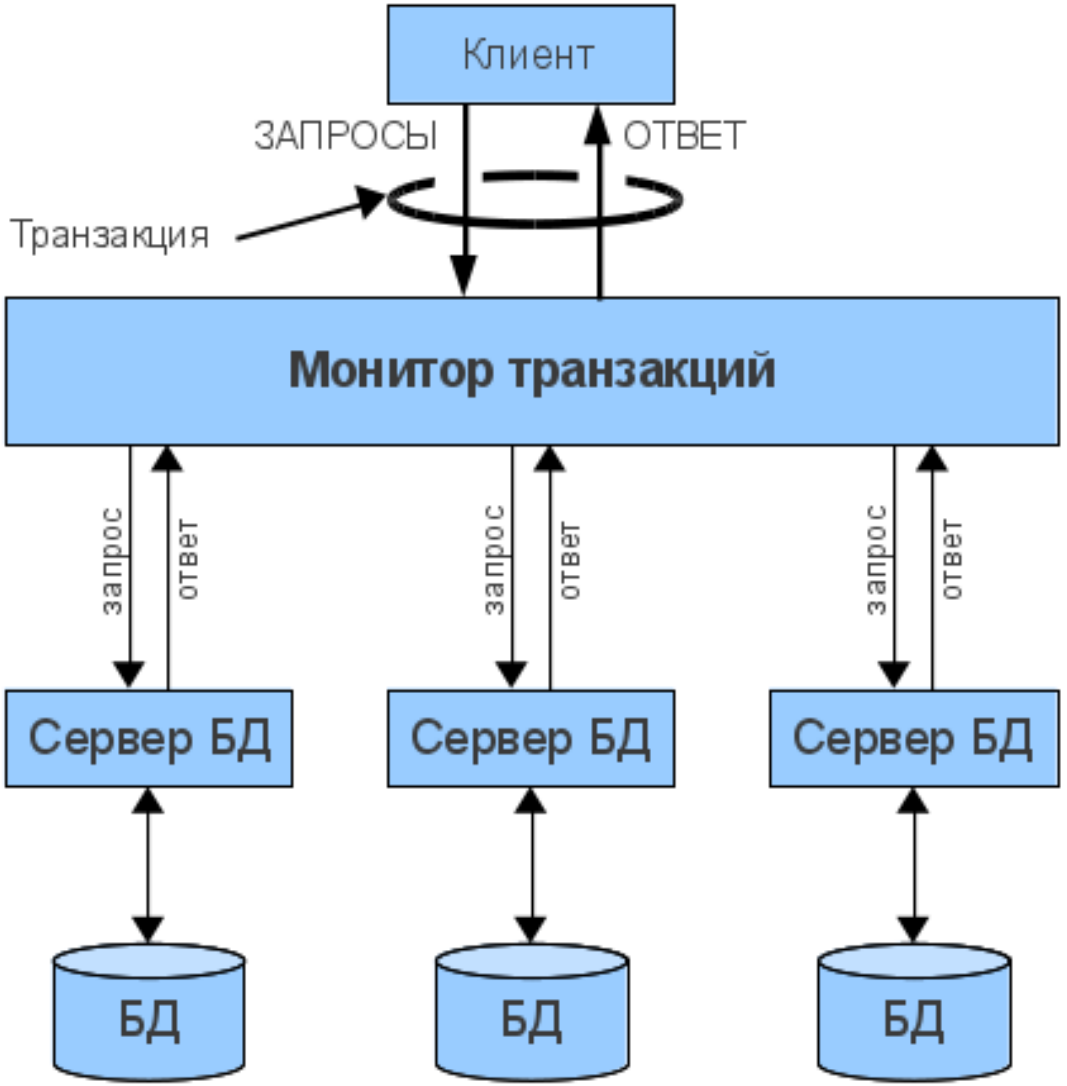
Мониторы обработки транзакций

Транзакция – это законченный блок обращений к базе данных, для которого гарантируется выполнение :

- **Атомарность** (**A**tomicity) – операции транзакции образуют неделимый, атомарный блок, который либо выполняется от начала до конца, либо не выполняется вообще. При невозможности выполнения транзакции происходит откат к исходному состоянию;
- **Согласованность** (**C**onsistency) – по завершении транзакции все задействованные ресурсы находятся в предопределенном и согласованном состоянии;
- **Изолированность** (**I**solation) – одновременный доступ транзакций различных приложений к разделяемым ресурсам координируется таким образом, чтобы исключить влияние транзакций друг на друга;
- **Долговременность** (**D**urability) – все модификации ресурсов в процессе выполнения транзакции будут долговременными.

ACID

Мониторы обработки транзакций



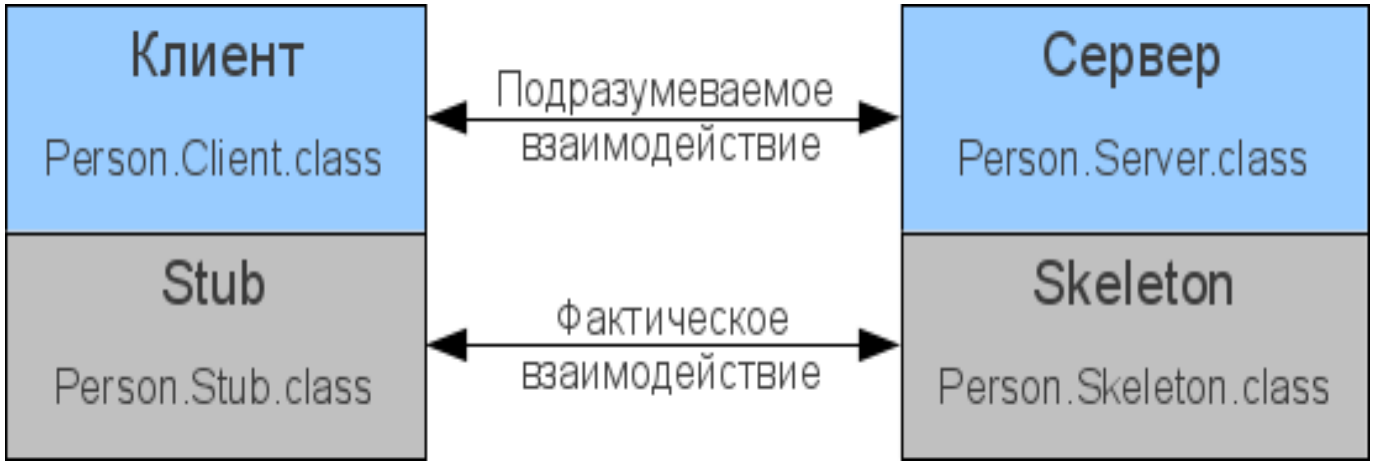
Распределенные объектные системы

Распределенные объектные системы (Distributed object systems) — это промежуточное программное обеспечение, реализованное в виде взаимодействующих друг с другом программных объектов.

Каждый такой объект уникальным образом идентифицируется в сети и обеспечивает доступ к представляемым им сервисам через публичные свойства и методы.

Реализация объекта и платформа, на которой он выполняется, полностью прозрачны для клиента.

Общий принцип взаимодействия распределенных объектов очень похож на RPC. Однако, в сравнении с RPC, распределенные объекты могут компоноваться ***динамически***, т.е. не на этапе компиляции, а во время выполнения приложений.



Архитектура распределенных объектных систем стандартизована и наиболее распространены спецификации CORBA, COM/DCOM и EJB.

CORBA (*Common Object Request Broker Architecture*, типовая архитектура брокера объектных запросов) — открытый стандарт, разработанный группой Object Management Group (OMG), который определяет интерфейсы между сетевыми объектами, позволяющие им работать совместно.

Microsoft COM (*Component Object Model*, компонентная объектная модель) — это семейство технологий, предназначенных для организации взаимодействия Windows-приложений. В это семейство входят COM+, DCOM (Distributed COM) и ActiveX Controls.

EJB (Enterprise JavaBeans) — технология, разработанная Sun Microsystems для корпоративных решений на платформе Java (Java EE/EJB). Спецификация EJB описывает архитектуру серверных компонентов и порядок их использования в клиент-серверных приложениях.