# Распределенные алгоритмы

Сухорослов Олег Викторович

Распределенные системы

Факультет компьютерных наук НИУ ВШЭ

28.11.2020

# План

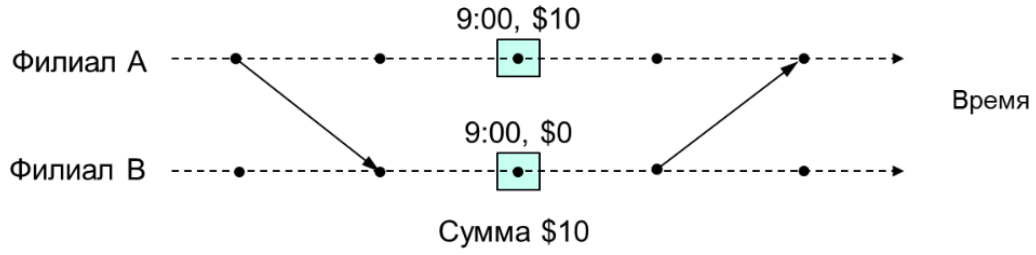
- Построение снимка
- Выборы лидера
- (Взаимное исключение - семинар)

# Особенности

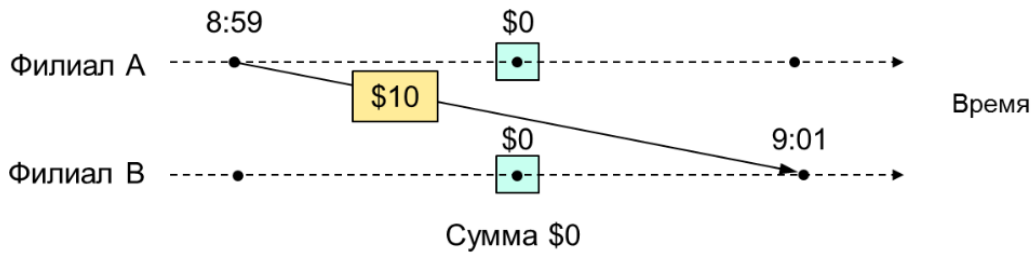
- ~~Классический~~ Распределенный алгоритм
- ~~Последовательное~~ Параллельное исполнение (concurrency)
- ~~Общая память~~ Обмен сообщениями
- ~~Полная~~ Частичная информация (нет глобальных часов)
- ~~Отсутствуют~~ Присутствуют независимые отказы частей системы
- Сложность ~~C(#Op)~~ C(#Op, #Comm)

# Построение снимка (distributed snapshot)

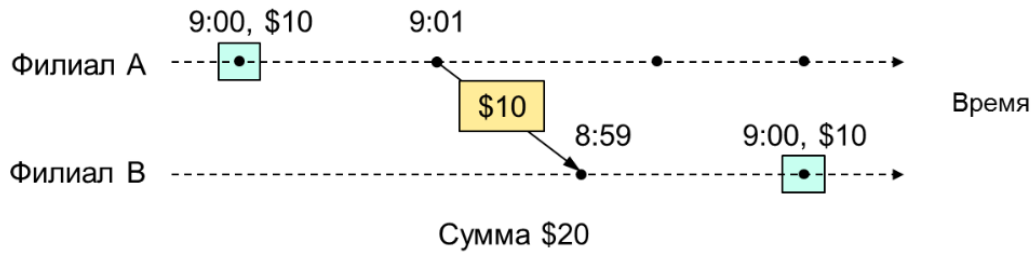
# Пример: деньги в банке




(а)




(б)




(в)

# Зачем нужен снимок РС?

- Checkpointing: восстановление после отказа
- Garbage collection: удаление объектов, на которые никто не ссылается
- Deadlock detection: обнаружение взаимной блокировки
- Termination of computation: определение окончания вычислений
- Проверка стабильных свойств

# Глобальное состояние PC

- Включает в себя
  - Состояние каждого процесса в системе
  - Состояние каждого канала между процессами (передаваемые сообщения)
- Изменяется в результате
  - Локального события (шага) внутри процесса
  - Отправки или получения сообщения
- Переходы между состояниями соответствуют причинному порядку

# Как получить глобальное состояние?

- Остановить на время работу системы
- Использовать физические часы

- Можно ли обойтись без синхронизации между процессами?

# Алгоритм Chandy-Lamport

- Требуется построить снимок с глобальным состоянием РС
  - построение снимка должно выполняться без остановки системы
  - любой процесс может инициировать построение снимка
  - состояние строится распределенно и может быть потом собрано в одно месте
- Модель системы
  - N процессов
  - между каждой парой процессов есть два канала (в обе стороны)
  - каналы надежные и не переупорядочивают сообщения (FIFO)
  - отказы отсутствуют

Chandy K., Lamport L. Distributed snapshots: Determining global states of distributed systems (1985)

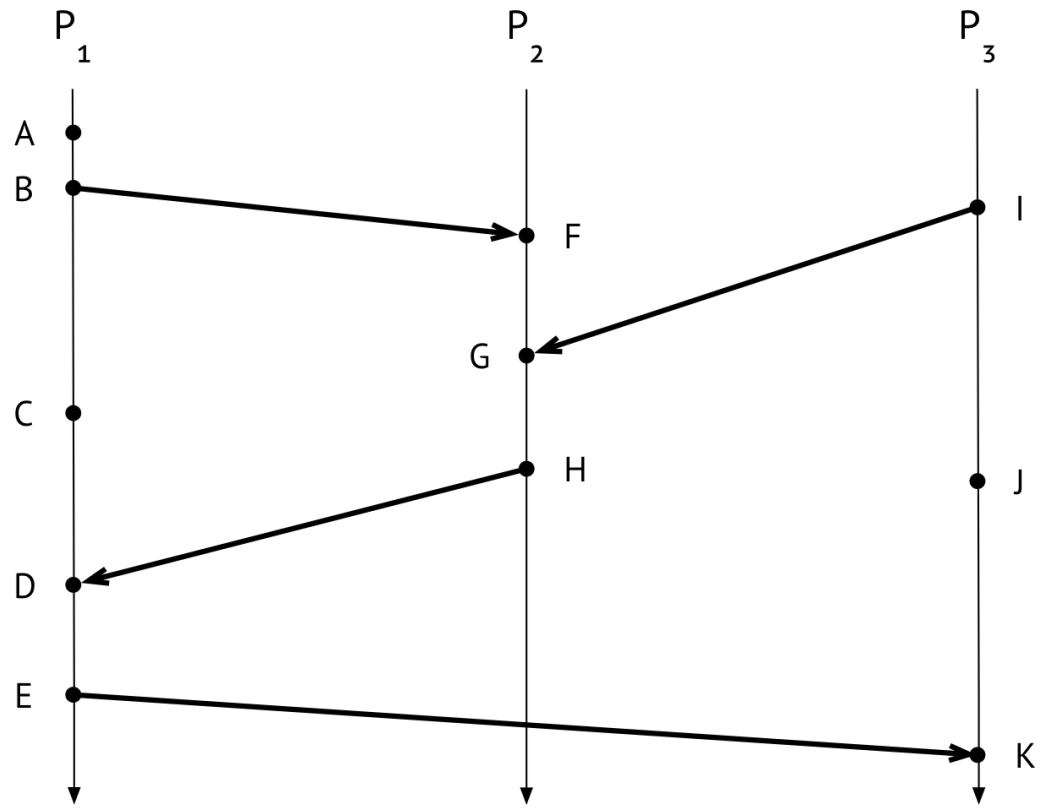
# Алгоритм Chandy-Lamport

- Инициатор снимка записывает свое состояние и рассылает *маркер*
- При получении маркера первый раз процесс
  - записывает свое состояние
  - записывает состояние входящего канала, откуда пришел маркер, как $empty$
  - рассылает маркер по всем исходящим каналам
  - начинает запись сообщений по всем остальным входящим каналам
- При получении маркера повторно процесс
  - останавливает запись сообщений в данном входящем канале
  - сохраняет стояние канала как набор записанных сообщений

# Алгоритм Chandy-Lamport

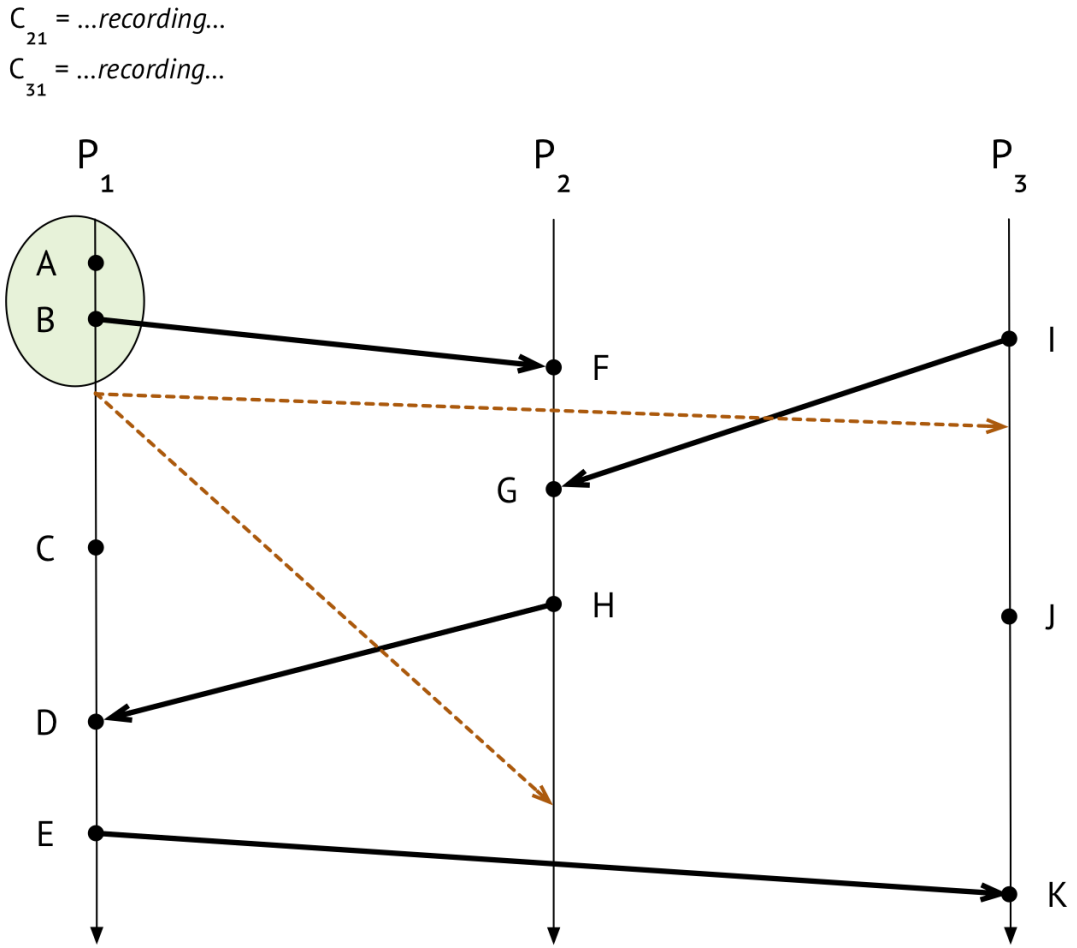
- Алгоритм завершается когда все процессы получили маркер N-1 раз
  - сохранили свое состояние при получении маркера первый раз
  - сохранили состояние каждого из N-1 входящих каналов
- После этого состояние может быть собрано на одном процессе
- Сложность алгоритма:
  - количество сообщений: $O(E)$, где $E$ - число каналов (ребер графа)
  - время работы: $O(d)$, где $d$ - диаметр графа

# Пример

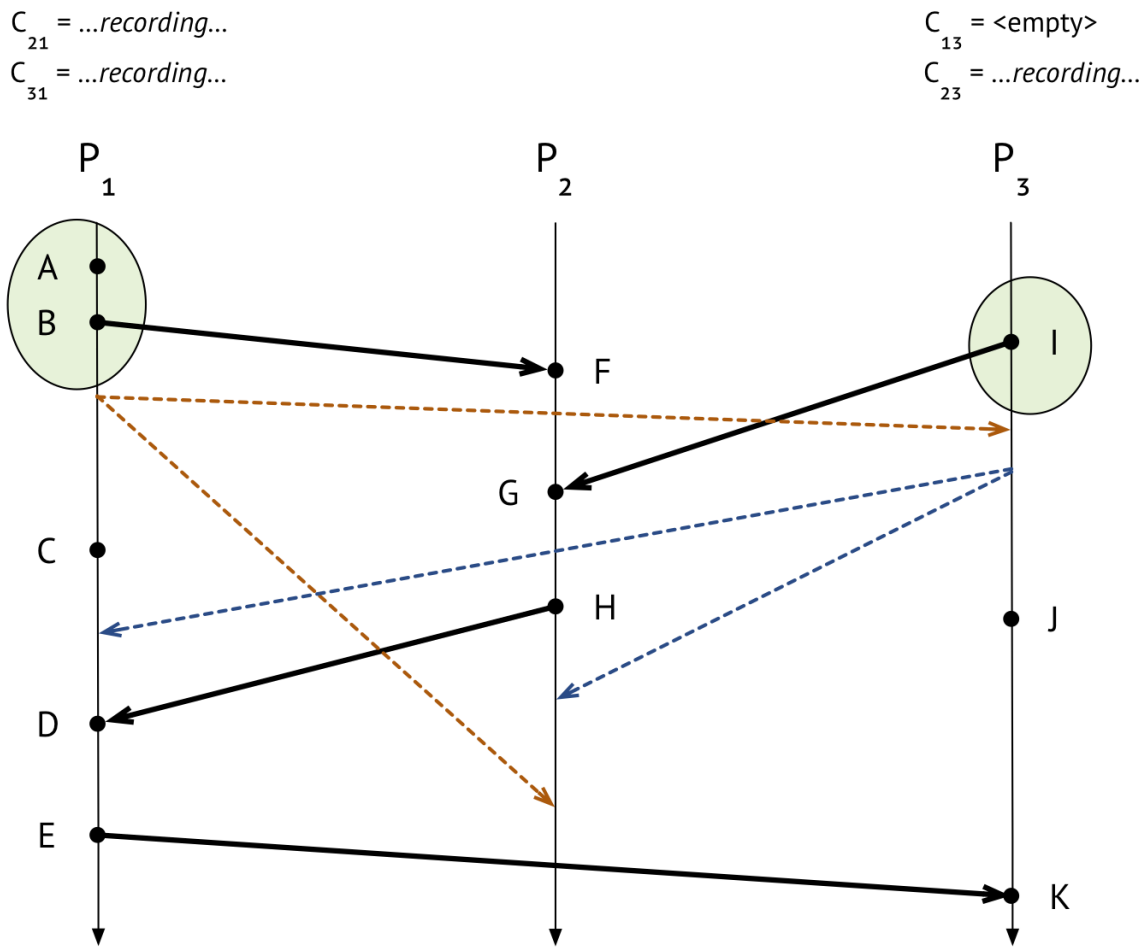



Lindsey Kuper. An example run of the Chandy-Lamport snapshot algorithm

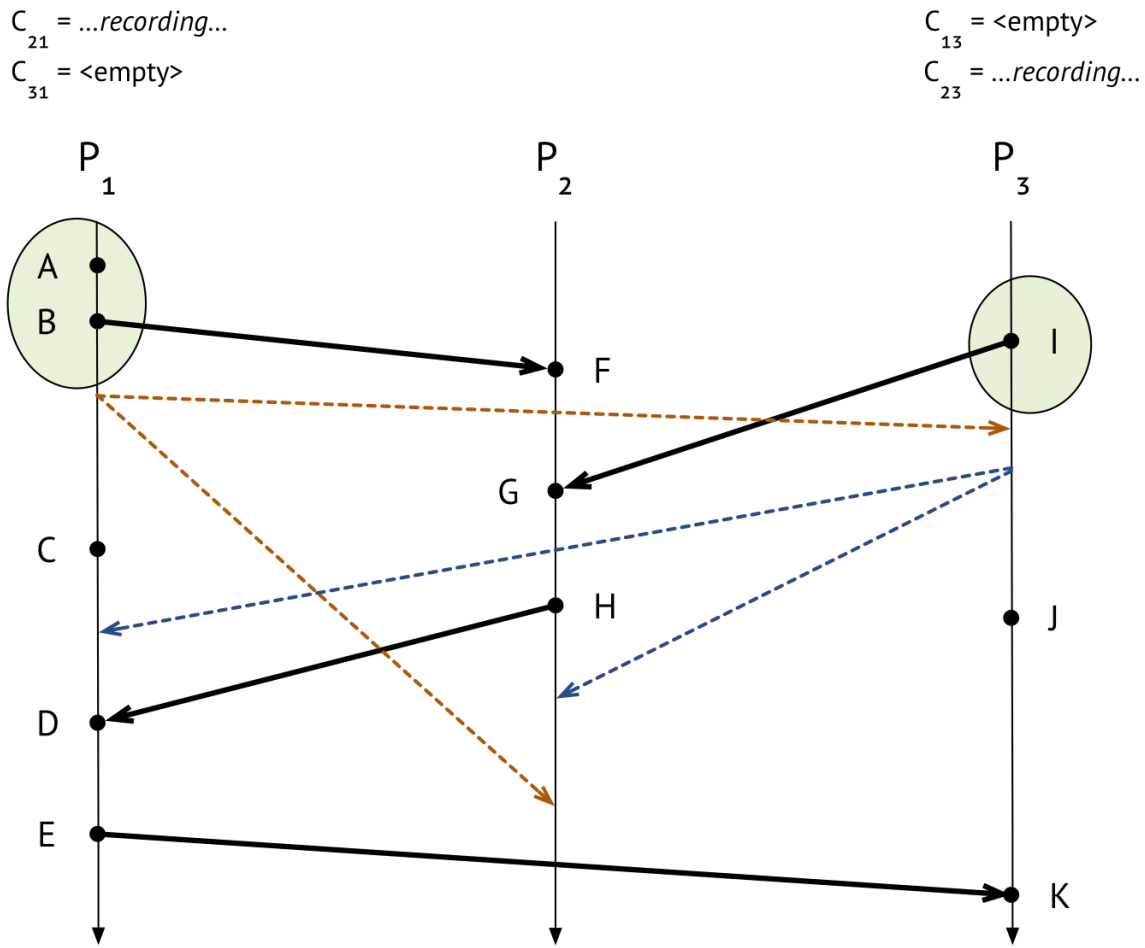
# Пример: выполнение (1)



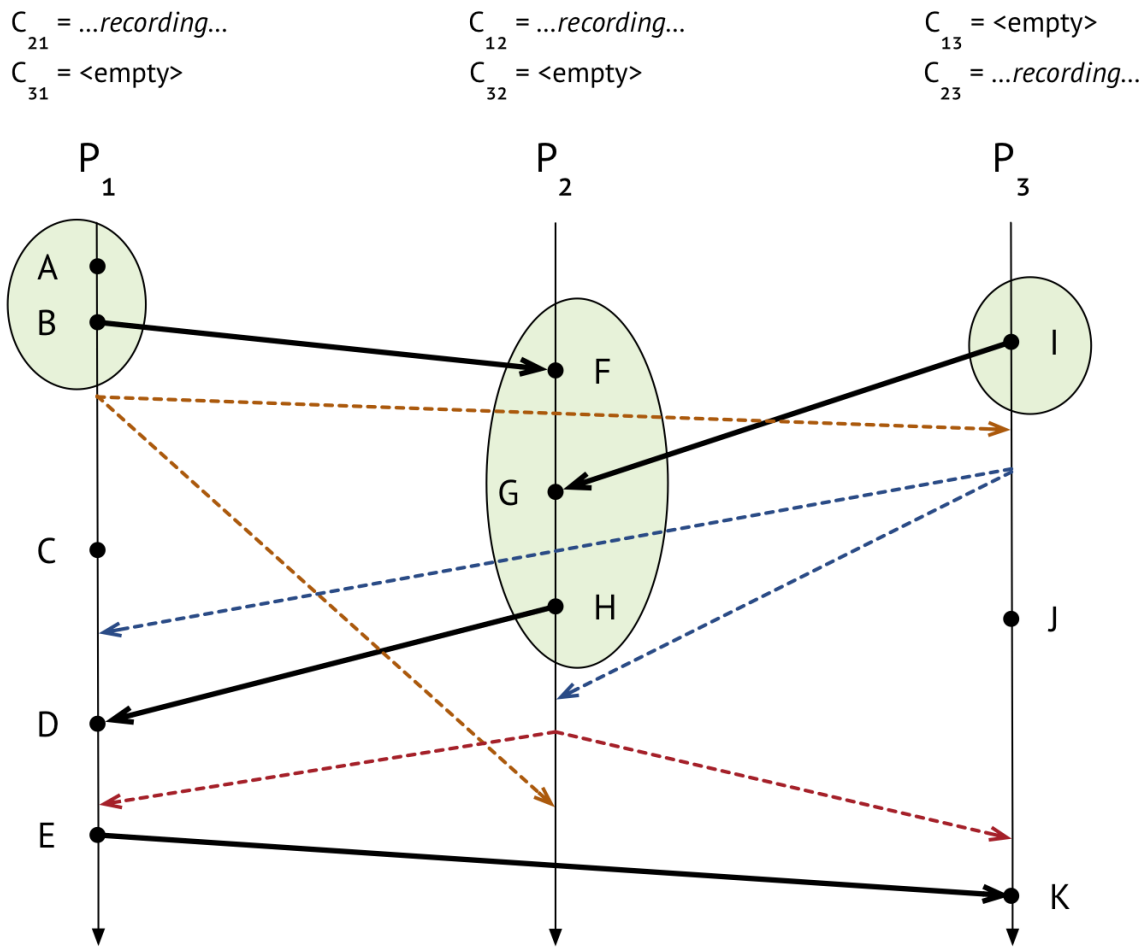
# Пример: выполнение (2)



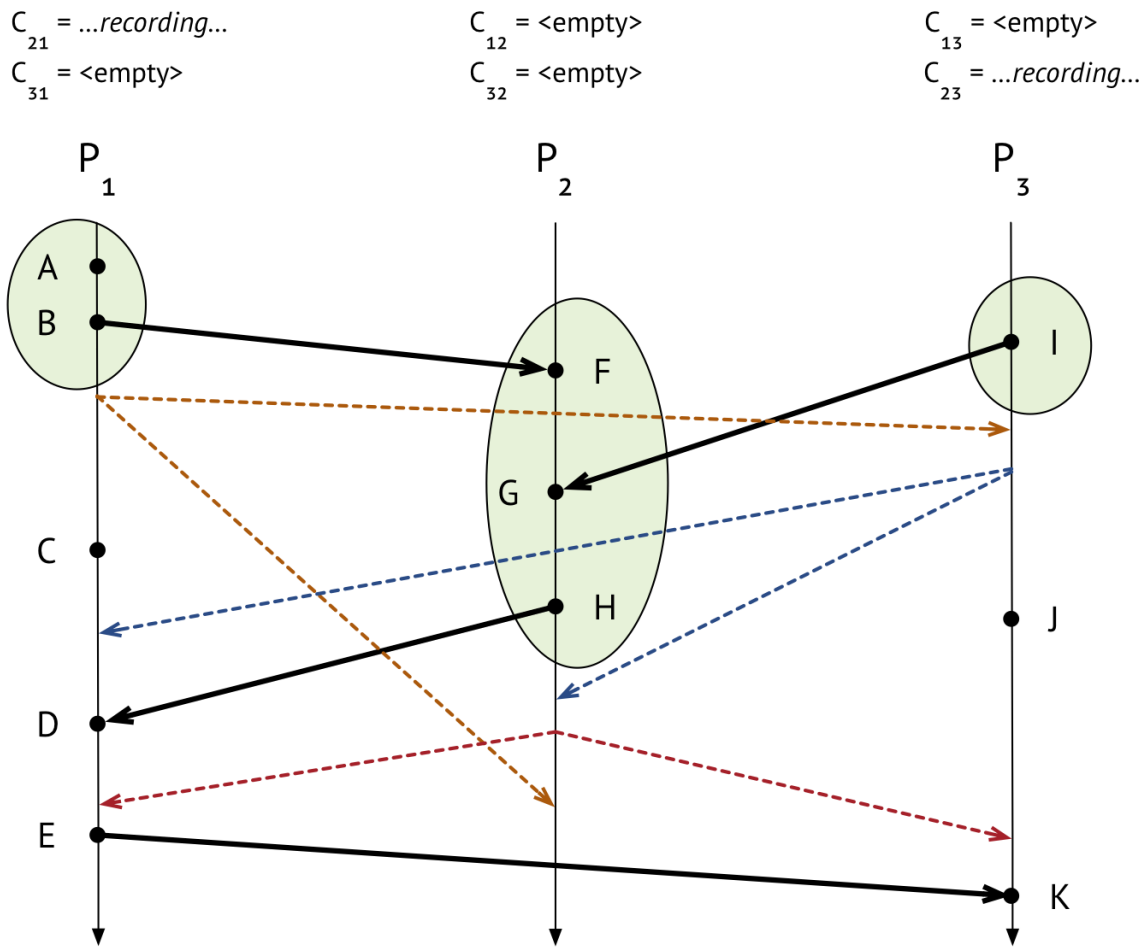
# Пример: выполнение (3)



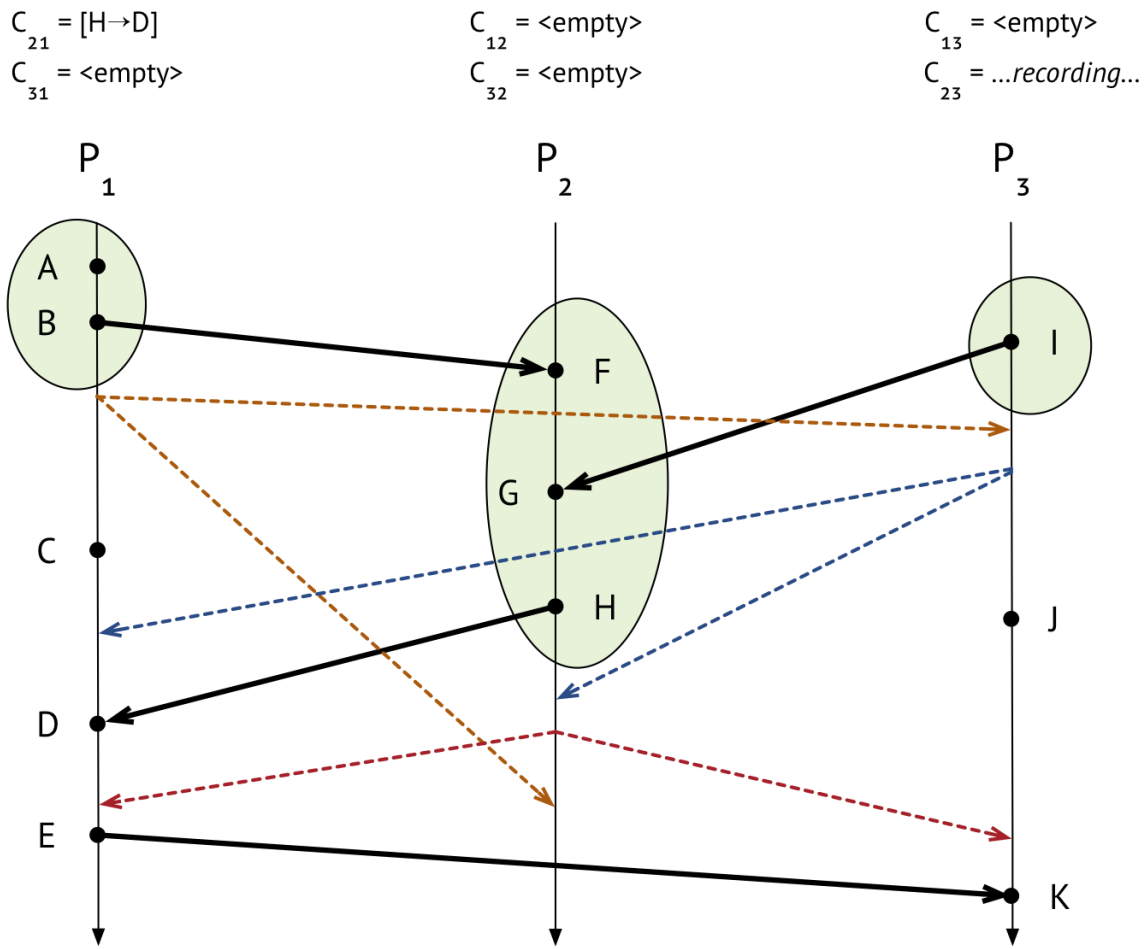
# Пример: выполнение (4)



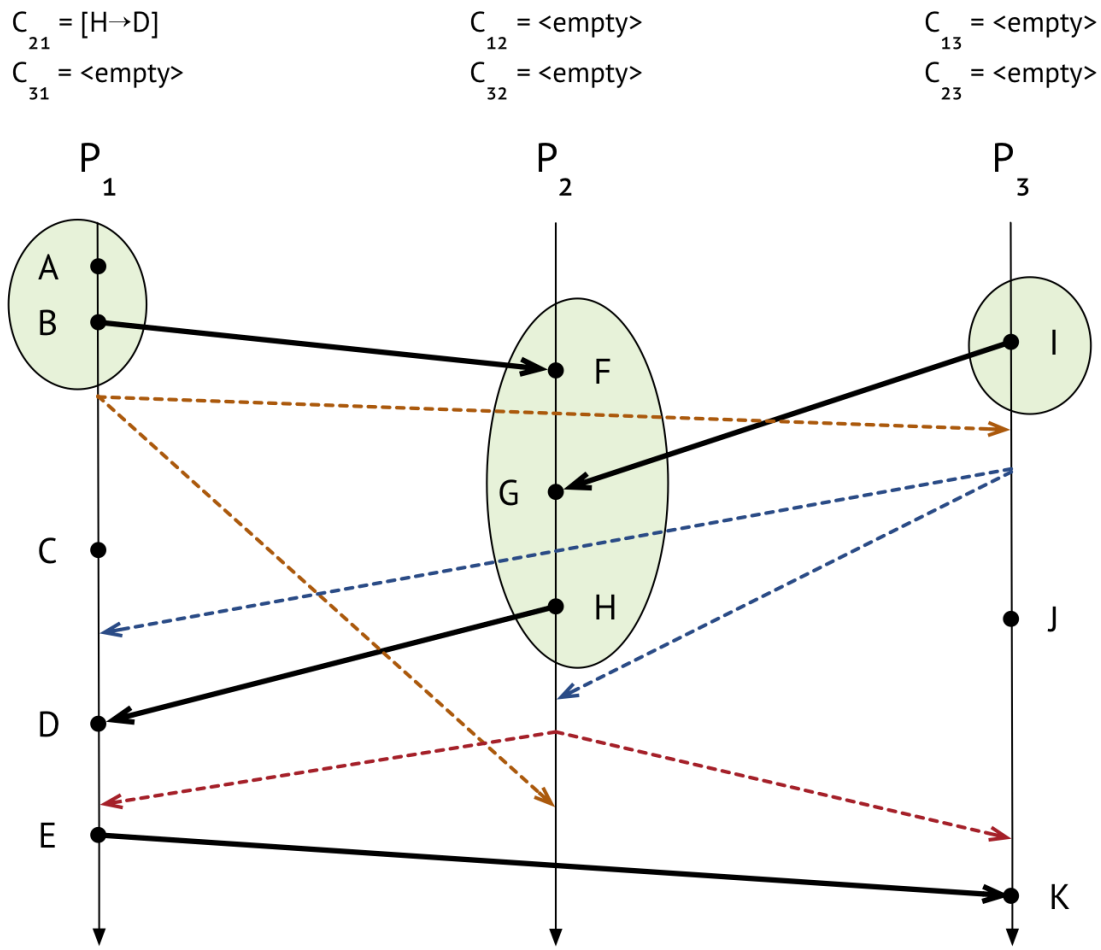
# Пример: выполнение (5)



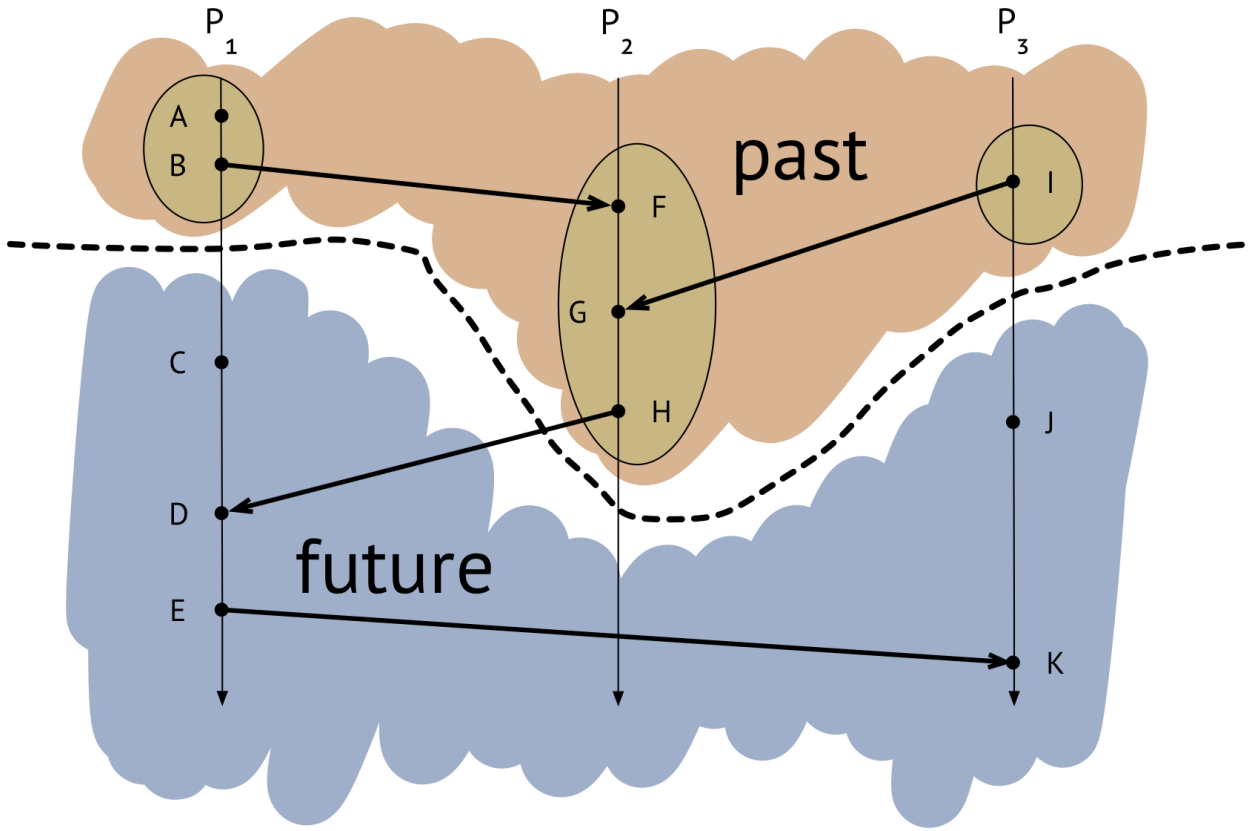
# Пример: выполнение (6)



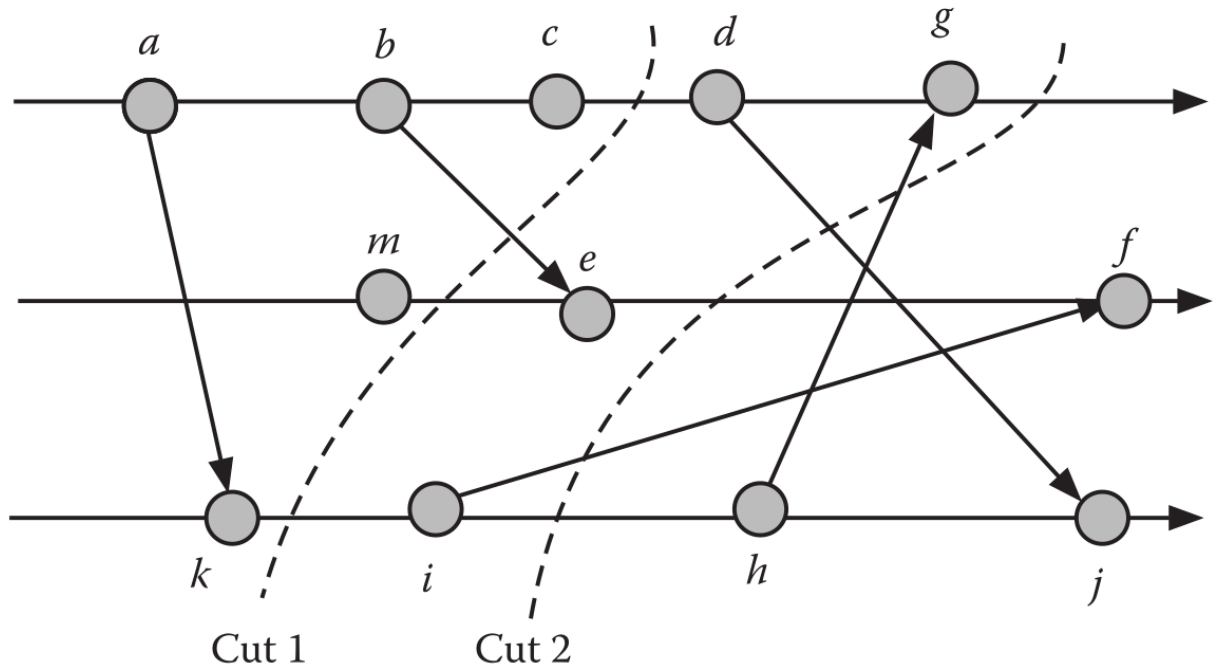
# Пример: выполнение (7)



# Пример: полученный снимок

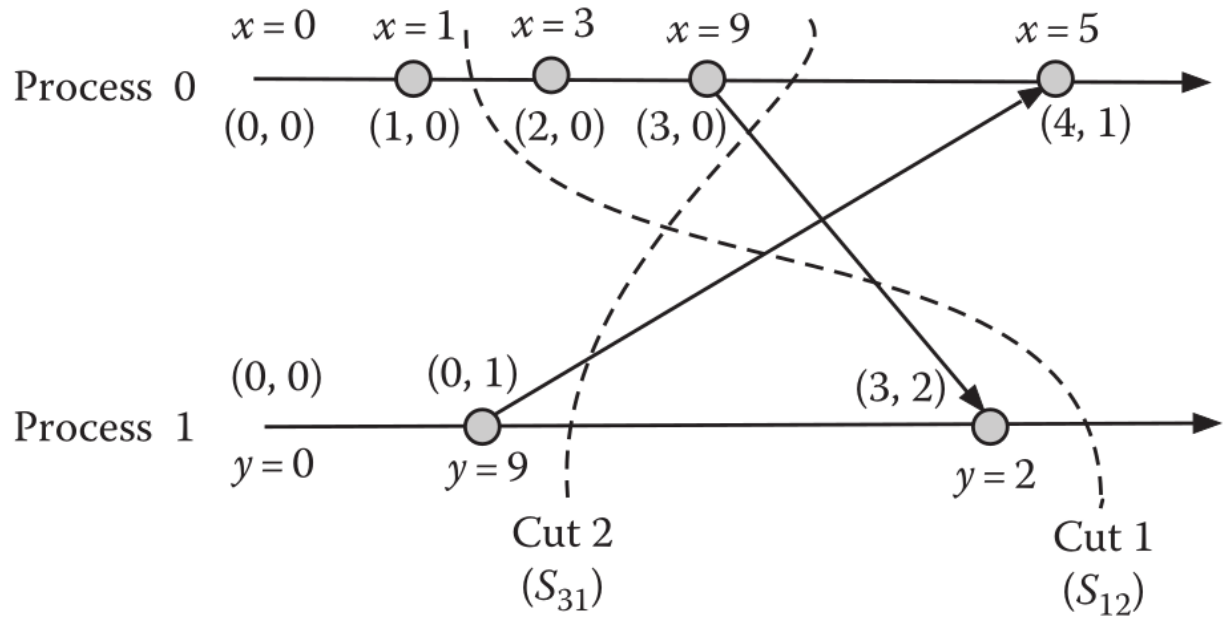


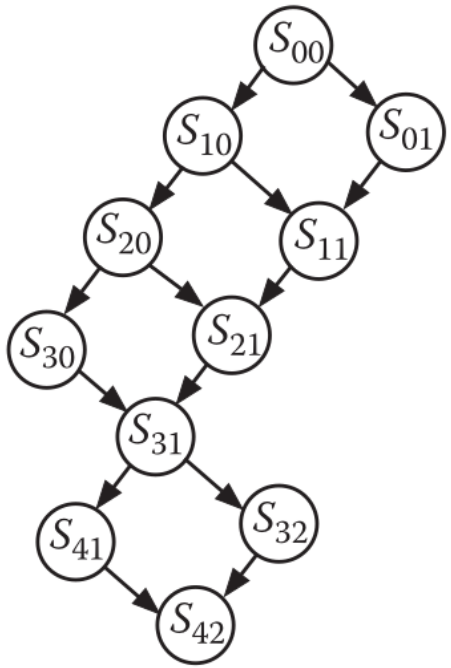
# Согласованный разрез




- Разрез (cut) делит события в системе на два множества - до (в разрезе) и после
- Согласованный разрез: если $b$ входит в разрез и $a \rightarrow b$, то $a$ также входит в разрез
- Результат алгоритма Chandy-Lamport всегда соответствует согласованному разрезу

# Возможные состояния PC



# Корректность PC

- Описывается в терминах двух классов свойств
- Liveness (живучесть)
  - гарантирует, что "со временем случится что-то хорошее"
  - преступник понесет наказание
  - детектор отказов обнаружит все отказы (полнота)
  - распределенный алгоритм завершит выполнение
- Safety (безопасность)
  - гарантирует, что "ничего плохого никогда не произойдет"
  - невинный человек никогда не будет осужден
  - детектор отказов никогда не ошибается (точность)
  - система никогда не попадет в нежелательное состояние

# Состояния РС и свойства корректности

- Безопасность для свойства $P$ в состоянии $S$
  - состояние удовлетворяет $P$
  - и все состояния $S'$, достижимые из него, также удовлетворяют $P$
- Живучесть для свойства $P$ в состоянии $S$
  - состояние $S$ удовлетворяет $P$
  - или из $S$ всегда можно попасть в такое состояние $S'$

# Использование снимка

- Снимок состояния PC может использоваться для проверки *стабильных* свойств
  - как только свойство выполняется, то оно будет выполняться всегда
- Стабильное свойство нарушения безопасности
  - возник deadlock
- Стабильное свойство живучести
  - алгоритм завершил выполнение

# Выборы лидера

# Лидер в PC

- Удобен в ситуациях, когда требуется координация процессов
  - Отслеживание участников группы
  - Рассылка в группе
  - Репликация данных
  - Взаимное исключение (блокировки)
- Требования
  - В системе не должно быть несколько лидеров
  - Все участники должны прийти к согласию о том, кто лидер
  - В случае отказа лидера должен быть выбран новый

# Алгоритм выбора лидера

- Цели
  - один из корректных процессов выбран лидером
  - все корректные процессы знают (договорились) кто новый лидер
- Модель системы
  - N процессов
  - каждый процесс имеет уникальный id (например, IP+port)
  - каналы надежные (все сообщения доставляются в конечном счёте)
  - во время выборов могут происходить отказы

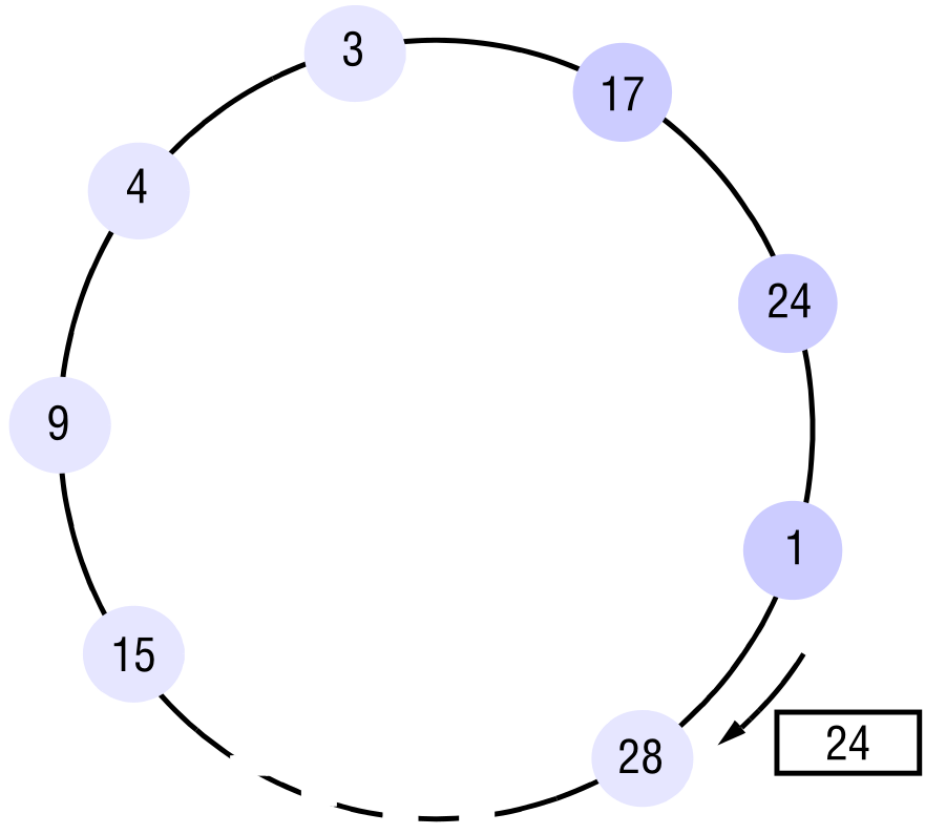
# Инициирование выборов

- Любой процесс в системе может инициировать проведение (раунд) выборов
- Процесс может одновременно инициировать не более одного раунда выборов
- Несколько процессов могут одновременно инициировать выборы
  - все они должны в итоге выбрать одного лидера
- Результат выборов не должен зависеть от того, кто явился инициатором

# Формальные требования

- В конце выполнения алгоритма (раунда выборов) лидером выбирается корректный процесс $q$ с лучшим значением некоторого атрибута
  - например, лидер имеет максимальный id, хранит самую свежую копию данных...
- Свойство безопасности (safety)
  - у любого корректного процесса $p$ переменная $elected$ содержит $q$ или $null$
- Свойство живучести (liveness)
  - раунд выборов всегда завершается
  - и у всех корректных процессов переменная $elected$ не равна $null$

# Выборы в кольце



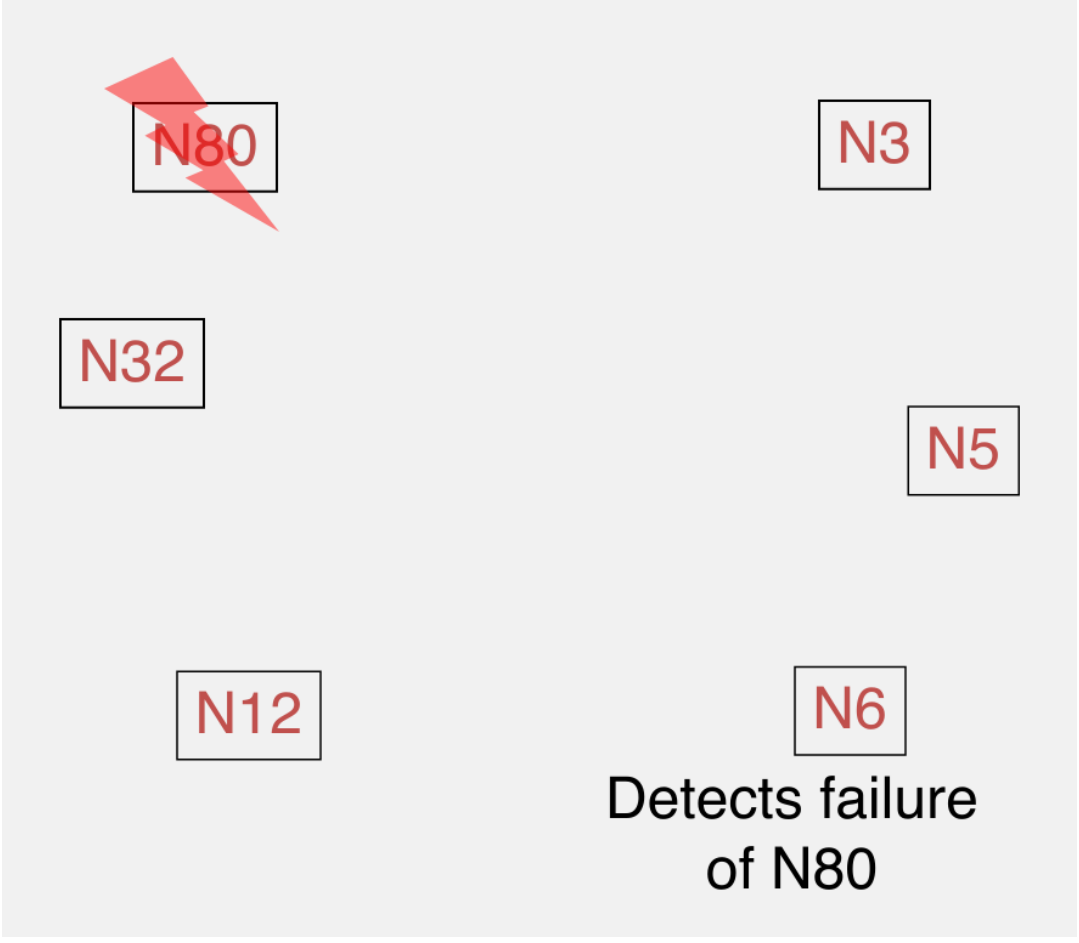
# Выборы в кольце: анализ

- Количество сообщений
  - худший случай: $3N - 1$
  - лучший случай: $2N$
- Время работы ~ количеству сообщений
- Несколько инициаторов?
- Отказы?

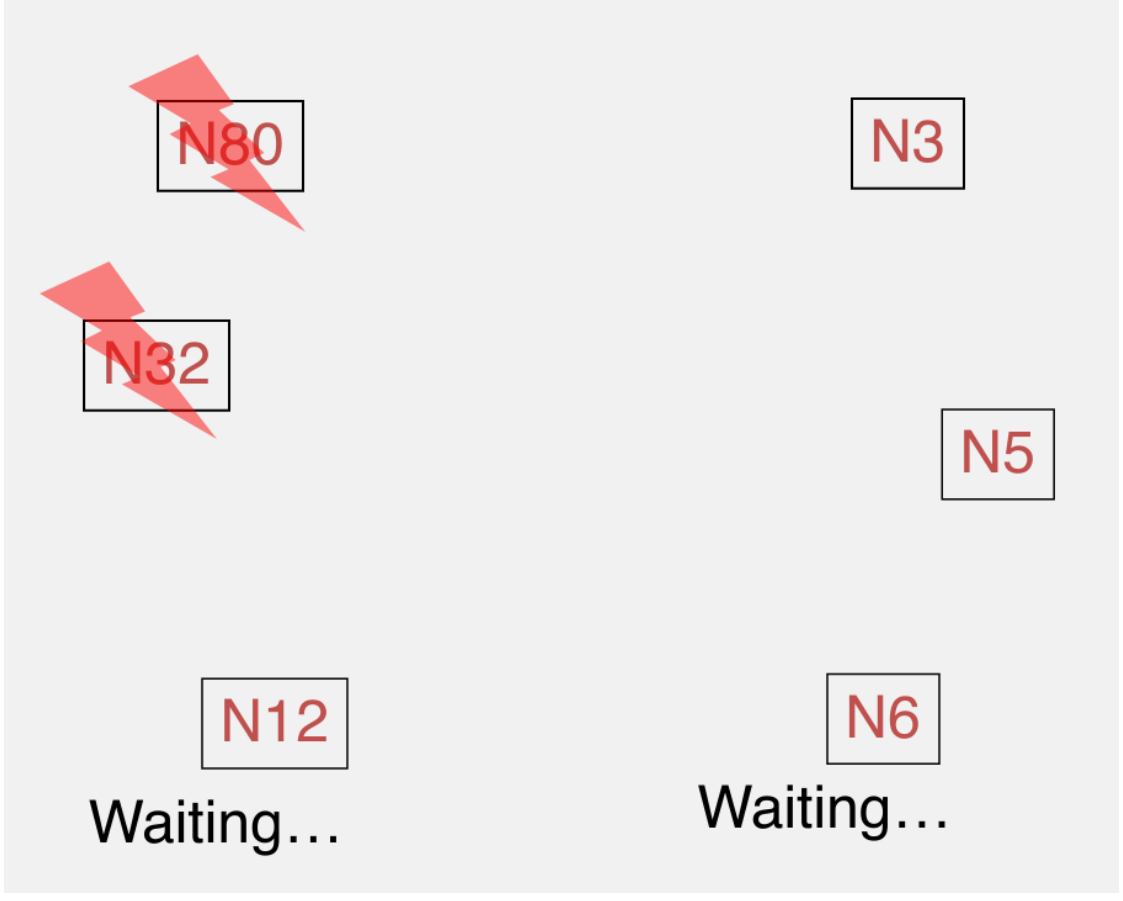
# Использование детектора отказов

- Любой процесс может обнаружить отказ процесса во время выборов и начать новый раунд
- Но в асинхронной PC нельзя одновременно обеспечить полноту и точность детектора
  - отсутствие полноты приведет к тому, что отказ не будет обнаружен (нарушение безопасности)
  - отсутствие точности приведет к тому, что процесс будет ложно признан отказавшим и выборы будут постоянно перезапускаться (нарушение живучести)

# Алгоритм Bully



# Алгоритм Bully: отказ во время выборов



# Алгоритм Bully: анализ

- Worst case
  - инициатор имеет наименьший id
  - время: передача 4 сообщений
  - количество сообщений: $O(N^2)$
- Best case
  - инициатор имеет максимальный id среди живых процессов
  - время: передача 1 сообщения
  - количество сообщений: $N - 2$
- Выбор значений таймаутов?
- В асинхронной PC живучесть не гарантируется

# Выборы лидера на основе кворума

- При обнаружении отказа лидера процесс инициирует выборы, рассылая остальным процессам запрос на голос
  - Каждый процесс голосует не более чем за одного лидера
  - Процесс становится лидером, если набирает большинство голосов
  - Победивший процесс информирует всех об исходе выборов
- После окончания выборов процессы обещают лидеру не перезапускать выборы в течение т.н. leader lease
  - Лидер должен периодически обновлять lease у каждого процесса
- Живучесть не гарантируется
  - Может возникать ситуация когда голоса разделяются
- Используется в алгоритме консенсуса Raft (следующий раз)

# Литература


- Coulouris G.F. et al. Distributed Systems: Concepts and Design. Pearson, 2011 (разделы 14.5, 15.3)
- van Steen M., Tanenbaum A.S. Distributed Systems: Principles and Paradigms. Pearson, 2017. (глава 8, Checkpointing)