Распределённые алгоритмы

mk.cs.msu.ru → Лекционные курсы → Распределённые алгоритмы

Блок 3

Иллюстрация трудности разработки распределённых алгоритмов: начало

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2025, февраль–май

Условие задачи

Рассмотрим такую **задачу**: требуется передать данные d от узла A к узлу B по каналу связи C

Положим, что сообщения, пересылаемые через C, имеют следующую структуру:

- $(\mathbf{t}, x)$: тип $\mathbf{t}$ и данные x — или
- $\mathbf{t}$: тип $\mathbf{t}$ без данных

Будем считать, что канал C всегда готов к пересылке сообщений, но при этом ненадёжен: сообщение может

- доставляться сколь угодно долго
- потеряться без доставки (но при многократной отправке рано или поздно будет доставлено)

Условие задачи

Положим, что

- до начала и после завершения своих частей передачи A и B не хранят никакую информацию о передаче сообщения — начинают выполнение в фиксированном начальном состоянии $(\square_A, \square_B)$
- каждый узел может выйти из строя с последующим перезапуском в начальном состоянии

Завершение узлом своей части передачи данных d обозначим выполнением команды $\mathrm{done}(d)$:

- Для узла A это обозначение уверенности в том, что данные d доставлены
- Для узла B это обзначение того, что данные d приняты
- Можно считать эту команду, например, оповещением следующего уровня модели OSI о завершении передачи

Условие задачи

Будем говорить, что по завершении пересылки данные

- потеряны, если A уверен, что эти данные доставлены, но B ни разу их не принимает
- дублированы, если B более одного раза принимает эти данные

Пересылку данных будем считать надёжной, если передача данных завершается без их потери и дублирования

Гарантированно надёжная пересылка данных в условиях выхода узлов из строя невозможна

Чтобы в этом убедиться, достаточно представить себе два сценария пересылки, начинающиеся с таких фрагментов:

1. A отправляет данные; B принимает сообщение с данными и выходит из строя непосредственно **перед** выполнением done_B
2. A отправляет данные; B принимает сообщение с данными и выходит из строя непосредственно **после** выполнения done_B

Выполнение узлов A и B в этих двух сценариях одинаково с точностью до обозначенного выполнения done_B

Следовательно:

- Если пересылка по сценарию 1 надёжна, то в сценарии 2 данные дублируются
- Если пересылка по сценарию 2 надёжна, то в сценарии 1 данные теряются

Попробуем задуматься над тем, как можно было бы устроить «разумное», хотя и не гарантированное надёжное решение поставленной задачи или её упрощённого варианта

Для начала положим, что узлы не выходят из строя и сообщения не теряются

Тогда схема (протокол) передачи d от A к B по каналу C может быть устроена очень просто:

- A отправляет d в C и завершает отправку
- B ожидает появления d в C и принимает эти данные

Как описать алгоритм, отвечающий такой схеме взаимодействия?

Для этого потребуется псеводкод, подходящий для описания действий, производимых узлами согласно распределённому алгоритму

Как можно, имея описание алгоритма, убедиться, что этот алгоритм устроен **правильно**?

В этом курсе остановимся на таких методах проверки правильности, которые могут быть применены без реализации алгоритма и при этом обеспечивают строгость анализа, позволяющую достоверно рассуждать о свойствах и правильности алгоритма