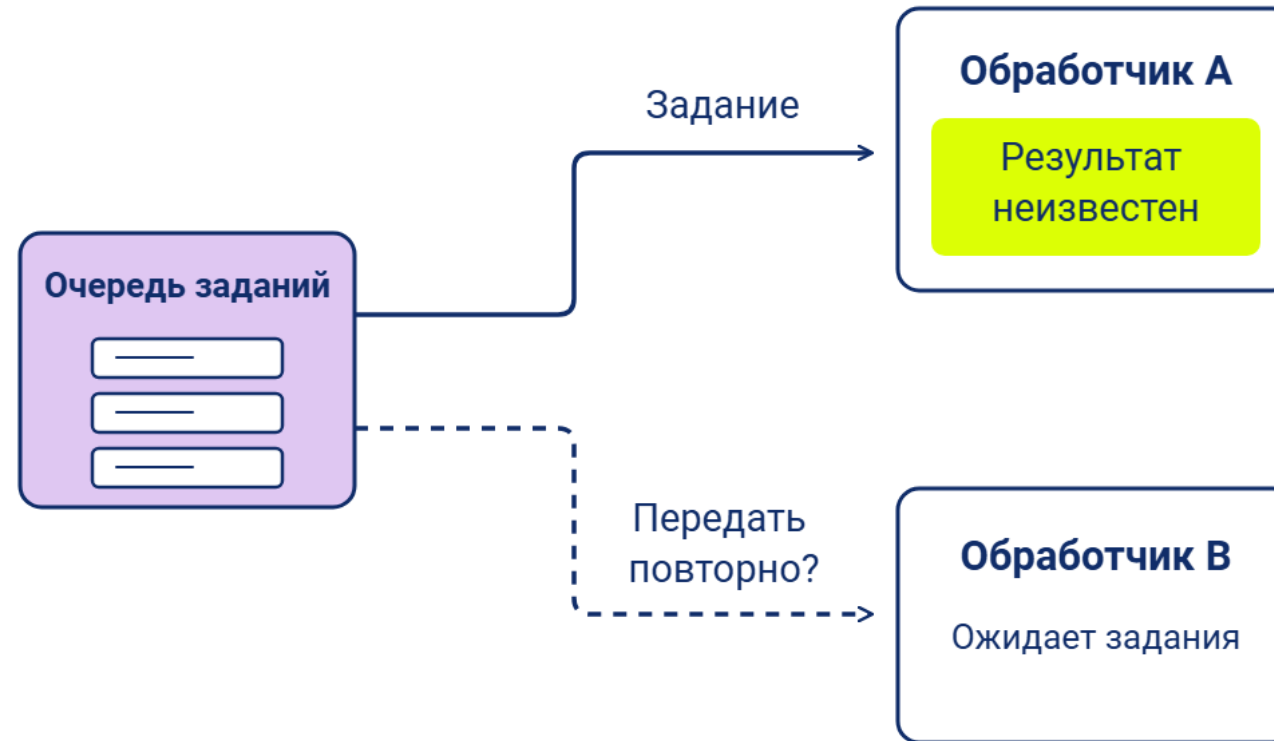


6. Обнаружение отказов

Сухорослов Олег Викторович

12.10.2026

Обработчик не отвечает



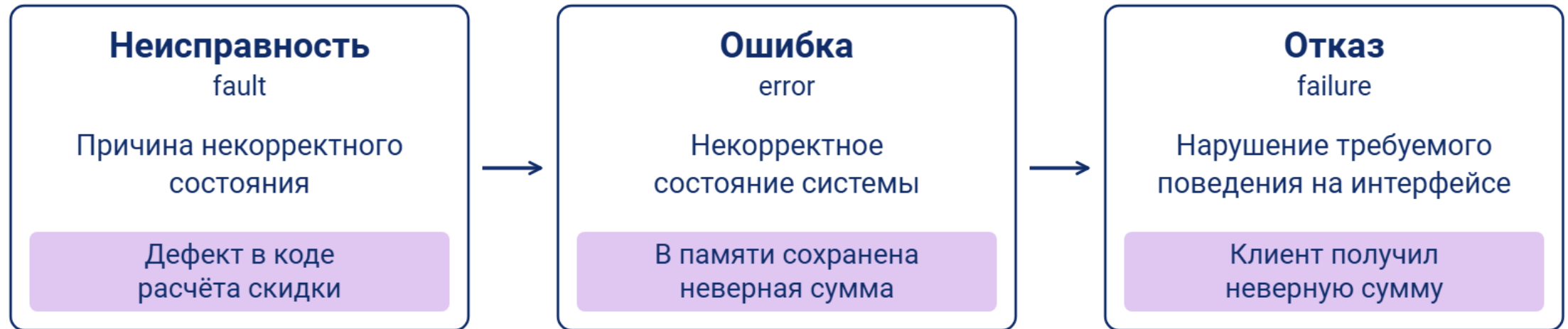
Обработчик **А** получил задание из очереди. Сообщение о завершении не пришло.

Продолжать ждать или передать задание обработчику В?

План лекции

- Модели отказов и синхронности.
- Детекторы отказов: свойства, ping и heartbeat.
- Масштабирование: all-to-all и gossip.
- SWIM: обнаружение отказов и состав группы.
- Адаптивные детекторы и действия при подозрении.

Неисправность, ошибка и отказ



Не всякая неисправность приводит к отказу системы.

Модели отказов процесса

Модель отказов задаёт, как процесс может отклоняться от протокола.

Модель	Поведение процесса
Остановка (crash)	Прекращает выполнение
Пропуски (omission)	Не отправляет или не принимает часть сообщений
Нарушение сроков (timing)	Выполняет действия вне допустимых временных границ
Произвольное поведение (Byzantine)	Может нарушать протокол любым образом, в том числе отправлять противоречивые сообщения

В этой лекции сосредоточимся на остановках процессов.

Остановка и восстановление процесса

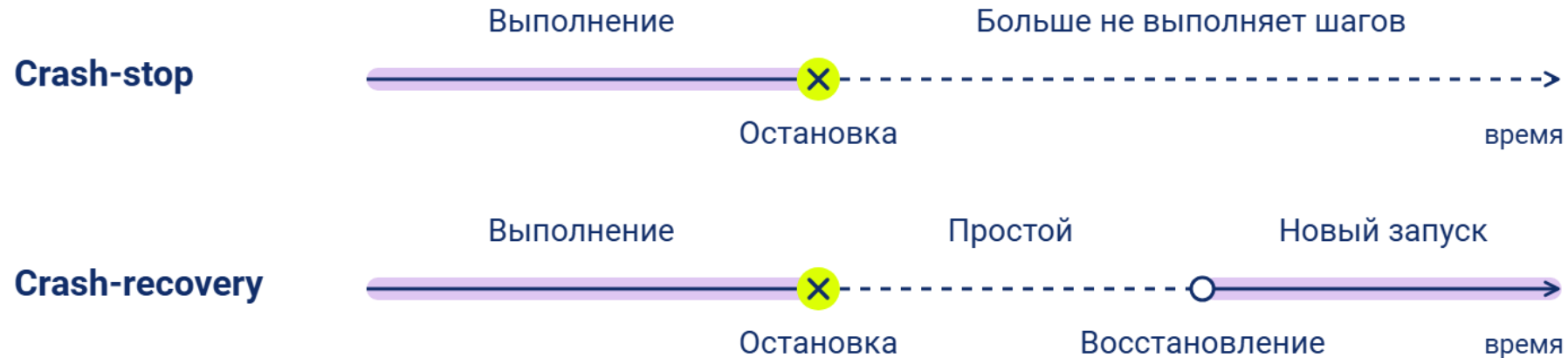
Crash-stop – остановка без восстановления

После отказа процесс больше не выполняет шагов.

Crash-recovery – остановка с возможностью восстановления

Процесс может возобновить работу после перезапуска.

Нужно учитывать потерю состояния в памяти и повторное выполнение действий.



Основная модель далее – crash-stop.

Модели синхронности

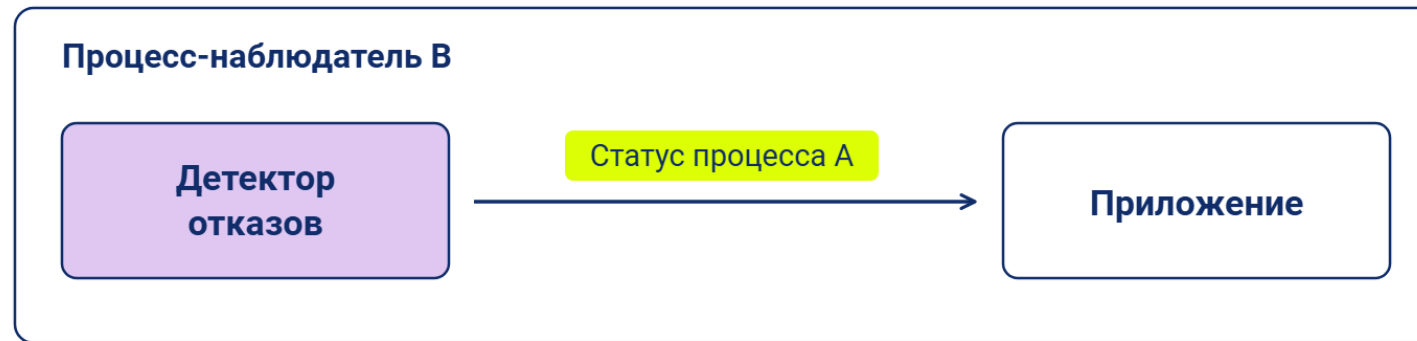
Учитываем время передачи сообщений и выполнения шагов процесса.

Модель	Предположения о времени
Синхронная	Известны верхние границы задержек, которые выполняются всегда
Асинхронная	Гарантированных верхних границ задержек нет
Частично синхронная	После некоторого неизвестного момента начинают постоянно выполняться известные границы задержек

Гарантии канала задаются на уровне сообщений алгоритма.

Что сообщает детектор отказов

Детектор отказов – локальный модуль, который сообщает, какие процессы подозреваются в отказе.



Статус наблюдаемого процесса:

- **Не подозревается** – детектор сейчас не предполагает отказ.
- **Подозревается** – детектор предполагает, что процесс остановился.

Подозрение может быть ошибочным и впоследствии снято.

Представления разных наблюдателей могут различаться.

Решение о реакции принимает приложение.

Свойства детектора отказов

Сильная полнота (strong completeness)

Каждый остановившийся процесс *в конце концов* начинает постоянно подозреваться всеми *корректными* процессами.

Сильная точность (strong accuracy)

Ни один процесс не подозревается до своей остановки.

Идеальный детектор (perfect failure detector)

Обладает сильной полнотой и сильной точностью.

Обнаружение отказа может происходить с задержкой.

Надёжные и ненадёжные детекторы

В heartbeat-схемах полноту обнаружения отказа обеспечивает таймаут.

Сложнее обеспечить точность, то есть отсутствие ложных подозрений.

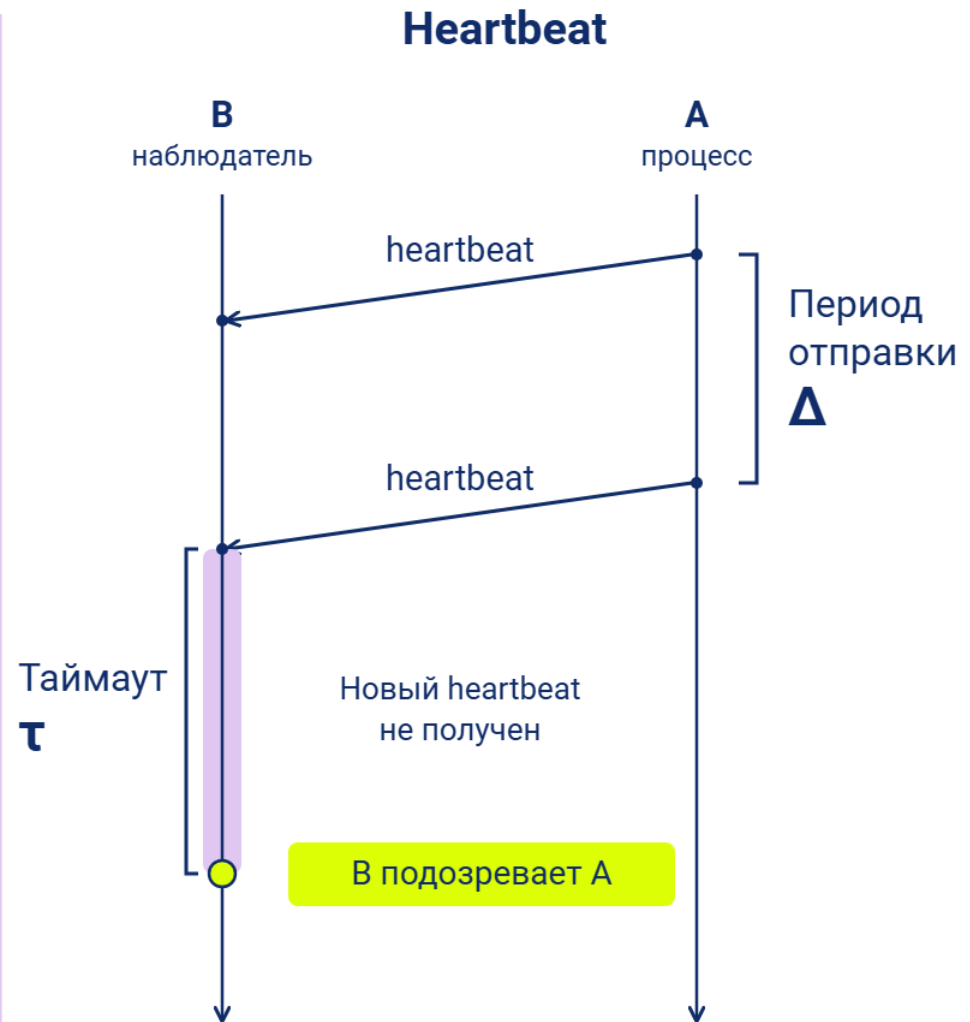
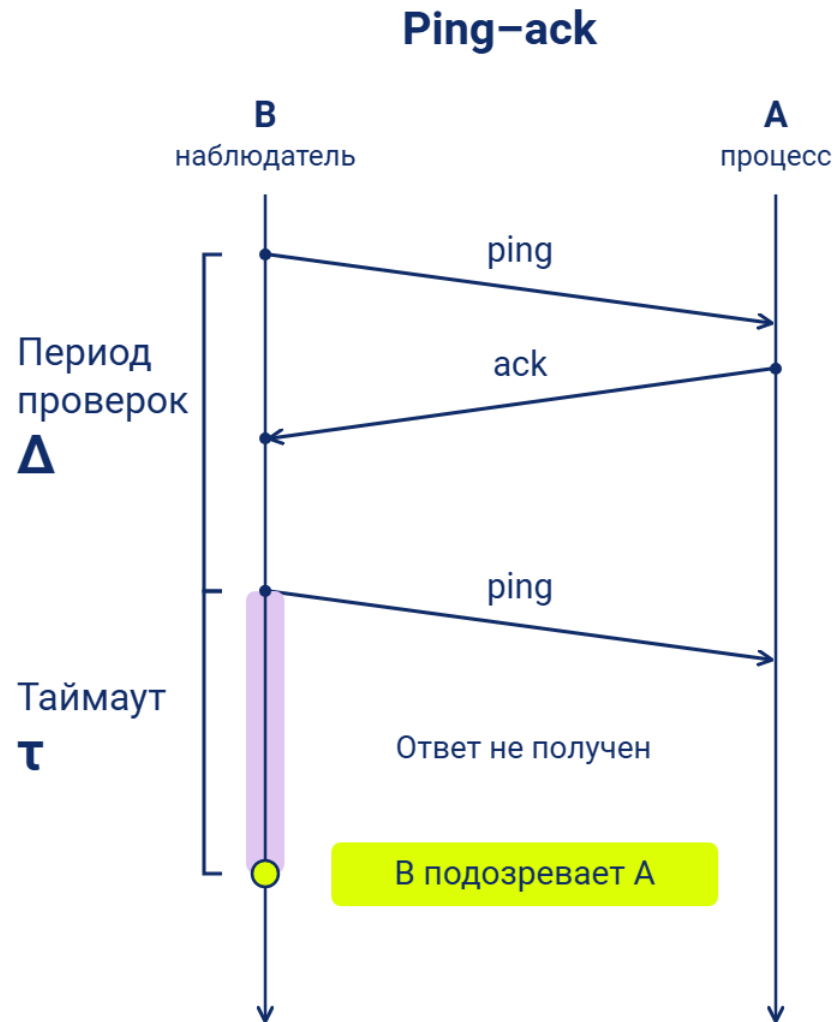
Детектор	Сообщение об отказе
Надёжный	Failed – процесс действительно остановлен
Ненадёжный	Suspected – процесс может быть остановлен, но возможна ошибка

От обоих требуем полноты: каждый отказ должен быть в конце концов обнаружен.

На практике важна и скорость: сколько времени проходит от отказа до его обнаружения.

Можно ли построить быстрый и надёжный детектор?

Простые детекторы: ping и heartbeat



Выбор периода и таймаута

Период отправки Δ – как часто получаем новые сведения

- Короткий период: чаще проверки, больше служебного трафика.
- Длинный период: дольше ждать следующей проверки или heartbeat.

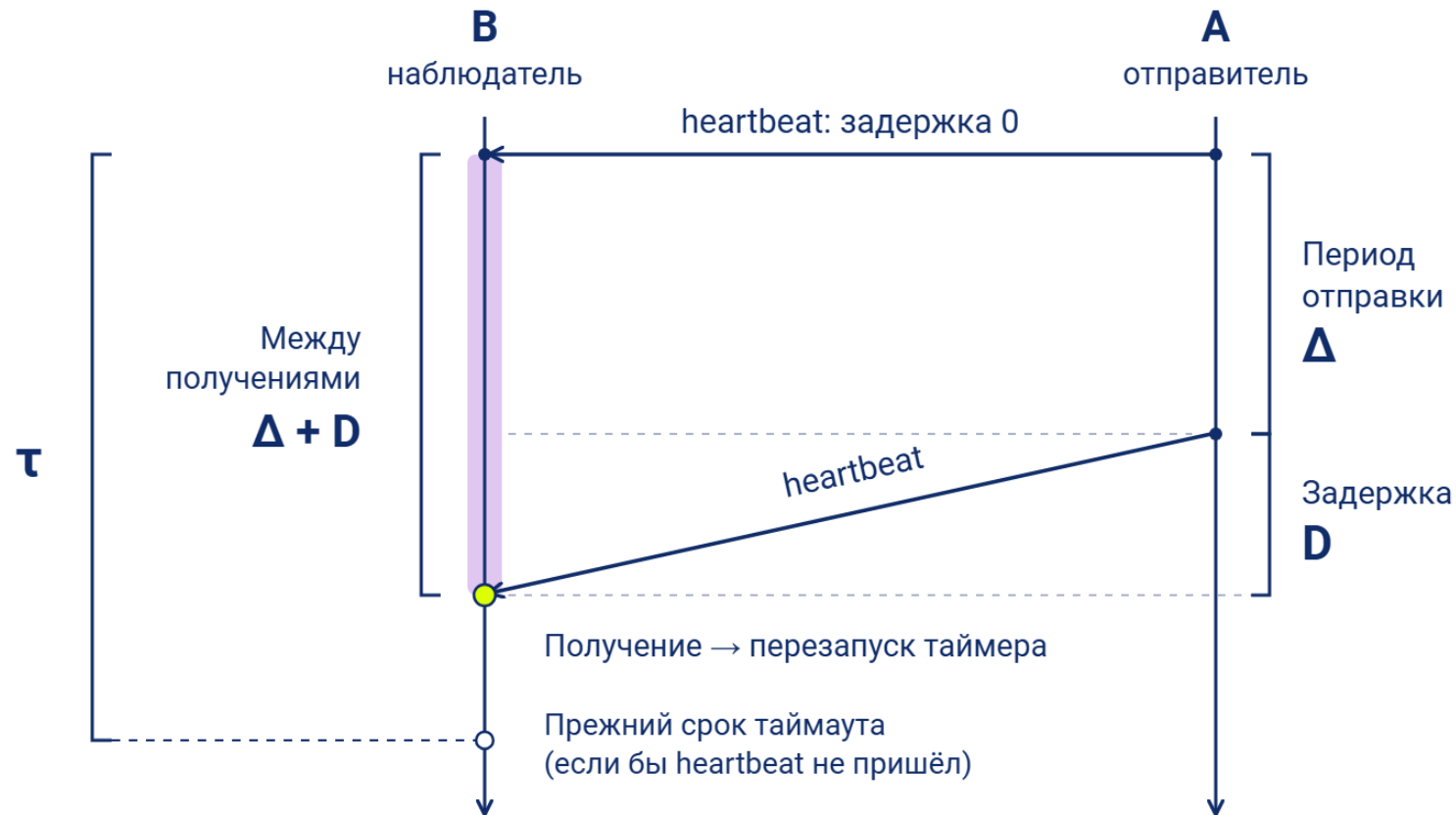
Таймаут τ – когда начинаем подозревать отказ

- Короткий таймаут: быстрее возникает подозрение, выше риск ошибки.
- Длинный таймаут: реже ложные подозрения, позже обнаружение отказа.

Выбор зависит от допустимой задержки обнаружения и цены ложного подозрения.

Когда таймаут не даёт ложных подозрений

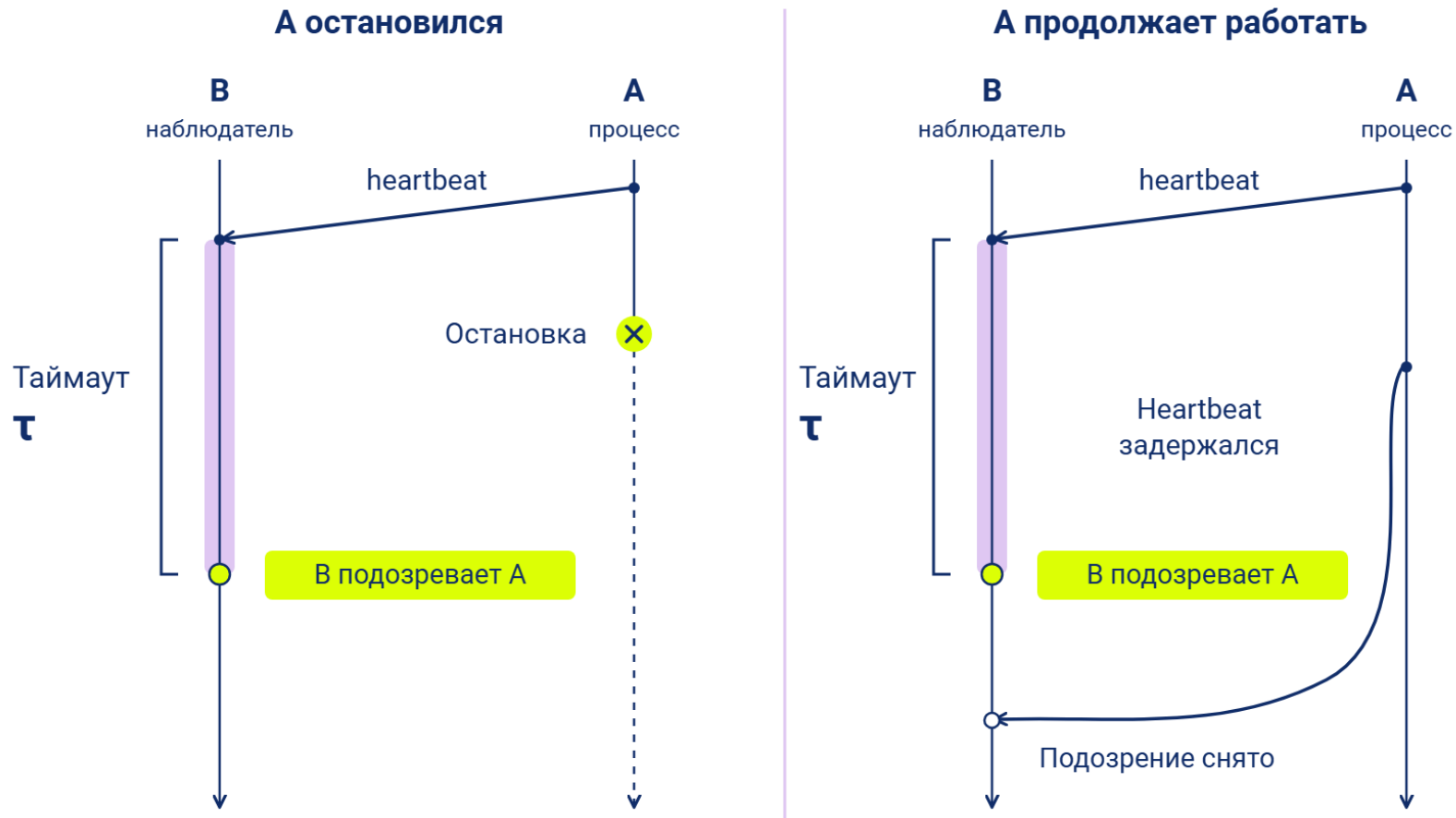
Рассмотрим heartbeat в **синхронной системе** с надёжными каналами.



Интервал между получениями heartbeat не превышает $\Delta + D$. Выбираем таймаут $\tau > \Delta + D$.

Отказ или задержка?

В асинхронной системе к моменту истечения таймута эти сценарии неразличимы для В.



Обмена сообщениями недостаточно для реализации идеального детектора.

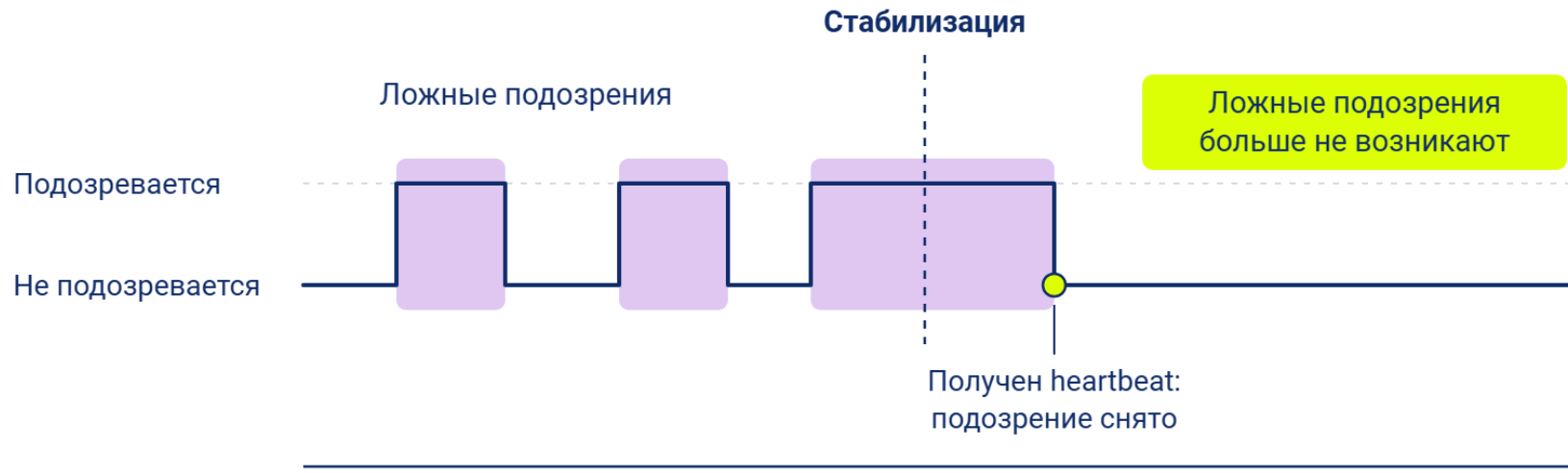
Точность в частично синхронной системе

После некоторого неизвестного момента соблюдаются границы Δ и D .

При надёжной доставке и $\tau > \Delta + D$ ложные подозрения в конце концов прекращаются.

Процесс А продолжает работать

Мнение детектора у наблюдателя В:



Идеальный в конечном счёте детектор (eventually perfect).

Сильная полнота + сильная точность в конечном счёте.

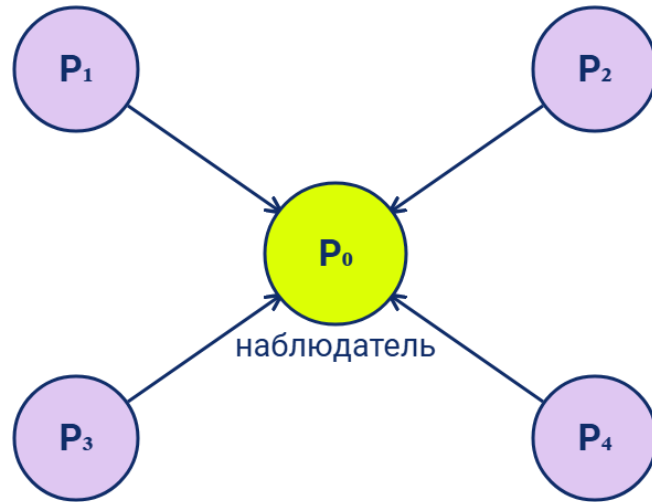
Как сравнивать детекторы?

Характеристика	Что оцениваем
Скорость	Время до первого подозрения и до получения информации другими участниками
Ложные подозрения	Как часто возникают и как долго сохраняются
Нагрузка на процесс	Отправка и приём сообщений, обработка данных, память
Нагрузка на сеть	Число сообщений и объём данных в единицу времени
Масштабируемость	Как растут затраты с размером группы, где возникают узкие места

Сравниваем при одинаковых условиях и требованиях к качеству обнаружения.

Кто наблюдает за процессами?

Центральный наблюдатель

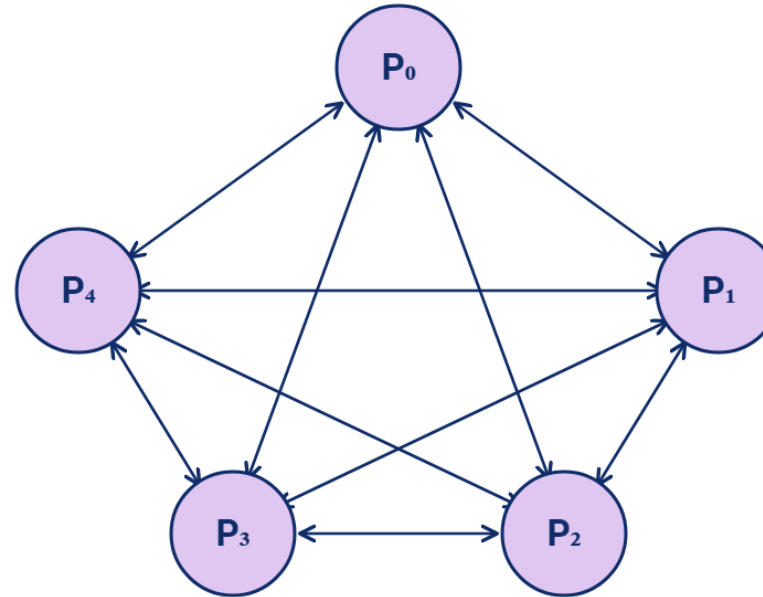


$N - 1$

heartbeat за период

Нагрузка сосредоточена у наблюдателя.
Его отказ требует отдельной обработки.

All-to-all



$N(N - 1)$

heartbeat за период

Каждый наблюдает за всеми.
Общее число сообщений растёт квадратично.

Как уменьшить число проверок, сохранив обнаружение отказов?

Обнаружение и распространение

1. Обнаружение

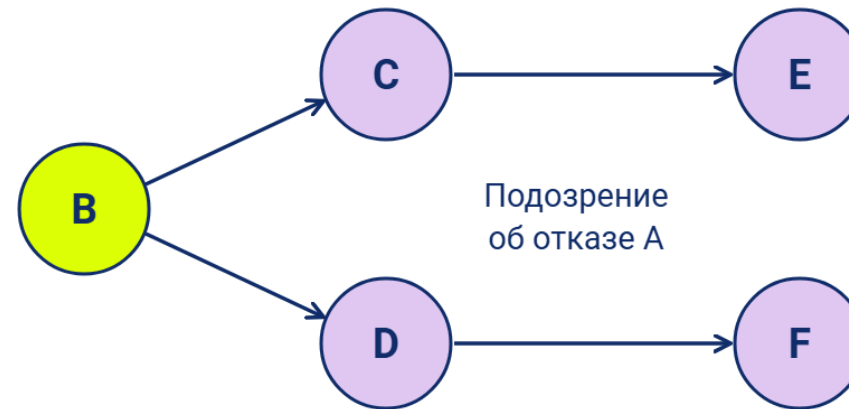


Ответ не получен за таймаут

В подозревает А

Подозрение возникает
у одного наблюдателя.

2. Распространение



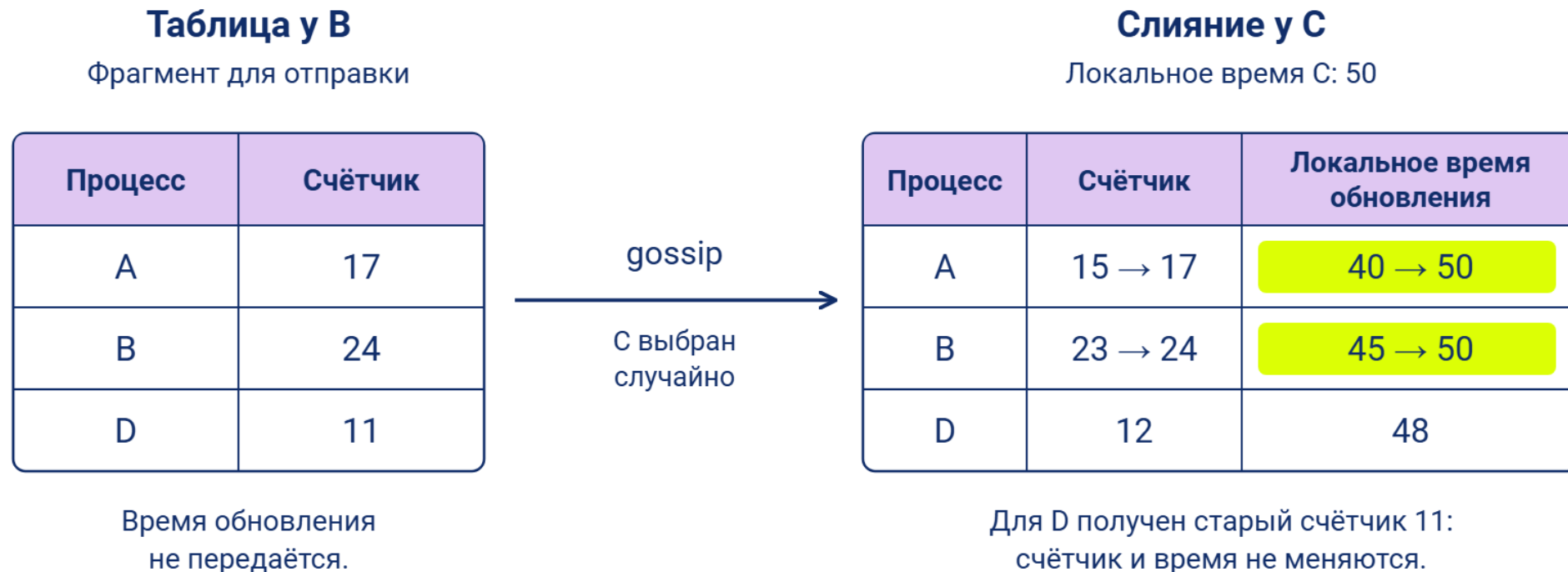
Подозрение
об отказе А

Другие участники
узнают о подозрении.

Время первого обнаружения и время распространения оцениваются отдельно.

Heartbeat через gossip

- Каждый процесс увеличивает свой счётчик и отправляет таблицу случайному участнику.
- При слиянии для каждого процесса сохраняется максимальный счётчик.
- Если счётчик не растёт дольше таймаута – возникает подозрение.



Почему нельзя сразу удалить запись

1. С остановился

Таблица у А

С	42
---	----

Оба помнят последний счётчик С

Таблица у В

С	42
---	----

 Новых heartbeat нет

2. А удаляет запись

Таблица у А

Записи о С нет

Таймаут у А истёк.
У В — ещё нет.

Таблица у В

С	42
---	----

А забыл счётчик С

3. В присылает старую запись

Таблица у А

С	42
---	----

Таблица у В

С	42
---	----

↑
gossip
С: 42

А добавляет С обратно
и запускает таймер заново

После подозрения запись некоторое время сохраняют.
Старый счётчик не должен перезапускать таймер.

Стоимость: сообщения и объём данных

Одинаковый период отправки. В gossip каждый процесс отправляет полную таблицу одному случайному участнику. Размер записи постоянный, заголовки сообщений не учитываем.

Для всей группы	All-to-all	Gossip с полной таблицей
Сообщений за период	$N(N - 1)$	N
Записей в сообщении	1	N
Переданных записей за период	$N(N - 1)$	N^2

Число сообщений уменьшается, но объём данных за период остаётся порядка N^2 .

Стоимость при одинаковой скорости обнаружения

Оценка для целевого времени обнаружения отказа порядка T .

Характеристика	Каждый со всеми	Gossip с полной таблицей
Период отправки	порядка T	порядка $T / \log N$
Размер сообщения	$O(1)$ записей	$O(N)$ записей
Отправок одним процессом в единицу времени	$O(N / T)$	$O(\log N / T)$
Записей, отправляемых одним процессом в единицу времени	$O(N / T)$	$O(N \log N / T)$
Сообщений в единицу времени, всего	$O(N^2 / T)$	$O(N \log N / T)$
Записей в единицу времени, всего	$O(N^2 / T)$	$O(N^2 \log N / T)$

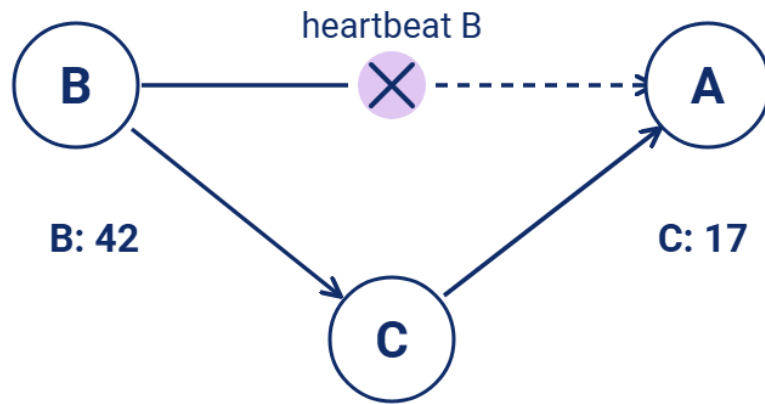
Gossip сокращает число обменов на процесс, но увеличивает объём передаваемых и обрабатываемых данных.

Чем помогает gossip?

Все процессы работают. Прямая передача $B \rightarrow A$ не проходит.

All-to-all

Каждый передаёт свой heartbeat



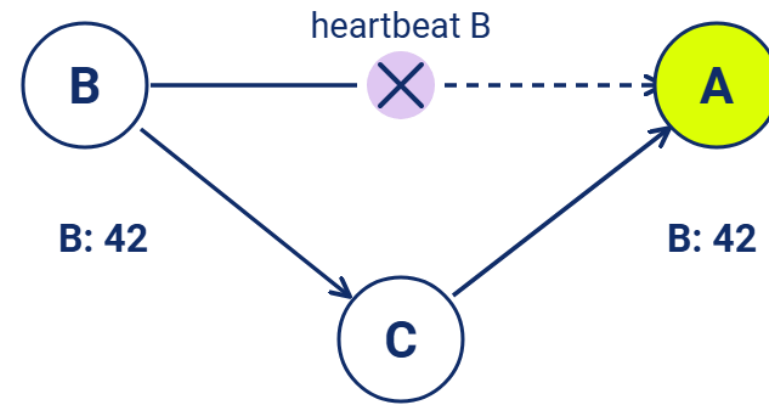
С передаёт только свой счётчик

У А истекает таймаут для В

А подозревает работающий В

Gossip

Передаётся таблица счётчиков



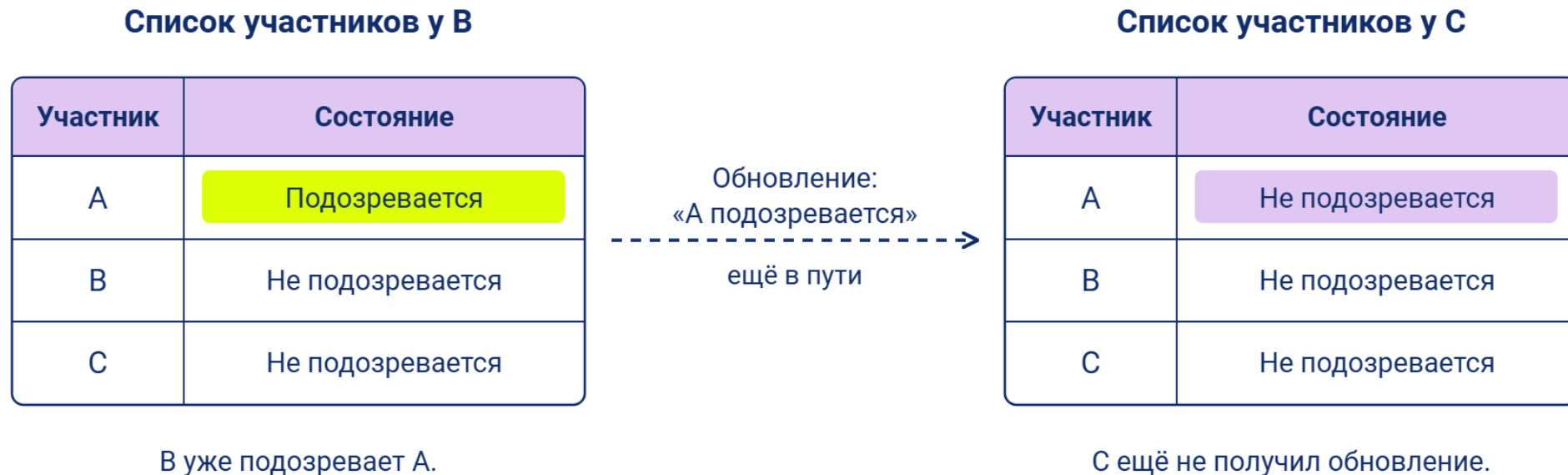
С передаёт также счётчик В

Свежие счётчики В доходят через С

А не подозревает В

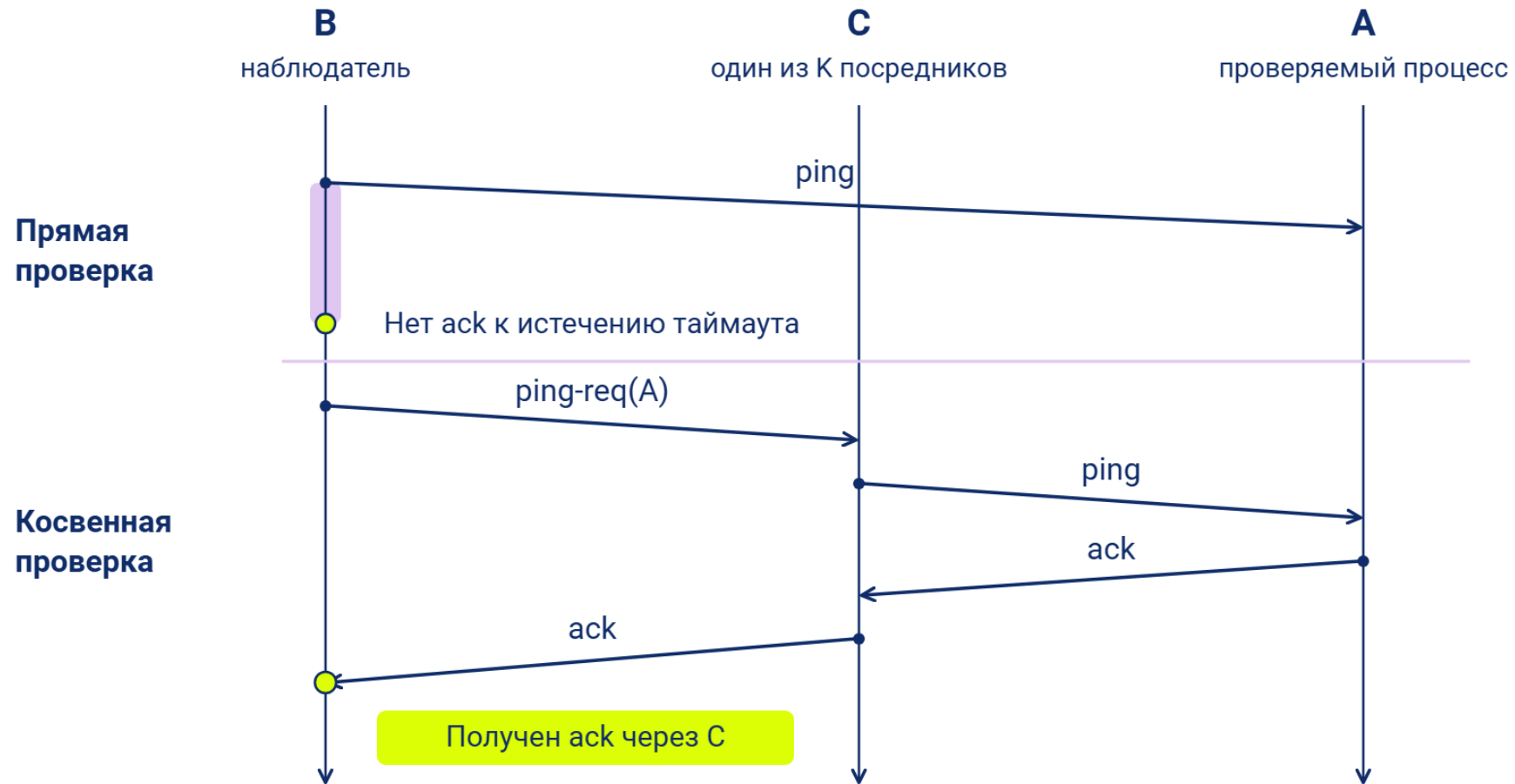
SWIM: состав группы и обнаружение отказов

- Каждый участник хранит локальный список процессов и их состояний.
- В каждом периоде участник выбирает случайного участника для проверки.
- Распространяет отдельные обновления через gossip.



Представления участников о группе могут временно различаться.

SWIM: прямая и косвенная проверка



Если подтверждение не получено ни напрямую, ни через **K** случайных посредников до окончания проверки, **В** **начинает подозревать А**.

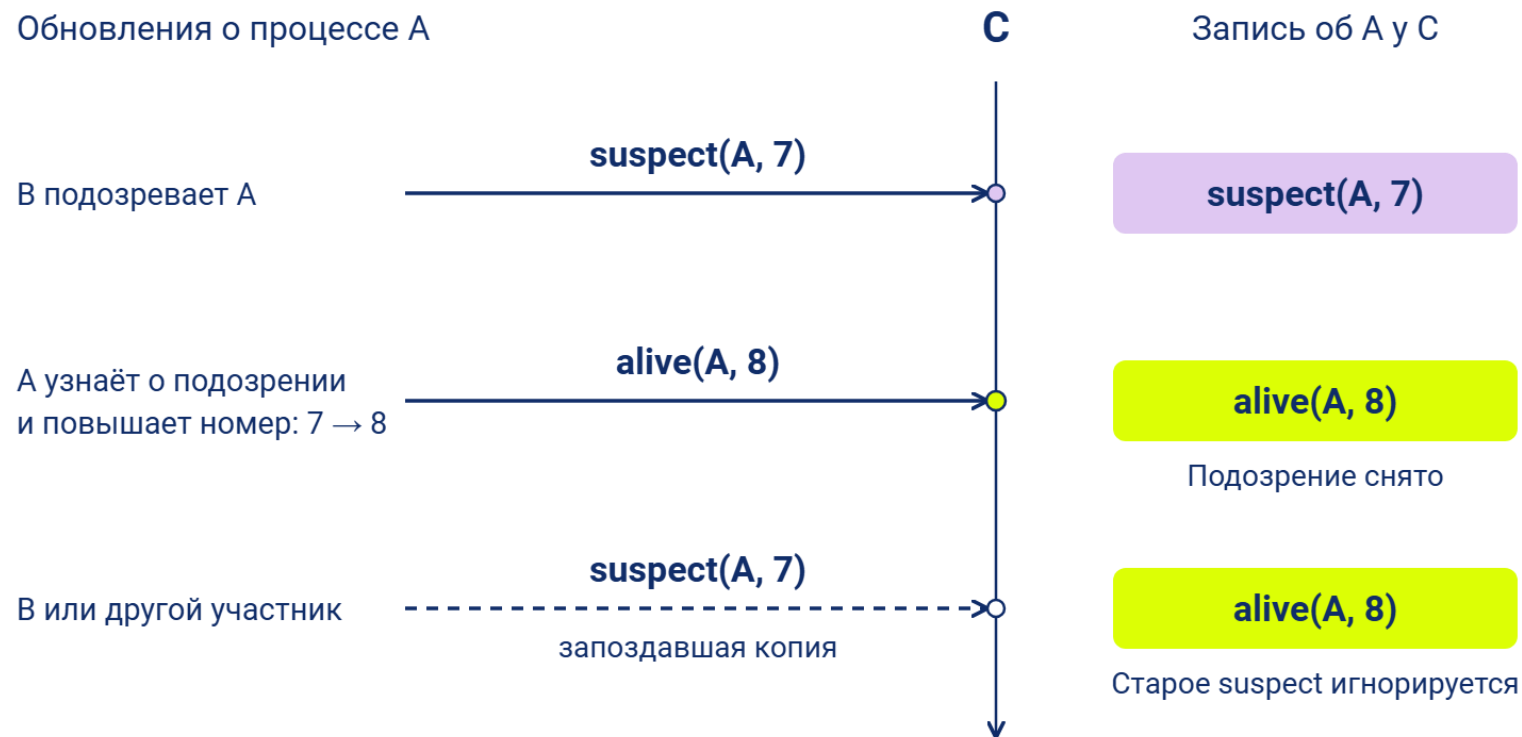
SWIM: подозрение перед исключением



Период подозрения даёт время опровергнуть ошибку, но откладывает окончательное исключение из группы.

SWIM: как отличить новое alive от старого

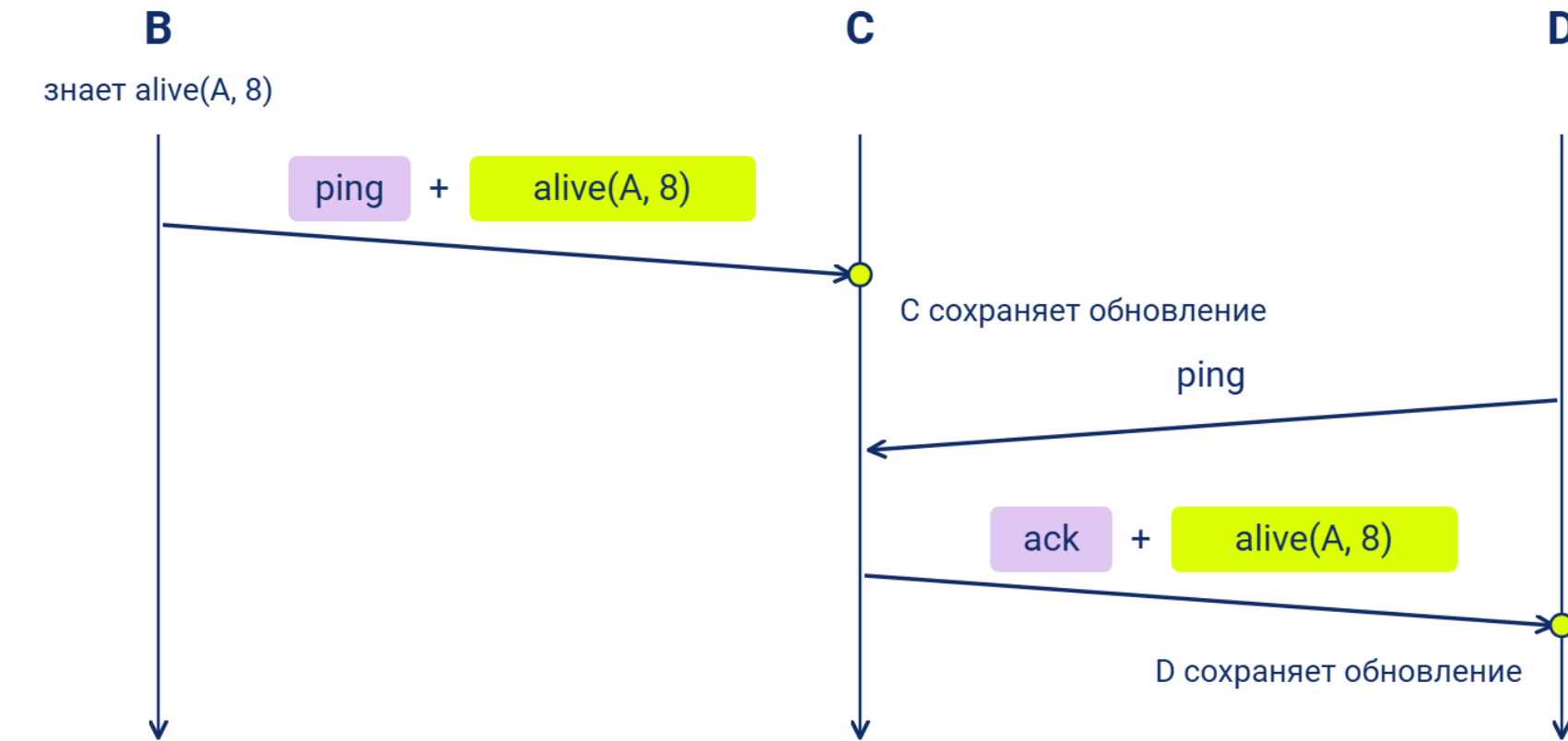
Номер **incarnation** задаёт версию состояния процесса.
Увеличивать свой номер может только сам процесс.



Для `alive` и `suspect` побеждает больший номер. При равных номерах `suspect` имеет приоритет.

SWIM: обновления вместе с ping и ack

К ping и ack добавляют ограниченное число обновлений о составе группы (т.н. piggybacking).



Показаны только сообщения, участвующие в примере распространения.

SWIM: стоимость и ограничения

При одинаковом периоде (T), фиксированном числе помощников и ограниченном размере piggyback.

За один период	Gossip с полной таблицей	SWIM
Сообщений на процесс, в среднем	$O(1)$	$O(1)$
Размер сообщения, в записях	$O(N)$	$O(1)$
Объём обмена на процесс, в среднем	$O(N)$	$O(1)$
Объём обмена во всей системе	$O(N^2)$	$O(N)$

SWIM распространяет изменения состава группы вместо полной таблицы счётчиков.

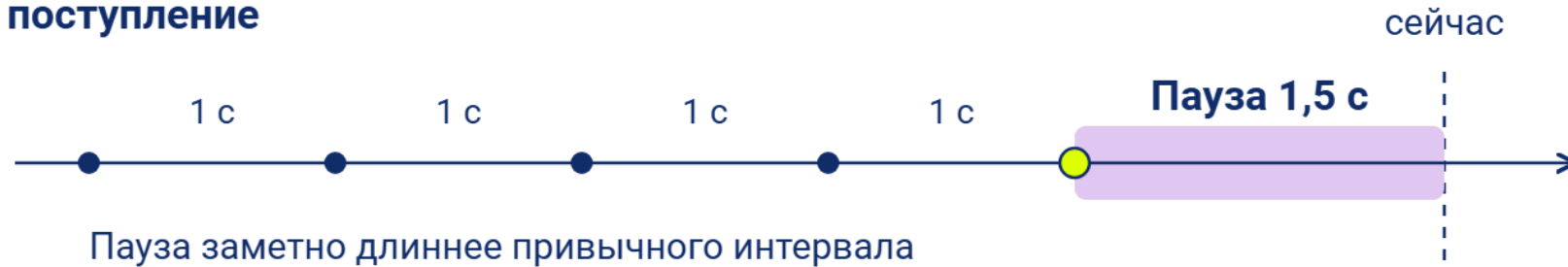
Ограничения:

- Первое подозрение – в среднем за $O(1)$ периодов; распространение обновления – порядка $O(\log N)$.
- Ложные подозрения остаются возможными; переход в Dead требует дополнительного ожидания.
- Каждый процесс по-прежнему хранит список участников: $O(N)$ памяти.

Адаптивный детектор: насколько необычна пауза?

Точки — получение heartbeat у наблюдателя. Масштаб одинаковый.

Регулярное поступление



Переменные интервалы



Условные примеры истории наблюдений

Оценивать паузу можно по недавней истории поступления heartbeat.

φ-детектор: степень подозрения

По недавним наблюдениям строим модель интервалов между heartbeat.

$$\varphi(d) = -\log_{10} P(X > d)$$

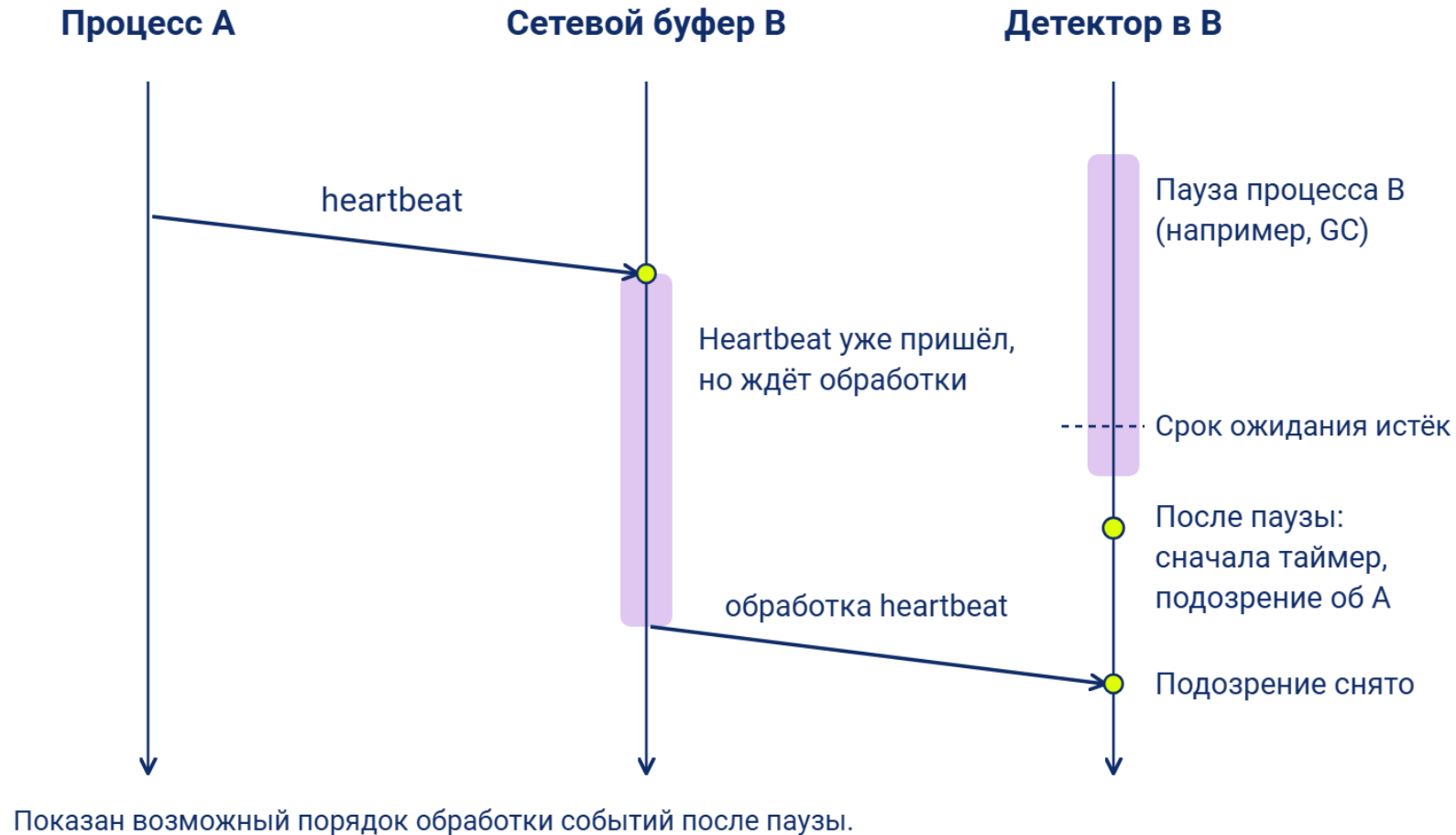
- **d** – время с последнего heartbeat.
- **X** – интервал между heartbeat в построенной модели.

Вероятность интервала длиннее d по модели	φ
10 %	1
1 %	2
0,1 %	3

φ характеризует редкость паузы по эмпирической модели, а не вероятность отказа.

Приложение начинает подозревать процесс при **φ ≥ Φ**. Чем выше порог **Φ**, тем осторожнее решение и дольше ожидание.

Задержка может быть у наблюдателя



При оценке чужого состояния нужно учитывать задержки собственной обработки.

Самодиагностика процесса

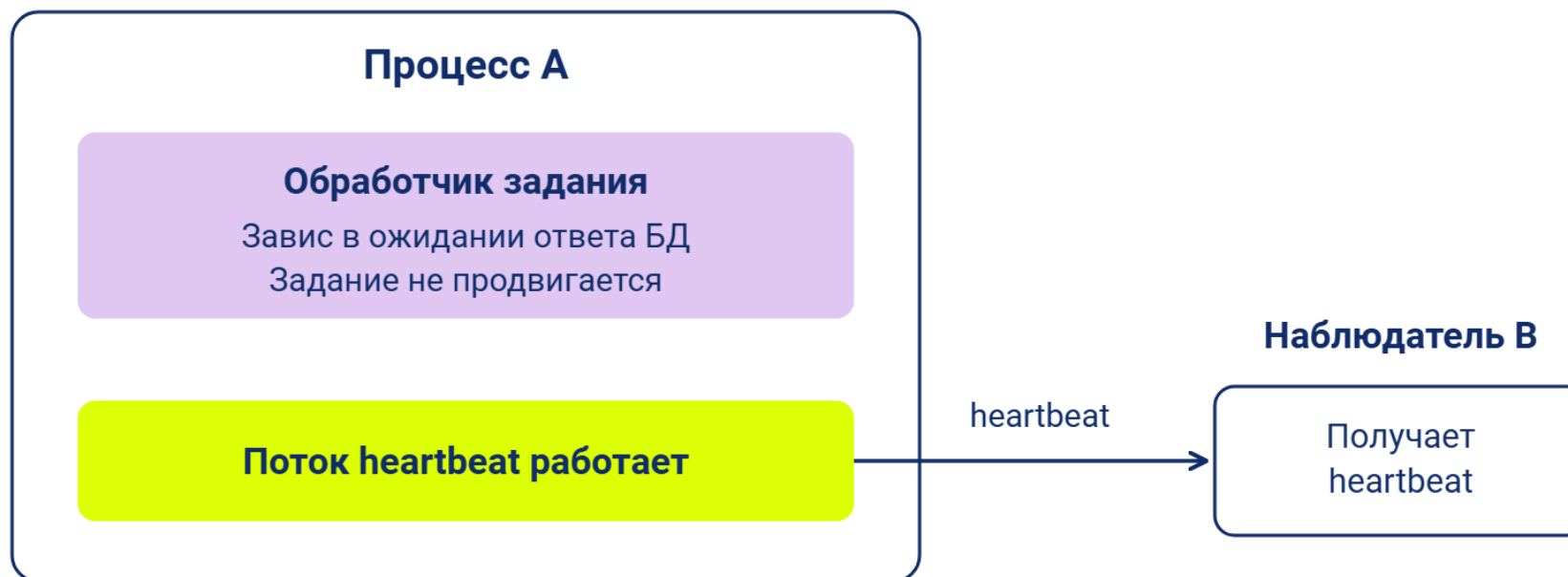
Процесс учитывает признаки проблем у себя (self-awareness).

Локальное наблюдение	Возможный вывод
Таймеры и обработчики выполняются с большим опозданием	Процесс не успевает обрабатывать события
ОС сообщает об отключении сетевого интерфейса	Нарушено локальное сетевое соединение
Одновременно пропали ответы от многих участников	Возможна собственная сетевая изоляция

Реакция: скорректировать проверки других участников и при необходимости ограничить собственную работу.

Lifeguard: учитывает состояние наблюдателя при выборе периода проверок и таймаутов.

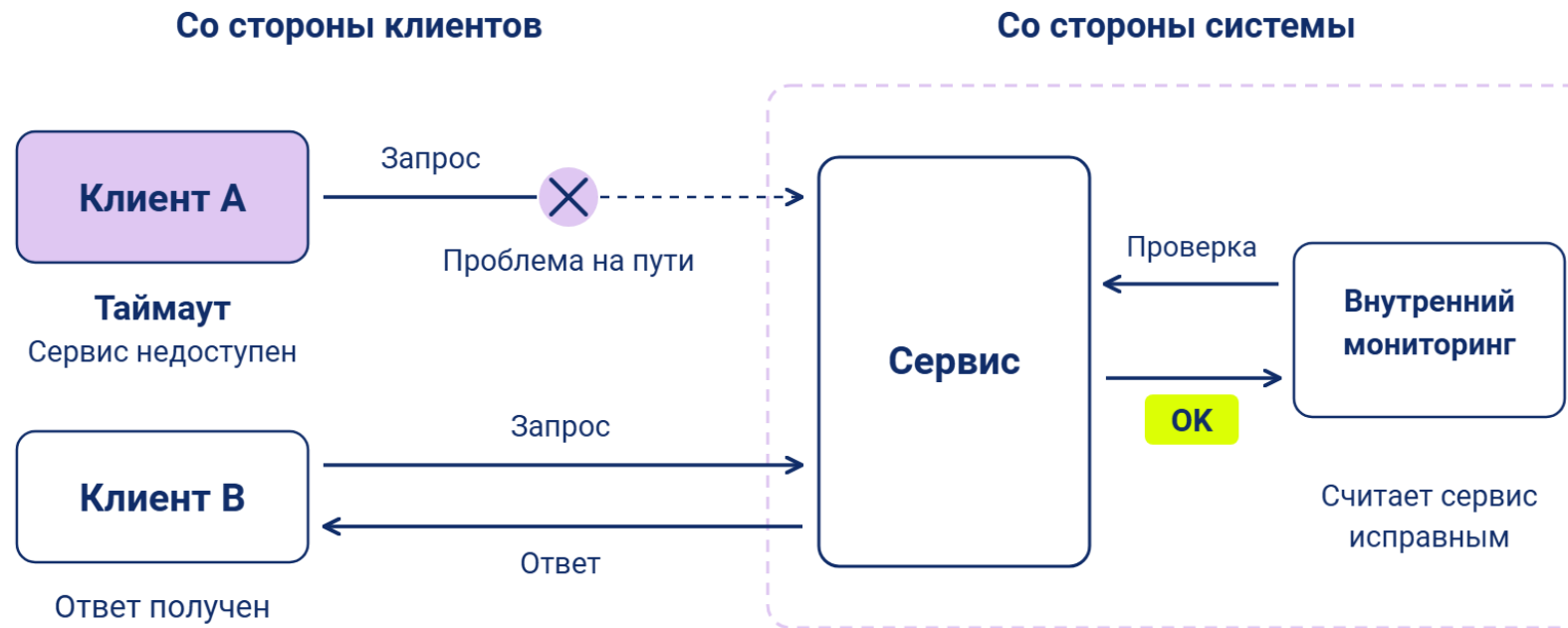
Heartbeat не гарантирует прогресс приложения



Проверка должна отражать тот прогресс, который нужен приложению.

Серые отказы (gray failures)

Наблюдаемость отказа может зависеть от точки наблюдения (differential observability).



Рапорта: компоненты сообщают об ошибках при обращении друг к другу, а детектор объединяет их наблюдения.

Что делать после подозрения?

Действие приложения	Что нужно учесть
Перестать выдавать новые задания процессу А	Уже выданное задание может продолжать выполняться
Повторить задание на процессе В	Побочные эффекты нужно защитить от повторов
Передать В исключительное право изменять данные	Нужно блокировать изменения от прежнего владельца (fencing)

Подозрение не останавливает процесс и не отзывает его полномочия.

Falcon: от подозрения к подтверждённой остановке

Локальные наблюдатели используют сведения приложения, ОС и инфраструктуры.
При необходимости они **принудительно останавливают** подозреваемый компонент.

Пример: процесс не отвечает на локальную проверку



Albatross: изоляция через сеть



Сначала обеспечить изоляцию процесса, затем сообщить о его исключении.

Детекторы на практике

Система	Механизм	Для чего используется
Consul	SWIM + Lifeguard через Serf/memberlist	Поддержание состава группы и обнаружение недоступных агентов
Apache Cassandra	Вариант φ -детектора по heartbeat, распространяемым через gossip	Оценка доступности узлов при обработке запросов
Akka Cluster	φ -детектор + распространение результатов через gossip	Обнаружение недоступных участников кластера
Kubernetes	Heartbeat через обновление Lease + таймаут	Контроль доступности рабочих узлов

Итоги

- Гарантии детектора зависят от модели отказов, синхронности и свойств каналов.
- Обнаружение отказа и распространение информации – разные задачи.
- SWIM сокращает объём регулярного обмена; φ оценивает подозрительность задержки по истории наблюдений.
- Доступность узла не гарантирует работоспособность приложения для всех клиентов.
- Реакция приложения должна сохранять корректность при ложных подозрениях.

Подозрение помогает начать восстановление, но само по себе не делает его безопасным.

Материалы: книги и конспекты

- [Distributed Systems](#), конспект курса, разделы 2.3–2.4.
Модели системы, отказоустойчивость и ограничения детекторов.
- [Database Internals](#), глава 9.
Обзор детекторов: heartbeat, ping, ф-детектор и gossip.
- [Designing Data-Intensive Applications](#), разделы главы 8.
Частичные отказы, сетевые задержки, таймауты и паузы процессов.
- [Distributed Systems](#), разделы 8.1, 8.2.8.
Основные понятия отказоустойчивости, модели отказов и их обнаружение.
- [Distributed Systems: Concepts and Design](#), раздел 15.1.1.
Предположения об отказах, надёжные и ненадёжные детекторы.

Материалы: статьи

Свойства детекторов и масштабирование

- [Unreliable Failure Detectors for Reliable Distributed Systems](#) – полнота, точность и классы детекторов.
- [A Gossip-Style Failure Detection Service](#) – распространение heartbeat через gossip.
- [SWIM](#) – проверки через посредников, подозрения и распространение обновлений.

Адаптация и дополнительные гарантии

- [The \$\phi\$ Accrual Failure Detector](#) – степень подозрения по истории heartbeat.
- [Lifeguard](#) – учёт состояния самого наблюдателя.
- [Falcon](#) – локальные наблюдатели и подтверждение остановки.
- [Albatross](#) – обнаружение отказов и изоляция через управляемую сеть.

Наблюдаемость отказов

- [Gray Failure](#) – различие между наблюдениями потребителей и системного детектора.
- [Panorama](#) – обнаружение отказов по наблюдениям взаимодействующих компонентов.