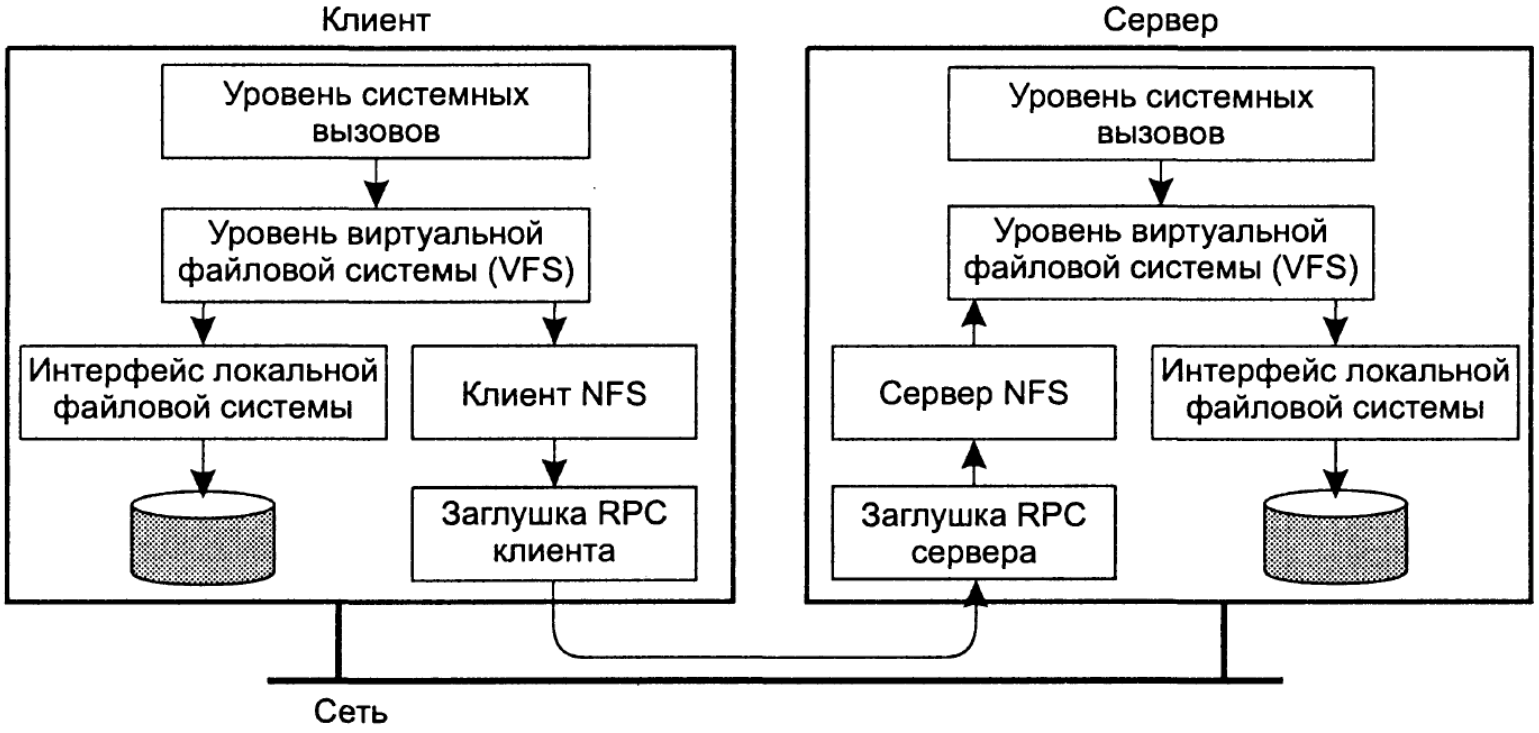
Распределенные файловые системы

Распределенные файловые системы

- Традиционная централизованная файловая система позволяет множеству пользователей, работающих на одной системе, разделять доступ к файлам, хранящихся локально на этой машине.
- Распределенная файловая система расширяет эти возможности, позволяя разделять доступ к файлам пользователям на разных машинах, объединенных между собой с помощью сети.
- В основе распределенных файловых систем лежит модель клиент-сервер. В данном случае под клиентом понимается машина, которая обращается к некоторому файлу, а под сервером - машина, хранящая файлы и обеспечивающая к ним доступ. Некоторые системы требуют, чтобы клиенты и серверы были разными машинами, в то время как другие допускают, чтобы одна машина работала и как клиент, и как сервер.

Свойства распределенных файловых систем

- **Сетевая прозрачность.** Клиенты должны иметь возможность обращаться к удаленным файлам пользуясь теми же самыми операциями, что и для доступа к локальным файлам.



Свойства

- **Устойчивость к сбоям, высокая доступность.** Система должна продолжать функционировать при неисправности отдельного компонента (сервера или сегмента сети). Однако это может приводить к деградации производительности или к исключению доступа к некоторой части файловой системы.
- **Прозрачность размещения.** Имя файла не должно определять его местоположения в сети.
- **Независимость размещения.** Имя файла не должно меняться при изменении его физического месторасположения.
- **Мобильность пользователя.** Пользователи должны иметь возможность обращаться к разделяемым файлам из любого узла сети.
- **Масштабируемость.** Система должна обладать возможностью масштабирования в случае увеличения нагрузки. Кроме того, должна существовать возможность постепенного наращивания системы путем добавления отдельных компонентов.
- **Мобильность файлов.** Должна быть возможность перемещения файлов из одного месторасположения в другое на работающей системе.

Файловый сервис

- Файловый сервис - это описание функций, которые файловая система предлагает своим пользователям. Это описание включает имеющиеся примитивы, их параметры и функции, которые они выполняют.

- С точки зрения пользователей файловый сервис определяет то, с чем пользователи могут работать, но ничего не говорит о том, как все это реализовано. В сущности, файловый сервис определяет интерфейс файловой системы с клиентами.

Файловый сервер

- Файловый сервер - это процесс, который реализует файловый сервис. В системе может быть один файловый сервер или несколько, но в хорошо организованной распределенной системе пользователи не знают, как реализована файловая система. В частности, они не знают количество файловых серверов, их месторасположение и функции. Они только знают, что если процедура определена в файловом сервисе, то требуемая работа каким-то образом выполняется, и им возвращаются требуемые результаты. Более того, пользователи даже не должны знать, что файловый сервис является распределенным. В идеале он должен выглядеть также, как и в централизованной файловой системе.
- Так как обычно файловый сервер - это просто пользовательский процесс (или иногда процесс ядра), выполняющийся на некоторой машине, в системе может быть несколько файловых серверов, каждый из которых предлагает различный файловый сервис. Например, в распределенной системе может быть два сервера, которые обеспечивают файловые сервисы систем UNIX и MS-DOS соответственно, и любой пользовательский процесс пользуется подходящим сервисом.

Примеры

- NFS — разработка компании Sun Microsystem, имеющая гигантское число установок и в настоящее время постепенно разрастающаяся до глобальных масштабов.
- Coda — потомок файловой системы AFS, крупномасштабной системы, которая разрабатывалась исключительно с целью обеспечить максимальную масштабируемость.
- Plan 9 — это распределенная система, в которой все ресурсы рассматриваются как файлы.
- xFS — это распределенная система, в которой отсутствуют серверы, а файловую систему реализуют клиенты.
- SFS, которая выделяется среди других распределенных файловых систем масштабируемой системой защиты.

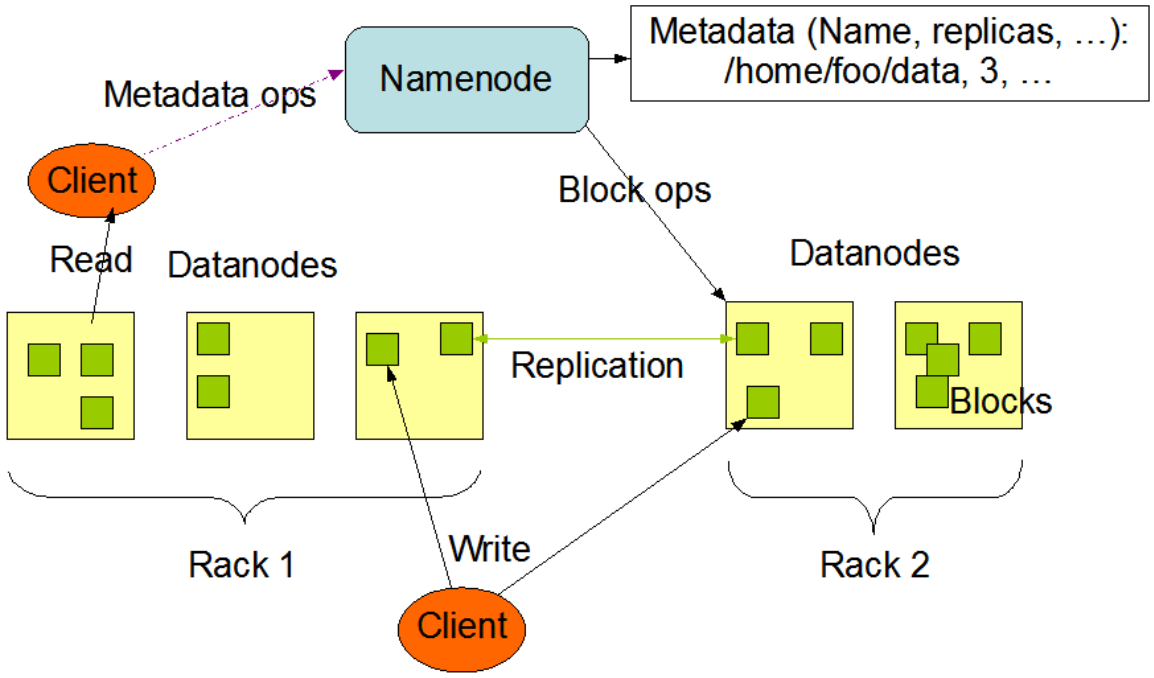
Исследование использования файлов

- Большинство файлов имеют размер менее 10К. => Следует перекачивать целиком.
- Чтение встречается гораздо чаще записи. => Кэширование.
- Чтение и запись последовательны, произвольный доступ редок.

  => Упреждающее кэширование, чтение с запасом, выталкивание после записи следует группировать.
- Большинство файлов имеют короткое время жизни. => Создавать файл в клиенте и держать его там до уничтожения.
- Мало файлов разделяются. => Кэширование в клиенте и семантика сессий.
- Существуют различные классы файлов с разными свойствами. => Следует иметь в системе разные механизмы для разных классов.

Архитектура распределенных файловых систем

- Распределенная файловая система обычно имеет два существенно отличающихся компонента - непосредственно файловый сервер и сервер директорий.



Интерфейс файлового сервера

- Для любой файловой системы первый фундаментальный вопрос - *что такое файл*.
- Во многих системах, таких как UNIX и MS-DOS, файл - не интерпретируемая последовательность байтов.
- На некоторых централизованных ЭВМ (IBM/370) файл представляется как последовательность записей, которую можно специфицировать ее номером или содержимым некоторого поля (ключом).
- Так, как большинство распределенных систем базируются на использовании среды UNIX, то они используют первый вариант понятия файла.

Интерфейс файлового сервера

- Файл может иметь *атрибуты* (информация о файле, не являющаяся его частью).

TYPE. Тип файла (обычный, каталог, символическая ссылка и т. д.)

SIZE. Длина файла в байтах.

CHANGE. Индикатор, который позволяет клиенту обнаружить, изменялся ли файл и/или когда он изменялся.

FSID. Уникальный идентификатор файловой системы сервера, на котором хранится файл.

ACL. Список контроля доступа, ассоциированный с файлом

FILEHANDLE. Дескриптор данного файла, установленный сервером

FILEID. Уникальный идентификатор файла в файловой системе

FST_LOCATIONS. Место расположения файловой системы в сети

OWNER. Имя владельца файла в виде символьной строки

TIME ACCESS. Время последнего доступа к содержимому файла

TIME_MODIFY. Время последнего изменения содержимого файла

TIME_CREATE. Время создания файла

Именованные атрибуты хранятся в виде массива пар {атрибут, значение).

Интерфейс файлового сервера

- Важный аспект файловой модели - могут ли файлы *модифицироваться* после создания. Обычно могут, но есть системы с неизменяемыми файлами. Такие файлы освобождают разработчиков от многих проблем при кэшировании и размножении.
- *Защита* обеспечивается теми же механизмами, что и в однопроцессорных ЭВМ - мандатами и списками прав доступа. Мандат - своего рода билет, выданный пользователю для каждого файла с указанием прав доступа. Список прав доступа задает для каждого файла список пользователей с их правами.

  Простейшая схема с правами доступа - UNIX схема, в которой различают три типа доступа (чтение, запись, выполнение), и три типа пользователей (владелец, члены его группы, и прочие).

Контроль доступа в NFS

Операция	Описание
read_data	Право на чтение данных файла
write_data	Право на изменение данных файла
append_data	Право на добавление данных в файл
execute	Право на выполнение файла
list_directory	Право на получение содержимого каталога
add_file	Право на добавление в каталог нового файла
add_subdirectory	Право на создание вложенного каталога внутри каталога
delete	Право на удаление файла
delete_child	Право на удаление файла или вложенного каталога из каталога

Контроль доступа в NFS

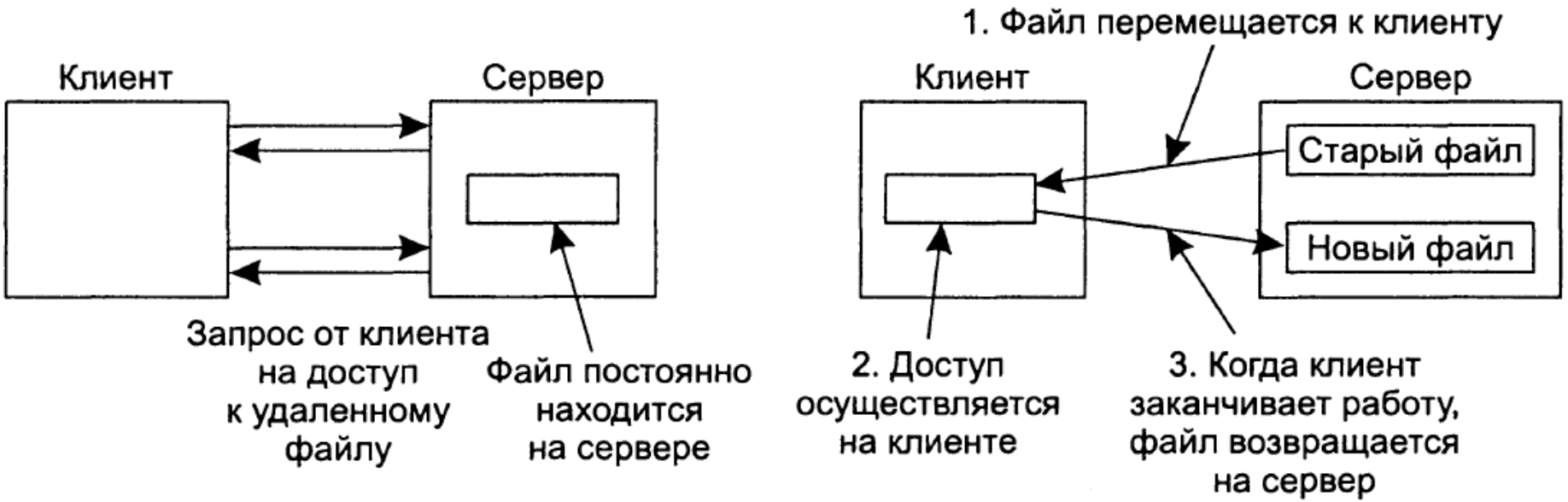
Операция	Описание
read_acl	Право на чтение списка ACL
write_acl	Право на запись в список ACL
read_attributes	Право на чтение других основных атрибутов файла
write_attributes	Право на изменение других основных атрибутов файла
read_named_attrs	Право на чтение именованных атрибутов файла
write_named_attrs	Право на запись именованных атрибутов файла
write_owner	Право на смену владельца
synchronize	Право на локальный доступ к файлу на сервере с синхронным чтением и записью

Контроль доступа в NFS

Тип пользователя	Описание
Owner	Владелец файла
Group	Группа пользователей, ассоциированных с файлом
Everyone	Любой пользователь или процесс
Interactive	Любой процесс, имеющий доступ к файлу через интерактивный терминал
Network	Любой процесс, имеющий доступ к файлу через сеть
Dialup	Любой процесс, имеющий доступ к файлу через коммутируемое соединение с сервером
Batch	Любой процесс, имеющий доступ к файлу в составе пакетного задания
Anonymous	Всякий, кто получает доступ к файлу без аутентификации
Authenticated	Всякий аутентифицированный пользователь или процесс
Service	Всякий служебный системный процесс

Интерфейс файлового сервера

- Файловый сервис может базироваться на одной из двух моделей - модели *удаленного доступа* и модели *загрузки/разгрузки*.



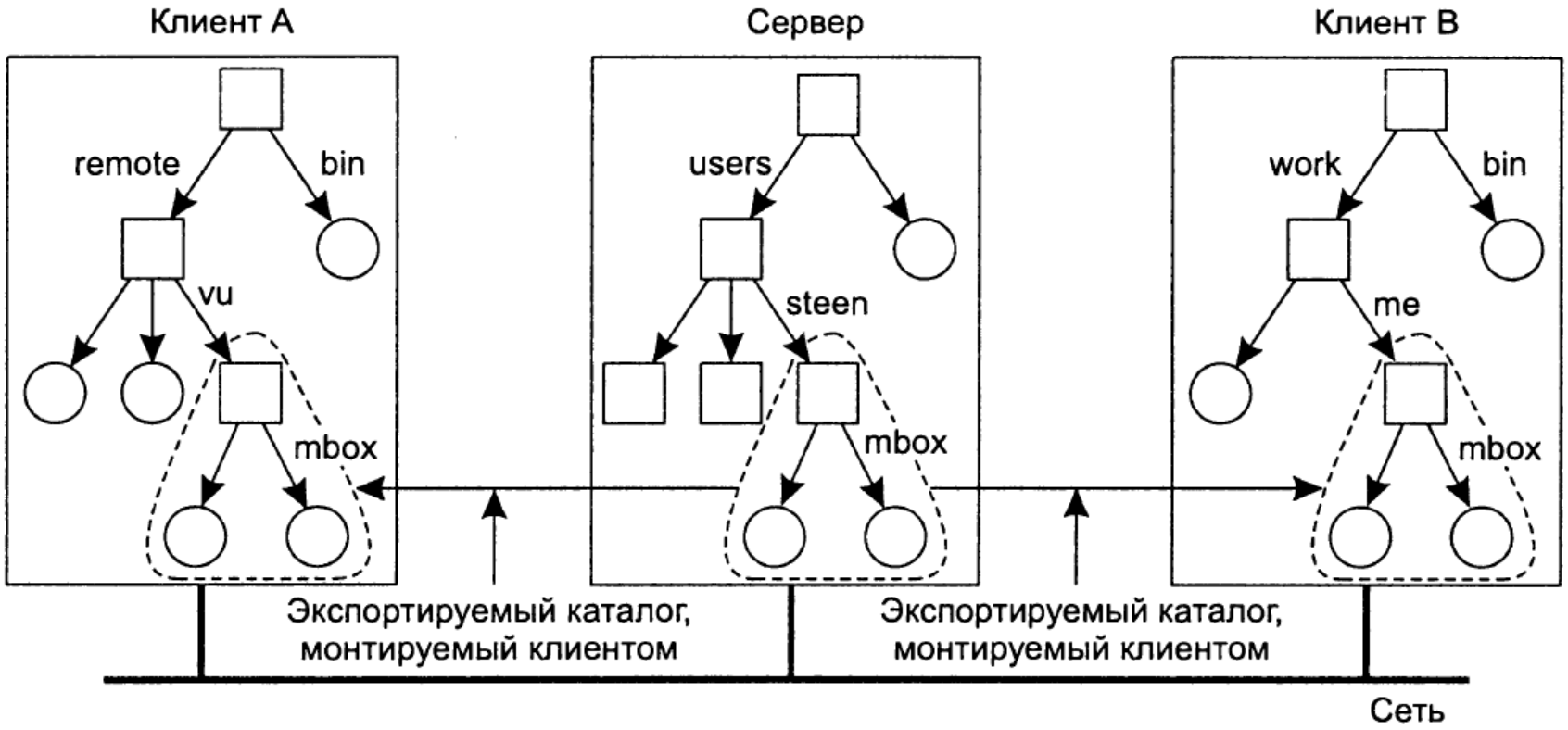
Интерфейс сервера директорий

- Обеспечивает операции создания и удаления директорий, именования и переименования файлов, перемещение файлов из одной директории в другую.
- Определяет алфавит и синтаксис имен. Для спецификации ***типа*** информации в файле используется часть имени (расширение) либо явный атрибут.
- Все распределенные системы позволяют директориям содержать поддиректории - такая файловая система называется ***иерархической***. Некоторые системы позволяют создавать указатели или ссылки на произвольные директории, которые можно помещать в директорию. При этом можно строить не только деревья, но и произвольные графы. Разница между ними очень важна для распределенных систем, поскольку в случае графа удаление связи может привести к появлению недостижимых поддеревьев, обнаруживать которые в распределенных системах очень трудно.
- Ключевое решение при конструировании распределенной файловой системы - должны или не должны машины (или процессы) одинаково видеть иерархию директорий. Тесно связано с этим решением наличие единой корневой директории.

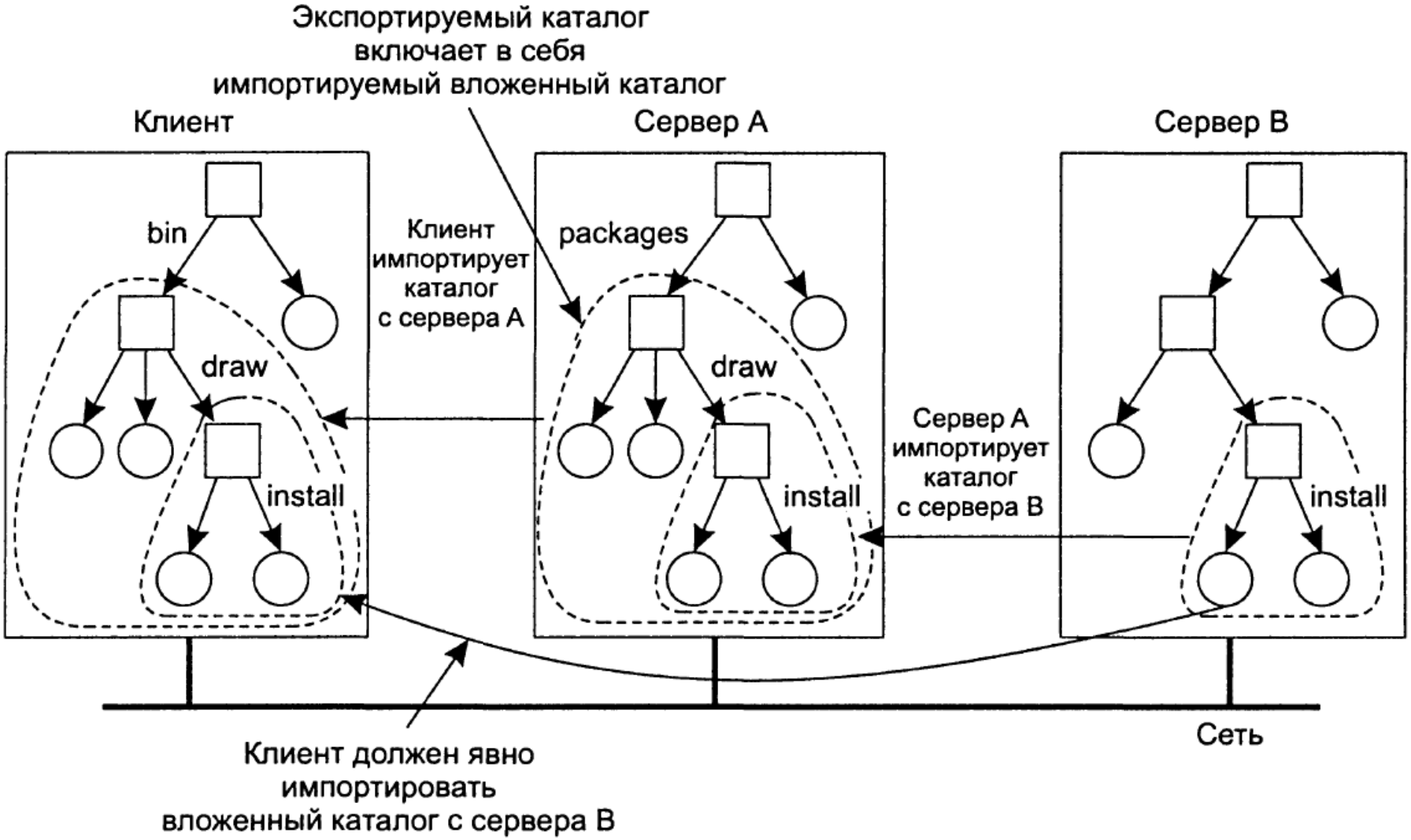
Подходы к именованию файлов

- Машина + путь (/server/d1/f1);
- Монтирование удаленных файловых систем в локальную иерархию файлов;
- Единственное пространство имен, которое выглядит одинаково на всех машинах.

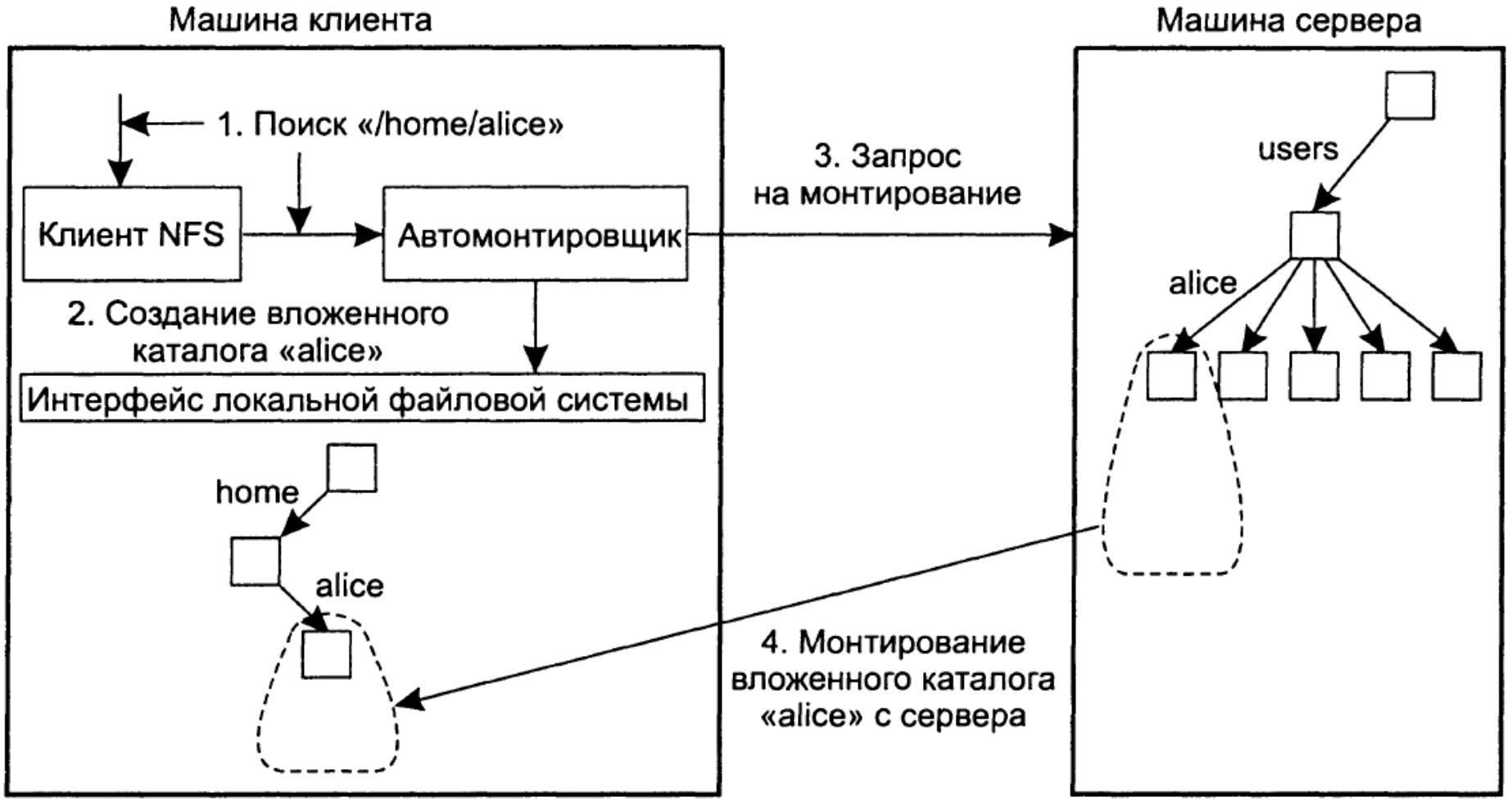
Монтирование части удаленной файловой системы в NFS



Монтирование вложенных каталогов с нескольких серверов NFS



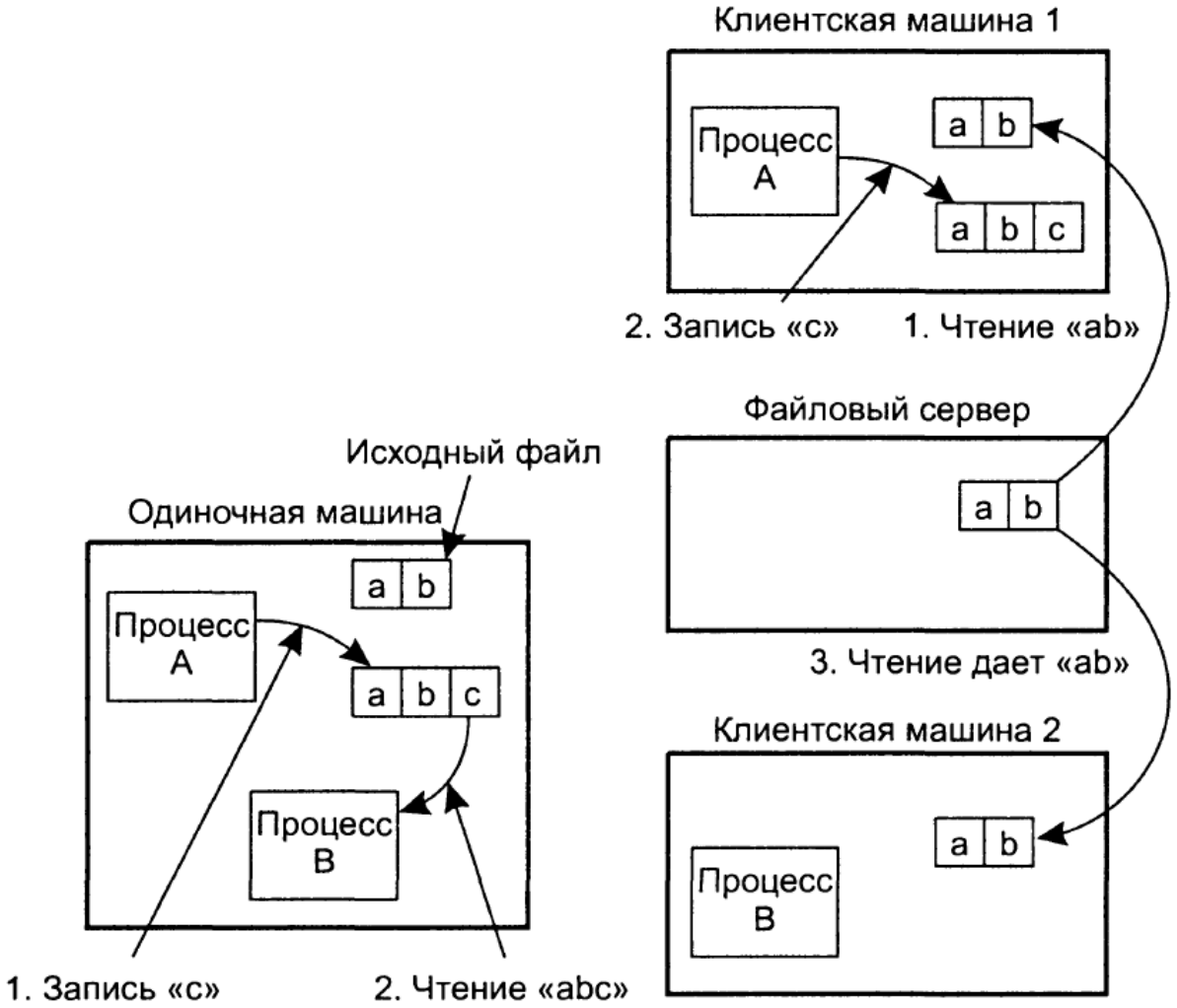
Автоматическое монтирование



Двухуровневое именование

- Файлы (и другие объекты) имеют символические имена для пользователей, но могут также иметь внутренние двоичные имена для использования самой системой. Например, в операции открыть файл пользователь задает символическое имя, а в ответ получает двоичное имя, которое и использует во всех других операциях с данным файлом.
- Способы формирования двоичных имен различаются в разных системах:
  - имя может указывать на сервер и файл;
  - в качестве двоичных имен при просмотре символьных имен возвращаются мандаты, содержащие помимо прав доступа либо физический номер машины с сервером, либо сетевой адрес сервера, а также номер файла.
- В ответ на символьное имя некоторые системы могут возвращать несколько двоичных имен (для файла и его дублей), что позволяет повысить надежность работы с файлом.

Семантика разделения файлов



Семантика разделения файлов

- ***UNIX-семантика***. Естественная семантика однопроцессорной ЭВМ - если за операцией записи следует чтение, то результат определяется последней из предшествующих операций записи. В распределенной системе такой семантики достичь легко только в том случае, когда имеется один файл-сервер, а клиенты не имеют кэшей. При наличии кэшей семантика нарушается. Надо либо сразу все изменения в кэшах отражать в файлах, либо менять семантику разделения файлов.

  Еще одна проблема - трудно сохранить семантику общего указателя файла (в UNIX он общий для открывшего файл процесса и его дочерних процессов) - для процессов на разных ЭВМ трудно иметь общий указатель.
- ***Неизменяемые файлы*** - очень радикальный подход к изменению семантики разделения файлов.

  Только две операции - создать и читать. Можно заменить новым файлом старый - т.е. можно менять директории. Если один процесс читает файл, а другой его подменяет, то можно позволить первому процессу доработать со старым файлом, в то время как другие процессы могут уже работать с новым.

Семантика разделения файлов

- ***Семантика сессий.*** Изменения открытого файла видны только тому процессу (или машине), который производит эти изменения, а лишь после закрытия файла становятся видны другим процессам (или машинам). Что происходит, если два процесса одновременно работали с одним файлом - либо результат будет определяться процессом, последним закрывшим файл, либо можно только утверждать, что один из двух вариантов файла станет текущим.
- ***Транзакции.*** Процесс выдает операцию «НАЧАЛО ТРАНЗАКЦИИ», сообщая тем самым, что последующие операции должны выполняться без вмешательства других процессов. Затем выдает последовательность чтений и записей, заканчивающуюся операцией «КОНЕЦ ТРАНЗАКЦИИ». Если несколько транзакций стартуют в одно и то же время, то система гарантирует, что результат будет таким, каким бы он был в случае последовательного выполнения транзакций (в неопределенном порядке).

Семантика разделения файлов

Метод	Описание
Семантика UNIX	Каждая операция с файлом немедленно становится видна всем процессам
Семантика сессий	Изменения невидимы для других процессов до закрытия файла
Неизменяемые файлы	Изменения невозможны, упрощены разделение и репликация
Транзакции	Все изменения происходят атомарно

Файловые серверы с состоянием

- Короче сообщения (двоичные имена используют таблицу открытых файлов).
- Выше эффективность (информация об открытых файлах может храниться в оперативной памяти).
- Блоки информации могут читаться с упреждением.
- Убедиться в достоверности запроса легче, если есть состояние (например, хранить номер последнего запроса).
- Возможна операция захвата файла.

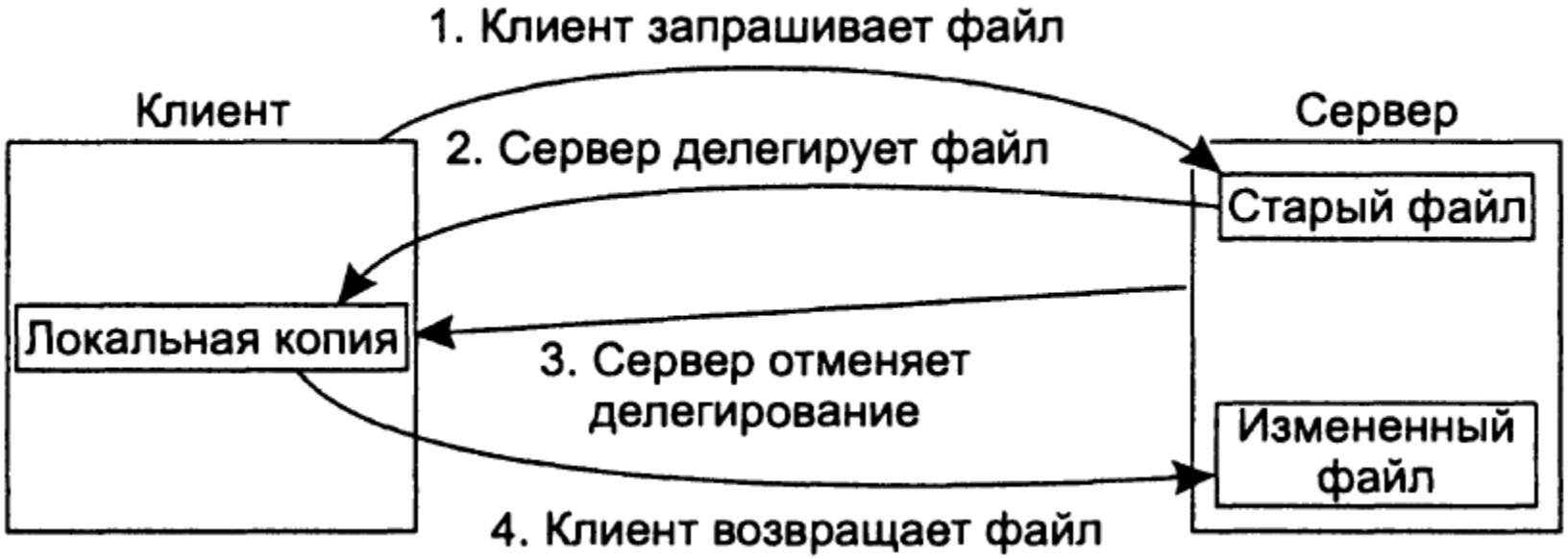
  Пример, NFS версии 4.

Файловые серверы без состояния

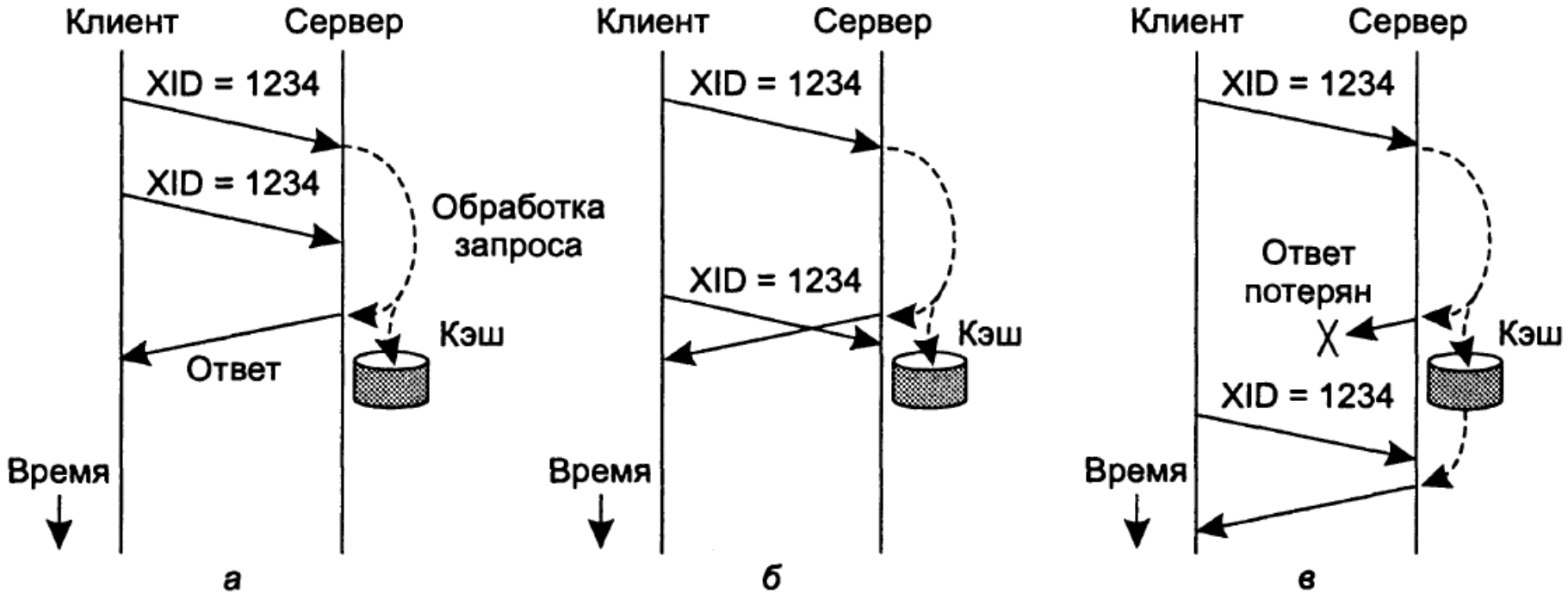
- Устойчивость к ошибкам.
- Не требуется операций ОТКРЫТЬ/ЗАКРЫТЬ.
- Не требуется память для таблиц.
- Нет ограничений на число открытых файлов.
- Нет проблем при крахе клиента.

Пример, NFS версии 2,3.

Делегирование прав клиенту в NFS версии 4



Кэш дублированных запросов



Захват файлов в NFS

Операция	Описание
lock	Блокировка набора байтов
lockt	Проверка, не назначена ли конфликтующая блокировка
locku	Снятие блокировки с набора байтов
renew	Продление аренды указанной блокировки

Блокировки устанавливаются на конкретное определяемое сервером время.
Если клиент не продлит аренду своей блокировки, сервер автоматически снимет ее.

Совместное резервирование файлов (share reservation)

- При открытии файла клиент определяет тип доступа, который требует для себя (READ, WRITE или BOTH), и тип доступа, в котором сервер должен отказать всем прочим клиентам (NONE, READ, WRITE или BOTH). Если сервер не в состоянии удовлетворить требования клиента, операция открытия файла завершается с ошибкой.

	Текущий тип запрещения			
Тип запраши-ваемого доступа	NONE	READ	WRITE	BOTH
READ	Успешно	Ошибка	Успешно	Ошибка
WRITE	Успешно	Успешно	Ошибка	Ошибка
BOTH	Успешно	Ошибка	Ошибка	Ошибка

	Тип запрашиваемого запрещения			
Текущий тип доступа	NONE	READ	WRITE	BOTH
READ	Успешно	Ошибка	Успешно	Ошибка
WRITE	Успешно	Успешно	Ошибка	Ошибка
BOTH	Успешно	Ошибка	Ошибка	Ошибка

Операции с файлами в NFS

Операция	v. 3	v. 4	Описание
create	Да	Нет	Создать обычный файл
create	Нет	Да	Создать нестандартный файл
link	Да	Да	Создать жесткую ссылку на файл
symlink	Да	Нет	Создать символическую ссылку на файл
mkdir	Да	Нет	Создать вложенный каталог в текущем каталоге
mknod	Да	Нет	Создать специальный файл
rename	Да	Да	Изменить имя файла
remove	Да	Да	Удалить файл из файловой системы
rmdir	Да	Нет	Удалить пустой вложенный каталог из каталога
open	Нет	Да	Открыть файл
close	Нет	Да	Закрыть файл
lookup	Да	Да	Найти файл по имени файла
readdir	Да	Да	Прочитать элементы в каталоге
readlink	Да	Да	Прочитать путь к файлу, записанный в символической ссылке
getattr	Да	Да	Получить значение атрибута файла
setattr	Да	Да	Установить значение одного или нескольких атрибутов файла
read	Да	Да	Прочитать данные, содержащиеся в файле
write	Да	Да	Записать данные в файл

Составные вызовы в NFS4