Tarantool

Чего стоит достижение линеаризуемости в распределённой системе

Сергей Петренко,
Tarantool

бизнес

Привет!

Меня зовут Сергей Петренко

Привет!

Меня зовут Сергей Петренко

Стал программистом,
чтобы делать игры

Привет!

Меня зовут Сергей Петренко

Стал программистом,
чтобы делать игры

В итоге уже 5 лет работаю
в Tarantool

О Tarantool

Платформа для работы с данными

О Tarantool

Платформа для работы с данными

В основе – in-memory СУБД

О Tarantool

Платформа для работы с данными

В основе – in-memory СУБД

Персистентность с помощью WAL

О Tarantool

Платформа для работы с данными

В основе – in-memory СУБД

Персистентность с помощью WAL

Репликация

План доклада

Репликация и связанные
с ней аномалии

Линеаризуемость
Реализация в Tarantool
и других базах

Цена линеаризуемости

Последствия масштабирования

Масштабирование

Репликация

Масштабирование

Репликация

- Резервирование данных

Масштабирование

Репликация

- Резервирование данных
- Распределение нагрузки на чтение

Масштабирование

Репликация

- Резервирование данных
- Распределение нагрузки на чтение

Шардирование

Масштабирование

Репликация

- Резервирование данных
- Распределение нагрузки на чтение

Шардирование

- Распределение нагрузки на запись

Асинхронная репликация

В фоне копируем данные с мастера на реплики

Последствия асинхронной репликации

Чтение старых данных

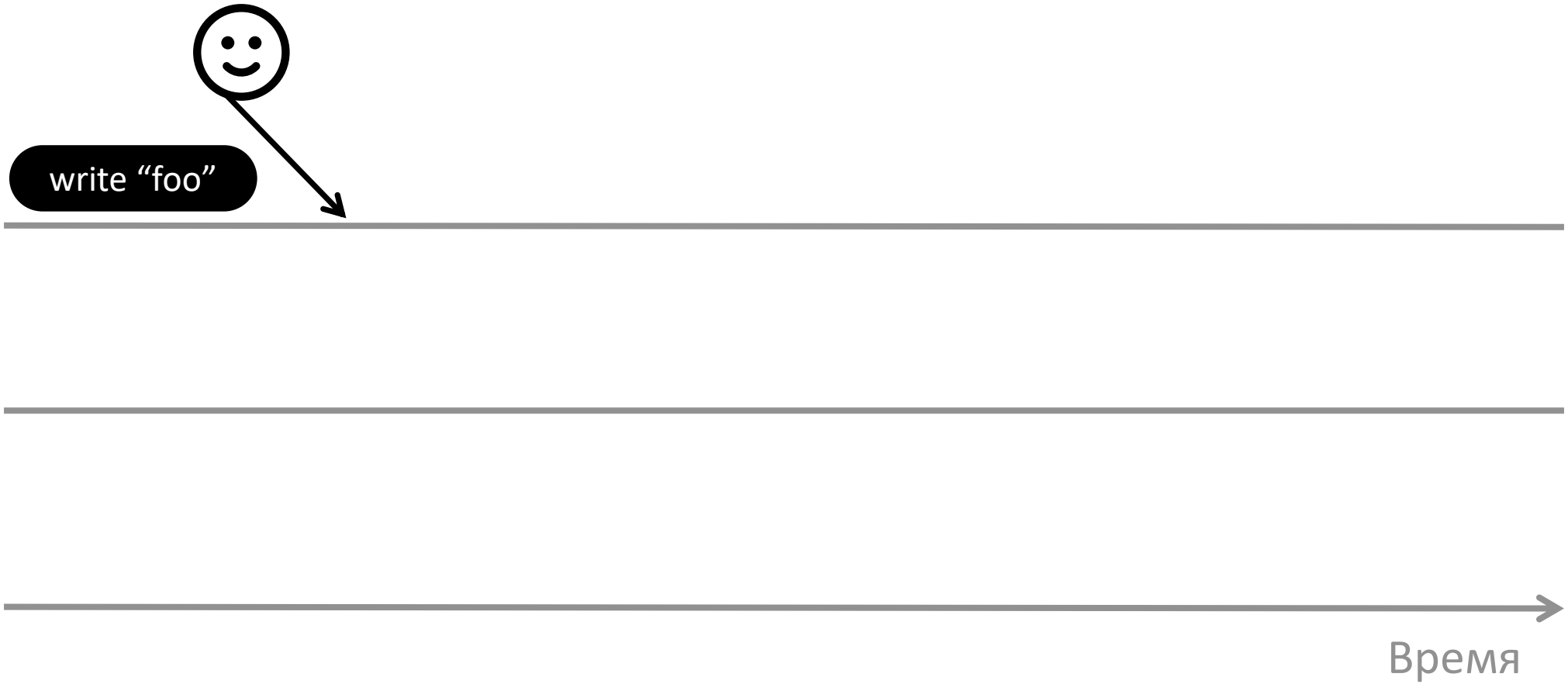
Чтение старых данных

Мастер

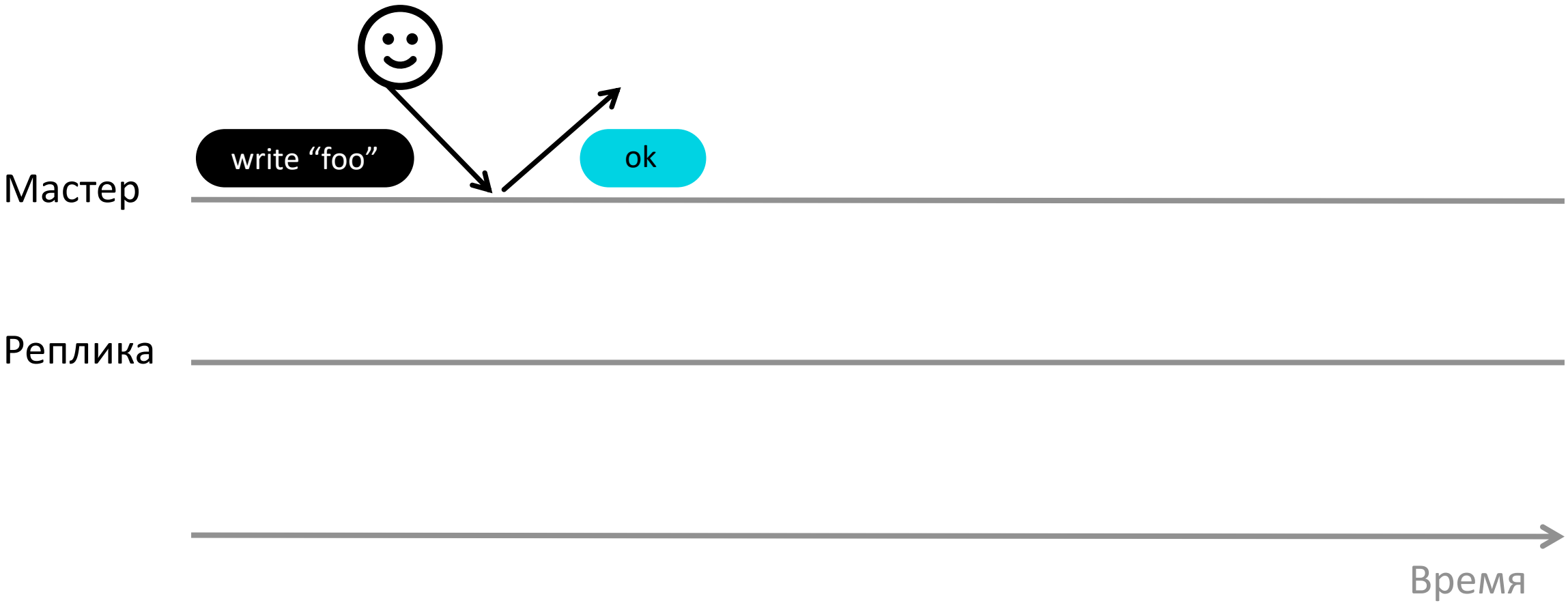
Реплика

Время

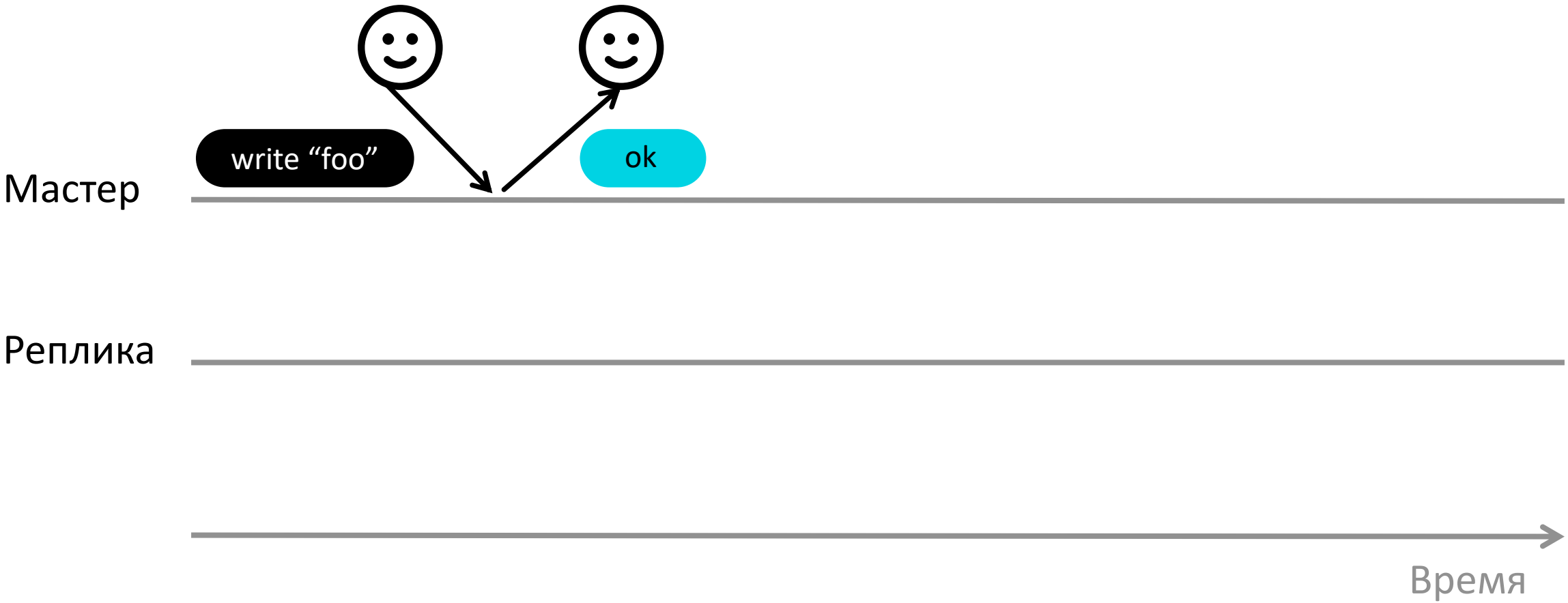
Чтение старых данных



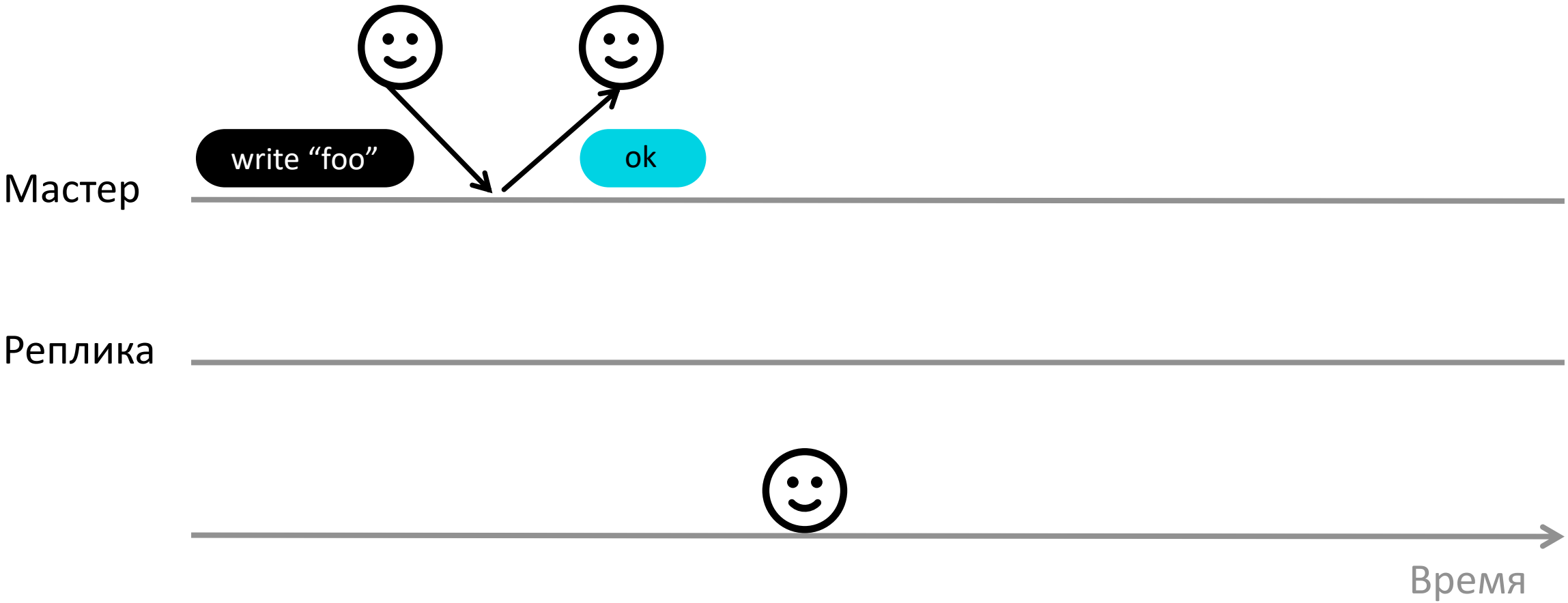
Чтение старых данных



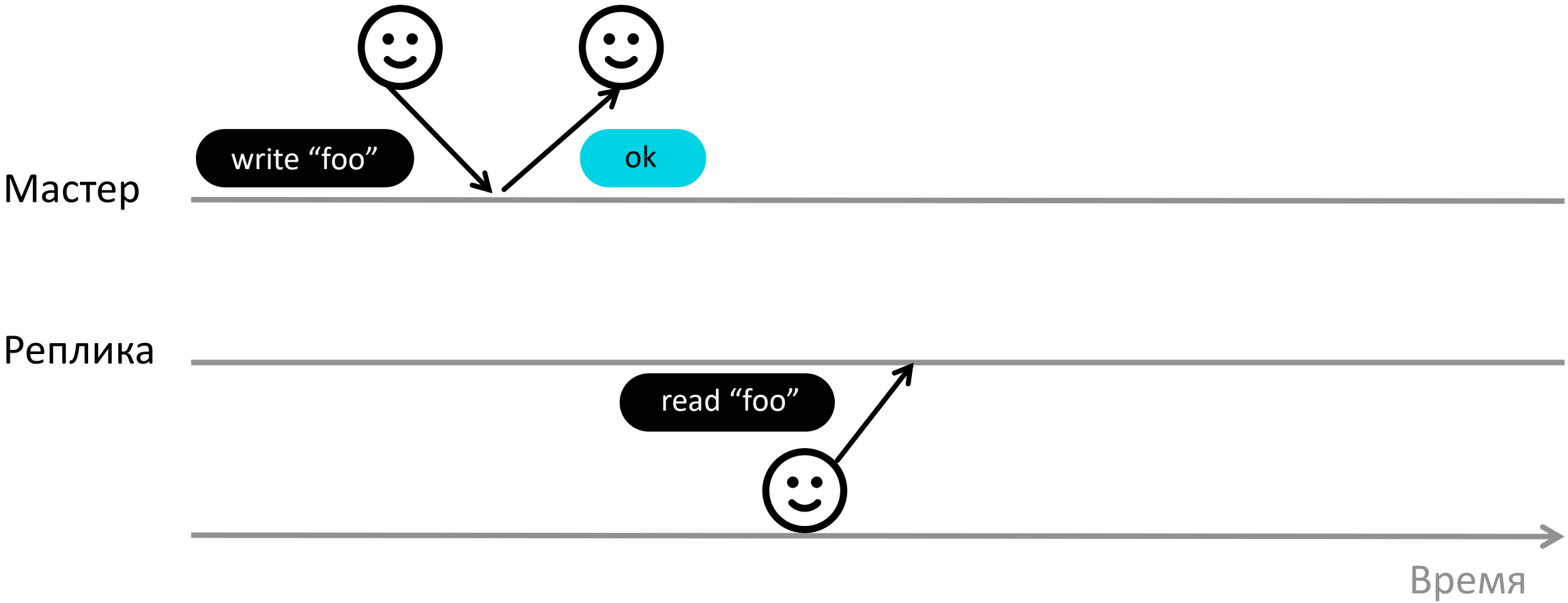
Чтение старых данных



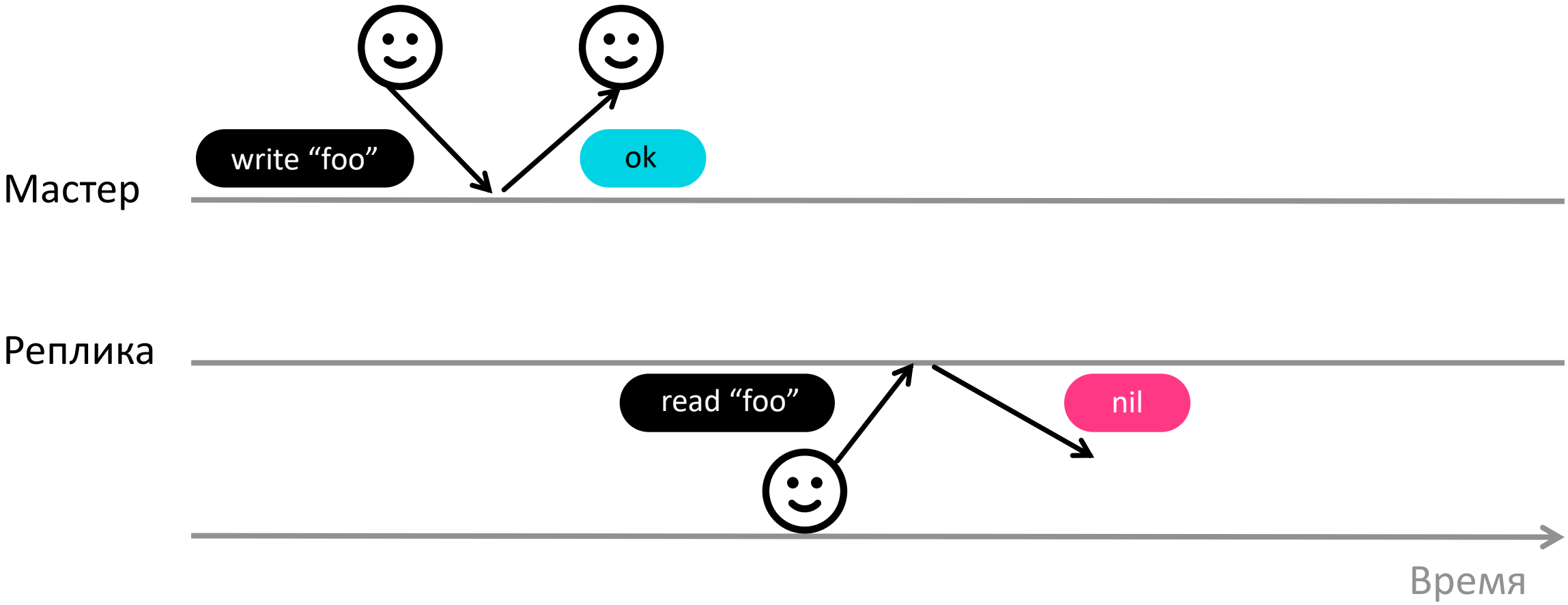
Чтение старых данных



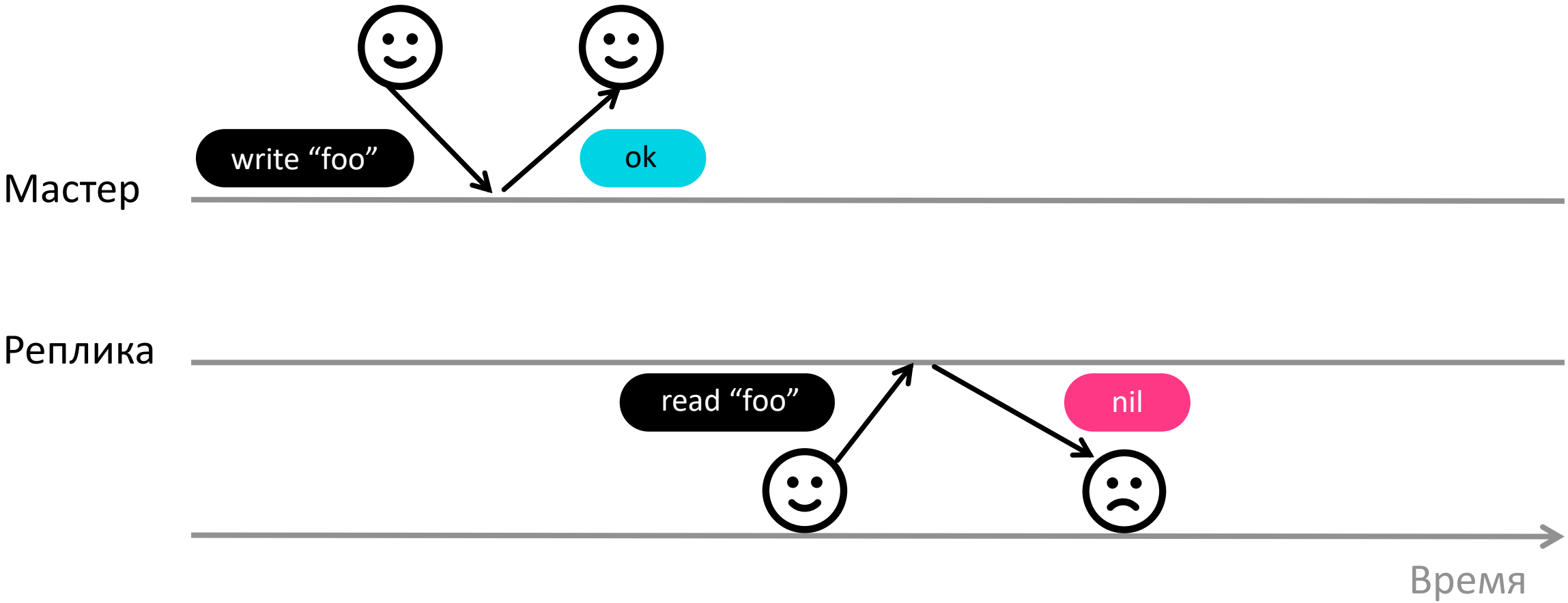
Чтение старых данных



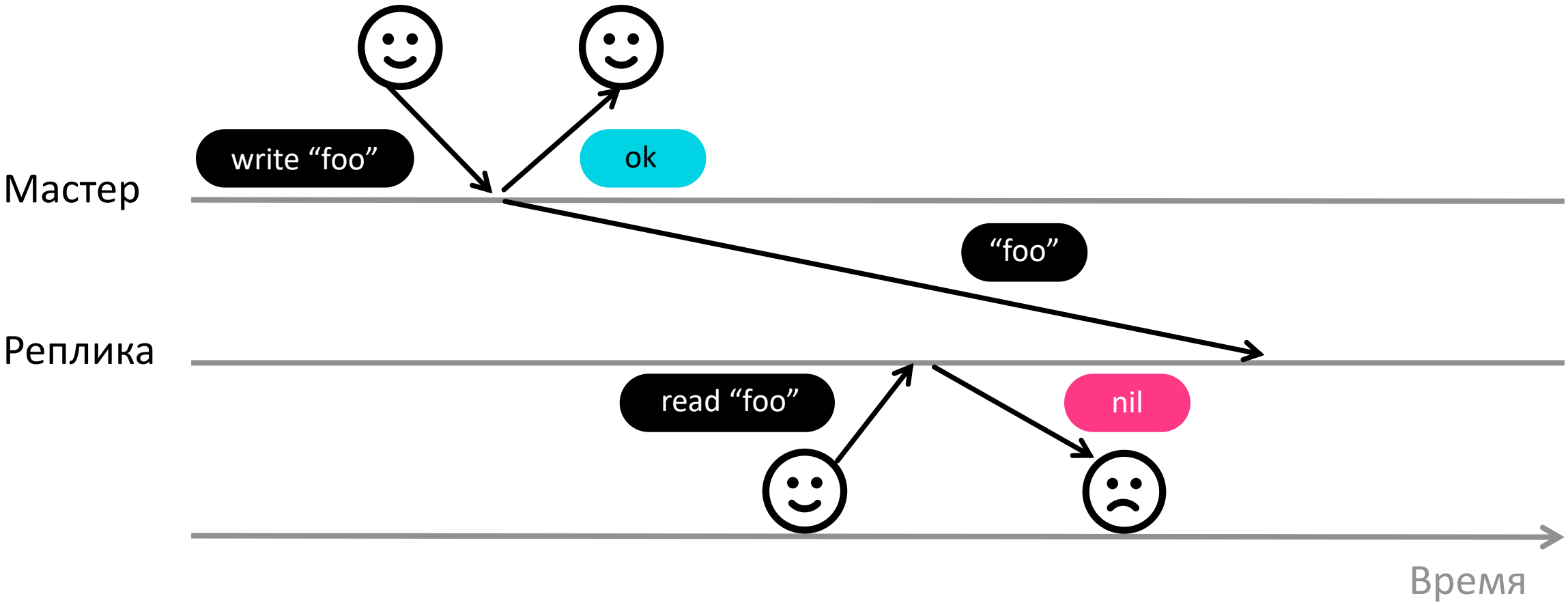
Чтение старых данных



Чтение старых данных



Чтение старых данных



Последствия асинхронной репликации

Чтение старых данных

Потеря данных

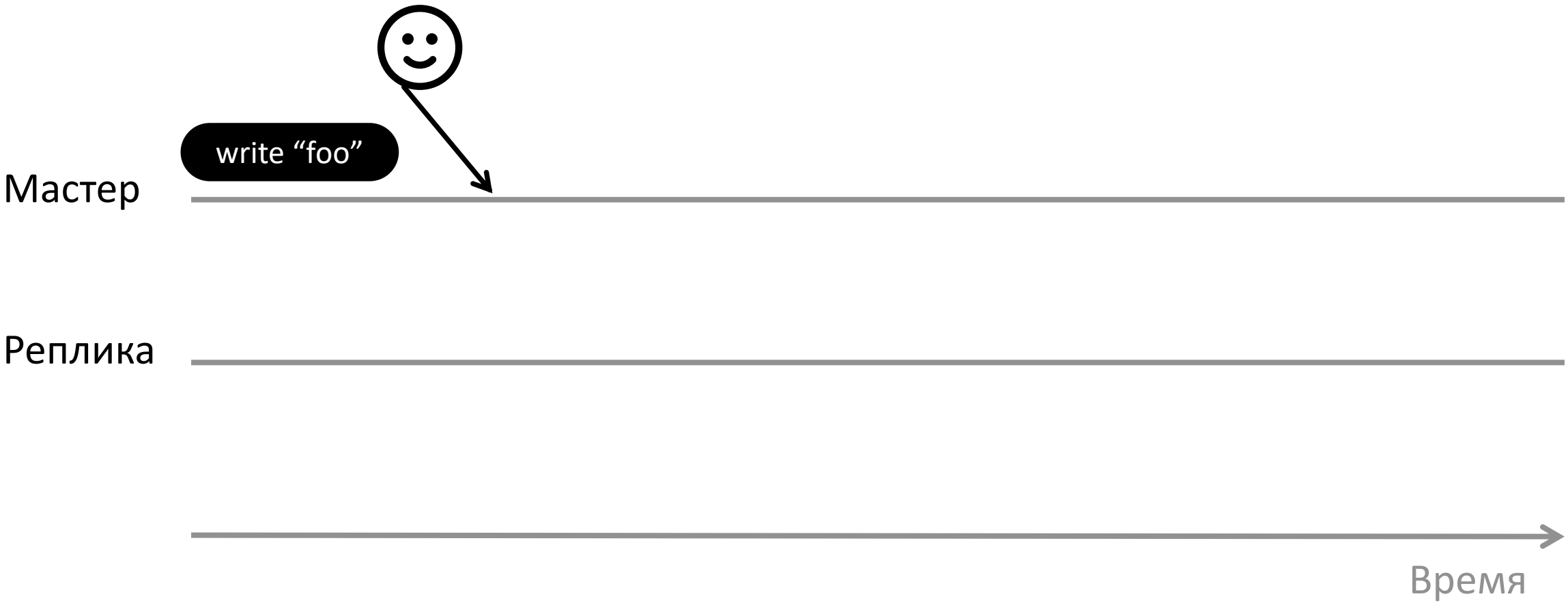
Потеря данных

Мастер

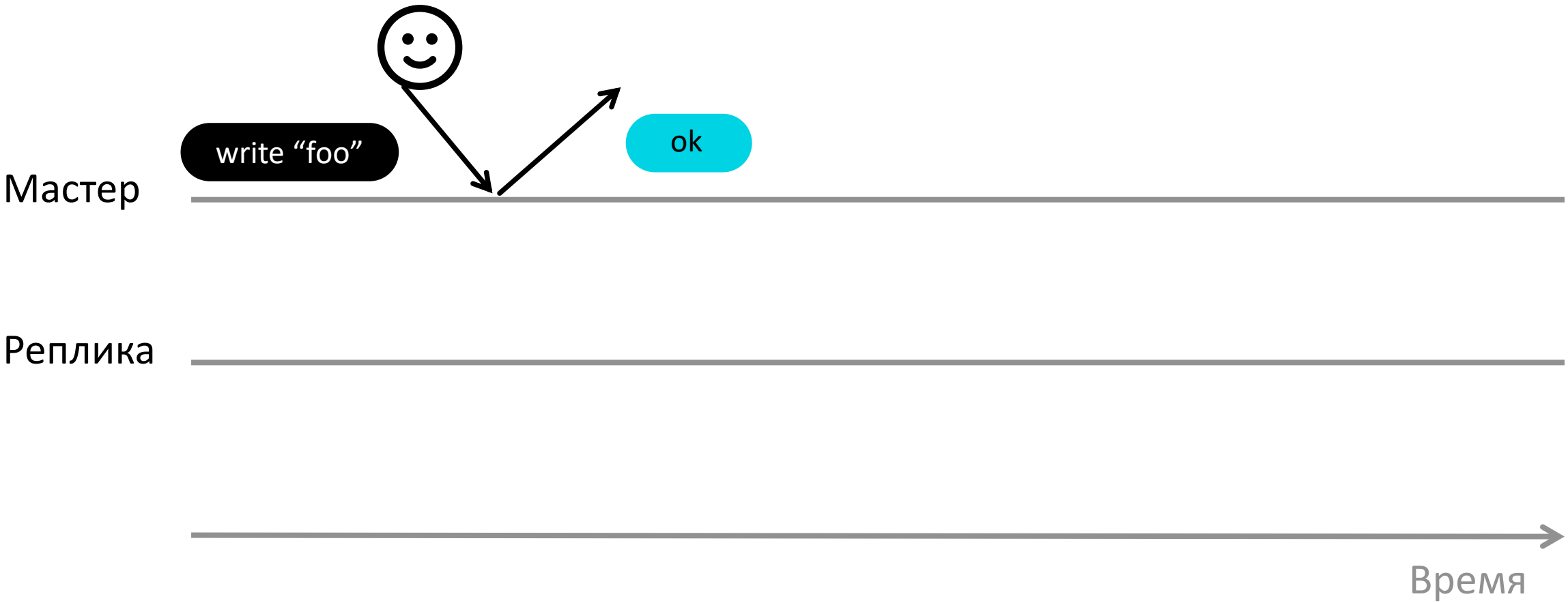
Реплика

Время

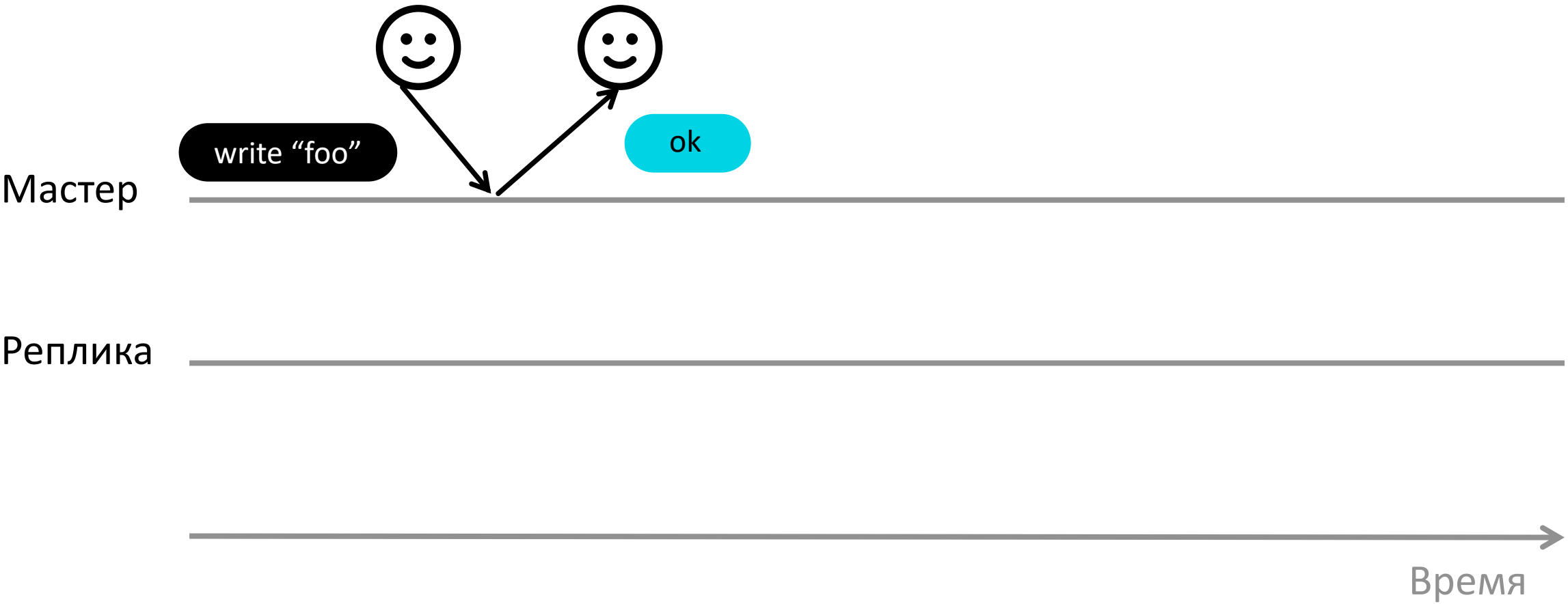
Потеря данных



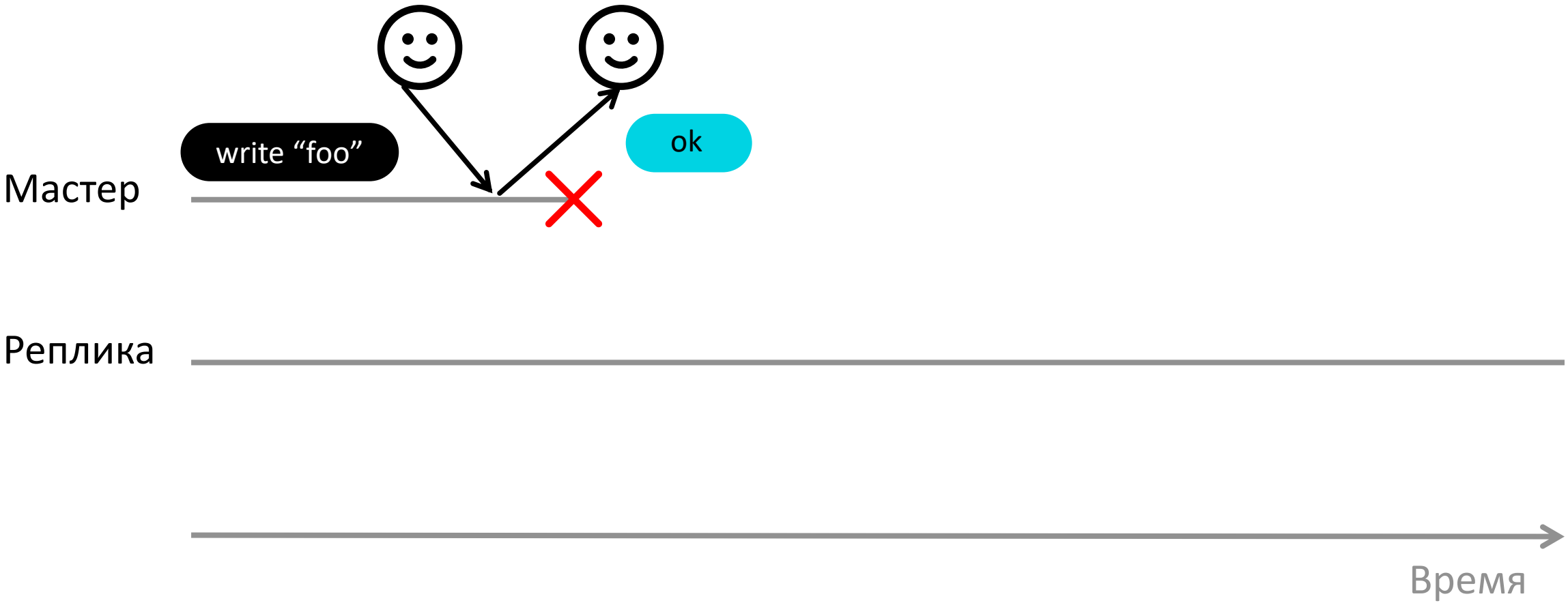
Потеря данных



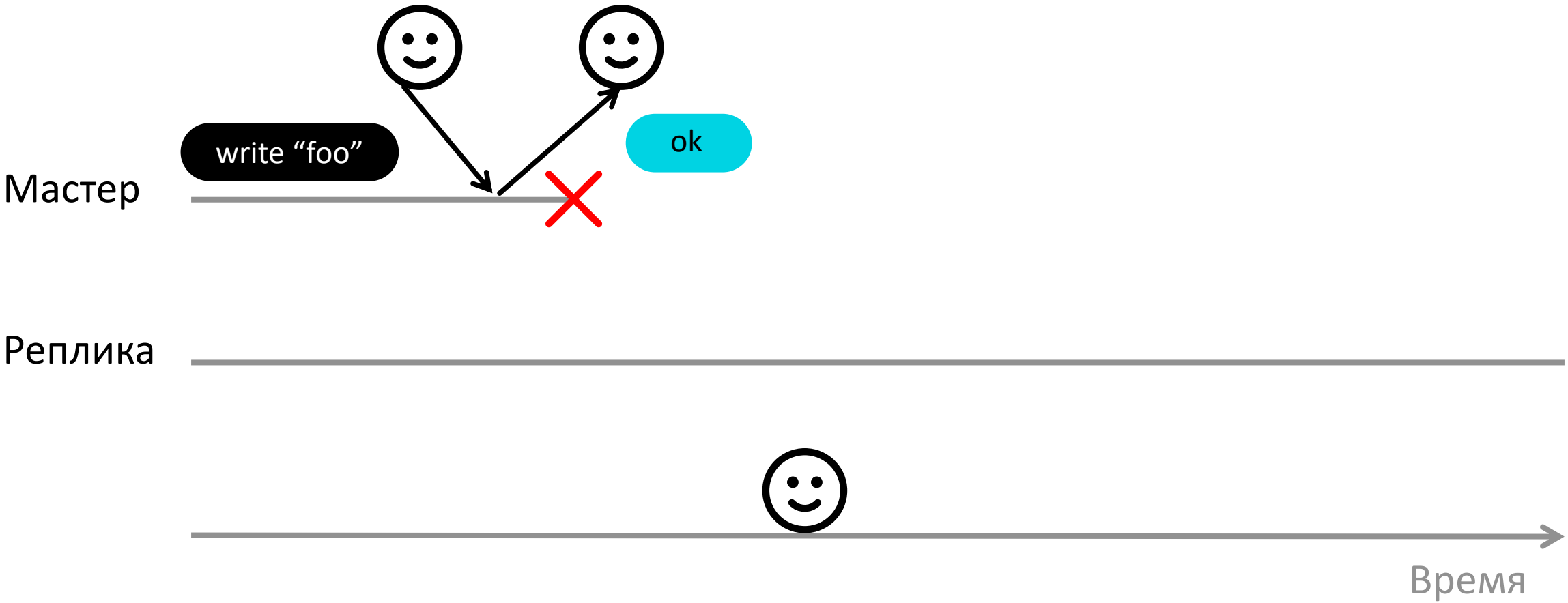
Потеря данных



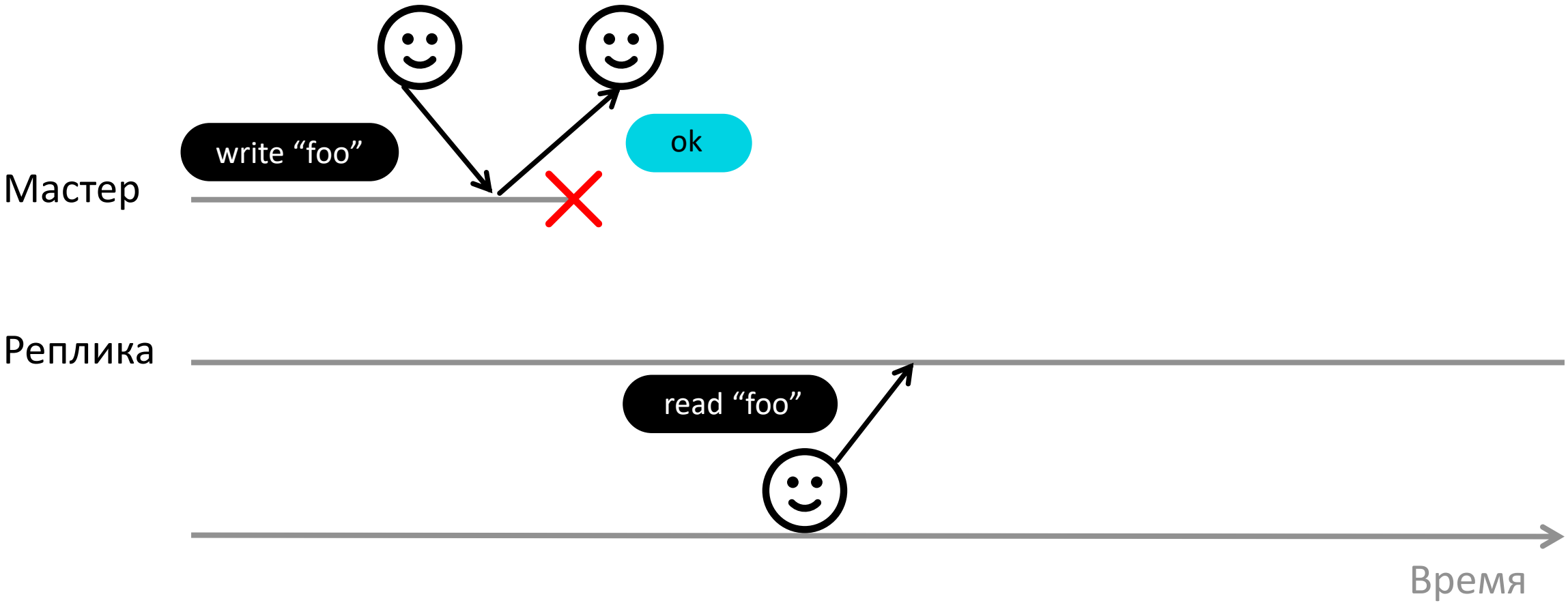
Потеря данных



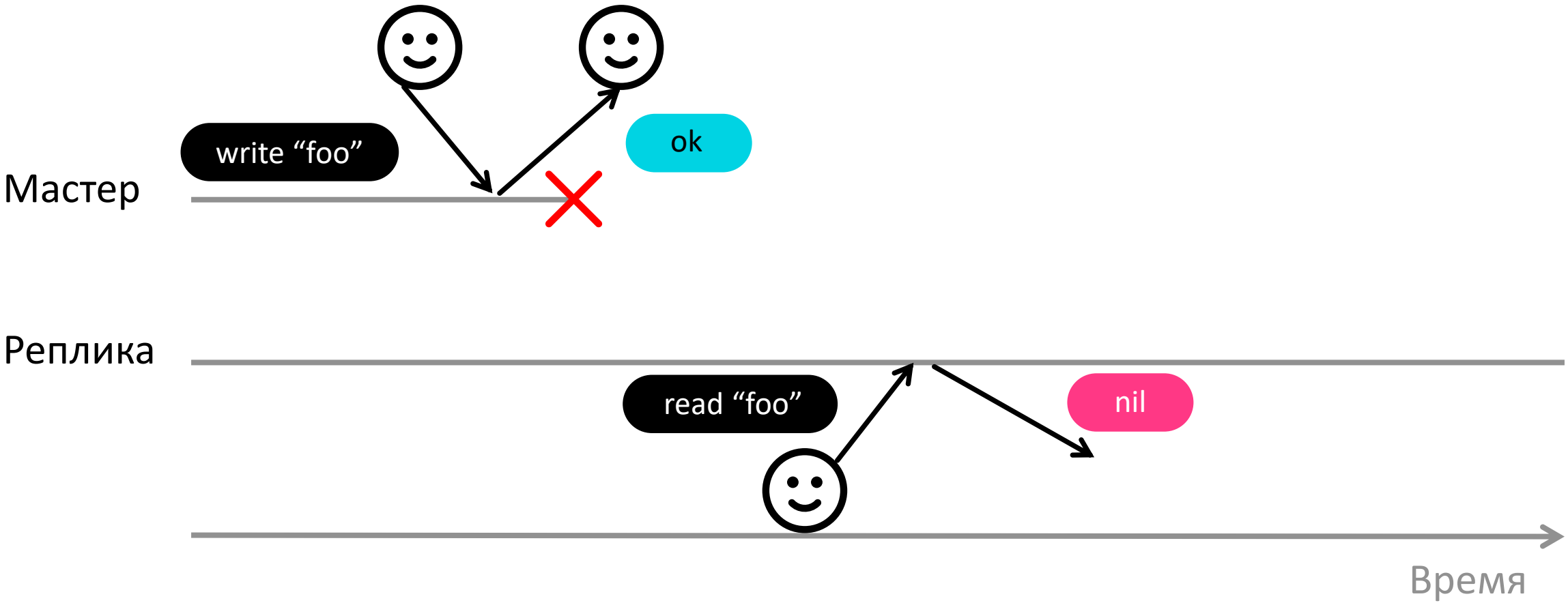
Потеря данных



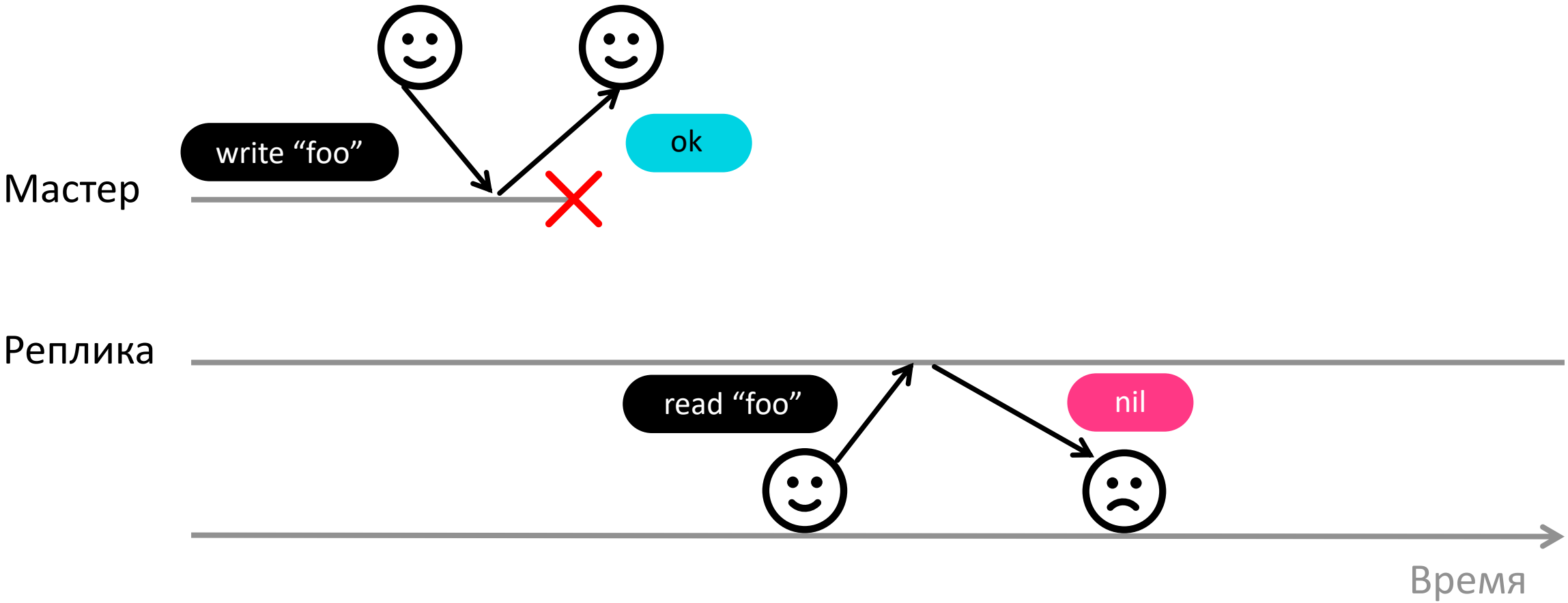
Потеря данных



Потеря данных



Потеря данных



Последствия асинхронной репликации

Чтение старых данных

Потеря данных

Конфликты

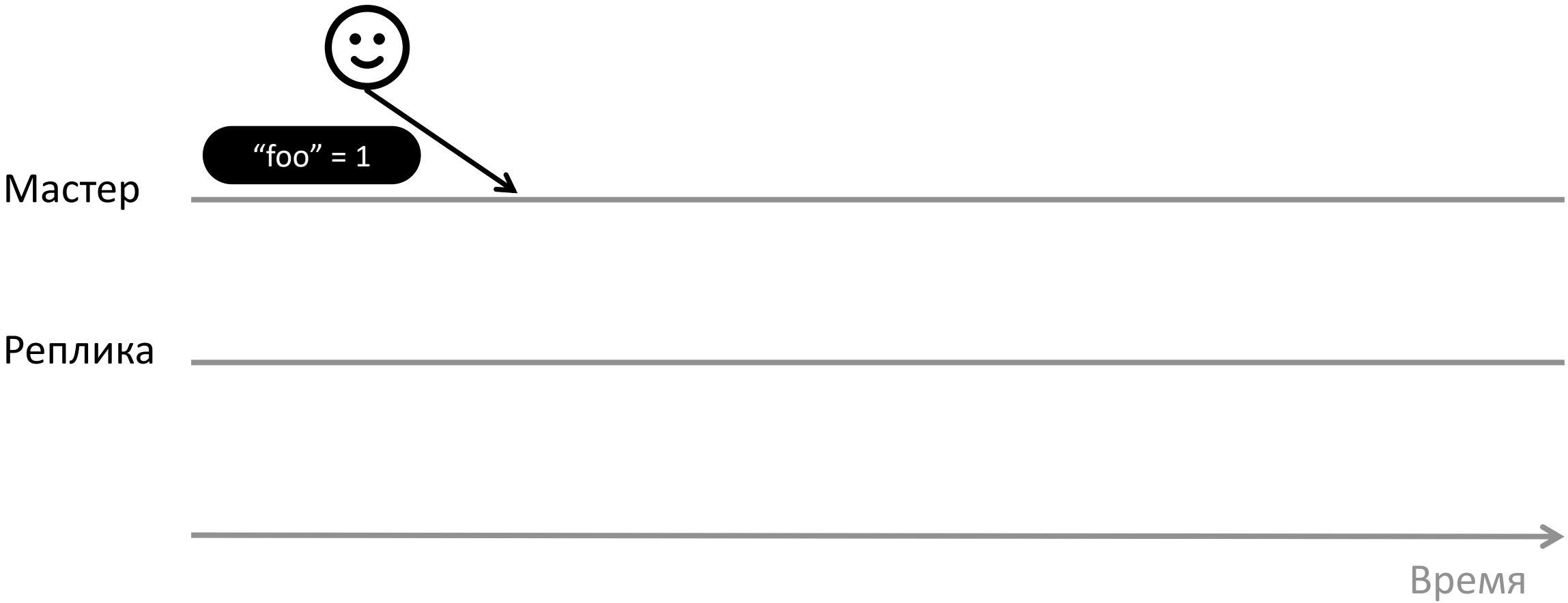
Конфликты

Мастер

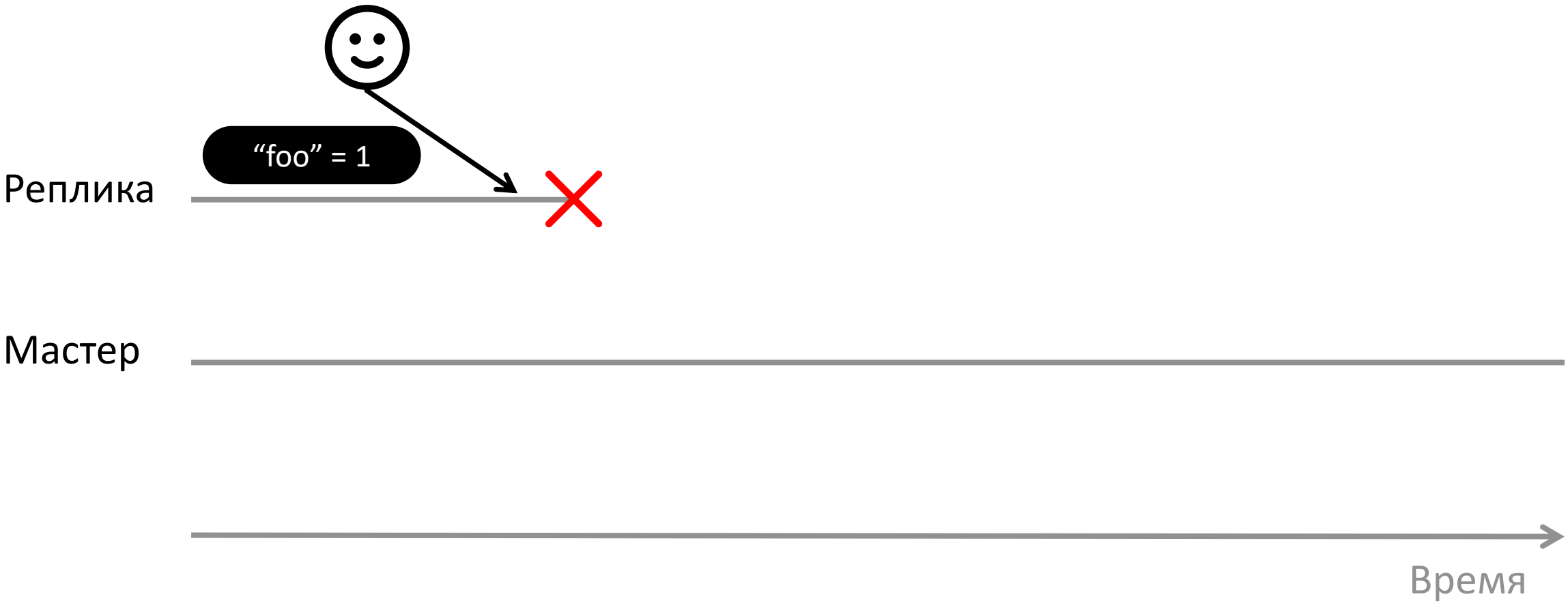
Реплика

Время

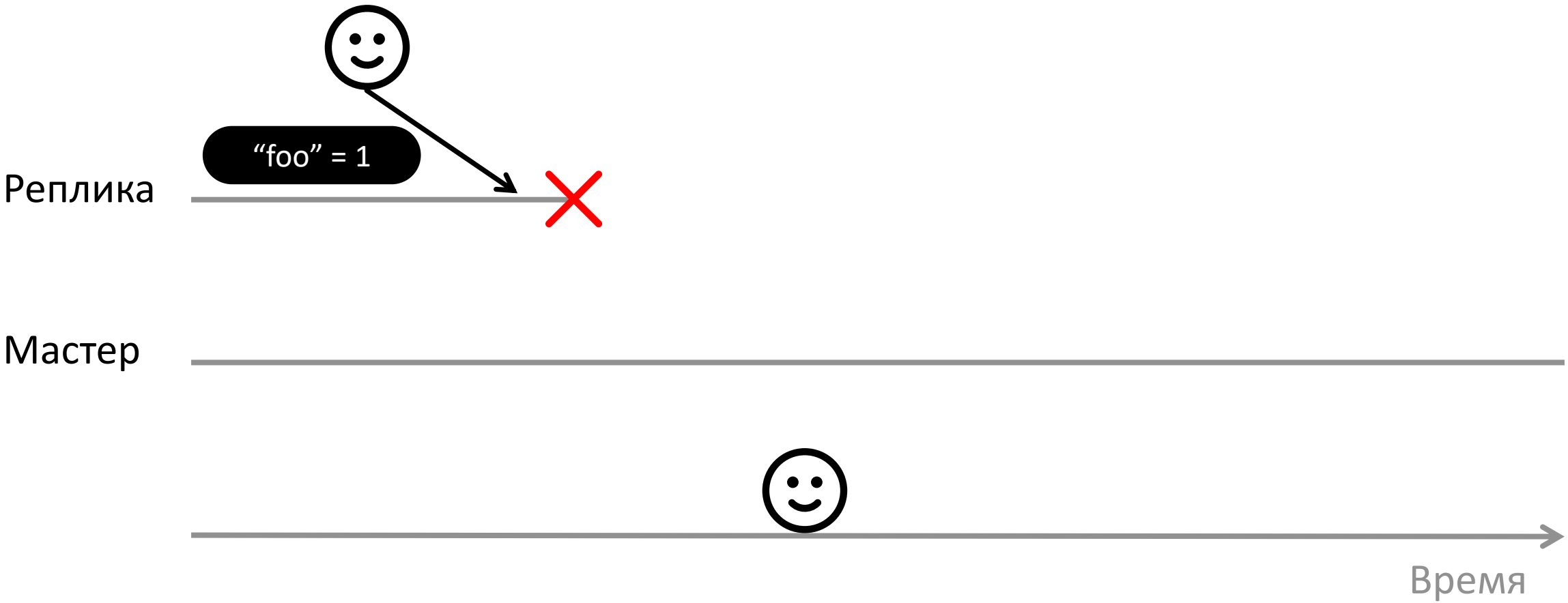
Конфликты



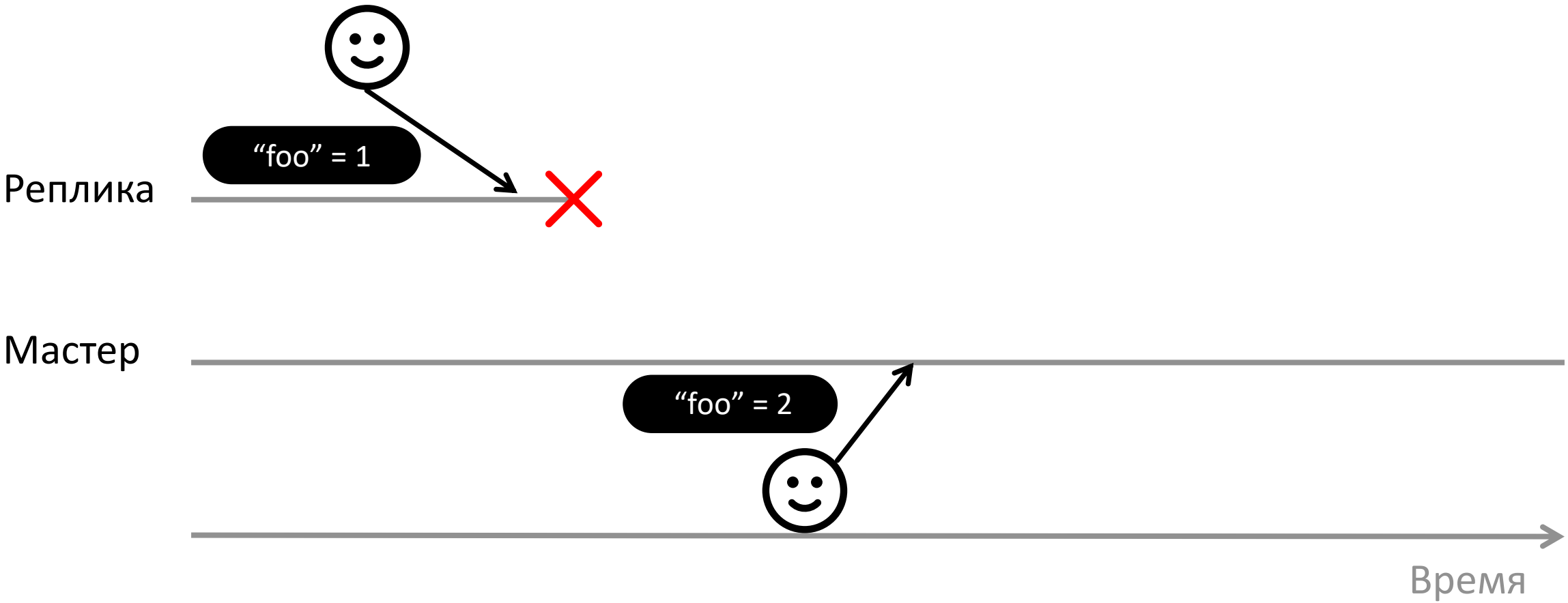
Конфликты



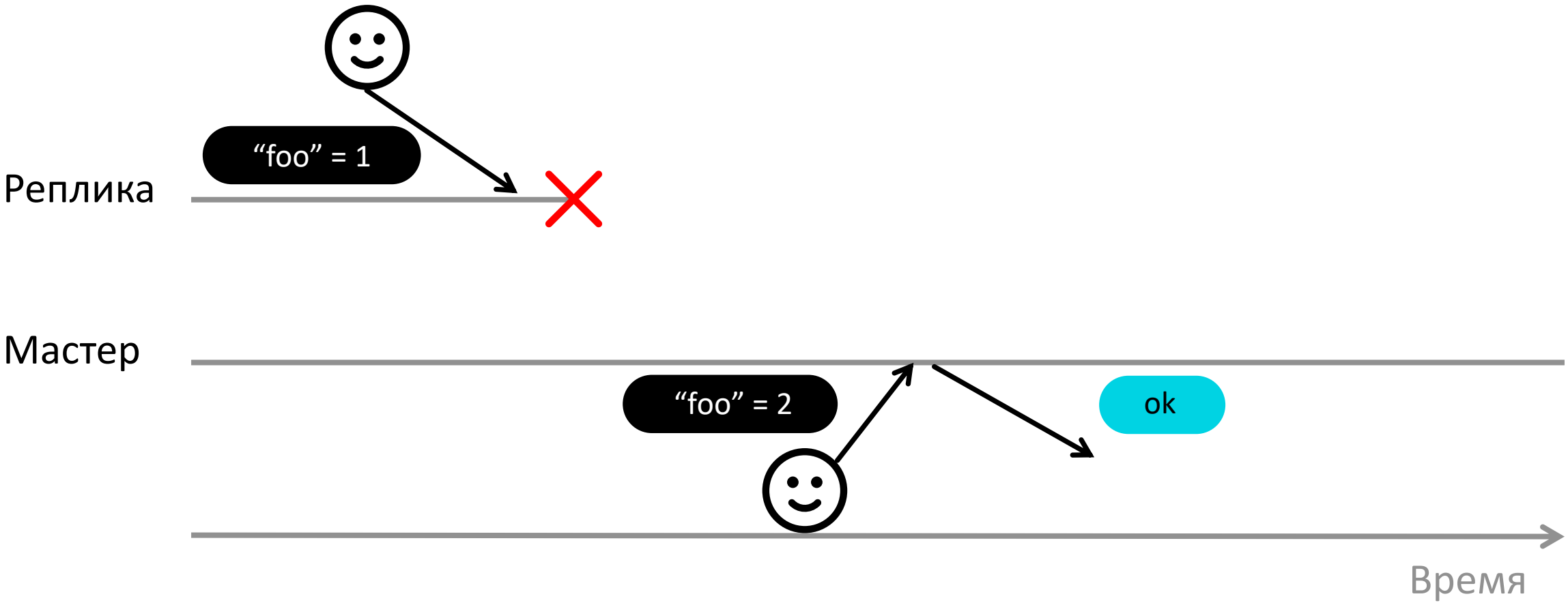
Конфликты



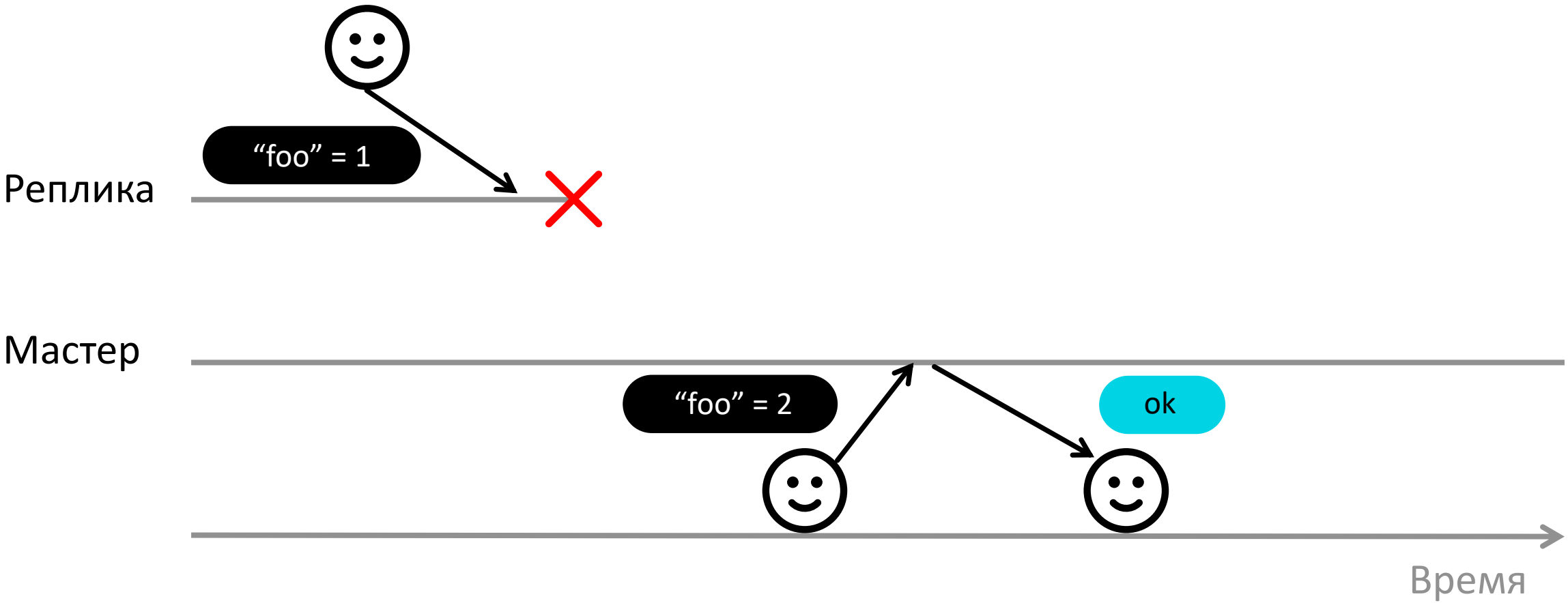
Конфликты



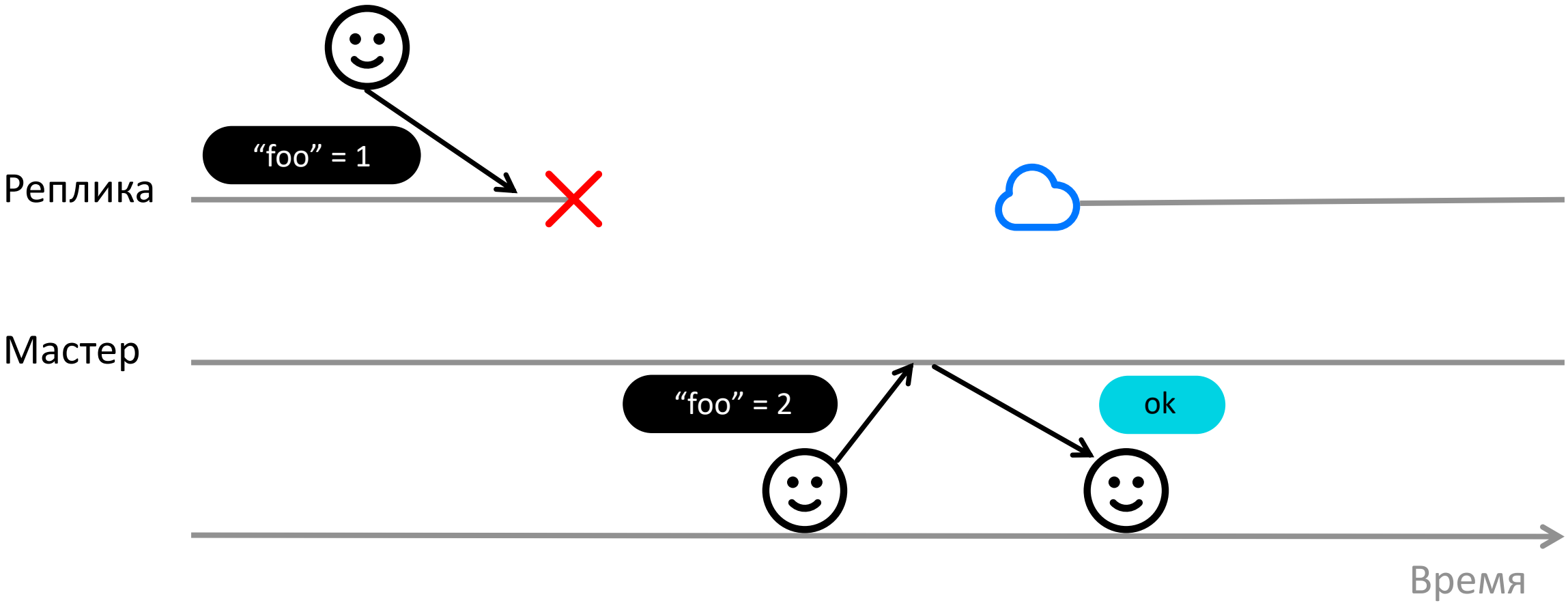
Конфликты



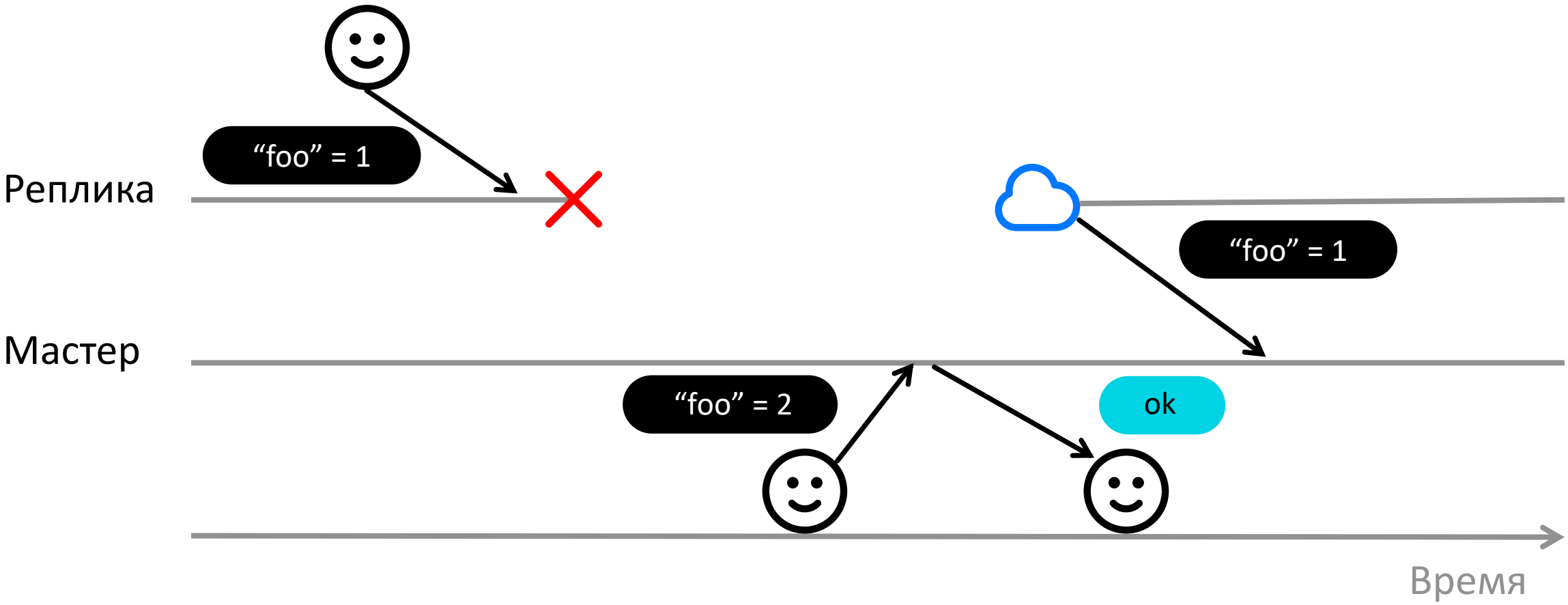
Конфликты



Конфликты



Конфликты



Синхронная репликация

Копируем на реплики прежде, чем ответить клиенту

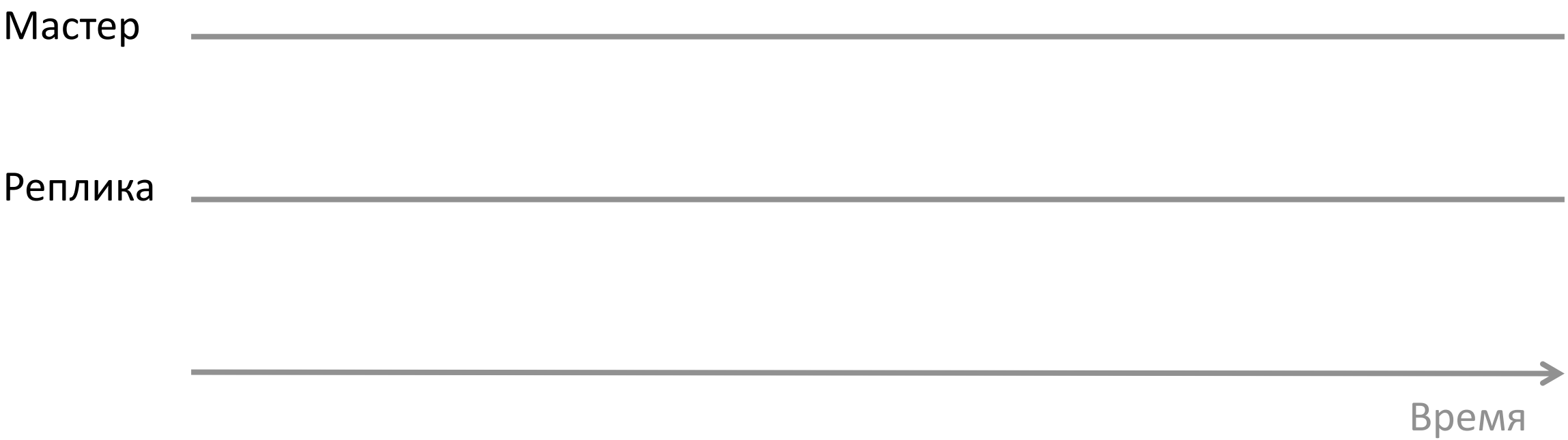
Последствия синхронной репликации

Потеря данных

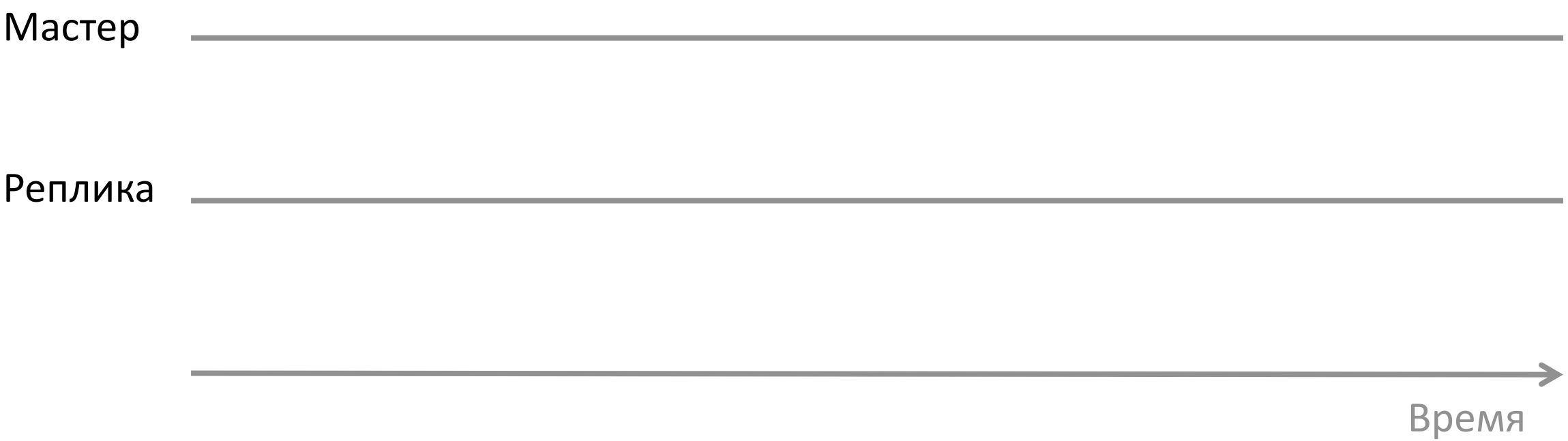
Последствия синхронной репликации

~~Потеря данных~~

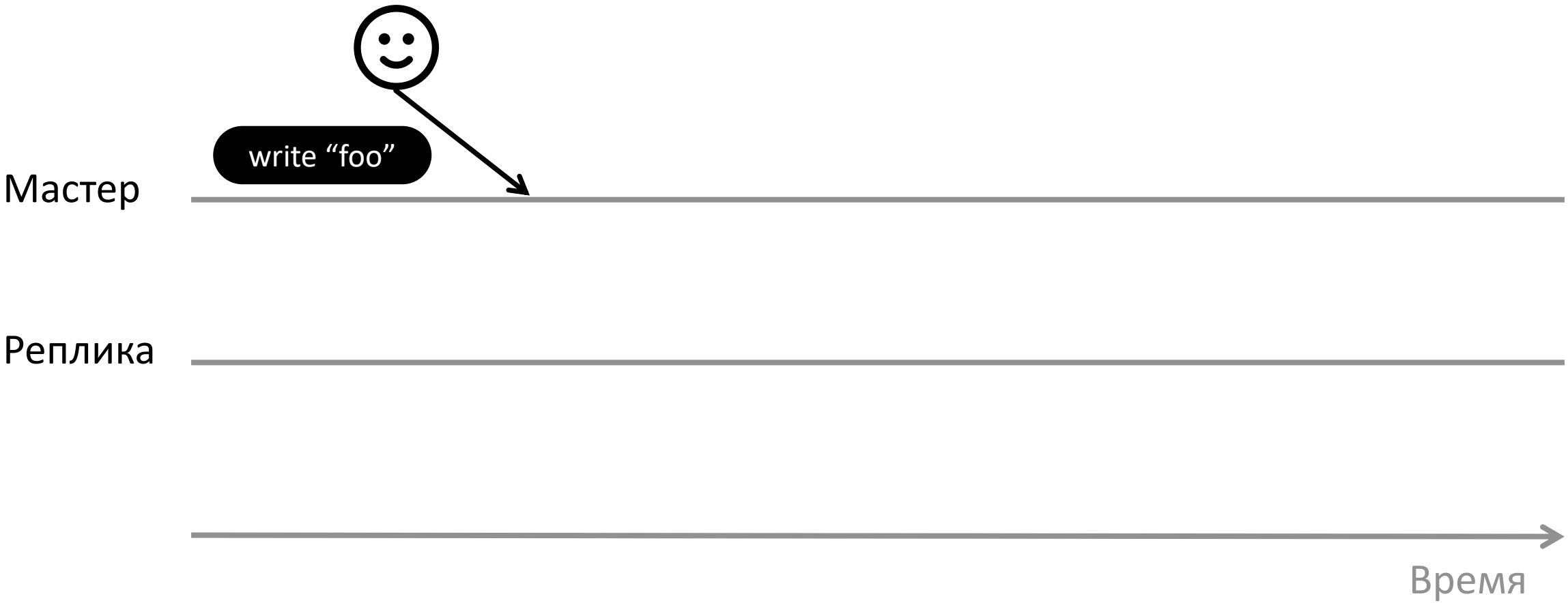
Нет потери данных



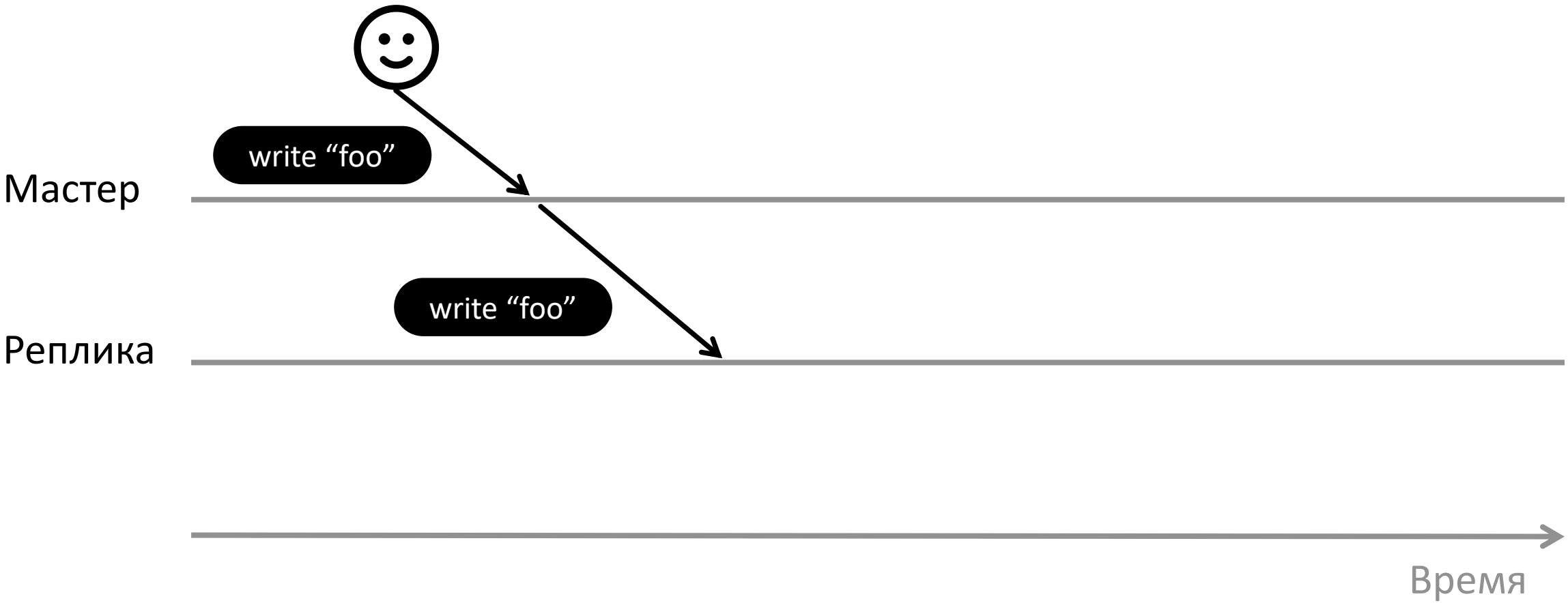
Нет потери данных



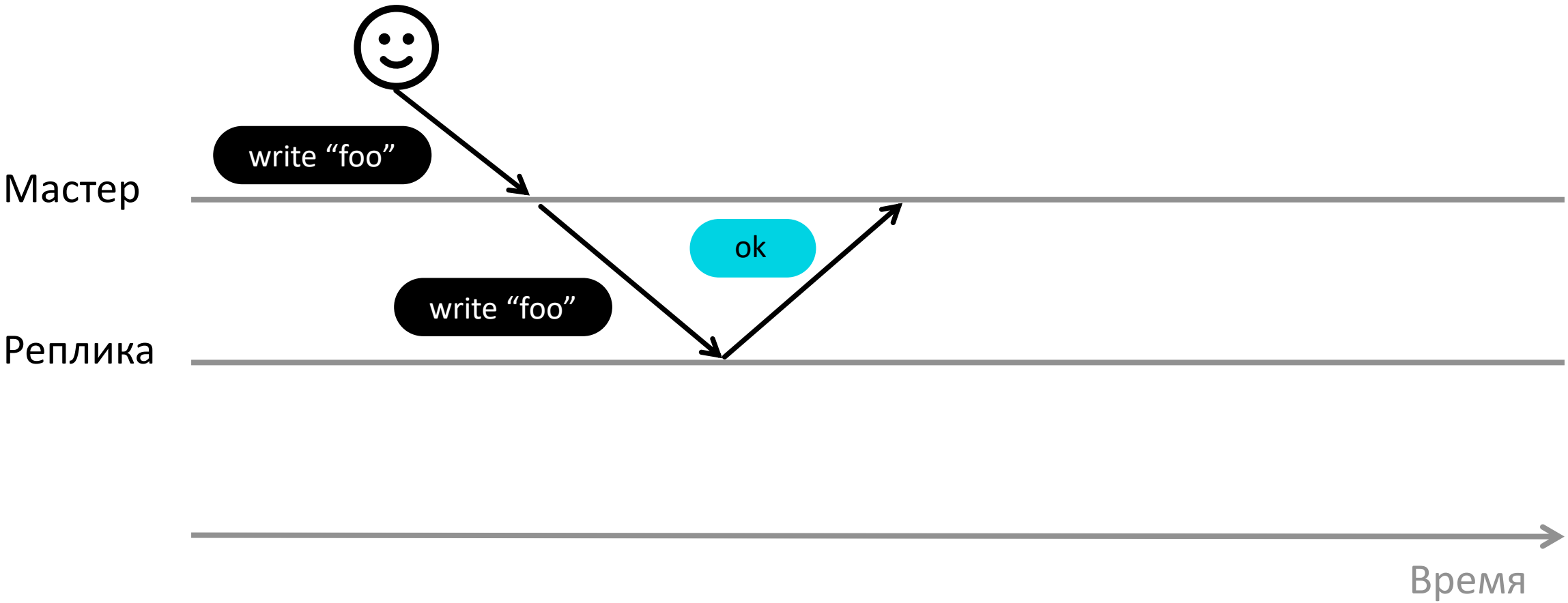
Нет потери данных



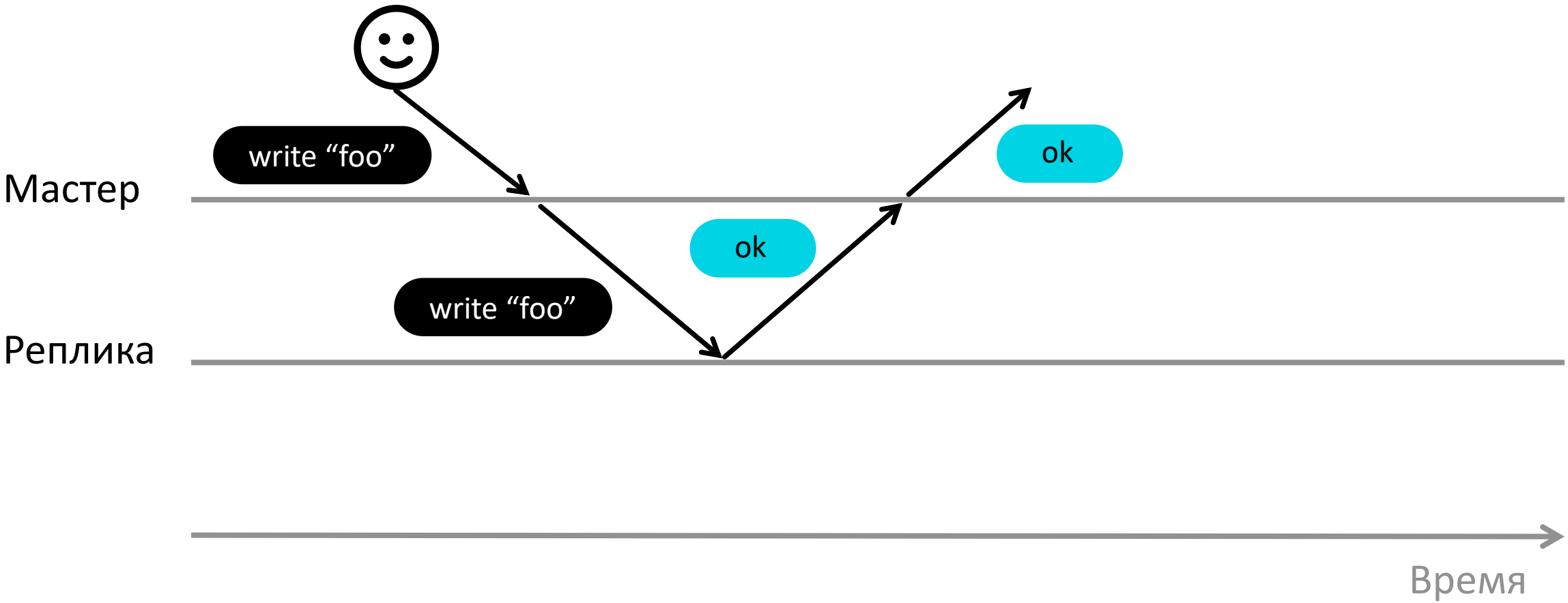
Нет потери данных



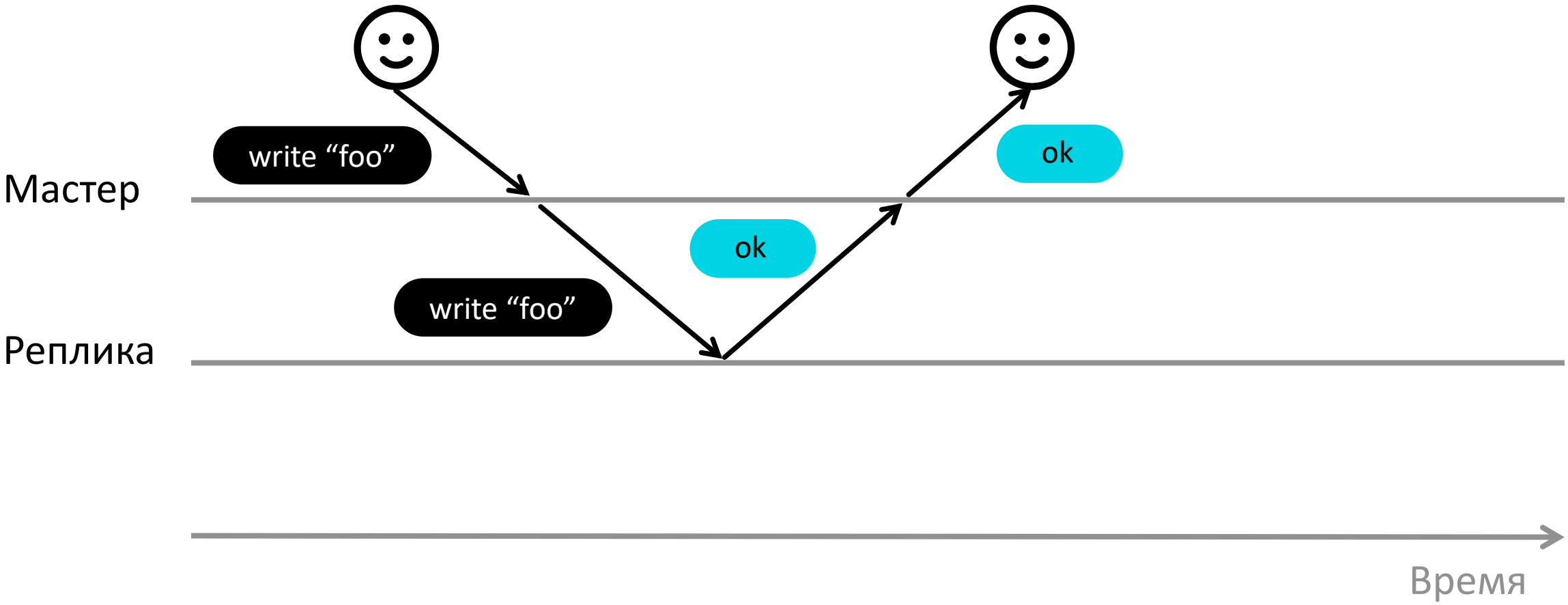
Нет потери данных



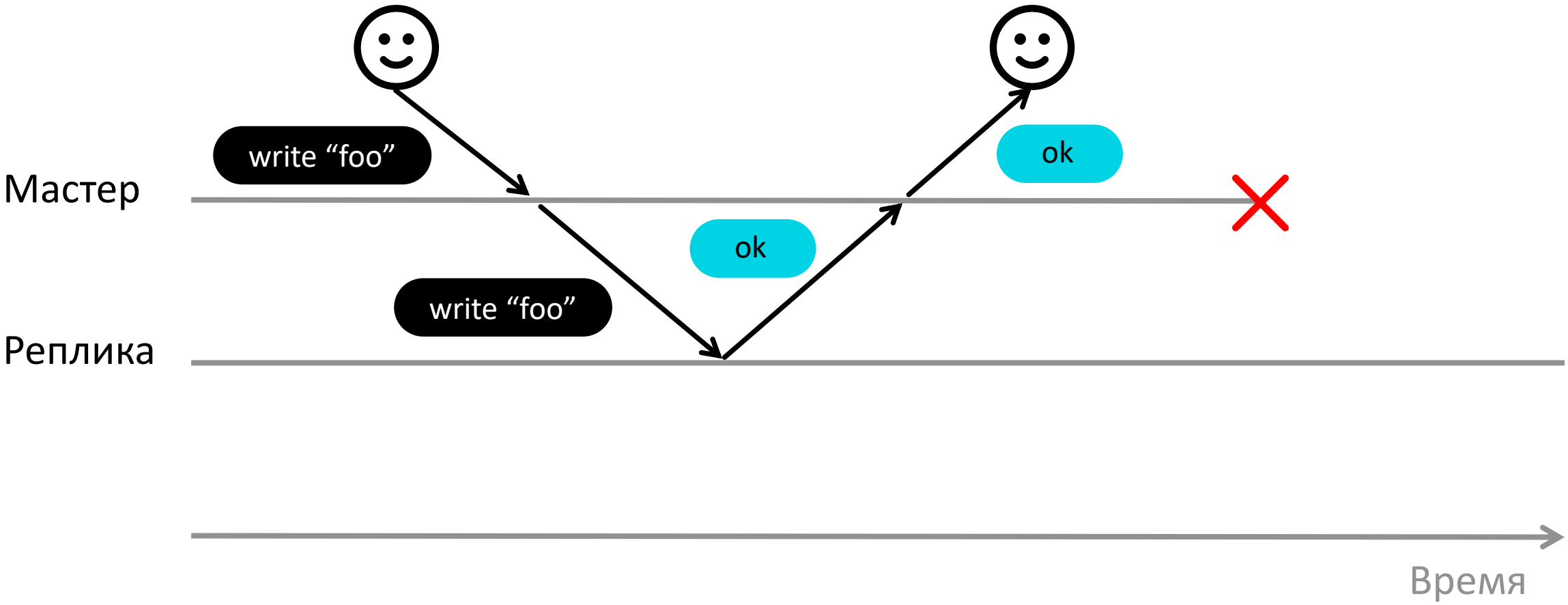
Нет потери данных



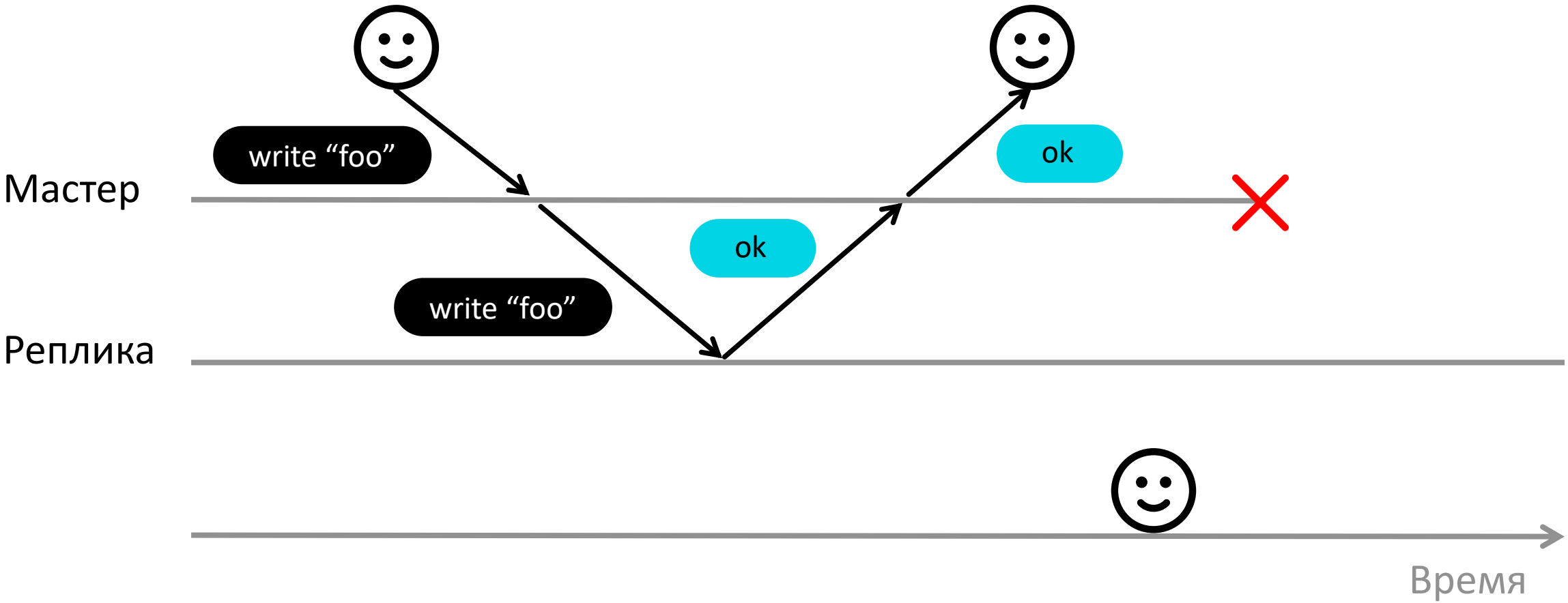
Нет потери данных



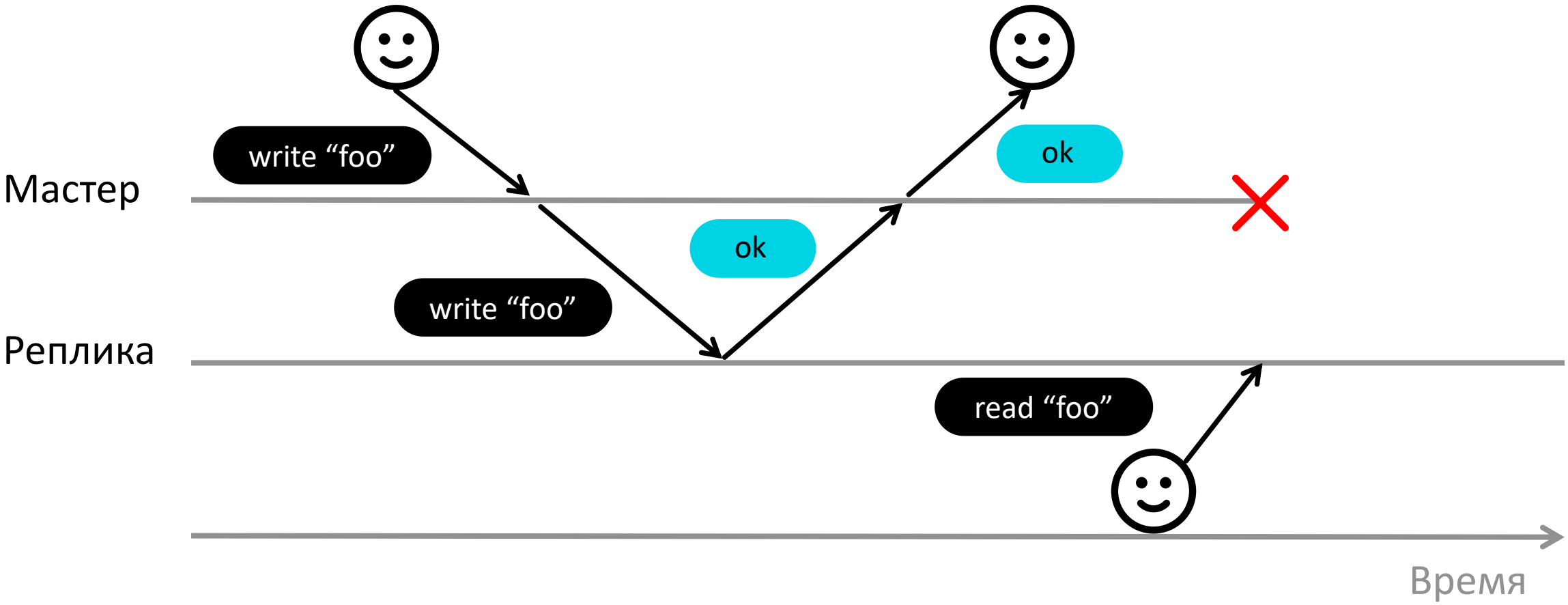
Нет потери данных



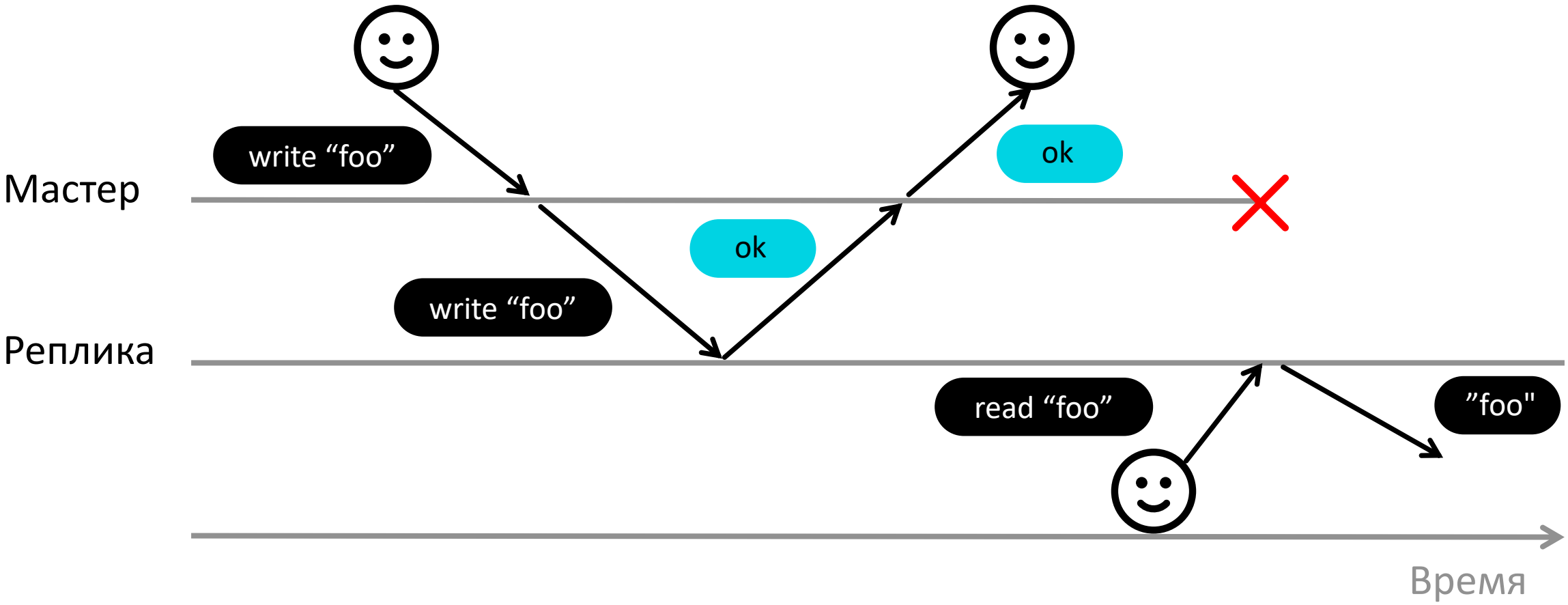
Нет потери данных



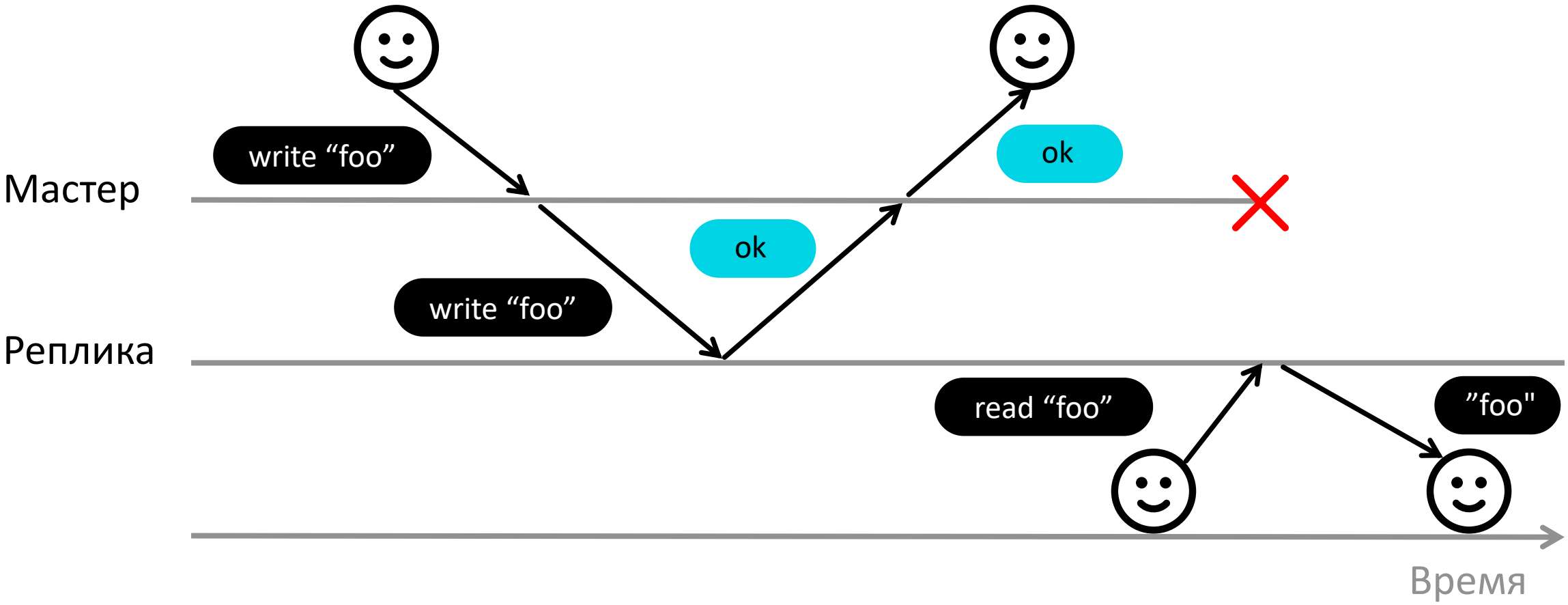
Нет потери данных



Нет потери данных



Нет потери данных



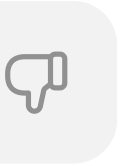
Последствия синхронной репликации

~~Потеря данных~~

Конфликты

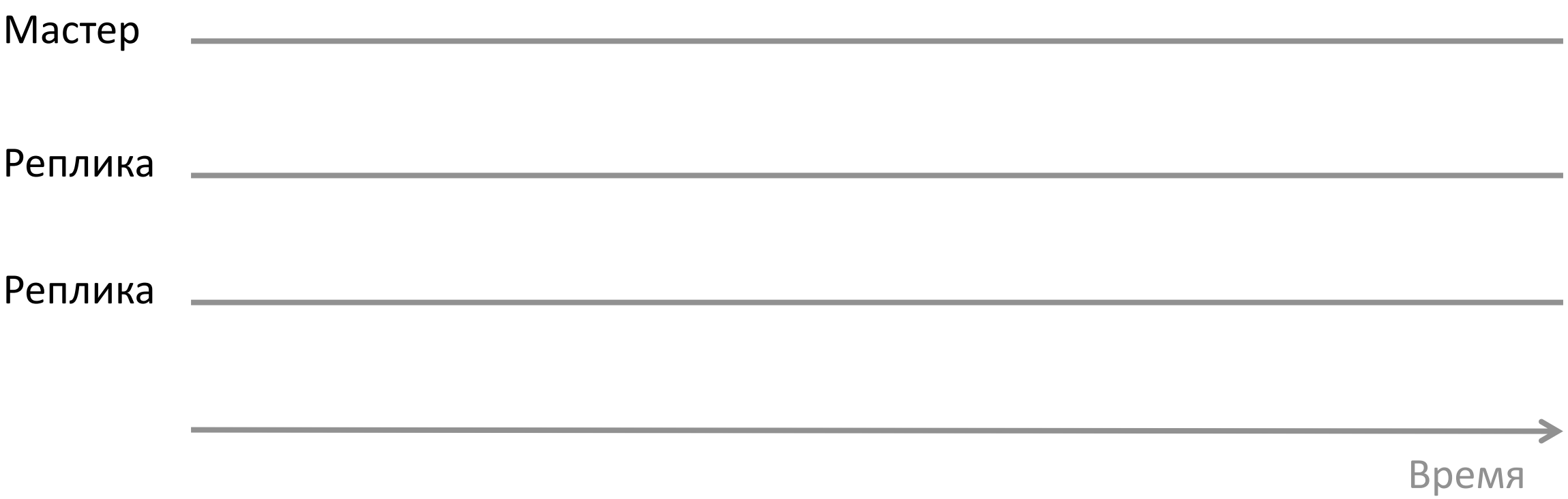
Последствия синхронной репликации

~~Потеря данных~~

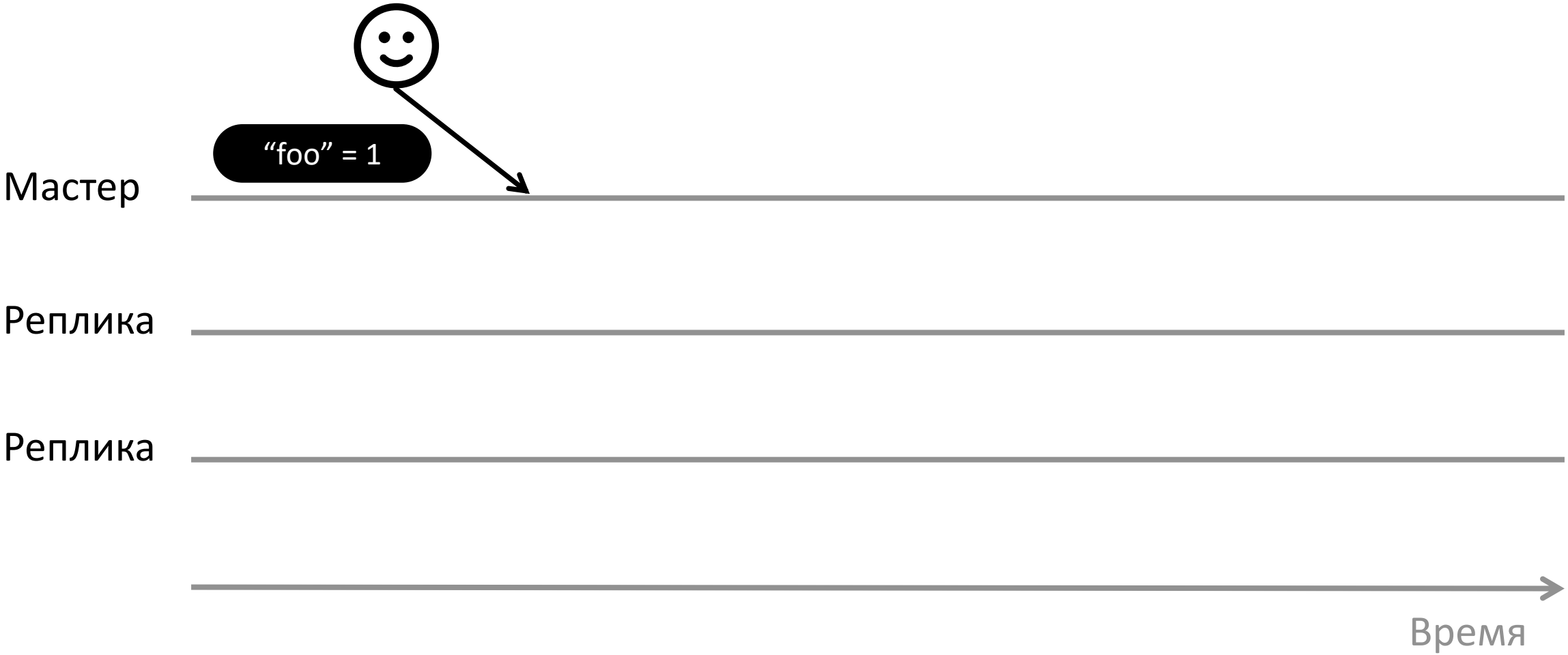


~~Конфликты~~

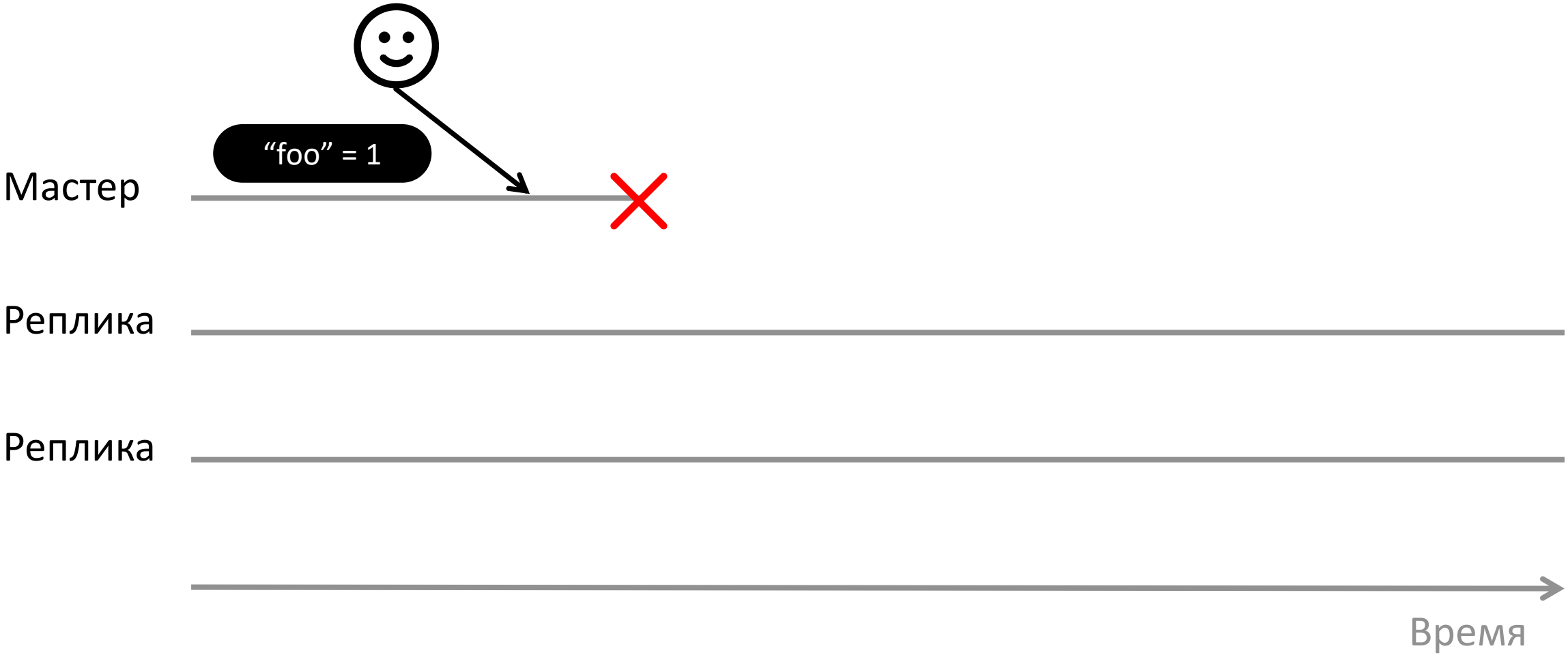
Нет конфликтов



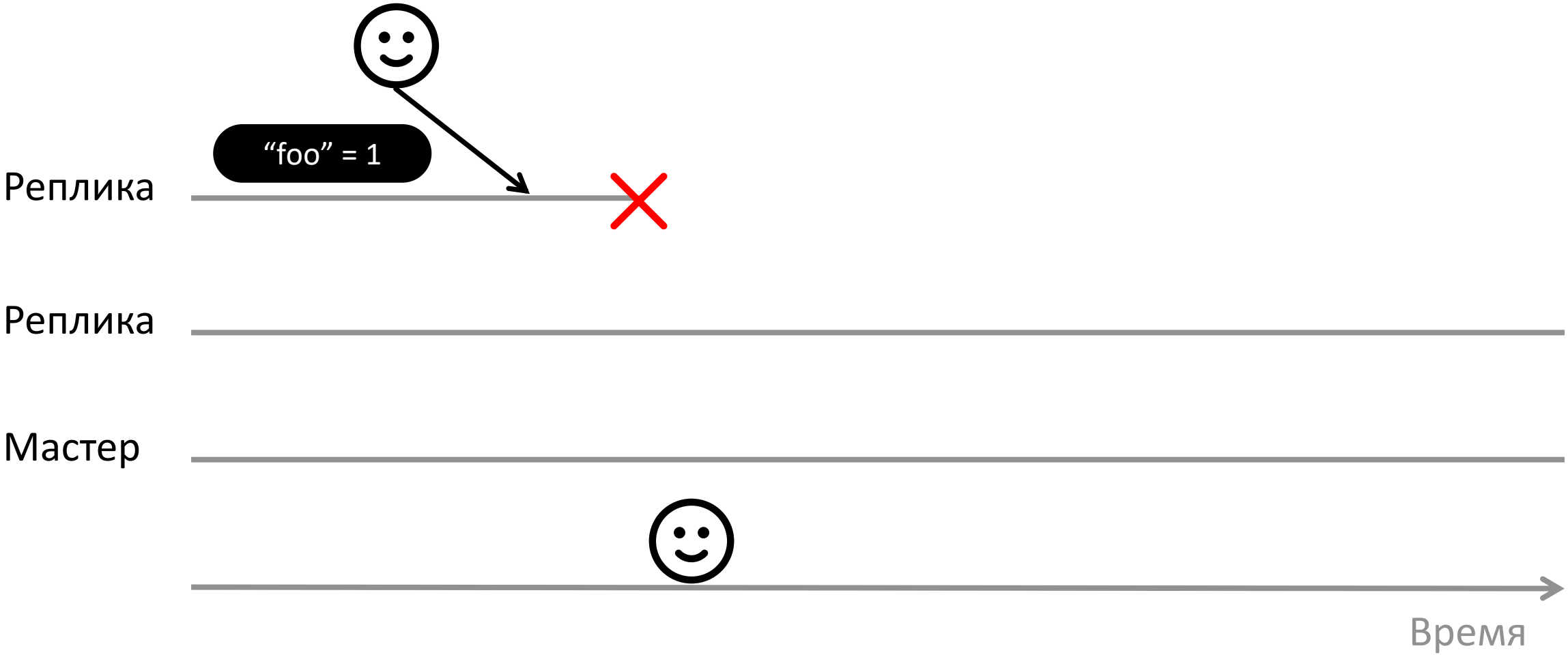
Нет конфликтов



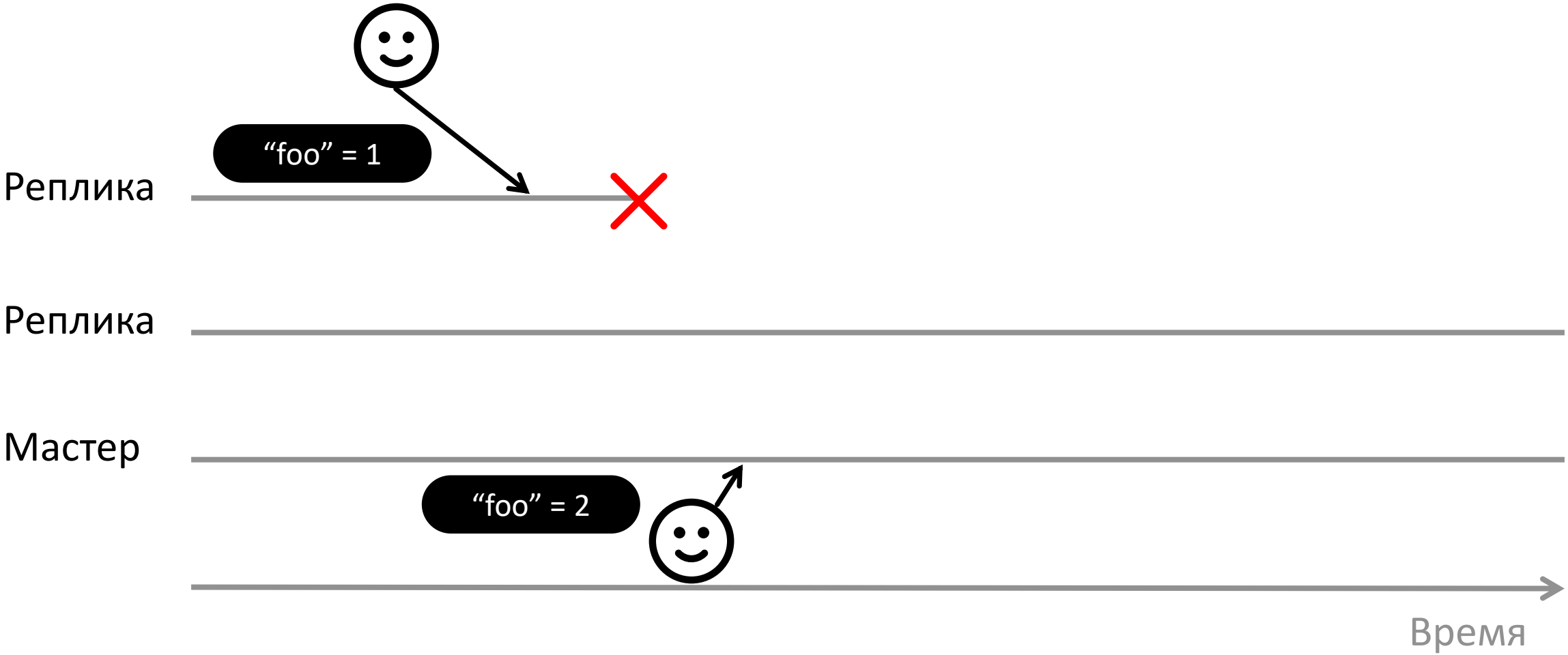
Нет конфликтов



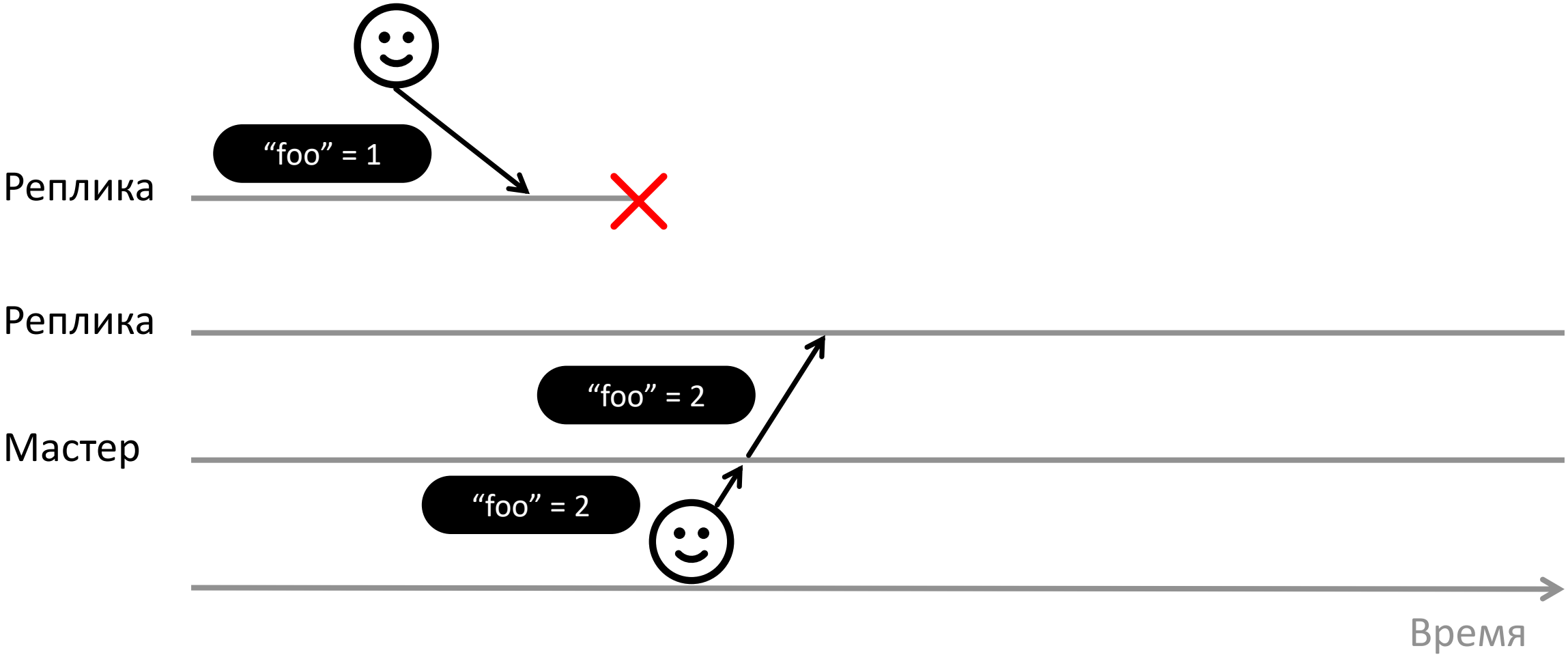
Нет конфликтов



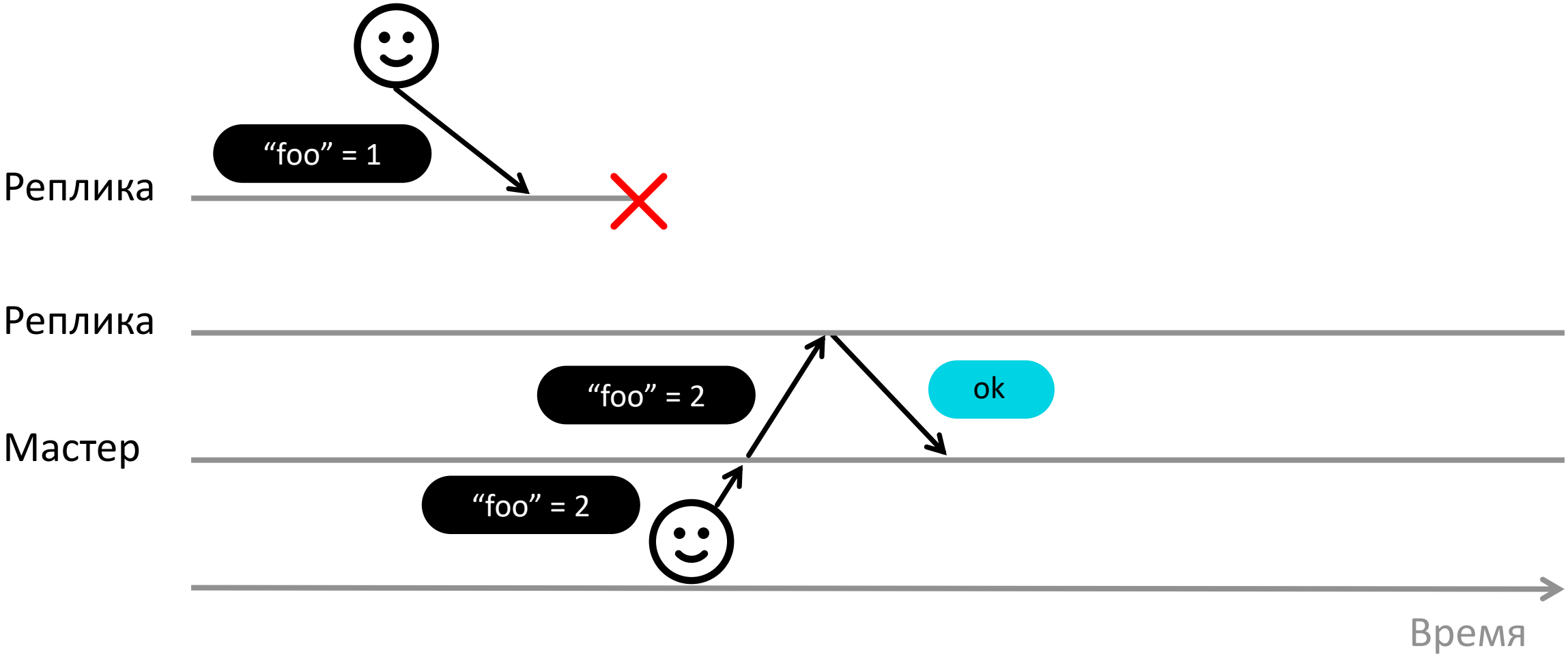
Нет конфликтов



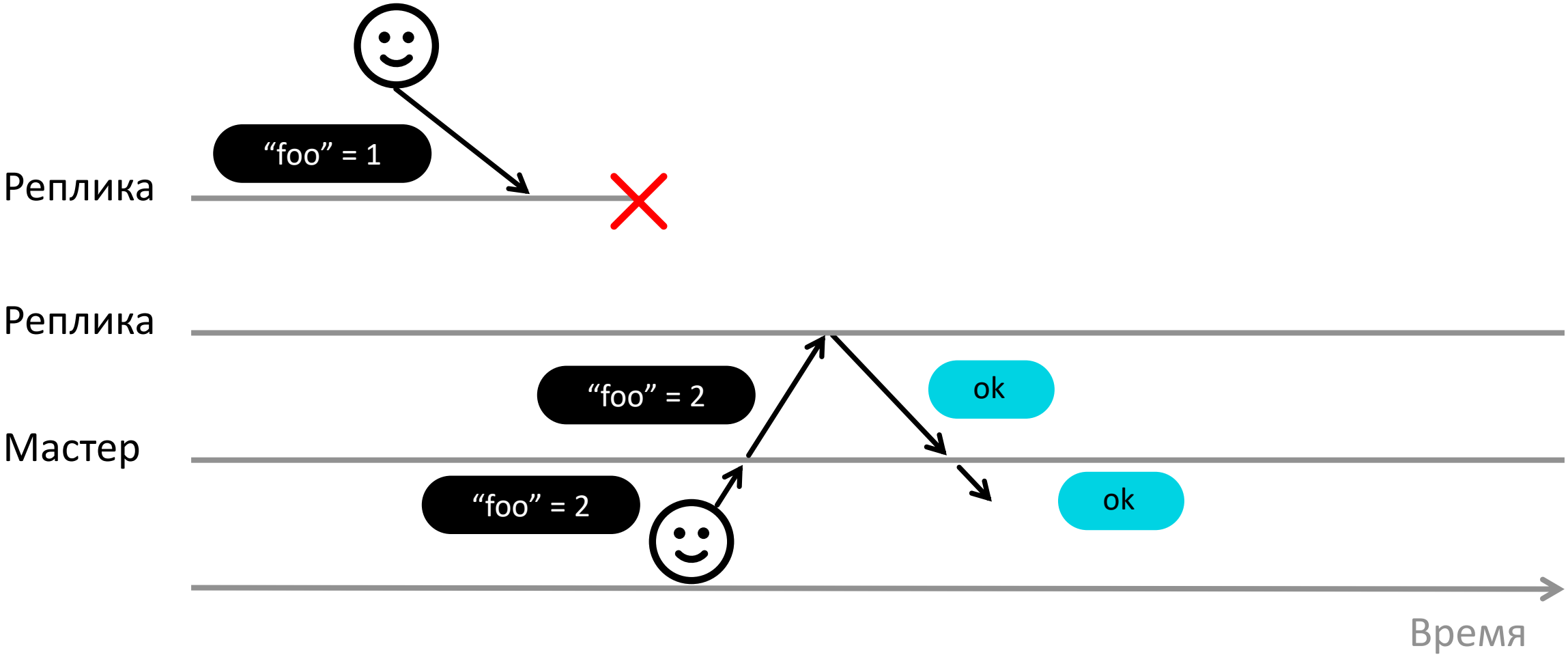
Нет конфликтов



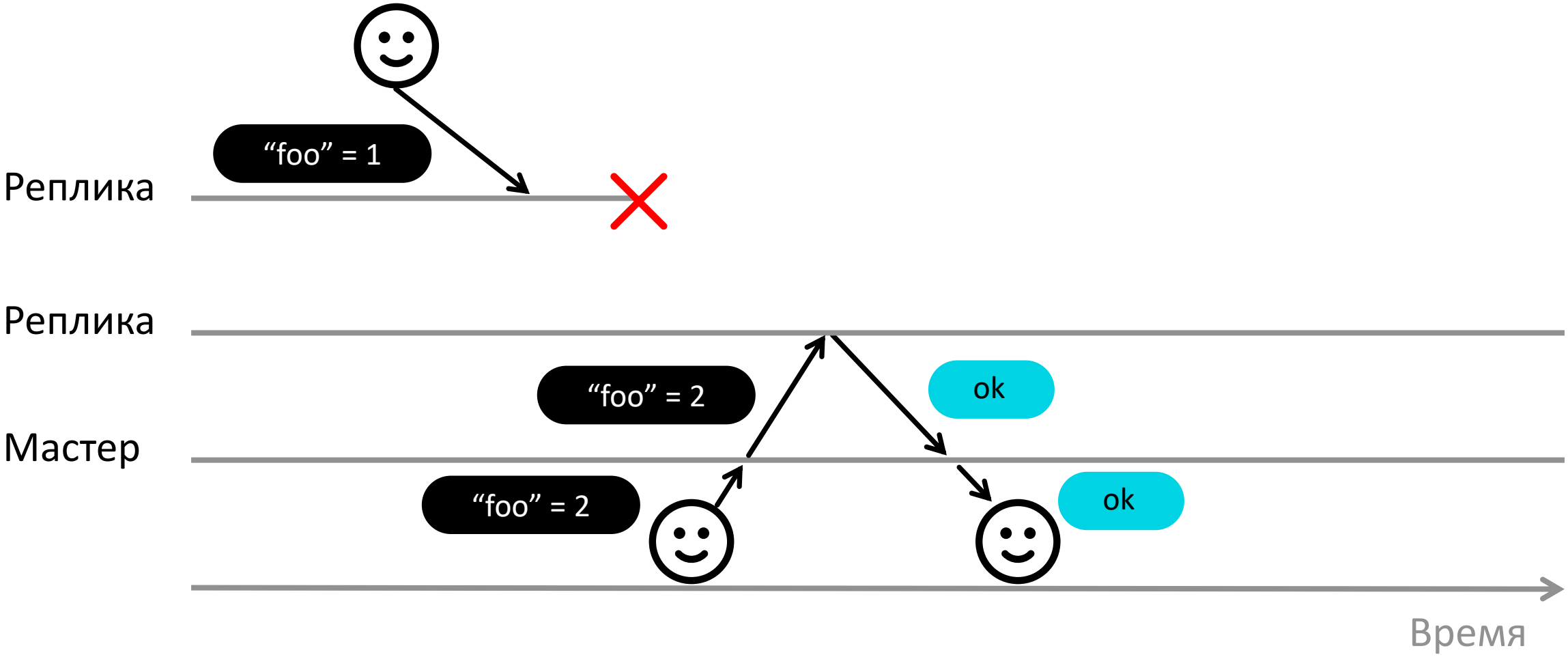
Нет конфликтов



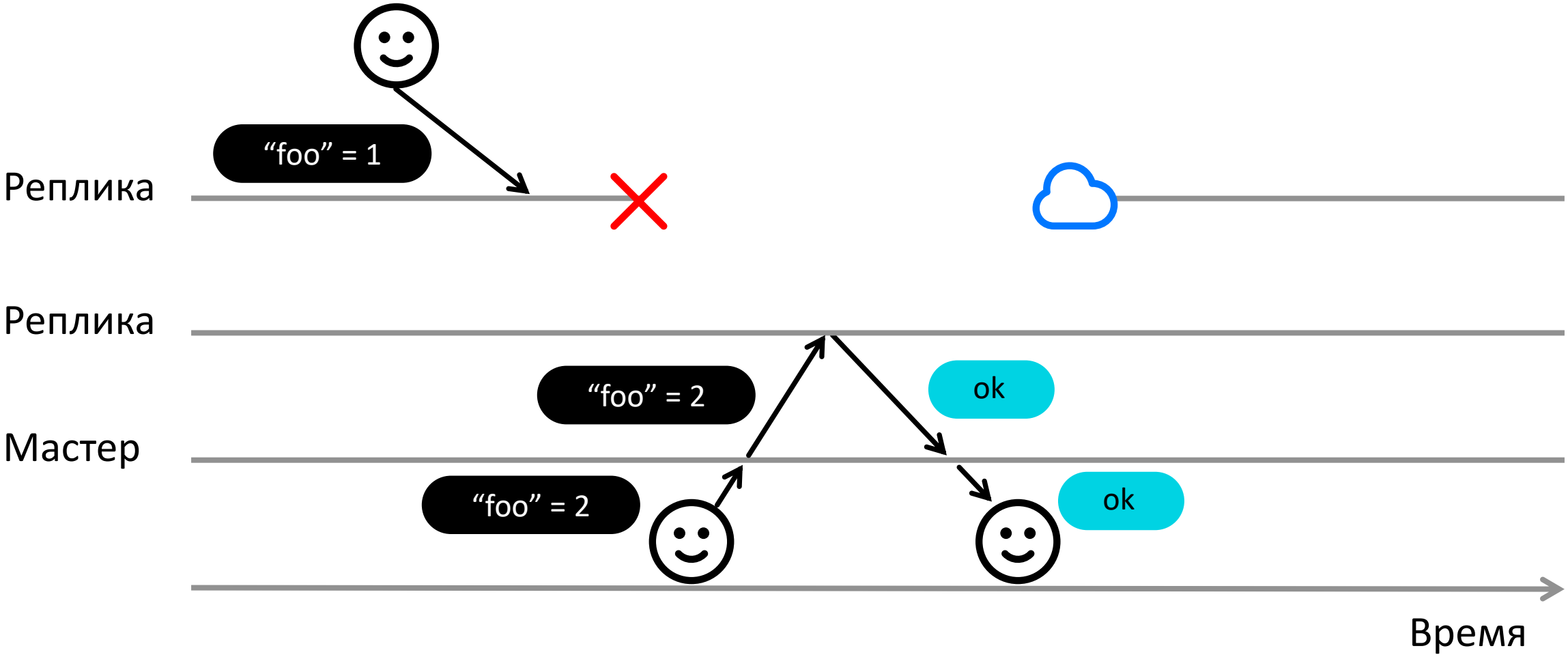
Нет конфликтов



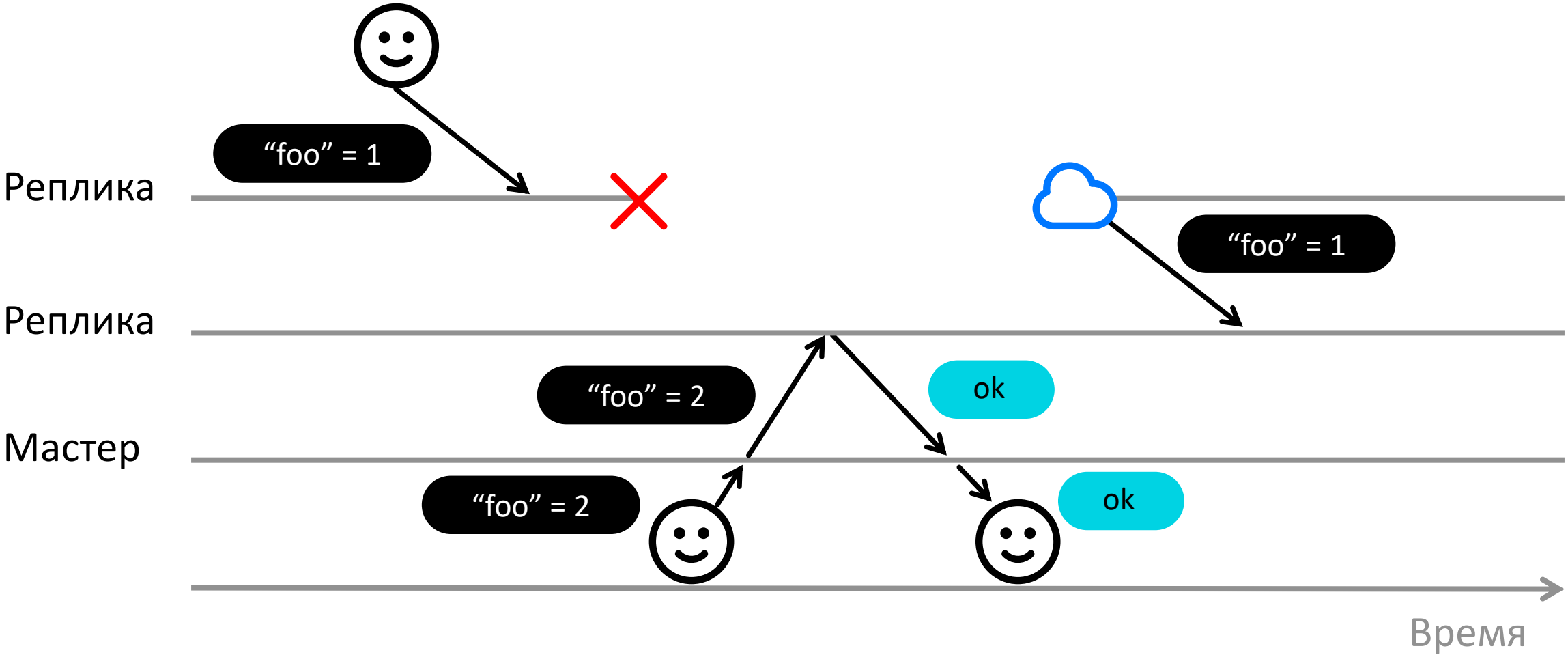
Нет конфликтов



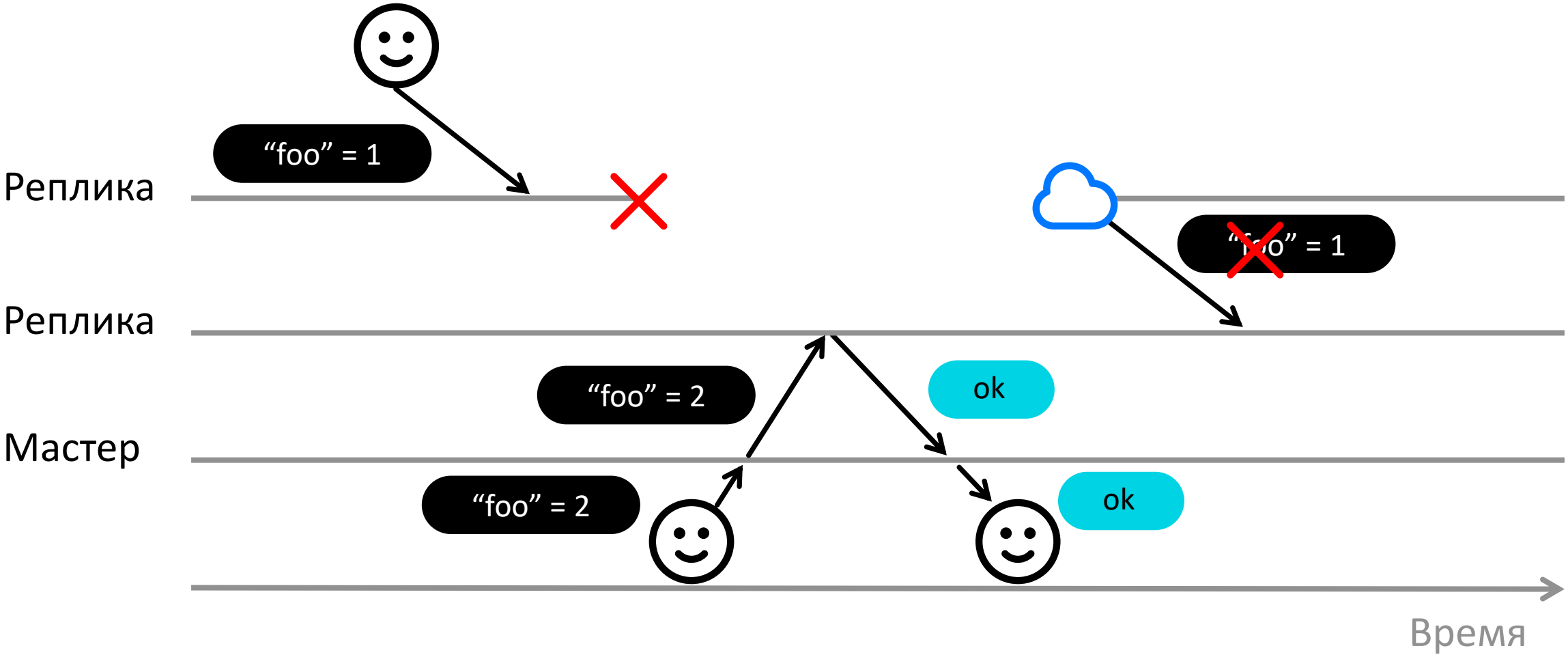
Нет конфликтов



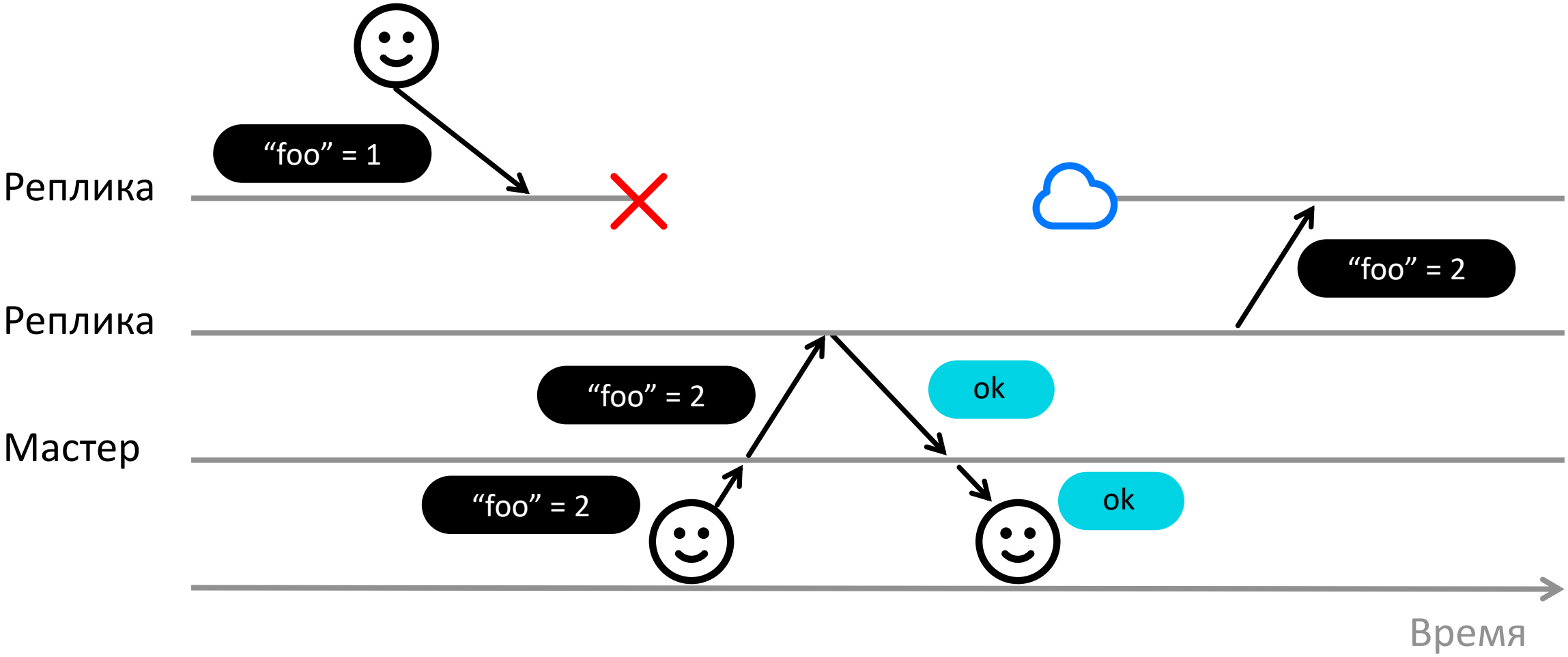
Нет конфликтов



Нет конфликтов

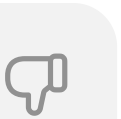


Нет конфликтов

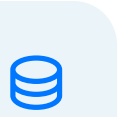


Последствия синхронной репликации

~~Потеря данных~~

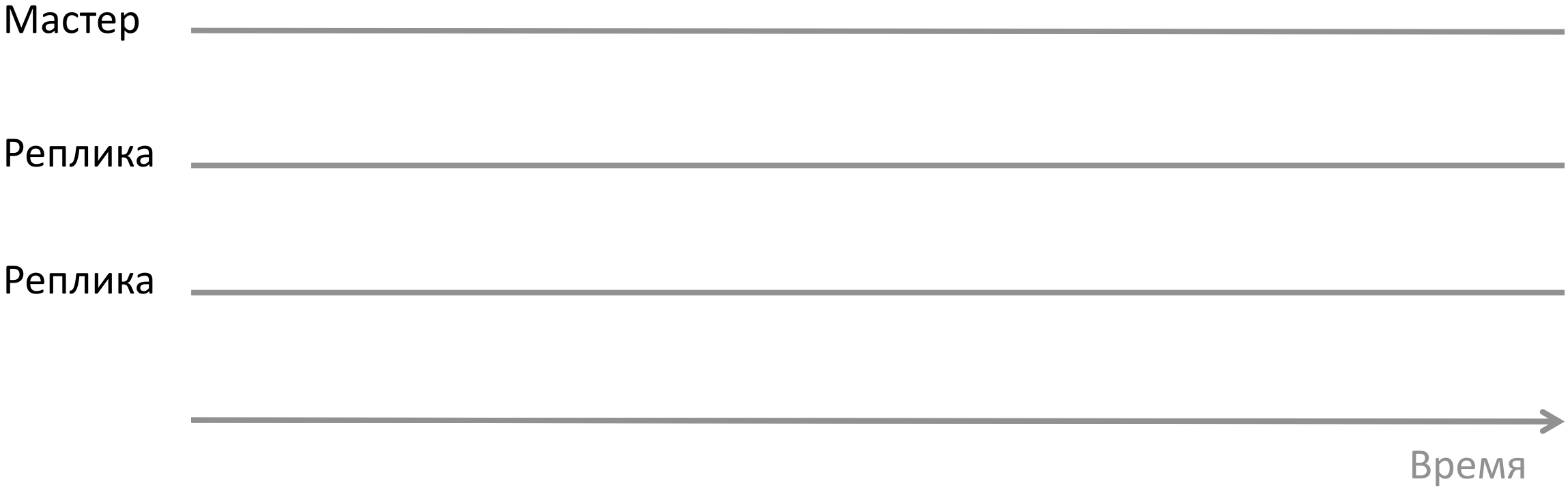


~~Конфликты~~

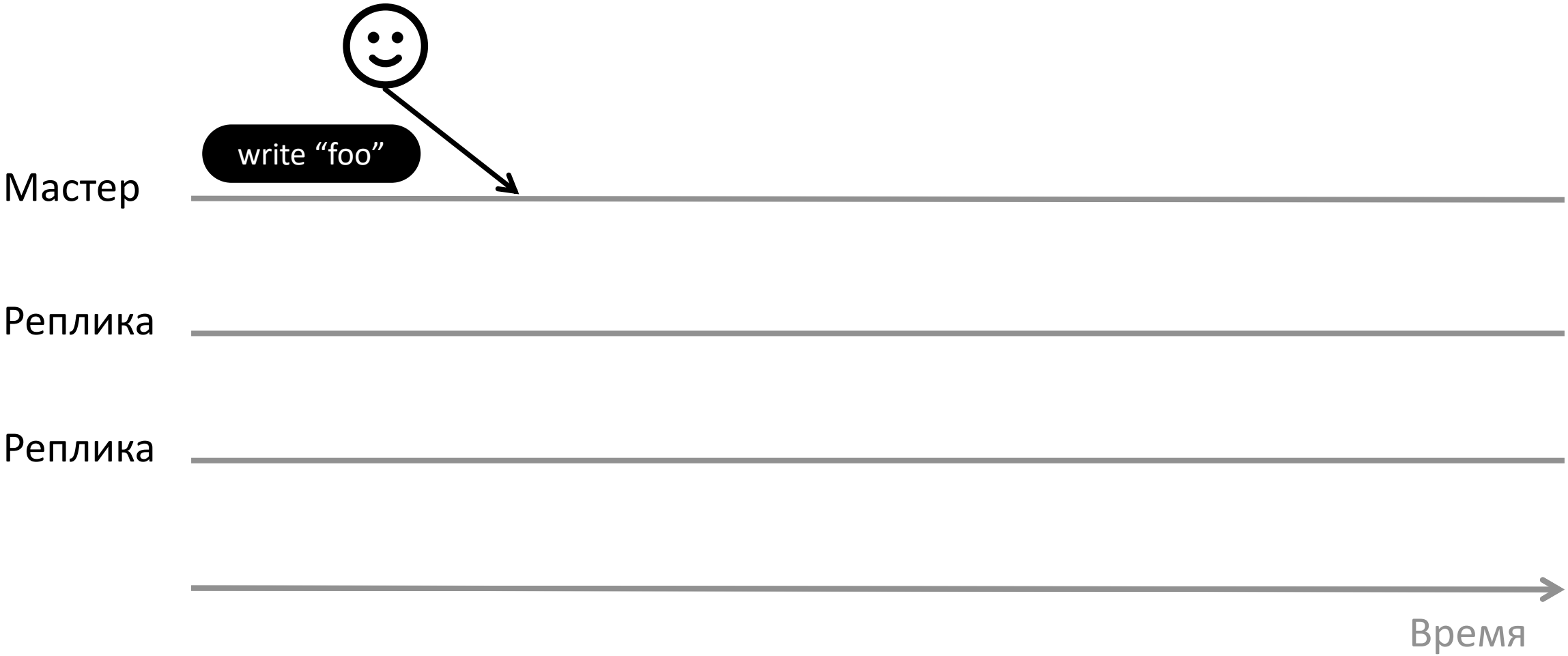


Чтение старых данных

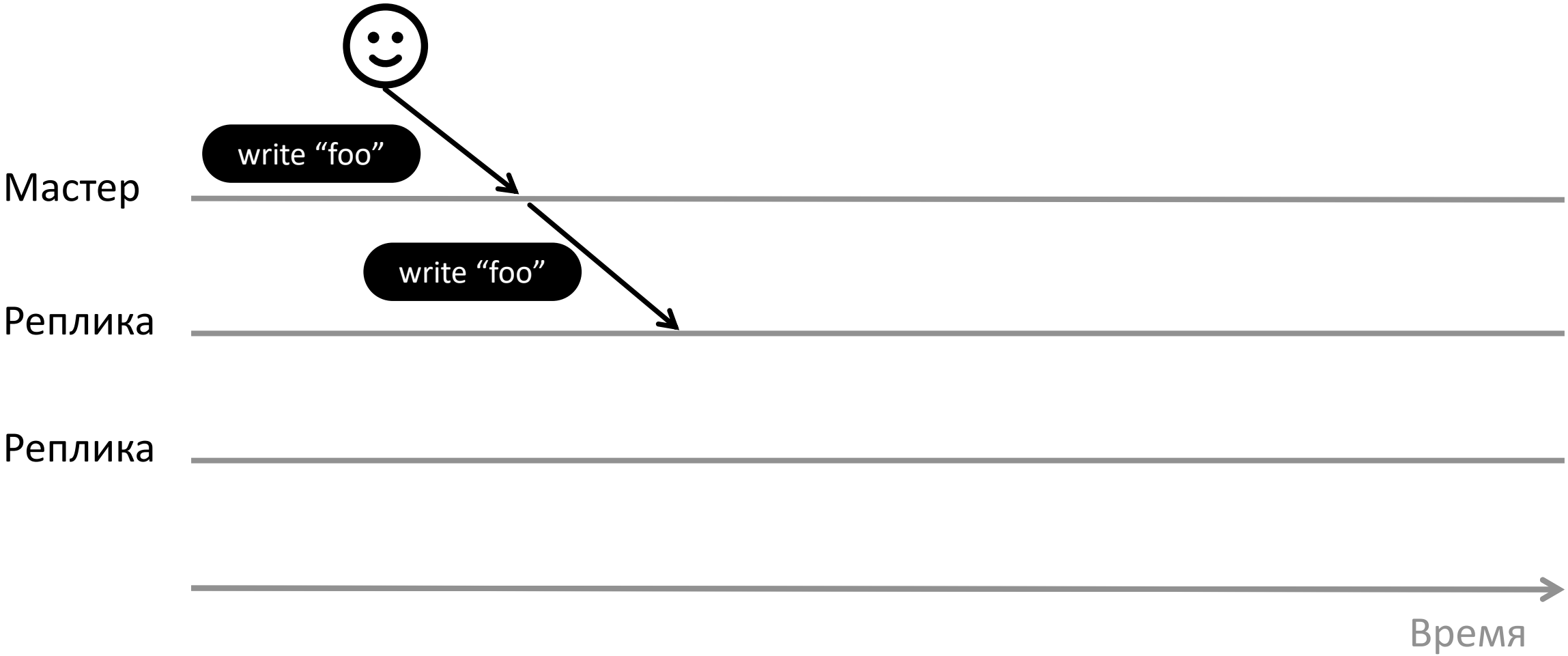
Чтение старых данных



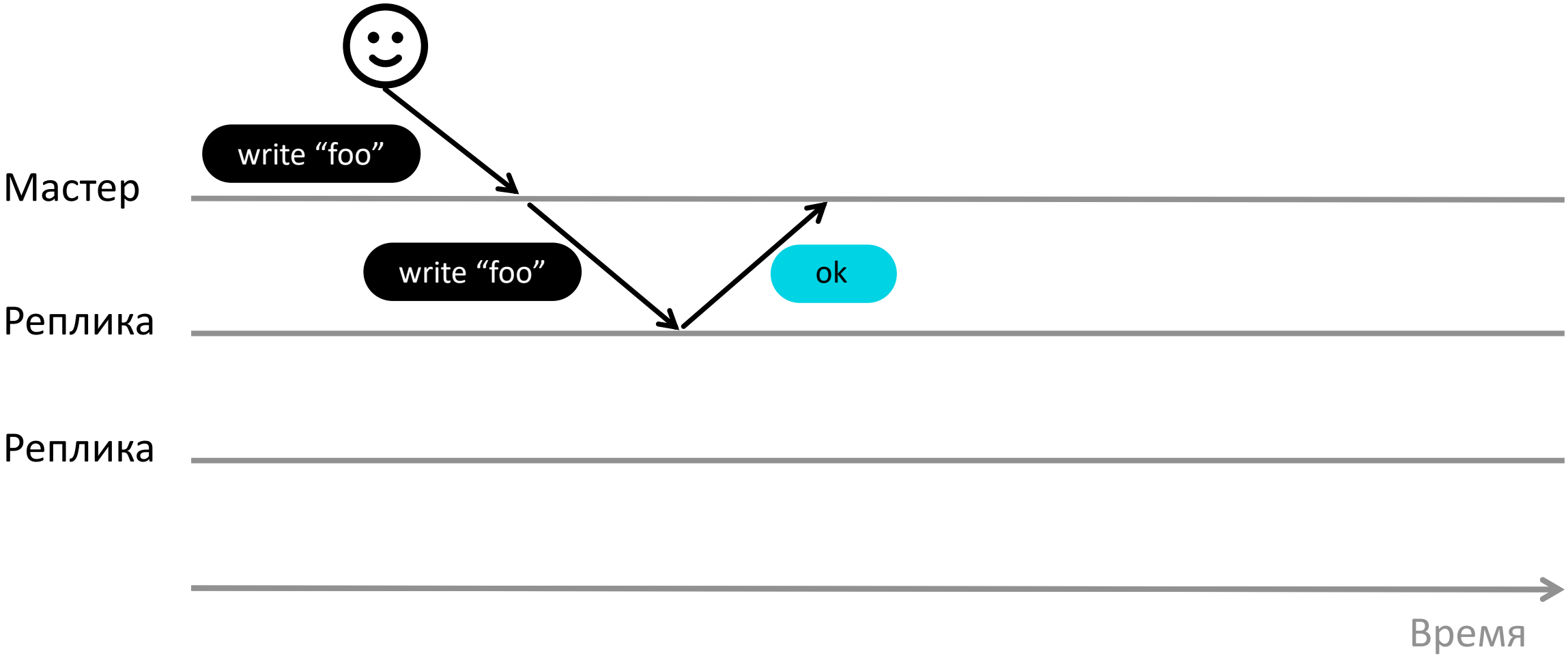
Чтение старых данных



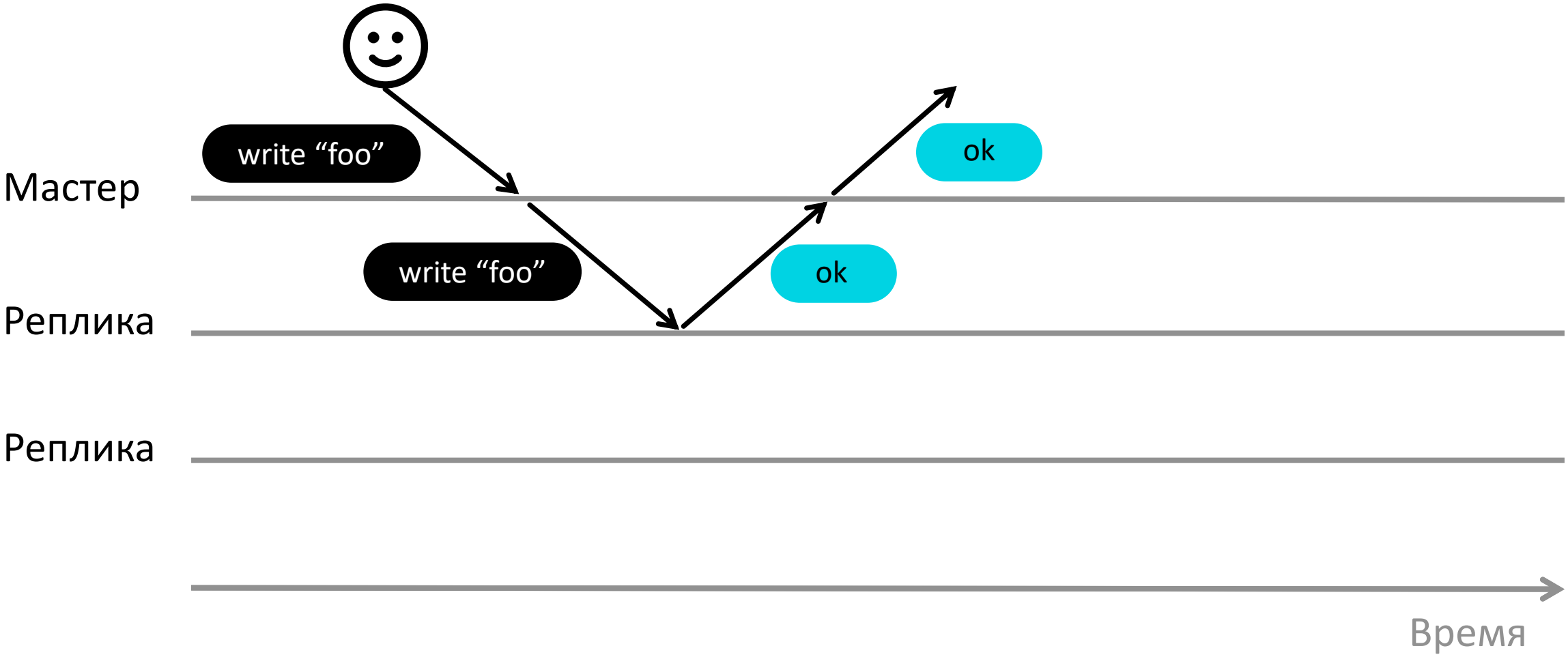
Чтение старых данных



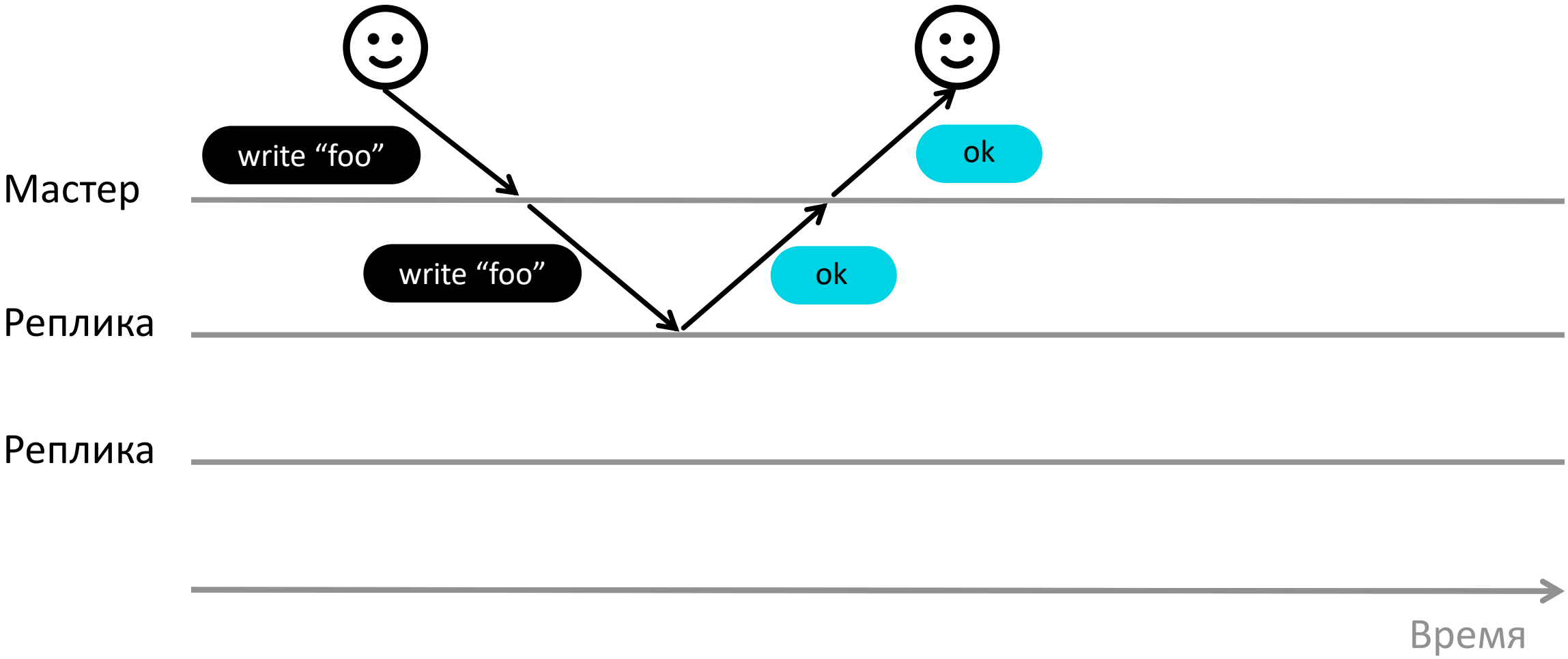
Чтение старых данных



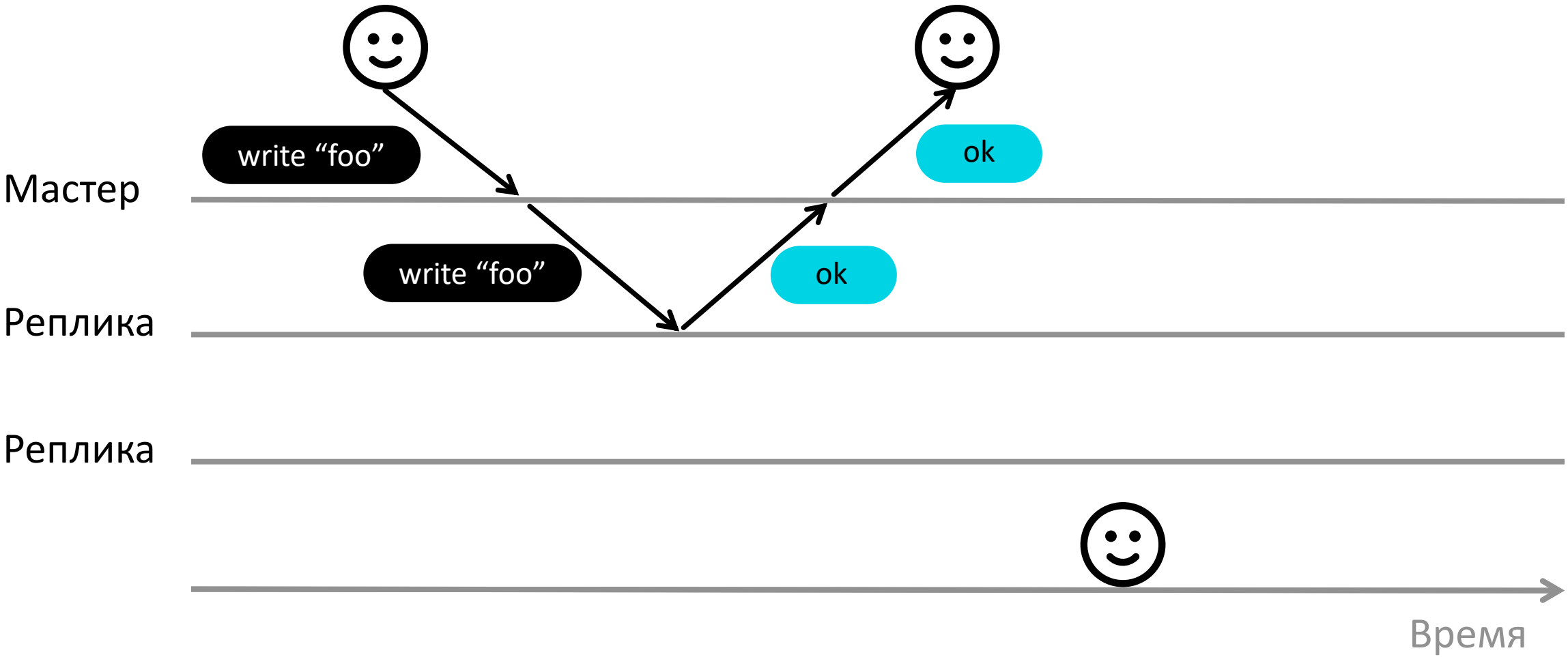
Чтение старых данных



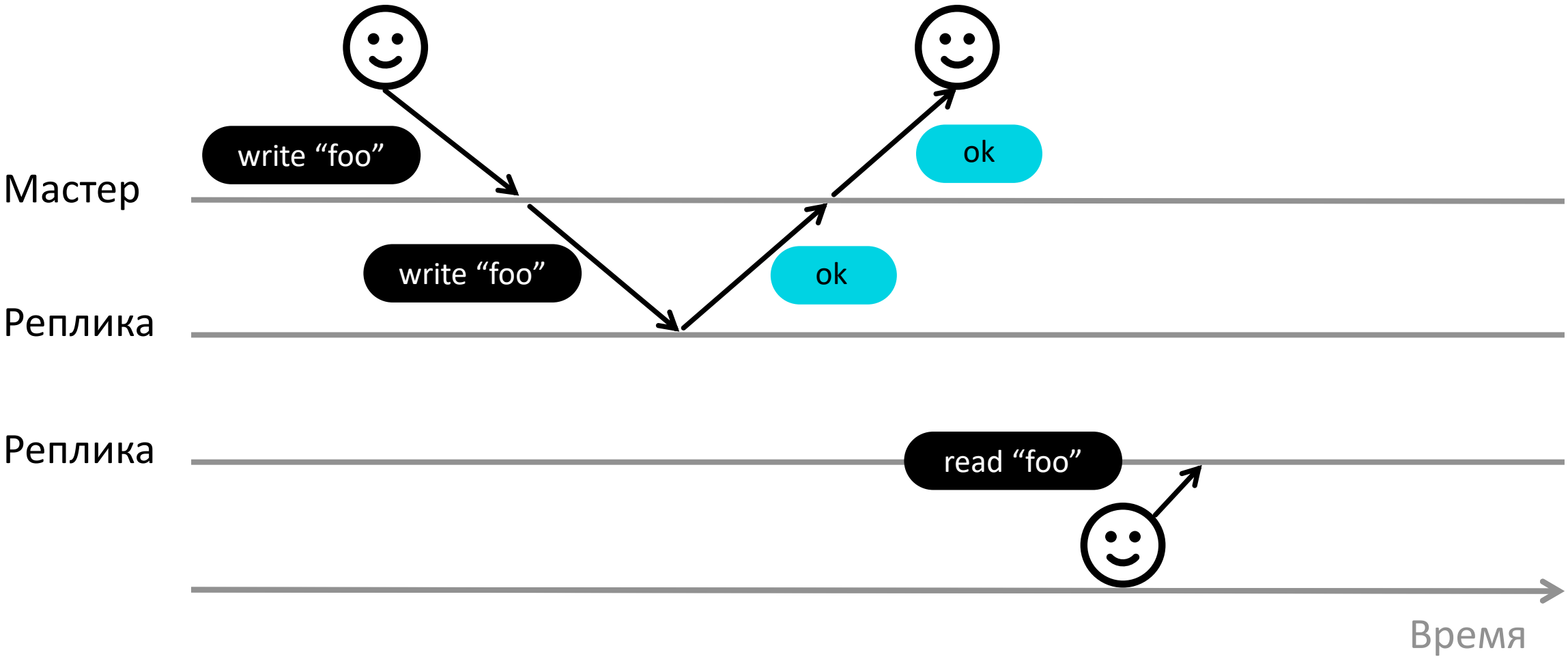
Чтение старых данных



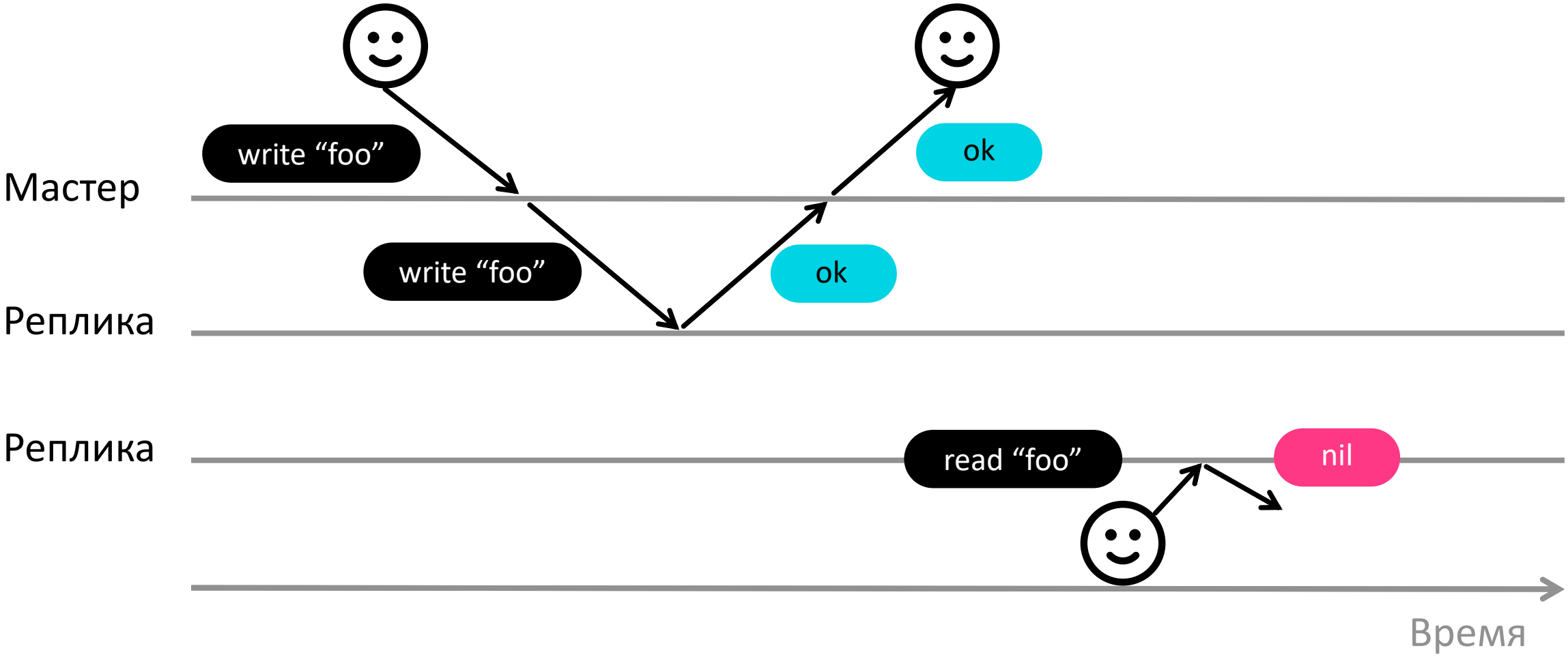
Чтение старых данных



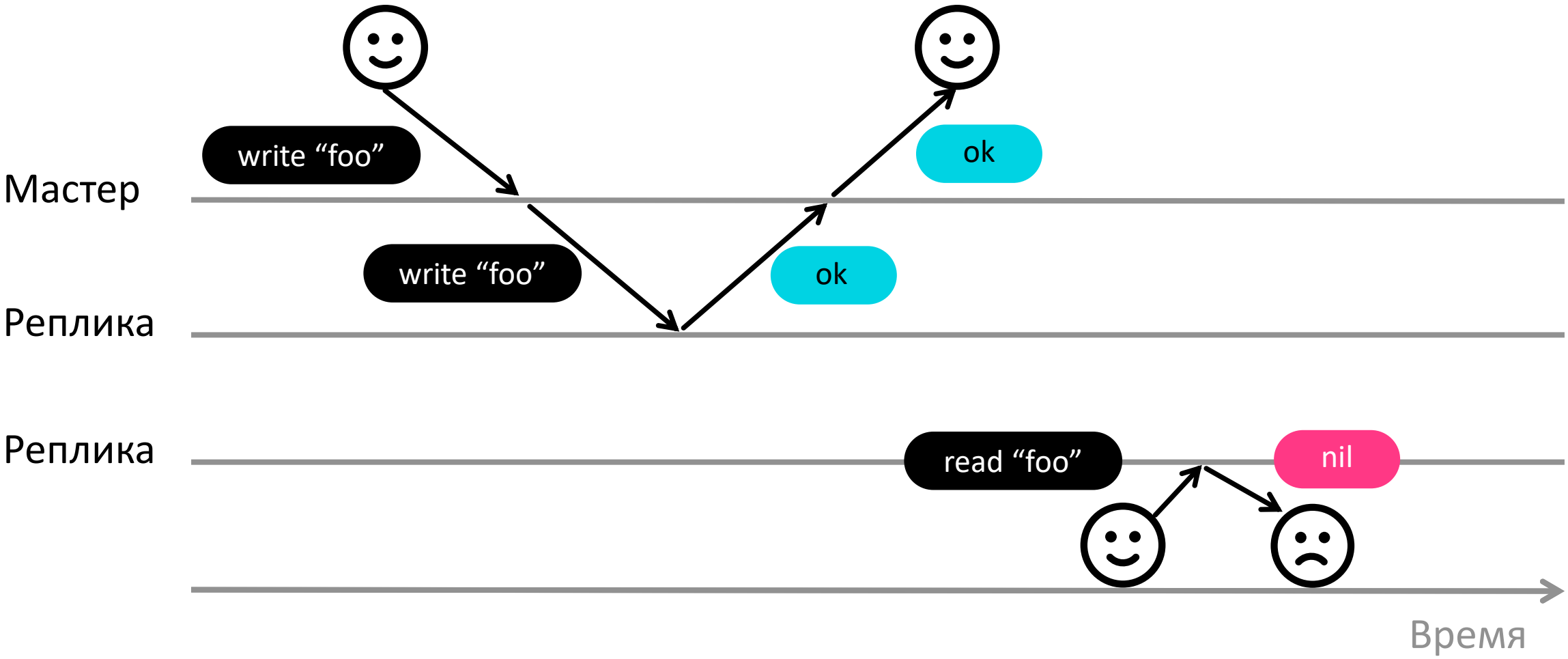
Чтение старых данных



Чтение старых данных



Чтение старых данных



Промежуточные выводы

- Цена масштабирования – новые аномалии
- Хотим, чтобы система выглядела, как единое целое

Линеаризуемость

Чего хотим?

Рассматриваем один объект и операции чтения / записи в него

Чего хотим?

Рассматриваем один объект и операции чтения / записи в него

Любая операция выполняется атомарно в какой-то момент времени между получением системой запроса и отправкой ответа клиенту

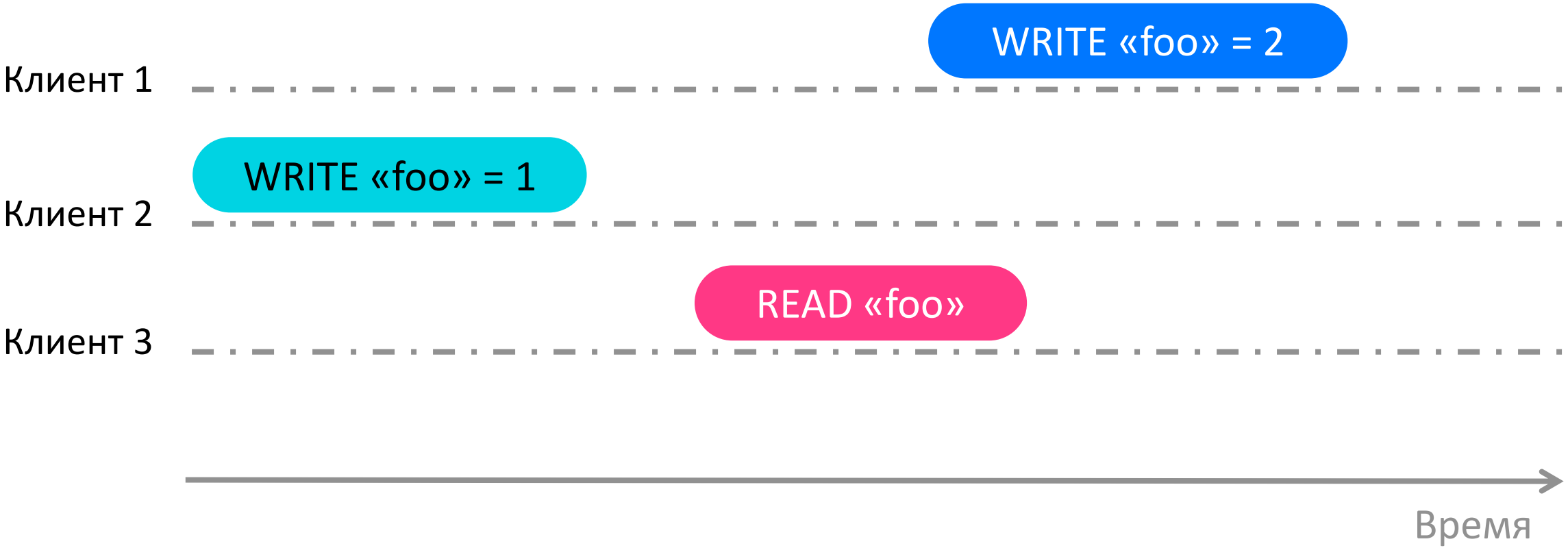
Чего хотим?

Рассматриваем один объект и операции чтения / записи в него

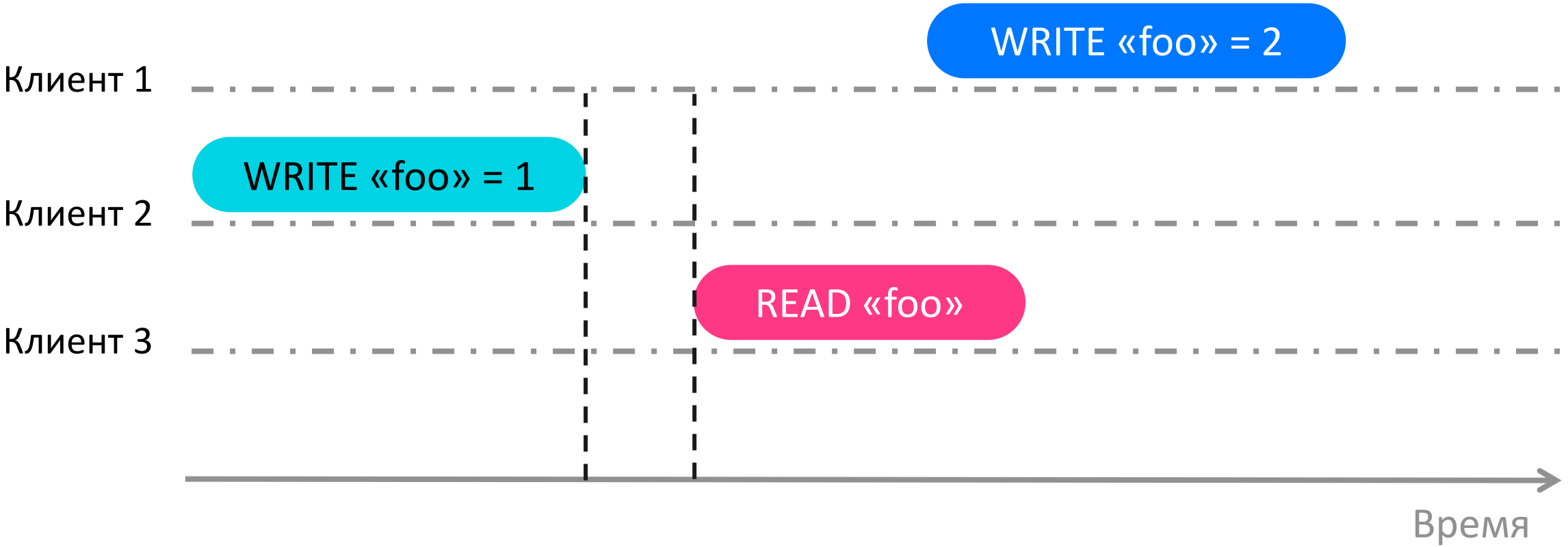
Любая операция выполняется атомарно в какой-то момент времени между получением системой запроса и отправкой ответа клиенту

Любое чтение возвращает результат всех выполненных на момент его прихода записей

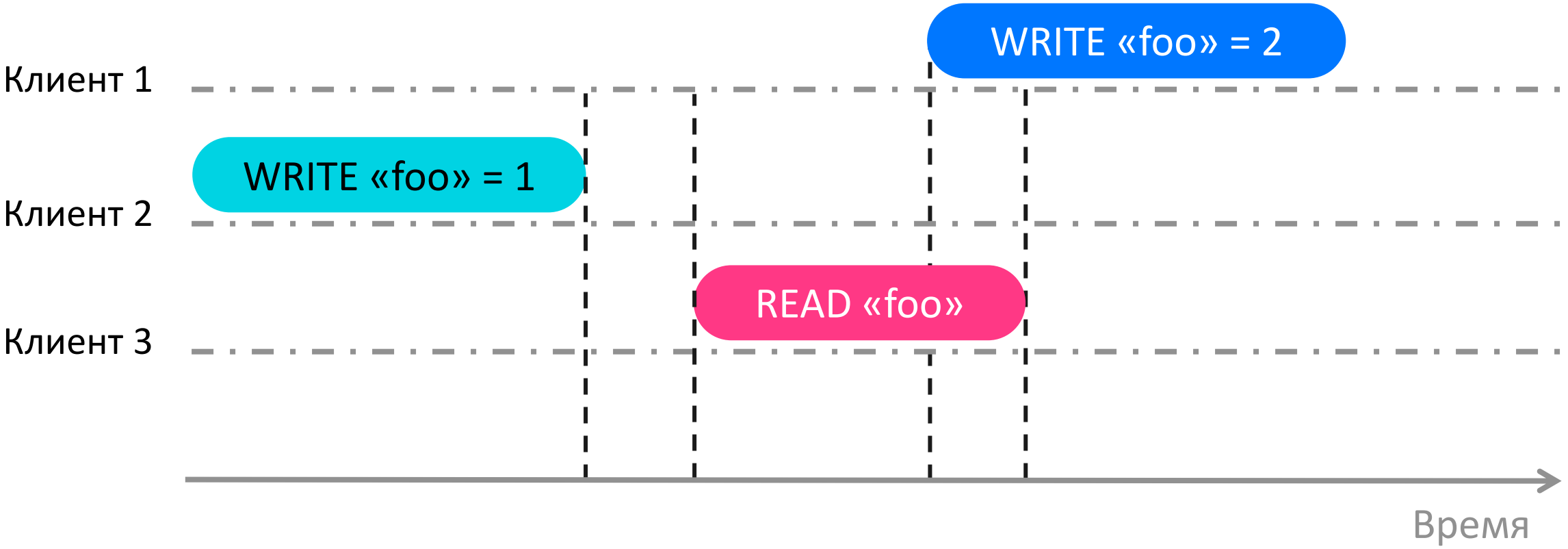
Пример


WRITE «foo» = 2
Клиент 1
WRITE «foo» = 1
Клиент 2
READ «foo»
Клиент 3
Время

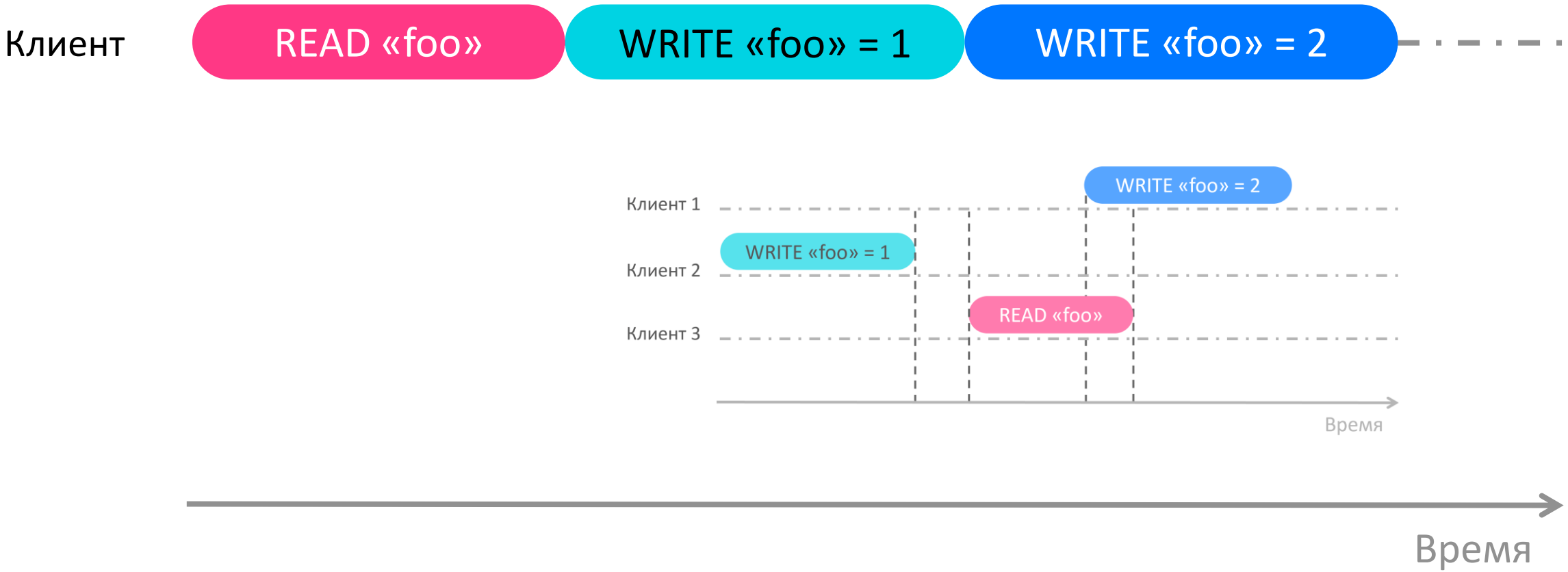
Пример


WRITE «foo» = 2
Клиент 1
WRITE «foo» = 1
Клиент 2
READ «foo»
Клиент 3
Время

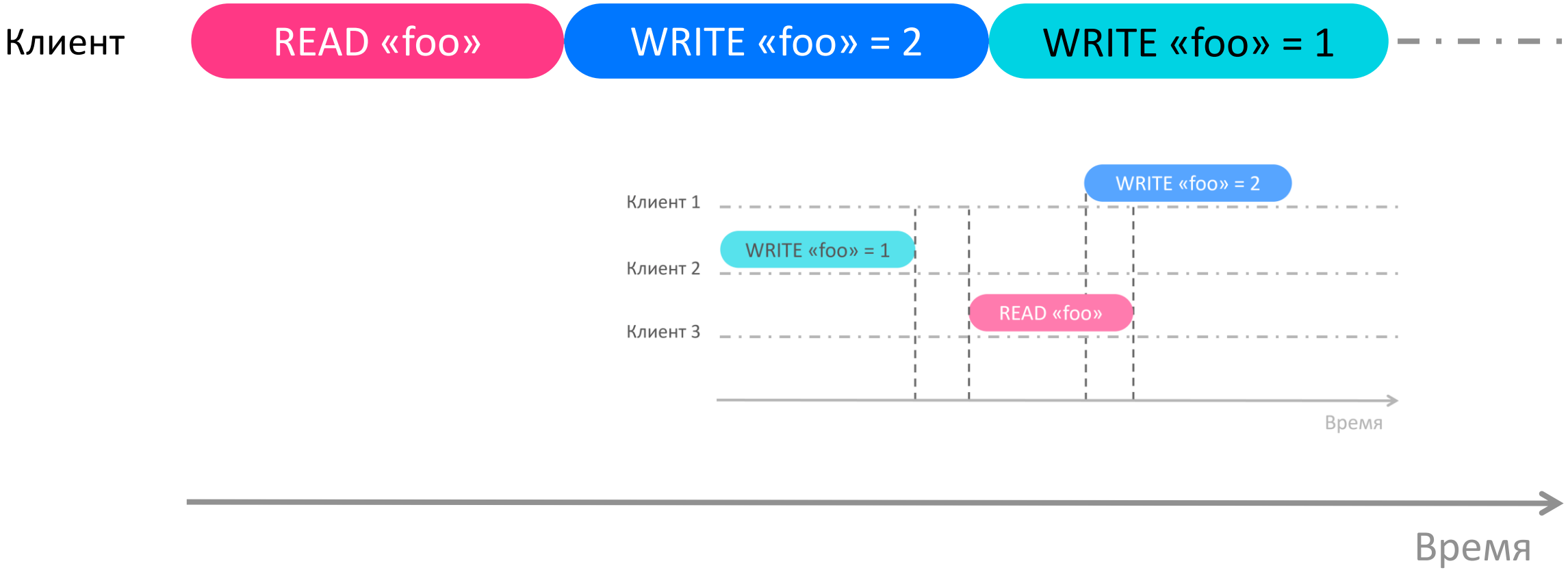
Пример


WRITE «foo» = 2
Клиент 1
WRITE «foo» = 1
Клиент 2
READ «foo»
Клиент 3
Время

Нелинеаризуемая история 1

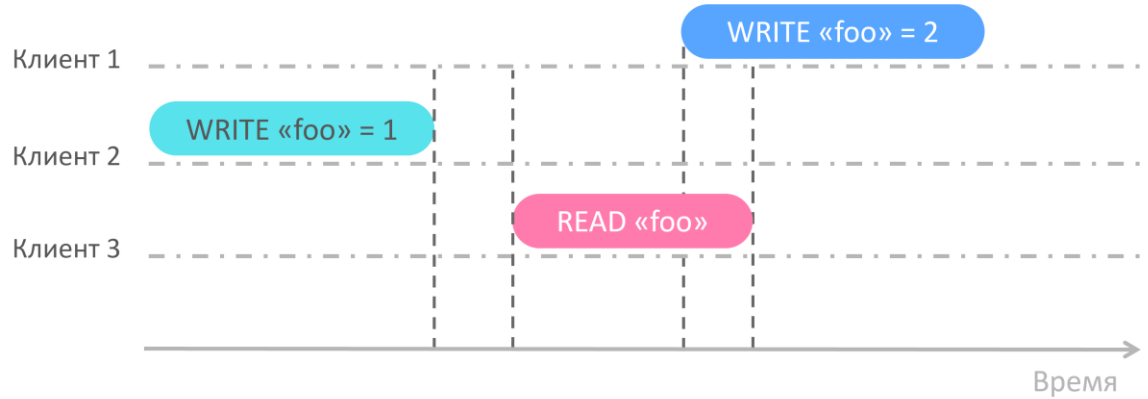


Нелинеаризуемая история 2



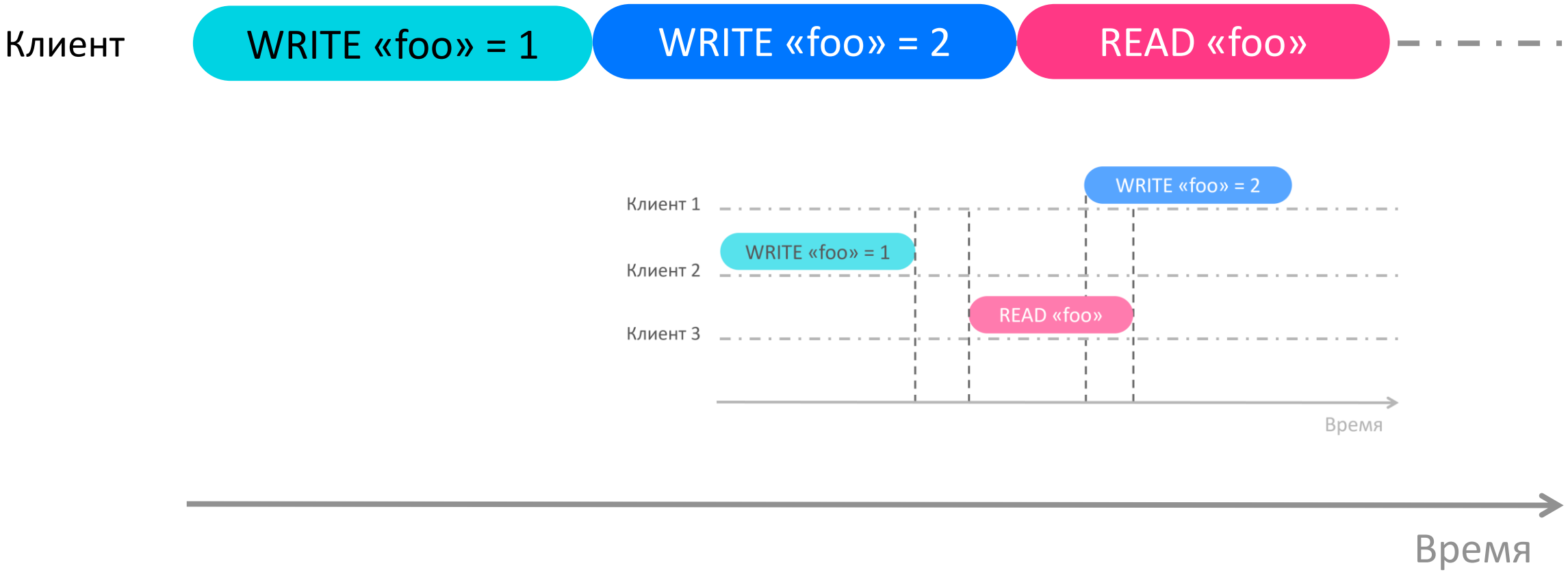
Линеаризуемая история

Клиент



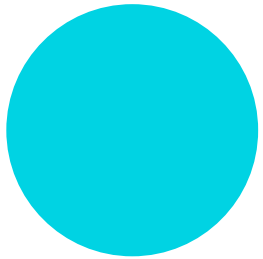
Время

Линеаризуемая история 2



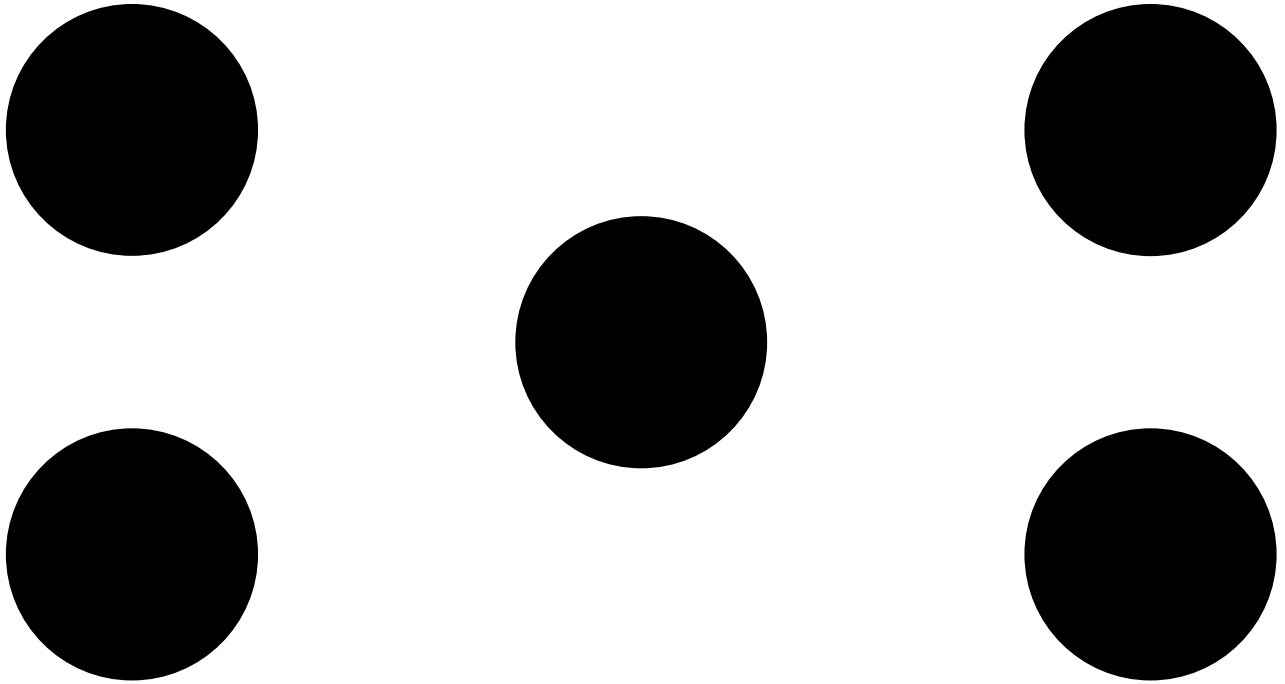
Чтение и запись по кворуму

- N = 1
- W = 1
- R = 1



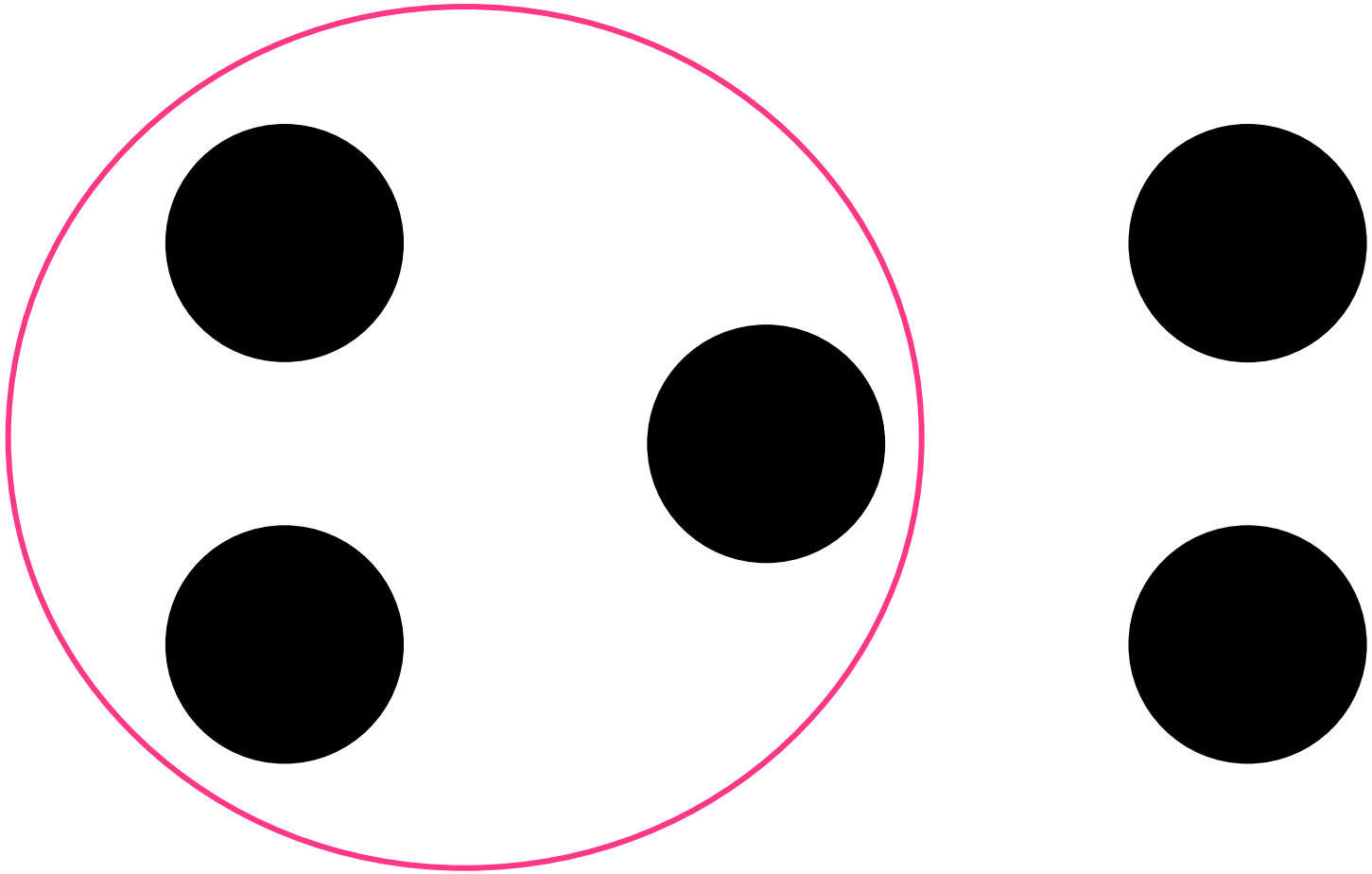
Чтение и запись по кворуму

- N = 5



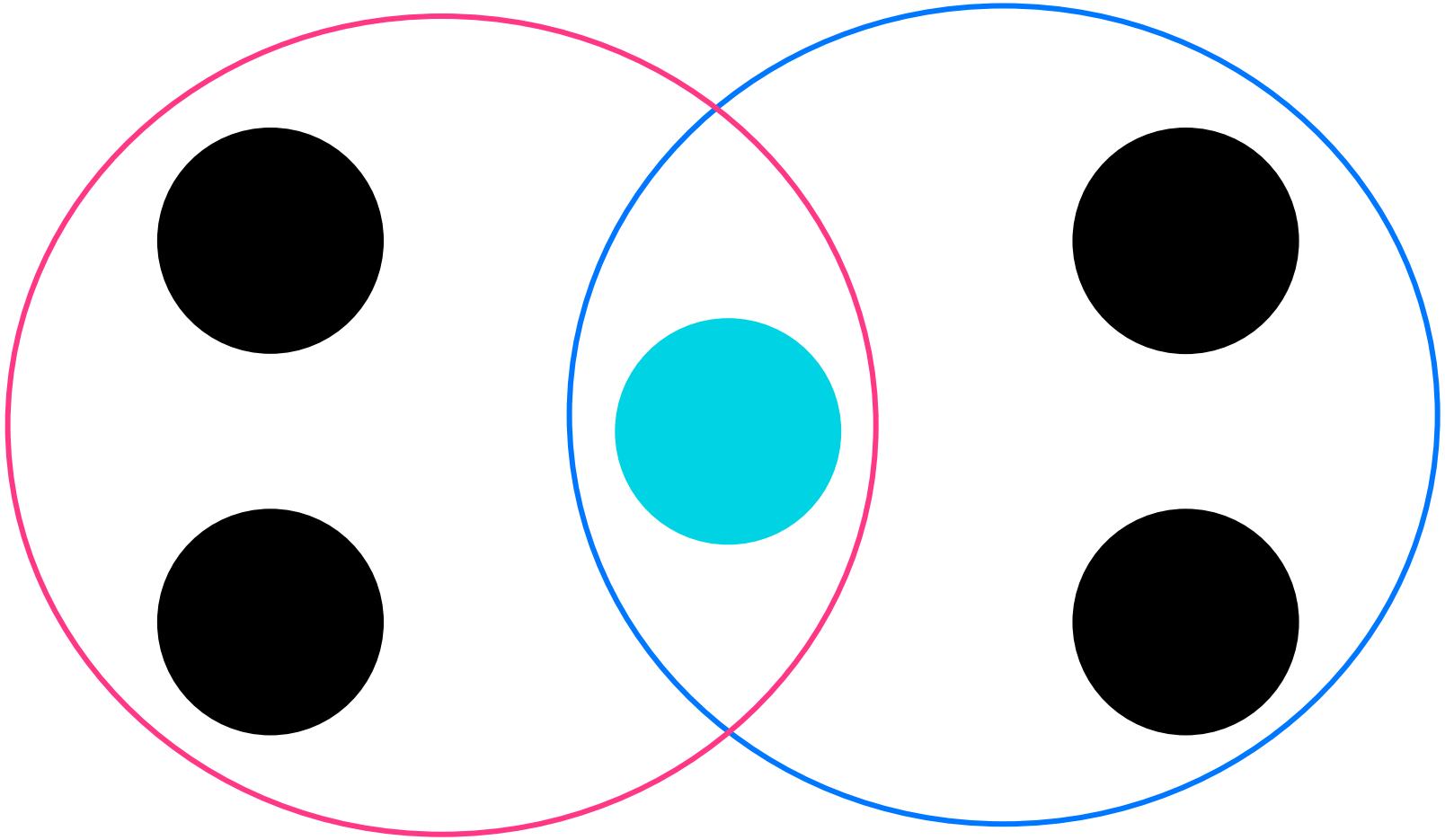
Чтение и запись по кворуму

- N = 5
- W = 3



Чтение и запись по кворуму

- N = 5
- W = 3
- R = 3



Чтение и запись по кворуму

1 N узлов в кластере

Чтение и запись по кворуму

1. N узлов в кластере
2. Кворум на чтение R

Чтение и запись по кворуму

1. N узлов в кластере
2. Кворум на чтение R
3. Кворум на запись W

Чтение и запись по кворуму

1. N узлов в кластере
2. Кворум на чтение R
3. Кворум на запись W
4. Линеаризуемость => R + W > N

Чтение и запись по кворуму

1. N узлов в кластере
2. Кворум на чтение R
3. Кворум на запись W
4. Линеаризуемость => R + W > N
5. Пользователь общается с кворумом напрямую либо через один из узлов в кластере

Raft

1. Единственный лидер

Raft

1. Единственный лидер
2. Лидера выбирает большинство (N / 2 + 1)

Raft

1. Единственный лидер
2. Лидера выбирает большинство (N / 2 + 1)
3. W = N / 2 + 1

Raft

1. Единственный лидер
2. Лидера выбирает большинство (N / 2 + 1)
3. W = N / 2 + 1
4. Лидер гарантировано имеет все подтверждённые записи

Raft

1. Единственный лидер
2. Лидера выбирает большинство (N / 2 + 1)
3. W = N / 2 + 1
4. Лидер гарантировано имеет все подтверждённые записи
5. Журнал совпадает на всех узлах – это необходимо, но недостаточно для линеаризуемости

Линеаризуемость в Raft. Идея

1. Выполнять чтения на лидере

Линеаризуемость в Raft. Идея

1. Выполнять чтения на лидере
2. Лидера могли сместить, а он и не заметил

Линеаризуемость в Raft. Идея

1. Выполнять чтения на лидере
2. Лидера могли сместить, а он и не заметил
3. Перед чтением надо проверить, что лидер всё ещё лидер

Линеаризуемость в Raft. Реализация

1. Запомнить текущий commit index

Линеаризуемость в Raft. Реализация

1. Запомнить текущий commit index

2. Выпустить heartbeat ко всем и дождаться ответа от большинства

Линеаризуемость в Raft. Реализация

1. Запомнить текущий commit index
2. Выпустить heartbeat ко всем и дождаться ответа от большинства
3. Уже не лидер? Вернуть ошибку

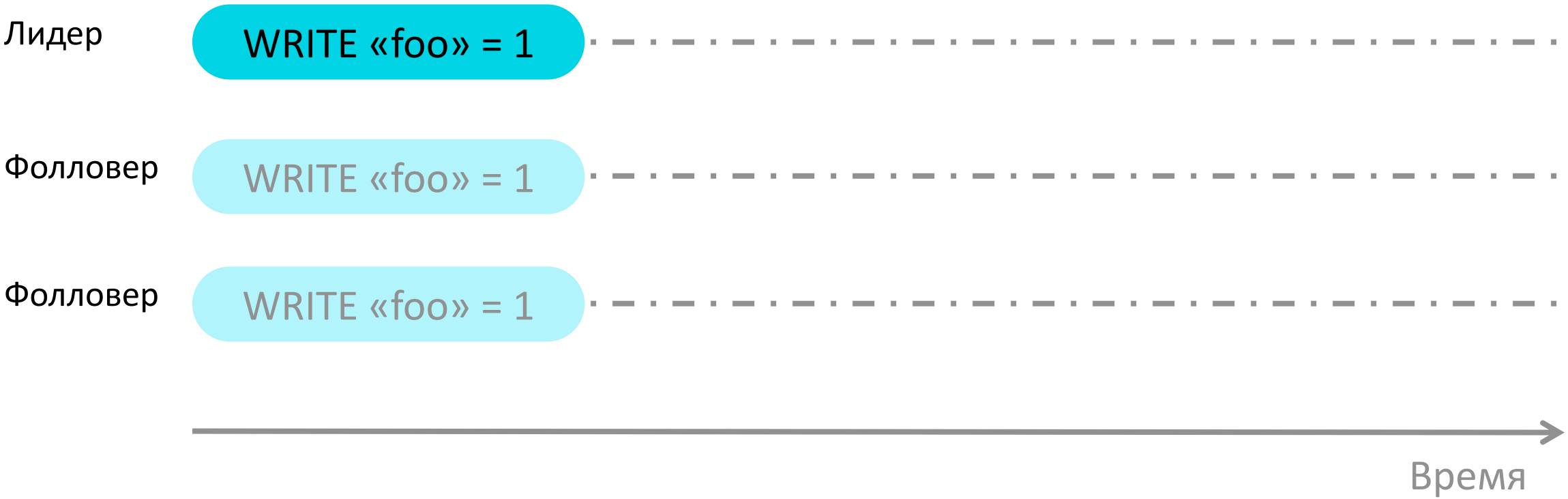
Линеаризуемость в Raft. Реализация

1. Запомнить текущий commit index
2. Выпустить heartbeat ко всем и дождаться ответа от большинства
3. Уже не лидер? Вернуть ошибку
4. Иначе дождаться применения записи, соответствующей commit index

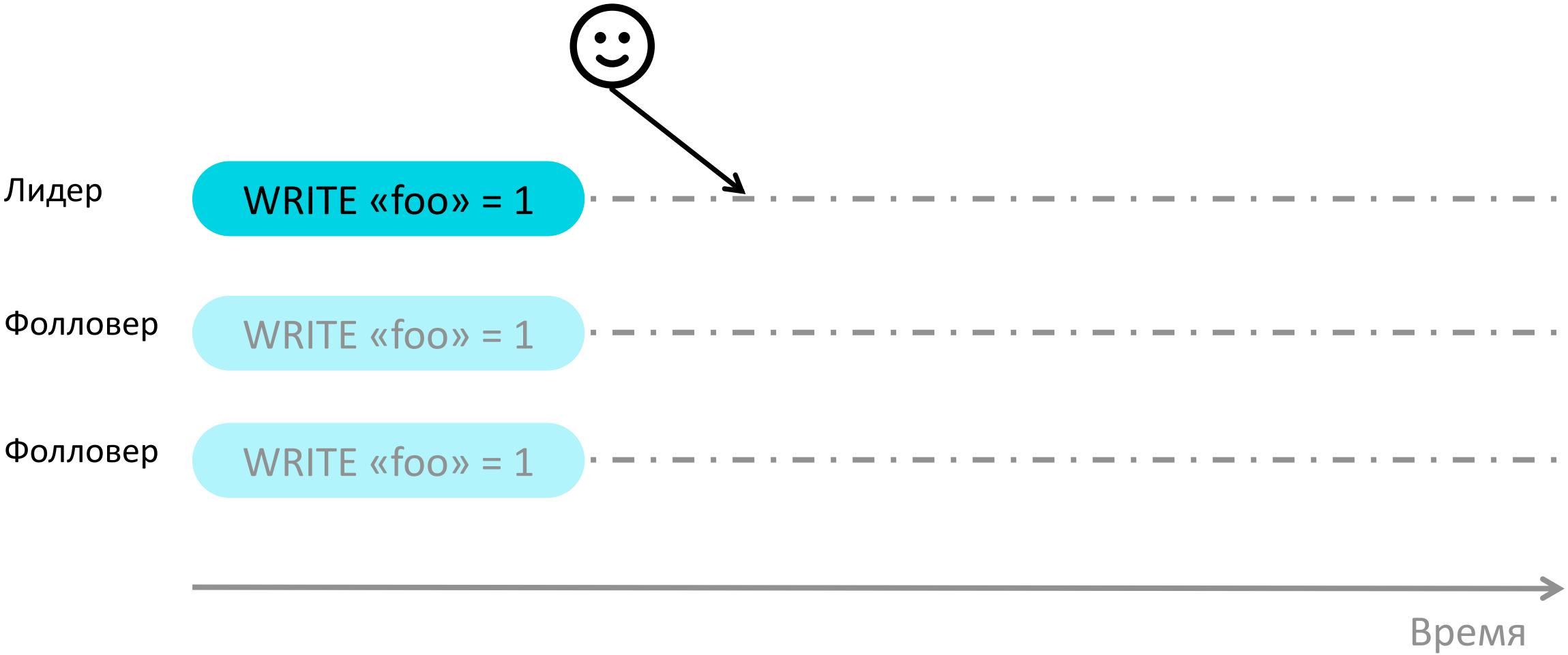
Линеаризуемость в Raft. Реализация

1. Запомнить текущий commit index
2. Выпустить heartbeat ко всем и дождаться ответа от большинства
3. Уже не лидер? Вернуть ошибку
4. Иначе дождаться применения записи, соответствующей commit index
5. Выполнить чтение

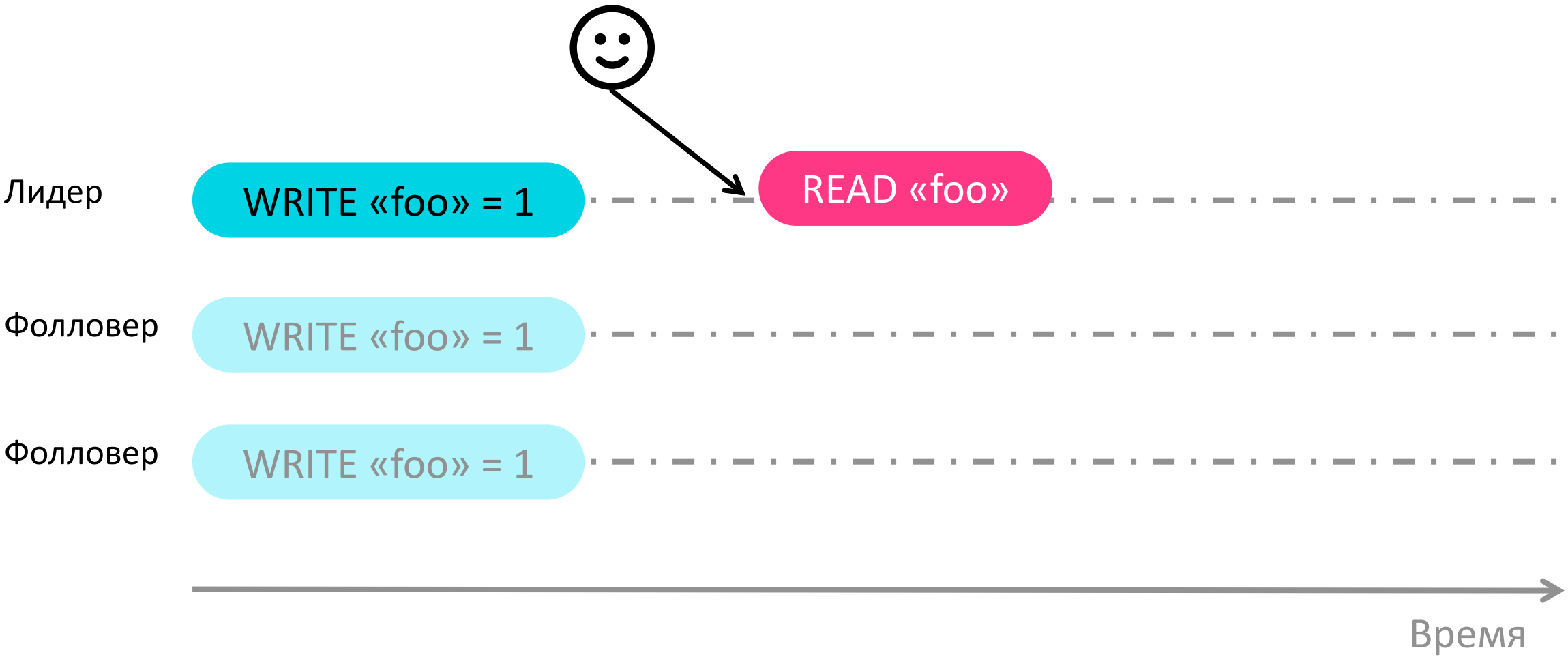
Линеаризуемость в Raft



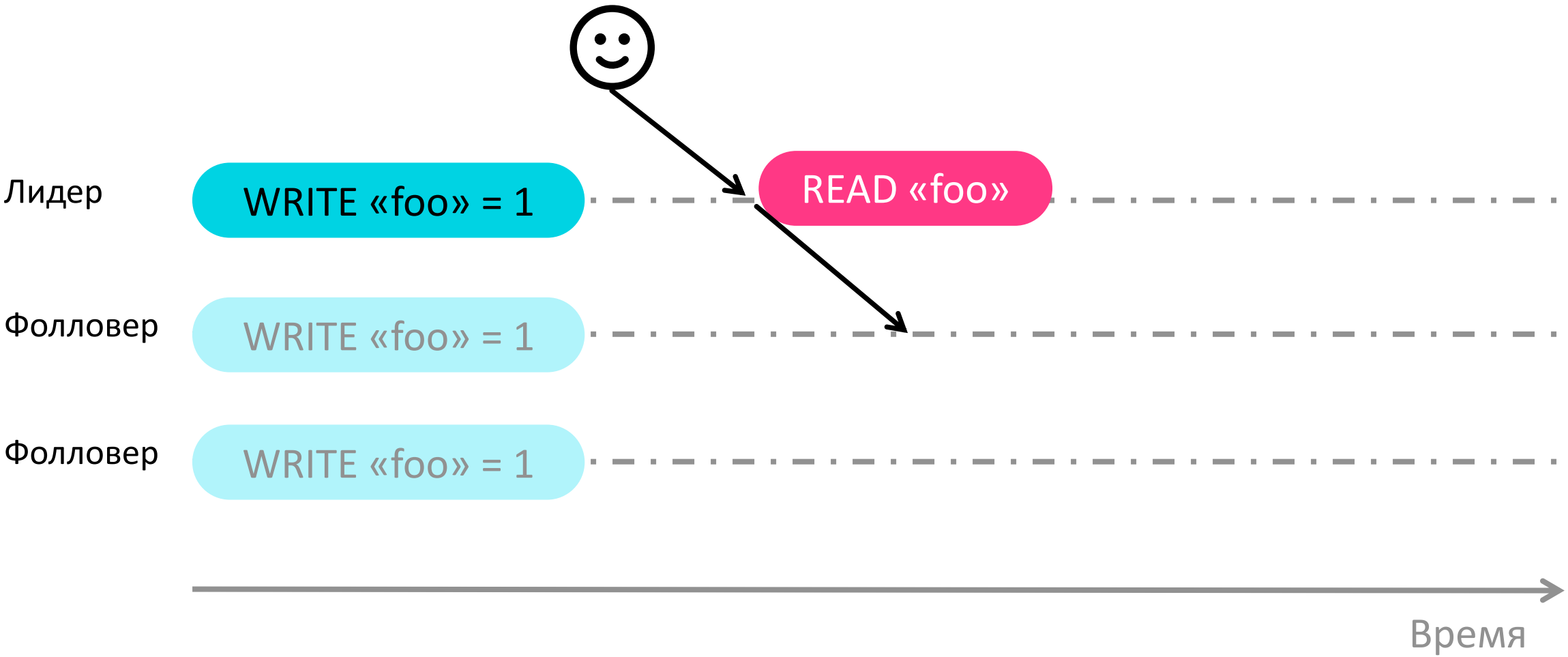
Линеаризуемость в Raft



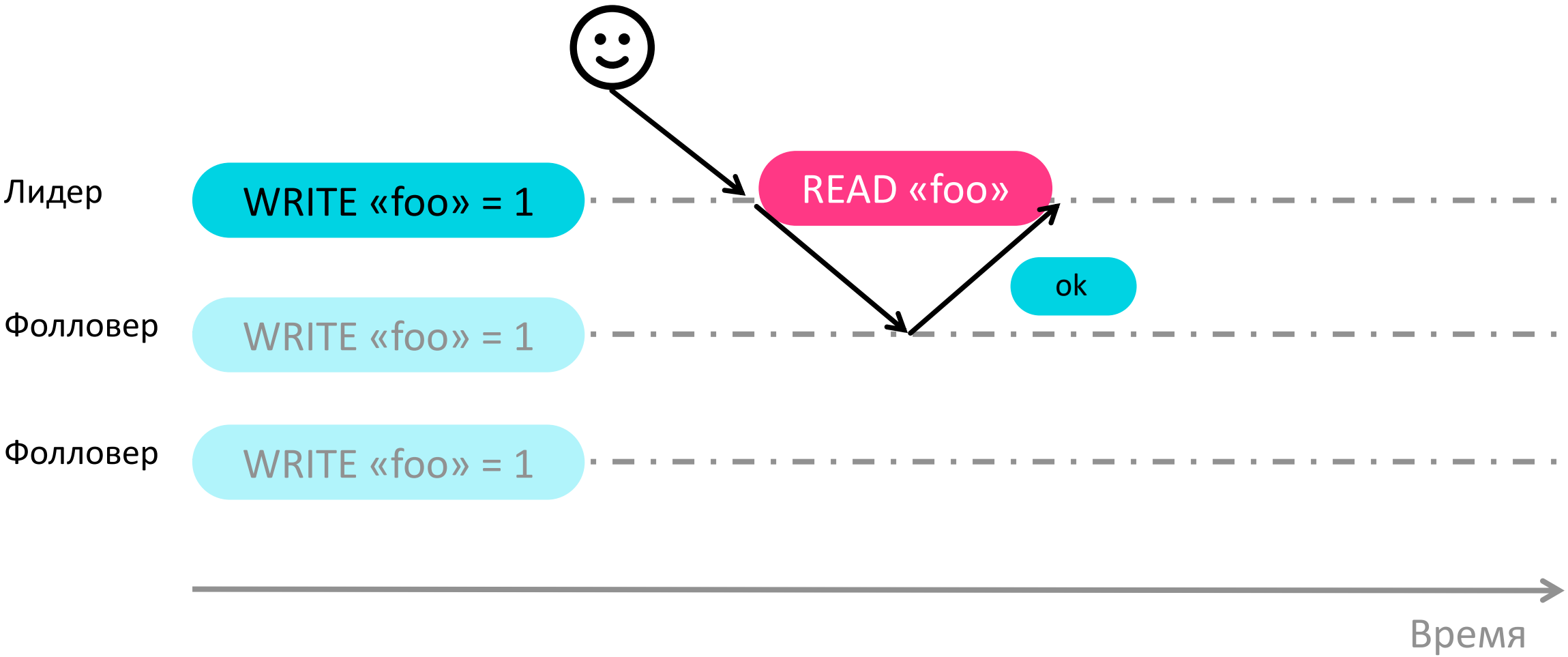
Линеаризуемость в Raft



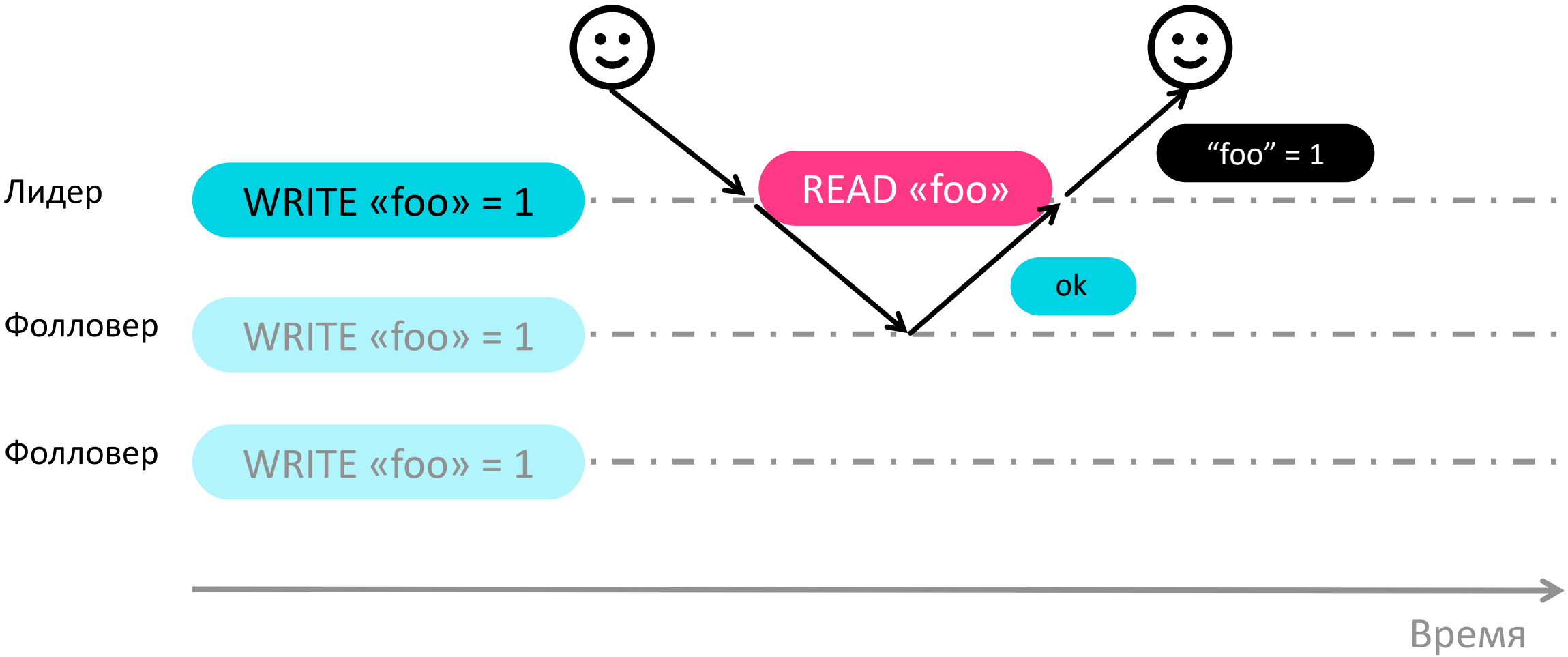
Линеаризуемость в Raft



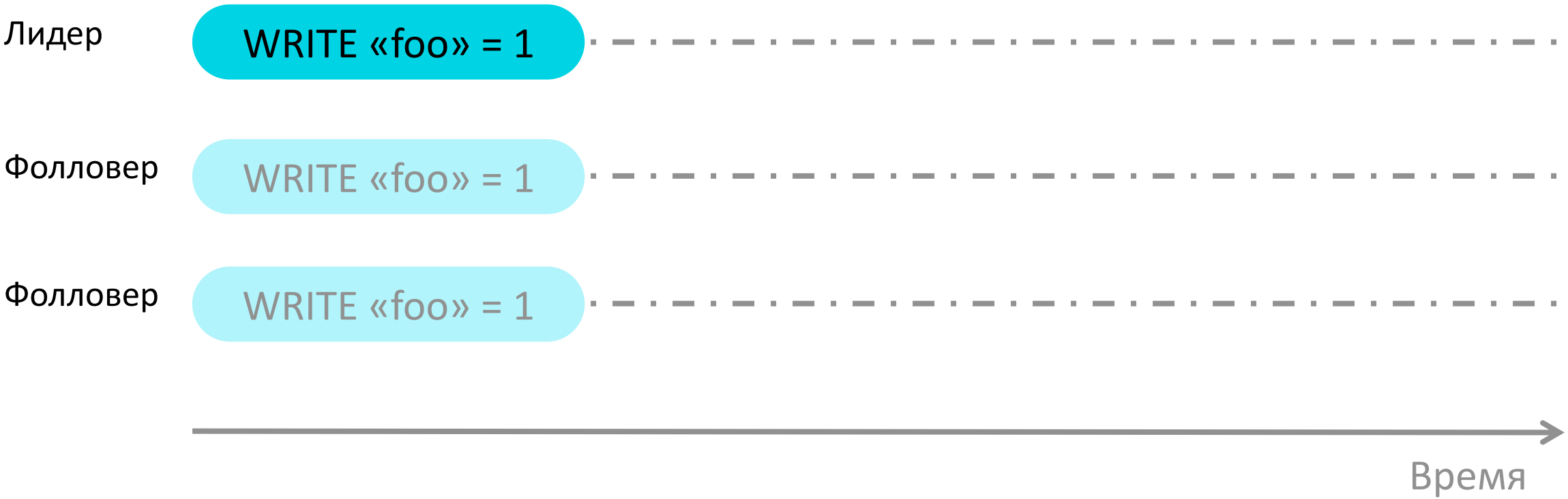
Линеаризуемость в Raft



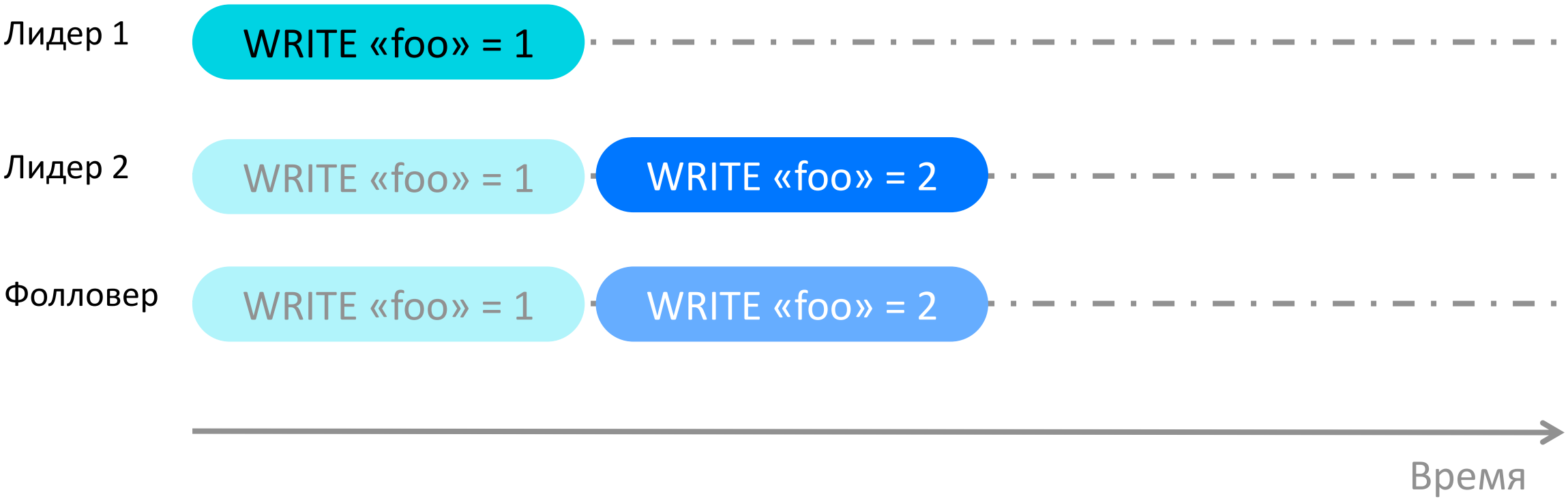
Линеаризуемость в Raft



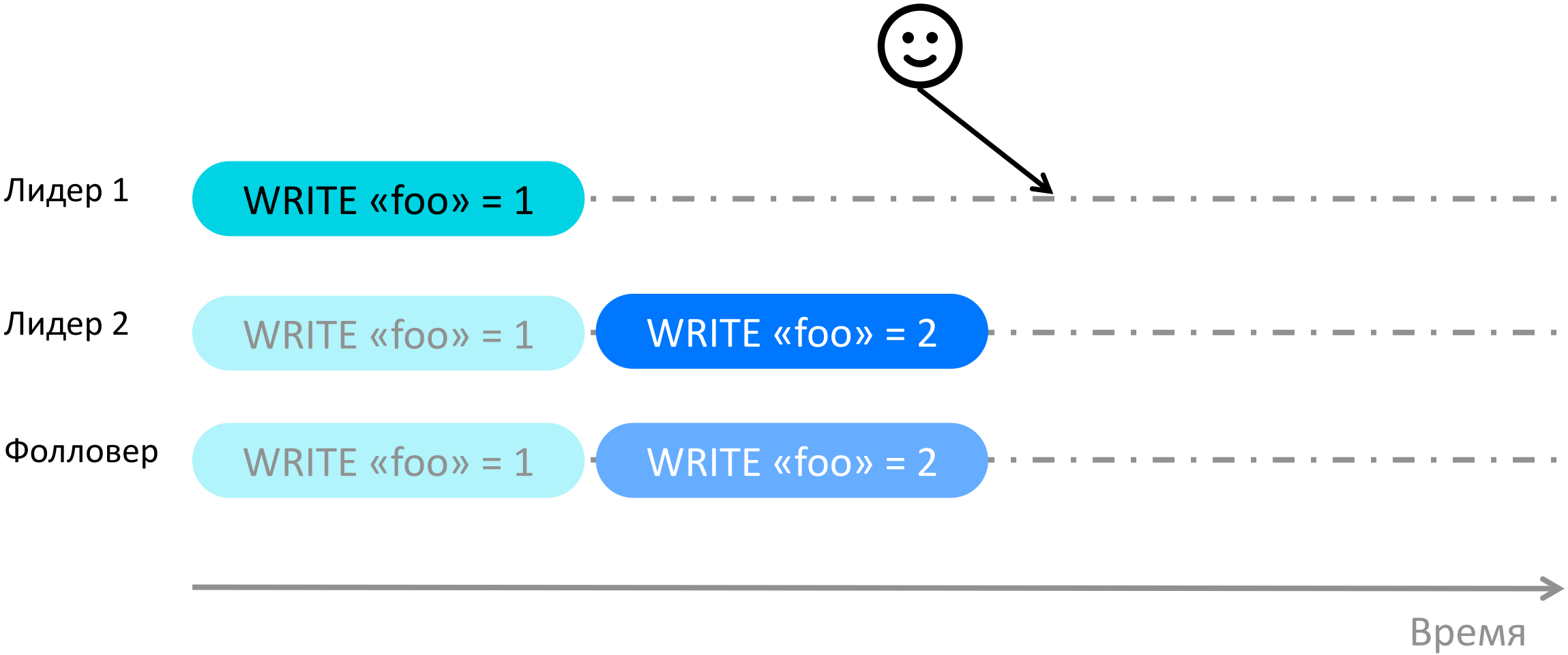
Линеаризуемость в Raft



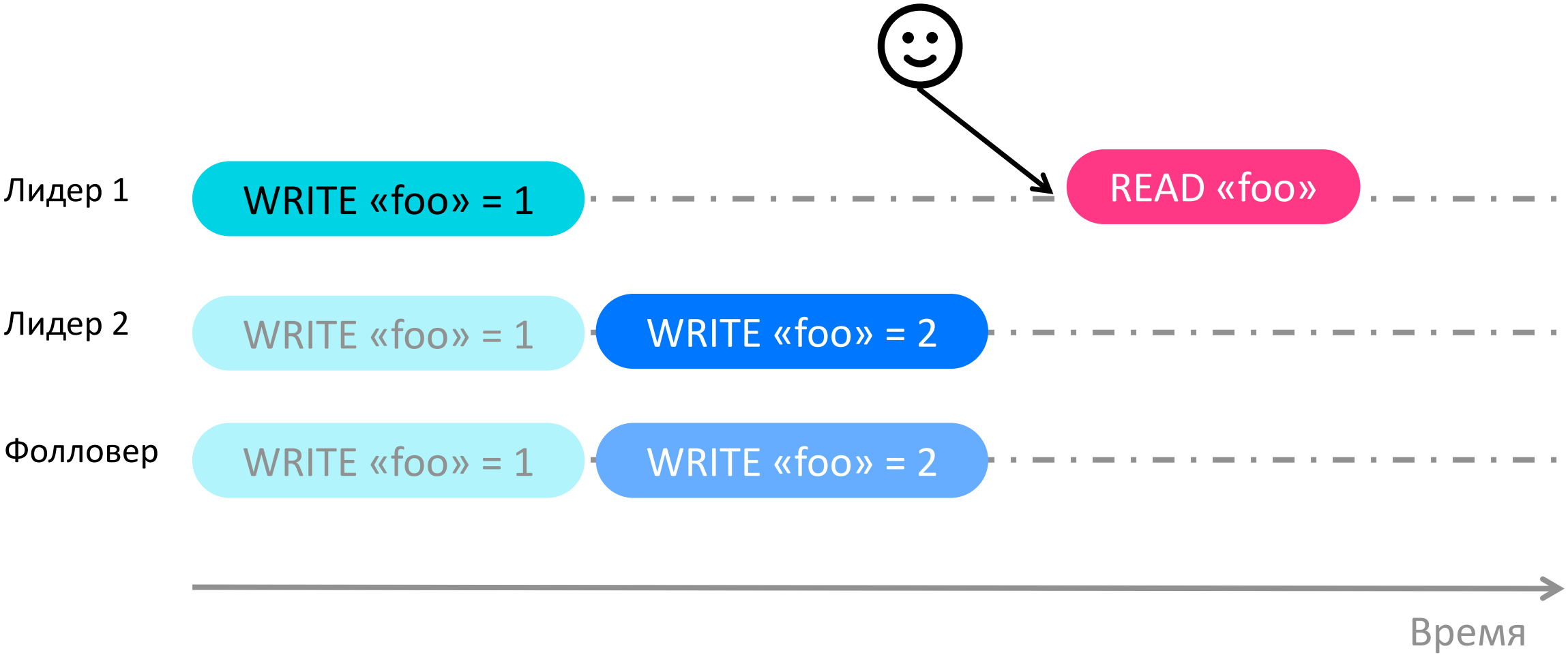
Линеаризуемость в Raft



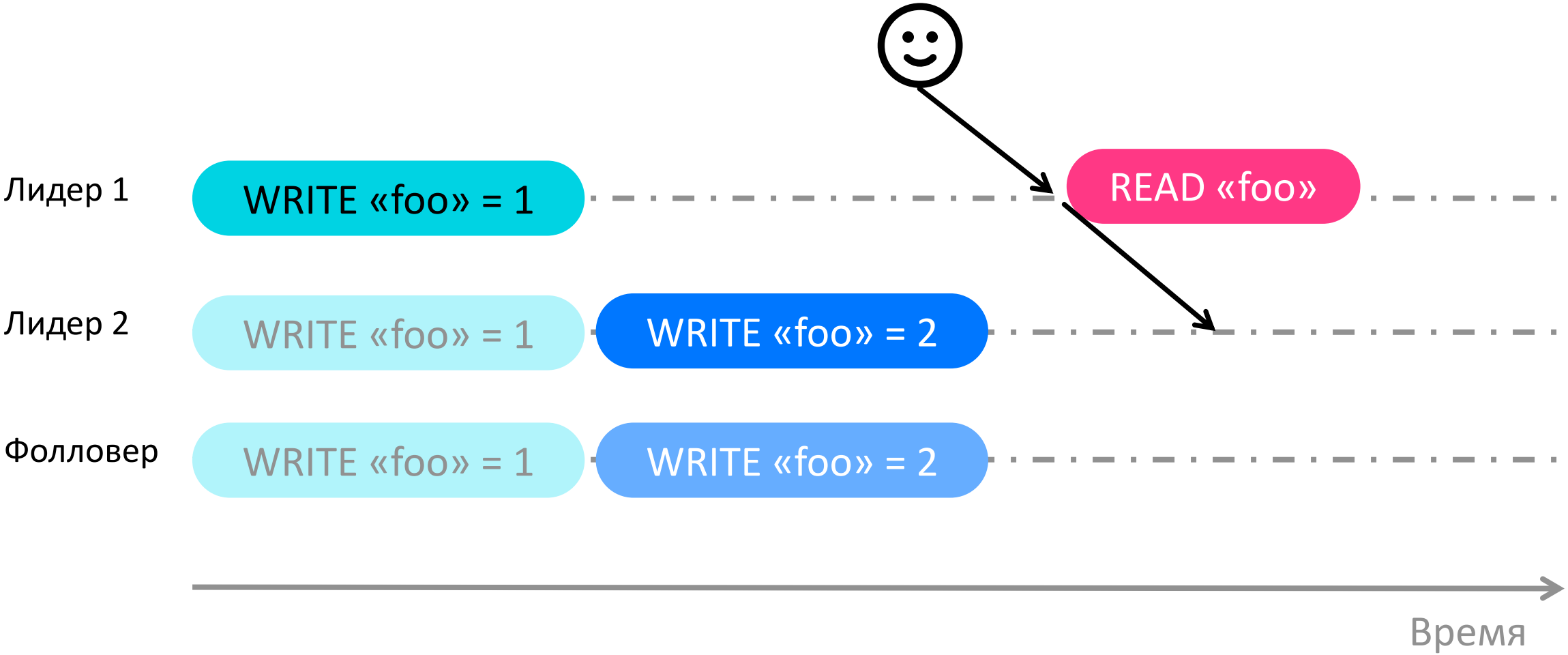
Линеаризуемость в Raft



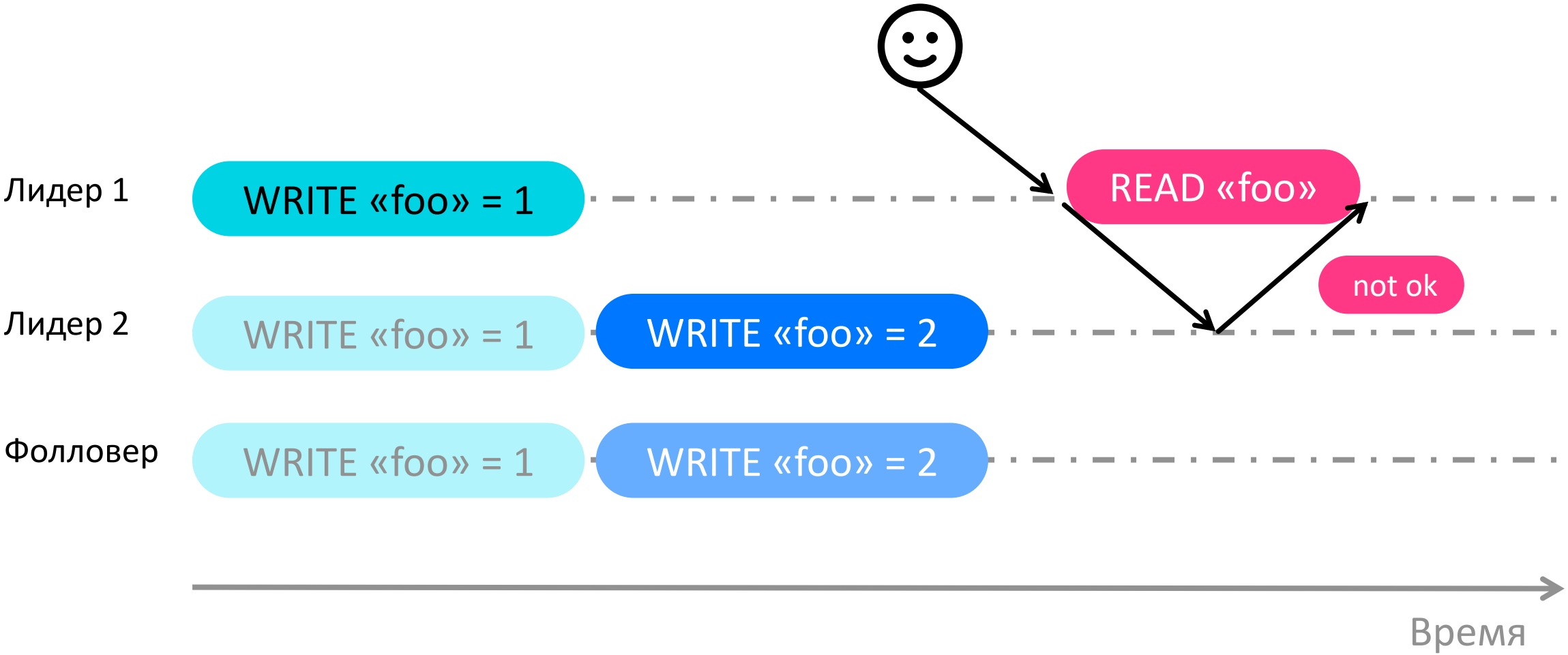
Линеаризуемость в Raft



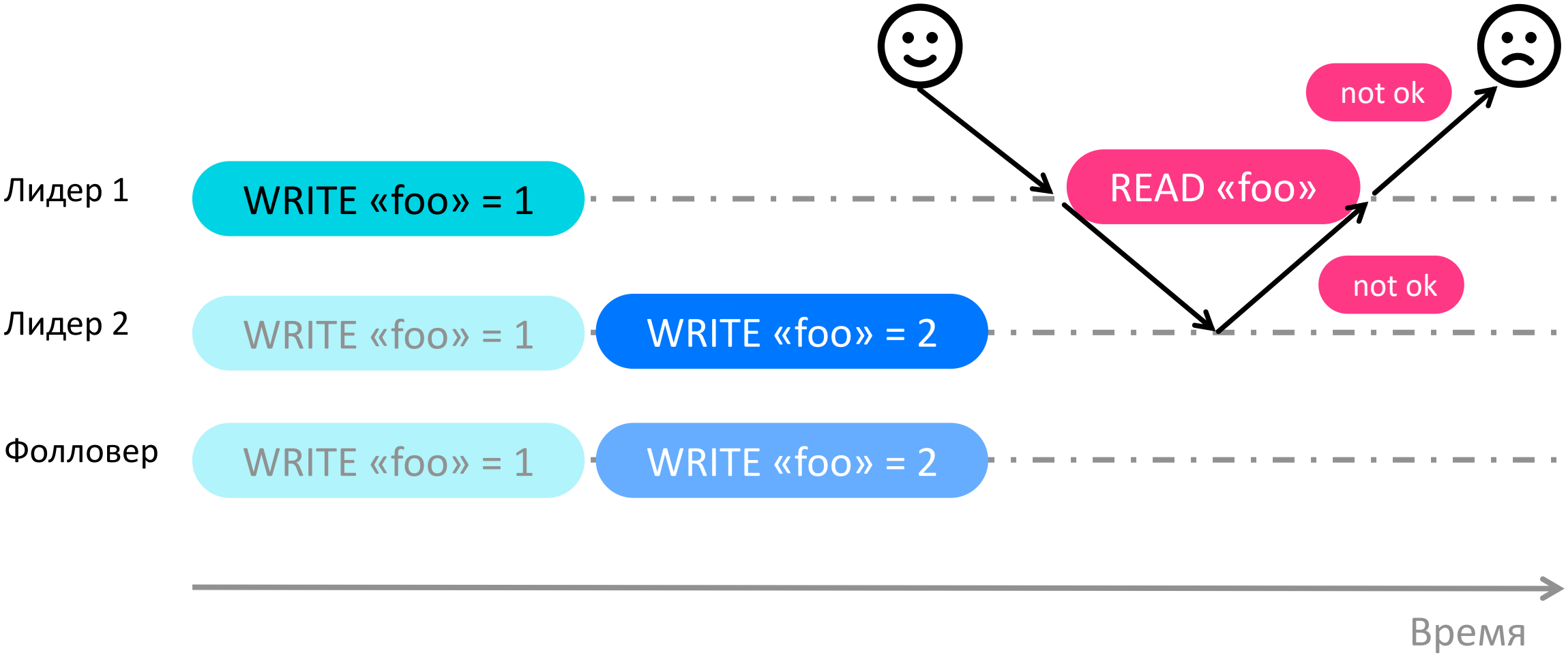
Линеаризуемость в Raft



Линеаризуемость в Raft



Линеаризуемость в Raft



Кто делает так же?

etcd/raft

- etcd
- CockroachDB
- TiKV/TiDB

Линеаризуемость в etcd/raft

1. Запомнить текущий commit index
2. Выпустить heartbeat ко всем и дождаться ответа от большинства
3. Больше не лидер? Вернуть ошибку
4. Иначе дождаться применения записи, соответствующей commit index
5. Выполнить чтение

Линеаризуемость в etcd/raft

1. Запомнить текущий commit index
2. Выпустить heartbeat ко всем и дождаться ответа от большинства
3. Больше не лидер? Вернуть ошибку
4. Иначе дождаться применения записи, соответствующей commit index
5. Выполнить чтение

Что не так с этим подходом?

Вся нагрузка ложится на лидера

Линеаризуемость в Tarantool

1. Выполнить на любом узле

Линеаризуемость в Tarantool

1. Выполнить на любом узле

2. Найти среди кворума наибольший log index

Линеаризуемость в Tarantool

1. Выполнить на любом узле
2. Найти среди кворума наибольший log index
3. Дождаться получения операций до этого log index

Линеаризуемость в Tarantool

1. Выполнить на любом узле
2. Найти среди кворума наибольший log index
3. Дождаться получения операций до этого log index
4. Дождаться, пока все операции до log index будут завершены

Линеаризуемость в Tarantool

1. Выполнить на любом узле
2. Найти среди кворума наибольший log index
3. Дождаться получения операций до этого log index
4. Дождаться, пока все операции до log index будут завершены
5. Выполнить чтение

Почему это работает?

1. Последняя подтверждённая запись присутствует на W = N / 2 + 1 узлов

Почему это работает?

1. Последняя подтверждённая запись присутствует на $W = N / 2 + 1$ узлов
2. Значит присутствуют в журналах этих узлов

Почему это работает?

1. Последняя подтверждённая запись присутствует на $W = N / 2 + 1$ узлов
2. Значит присутствуют в журналах этих узлов
3. Опросив $R = N / 2 + 1$ узлов, встретим хотя бы один с этой записью

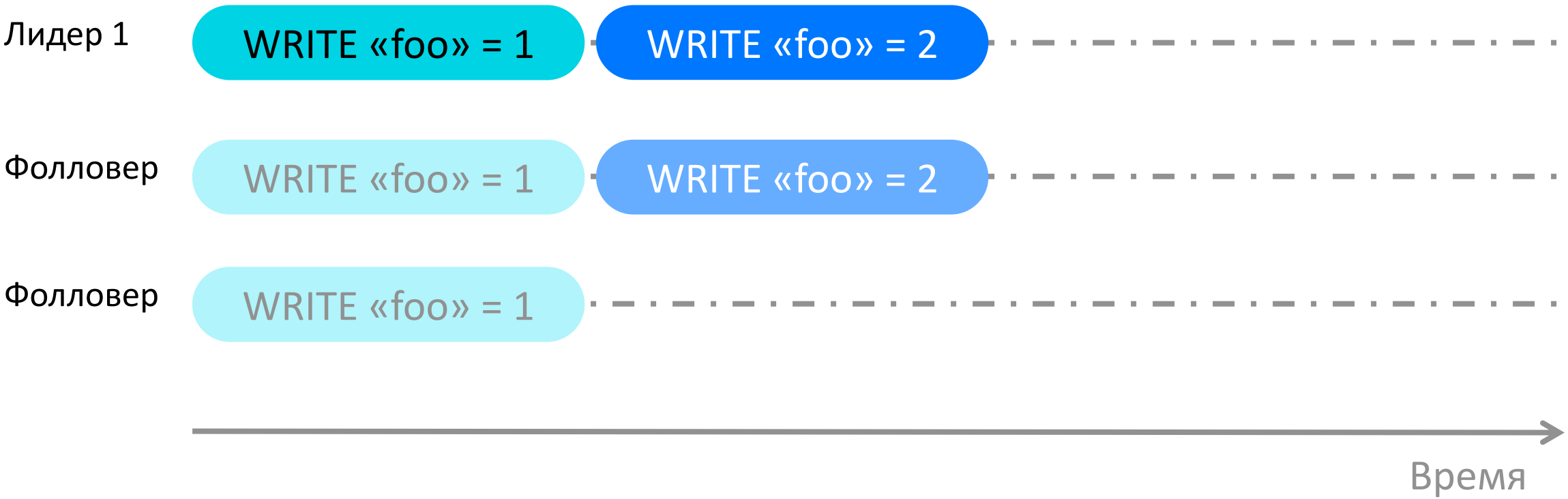
Почему это работает?

1. Последняя подтверждённая запись присутствует на W = N / 2 + 1 узлов
2. Значит присутствуют в журналах этих узлов
3. Опросив R = N / 2 + 1 узлов, встретим хотя бы один с этой записью
4. Максимальный log index из кворума >= log index этой записи

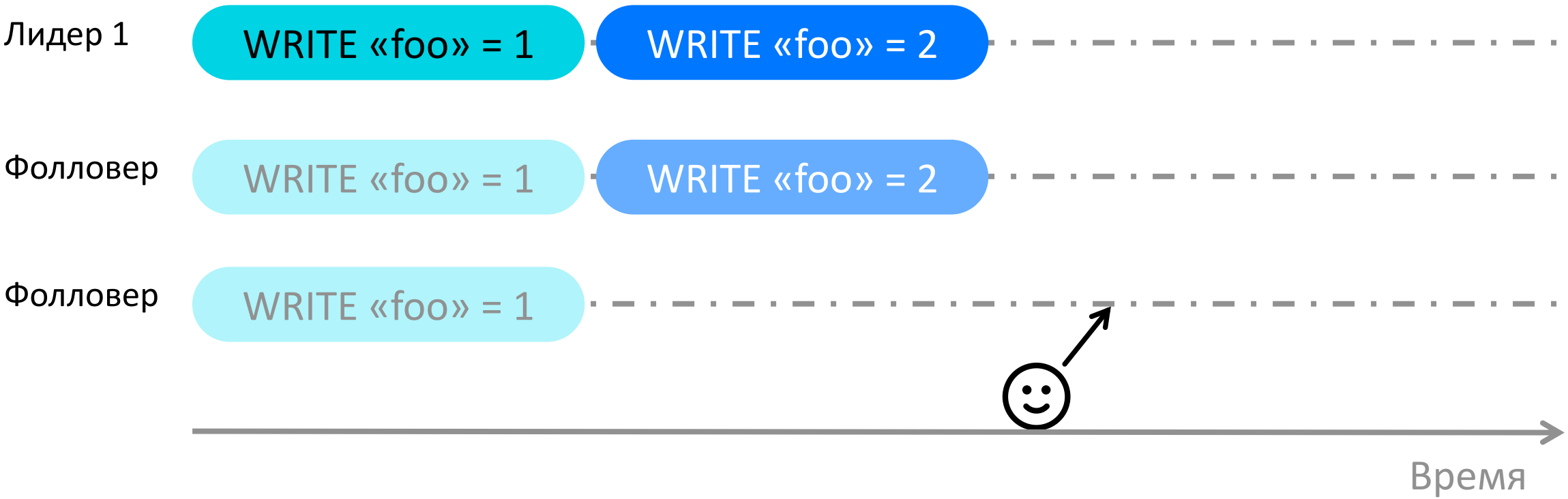
Почему это работает?

1. Последняя подтверждённая запись присутствует на W = N / 2 + 1 узлов
2. Значит присутствуют в журналах этих узлов
3. Опросив R = N / 2 + 1 узлов, встретим хотя бы один с этой записью
4. Максимальный log index из кворума >= log index этой записи
5. Дождавшись завершения всех записей вплоть до log index, можем выполнить чтение

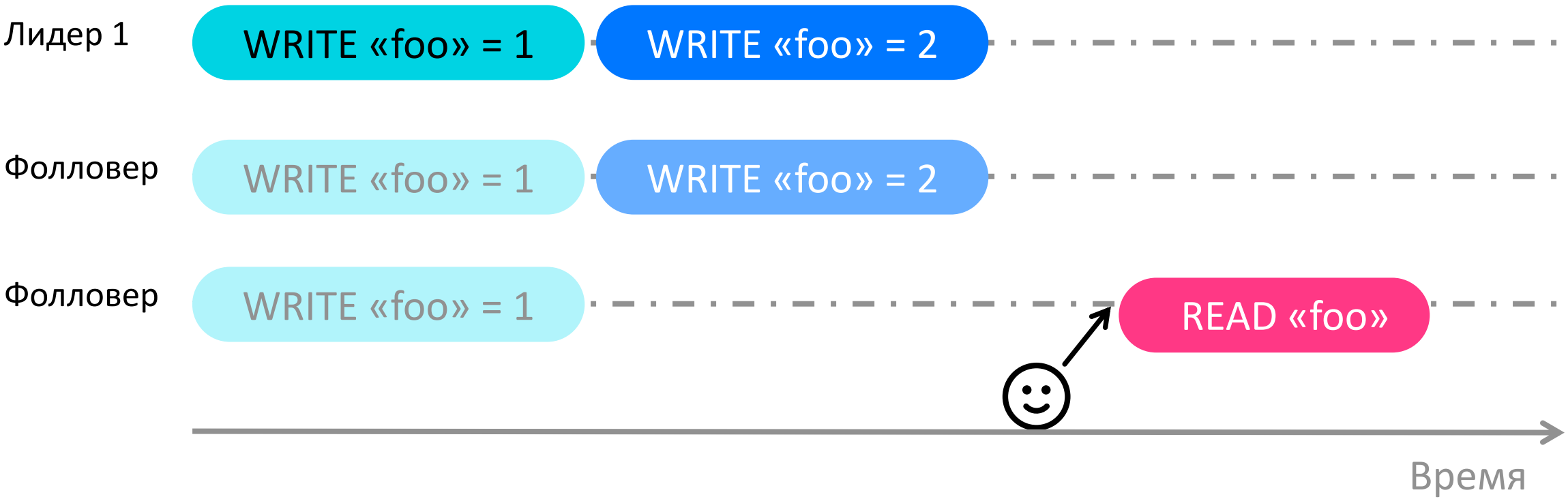
Линеаризуемость в Tarantool



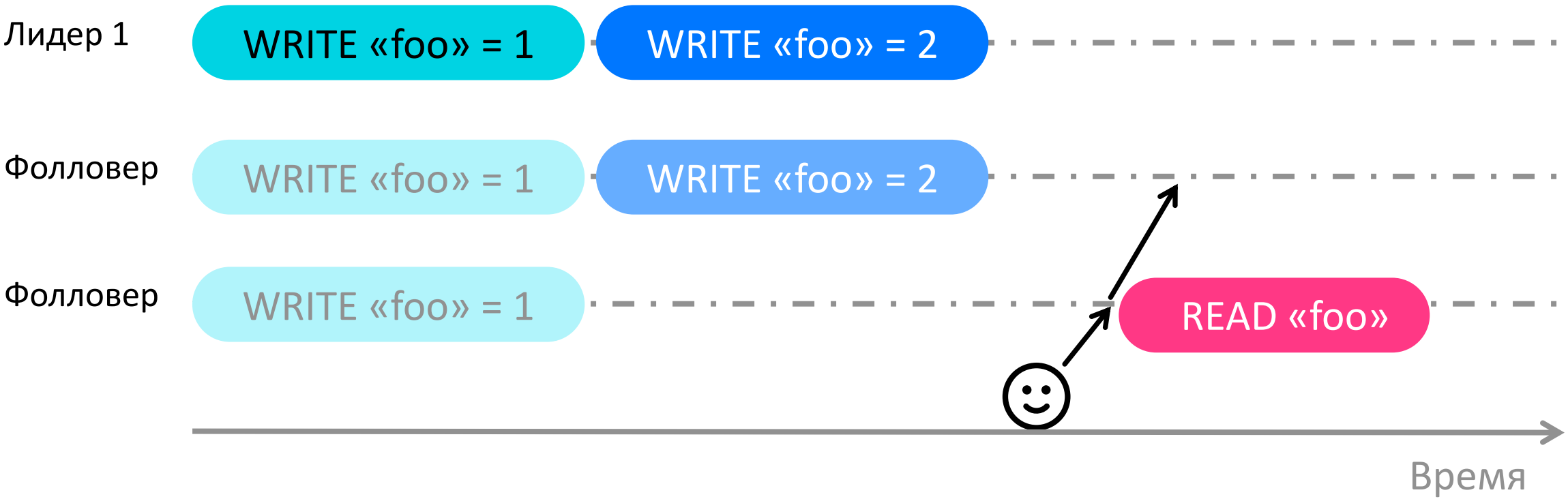
Линеаризуемость в Tarantool



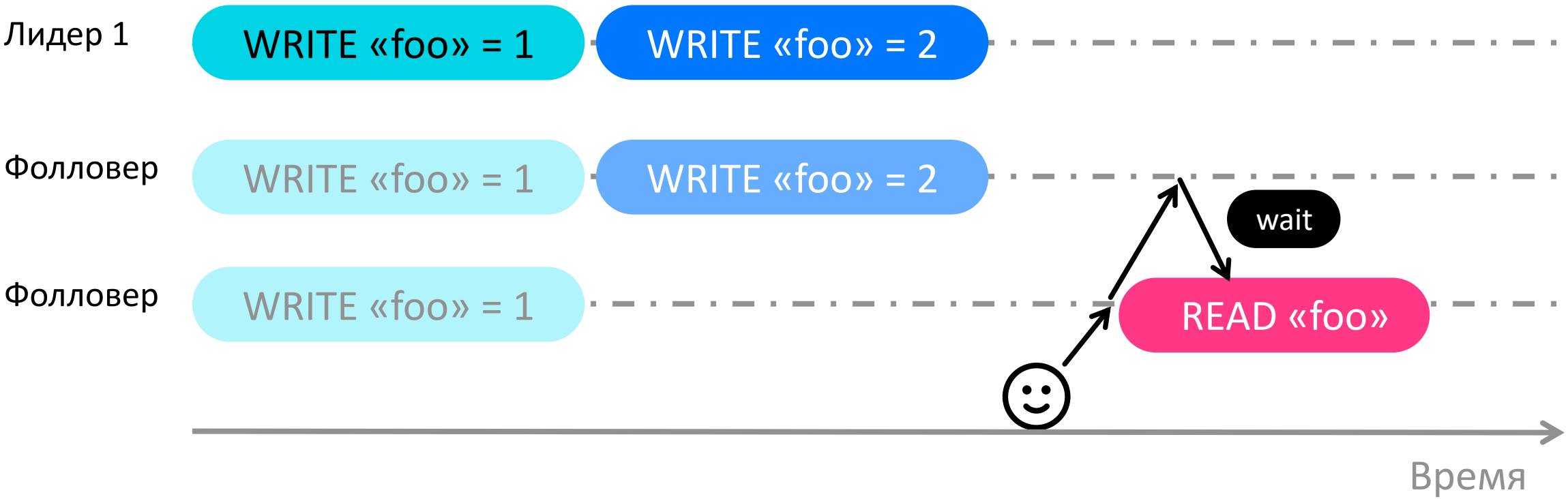
Линеаризуемость в Tarantool



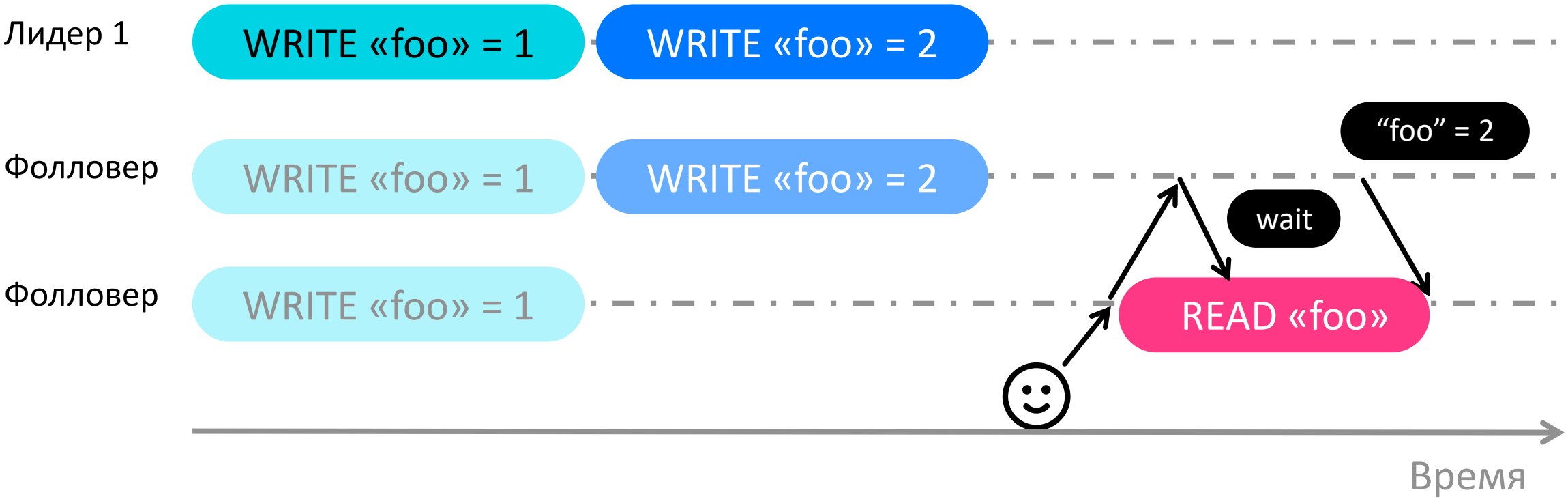
Линеаризуемость в Tarantool



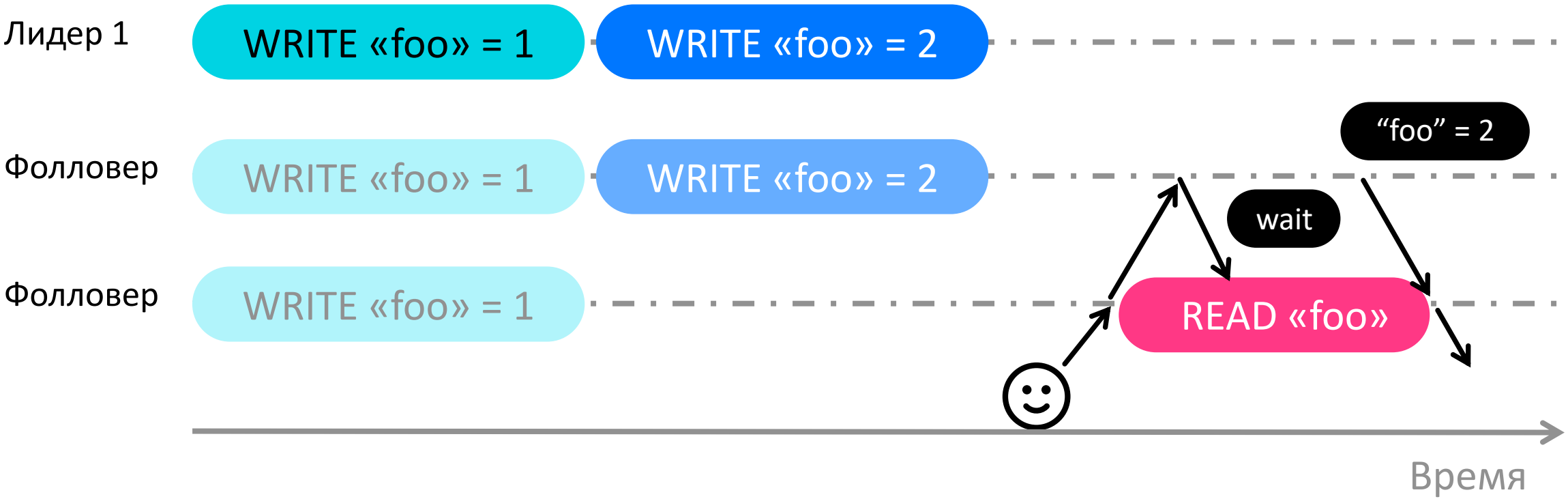
Линеаризуемость в Tarantool



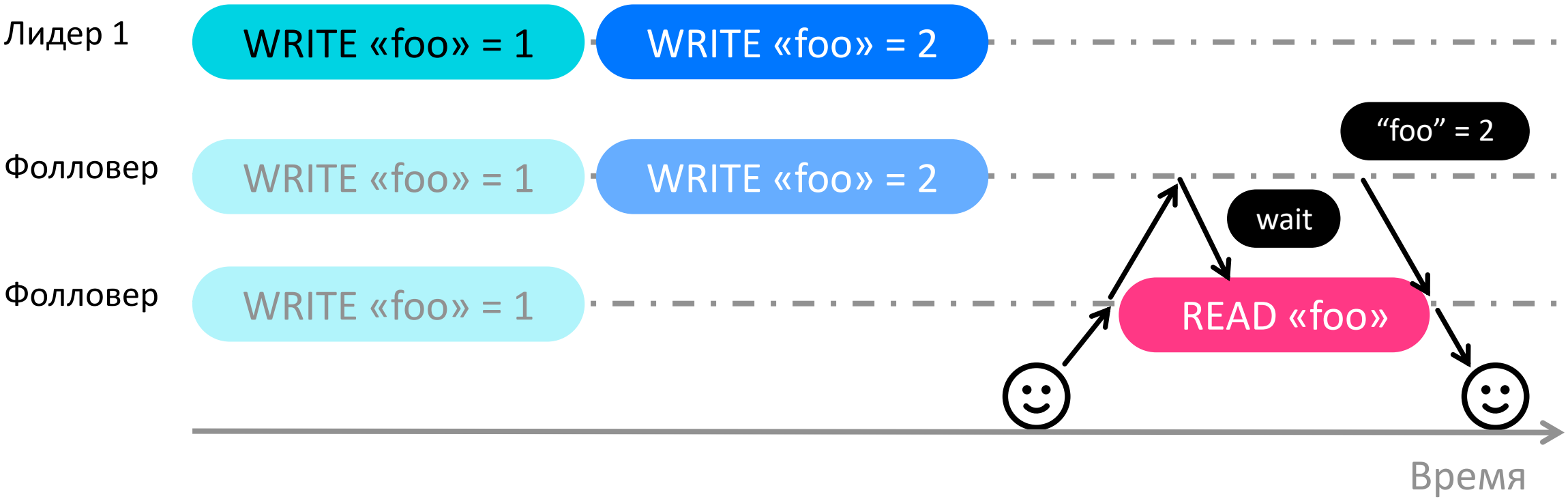
Линеаризуемость в Tarantool



Линеаризуемость в Tarantool



Линеаризуемость в Tarantool



Подходы к линеаризуемости

etcd & co

1 чтение на лидере

Подходы к линеаризуемости

etcd & co

1. чтение на лидере

Tarantool

1. чтение не нагружает лидера

Подходы к линеаризуемости

etcd & co

1. чтение на лидере
2. раунд heartbeat раз в ~100 ms

Tarantool

1. чтение не нагружает лидера

Подходы к линеаризуемости

etcd & co

1. чтение на лидере
2. раунд heartbeat раз в ~100 ms

Tarantool

1. чтение не нагружает лидера
2. раунд heartbeat раз в ~10 ms (только если есть чтения)

Подходы к линеаризуемости

etcd & co

1. чтение на лидере
2. раунд heartbeat раз в ~100 ms
3. latency изредка больше на ~RTT

Tarantool

1. чтение не нагружает лидера
2. раунд heartbeat раз в ~10 ms (только если есть чтения)

Подходы к линеаризуемости

etcd & co

1. чтение на лидере
2. раунд heartbeat раз в ~100 ms
3. latency изредка больше на ~RTT

Tarantool

1. чтение не нагружает лидера
2. раунд heartbeat раз в ~10 ms (только если есть чтения)
3. latency всегда больше на ~RTT

Цена линеаризуемости

Снижение устойчивости к сбоям

Обычная система

Линеаризуемая система

Функционирует пока
функционирует хотя бы 1 узел

Снижение устойчивости к сбоям

Обычная система

Функционирует пока функционирует хотя бы 1 узел

Линеаризуемая система

Функционирует пока функционирует большинство узлов

Снижение устойчивости к сбоям

Обычная система

Функционирует пока функционирует хотя бы 1 узел

Линеаризуемая система

Функционирует пока функционирует большинство узлов

Плохо работает в двух ЦОДах

Снижение производительности

Обычная система

1 Получить запрос от клиента

Линеаризуемая система

1 Получить запрос от клиента

Снижение производительности

Обычная система

1. Получить запрос от клиента
2. Обработать запрос локально

Линеаризуемая система

1. Получить запрос от клиента
2. Отправить запрос кворуму

Снижение производительности

Обычная система

1. Получить запрос от клиента
2. Обработать запрос локально
3. Отправить ответ клиенту

Линеаризуемая система

1. Получить запрос от клиента
2. Отправить запрос кворуму
3. Дождаться ответа кворума

Снижение производительности

Обычная система

1. Получить запрос от клиента
2. Обработать запрос локально
3. Отправить ответ клиенту

Линеаризуемая система

1. Получить запрос от клиента
2. Отправить запрос кворуму
3. Дождаться ответа кворума
4. Отправить ответ пользователю

Снижение производительности

Обычная система

1. Получить запрос от клиента
2. Обработать запрос локально
3. Отправить ответ клиенту
4. Latency ≈ RTT
от клиента до узла

Линеаризуемая система

1. Получить запрос от клиента
2. Отправить запрос кворуму
3. Дождаться ответа кворума
4. Отправить ответ пользователю
5. Latency ≈ RTT от клиента
до узла + max RTT в кворуме

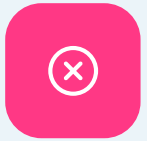
Выводы

Всем нужна линеаризуемость?

Если бесплатно – да

Всем нужна линеаризуемость?

Если бесплатно – да



Но она не бесплатна.

Так что почти никому

Всем нужна линеаризуемость?

Минусы

↓ Низкая производительность

Всем нужна линеаризуемость?

Минусы

Низкая производительность

Для записи – всегда

Всем нужна линеаризуемость?

Минусы

Низкая производительность

Для записи – всегда

Для чтений – во время использования

Всем нужна линеаризуемость?

Минусы

Низкая производительность

Для записи – всегда

Для чтений – во время использования

Плюсы

Легко писать приложения

Tarantool

Что ещё почитать

Документация Tarantool

бизнес

Tarantool

Что ещё почитать

блог CockroachDB

обзор уровней консистентности

и ещё один

и ещё

бизнес