Консенсус в асинхронной системе, Replicated State Machines

Илья Кокорин

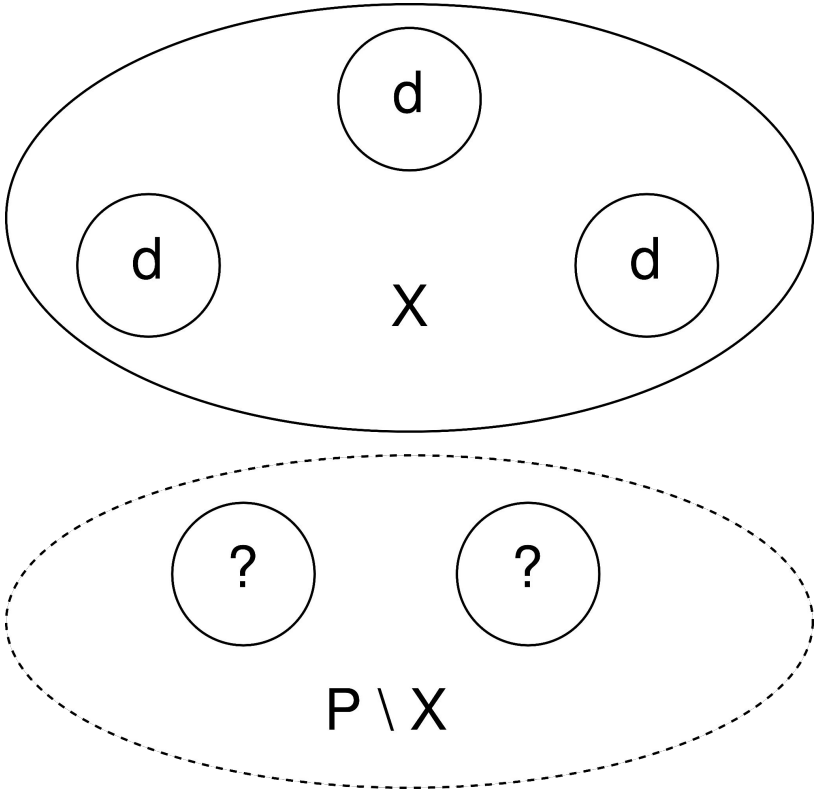
kokorin.ilya.1998@gmail.com

FLP теорема

- Консенсус невозможен, если
  - Сеть асинхронная
  - Возможен отказ хотя бы одного узла
  - Алгоритм детерминированный
  - **Несбойные узлы должны прийти к нетривиальному консенсусу за конечное число шагов**

Кворум

- Всё множество процессов P может не прийти к решению из-за сбоев
- Пусть подмножество процессов $X \subseteq P$ пришло к единому решению
- В каком случае можно быть уверенным, что ни один процесс из $P \setminus X$ не придёт к другому решению?



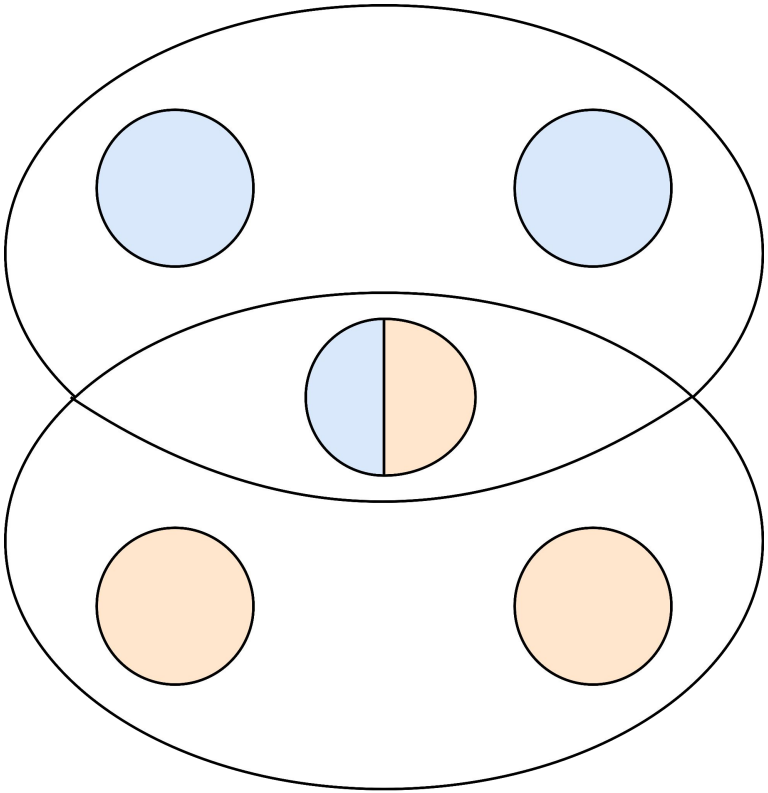
Кворум

- Множество процессов $\mathbb{P}$
  - {1, 2, 3}
- Множество всех подмножеств $2^{\mathbb{P}}$
  - { { }, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3} }
- Подмножество множество всех подмножеств $\mathbb{Q} \subset 2^{\mathbb{P}}$
  - Такое, что $A, B \in \mathbb{Q} \Rightarrow A \cap B \neq \emptyset$
  - Называется множеством кворумов
  - { { }, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3} }

Кворум простого большинства

- Больше половины процессов

$$A \in \mathbb{Q} \text{ iff } |A| > \frac{|\mathbb{P}|}{2}$$



Кворум взвешенного большинства

- У каждого процесса есть положительный вес

$$\{w_i \in (0; +\infty)\}_{i=1}^{|\mathbb{P}|}$$

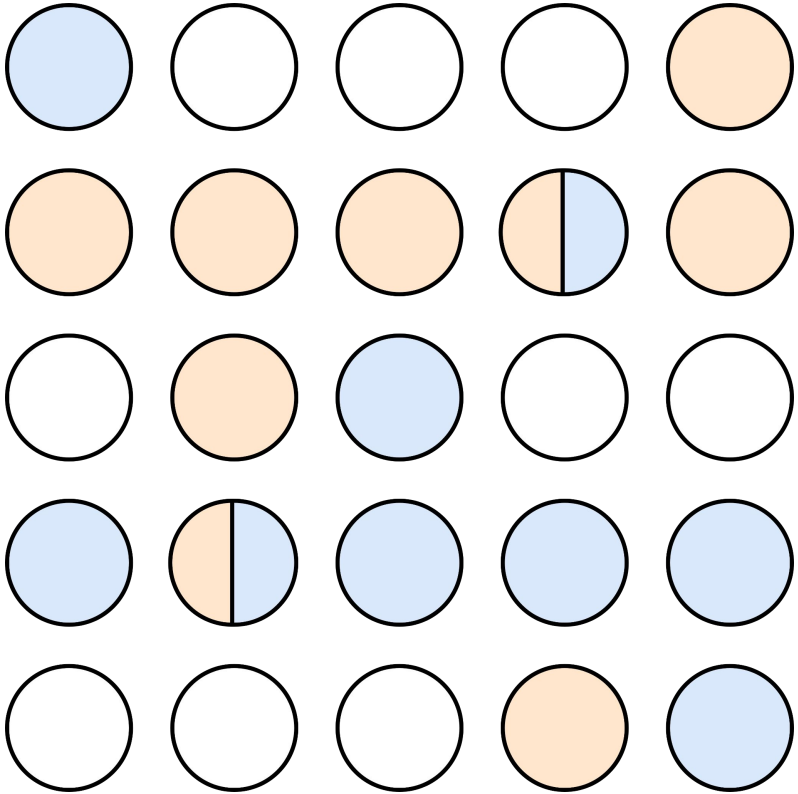
- Множество процессов является кворумом, его вес составляет больше половины общего веса

$$A \in \mathbb{Q} \text{ iff } \sum_{p \in A} w_p > \frac{\sum_{p \in \mathbb{P}} w_p}{2}$$

- При $w_c = 1,\ \forall i \neq c : w_i = 0$ получаем тривиальный кворум
- Множество является кворумом если в нём есть координатор
- При равных весах получаем простое большинство

Кворум рушащихся стен

- Выстраиваем процессы в матрицу
- Выбираем произвольную строку целиком
- И минимум по одному элементу в каждой другой строке
- Размер $O(\sqrt{|\mathbb{P}|})$

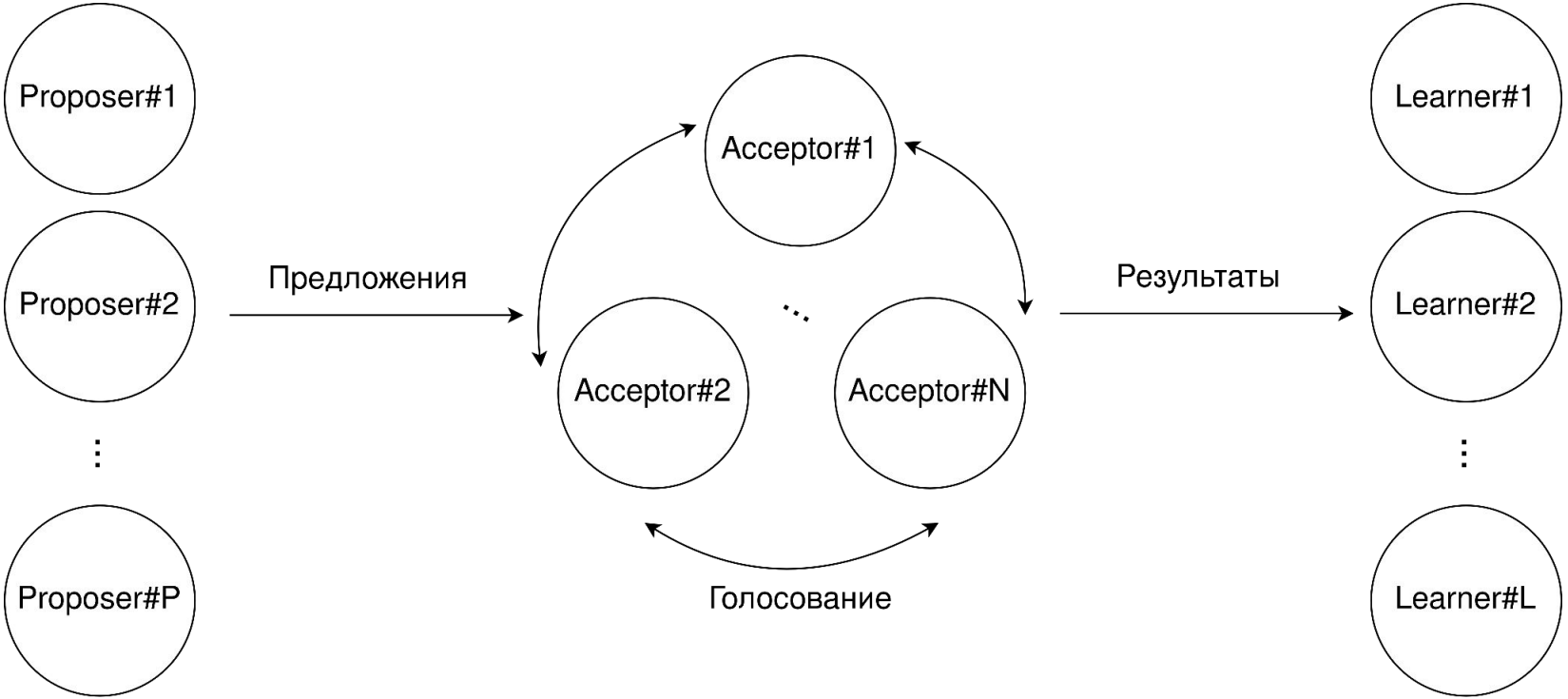


Болотный кворум

- Консенсус, мои чуваки

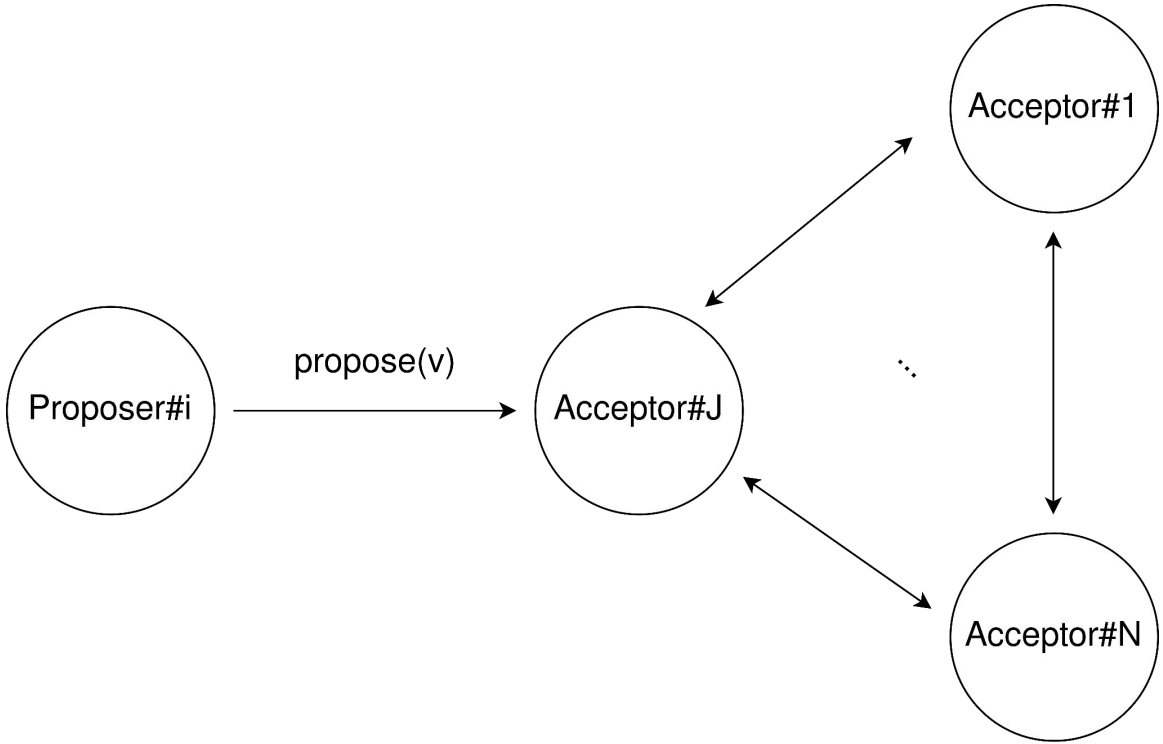
Paxos: типы процессов

- Процессы разделены на предлагающих, принимающих и узнающих



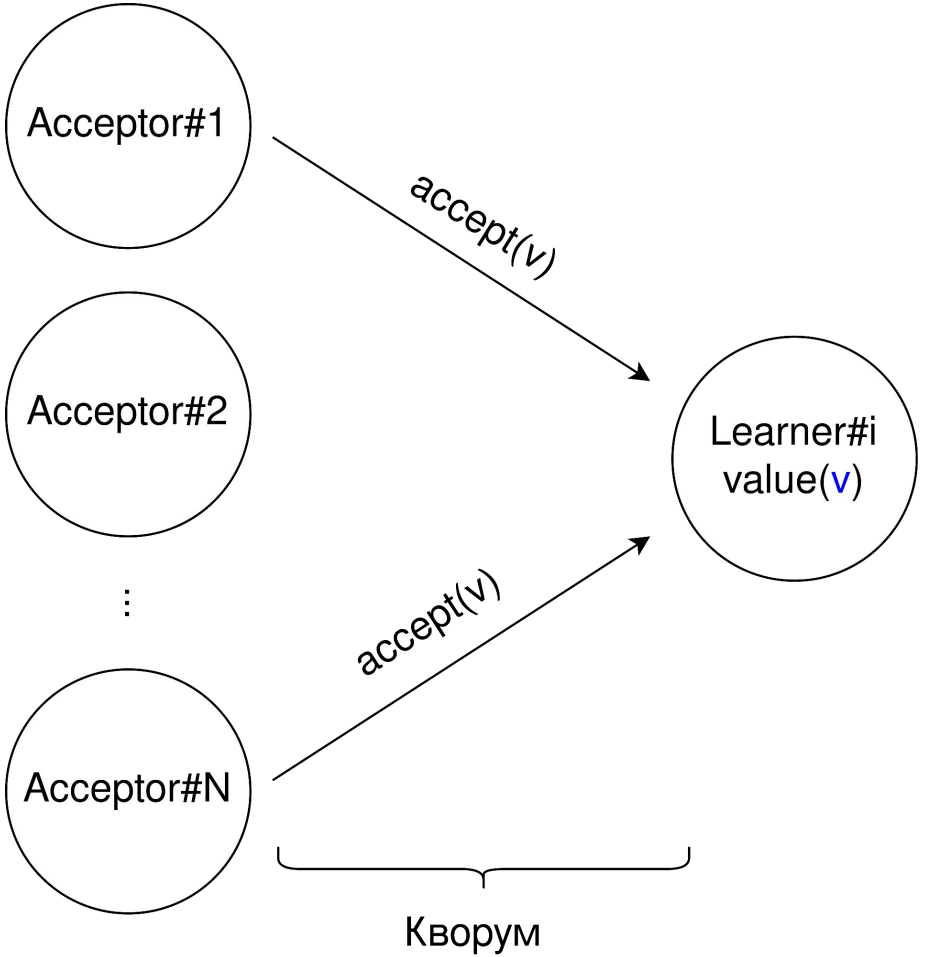
Paxos: предлагающие

- Просто выбирает случайного принимающего
- И пересылает ему своё предложение
- Чтобы принимающий поставил его на голосование
- Обращение гражданина к законодателю



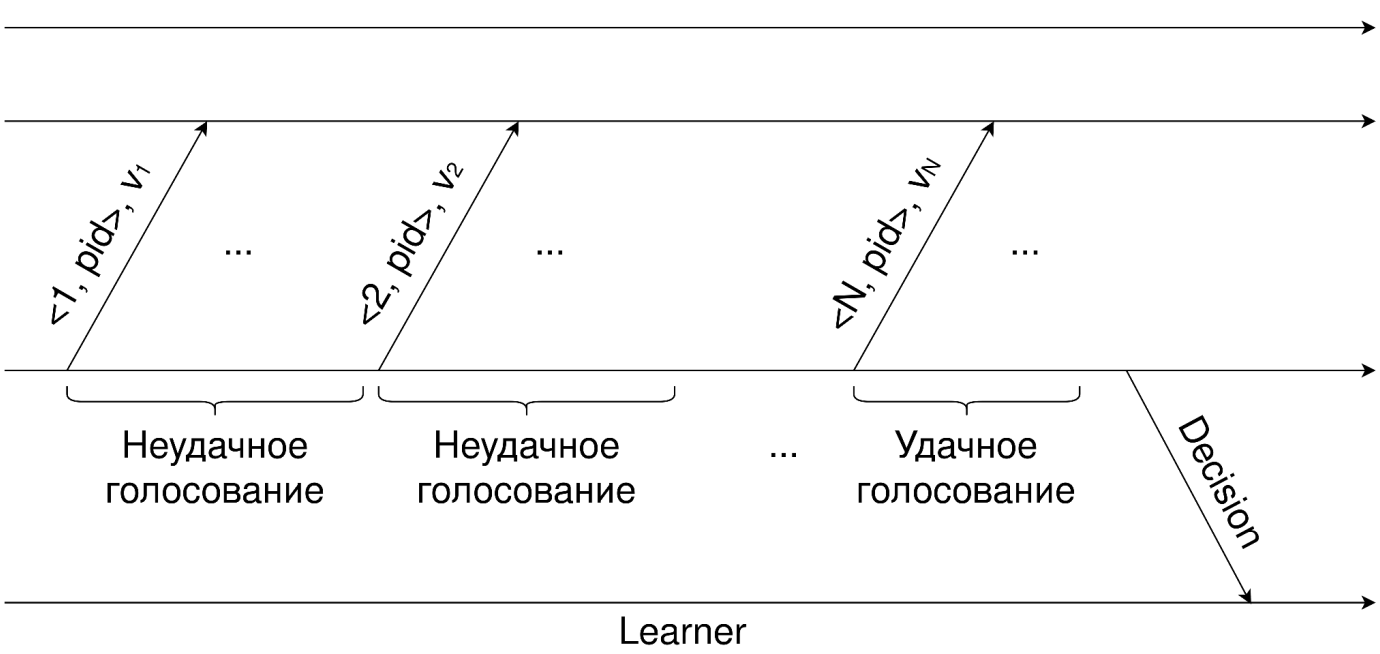
Paxos: узнающие

- Ждёт сообщений от принимающих
- Получив от кворума принимающих сообщения о том, что они приняли одно значение, помечает это значение как принятое
- Чиновники, начинающие исполнять принятые законы



Paxos: голосование

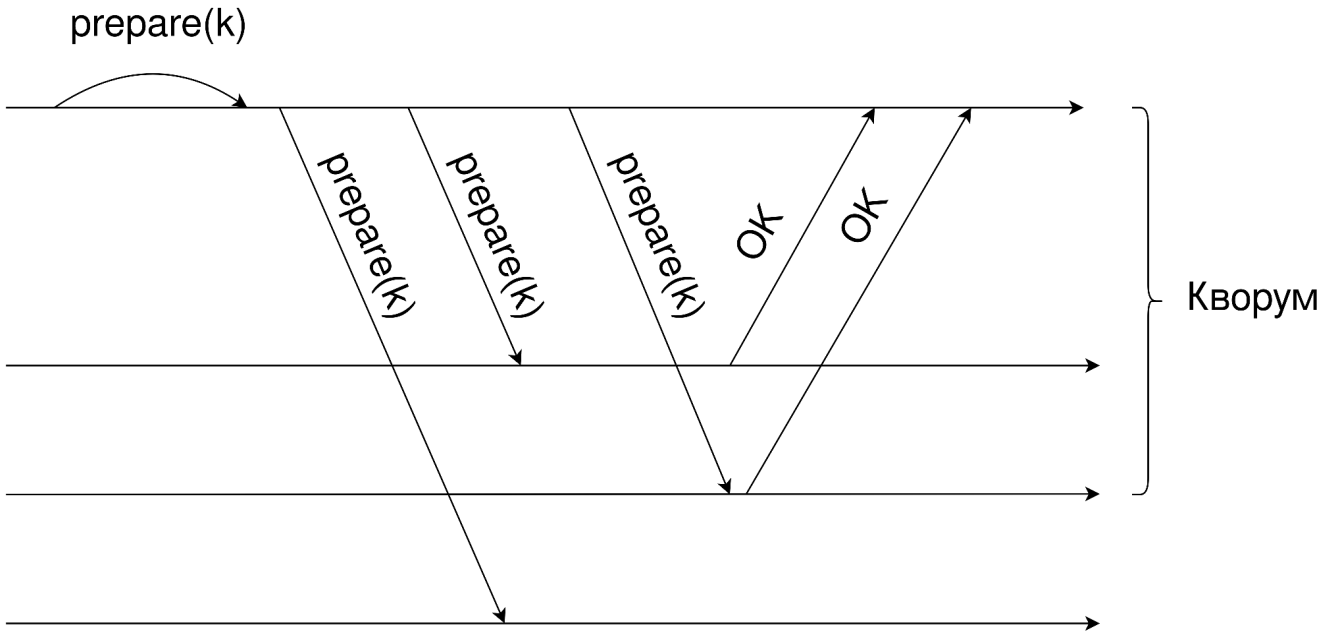
- Принимающий может инициировать много раундов голосования
- Раунды нумеруются <cnt, pid>
- Уникальный номер раунда



- cnt увеличивается в каждом процессе независимо

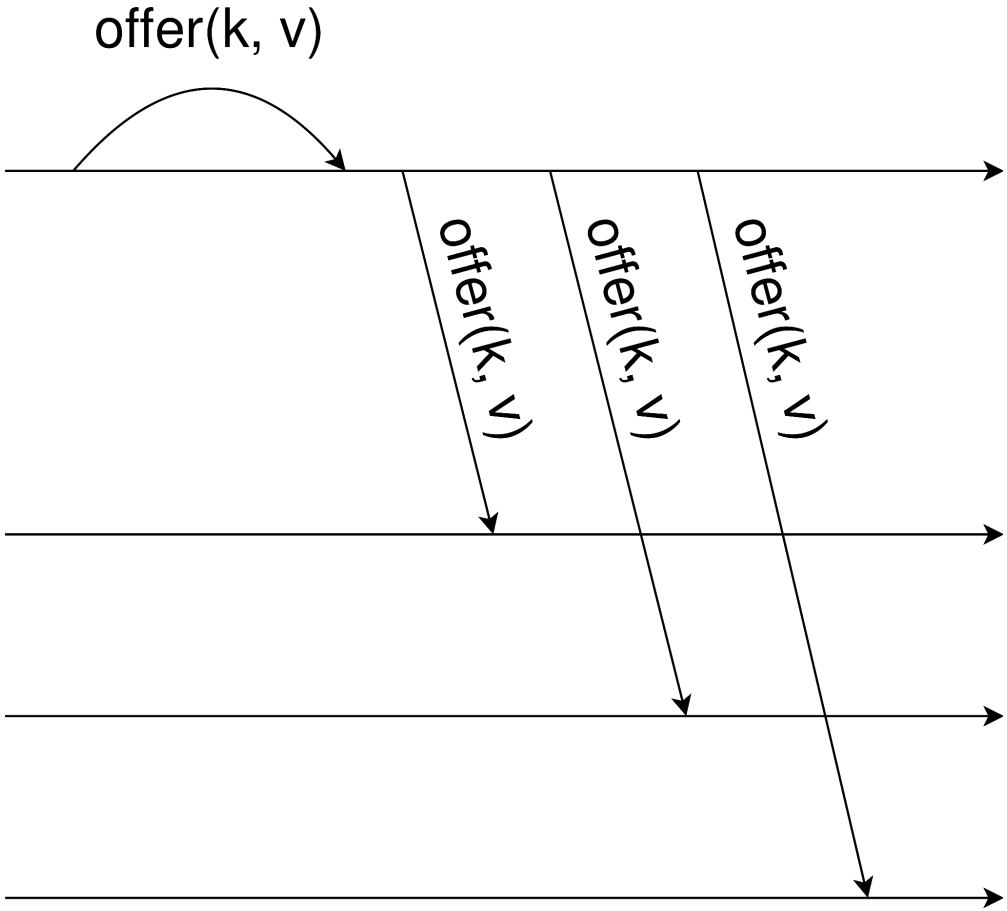
Paxos: первая фаза

- Послать всем процессам сообщение “не участвуй в голосованиях с номером меньше <num, pid>”
- Собрать с кворума подтверждения



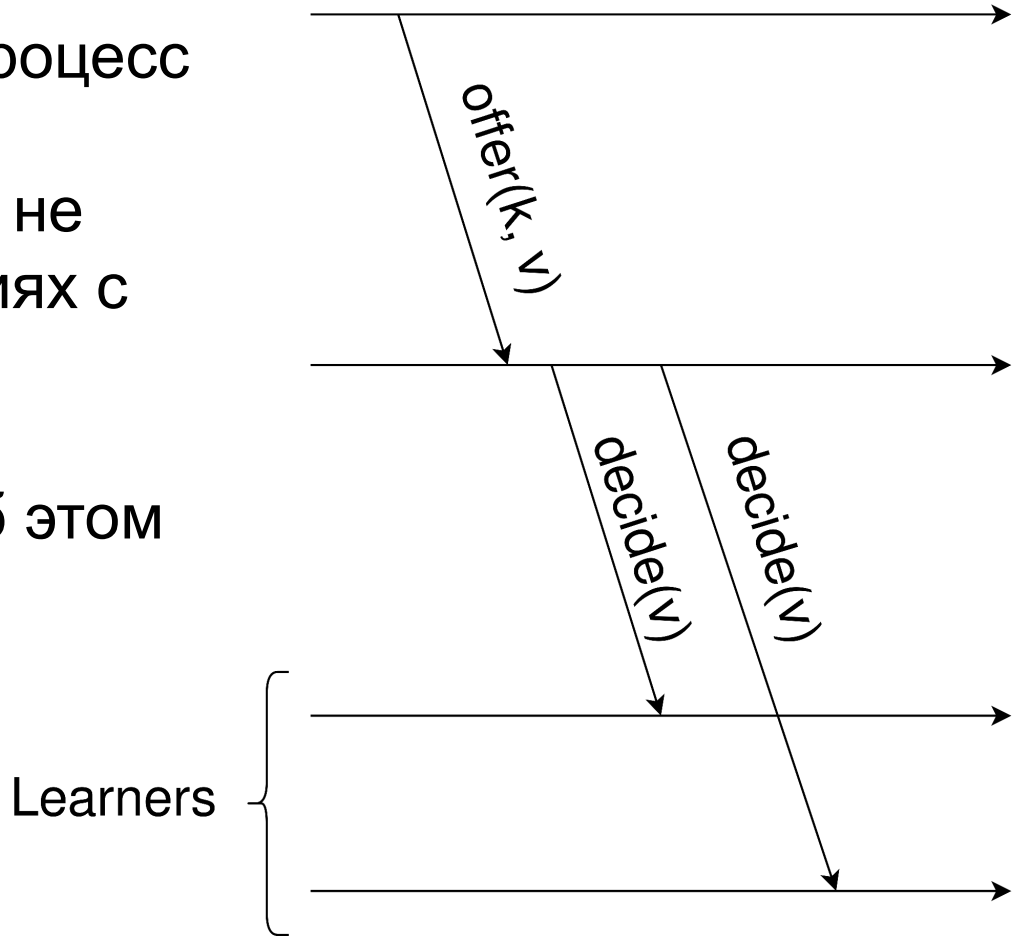
Paxos: вторая фаза

- Предложить всем процессам своё значение
- С указанием номера раунда



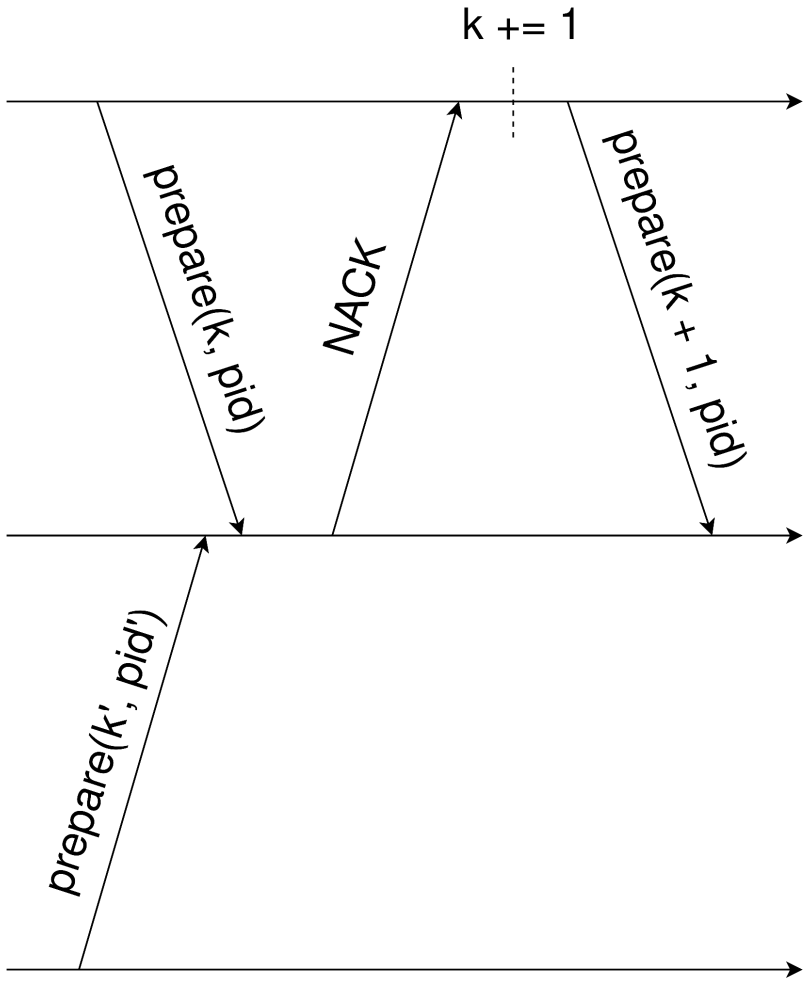
Paxos: вторая фаза

- Получив предложение, процесс принимает его
- Если не давал обещаний не участвовать в голосованиях с номером меньше k’
  - Где k’ > k
- Рассылает сообщение об этом всем узнающим



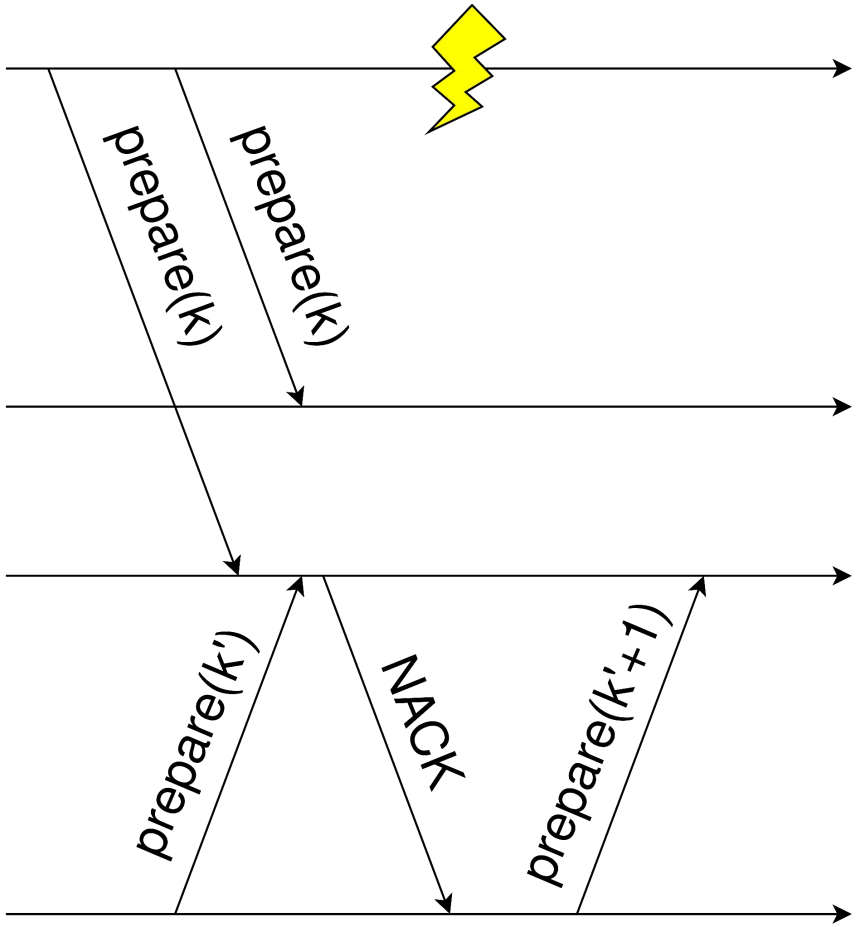
Paxos: первая фаза

- Пусть процесс дал обещание не участвовать в голосованиях с номером меньше k’, k’ > k
- Увеличиваем локальный счётчик на 1 и пробуем ещё раз



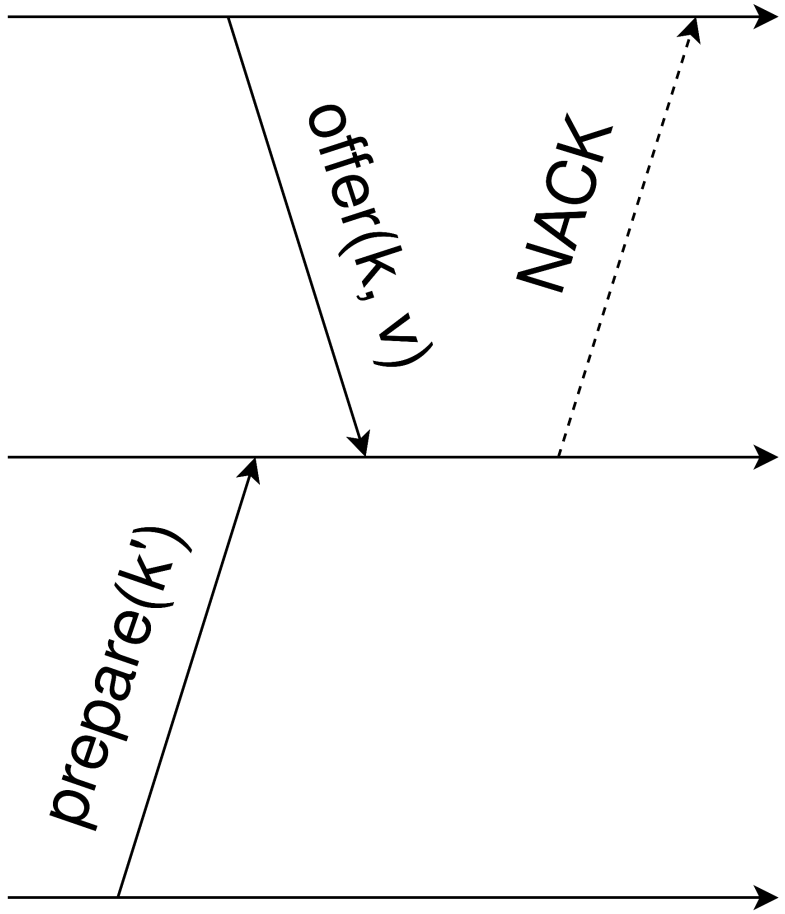
Paxos: зачем начинать ещё один раунд?

- Пусть инициатор собрал обещания и умер
- Другой процесс должен иметь возможность привести систему к консенсусу
- Иначе система останется навечно заблокированной



Paxos: вторая фаза

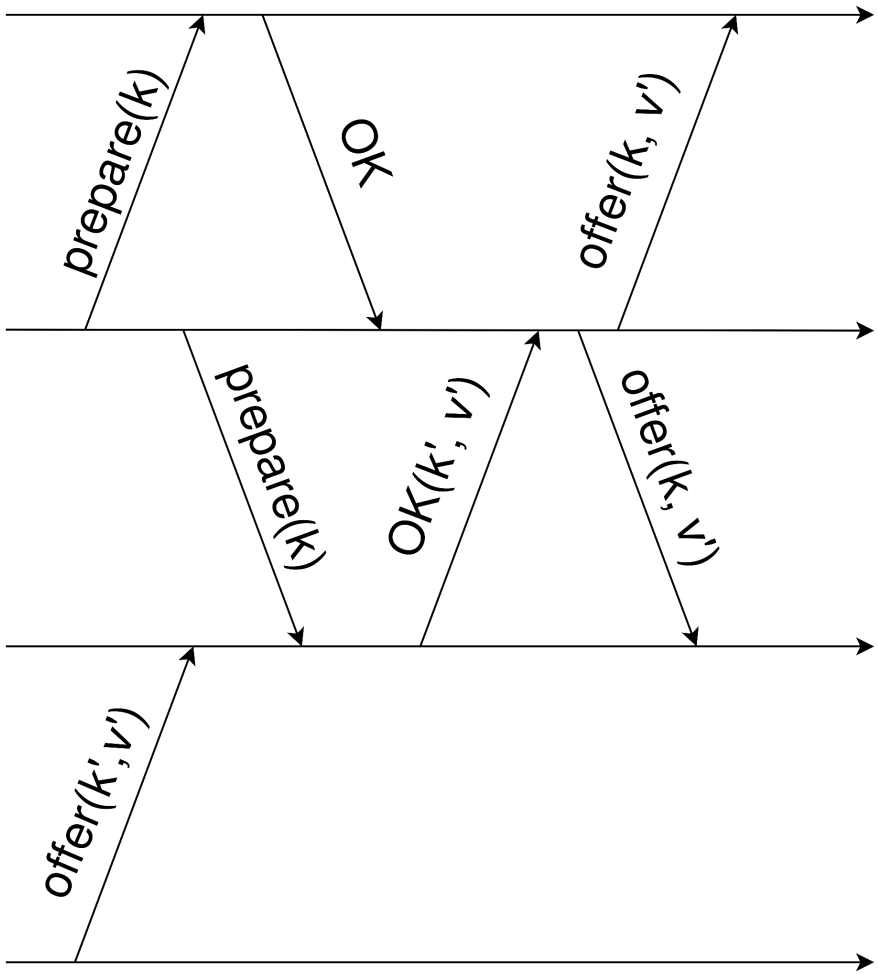
- Если процесс дал обещание не участвовать в голосованиях с номером меньше k’, k’ > k
- То не принимает предложение
- Не рассылает узнающим
- **Может** послать сообщение об этом
- Чтобы инициатор прекратил свой раунд



Paxos: смена значения

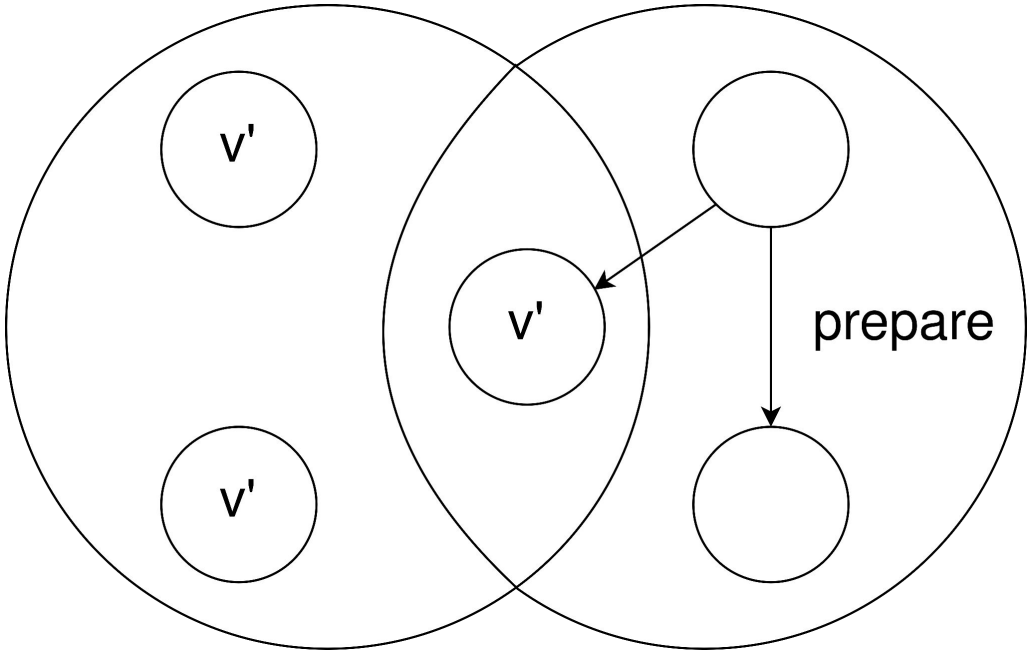
- Пусть процесс уже принял предложение v’ с номером k’ < k
- В первой фазе отвечает OK(k’, v’)
- Предлагающий меняет своё предложение на v’
- Вторую фазу исполняет для v’

$$v' := argmax_{k'} OK(k', v')$$



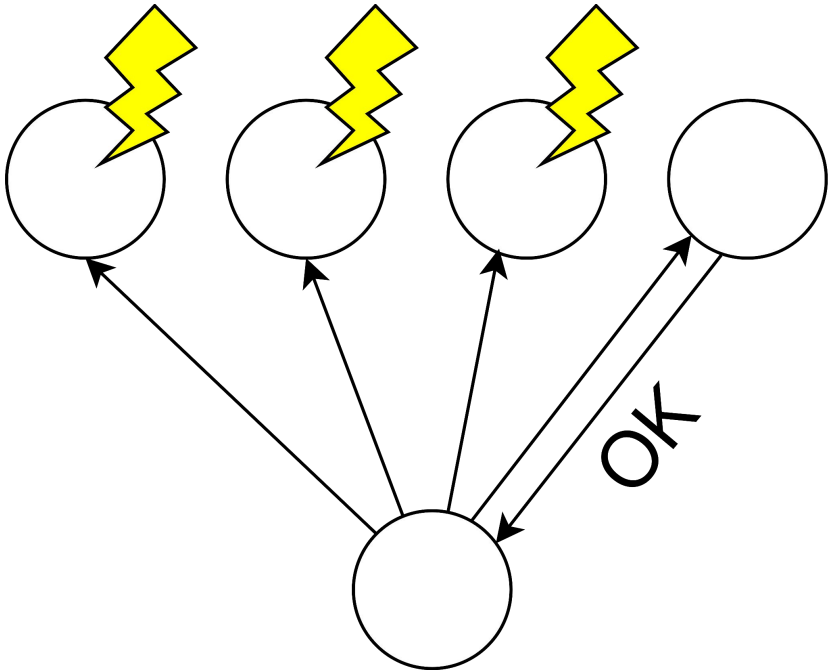
Paxos: доказательство

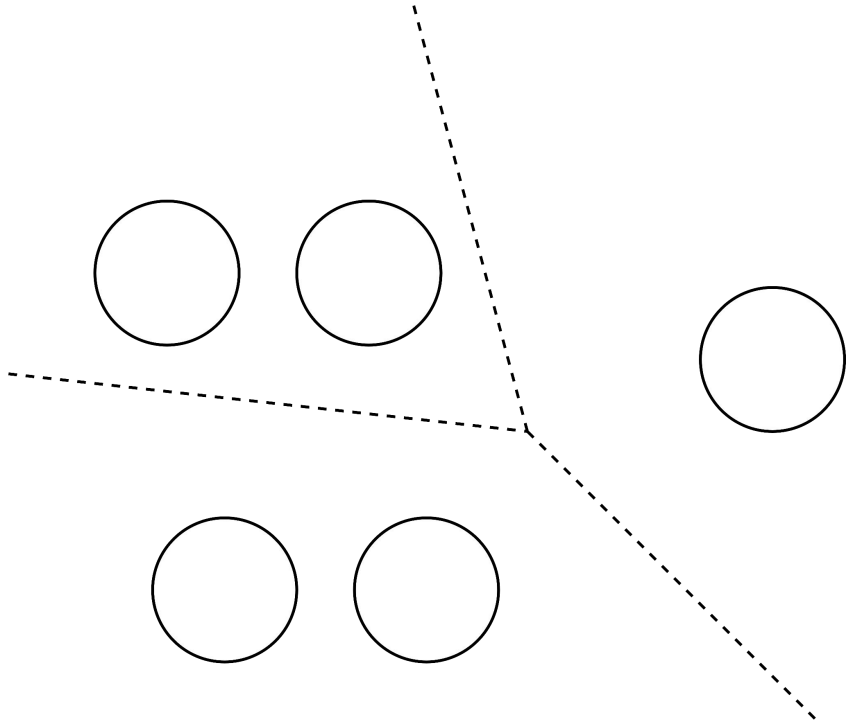
- Если значение v’ принято большинством, каждая дальнейшая вторая фаза будет проходить с этим значением



Paxos: отсутствие прогресса

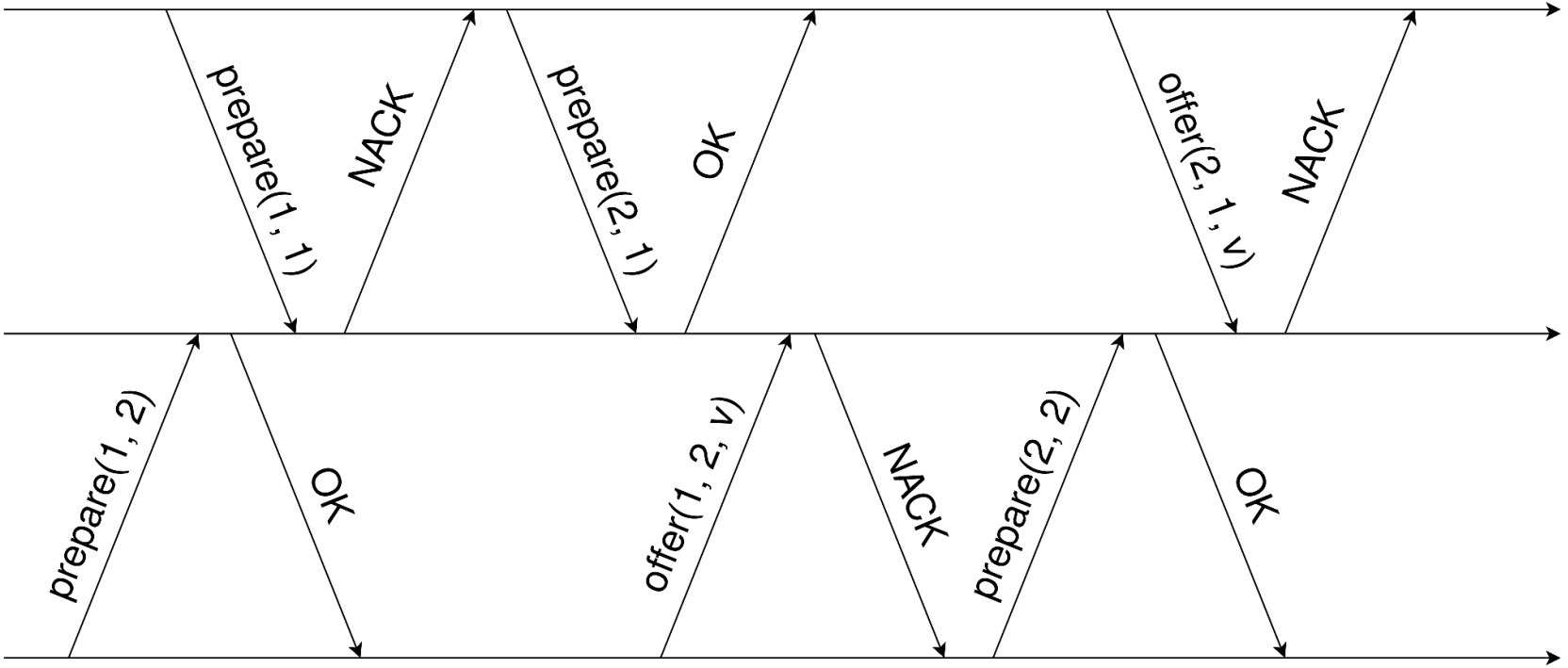
- Не собирается кворум из-за сбоев узлов или каналов





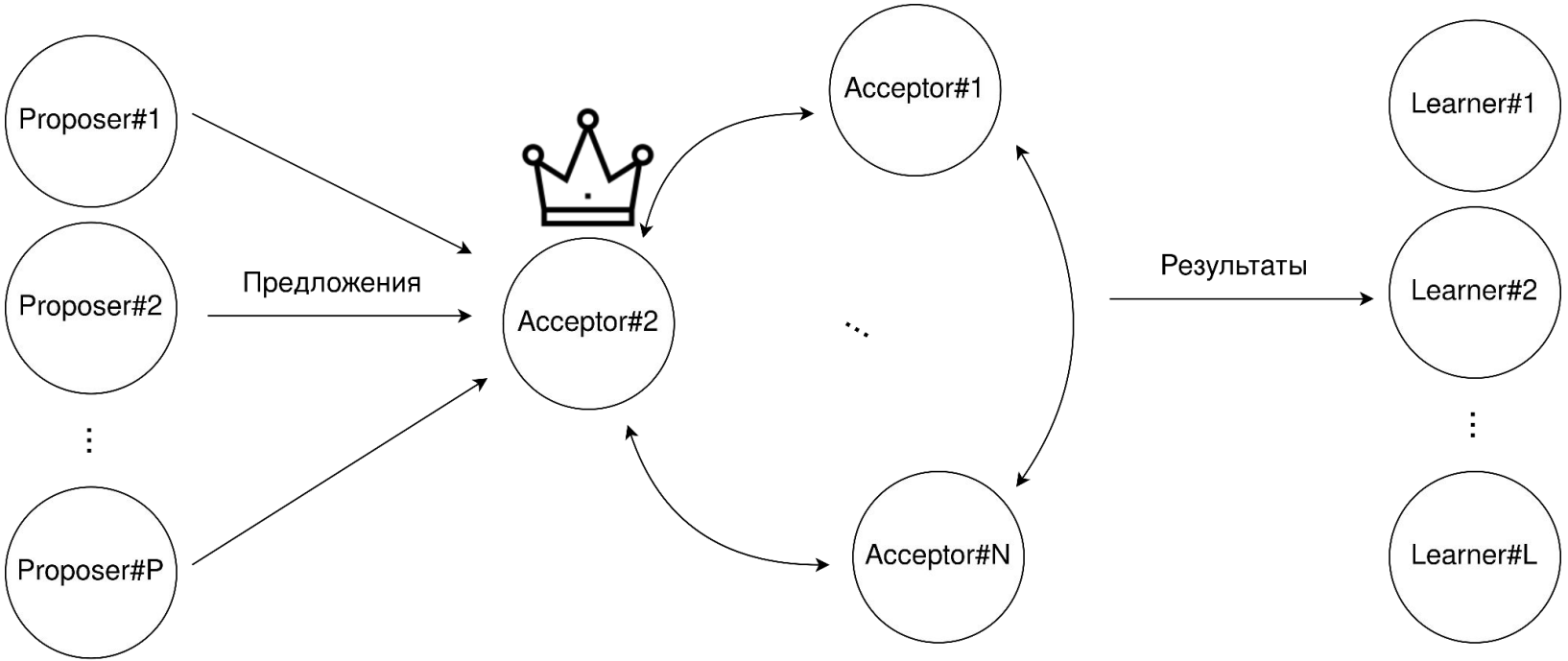
Paxos: отсутствие прогресса

- Может случаться даже в отсутствие сбоев
- Помогает рандомизация таймаутов



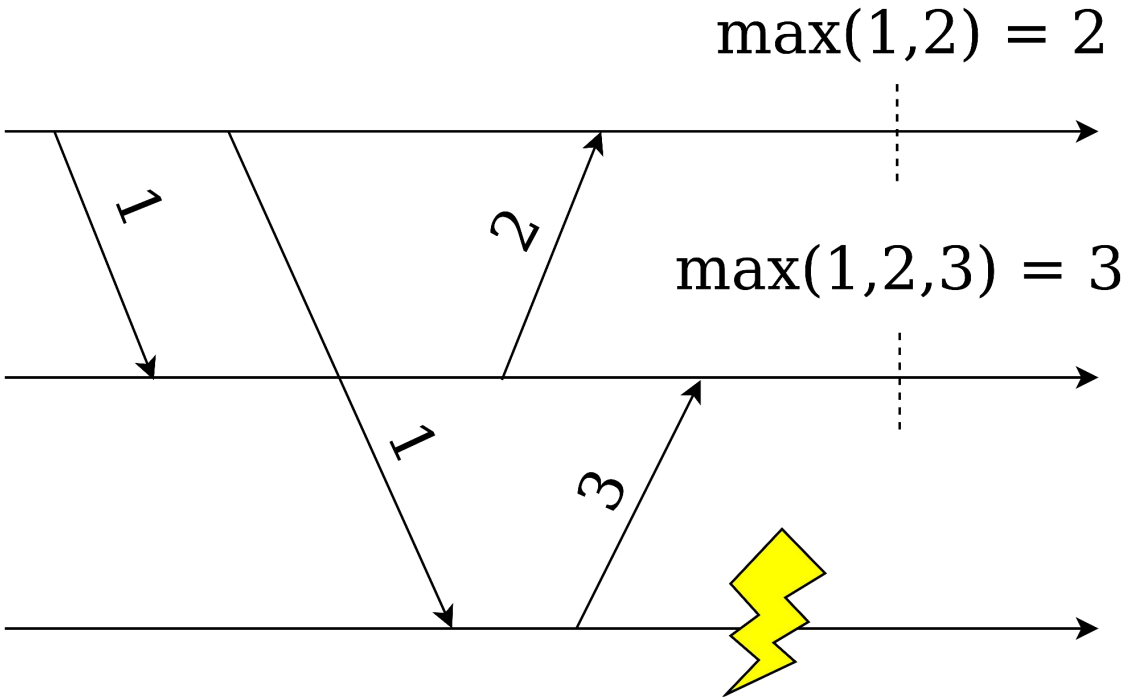
Paxos: лидер

- Выбираем лидера среди принимающих
- Все предложения идут через через него
- Никто лидеру не мешает



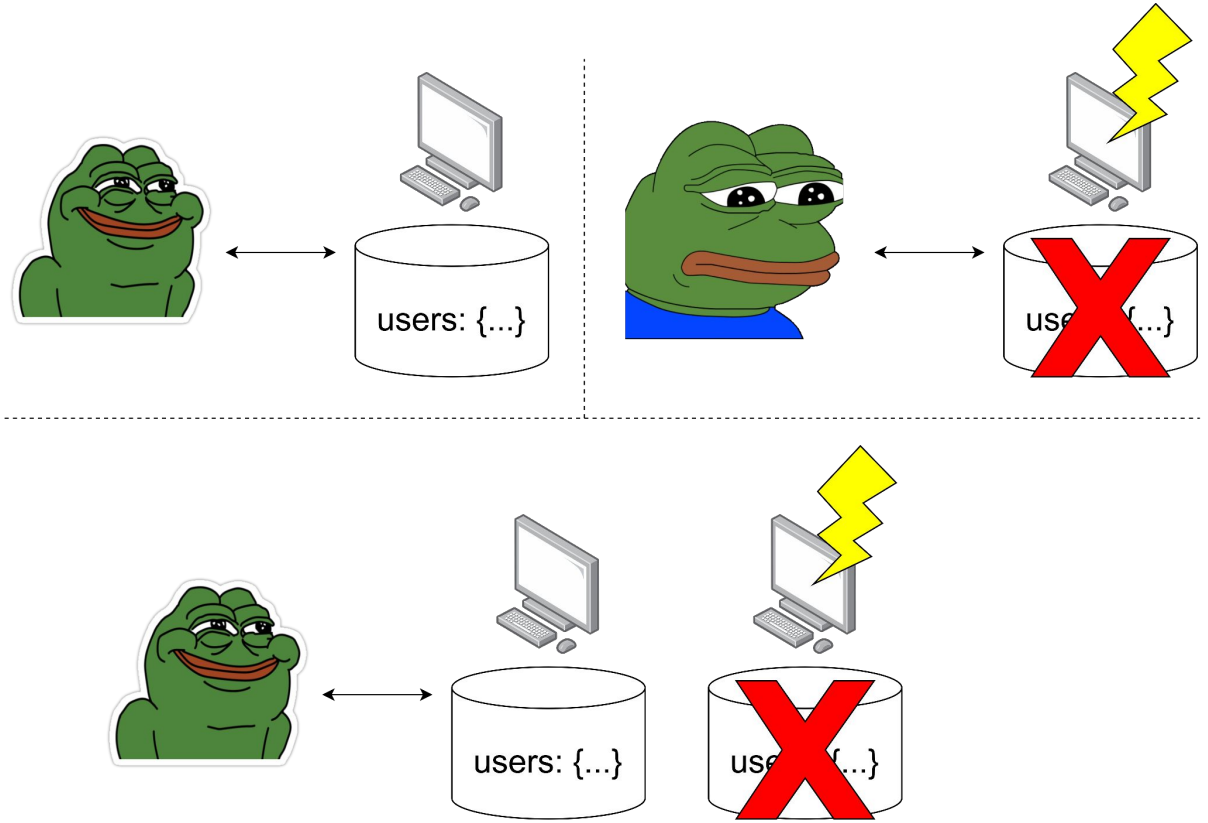
Paxos: лидер

- Нельзя надёжно выбрать лидера за конечное время
- Выберем как-то
- Paxos гарантирует согласие даже при нескольких лидерах
- Оптимизация, не влияет на корректность



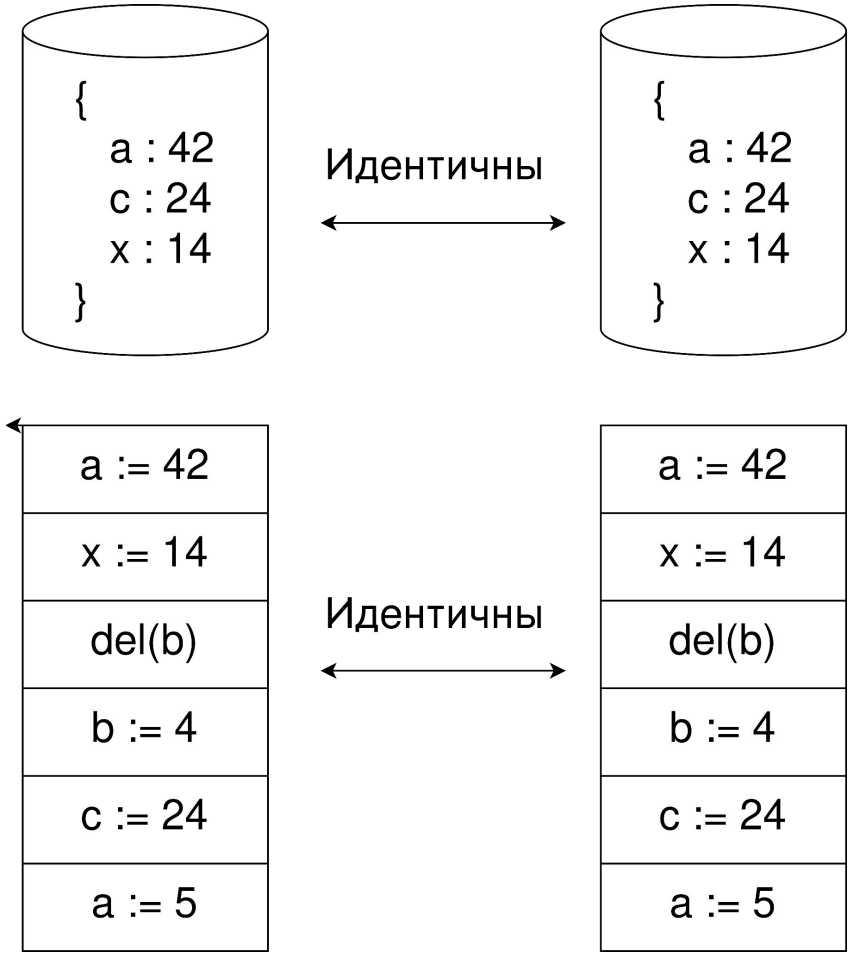
Репликация

- Хотим увеличить надёжность системы



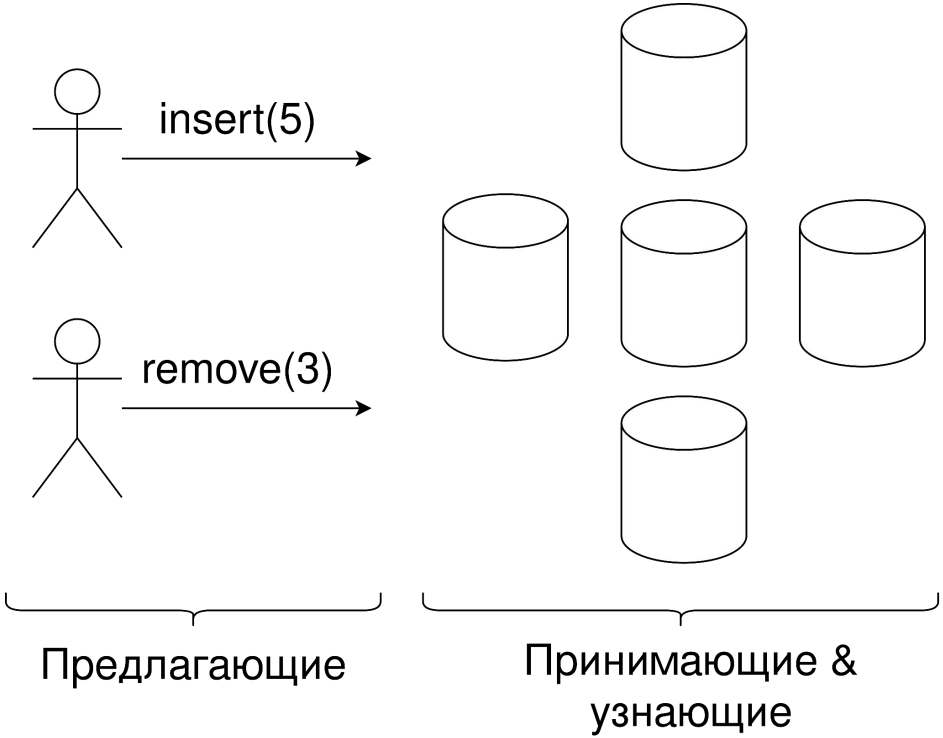
Replicated State Machines

- Структура данных детерминировано восстанавливается из журнала операций
- Добиваемся идентичности журналов на разных узлах
- Структуры данных тоже будут идентичны
- Консенсус на i-ой операции в журнале



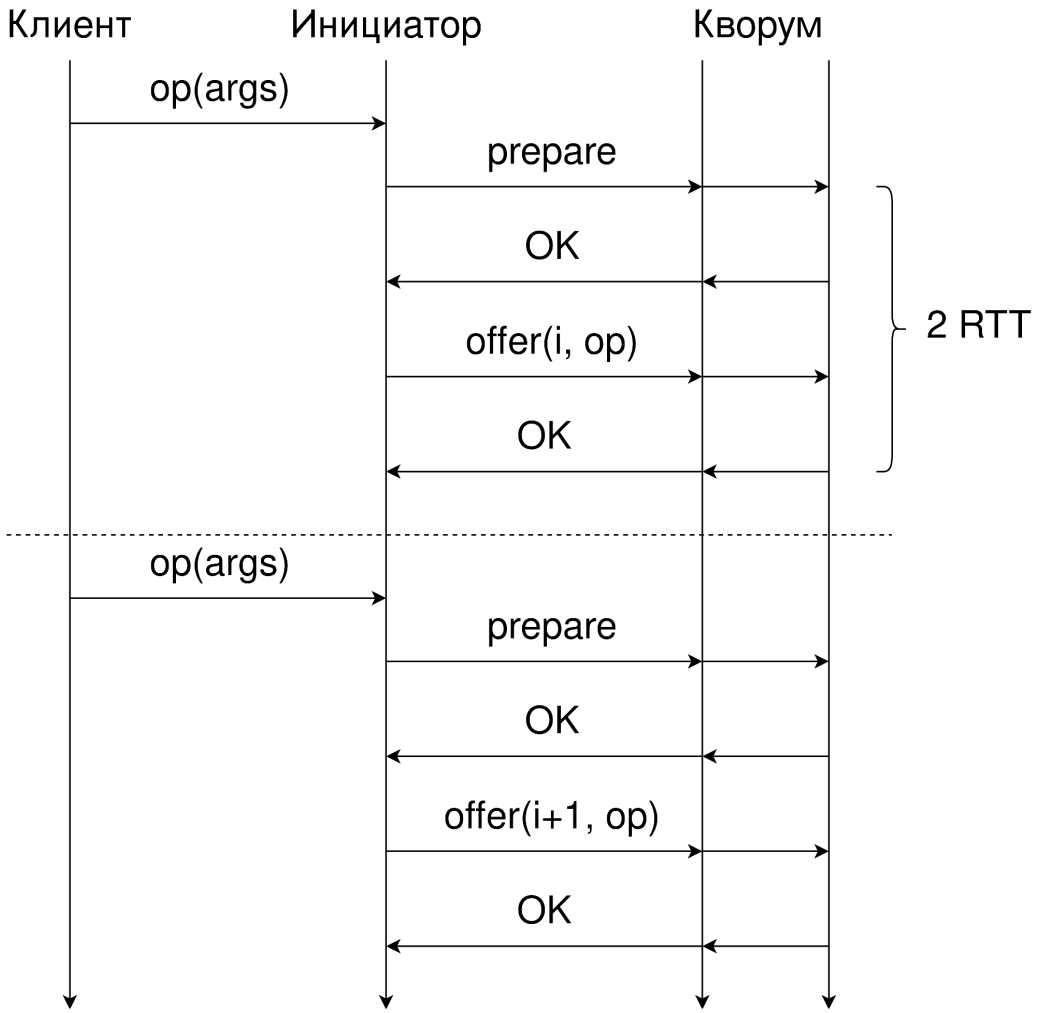
RSM x Paxos

- Клиенты хотят исполнять операции над структурой данных
- Предлагают свою операцию на текущую позицию в логе
- Узлы с данными участвуют в голосовании
- Узнают, какая операция находится на i-ой позиции в журнале
- Конструируют структуру по журналу



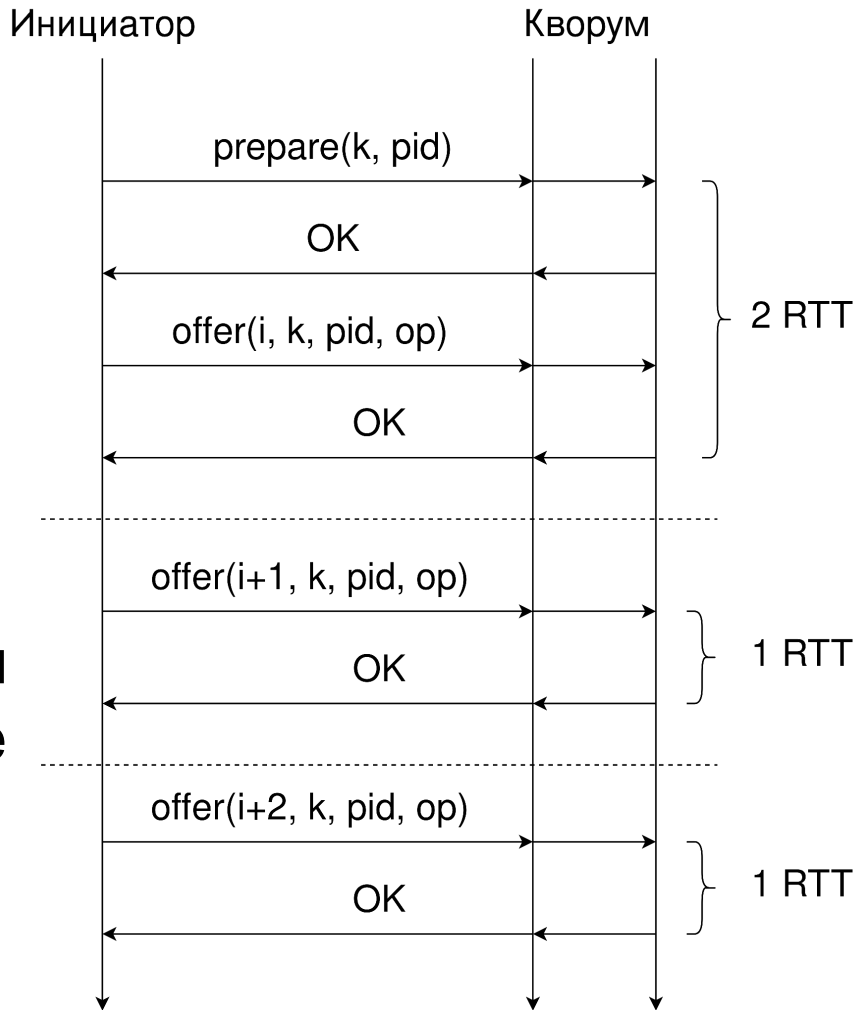
RSM x Paxos

- Каждая операция стоит 2 RTT



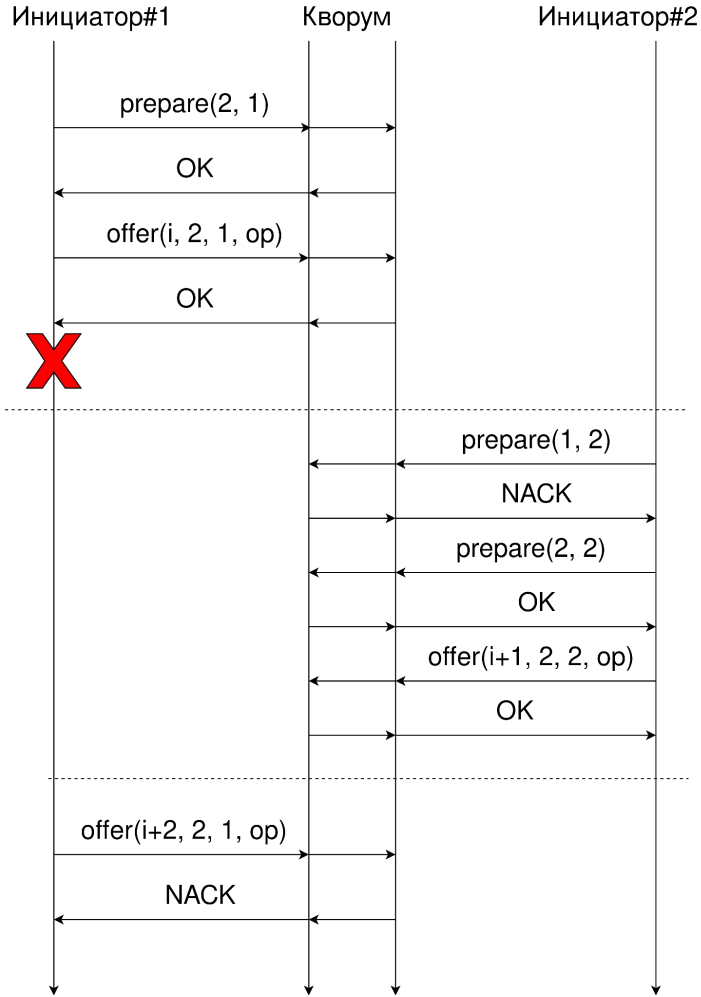
Multi Paxos

- Первая фаза не зависит от предлагаемого значения
- Можно сделать её один раз
- А потом сделать сколько угодно вторых фаз
- Номер голосования <k, pid> становится глобальным
- Принимающие помнят, что при прошлых выборах обещали не участвовать в голосованиях с меньшим номером



Multi Paxos

- При сбое всё происходит как обычно
- Начинаем первую фазу с большим номером
- Если сбоев мало — в среднем будет 1 RTT на добавление в журнал



Что почитать: консенсус в асинхронной сети

- P*eleg D., Wool A.* Crumbling walls: A class of practical and efficient quorum systems
- *Ben-Or M.* Another advantage of free choice completely asynchronous agreement protocols
- *Lamport L.* The part-time parliament
- *Lamport L.* Paxos made simple
- *Van Renesse R., Altinbuken D.* Paxos made moderately complex

Что почитать: Paxos на практике

- *Chandra T. D., Griesemer R., Redstone J.* Paxos made live: an engineering perspective
- *Baker J. et al.* Megastore: Providing scalable, highly available storage for interactive services.
- *Zheng J. et al.* PaxosStore: high-availability storage made practical in WeChat

> There are significant gaps between the description of the Paxos algorithm and the needs of a real-world system. In order to build a real-world system, an expert needs to use numerous ideas scattered in the literature and make several relatively small protocol extensions. The cumulative effort will be substantial and the final system will be based on an unproven protocol.

Thanks for your attention