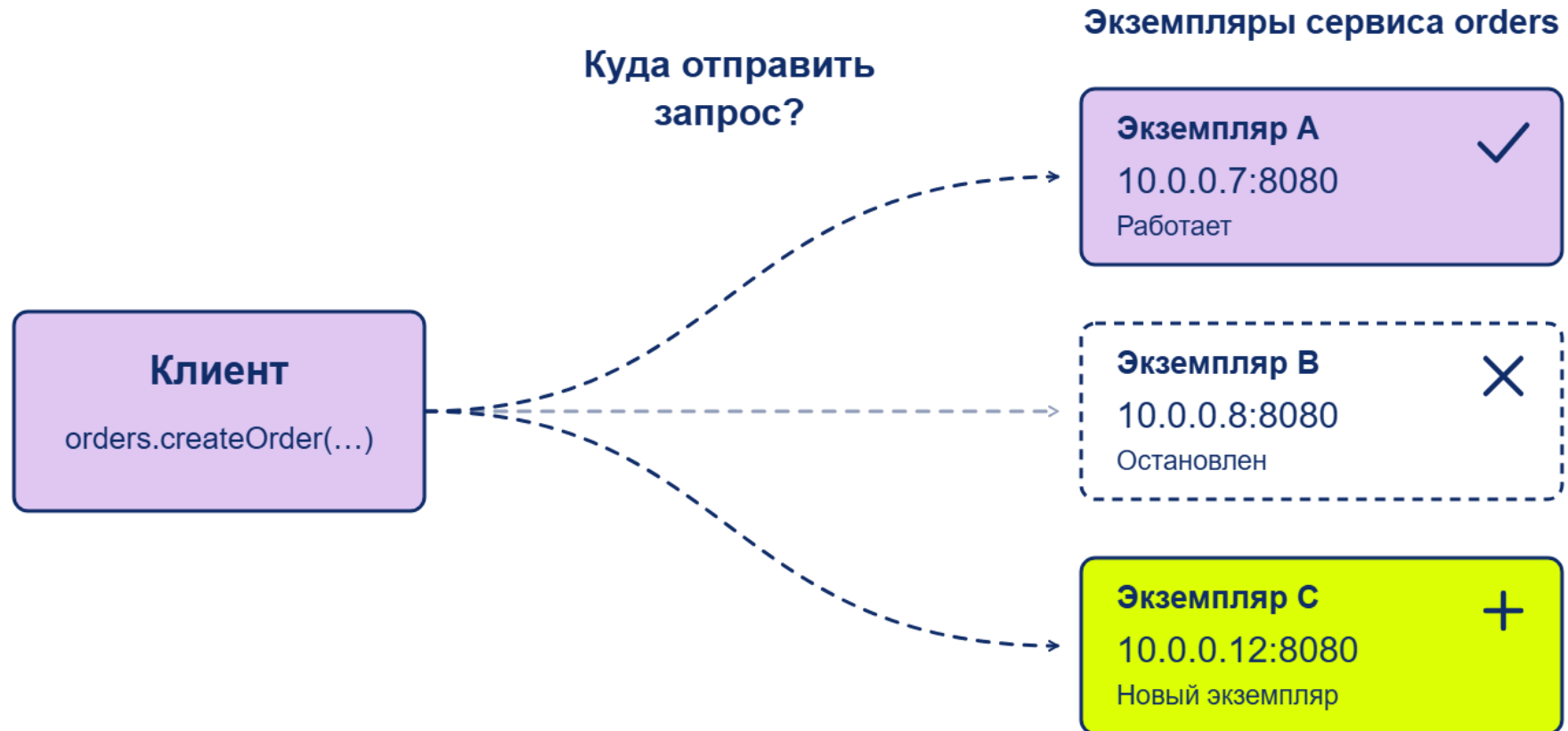


7. Именованное и поиск

Сухорослов Олег Викторович

19.10.2026

Как найти сервис заказов?



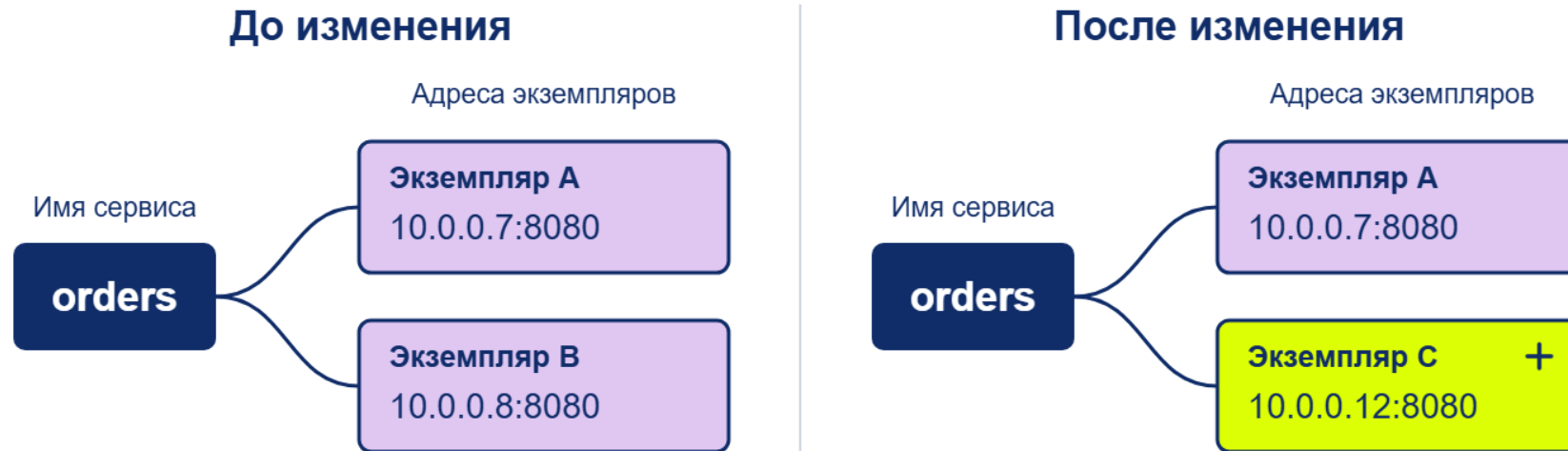
У сервиса может быть несколько экземпляров.
Они запускаются, останавливаются и меняют адреса.

План лекции

- Имена, адреса и генерация идентификаторов.
- Обнаружение сервисов: локальная сеть, реестр, Consul.
- DNS: глобальное пространство имён и разрешение запросов.
- P2P и DHT: имена по содержимому, поиск по ключу, Chord.
- Сравнение подходов и основные выводы.

Имя и адрес

- **Имя** обозначает объект: сервис, узел, файл, пользователя.
- **Адрес** указывает точку доступа к объекту, например IP-адрес и порт.
- Одному имени может соответствовать несколько адресов.
- Адреса могут меняться, пока имя остаётся прежним.



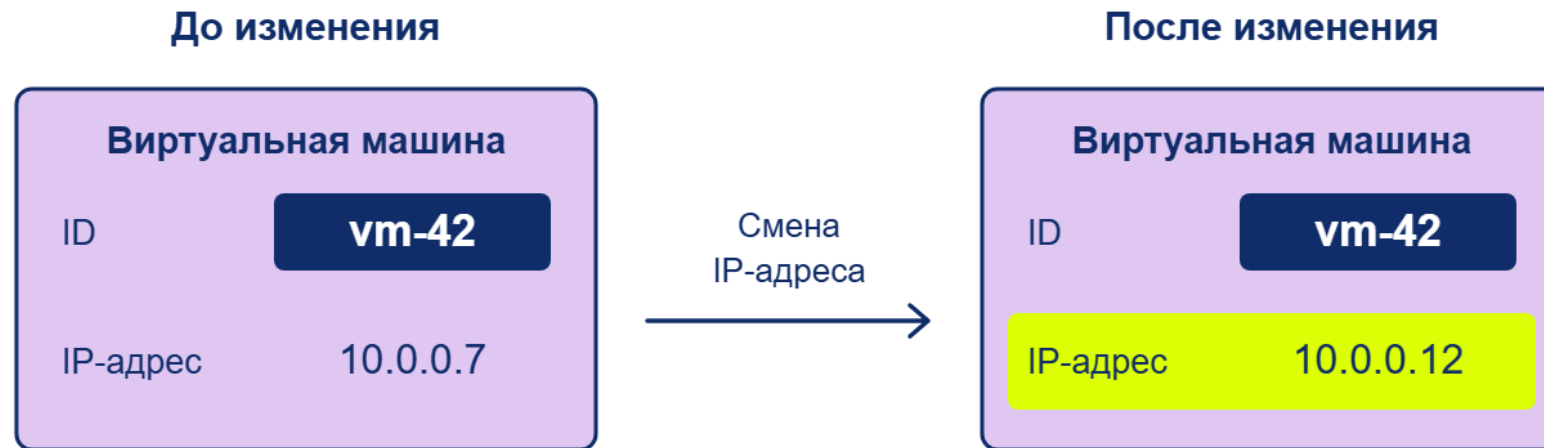
Устойчивый идентификатор

Идентификатор – имя, которое однозначно обозначает конкретный объект.

Устойчивый идентификатор:

- сохраняется при смене адреса объекта;
- не переиспользуется для другого объекта.

У одного объекта может быть несколько имён, в том числе идентификаторов.



Генерация уникальных идентификаторов

Подход	Как получаем уникальность	Что требуется
Общий счётчик	Согласованная выдача номеров или диапазонов	Координация выдачи
ID узла + локальный номер	Каждый узел использует своё пространство	Уникальные ID узлов, защита от повторов после перезапуска
Случайный UUID	Вероятность совпадения пренебрежимо мала	Качественный генератор случайных чисел

Snowflake

Время

ID узла

Счётчик

в пределах миллисекунды

Разрешение имени

- **Разрешение имени** – получение связанных с ним данных.
- Для сервиса результатом могут быть **адреса экземпляров**.
- Обращение к выбранному экземпляру – **отдельный шаг**.

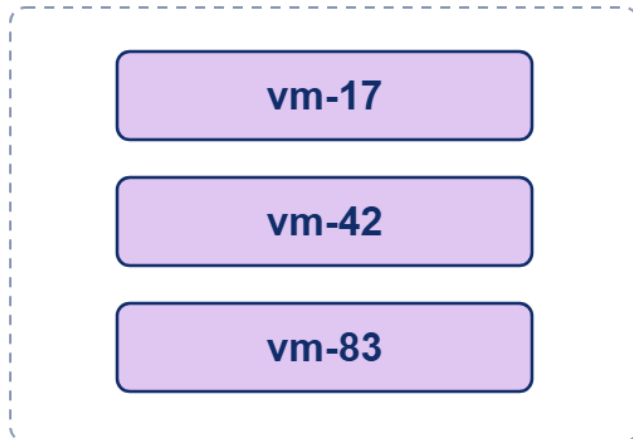


Пространство имён

Пространство имён задаёт допустимые имена и правила их организации.

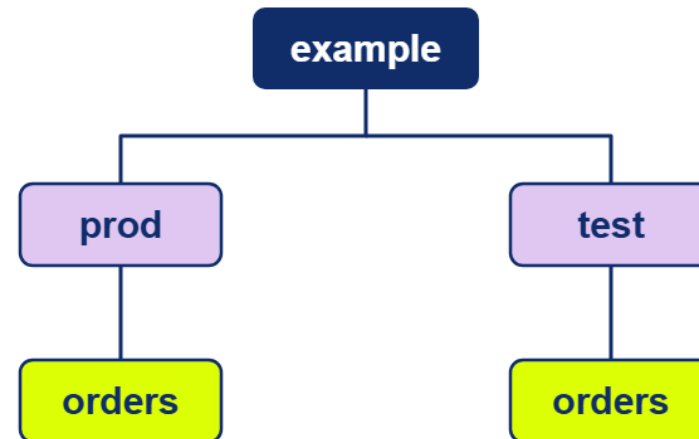
- **Плоское:** каждое имя рассматривается как единое целое.
- **Иерархическое:** компоненты имени задают путь в дереве.

Плоское пространство



Имена рассматриваются целиком

Иерархическое пространство



orders.prod.example

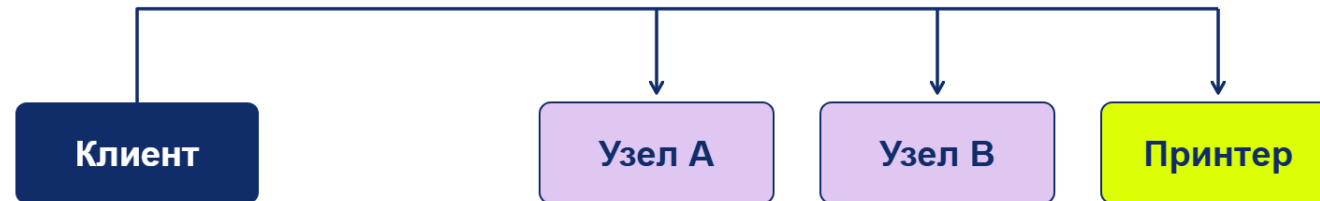
orders.test.example

Поиск имён в локальной сети

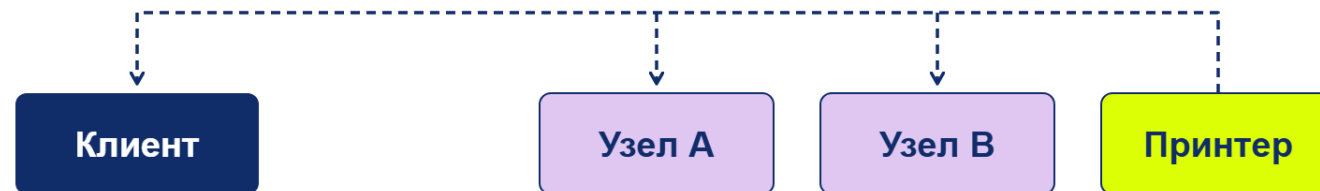
- Запрос рассылается широковещательно или в multicast-группу.
- Участник, отвечающий за имя, сообщает адрес.
- Выделенный сервер имён не требуется, но область поиска ограничена.

mDNS · один локальный сегмент

1. Запрос: где `printer.local`?



2. Ответ: `printer.local` → 192.168.1.50



Реестр сервисов

Реестр сервисов (*service registry*) – служба имён, которая хранит сведения об экземплярах сервисов.



Актуальность сведений в реестре

- При штатной остановке экземпляр снимают с регистрации.
- Сбои обнаруживают проверками состояния или истечением регистрации.
- До обнаружения сбоя реестр может выдавать устаревший адрес.

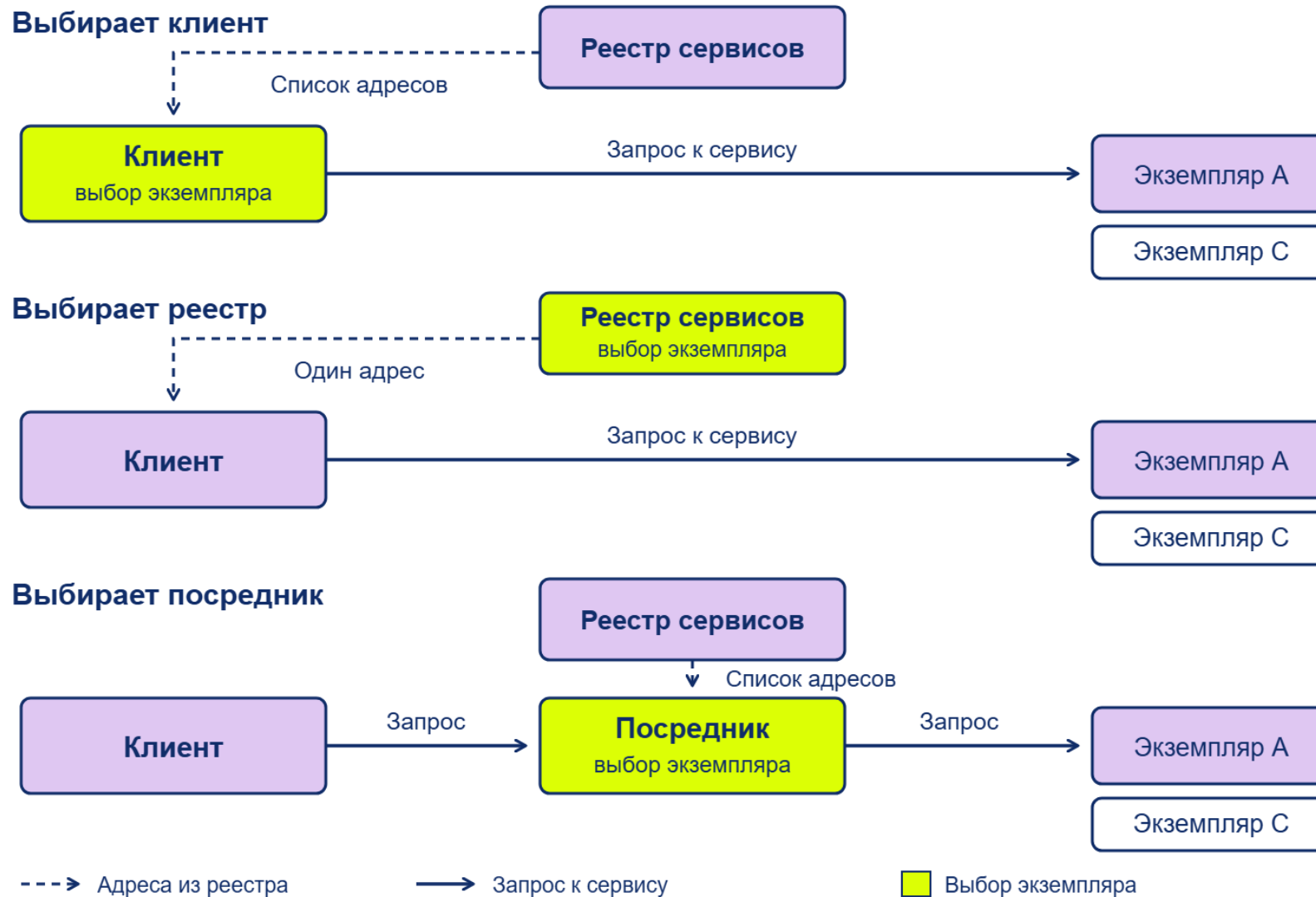
Срок регистрации — 30 секунд

В примере используются только продления



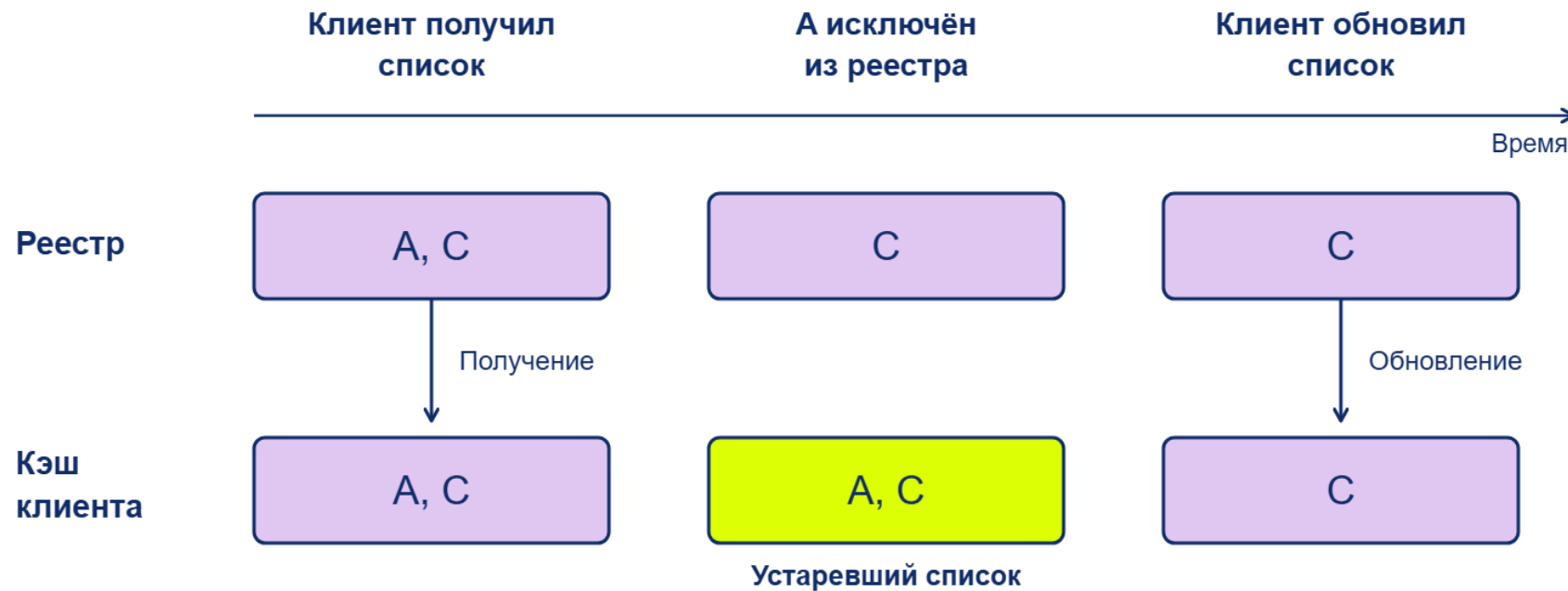
Адрес выдаётся ещё 25 секунд после остановки

Кто выбирает экземпляр сервиса?



Кэширование адресов

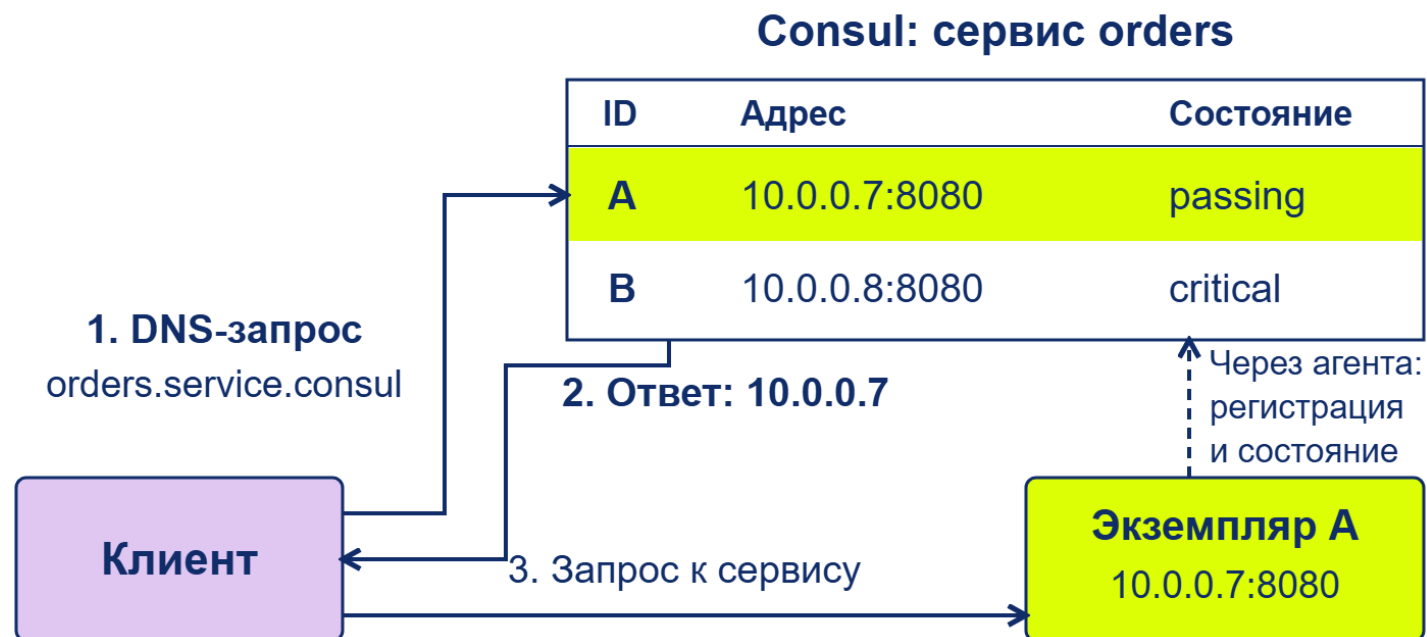
- Клиент или посредник сохраняет полученные адреса, чтобы реже обращаться к реестру.
- После изменения реестра сохранённый список может устареть.
- Список обновляют периодически, после истечения срока хранения или по уведомлению.



А и С обозначают адреса экземпляров сервиса orders.

Пример: обнаружение сервисов в Consul

- Экземпляры регистрируются вместе с проверками состояния.
- Адреса доступны через **DNS** или **HTTP API**.
- Клиент обращается к выбранному экземпляру напрямую.



DNS-ответ не включает экземпляр B со статусом critical

Единый каталог имён для всего Интернета?

Требование	Проблема централизованного подхода
Доступность	Отказ общего сервиса затрагивает множество приложений
Масштабирование	В одной службе сходятся запросы и обновления
Низкая задержка	Клиенты находятся по всему миру
Независимое управление	Организации должны самостоятельно управлять своими именами

Нужно распределить хранение, обработку запросов и ответственность за данные.

DNS – распределённая служба имён

- Имена образуют **иерархическое пространство**
- По **имени и типу записи** можно получить адреса и другие сведения
- Ответственность за части пространства имён распределена между организациями с помощью **делегирования**
- Результаты запросов **кэшируются**

Пример запроса и ответа:

Запрос

Имя: cs.hse.ru.

Тип: A

Класс: IN

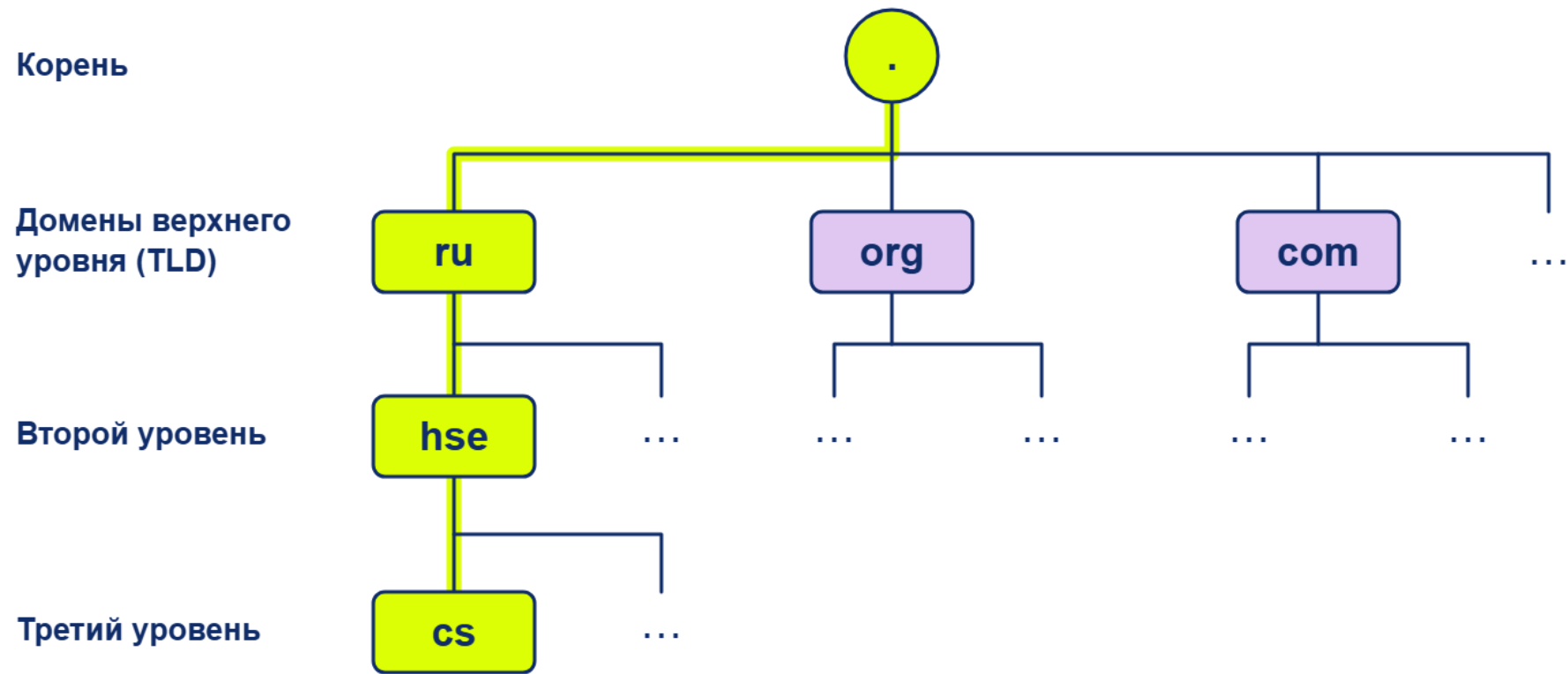
Ответ

Статус: NOERROR

Найдена одна запись

cs.hse.ru. 300 IN A 178.248.234.104

Глобальное пространство имён DNS



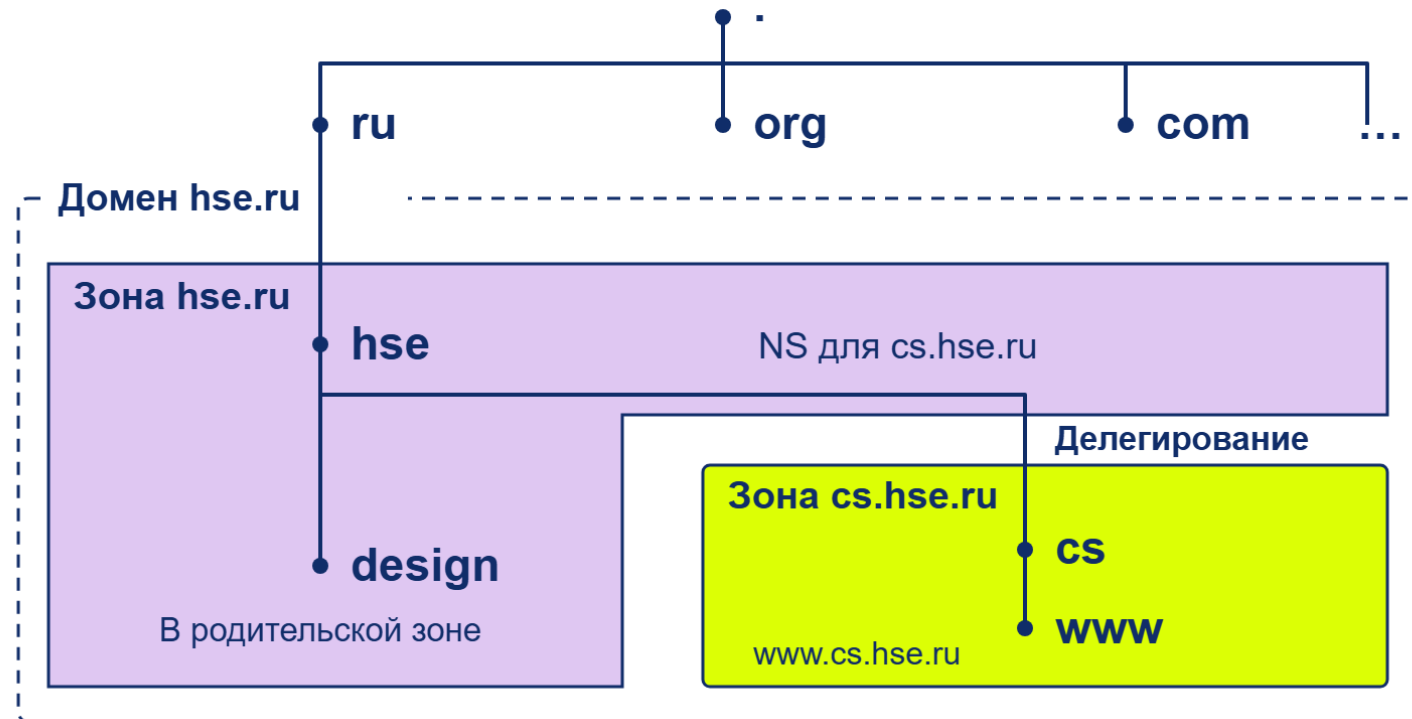
Полное имя: от выбранного узла к корню

cs.hse.ru.

Показана часть дерева

Домены, зоны и делегирование

- **Домен** – узел дерева имён и все его потомки.
- **Зона** – часть дерева без делегированных поддеревьев.
- **Делегирование** – выделение дочерней зоны; её серверы указаны в NS-записях родителя.



Итеративное и рекурсивное разрешение имён

Итеративная схема

Поиск www.hse.ru



Инициатор идёт по уровням пространства имён

Рекурсивная схема

Поиск www.hse.ru

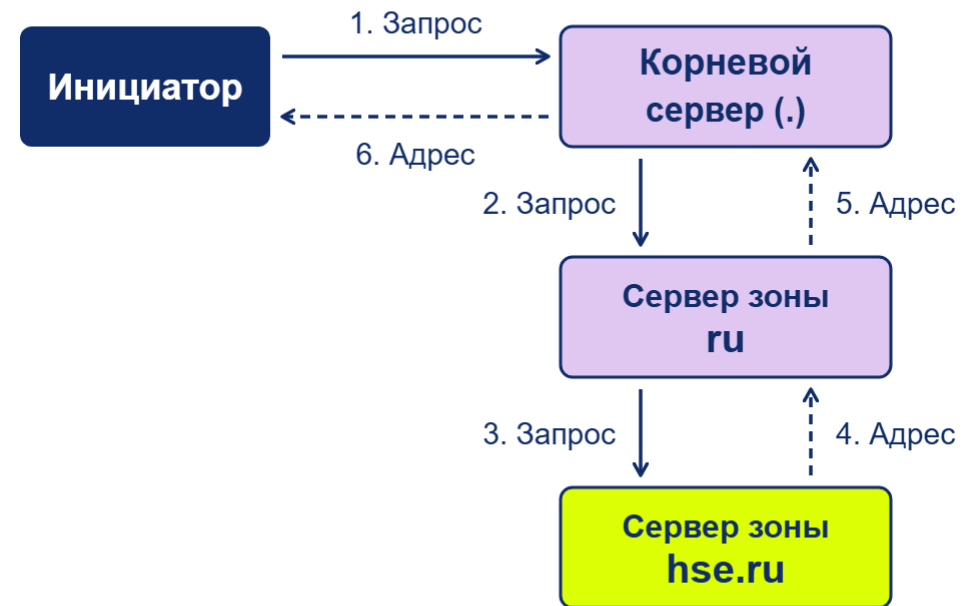


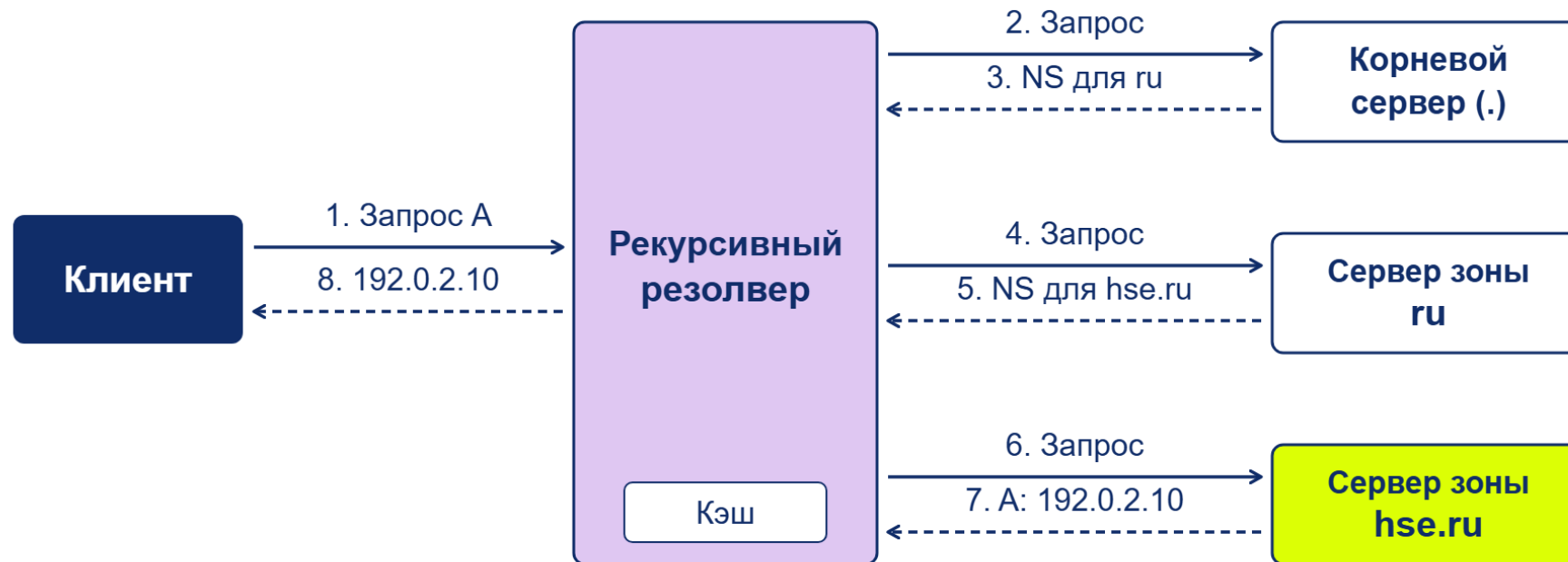
Схема для сравнения: в реальном DNS корень и TLD не выполняют рекурсию

Как выполняется DNS-запрос

Рекурсивный запрос

Итеративный поиск

Имя: www.hse.ru. · Тип: A · Нет подходящих записей в кэше



- Клиент поручает поиск рекурсивному резолверу.
- Резолвер итеративно следует делегированиям до нужной зоны.
- Полученный ответ кэшируется и возвращается клиенту.

DNS: доступность при отказах

- Зону обслуживают несколько **авторитетных серверов**.
- Если один недоступен, резолвер обращается к другому.
- Серверы размещают в разных сетях и на разных площадках.



Оба сервера авторитетны для зоны; копии получены заранее

Кэширование в DNS: TTL

- **TTL** ограничивает срок использования записи из кэша.
- Ответ из кэша содержит **оставшийся TTL**.



Попадание в кэш не продлевает TTL

В примере нет досрочного обновления и выдачи просроченных записей.

Основные типы DNS-записей

Тип записи определяет смысл и формат данных: адрес, другое имя, сервер зоны или параметры сервиса.

Тип	Что содержит	Пример данных
A	IPv4-адрес	192.0.2.10
AAAA	IPv6-адрес	2001:db8::10
CNAME	Каноническое имя	web.example.
NS	Имя сервера зоны	ns1.example.
SRV	Узел и порт сервиса	10 5 8080 node.example. приоритет · вес · порт · имя узла

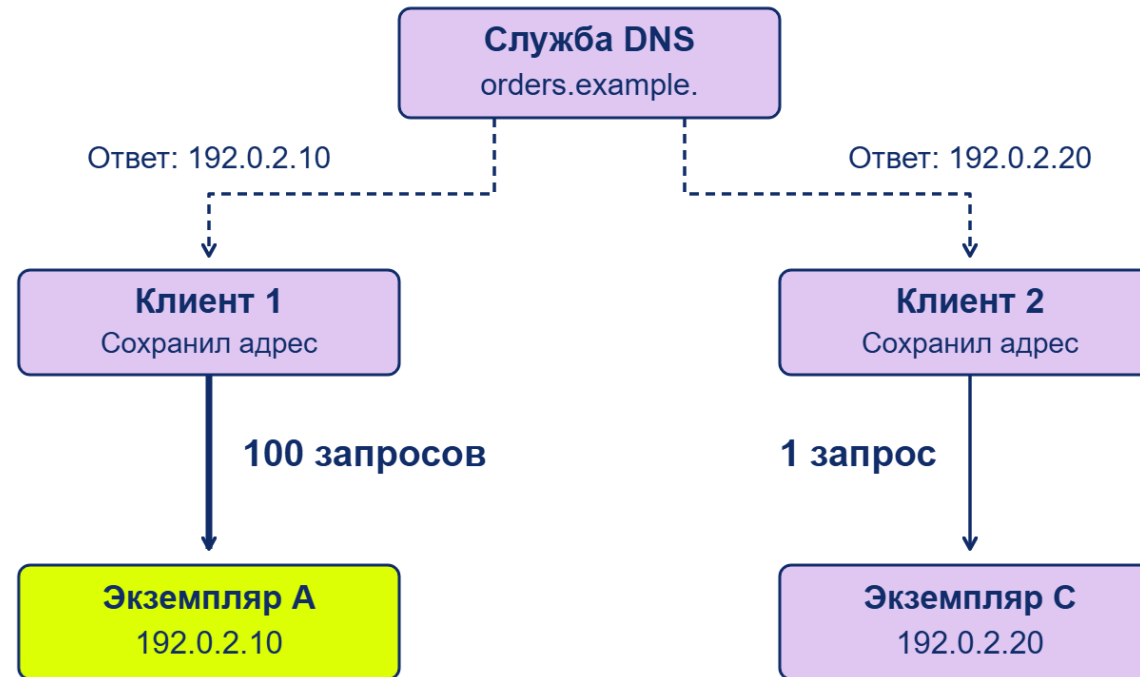
Имена и адреса в примерах условные.

DNS: отсутствие данных и сбой поиска

Результат	Что означает
NXDOMAIN	Имя не существует
NODATA	Имя есть, записей нужного типа нет
SERVFAIL	Поиск завершился ошибкой
Тайм-аут	Ответ не получен вовремя

DNS и распределение нагрузки

- DNS может выдавать разные адреса одного сервиса.
- Один DNS-ответ может использоваться для множества запросов.
- Равномерная выдача адресов не гарантирует равномерную нагрузку.



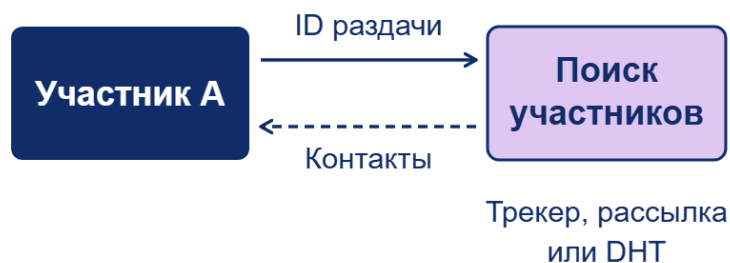
Условный пример: каждый клиент использует сохранённый адрес.

P2P: как участникам найти друг друга?

- В P2P участники могут и получать, и предоставлять данные.
- Данные распределены между узлами; состав участников меняется.
- Перед обменом нужно найти тех, у кого есть нужные данные.

Задача поиска: известный идентификатор → контакты участников.

1. Найти участников



Сначала контакты, затем обмен данными

2. Обмениваться данными

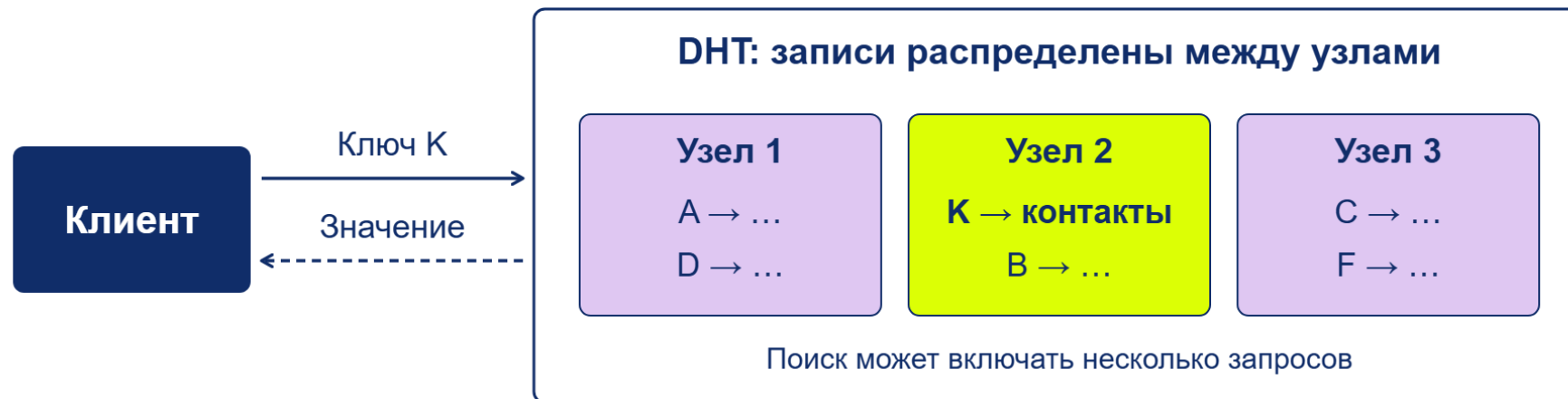


DHT: поиск по ключу

- **DHT (distributed hash table)** – распределённая таблица «ключ → значение».
- Пространство ключей плоское; ответственность распределена между узлами.
- Поиск находит нужные узлы без центрального каталога и опроса всей сети.

Пример BitTorrent: идентификатор раздачи → контакты участников.

Поиск по точному ключу

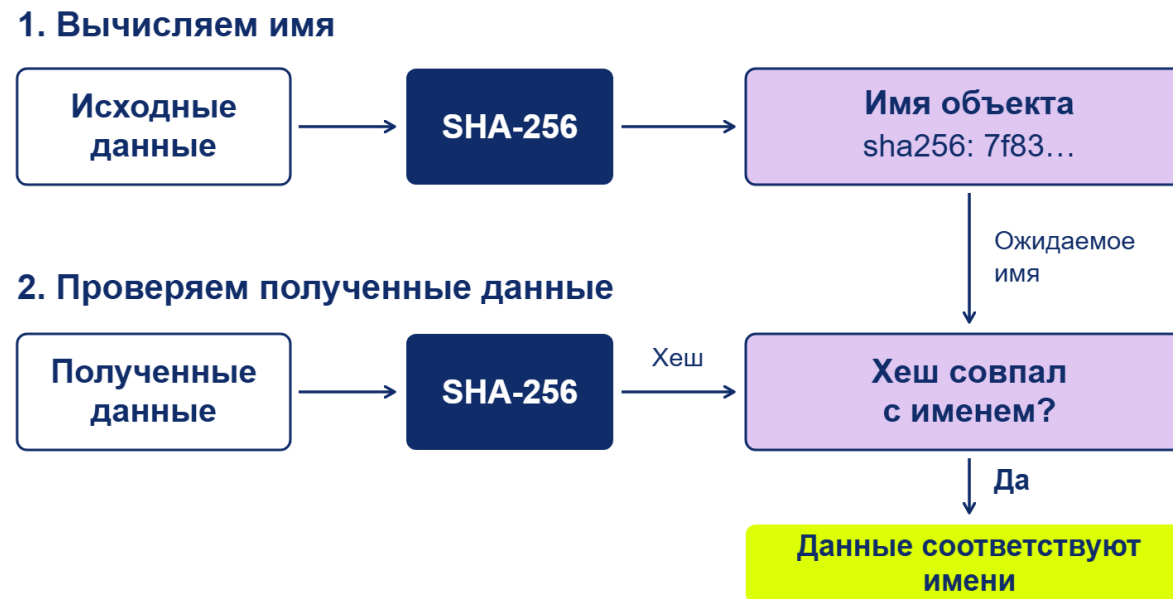


Пример BitTorrent: K = infohash, значение = контакты участников раздачи

Хеш содержимого как имя

Идентификатор = криптографический хеш содержимого объекта.

- Одинаковые данные получают одинаковое имя.
- Имя обозначает конкретную версию данных.
- Получатель вычисляет хеш и сравнивает его с ожидаемым именем.

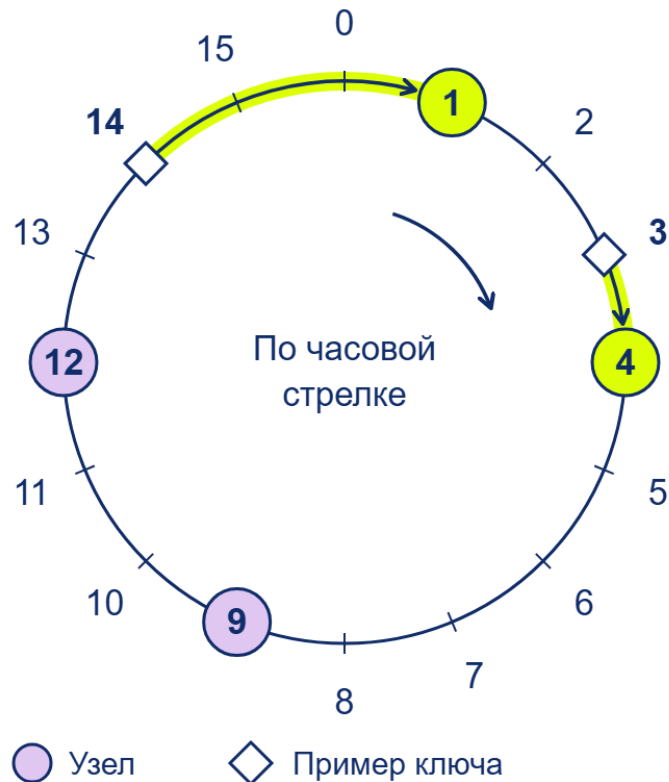


Self-certifying name: соответствие данных имени можно проверить самостоятельно.

Chord: поиск ответственного узла

Chord – протокол распределённого поиска: по ключу находит ответственный узел.

- Идентификаторы узлов и ключей образуют кольцо.
- За ключ отвечает первый узел по часовой стрелке.



Ответственный узел

Ключ 3 → узел 4

Ключ 14 → узел 1

Ключ 9 → узел 9

Узел 1 отвечает за:

13, 14, 15, 0, 1

После 15 снова идёт 0

Дуги показывают назначение ключа

Chord: поиск по цепочке соседей

- Каждый узел знает адрес следующего узла в кольце.
- Запрос передаётся по цепочке, пока ответственный узел не станет известен.
- В худшем случае – $O(N)$ переходов, где N – число узлов.

Ищем ключ 10 из узла 1

Пространство: $0 \dots 15$ · Узлы: 1, 4, 9, 12



Узел 9 возвращает результат: узел 12

Обращение к узлу 12 за данными — отдельный шаг

Chord: ускорение поиска

- **Таблица finger:** ссылки на узлы, отвечающие за точки со смещениями 1, 2, 4, 8, ...
- Следующий шаг – к ближайшему к ключу известному узлу, **предшествующему ключу по кольцу.**
- При равномерном размещении узлов и корректных таблицах – **$O(\log N)$** переходов.

Ищем ключ 10 из узла 1

Пространство: 0...15 · Узлы: 1, 4, 9, 12

Таблица finger узла 1

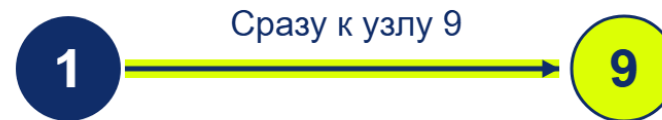
Сдвиг	Точка	Узел
+1	2	4
+2	3	4
+4	5	9
+8	9	9

4 строки, 2 разных узла

По цепочке соседей



С таблицей finger



Узел 9: $10 \in (9, 12]$

Результат в обоих случаях: узел 12

Chord: присоединение нового узла

- Новый узел получает часть диапазона ключей следующего соседа.
- Ответственность за остальные ключи сохраняется.
- Ссылки соседей и таблицы finger обновляются постепенно.

Новый узел 6 между узлами 4 и 9

Фрагмент кольца · Стрелки — ссылки на следующего узла

До



После



За ключи 5 и 6 теперь отвечает узел 6

Показан результат присоединения после обновления ссылок

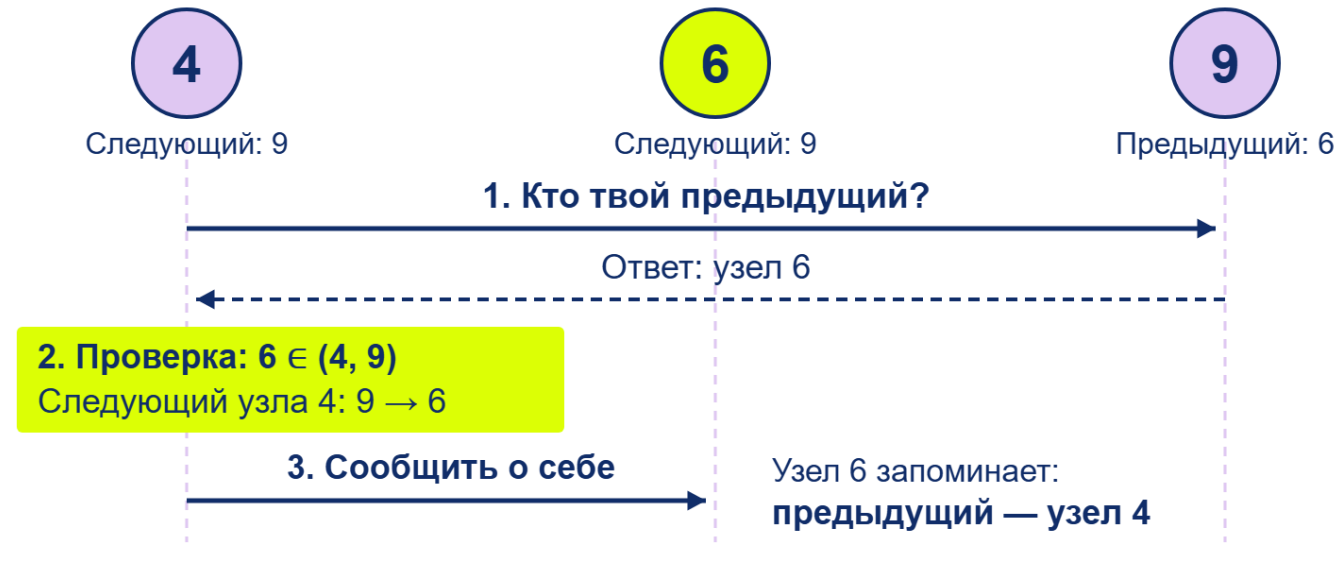
Chord: стабилизация кольца

Каждый узел периодически:

- Узнаёт предыдущего соседа своего следующего узла.
- Проверяет, не появился ли между ними новый узел.
- Уточняет ссылку и сообщает следующему соседу о себе.

Один цикл стабилизации на узле 4

Узел 6 уже сообщил о себе узлу 9 · Время идёт сверху вниз



Итог: следующие соседи $4 \rightarrow 6 \rightarrow 9$

Таблицы finger обновляются отдельной процедурой

Chord: если сосед не отвечает

- Узел хранит список следующих соседей.
- При недоступности соседа выбирает первый доступный узел из списка.
- Для сохранности значений нужна репликация данных.

Узел 4: список следующих узлов

6 **9** 12



Путь после переключения: 4 → 9 → 12

Для сохранения значений нужны реплики данных

Реестр сервисов, DNS и DHT

Подход	Задача и требования	Пространство имён	Архитектура
Реестр сервисов	Находить меняющиеся экземпляры, учитывать их состояние	Обычно плоское внутри каталога	Выделенная служба, логически единый каталог
DNS	Находить сведения в глобальном пространстве; независимо управлять его частями	Иерархическое	Делегирование зон, кэширование
DHT	Искать по точному ключу без центрального каталога; распределять нагрузку и данные	Плоское	Распределение ключей и маршрутизация между участниками

Подходы можно сочетать:

- Consul предоставляет DNS-интерфейс к реестру
- DHT можно использовать как основу службы поиска адресов по идентификаторам

Итоги: именование и поиск

- Имя обозначает объект, адрес указывает точку доступа.
- Способ генерации идентификатора определяет гарантии уникальности.
- Хеш содержимого позволяет проверить данные по имени.
- Структура имён и архитектура поиска – разные свойства системы.
- Обнаружение участников и обмен данными – разные задачи.
- При изменениях и отказах нужно поддерживать записи, кэши и маршруты.

Материалы

- **Именованние:** [Distributed Systems](#) (глава 6)
- **DNS:** [Computer Networking: A Top-Down Approach](#) (раздел 2.4)
- **DNS и P2P:** [Computer Networks: A Systems Approach](#) (разделы [9.3.1](#), [9.4.2](#)).
- **Имена и P2P:** [Distributed Systems: Concepts and Design](#) (главы 13 и 10)
- **Алгоритм Chord:** [оригинальная статья с SIGCOMM 2001](#) (разделы 4-5)
- **Корректность Chord:** [Reasoning about Identifier Spaces: How to Make Chord Correct.](#)
- **Подробнее о DNS:** [Addressing the Challenges of Modern DNS.](#)
- **Практика:** Consul – [регистрация сервисов](#), [обнаружение через DNS](#).

Спецификации

- **UUID:** [RFC 9562](#) – UUIDv4 (§ 5.4), UUIDv7 (§ 5.7).
- **DNS:** [RFC 1034 — основные понятия](#), [RFC 1035 — спецификация](#), [RFC 9499 — терминология](#).
- **Локальное разрешение имён:** [RFC 6762 – Multicast DNS](#).
- **Имена на основе хеша:** [RFC 6920 – Naming Things with Hashes](#).
- **BitTorrent:** [BEP 3 – основной протокол](#), [BEP 5 – DHT](#).