

Обнаружение отказов

Олег Сухорослов

Распределенные системы

Факультет компьютерных наук НИУ ВШЭ

11.10.2021

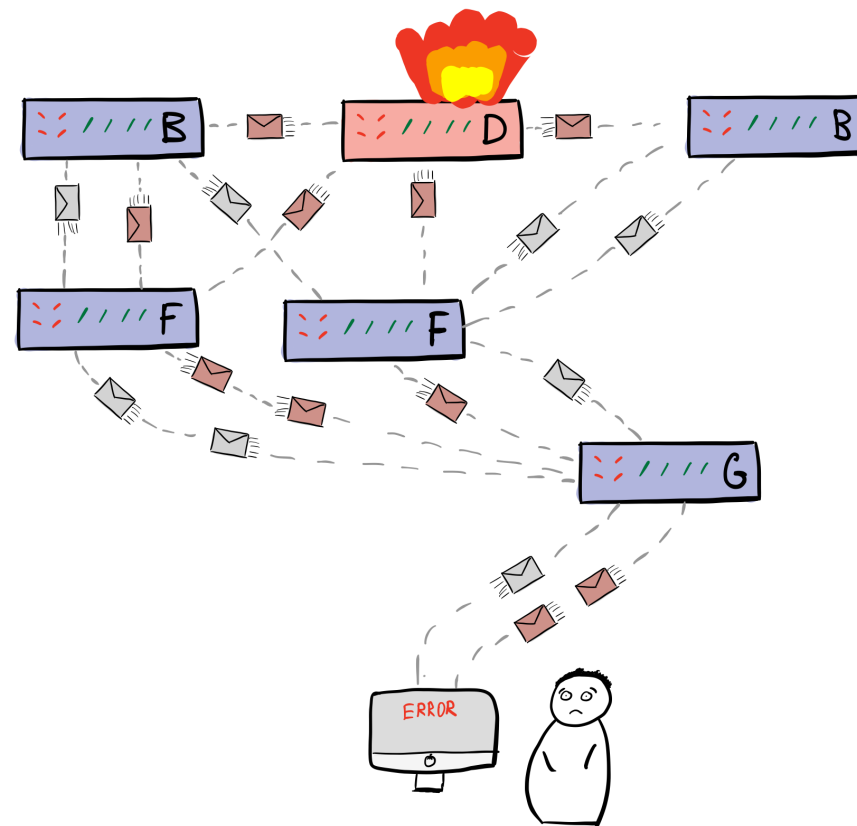
Отказы в РС

A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

-- Leslie Lamport

Failure is the defining difference between distributed and local programming.

-- Ken Arnold

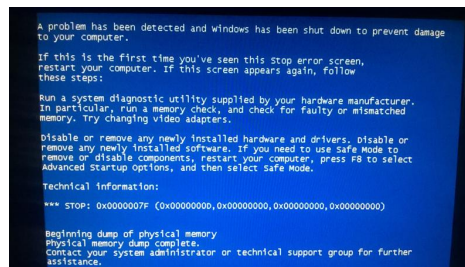


Терминология

- Сбой (fault)
 - Неисправность, нарушение условий
 - Является причиной ошибки
- Ошибка (error)
 - Проявление сбоя в состоянии системы
 - Может приводить к отказу
- Отказ (failure)
 - Отклонение поведения системы от ожидаемого

Сбои

- Источники
 - Оборудование
 - Сеть
 - Программное обеспечение
 - Внешние факторы
- Виды
 - Временный
 - Периодический
 - Постоянный



Обычные отказы

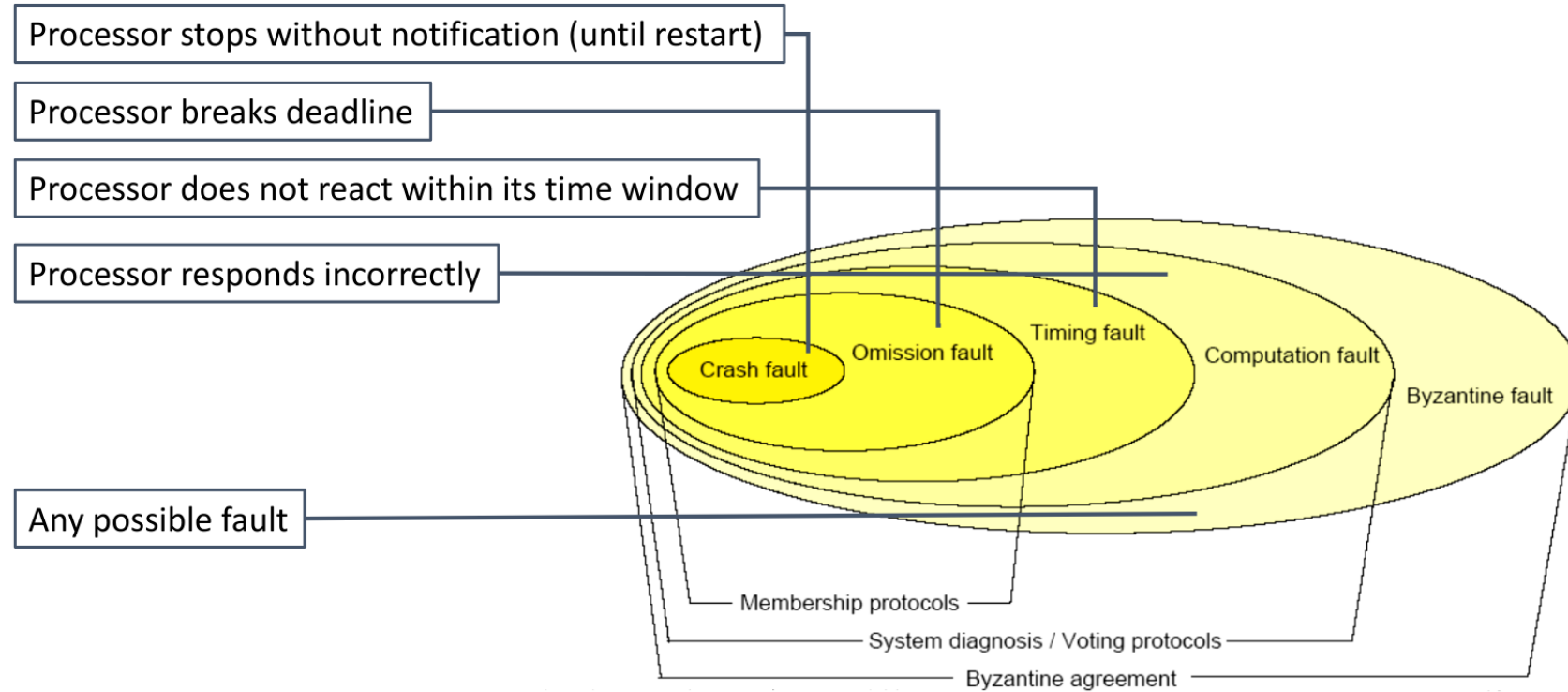
- Остановка (crash failure)
 - Процесс внезапно прекратил свою работу
 - Навсегда (crash-stop), временно (crash-recovery)
- Пропуск (omission failure)
 - Процесс пропускает часть действий
 - Действия процесса не видны другим
 - Процесс не получает или не отправляет сообщения
 - Включает отказы, вызванные сбоями сети
- Нарушение гарантий на время работы (timing failure)
- Некорректный ответ (response failure)

Произвольные (византийские) отказы

- Процесс активен и реализует произвольную логику поведения, нарушая спецификацию системы
 - Процесс может пропускать действия или выполнять дополнительные действия
 - Процесс может отправлять сообщения, только выглядящие корректными
- Могут быть вызваны случайными сбоями или намеренной атакой на систему



Модели отказов

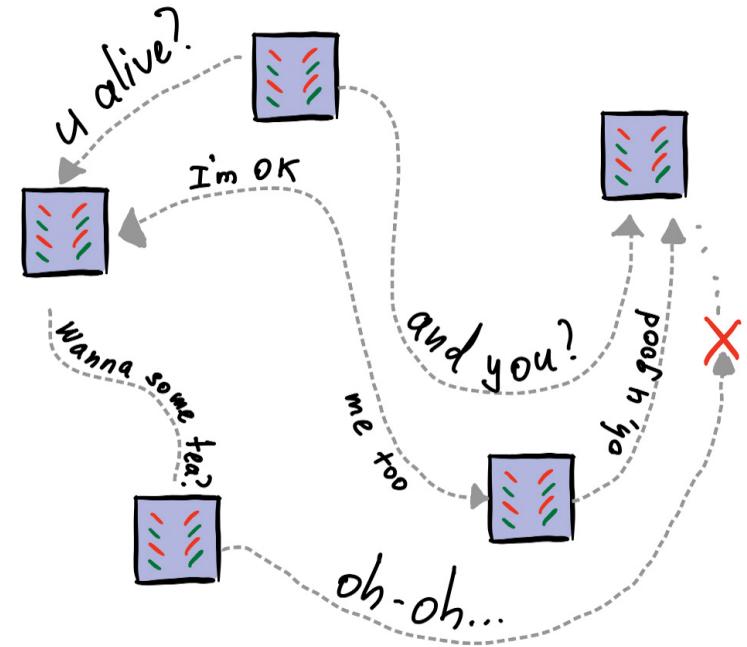


Аспекты отказоустойчивости

- Предотвращение отказов
- **Обнаружение отказов**
- Реагирование на отказы
 - маскировка отказов за счёт избыточности
 - устранение отказов (восстановление)
- Прогнозирование отказов

Детектор отказов

- Компонент, определяющий состояние процессов в системе
 - Какой сейчас статус процесса X?
 - Healthy, Failed, Unsuspected, Suspected
- Часто присутствует на каждом процессе (локальный детектор)
- Детектор может давать разные ответы на разных процессах
- Должен быстро и точно обнаруживать отказы, не нагружая систему



Свойства детектора отказов

- Полнота (completeness)
 - Каждый отказавший процесс должен в конце концов стабильно подозреваться
 - Сильная (strong) полнота: ... каждым корректным процессом
 - Слабая (weak) полнота: ... некоторым корректным процессом
- Точность (accuracy)
 - Корректные процессы не должны подозреваться
 - Сильная точность: никакой корректный процесс не подозревается ...
 - Слабая точность: некоторый корректный процесс никогда не подозревается ...
 - В конечном счете (eventual): свойство сильной или слабой точности выполняется спустя некоторое время

Классы детекторов

- Полнота обеспечивается легко
 - Как из слабой полноты получить сильную?
- Точность обеспечить гораздо сложнее
 - Надежный детектор
 - Не ошибается (сильная точность)
 - Ответы: *Unsuspected*, *Failed*
 - Ненадежный детектор
 - Может ошибаться (false positives)
 - Ответы: *Unsuspected*, *Suspected*

Простейший детектор отказов

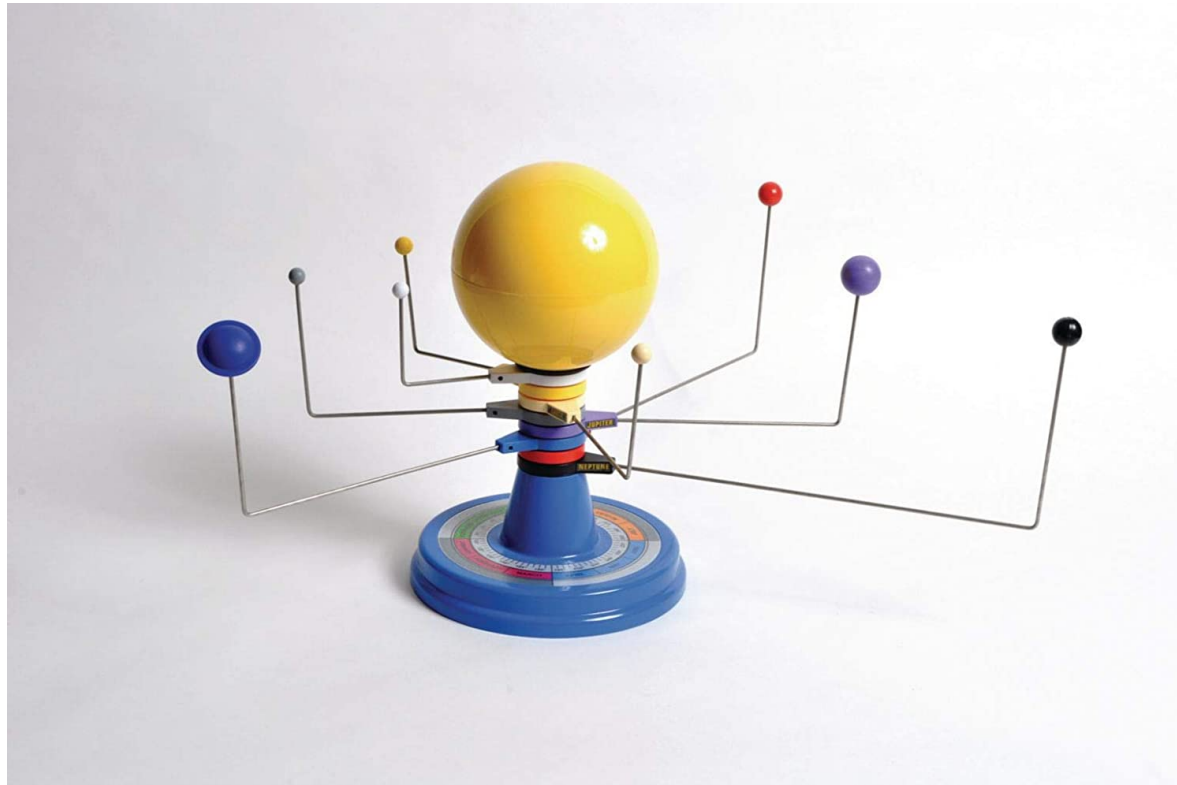
- Периодический прием сообщений от наблюдаемого процесса
 - Активная (pings) или пассивная (heartbeats) проверка
 - Интервал между отправками сообщений - T_{send}
- Процесс *Suspected*, если от него ничего не поступало в течении некоторого таймаута T_{fail}
- Если потом будет получено сообщение, то процесс становится *Unsuspected*

Выбор параметров детектора

- Интервал между проверками
 - Малые значения увеличивают нагрузку на сеть
 - Большие значения увеличивают время обнаружения отказа
- Таймаут
 - Heartbeats: $T_{fail} = T_{send} + D$
 - D - оценка максимального времени передачи сообщения
 - Малые значения могут приводить к частым ложным срабатываниям
 - Большие значения увеличивают время обнаружения отказа
 - Сетевая задержка может изменяться во время работы системы

Является ли наш детектор надежным?

Смотря в какой модели Вселенной распределенной системы



Модели распределенных системы

- Синхронная система
 - Времена обработки и передачи сообщений ограничены сверху
 - Наш детектор можно сделать надежным, выбрав соответствующее значение таймаута
- Асинхронная система
 - Процессы могут обрабатывать сообщения с произвольной скоростью
 - Время передачи сообщений не ограничено
 - Нельзя отличить медленно выполняющийся процесс от отказавшего
- Частично синхронная система
 - Поведение приближено к синхронной системе
 - Но верхние границы могут быть не точными и соблюдаться не всегда

Полезен ли ненадежный детектор?

Chandra T. D., Toueg S. Unreliable failure detectors for reliable distributed systems (1996)

Для консенсуса в асинхронной системе достаточно т.н. **eventually weak** детектора, обладающего следующими свойствами

- **Слабая полнота:** каждый отказавший процесс в конце концов постоянно подозревается некоторым корректным процессом
- **Слабая точность в конечном счёте:** в конце концов некоторый корректный процесс никогда не подозревается ни одним корректным процессом

Можно ли построить такой детектор?

- В асинхронной системе такой детектор построить нельзя
 - ...используя только обмен сообщениями
- На практике (~частично синхронные системы) можно к нему приблизиться
 - Полнота гарантируется
 - Точность можно улучшить и дать вероятностные гарантии

Как повысить точность нашего детектора?

Другие важные свойства детектора

- Время обнаружения отказов (detection time)
 - Полнота не говорит о том, насколько быстро происходит обнаружение
 - На практике важно уменьшить это время
 - Достаточно рассматривать время первого обнаружения отказа
- Эффективность
 - Быстрота + точность обнаружения отказов
- Масштабируемость
 - Нагрузка на процесс (число получаемых и отправляемых сообщений)
 - Нагрузка на сеть (число циркулируемых сообщений, трафик)
 - Отсутствие узких мест (выделенный процесс)

Варианты реализации

- Централизованная схема
 - Все процессы отправляют heartbeats выделенному процессу
- Схема "каждый с каждым"
 - Каждый процесс отправляет heartbeats все остальным процессам
- Существуют ли другие схемы?

Другие детекторы отказов

- Timeout-free (1997)
- Gossip-style (1998)
- SWIM (2001-2002)
- ϕ -accrual (2004)
- FALCON (2011)
- Albatross (2015)
- Panorama (2017)

Детектор отказов без таймаутов

Aguilera et al. Heartbeat: a Timeout-Free Failure Detector for Quiescent Reliable Communication (1997)

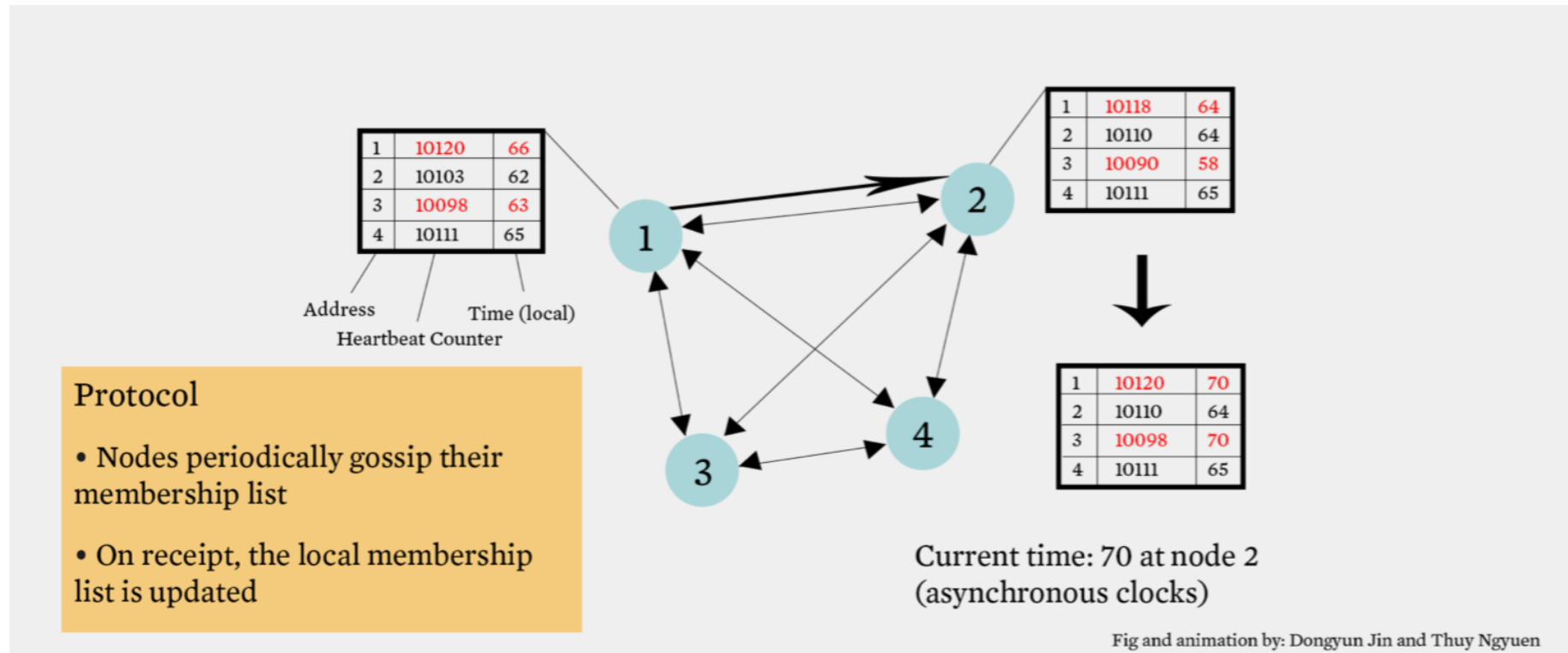
- Процессы и каналы между ними образуют граф
 - Между любой парой процессов есть путь
- Процесс хранит список своих соседей и счётчик для каждого из них
- Процессы периодически отправляют heartbeat-ы своим соседям
- Процессы пересылают чужие heartbeat-ы и обновляют локальные счётчики
- Детектор отказов выводит вектор счётчиков без их интерпретации

Детектор отказов без таймаутов

```
1  For every process  $p$ :
2
3      Initialization:
4          for all  $q \in neighbor(p)$  do  $\mathcal{D}_p[q] \leftarrow 0$ 
5
6      cobegin
7          || Task 1:
8              repeat periodically
9                  for all  $q \in neighbor(p)$  do send $p,q$ (HEARTBEAT,  $p$ )
10
11         || Task 2:
12             upon receive $p,q$ (HEARTBEAT,  $path$ ) do
13                 for all  $q$  such that  $q \in neighbor(p)$  and  $q$  appears in  $path$  do
14                      $\mathcal{D}_p[q] \leftarrow \mathcal{D}_p[q] + 1$ 
15                      $path \leftarrow path \cdot p$ 
16                 for all  $q$  such that  $q \in neighbor(p)$  and  $q$  does not appear in  $path$  do
17                     send $p,q$ (HEARTBEAT,  $path$ )
18      coend
```

Gossip-style детектор

Van Renesse et al. A Gossip-Style Failure Detection Service (1998)



Gossip-style детектор

- Каждый процесс хранит список $[(process, counter, last_update_time)]$
- Периодически каждый процесс увеличивает свой счётчик и отправляет свой список случайным процессам
- При получении сообщения процесс обновляет свой список
- Процесс, счётчик которого давно не обновлялся, считается *Suspected*
- Дополнительный таймаут для удаления процесса из списка

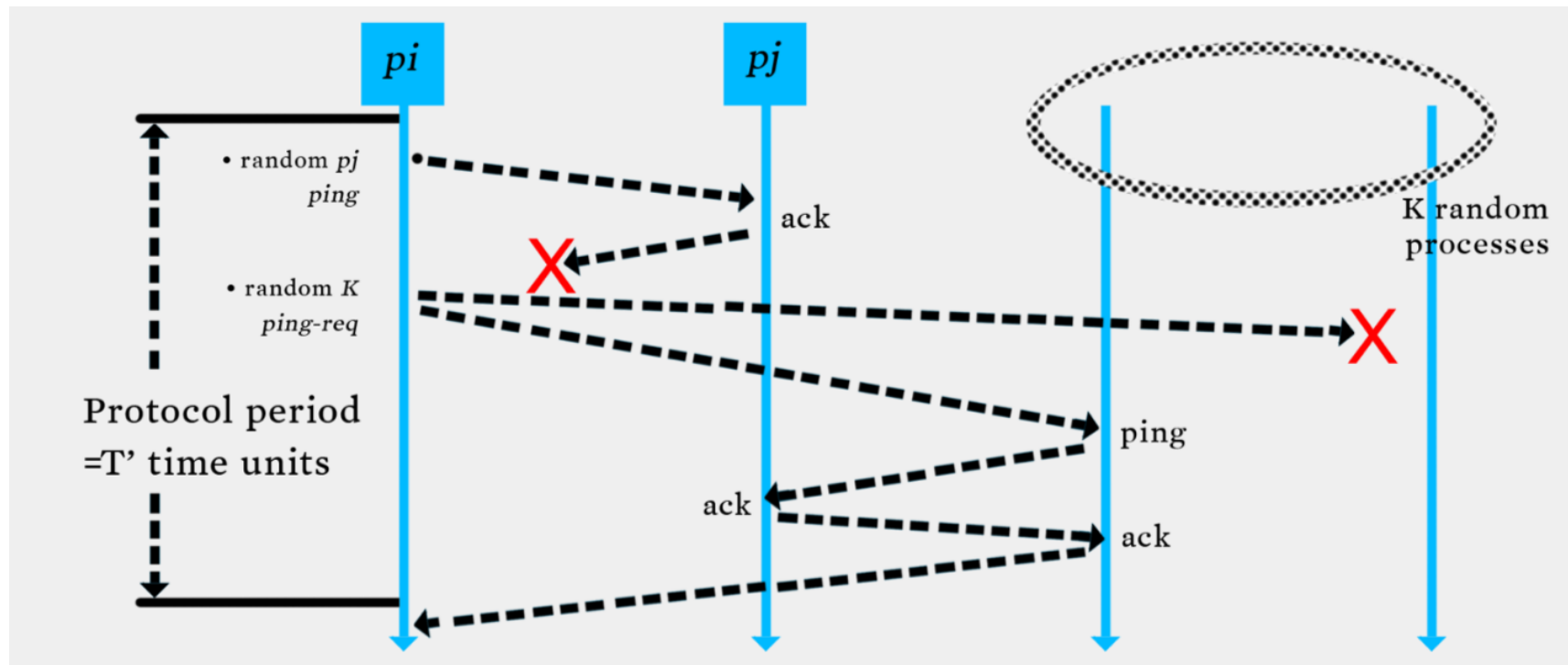
Характеристики детектора

- Зафиксируем требуемое нам время обнаружения отказа T
- Детектор all-to-all heartbeats
 - каждый процесс отправляет N сообщений размера $O(1)$ с периодом T
 - суммарное число сообщений: N^2 , сетевой трафик: N^2
 - нагрузка на процесс: N/T
- Детектор gossip-style
 - каждый процесс отправляет 1 сообщение размера $O(N)$ с периодом T_g
 - для распространения gossip требует $\log N$ шагов, отсюда $T = T_g \log N$
 - суммарное число сообщений: $N \log N$, сетевой трафик: $N^2 \log N$
 - нагрузка на процесс: $N \log N / T$

SWIM

Gupta et al. On Scalable and Efficient Distributed Failure Detectors (2001)

Das et al. SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol (2002)



Почему SWIM?

- Process Group Membership Protocol
 - кто сейчас является (живым) участником (системы, группы)?
 - отслеживание штатных входов (join) и выходов (leave) процессов
 - обнаружение отказов процессов (failure detection module)
 - распространение информации между участниками (dissemination module)
- Scalable
 - масштабируется лучше heartbeats и gossip-style на большое число процессов
- Weakly-consistent
 - нет строгой согласованности, информация на разных процессах может отличаться
- Infection-style
 - для распространения информации о состоянии процессов используется gossip

SWIM: Обнаружение отказов

- Процесс периодически выбирает случайный процесс P и отправляет ему *ping*
 - Если ответ не получен, то процесс просит K случайных процессов выполнить *ping(P)*
 - Если ответ по-прежнему не получен, то процесс P объявляется отказавшим
- Процесс рассылает информацию об отказе с помощью multicast или gossip
 - Обнаружение отказов и распространение информации реализуют разные модули
- Оптимизации
 - Выбор процесса для *ping* по схеме round-robin
 - Промежуточный статус *suspected*, с возможностью его снятия
 - Техника piggybacking для передачи информации в *ping/ack*

SWIM: Свойства

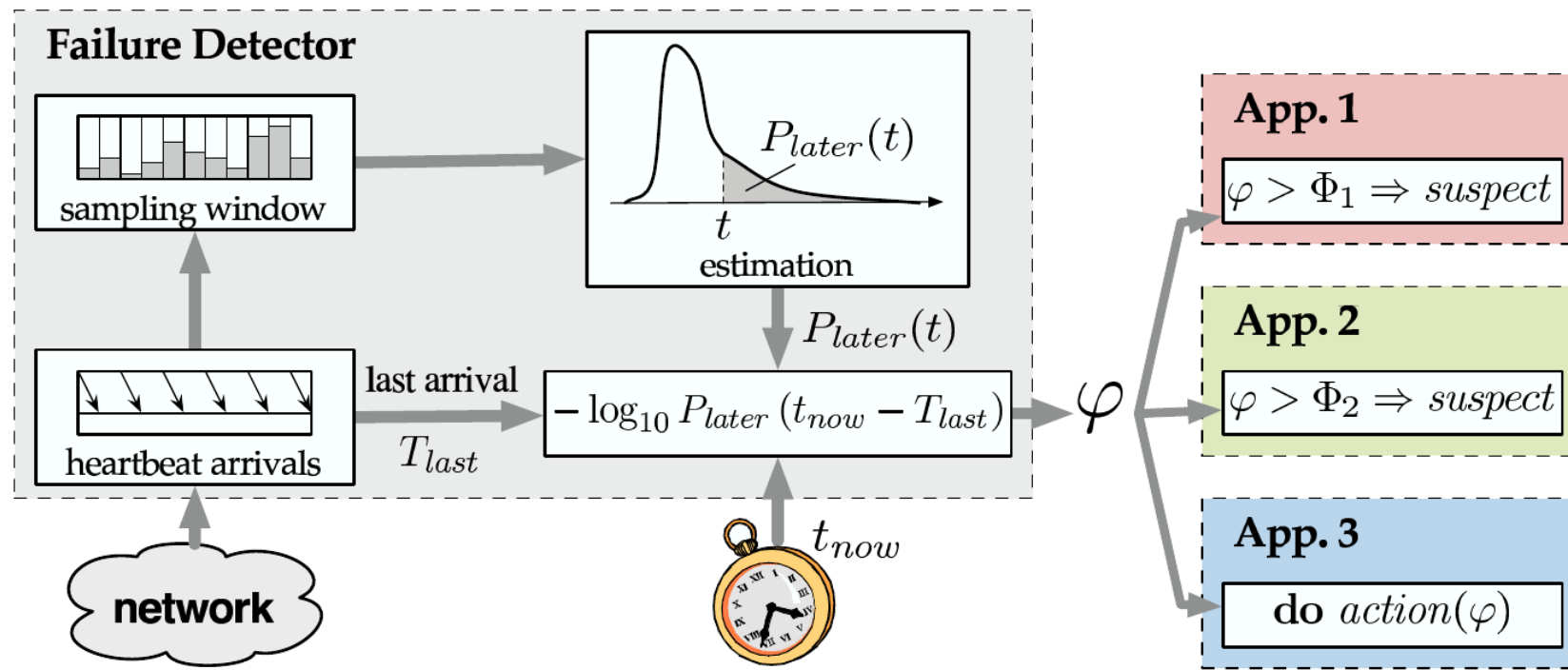
- Полнота
 - каждому процессу будет отправлен *ping*
- Время (первого) обнаружения отказа \sim периоду протокола
 - не зависит от N
- Высокая точность
 - false positive rate уменьшается экспоненциально с ростом K
- Отличная масштабируемость
 - нагрузка на процесс не зависит от N
 - нагрузка на сеть $O(N)$

SWIM: Применение и улучшения

- HashiCorp: memberlist, Serf, Consul
- Apple: Swift Cluster Membership
- Tarantool: Module swim, доклад
- Dadgar et al. Lifeguard: Local Health Awareness for More Accurate Failure Detection (2018)

φ -accrual детектор

Hayashibara et al. The φ Accrual Failure Detector (2004)

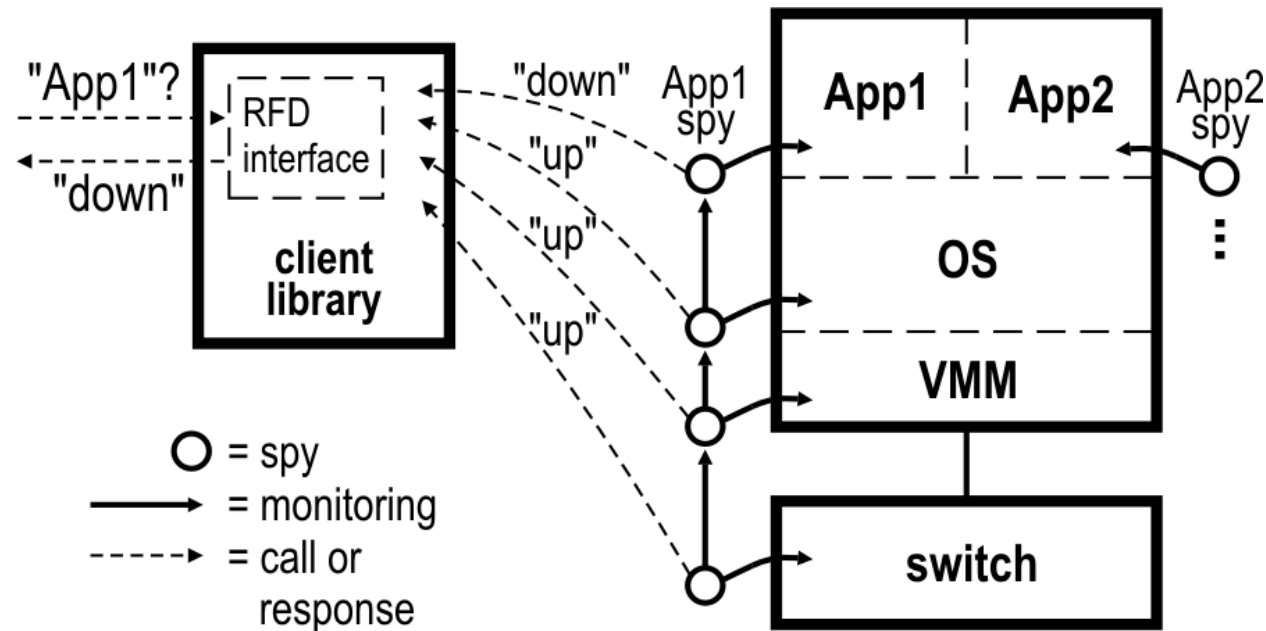


φ -ассуал детектор

- Возвращает значение φ , соответствующее уровню подозрения отказа
- Детектор динамически подстраивается под текущую нагрузку сети
- На основе истории интервалов поступления предыдущих сообщений вычисляется вероятность прихода сообщения
- Настраиваемое пороговое значение φ определяет момент, когда процесс считается отказавшим
- Сочетание мониторинга, анализа истории и прогнозирования
- Применение: Akka, Cassandra

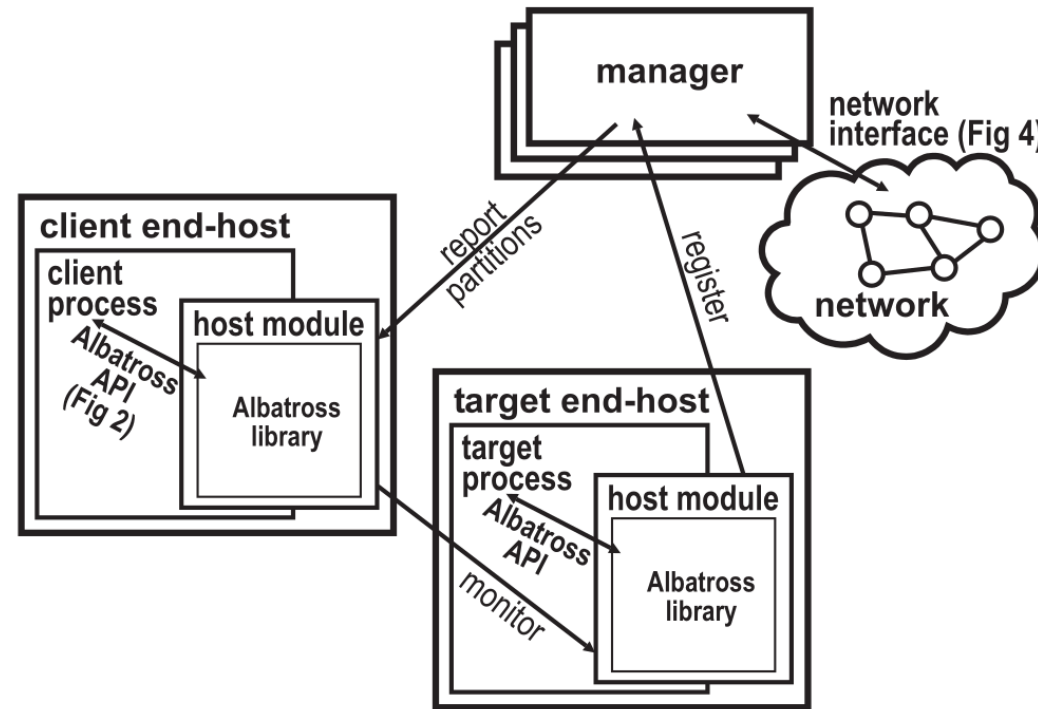
FALCON (Fast And Lethal Component Observation Network)

Leners et al. Detecting failures in distributed systems with the FALCON spy network (2011)



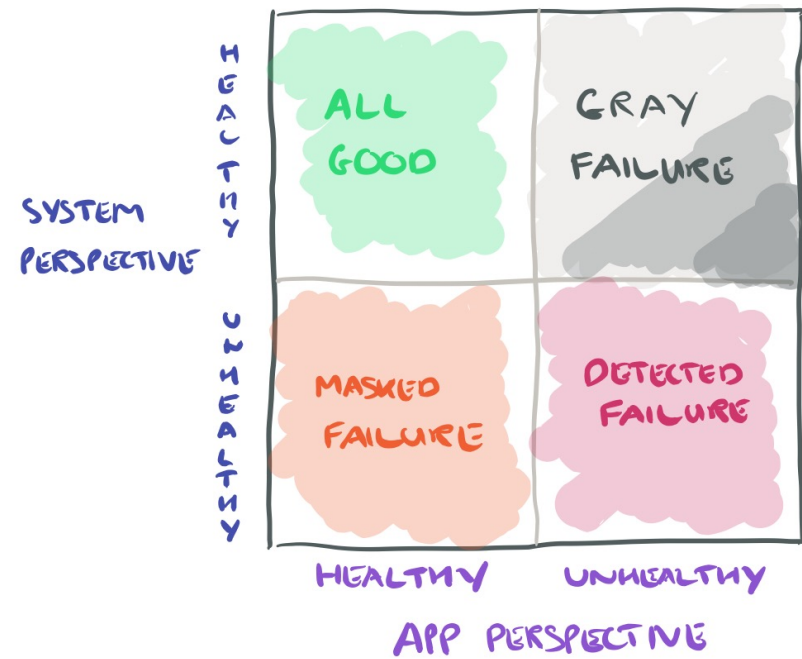
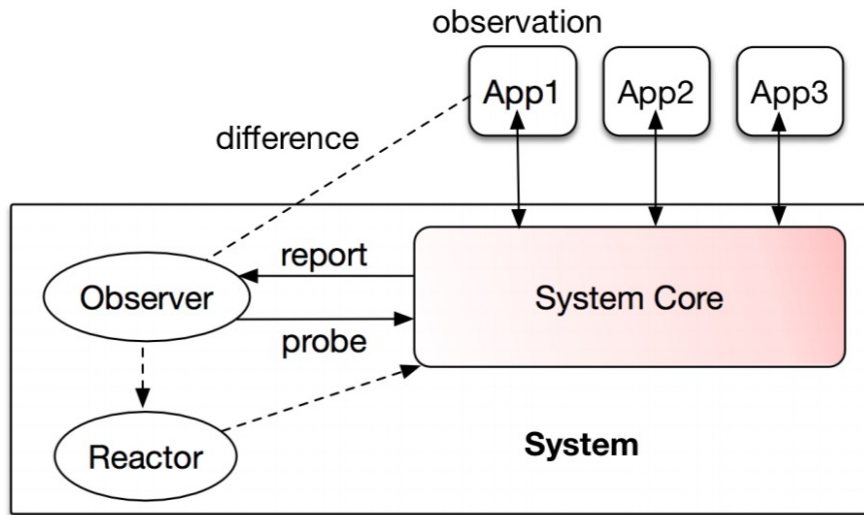
Albatross

Leners et al. Taming uncertainty in distributed systems with help from the network (2015)



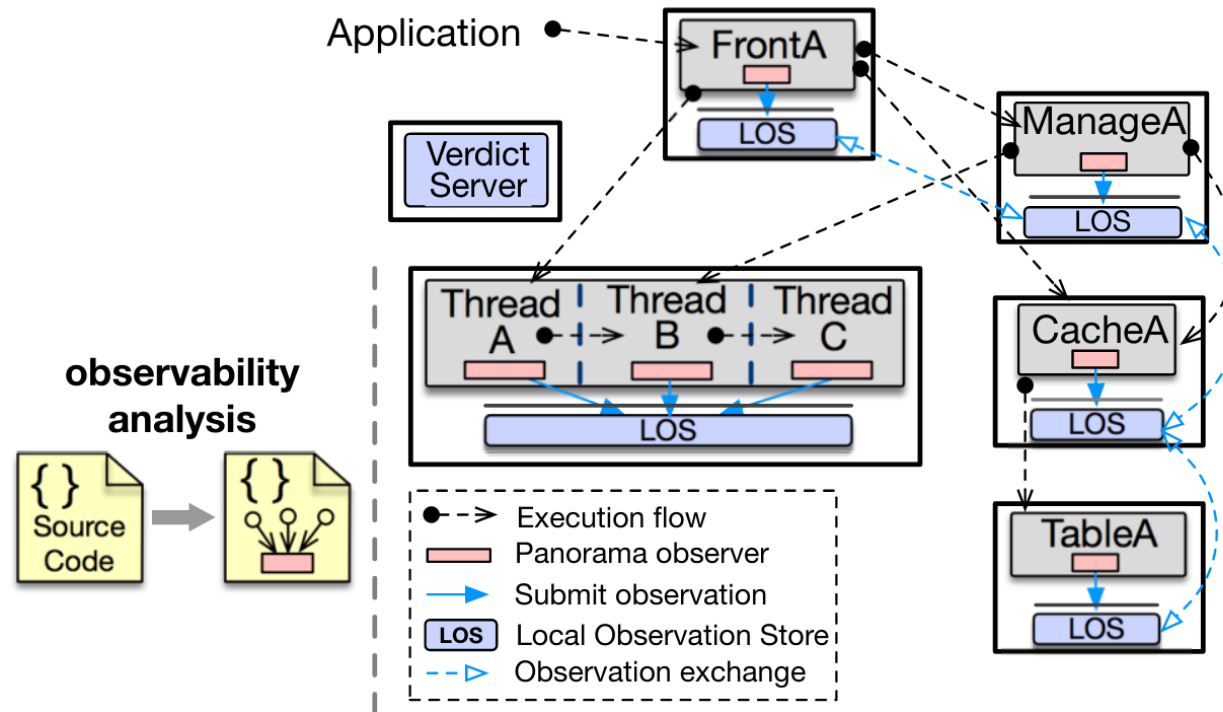
Серые отказы

Huang et al. Gray Failure: The Achilles' Heel of Cloud-Scale Systems (2017)



Panorama

Huang et al. Capturing and enhancing in situ system observability for failure detection (2018)



Литература и материалы

- van Steen M., Tanenbaum A.S. Distributed Systems: Principles and Paradigms. Pearson, 2017. (раздел 8.1)
- Coulouris G.F. et al. Distributed Systems: Concepts and Design. Pearson, 2011 (разделы 15.1.1, 15.5.4)
- Petrov A. Database Internals. O'Reilly, 2019 (глава 9)
- PWLSF- 09/2014 - Armon Dadgar on SWIM

Литература и материалы (дополнительно)

- Упомянутые статьи про детекторы отказов
- Making Gossip More Robust with Lifeguard
- Aguilera M., Walfish M. No Time for Asynchrony. (2009)