**Распределенные информационные системы**

Лабораторная работа 7.

**Создание макета кластера высокой готовности**

## Задание для выполнения

1. Ознакомиться с материалом приведенном в приложении А. Изучить алгоритм забияки

2. Создайте три UDP-сервера, которые будут по запросам клиентов выдавать системное время в формате DDMMYYYY:mm:ss.

Все сервера работают на порту 5555. Каждый сервер имеет свой уникальный IP адрес из общей сети класса С (префикс /24), в которую подключена сетевая карта машины.

Эти сервера совместно образуют, кластер высокой готовности (НА – High Availability) сервиса выдачи времени (СВВ) по запросам пользователей.

В качестве идентификаторов серверов используются их IP адреса. В момент начального старта сервер с наибольшим номером оповещает соседей, о том, что взял на себя роль координатора.

3. Создайте сервер-UDP выполняющий роль посредника. Этот сервер принимает запросы от клиентов на порту 5555 по IP адресу сетевой карты машины и перенаправляет их на один из серверов UDP СВВ, являющийся в настоящее время сервером координатором, а полученные от координатора результаты отправляет клиенту. Адрес текущего сервера координатора он узнает из конфигурационного файла службы СВВ.

В процессе работы сервер посредник выводит на консоль протокол работы, в который заносит IP адрес клиента и IP адрес координатора которому он перенаправил запрос, а также сообщения о сбоях в работе с серверами СВВ.

4. Сервера СВВ с интервалом в 5 секунд проверяют работоспособность сервера координатора и если последний находится в неработоспособном состоянии (координатор трижды не ответил на запрос о готовности), то инициирует выборы нового координатора по алгоритму забияки. Сведения о серверах входящих в состав СВВ, содержатся в конфигурационном файле службы СВВ.

5. Продемонстрируйте работу кластера HA CВВ, используя в качестве показателя работоспособности кластера журнал работы сервера посредника.

**Контрольные вопросы.**

1. Представьте себе, что два процесса одновременно обнаруживают отказ в работе координатора и решают провести голосование по алгоритму забияки. Что произойдет в этом случае?

# Приложение А

# Алгоритмы голосования. Алгоритм забияки

(выдержка из раздела 5.4 книги Танненбаум Э. Распределенные системы 1-е изд. 2003г.)

Многие распределенные алгоритмы требуют чтобы один из процессов был координатором. Если все процессы одинаковы, то способа выбрать один из них не существует. Поэтому применяются различные системы голосования. При этом считаем, что у них есть уникальные идентификаторы и они не знают какие процессы работают, а какие нет.

**Алгоритм** **забияки**

Когда один из процессов замечает, что координатор больше не отвечает на запросы, он инициирует голосование. Голосование проходит следующим образом:

1. Процесс P посылает всем процессам с большими, чем у него, номерами **сообщение голосования**

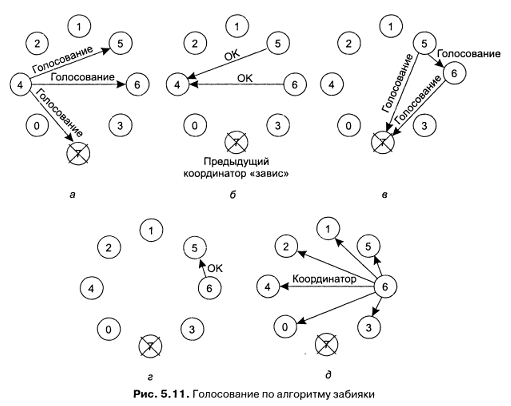
2. Далее возможно 2 варианта развития:

a. Если никто не отвечает, то процесс P выигрывает голосование и становится координатором.

b. Если один из процессов с большими номерами отвечает, то он становится координатором, а работа P заканчивается

Процесс может получить сообщение голосования от одного из коллег с меньшим номером. Получатель посылает сообщение «OK» отправителю, показывая, что готов стать координатором. Затем получатель сам организует голосование. В конце концов все процессы кроме одного отпадут, этот последний и будет координатором. Он уведомит всех что он координатор и приступит к работе. Побеждает всегда процесс, имеющий самый большой номер. Отсюда и название алгоритм забияки.

На рис. 5.11 приведен пример работы алгоритма забияки. Группа состоит из восьми процессов, пронумерованных от 0 до 7. Ранее координатором был процесс 7, но он завис. Процесс 4 первым замечает это и посылает сообщение ГОЛОСОВАНИЕ всем процессам с номерами больше, чем у него, то есть процессам 5, 6 и 7, как показано на рис. 5.11а. Процессы 5 и 6 отвечают ОК, как показано на рис. 5.11б. После получения первого из этих ответов процесс 4 понимает, что он не может претендовать на роль координатора и что координатором будет один из процессов имеющих более высокие номера. Он прекращает свое участие в голосовании и ожидает, кто станет победителем (хотя уже сейчас он может сделать довольно точное предположение).



На pис. 5.11в, показано, как оба оставшихся процесса, 5 и 6, продолжают голосование. Каждый посылает сообщения только тем процессам, номера у которых больше их собственных. На рис. 5.11г, процесс 6 сообщает процессу 5, что голосование будет вести он. В это время 6 понимает, что процесс 7 мертв, а значит, победитель — он сам. Если информация о состоянии сохраняется на диске или где-то еще, откуда ее можно достать, когда с прежним координатором что-нибудь случается, процесс 6 должен записать все что нужно на диск. Готовый занять свою должность процесс 6 заявляет об этом путем рассылки сообщения КООРДИНАТОР всем работающим процессам. Когда 4 получит это сообщение, он продолжит работу с той операции, которую пытался выполнить, когда обнаружил, что процесс 7 мертв, используя теперь в качестве координатора процесс 6. Таким образом, мы обошли сбой в процессе 7, и работа продолжается.

Если процесс 7 запустится снова, ему будет достаточно послать всем остальным сообщение КООРДИНАТОР и вынудить их подчиниться.

node cbb\_server.js 192.168.1.101

node proxy\_server.js

node client.js