

# Использование model checking для тестирования распределенных систем в фреймворке DSLab

Амеличев Константин, БПМИ195

Академический руководитель: Сухорослов Олег Викторович, ФКН

# Решаемая проблема

Тестировать распределенные системы сложно, нужно учитывать редкие случаи.

При проведении курса “Распределенные системы” требуется автоматически проверять решения на корректность и эффективность.

Текущие тесты (симуляция + fault injection + chaos monkey) на основе фреймворка dslib не дают уверенности в корректности алгоритма — симуляция может пропустить ситуацию, если вероятность ее возникновения была мала.

Также dslib больше не поддерживается, вместо него развивается более функциональный симуляционный фреймворк DSLab

## Цель работы

- Улучшение тестов к заданиям в DSLab на основе model checking (**MC**).

## Поставленные задачи

- Изучение MC и его применения для тестирования PC
- Улучшение ядра MC в DSLab
- Разработка тестов к заданиям курса PC с использованием MC
- Тестирование архивных решений и анализ найденных багов

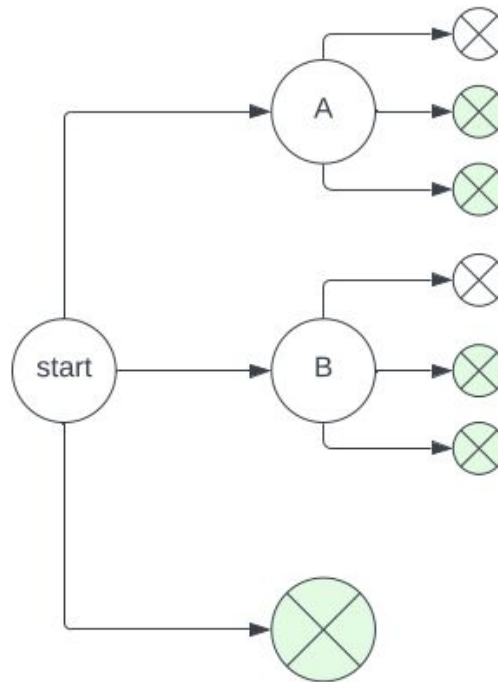
## Требования

- Интерпретируемость ошибок
- Воспроизводимость ошибок
- Тесты работают не более нескольких минут.

# Методы формальной верификации

- Спецификация
  - TLA+ [1]
  - Verdi [2]
- Model checking
  - FlyMC [3]
  - CDSchecker [4]
  - MaceMC [5]

Model checking использует только исходный код, явно обходит граф состояний системы и лучше подходит для тестирования заданий.



1) Lamport L. Specifying systems: the TLA+ language and tools for hardware and software engineers.  
2) Wilcox, James R., et al. "Verdi: a framework for implementing and formally verifying distributed systems."  
3) Lukman J. F. et al. Flymc: Highly scalable testing of complex interleavings in distributed systems  
4) Norris B., Demsky B. CDSchecker: checking concurrent data structures written with C/C++ atomics  
5) Killian C. et al. Life, death, and the critical transition: Finding liveness bugs in systems code.

# Model checking в образовательных проектах

- DSLabs <sup>[1]</sup>
  - Другой набор заданий (см. таблицу)
  - В сложных заданиях используется отсечение по времени
- Курс по РС в Columbia University <sup>[2]</sup>
  - Одно задание
  - Требуется составить тесты в режиме MC

Название	Статус model checking	Размер решения, LOC
0-pingpong	полный	10
1-clientserver	полный	75
2-primarybackup	полный	300
3-paxos	неполный	400
4-shardedstore	неполный	1050

1) Michael E. et al. Teaching rigorous distributed systems with efficient model checking

2) Columbia University, Distributed Systems Fundamentals, Assignment 5

# Работа над МС в ВШЭ



- **dslib<sub>[1,2]</sub>**
  - Нацелен исключительно на курс по РС
  - Были работы по внедрению МС, но результаты не были применимы для тестирования студенческих решений
- **DSLAb<sub>[3]</sub>**
  - Фреймворк для симуляции более общего характера
  - Ядро МС перенесено из dslib с некоторыми улучшениями

1) Удовиченко А., Реализация поддержки model checking в рамках фреймворка dslib. --- НИУ ВШЭ, 2022.

2) Ильговский Р., Разработка фреймворка для практических заданий по распределенным системам --- НИУ ВШЭ, 2021.

3) DSLab, проектная группа ФКН, <https://github.com/osukhoroslov/dslab>

# Ядро МС

Поддерживается модель распределенной системы, идентичная модели в DSLab-MP.

Можно применить любое событие (таймер/сообщение) из множества доступных. Как результат, могут появиться новые события.

Обход графа состояний системы делается стратегией, которая пользуется заданными предикатами **Invariant**, **Goal**, **Prune**.

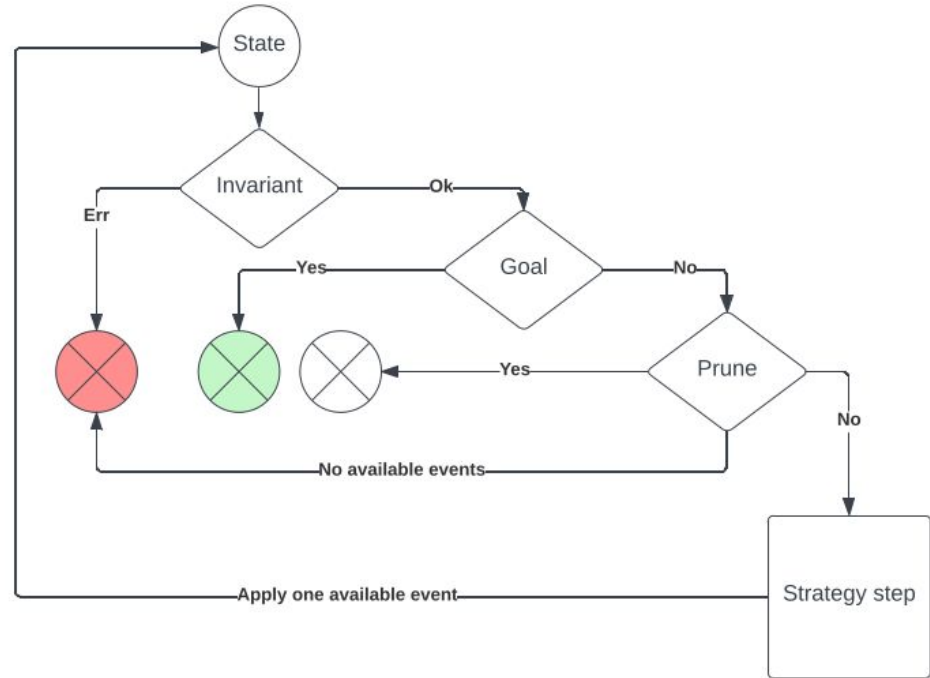


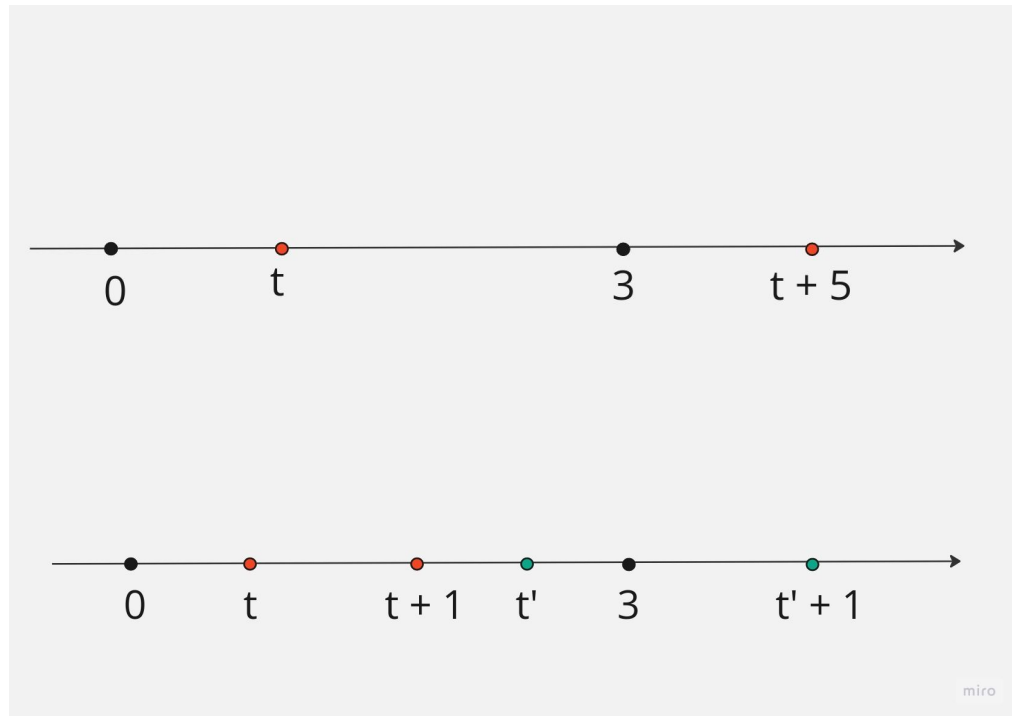
Схема работы стратегии

# Улучшение ядра МС: зависимые события

Изучены критерии, по которым события оказываются связаны happened-before.

Событие недоступно, если оно зависит от неслучившегося события.

**Результат:** порядок срабатывания событий корректен





# Улучшение ядра МС: Оптимизации обхода

- Добавлено кеширование эквивалентных состояний
- Задан порядок обработки дублирующихся сообщений

	Без кеширования	С кешированием
# goal	677	87
# prune	1422	196

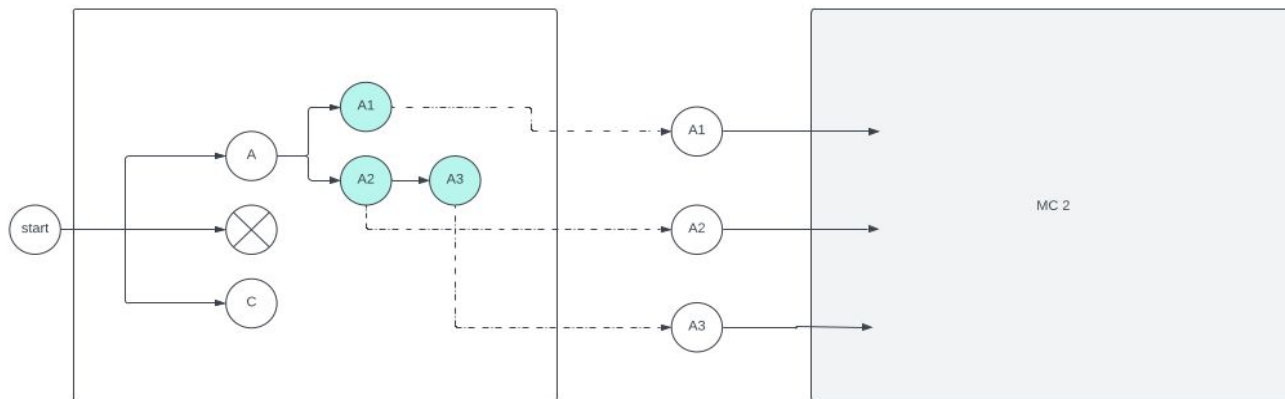
	До ограничения	После ограничения
# goal	222	87
# prune	629	196

**Результат:** Обход графа состояний ускорен в несколько раз

Тесты оптимизаций на задании Ping Pong

# Улучшение ядра МС: дополнительные режимы

- Instant Mode: временем доставки сообщения можно пренебречь.
  - **Результат:** удобный режим для детекторов отказов
- Collect Mode: позволяет сделать серию последовательных запусков model checking.
  - **Результат:** возможность тестирования произвольных сценариев (Пример: kv-хранилище, GET + PUT)



# Разработка тестов: общий цикл

1. Выделение гарантий в задании
2. Выбор сценария (отказ узла, разделение сети, ...)
3. Настройка **goal + invariant**
4. Оптимизация обхода через **prune**
5. Анализ архивных решений
6. Если в решении баг, баг документируется
7. Если ошибка в тесте, тест чинится

Таблица 4.1: Тесты для ping-pong

Тест	Goal	Invariant	Prune
mc reliable network	Получение 2 сообщений	Имеющиеся ответы верны	Каждый из узлов отправил не больше 4 сообщений
mc unreliable network	Получение 2 сообщений	Имеющиеся ответы верны	Глубина состояния не больше 7
mc limited drops	Получение 2 сообщений	Имеющиеся ответы верны	Не больше 3 потерь сообщения, узел отправил не больше 5 сообщений

**Результат:** 20 тестов для 7 заданий, проанализированы на архиве из ~80 решений к каждому заданию.

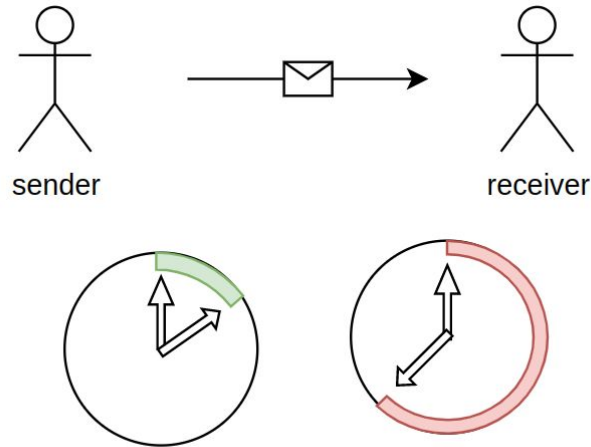
# Пример тестов в режиме Collect: Broadcast

Таблица 4.6: Тест mc\_sender\_crash

Этап	Goal	Invariant	Prune	Collect
До отказа	Сообщение доставлено всеми корректными узлами	Соблюден формат доставленных клиенту сообщений	Глубина не более 4	Хотя бы один узел получил сообщение
Выведение из строя отправителя	-	-	-	-
Доставка после отказа	Сообщение доставлено всеми корректными узлами	Соблюден формат доставленных клиенту сообщений	Каждый узел отправил не больше 4 сообщений, эквивалентные состояния игнорируются, глубина не более 6	-

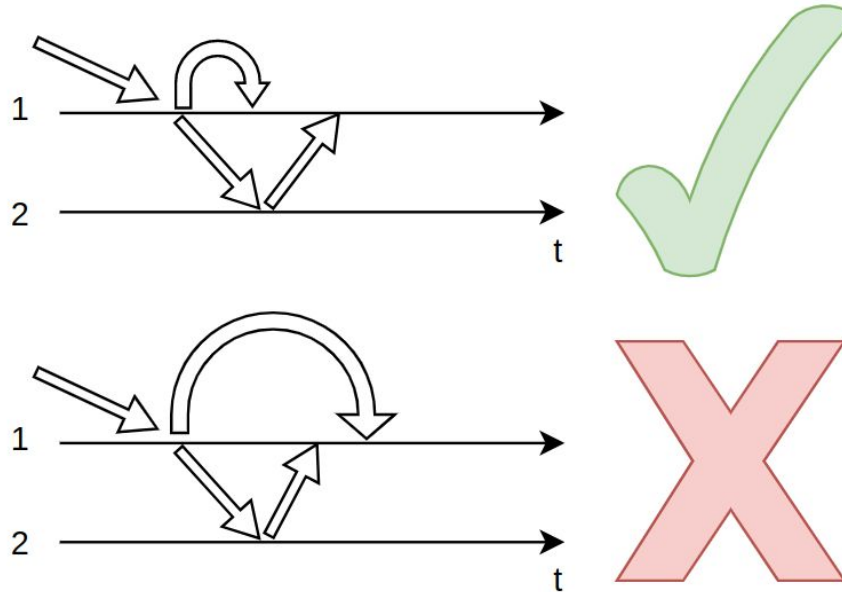
# Найденные ошибки: примеры (1)

Решение чистит ресурсы (чтобы пройти тесты на оптимизацию работы с памятью), но делает это слишком рано и нарушает требование на доставку сообщения в конечном итоге.



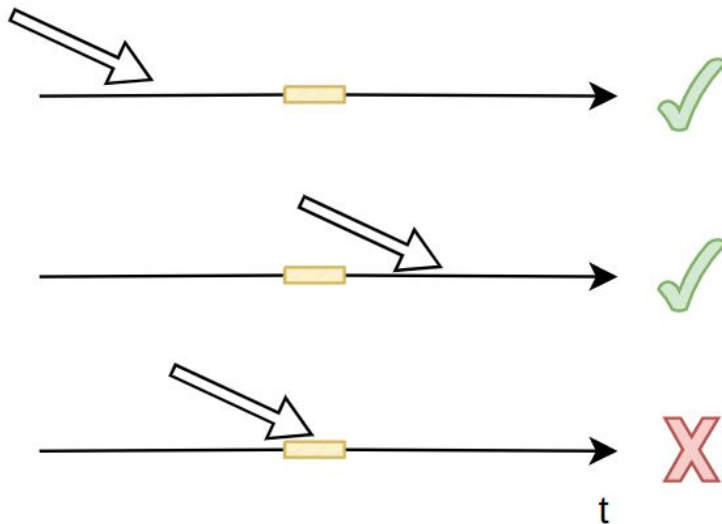
## Найденные ошибки: примеры (2)

Решение делает массовую рассылку сообщения и поддерживает счетчик количества ответов, но не инициализирует счетчик, пока не получит ответ от самого себя.



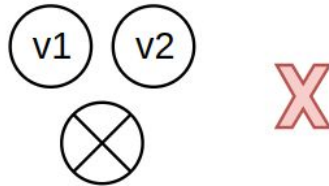
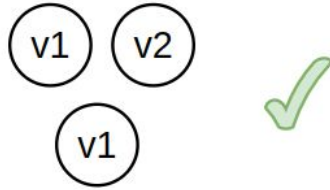
## Найденные ошибки: примеры (3)

Решение ставит два таймера вплотную и между срабатыванием первого и второго находится в некорректном состоянии.



## Найденные ошибки: примеры (4)






















Решение некорректно реализует разрешение конфликтов при работе с кворумом (Last Write Wins). В симуляции PUT-запросы обрабатывали корректно на всех трех репликах, и такая ситуация не возникала.





# Разработка тестов: выводы

- MS-тесты помогают найти сценарии для стандартных тестов
  - Может быть полезно для системы бонусов на курсе PC
- Ограничение времени работы MS через **prune** лучше, чем явное ограничение по времени
  - Ограничение по времени: чем медленнее обход состояний, тем выше шанс пройти тесты
  - Ограничение через **prune**: чем больше состояний порождает система, тем меньше шанс пройти тесты

Aa Name	Status	Task
 ping-pong normal	● Done	ping-pong
 ping-pong half-reliable	● Done	ping-pong
 ping-pong unreliable	● Done	ping-pong
 guarantees half-reliable [EO]	● Done	guarantees
 guarantees unreliable [EO]	● Done	guarantees
 guarantees normal [EO]	● Done	guarantees
 broadcast normal	● Done	broadcast
 broadcast sender crash	● Done	broadcast
 membership normal	● Done	membership
 membership local message test	● Done	membership
 membership partition + recovery	● In progress	membership
 kv-sharding normal	● Done	kv-sharding
 kv sharding rebalance	● Done	kv-sharding
 kv-replication normal	● Done	kv-replication
 kv-replication hinted handoff	● Done	kv-replication
 kv-replication-concurrent	● Done	kv-replication
 kv-replication-v2 normal	● Done	kv-replication-v2
 kv-replication-v2 hinted handoff	● Done	kv-replication-v2
 kv-replication-v2 concurrent operations	● Done	kv-replication-v2
 kv-replication-v2 concurrent cart	● Done	kv-replication-v2
 kv-replication-v2 concurrent xcart	● Done	kv-replication-v2

# Результаты

## Практические:

- Перенос заданий курса PC: dslib ➡ DSLab
- Улучшение ядра MC: DependencyResolver, Collect mode, ... (~700 LOC)
- 20 новых тестов ко всем заданиям курса (~2400 LOC)
- Найдено 19 различных ошибок в архивных решениях

## Новизна:

- Полученные MC-тесты не требуют отсечений по времени, достаточно отсечений на обход графа состояний
- Punctuated search (DSLabs) ➡ Collect mode
- Опыт исследования архива студенческих решений

# Ссылки на программный код

- Совместная работа по преносу ядра MC dslib DSLab: <https://github.com/osukhoroslov/dslab/pull/172>
- Улучшения ядра MC: <https://github.com/KiK0S/dslab/pull/12>
- Тесты к заданиям курса PC: <https://github.com/KiK0S/dslab/pull/5>,  
<https://github.com/KiK0S/dslab/pull/6>, <https://github.com/KiK0S/dslab/pull/7>,  
<https://github.com/KiK0S/dslab/pull/8>, <https://github.com/KiK0S/dslab/pull/9>,  
<https://github.com/KiK0S/dslab/pull/10>, <https://github.com/KiK0S/dslab/pull/11>