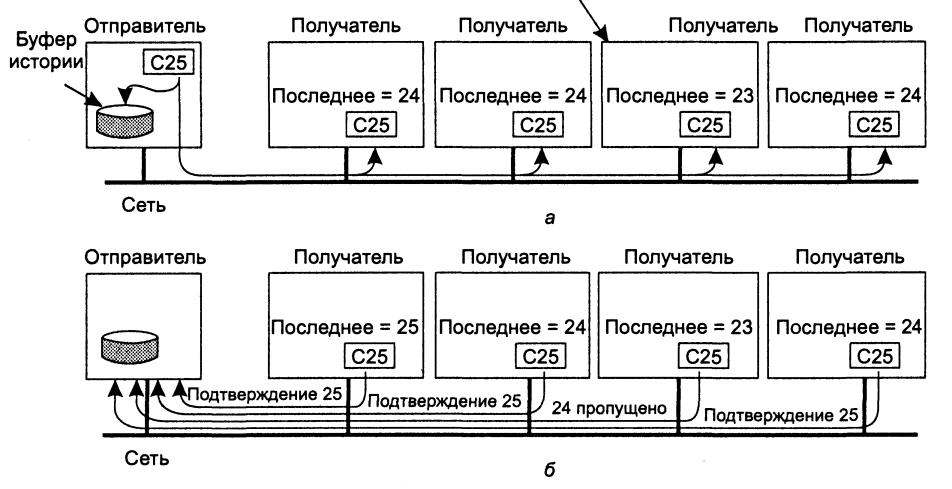
Лекция 11

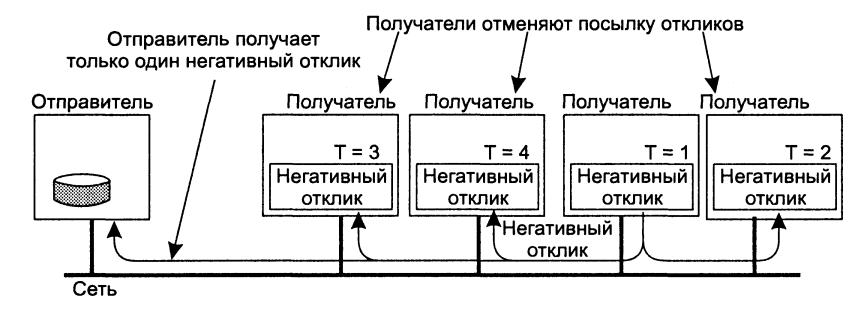
РИС, III курс, ИСиТ

**Распределенное подтверждение**

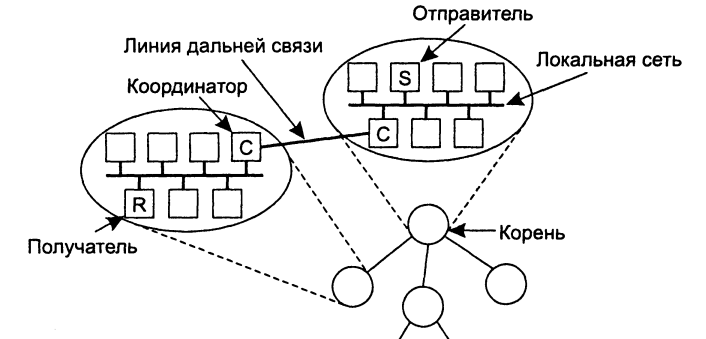
1. **Постановка задачи:** распределенная транзакция (двухфазная блокировка); как организовать распределенный COMMIT (distributed commit – распределенное подтверждение)? на одном из узлов РИС, может быть отказ, откатиться должны операции на всех узлах; как организовать распределенный ROLLBACK (distributed rollback – распределенный откат)?
2. **Координатор** –один из узлов распределенной системы, участвующих в распределенном подтверждении и управляющий этим процессом; как правило, координатором является инициатор распределенного подтверждения (узел, выполняющий распределенную транзакцию).
3. **Протокол однофазного подтверждения**: координатор выдал commit и сообщил всем участникам распределенной транзакции, что надо выполнить commit; если участник не может выполнить локальный commit, он не имеет механизма сообщить об этом.
4. **Протокол двухфазного подтверждения (2PC):**
5. координатор перед выполнением локального COMMIT, рассылает всем участникам сообщение **vote\_request** и переходит в состояние ожидания;
6. участник получив **vote\_request** проверяет может ли он выполнить COMMIT, если да, то возвращает **vote\_commit** координатору, если не может возвращает **vote\_rollback**;
7. координатор собирает ответы участников, если все **vote\_commit**, то рассылает всем участникам **global\_commit**; если есть хотя бы один участник ответил **vote\_rollback,** то рассылает всем участникам **global\_ rollback**;
8. каждый из проголосовавших участников ждет итогового сообщения координатора; если **global\_commit**, то выполняет COMMIT, если **global\_rollback**, то выполняет ROLLBACK.
9. Основное преимущество 2PC: участник может попробовать выполнить действие предусмотренное распределенной транзакцией. Например, участник может проверить соответствие операции констрейнтам, если соответствует, то **vote\_commit**, иначе **vote\_rollback.**
10. **Проблема 1:** участник выполнил операцию и ждет от координатора **vote\_request**, а его нет. Устанавливается timeout, после которого отсылается координатору **vote\_rollback** и выполняется локальный ROLLBACK.
11. **Проблема 2:**  координатор не получает ответов **vote\_commit/vote\_rollback** от всех участников. Устанавливается timeout, после которого отсылается всем участникам **global\_ rollback** и выполняется локальный ROLLBACK.
12. **Проблема 3:**  участник не получает ответ **global\_commit/global\_rollback** от координатора. Устанавливается timeout, после которого осуществляется опрос других участников (если они есть) если, кто-то получил **global\_commit**, то выполняется локальный COMMIT, иначе ROLLBACK.
13. **Процесс восстановления участника:** если на узле был сбой после отправки координатору **vote\_commit**, то после восстановления работоспособности узел должен решить, что делать COMMIT/ROLLACK; один из вариантов связаться с другим участником или координатором выяснить; для полноценного процесса восстановления требуется сохранение информации в журналах.
14. **Проблема 4:** координатор всем отравил **vote\_request** и сам сломался, то участники никак не смогут определить, что им дальше делать; обычно ждут восстановление координатора. Поэтому 2PC – протокол блокирующего подтверждения.
15. **Протокол трехфазного подтверждения (3PC)**: используется редко. Суть протокола: состояние координатора и любого из участников удовлетворяет следующим 2м условиям:
16. нет такого состояния, при котором возможен прямой переход в состояния COMMIT или ROLLBACK;
17. нет такого состояния, в котором невозможно принять итоговое решение, но возможен переход в COMMIT.
18. **Протокол трехфазного подтверждения (3PC)**:
19. **О**
20. **сновная особенность обеспечения надежности распределенных систем:** отсутствие достоверной информации о реальном состоянии удаленных компонент системы.
21. **Надежность распределенной системы:** можно рассматривать как надежность распределенныхсерверных процессов плюс надежности связи между ними (клиент-сервер).
22. **Типичные ошибки связи между клиентом и сервером:** 1)поломка; 2) пропуски; 4) ошибки синхронизации (в т. ч. дублирование)**.** Всеусилия направлены на маскировку ошибок.
23. **Сквозная передача данных:** P2P(Point to Point), например TCP – надежный канал: маскирует потери сообщений путем подтверждения о получении данных и повторной передачи, для пользователя ошибки прозрачны, поломка (обрыв связи) – не маскируется: формируется исключение.
24. **RPC (Remote Procedure Call):** 1) не найден сервер (исключение, обычно не маскируется, иногда указывается несколько серверов или failover-сервер сообщает о причине); 2) потерян запрос от клиента к серверу (timeout на подтверждение, повторная отправка, нумерация, чтобы сервер обнаружил дублирование); 3) поломка сервера после получения запроса (timeout и думает, что потерян запрос; повторный запрос, потом увеличивается timeout и повторяет несколько раз, потом решает, что сервер поломался и ждет от него уведомления о возобновлении работы; серверы печати – часто не могут распознать дублирование); 4) потеря ответного сообщения от сервера клиенту (timeout, клиент не знает: потерялся запрос или сломался сервер, банковский запрос на перевод денег – запрос выполнен?; сервер: повтор запроса или новый запрос?); 5) поломка клиента (клиентские сообщения-сироты, а) журнал на стороне клиента, истребление сирот; б) реинкарнация: при перезапуске сообщение о новой эпохе, все старые вычисления прекращаются; в) мягкая реинкарнация: после получения сообщения о новой эпохе сервер связывается с клиентом и пытается довыполнить начатые вычисления; г) каждому запросу – срок исполнения, если не уложится сервер требует повтора).
25. **Надежная групповая рассылка:** использовать отдельный надежный (например, TCP) канал для каждого получателя возможно только, если получателей мало.
26. **Надежная групповая рассылка:** протокол, который гарантировано доставляет сообщения всем работающим в группе серверам; если считать, что группа получателей не изменяется (серверы не входят и не выходят из группы), то все просто.

****

1. **Проблемы масштабируемости надежной групповой рассылки:** если много получателей, то много ответов – все необходимо обработать (обратный удар); можно снизить обратный удар - получатель отправляет только сообщения об ошибках, увеличивается буфер сообщений на отправителе;
2. **Неиерархическое управление обратной связью:** SRM (Scalable Reliable Multicasting) – протокол надежной групповой рассылки: нет подтверждения о успешного приема; широковещательный групповой отклик в случае потери – другие, если потеряли, не посылают сообщения, в идеальном случае до первичного отправителя дойдет только одно сообщение об потере, но в этом случае – очень много служебных сообщений, которые требуются для обработки.

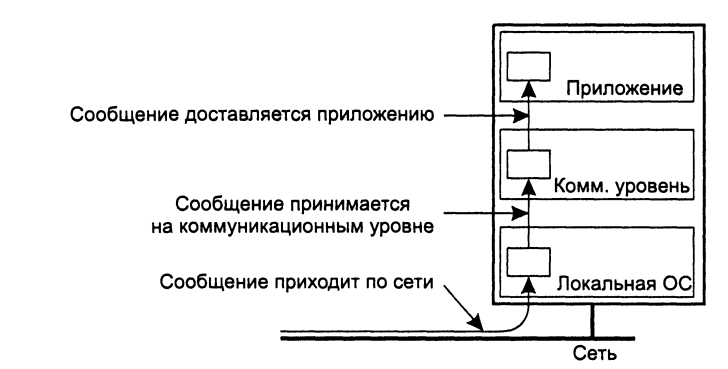


1. **Неиерархическое управление обратной связью:** статистика и перегруппировка получателей – получатели, часто имеющие ошибки в отдельную группу. Пример: групповое телевещание по истечении некоторого времени, работает лучше.
2. **Иерархическое управление обратной связью:** группа получателей разбита на множество подгрупп, организованных в виде дерева, внутри группы любая схема рассылки; у каждой группы – локальный координатор, который отвечает за обработку запросов на повторную передачу и поддерживает буфер истории.

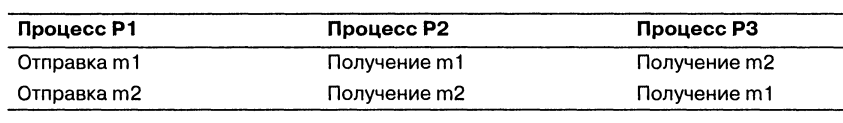


1. **Иерархическое управление обратной связью:** есликоординатор потерял сообщение, то запрашивает его у другого координатора; если кто-то в группе потерял сообщение – запрашивает его у своего координатора. Декомпозиция – способ решения сложных задач: сами координаторы образуют группу.

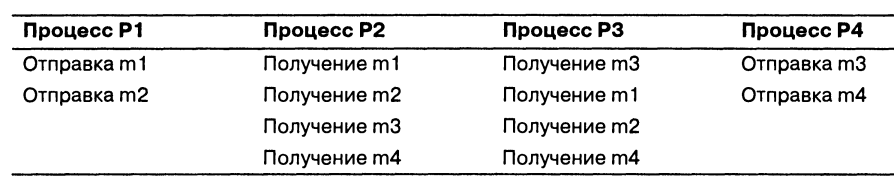
1. **Атомарная групповая рассылка:** 1) доставка сообщения либо всем, либо никому; 2) доставка сообщений получателям в определенном порядке.
2. **Атомарная групповая рассылка:** репликация - один издатель, группа подписчиков; выполнилось изменение на издателе, одна реплика сломалась, она исключается из группы, условие вхождения после восстановления в группу – синхронизация с группой.
3. **Виртуальная синхронность:** есть группа процессов получателей; снимок состава группы – представление группы в данный момент; надежная групповая рассылка внутри группы, сломанный получатель исключается из группы (не получает групповую рассылку, все члены группы уведомляются); исправленный процесс входит в группу и синхронизируется; все это реализуется специальной надстройкой над уровнем OC

****

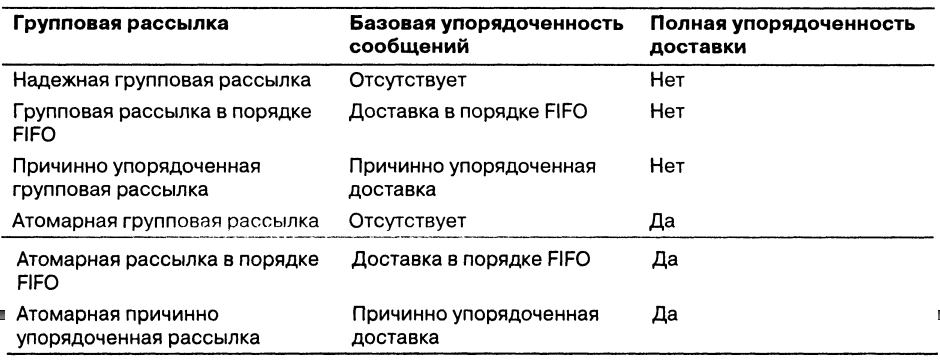
1. **Порядок сообщений:** хронологическая последовательность их отправки.
2. **Упорядочивание сообщений:** 1) неупорядоченные групповые рассылки; 2) FIFO-порядок; 3) причинно упорядоченные групповые рассылки; 4) полностью упорядоченные рассылки.
3. **Надежная неупорядоченная рассылка:** сообщения доходят, но порядок не гарантируется.

****

1. **Надежная FIFO-рассылка:** сообщения доходят в порядке их оправки.

****

1. **Надежная причинно упорядоченная рассылка:** если в группе серверов сообщение M является следствием сообщения S, то порядок их получения S, M.
2. **Полностью упорядоченная групповая рассылка:** не зависимо от порядка получения, все члены группы получают в одном порядке.
3. **Виды надежной групповой рассылки**

****

1. **Отказ информационной системы:** поведение информационной системы, не удовлетворяющее ее спецификации.
2. **Отказы:**

* проходные отказы – однократные;
* перемежающиеся отказы – появляются, пропадают и снова появляются с непредсказуемой периодичностью;
* постоянные отказы – появился и существует, пока не исправят.

1. **Ошибка информационной системы:** состояние информационной системы, которое может привести к отказу.
2. **Маскирование ошибок:** сокрытие ошибок в процессе от других процессов распределенной системы. Основной метод – применение избыточности: информационная избыточность, временная избыточность, физическая избыточность.
3. **Маскирование ошибок информационной избыточностью:** помехоустойчивое кодирование данных.
4. **Маскирование ошибок временной избыточностью:** повторение действия в случае ошибки.
5. **Маскирование ошибок физической избыточностью:** резервирование элементов системы (резервирование в авиации, кластеры - резервирование в информационных системах).
6. **Особенности отказа распределенной системы**: отказ в распределенной системе может быть частичным, отдельные ее компоненты могут утратить свою работоспособность, в то время, как другие могут полностью сохранить свою функциональность.
7. **Типы отказов в распределенной системе:**

* поломка - сервер перестал работать; остановка сервера; зависание ОС, требующее перезагрузку;
* пропуск данных – сервер неправильно реагирует на входные запросы (например, переполнение буферов соединений или данных, превышение timeout и пр. );
* ошибка синхронизации – реакция сервера происходит не в определенный интервал времени (например, на сервере взаимная блокировка процессов);
* ошибка отклика – в ответе сервера ошибка;
* произвольная ошибка сервера – сервер отправляет непредсказуемые сообщения в непредсказуемые моменты времени;

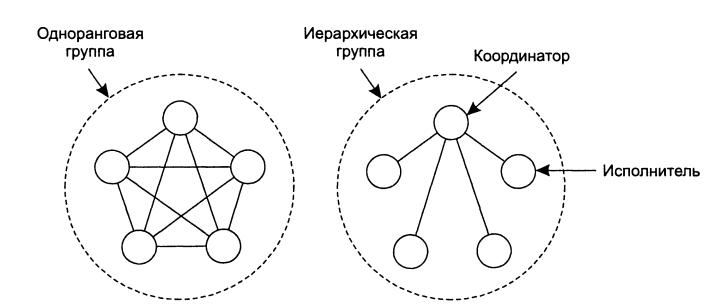
1. **Отказоустойчивость информационной системы:** свойство технической системы сохранять свою работоспособность после отказа одного или нескольких составных компонентов.
2. **Надежность информационной системы:**

* доступность (availability);
* безотказность (reliability);
* безопасность (safety);
* ремонтопригодность (maintainability).

1. **Доступность:** свойство системы находится в состоянии готовности к работе. Система с высокой степенью доступности – это система, которая в произвольный момент времени, вероятнее всего, находится в работоспособном состоянии.
2. **Безотказность:** свойство системы длительное время работать без отказов. Системы с высокой безотказностью – это системы, которая вероятнее всего, будет непрерывно работать в течении относительно (зависит от системы, авиационные пушки живут 3-4 секунды) долгого времени. Система может иметь высокую безотказность, но низкую доступность (снабжение горячей водой, редко отказывает, но отключают на длительное время на ремонт).
3. **Безопасность:** свойство системы, определяющее последствие отказа (отказ авиационной техники, отказ почтового сервера общего назначения).
4. **Ремонтопригодность:** свойство системы, определяющее сложность (продолжительность) восстановления работоспособности системы. Большинство отказов – монотонные процессы. Системы с обслуживанием по состоянию, системы с обслуживанием по регламенту.
5. **Отказоустойчивость процессов:** достигается объединением нескольких идентичных процессов в группу (кластер). Сообщение получают все члены группы, если один из процессов перестает работать, то его место занимает другой. Группы могут быть динамическими и статическими, один и тот же процесс может входить в несколько групп одновременно.



1. **Отказоустойчивость процессов: группы процессов**  - одноранговые, иерархические. В иерархических группах присутствует координатор распределяющий запросы (во многих системах называется балансировщиком),в случае отказа координатора – выбирается другой координатор из исполнителей.



1. **Отказоустойчивость процессов: группы процессов:**  при централизованном управлении группами, требуется специальный сервер – сервер групп, следящий за членством процессов входящих в группы. Такой подход прост в реализации, но уменьшает надежность - отказ сервера групп, приведет к отказу всей группы.
2. **Отказоустойчивость процессов: группы процессов:** прираспределенном управлении, все процессы хранят список членов группы; требуется надежная групповая рассылка (широкополосная); процесс извещает все процессы группы о своем желании вступить в группу или покинуть ее; нужен механизм позволяющий определить аварийный отказ одного из членов группы (постоянный опрос друг друга); при включении в группу необходим механизм синхронизации (реплицирования) нового члена группы с другими членами (пример, grid Oracle).
3. **Отказоустойчивость процессов: группы процессов:** группапроцессовпозволяет замаскировать отказы одного или более процессов; другими словами можно реплицировать процессы, заменяя одиночный процесс **отказоустойчивой группой (failover group/cluster –** понятие в Microsoft, в Oracle и др.**)**
4. **Отказоустойчивость процессов: группы процессов:** соглашения в системах с ошибками; процесс посылающий, какое-то уведомление другому процессу не знает точно дошло ли сообщение до адресата; при подтверждении получения запроса, не ясно дошло ли подтверждение, …, дошло ли подтверждение на подтверждение; требуется специальные протоколы.