Распределенные вычислительные системы

CAP-теорема

Консенсус в PBC

Консенсус в РВС

- *Проблема консенсуса* в распределенной вычислительной системе состоит в том, как набору узлов РВС договориться о чем-либо – значении переменной, следующем действии или каком-либо другом решении:
  - решение, фиксировать транзакцию в базу данных или нет?
  - синхронизация часов для отслеживания общего времени
  - решение о переходе на следующую стадию распределенного алгоритма (применяется для реализации распределенной машины состояний)
  - выбор лидера для имплементации какого-либо высокоуровневого протокола

Консенсус в РСВ

- Будем говорить, что узел *определился с решением* в том случае, если он достиг своего решения о некотором значении это считает, что все остальные соглашаются с этим значением.
- Более формально, что каждый узел имеет *выходной регистр*, значение которого, после того как протокол прекращается, содержит значение для согласования. Запись в этот регистр является актом *принятия решения*.

Консенсус в РСВ

- Будем считать, что каждый узел имеет *выходной регистр*, значение которого содержит значение для согласования. Запись в этот регистр является актом *принятия решения*.
- Узлы *предлагают* значения для согласования.
- Будем считать алгоритм согласования корректным только в том случае, если:
  - *Договор* - все узлы РВС *решили выбрать* одно значение;
  - *Обоснованность* - значение, которое решили выбрать узлы РВС должно быть предложено некоторым узлом из РВС;
  - *Завершенность* - все узлы в конечном итоге должны прийти к решению.

CAP-теорема

Управление Данными в PBC

В любой сетевой системе, *обеспечивающей хранение совместно доступных данных*, разработчики хотят поддерживать следующие свойства:

- **Согласованность данных (Consistency)** – в системе существует единственная версия данных, соответствующая последней по времени операции обновления.
- **Доступность данных (Availability)** – запрос к PBC в любой момент времени должен завершиться корректным откликом, не зависимо от того, к какому серверу производится подключение.
- **Устойчивость к разделению (Partition tolerance)** - расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

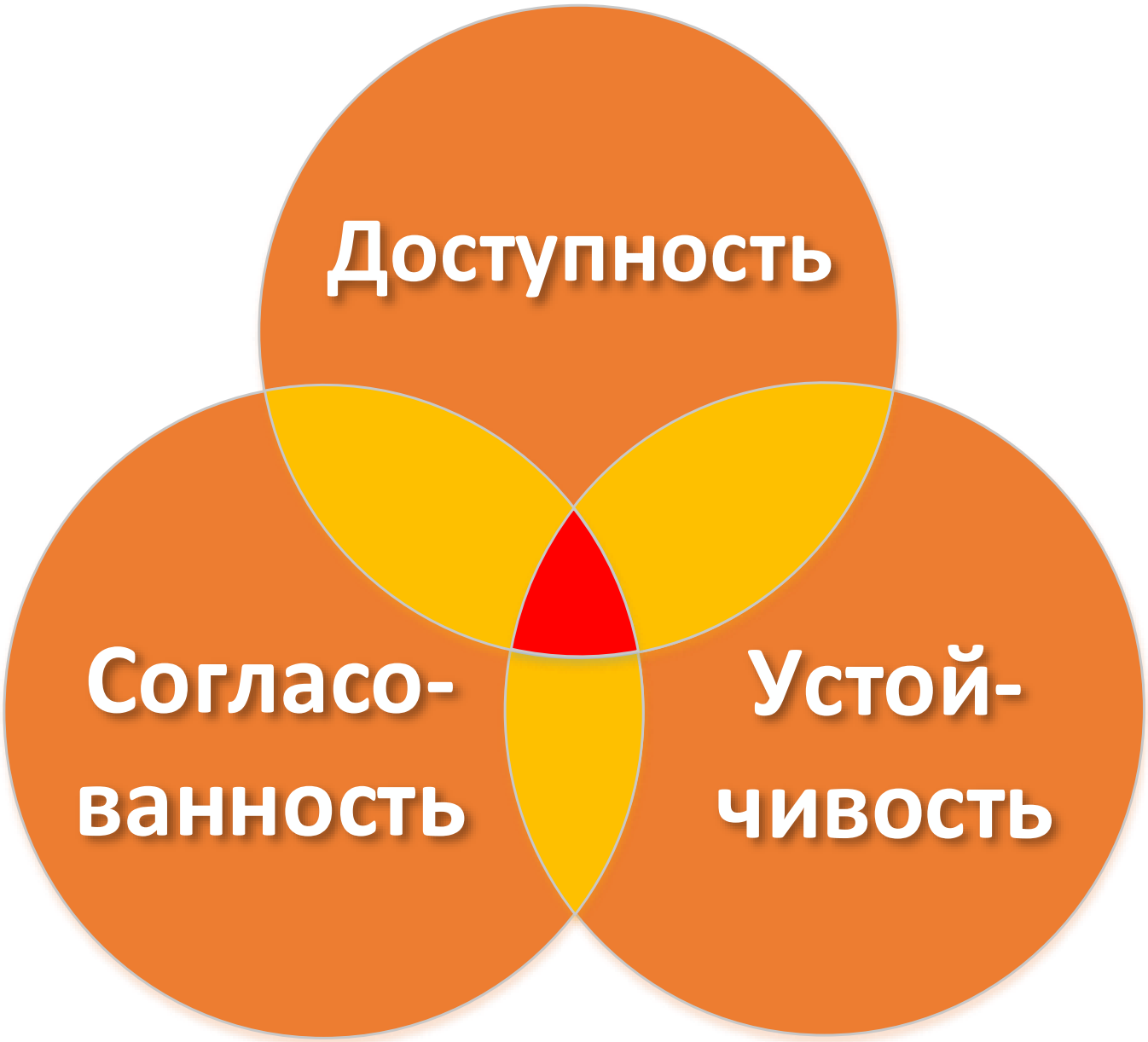
CAP-теорема

В 2000-м году Эрик Брюер (Eric Brewer – проф. Калифорнийского университета) предложил следующую теорему:

В любой сетевой системе, *обеспечивающей хранение совместно доступных данных*, одновременно могут поддерживаться только **два** из следующих трех свойств:

- **Согласованность данных (Consistency)**
- **Доступность данных (Availability)**
- **Устойчивость к разделению (Partition tolerance)**

В 2002-м году теорема была математически доказана в условиях отсутствия синхронизации (общих часов).


Доступность
Согласо-
ванность
Устой-
чивость

Что же такое Partition?

Разделение PBC это не только длительный разрыв между серверами, при котором один из них не может связаться с другим

» *Латентность сети также является разделением PBC.*

- Предположим, что у нас есть 2 сервера баз данных: один в России, другой в США.
- Они настроены на полное реплицированные
- Данные обновились на сервере в России. Через какое время сервер в США узнает об этом?
- **200 миллисекунд – лучший возможный результат** (худший возможный результат – никогда не узнает)
- В это «окно» они разделены – таким образом разделение системы происходит постоянно, даже в нормальных условиях работы PBC

Выбираем ли мы устойчивость?

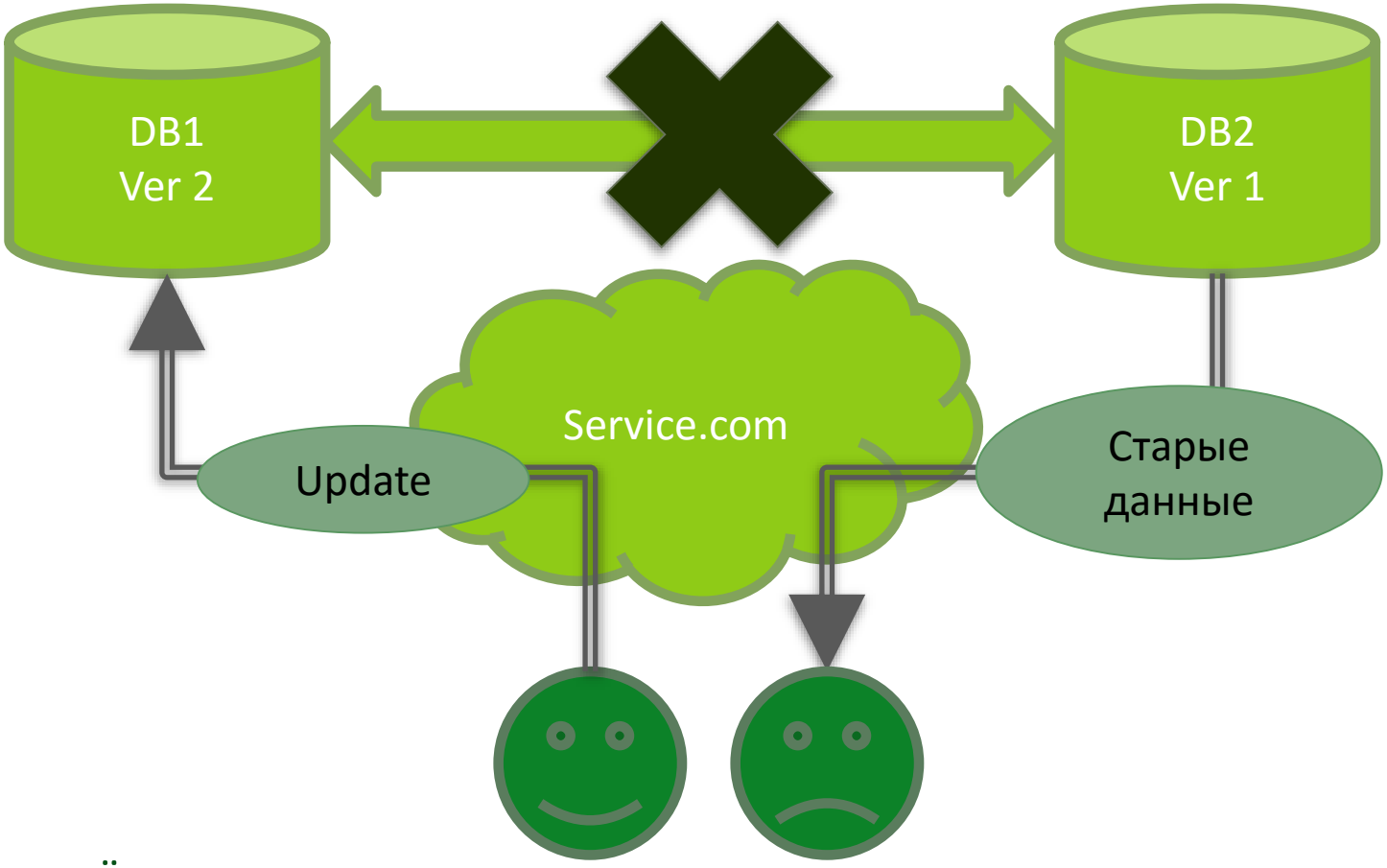
В реальном мире мы не можем выбирать, будут у нас сбои или нет. В РВС всегда будут возникать неполадки, зависит это от нас или нет:

- сетевые сбои,
- сбои работы оборудования,
- ошибки администрирования.

Плюс, любая латентность также вызывает разделение.

Поэтому приходится выбирать из двух вариантов:

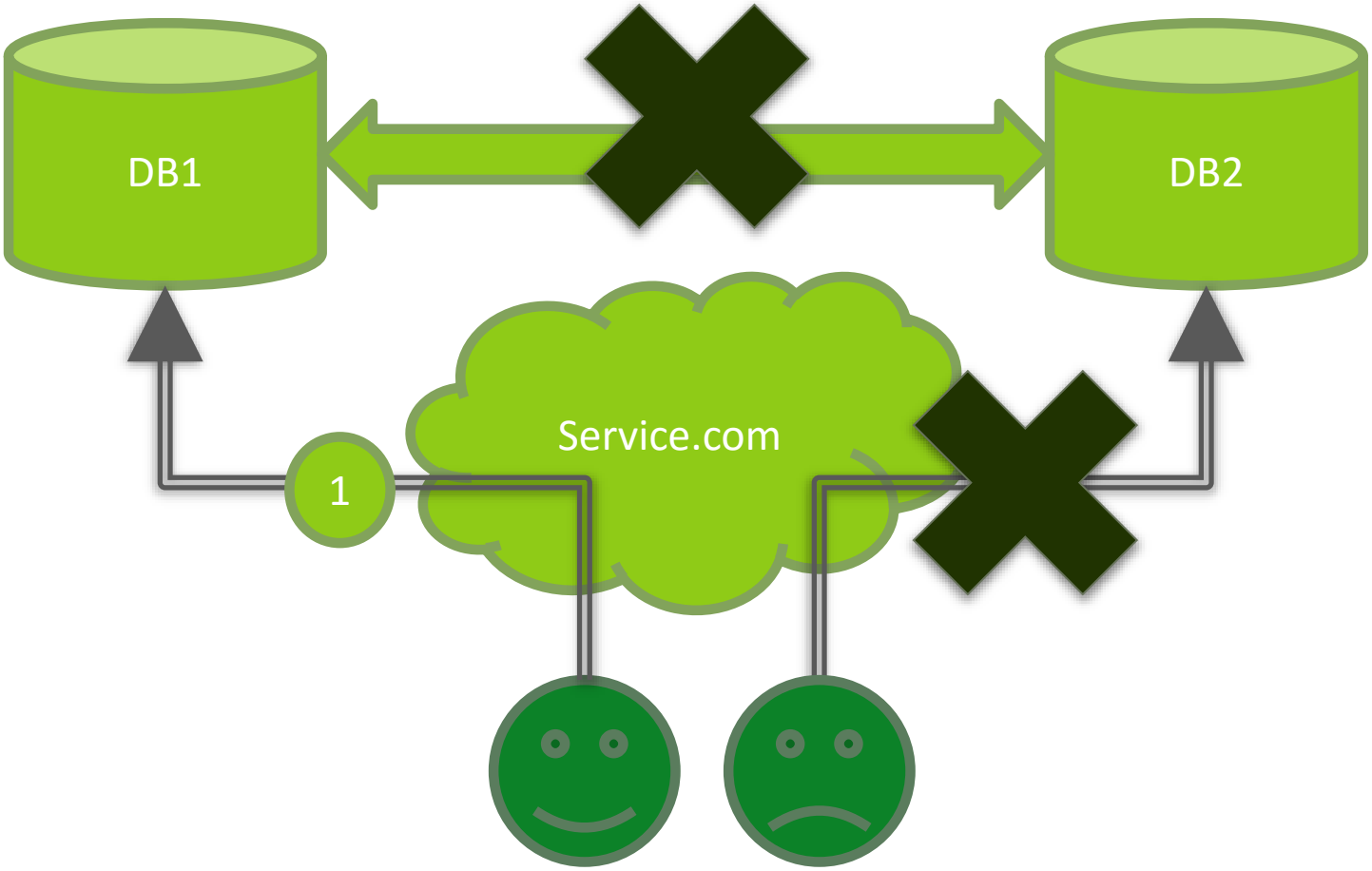
AP: 100% доступность, но несогласованность данных:



Распределённая система, отказывающаяся от целостности результата. Большинство NoSQL-систем принципиально не гарантируют целостности данных («целостные в конечном итоге» - eventually consistent).

CAP

CP: 100% согласованность, но недоступность данных при распаде:

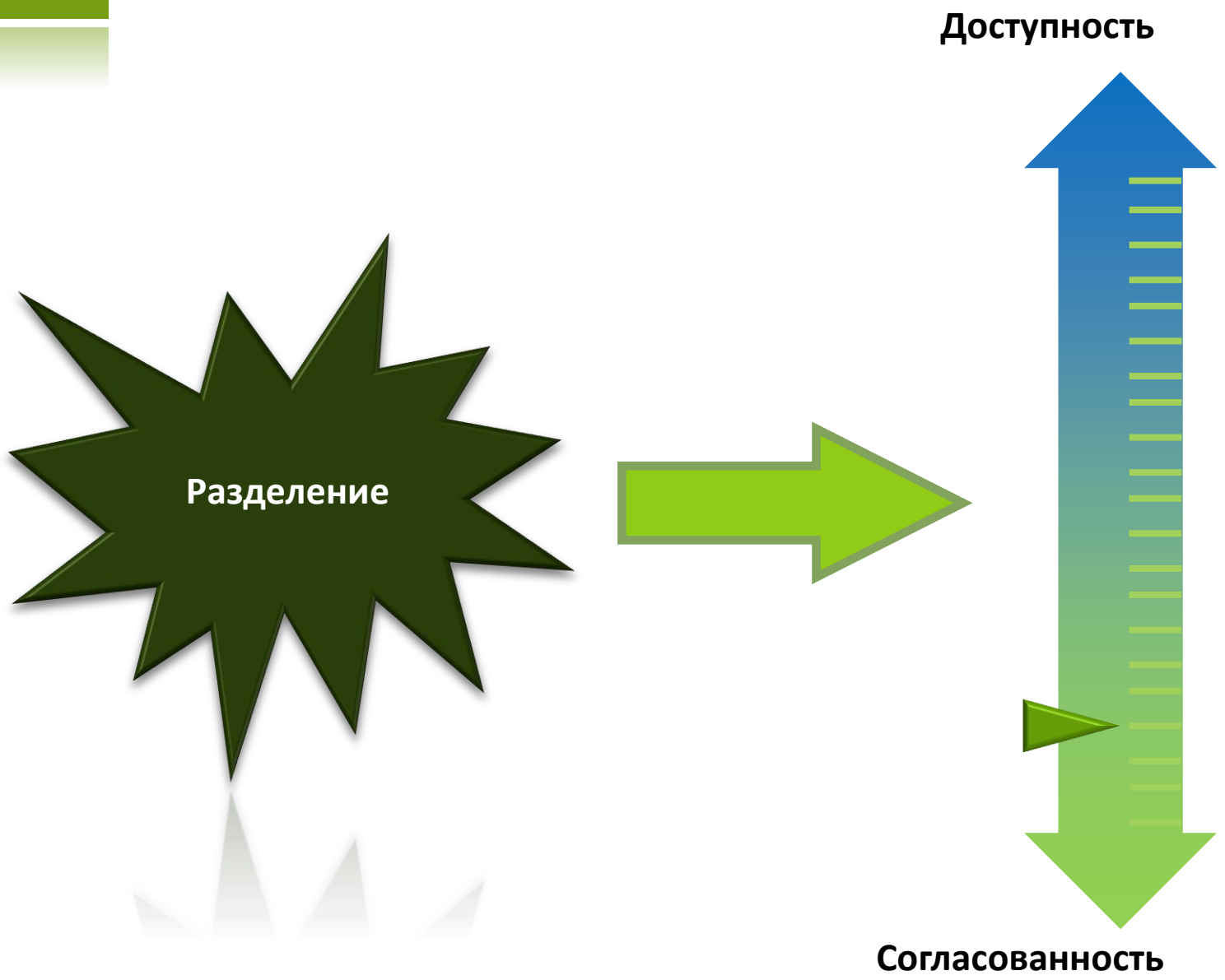


Распределённая система, в каждый момент обеспечивающая целостный результат и способная функционировать в условиях распада, в ущерб доступности может не выдавать отклик. Пессимистические блокировки для сохранения целостности.

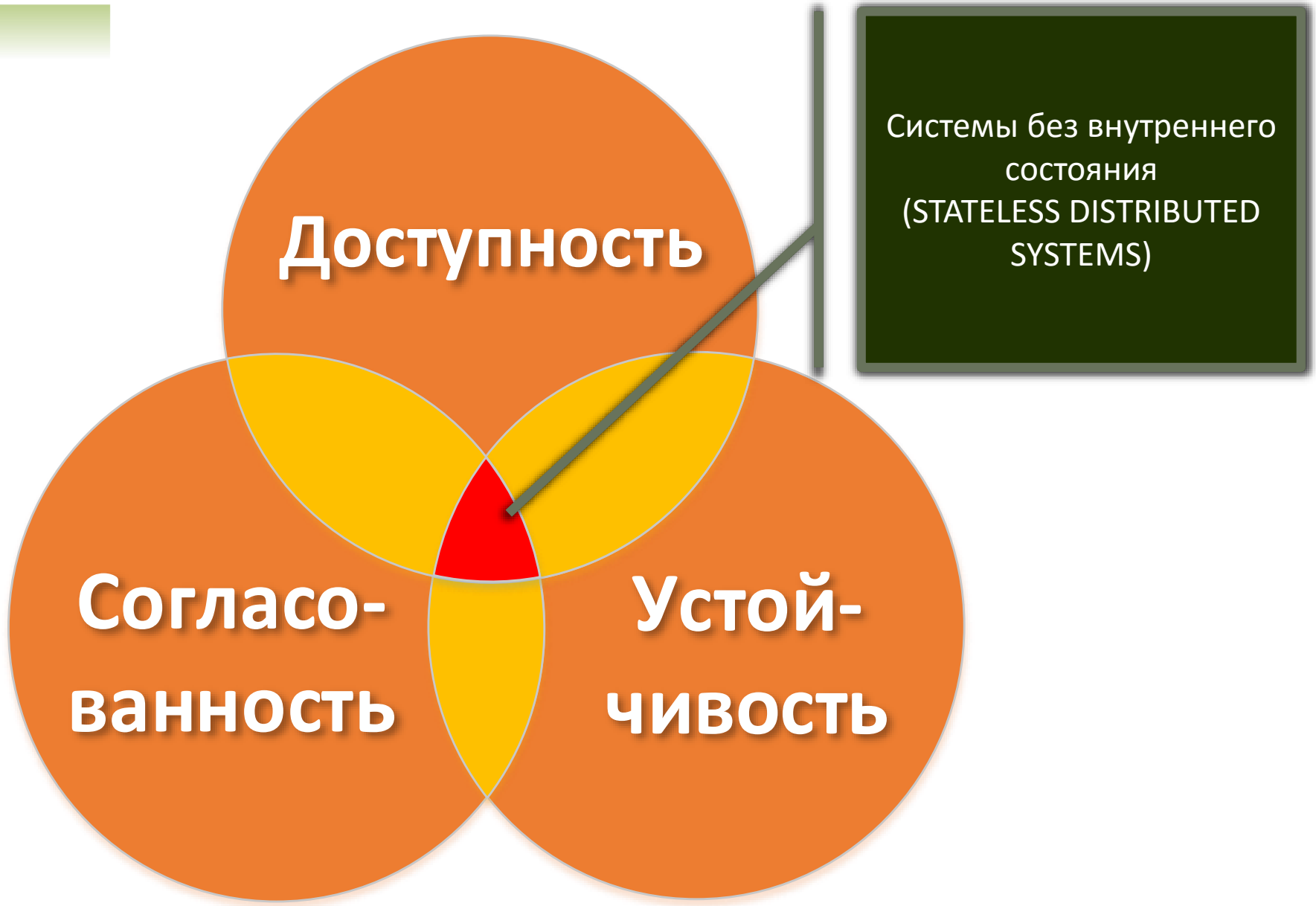
Выбор между C и A

- Нет 100% работающего универсального решения, каким путем решать задачу обработки запросов при возникновении расщепления распределенной системы – выбирать доступность или же выбирать
- Этот выбор может быть сделан только разработчиком, на основе анализа **бизнес-модели** приложения.
- Более того, этот выбор может быть сделан не для всего приложения в целом, а отдельно для каждой его части:
  - *Согласованность* для процедуры покупки и оплаты в интернет-магазине;
  - *Доступность* для процедуры просмотра каталога товаров, чтения отзывов и т.п.
- Или же смириться с несогласованностью, в случае коротких периодов (1-2 минуты) разделения системы

Доступность и согласованность



Исключение из CAP- теоремы?



Eventual Consistency

В связи с невозможностью 100% времени поддерживать согласованность и доступность системы, пришлось искать компромисс.

- *Согласованность в конечном счете (eventual consistency) или слабая согласованность (weak consistency)* - означает, что если в течение достаточно долгого периода времени в систему не поступают новые операции обновления данных, то можно ожидать, что результаты всех предыдущих операций обновления данных в конце концов распространятся по всем узлам системы, и все реплики данных **согласуются**.
- При отсутствии сбоев, максимальный размер окна несогласованности может быть определен на основании таких факторов, как задержка связи, загруженность системы и количество реплик в соответствии со схемой репликации.
- Самая популярная система, реализующая «согласованность в конечном счете» – DNS. Обновленная запись распространяется в соответствии с параметрами конфигурации и настройками интервалов кэшированя. В конечном счете, все клиенты увидят обновление.

Более подробно о согласованности в конечном итоге: http://habrahabr.ru/post/100891/