Д.В. Шкурко

ОТКАЗОУСТОЙЧИВОСТЬ В РАСПРЕДЕЛЕННЫХ СЕТЯХ: ПРОБЛЕМЫ КОНСЕНСУСА

1. ПРОБЛЕМЫ СОГЛАШЕНИЯ

Распределенной системой будем считать некоторое множество активных компонентов системы — *процессов*, взаимодействующих посредством компонентов связи. Отказоустойчивость распределенной системы определяется возможностью функционирования системы, несмотря на отказы ограниченного числа ее компонентов. Используемые для этого алгоритмы связаны с взаимодействием различных ее частей. Одной из фундаментальных проблем является соглашение о значении каких-либо входных данных. Например, управление базой данных в распределенных системах требует достижения соглашения относительно совершения (commit) или отбрасывания (abort) транзакции. Другим примером служит определение значения, считываемого датчиками. Еще один образец применения дают алгоритмы синхронизации часов, если каждый процесс не имеет доступа к централизованной службе синхронизации времени.

Естественным решением проблемы является следующий сценарий. Процессы сообщают о своем предложении и затем выбирают значение, встречающееся чаще всего. Этот метод хорошо работает, если нет ошибок, но не приемлем, если процессы могут сообщать разные данные различным процессам и влиять таким образом на правильные процессы, заставляя их принимать разные решения.

Простая постановка задачи состоит в соглашении о значении одного бита. Предположим, что имеется распределенная система, состоящая из фиксированного множества процессов, часть которых или изначально некорректны или могут давать сбой во время исполнения алгоритма. Каждый процесс i имеет входную бинарную переменную x_i. Задача *консенсуса* состоит в достижении соглашения относительно некоторого значения y. Более точно, требуется, чтобы все корректные процессы i когда-нибудь закончили исполнение алгоритма со значением переменной $y_i = y$. Это требование называется требованием *соглашения*.

Значение y в общем случае будет зависеть от начальных значений x_i, в противном случае имеется тривиальное решение: каждый процесс

присваивает $y_i = 0$ и останавливается. Рассматриваются несколько требований зависимости y от значений x_i в порядке усиления.

- Нетривиальность. Для любого $y \in \{0, 1\}$ должно существовать допустимое исполнение, ведущее к соглашению с результатом y. Допустимые исполнения могут иметь ограничения на количество ошибок, на тип системы (синхронная или асинхронная) и т.п.
- Слабое соглашение. Если $x_i = x \in \{0, 1\}$ для всех i, то $y = x$ при условии, что во время исполнения не было ошибок.
- Сильное соглашение. Если $x_i = x \in \{0, 1\}$ для всех корректных i, то $y = x$.

Требования зависимости называют еще требованием *обоснованности (validity).*

Имеются две другие проблемы, тесно связанные с алгоритмами консенсуса и которые широко рассматривались в литературе. Первая проблема, называемая проблемой *соглашения византийских генералов* (коротко *проблемой генералов*) или проблемой *отказоустойчивой рассылки*, предполагает наличие выделенного процесса ("генерала" или "отправителя"), который пытается разослать входное значение x всем остальным. Как и раньше все корректные процессы должны достичь соглашения при одном условии зависимости результата от x.

- Слабая зависимость: $y = x$, если во время выполнения алгоритма не было ошибок.
- Сильная зависимость: $y = x$, если "генерал" работает корректно.

В дальнейшем, если не оговорено обратное, используется строгая зависимость.

Вторая проблема будет рассмотрена в разд. 3. Эта проблема рассматривалась в ранней работе [1] и легко сводится к проблеме соглашения генералов [2]. Поэтому эта проблема практически не встречается в более поздних работах.

2. МОДЕЛЬ ВЫЧИСЛЕНИЙ

Решения задач соглашений, которые могут быть получены, зависят существенно от предположений, сделанных относительно моделей вычислений и взаимодействий и относительно допустимых ошибок/сбоев, устойчивость к которым требуется от решения. Далее предполагается, что число процессов фиксировано и равно N. Протокол называется *t-устойчивым,* если он корректно работает при условии, что не более t процессов дают сбои в течение работы алгоритма (процесс также может

давать сбой, просто не участвуя в протоколе).

Рассматриваются несколько моделей ошибок.

Остановка с сигналом о сбое (fail-stop). Процесс прекращает работу и сообщает о своем сбое другим процессам [3]. В этой модели можно точно определить, сломался ли процесс или нет. Эта модель предоставляет самые удобные для программирования предположения относительно сбоев в системе [4].

Остановка (crash) происходит, если процессор преждевременно прекращает всю активность. Вплоть до остановки процессор работает корректно, а после остановки он полностью останавливает работу. Не предполагается, что процессор восстанавливается после остановки и включается в работу системы.

Потеря части сообщений (omission). Потери сообщений могут происходить как при отсылке, так и при отправлении сообщений [5].

Ошибка согласования времени (timing fault). Происходит, когда процесс заканчивает задачу раньше или позже указанного срока. Такие ошибки существенны в задаче синхронного старта, тесно связанной с задачей соглашения [6, 7].

Наиболее серьезными ошибками являются *византийские ошибки*, при этом не делается никаких предположений относительно поведения некорректного процесса. Например, он может посылать лишние или противоречивые сообщения, не работать какое-то время и т.д. Протокол, который позволяет обойти такие ошибки, называется *византийским протоколом.*

Чтобы показать устойчивость к произвольным ошибкам, требуется рассмотреть все возможные поведения процессора, включая случаи, когда некорректный процесс пытается помешать исполнению протокола. Это может показаться очень сильным требованием, однако в отсутствии точного описания ошибок принятие консервативного подхода является подходящим решением.

Предполагается, что система передачи сообщений полностью отказоустойчива и только процессы могут давать сбои. Далее предполагается, что каждый получатель может безошибочно определить отправителя сообщения и все сообщения доставляются без изменений и ошибок. Если не оговорено обратное, предполагается, что процессы взаимодействуют попарно и любые два процесса могут взаимодействовать непосредственно.

Конечно, в реальных системах система сообщений, так же как и

процессы, могут давать сбои. Поэтому сбой канала обмена сообщениями обычно приписывается одному из процессов на одном из концов канала. Таким образом, t-устойчивый протокол позволяет обойти до t ошибок процессов или каналов связи.

Делались попытки более точно характеризовать ошибки в каналах связи. Например, более точное описание отказоустойчивости в случае ошибок-остановок и в случае потерь сообщений с разделением ошибок процессов и ошибок каналов связи есть в [8]. Введение ошибок, связанных с потерей сообщений [5], тоже было сделано с целью, чтобы более аккуратно описывать ошибки при передаче сообщений.

Очень важным предположением о поведении системы является возможность обнаружения ошибки процесса: было отослано сообщение или произошел сбой при отсылке сообщения. В этом случае получатель имеет очень важное знание о некорректности отправителя. В модели с аккуратными часами и ограничением на время доставки сообщения это достигается посредством тайм-аутов (ср. [9]). Таким же образом ошибки при отсылке автоматически обнаруживаются в синхронной модели, в которой вычисления и обмен сообщениями происходит по шагам: каждый шаг состоит из рассылки сообщений процессом, получения всех сообщений, адресованных процессу и отосланных на текущем шаге, и локальных вычислений. Однако, обнаружение невозможно в полностью асинхронной модели, в которой не делается никаких предположений об относительной скорости процессов и о времени доставки сообщений, так как в этом случае нельзя отличить медленно работающий процесс от остановившегося. Этот момент во многом определяет разрешимость задач соглашения.

Далее, если не оговорено обратное, будут использоваться термины *синхронная* и *асинхронная* системы для различения этих двух крайних случаев. Предполагается, что в синхронной модели посылка сообщений происходит раньше, чем прием сообщений, и поэтому содержимое и адресаты рассылаемых сообщений не зависят от содержимого сообщений, полученного в текущем раунде.

Другим существенным предположением является наличие подписей. Предполагается, что автор подписанного сообщения может быть определен любым процессом, независимо от того, получено ли сообщение от автора (или опосредованно) и независимо от действий некорректных процессов (если автор корректный). Другими словами, если корректный процесс A получил сообщение от B, подписанное корректным процес-

сом C, то A может быть уверен в том, то C действительно отсылал сообщение и B не мог подделать. Наличие подписей также во многом определяет разрешимость задач соглашения.

Цифровые подписи или простой контроль четности могут быть использованы в качестве подписей в зависимости от приложения: достаточно, чтобы некорректный процесс не мог сгенерировать сообщение корректного. В случае ошибок остановок или потерь сообщений, очевидно, механизм подписей не нужен.

С точки зрения применения для оценки протоколов соглашения используются различные оценки сложности алгоритмов. Среди важных характеристик считаются *время*, необходимое для достижения соглашения, и *количество и длина сообщений*, отсылаемых всеми корректными процессами во время исполнения протокола. Все эти характеристики в общем случае зависят от того, какие сбои происходит и когда.

В синхронной системе время измеряется в количестве раундов обмена сообщениями. Предполагается, что в течение одного раунда каждый процесс имеет возможность обменяться сообщениями с любым другим. Относительно пересылаемых сообщений интересуются также общим числом пересланных битов и количеством пересланных подписей (в случае протоколов с подписями).

3. СВЯЗЬ МЕЖДУ ПРОБЛЕМАМИ СОГЛАШЕНИЯ

Кроме задачи соглашения генералов рассматривалась задача *взаимной согласованности (interactive consistency)*. Эта проблема состоит в построении общего вектора $\mathbf{y}$ и для нее рассматривались следующие требования к зависимости компонент вектора от начальных значений x_i.

- Слабая зависимость: для всех j $\mathbf{y}_j = x_j$, если во время исполнения алгоритма не было ошибок.
- Сильная зависимость: для всех корректных процессов j $\mathbf{y}_j = x_j$.

Видно, что проблема соглашения генералов является специальным случаем проблемы взаимной согласованности, в которой значение только одного процесса представляет интерес, т.е. решение задачи взаимной согласованности дает решение задачи соглашения генералов. С другой стороны N параллельных исполнений протокола задачи соглашения генералов дают протокол для задачи взаимной согласованности.

Задачу консенсуса сложнее напрямую сравнивать с другими задачами соглашения. С одной стороны, она может быть решена с помо-

щью задачи взаимной согласованности выбором самого часто встречающегося значения в результирующем векторе. При условии, что число некорректных процессов меньше $N/2$ получается решение сильного консенсуса, в противном случае получается решение слабого консенсуса. С другой стороны, для сведения задачи соглашения генералов к задаче консенсуса требуется дополнительный раунд на рассылку начального значения выделенным процессом. Несложно показать, что построенный таким образом протокол решает задачу соглашения генералов. Требование соглашения выполняется тривиально, а требование зависимости получается с помощью следующего замечания. Если генерал корректный, то в начале протокола консенсуса все корректные процессы будут иметь одинаковое входное значение и по требованию зависимости для задачи консенсуса все корректные генералы в качестве финального значения примут начальное значение.

Многие алгоритмы соглашения генералов имеют в качестве первого шага рассылку начального значения и имеют в качестве встроенного алгоритм консенсуса. Однако встроенный алгоритм не всегда решает полную задачу консенсуса. Например, решение задачи консенсуса из [10] использует тот факт, что наличие разных входных значений у всех процессов после рассылки выделенным процессом означает некорректность выделенного процесса и задача консенсуса решается с ограничением в $t-1$ для числа некорректных процессов. В результате модификация алгоритма соглашения для решения задачи консенсуса требует даже большего числа раундов: $2t+4$ вместо $2t+3$.

Похожие соображения работают и для слабых версий соглашений. Очевидно, что сильные версии дают решения слабых версий задачи, но нет явной схемы преобразования решения слабого соглашения к решению сильного соглашения.

Более того, для “приблизительного” соглашения в слабом случае решение существует, а для сильного нет [11]. См. также разд. 4.

4. ПРИБЛИЗИТЕЛЬНЫЕ И НЕТОЧНЫЕ СОГЛАШЕНИЯ

Кроме задач точного консенсуса, рассматривались некоторые задачи приблизительного и неточного консенсуса. Предполагается, что каждый процесс i имеет входное действительное значение x_i.

Один из вариантов проблемы консенсуса позволяет показать, что слабая зависимость накладывает существенно меньше ограничений по сравнению с сильной зависимостью. Требование соглашения состоит в

том, чтобы в конце работы алгоритма все получили значение y_i, удовлетворяющие условию $|y_i - y_j| < \epsilon$ для любых корректных процессов i и j. Рассматриваются следующие требования зависимости.

- Сильная зависимость. Если все корректные процессы имеют одинаковое входное значение y, то все корректные процессы примут значение y.
- Слабая зависимость. Если все процессы имеют одинаковое входное значение y и во время исполнения протокола не происходит ошибок, то все процессы процессы примут значение y.

При условии, что множество входных значений ограничено, в работах [11, 2] показано, что проблема со слабой зависимостью разрешима при $N \leq 3t$, а с сильной зависимостью нет.

Другая более содержательная проблема консенсуса рассматривалась в работе [12]. Требование зависимости в этой проблеме усилено: выходное значение процесса должно лежать между входными значениями корректных процессов. Решение этой задачи основано на обмене текущими значениями, отбрасывании крайних значений (t максимальных и t минимальных) и вычислении некоторой статистики (усреднения). На основе результатов [12] в [13] построен алгоритм синхронизации часов. Требование приблизительности консенсуса позволило построить решения в полностью асинхронных системах при условии $N > 5t$.

В работе [14] описано более слабая проблема неточного соглашения для применения в алгоритме синхронизации часов. Предполагается, что каждый процесс i имеет входное действительное значение x_i, удовлетворяющее требованию начальной точности $\max_{i,j \in G} |x_i - x_j| \leq \delta$ и начальной аккуратности $\max_{i \in G} |x_i - \bar{x}|$. G — множество корректных процессов. Требуется улучшить начальную точность (в применении к синхронизации часов — ограничить разброс локальных часов).

Невозможность решения задачи неточного и приблизительного соглашения при $N \leq 3t$ есть в работе [15].

Теорема 1. *Задачи приблизительного и неточного соглашения имеют решения без использования подписей тогда и только тогда, когда* $N > 3t$.

5. РАЗРЕШИМОСТЬ ЗАДАЧ СОГЛАШЕНИЯ

5.1. Синхронные системы

Базовой проблемой при построении алгоритмов соглашения является вопрос о существовании решения. По соображениям, изложенным в разд. 3, t-устойчивые решения проблем консенсуса и взаимной согласованности имеют решения тогда и только тогда, когда есть t-устойчивое решение задачи соглашения генералов. Поэтому можно ограничиться рассмотрением только задачи соглашения генералов.

В синхронном случае в [1, 2] показано, что ограничений для t нет.

Теорема 2. *В синхронных системах с использованием подписей можно построить протокол для решения t-устойчивой сильной (слабой) задач соглашения генералов при $t \leq N$.*

На рис. 1 представлено решение с использованием подписей, оно проще и эффективнее первого решения [1, 2]. Впоследствии это решение было улучшено в [16, 17].

Доказательство корректности алгоритма достаточно простое. Если G корректный, то все корректные процессы могут извлечь только одно значение m, подписанное G. Если процесс извлек первое или второе значение во время раунда $i < t + 1$, то все корректные процессы тоже извлекут это значение в раунде $i + 1$. Наконец, если корректный процесс извлек второе значение в раунде $t + 1$, то среди сообщений, вызвавших его сделать это, есть сообщение от корректного процесса q, и этот процесс дает возможность извлечь это значение всем остальным корректным процессам в раунде $i \leq t + 1$.

Замечание. Видно, что если процесс G дал сбой и послал сообщение m не всем процессам в первом раунде, то этот сбой не отразится на конечном решении процессов, и процессы считают, то G некорректный, только если он отсылает противоречивые сообщения.

Без использования подписей решение существует при достаточно большой избыточности системы.

Теорема 3. *В синхронных системах можно построить протокол без использования подписей для решения t-устойчивой сильной (слабой) задач соглашения генералов тогда и только тогда, когда $N > 3t$.*

Невозможность для сильного соглашения при $N \leq 3t$ была получена в [1, 2], а для слабого соглашения — в [11]. Решение для $N > 3t$ было

получено в [1, 2].

На рис. 2 приведено описание алгоритма из [1, 2] в той форме, которая была представлена в работе [19] и потом многократно использовалась в более поздних работах [20, 21, 22, 23, 24, 25]. Доказательство корректности алгоритма опирается на следующие два факта.

1. Если процесс p корректный, то при переразметке дерева узел σp сохраняет свое значение, которое (вследствие корректности p) одно и то же в узлах σp всех корректных процессов.
2. Если большинство наследников узла при переразметке получили одно и то же значение в деревьях всех корректных процесов, то и сам узел получит одно и то же значение в деревьях всех корректных процесов.

Замечание. Построенный алгоритм легко модифицируется в алгоритм консенсуса. В этом случае корень дерева хранит входное значение процесса и имеет пустую метку, построение дерева продолжается $t + 1$ раунд. Правила переразметки те же самые.

5.2. Асинхронные системы

В полностью асинхронном случае решения не существует. Более того, даже существенное усиление ограничений на поведение некорректных процессов не позволяет получить решение [26].

Теорема 4. *В полностью асинхронном случае не существует протокола для решения задачи консенсуса с одним некорректным процессом и с допустимыми ошибками, являющимися остановками. Результат верен, если предполагать нетривиальность зависимости от входных данных.*

Идея доказательства теоремы состоит в следующем.

1. Изначально каждый процесс потенциально может принять любое решение, как 0, так и 1.
2. Чтобы выбрать окончательное решение процесс должен получать сообщения от других процессов, при этом полученные сообщения могут как склонять процесс к принятию решения, так и оставлять его дальше перед дилеммой.
3. Показано, что если процесс еще не склонился ни к какому решению, то существуют приемы таких сообщений, которые не помогут принять решение. Таким образом, возможно бесконечное

Предусловие: Выделенный процесс G имеет значение m, которое он должен разослать остальным
Постусловие: Все процессы достигают соглашения генералов
/* Запись (q, m) означает сообщение m подписанное процессом q */
/* Первый раунд */
1 Процесс G рассылает сообщение (G, m)
2 Остальные процессы получают сообщение и извлекают из него m.
3 $temp \leftarrow m$
/* 2 - $(t+1)$ раунды (процесс (q, m)) */
4 **for** $i \leftarrow 2$ **to** $t+1$ **do**
5 **if** *temp первое или второе извлеченное значение, подписанное* G **then**
6 разослать $(q, (G, temp))$ и разослать доказательство (i сообщений $(p, (G, m))$ или (G, m))
7 **end**
8 **if** *Если получены сообщения* $(p, (G, m))$ *или* (G, m) *от* i *разных процессов в предыдущих раундах* **then**
9 извлечь m
10 $temp \leftarrow m$
11 **end**
12 **end**
/* Решение */
13 **if** *Если извлечено только одно значение* m **then**
14 принять m
15 **else**
16 G некорректный
17 **end**

Алгоритм 1: Алгоритм соглашения с использованием подписей [18]

Предусловие: Выделенный процесс G имеет значение m, которое он должен разослать остальным

Постусловие: Все процессы достигают соглашения генералов

`/* Алгоритм состоит в построении помеченного дерева с последующей переразметкой. Узлы дерева соответствуют последовательностям` σ `имен процесов без повторений */`

`/* Первый раунд */`

1 G сохраняет в корне G дерева значение m и рассылает значение m остальным процессам. Остальные процессы получают сообщение m от G и тоже сохраняют в корне дерева G значение m

`/* 2 -` $(t+1)$ `раунды (процесс` (q, m)`) */`

2 **for** $i \leftarrow 2$ **to** $t+1$ **do**

3 Процесс q рассылает значения, сохраненные в листьях σ построенного дерева

4 Процесс q получает сообщения от других процессов и достраивает дерево: значение для узла σ, полученное от процесса $p \notin \sigma$, сохраняется в новом листе σp

5 **end**

`/* Решение. */`

6 Значения в листьях дерева сохраняются, а во внутренних узлах заменяются рекурсивно от листьев к корню. В качестве нового значения для внутреннего узла выбирается значение, которое встречается в более чем половине наследников (или просто выбирается 0, если 0 и 1 поровну).

7 В качестве финального значения берется значение в корне дерева.

Алгоритм 2: Алгоритм соглашения без использования подписей [19]

откладывание приема ключевых сообщений за счет приема сообщений, которые не влияют на прогресс всего алгоритма.

В работе [27] был предпринят тщатетльный разбор доказательства из [26]. Были разобраны следующие предположения относительно системы:

- процессы синхронные или асинхронные, т.е. гарантирована или нет скорость исполнения;
- доставка сообщений синхронная или асинхронная, т.е. гарантировано или нет время доставки;
- порядок получения сообщений произвольный или в порядке отправления;
- отсылка сообщений только одному процессу или широковещательная рассылка;
- атомарная обработка сообщений (получить–обработать–отправить) или неатомарная.

Были выделены 4 минимальных разрешимых случая консенсуса с ошибками-остановками:

1) синхронное отправление сообщений и синхронные процессы (стандартные предположения в синхронной модели);
2) синхронные процессы и получение сообщений в порядке отправления;
3) широковещательная рассылка и получение сообщений в порядке отправления;
4) синхронная доставка сообщений, широковещательная рассылка и атомарная обработка сообщений.

Новые разрешимые случаи не получили большого внимания исследователей, известно только о работе [28], посвященной разработке последнего разрешимого случая.

В работе [29] рассматривались другие ослабления синхронности системы. Были исследованы случаи частичной синхронности доставки сообщений и частичной синхронности работы процессов. Под частичной синхронностью подразумевается следующее. Предполагается, что есть оценка скорости передачи сообщений или скорости работы процессов, но либо она верна только начиная с некоторого момента, либо она не известна процессам и не должна быть явно использована в алгоритме. Для всех вариантов частичной синхронизации можно построить алгоритм, но при этом необходима дополнительная избыточность, например, для

остановок при частично синхронных обменах требуется $N > 2t$ а для произвольных ошибок с использованием подписей требуется $N > 3t$ (в полностью синхронном случае системе достаточно в обоих случаях $N > t$).

6. РАСШИРЕНИЕ МНОЖЕСТВА ВХОДНЫХ ЗНАЧЕНИЙ

В предыдущих главах рассматривались проблемы соглашения генералов и проблемы консенесуса для случая, когда входные значения лежат в двухэлементном множестве $\{0, 1\}$. Довольно часто, однако, требуется достичь консенсуса относительно значения из произвольного конечного множества. Наиболее простое решение состоит в соглашении относительно $\lceil \log V \rceil$ битов, где $V = |M|$ мощность множества входных значений, этот метод сохраняет число раундов, но увеличивает пропорционально количество сообщений.

Другой метод состоит в введении двух дополнительных раундов [30]. Этот метод работает при условии $N > 3t$ и не требует использования подписей. В первом руанде процессы обмениваются входными значениями m_i. Если процесс уверен, что большинство корректных процессов имеют одно и то же входное значение и некорректные процессы не могут привести к противоречивому решению другой корректный процесс (т.е. получено больше $\lfloor (N+t)/2 \rfloor$ одинаковых значений), то процесс отсылает во втором раунде это значение n_i, в противном случае отсылается значение по умолчанию n'. Во втором раунде корректный процесс, который получил сообщение n_i от корректного процесса (т.е. получил больше t сообщений n_i), использует в качестве входного значения для бинарного алгоритма значение 1, иначе 0. В конце работы алгоритма бинарного соглашения, если результат равен 0, принимается значение по умолчанию, иначе значение n_i. Доказательство достаточно простое. Принципиальными моментами являются

- требование, чтобы корректные процессы не принимали противоречивых решений в первом раунде;
- возможность при одинаковых входных данных принимать одинаковое (не n') значение после второго раунда.

Расширение множества входных значений имеет важные последствия для разрешимости проблемы консенсуса. В [25] показано, что если требовать, чтобы результат консенсуса был входным значением какого-либо корректного процесса, то необходимо, чтобы число процессов удо-

влетворяло неравенству $N > \max(3t, Vt)$, где $V = |M|$ мощность множества входных значений.

7. СОГЛАШЕНИЯ В СЕТЯХ С ПРОИЗВОЛЬНОЙ ТОПОЛОГИЕЙ

В [2] показано, что соглашение генералов при использовании подписей достигается, если подграф, индуцированный корректными процессами, связанный. В той же работе приведен пример класса графов, в которых можно построить задачи соглашения без использования подписей. В [31] дано описание полного класса графов, в которых разрешима задача соглашения без подписей.

Теорема 5. *Рассмотрим синхронную сеть с графом, имеющим связность по ребрам, равную k. Пусть граф состоит из N вершин-процессов, t из которых могут быть некорректными. Тогда задача соглашения генералов имеет решение тогда и только тогда, когда $N > 3t$ и $k > 2t$.*

Поскольку в теореме 5 требуется достаточно большая связность сети, чего сложно достичь на практике, то в [32] было определено ослабление требования соглашения: общее решение должно быть достигнуто для всех корректных процессов, за исключением некоторого числа X. Такое соглашение называется X-соглашением. Среди работ, связанных с построением X-соглашений, можно отметить [32, 33, 34, 35].

8. СЛОЖНОСТЬ ПРОТОКОЛОВ

8.1. Верхние оценки

t-устойчивый алгоритм из [1, 2] требует $t + 1$ раундов и требует обмена экспоненциальным от t числом битов. Первым алгоритмом с полиномиальным числом отсылаемых процессом битов был алгоритм из [36], улучшенный в [37, 10, 38]

Теорема 6 ([38]). *Пусть $N > 3t$. Существует протокол для решения задачи соглашения генералов без использования подписей, в котором пересылается $O(nt + t^3 \log t)$ битов и который работает $2t + 1$ раунда.*

Какое-то время даже считалось, что нельзя получить алгоритм с полиномиальным числом передаваемых битов и работающий меньше $2t$ раундов. Однако, в работах [39, 40, 19, 20, 41, 23, 24] были построены алгоритмы, которые сначала позволили сократить число раундов до $t+t/d$

при числе пересылаемых битов $O(n^d)$, а потом сократили число пересылаемых битов до полиномиального и число раундов до $t+1$ за счет небольшого уменьшения отказоустойчивости ($N > (3+\epsilon)t$). Наилучший результат представлен в работе [21].

Теорема 7. *Существует алгоритм решения задачи консенсуса (соглашения), без использования подписей с требованием сильной зависимости и оптимальной отказоустойчивостью ($N > 3t$), полиномиальным числом пересылаемых битов и работающий в течение $t+1$ раунда*

С использованием подписей и считая количество сообщений вместо количества пересылаемых битов получаем следующий результат.

Теорема 8.

1. *Существует t-устойчивый протокол для решения задачи соглашения с использованием подписей, в котором пересылается $O(nt)$ сообщений и который работает $t+1$ раунд.*
2. *Существует t-устойчивый протокол для решения задачи соглашения с использованием подписей, в котором пересылается $O(n+t^2)$ сообщений и который работает $O(t)$ раундов.*

Первая часть теоремы доказана в [18], а вторая часть доказана в [16].

С практической точки зрения эти оценки не очень хорошие, особенно оценка в $t+1$ раундов. В общем случае эта оценка не улучшается при условии, что число ошибок было t. Однако, в [42] был рассмотрен вопрос о раннем окончании работы алгоритма при условии, что число ошибок в действительности было $f < t$. Оказывается, что ответ зависит от того, требуется синхронизация при окончании или нет.

Пусть процесс *останавливается в течение r раундов,* если он корректный, не отсылает и не принимает сообщений после r раунда и выбирает выходное значение до $r+1$ раунда. Он *останавливается в r раунде*, если он останавливается в течение r раундов, но не останавливается в течение $r-1$ раунда. Протокол соглашения останавливается, когда останавливаются все корректные процессы. Если все процессы останавливаются в одном раунде, то соглашение называется *немедленным*, в противном случае оно называется *достижимым.* Таким образом, немедленное соглашение помимо принятия решения включает в себя еше и синхронизацию при принятии решения.

Теорема 9 ([43]). *Пусть $N > 3t$. Тогда существует t-устойчивый алгоритм без использования подписей, который решает задачу соглашения и решает достижимое соглашение в течение* $\min(2t+3, 2f+5)$ *раундов, где $f \leq t$ действительное число ошибок.*

Позднее этот результат был улучшен в уже упомянутых работах [39, 19, 20, 41, 23, 24, 21] и в [22] до алгоритмов с полиномиальным числом пересылаемых битов и числом раундов $\min(t+1, f+2)$.

Еще одним направлением улучшения алгоритмов являются алгоритмы с размером сообщения в 1 бит [44, 33, 34].

Теорема 10 ([45]). *Существует алгоритм решения задачи соглашения $N = O(t \log t)$, работающий $t+1$ раунд, требующий пересылки $O(Nt)$ 1-битовых сообщений.*

8.2. Нижние оценки

В работе [26] показано, что в случае t некорректных процессов решить задачу взаимной согласованности меньше чем за $t+1$ раунд невозможно, даже если считать, что ошибки являются простыми остановками. В [26] предполагается, что $N > 2t$, и поэтому доказательство работает только для случаев без использования подписей. Позднее этот результат был усилен в разных направлениях: в [36, 46] он был обобщен на случаи с использованием подписей, в [47] результат был обобщен на проблему слабого консенсуса, в [42] результат был обобщен на случай достижимого и немедленного соглашений.

Теорема 11.

1. *Любой t-устойчивый алгоритм для слабого консенсуса работает в худшем случае $t+1$ раунд.*
2. *Любой t-устойчивый алгоритм для задачи соглашения генералов работает в худшем случае $t+1$ раунд.*
3. *Любой t-устойчивый алгоритм для задачи немедленного соглашения генералов работает в худшем случае $t+1$ раунд, даже если никаких ошибок в действительности не было.*
4. *Любой t-устойчивый алгоритм для задачи достижимого соглашения генералов работает в худшем случае $\min(t+1, f+2)$ раунд, если в действительности есть $f \leq t$ некорректных процессов.*

В работе [16] доказаны следующие соотношения для числа пересылаемых сообщений:

Теорема 12.

1. *Общее число собщений и подписей в любом t-устойчивом протоколе соглашения генералов равно $O(nt)$.*
2. *Общее число сообщений в любом t-устойчивом протоколе соглашения генералов равно $O(n + t^2)$.*

Так как каждое сообщение может содержать более одной подписи, то вторая часть теоремы дает оценку для алгоритмов с использованием подписей, а первая часть дает нижнюю оценку для алгоритмов без использования подписей.

9. РАНДОМИЗИРОВАННЫЕ И ВЕРОЯТНОСТНЫЕ АЛГОРИТМЫ

Как показано в [26], детерминированного решения для консенсуса не существует. Как решение этой проблемы были предложены рандомизированные (использующие подбрасывание монеты) [48] и вероятностные (предполагающие ограничения на произвольность поведения асинхронной системы).

В качестве примера предположений относительно поведения системы в асинхронном случае можно привести предположения из [49]. Предполагается, что алгоритм организован в виде раундов (с тем ограничением, что гарантировано получение сообщений только $N-t$ процессов). Далее считается, что вероятность $R(q,p,t)$ получения сообщения процессом q от процесса p в течение раунда t ограничена снизу ненулевым числом $R(q,p,t) > \epsilon$. Еще одно предположение состоит в независимости получения сообщений от разных процессов. Следствием этих допущений является то, что вероятность получения сообщений от одного и того же множества процессов в последовательных раундах ненеулевая, т.е. поведение некорректных процессов не является полностью произвольным.

С точки зрения разрешимости проблемы с помощью рандомизированных и вероятностных алгоритмов в [49] доказана следующая теорема.

Теорема 13. *Для разрешимости консенсуса в асинхронном случае необходимо, чтобы в случае ошибок остановок было выполнено неравенство $N > 2t$ и в случае произвольных ошибок — $N > 3t$. Для вероятностных алгоритмов выполнение этих оценок является достаточным условием разрешимости.*

В работе [50] показано, что в случае произвольных ошибок выполнение оценки теоремы 13 является достаточным условием разрешимости с помощью рандомизированных алгоритмов.

Другим преимуществом рандомизированных алгоритмов является возможность получать соглашения быстрее, чем за $t+1$ раунд. В ра-

ботах [51, 30, 52] построены рандомизированные алгоритмы, которые заканчивают работу в среднем за постоянное число раундов, не зависящее от t.

10. ЗАКЛЮЧЕНИЕ

Активное изучение проблемы соглашения ведется до сих пор, и данный обзор включает небольшую часть результатов из данной области. В частности, описание рандомизированных алгоритмов достаточно краткое и не включает некоторые более поздние, но важные результаты из этой области. Совсем не затронуты исследования, связанные с модальными логиками "знаний". В расширенной версии предполагается описание работ по этим вопросам с акцентом на исследования по применению алгоритмов соглашения на практике.

СПИСОК ЛИТЕРАТУРЫ

1. **Lamport L.** Reaching agreement in the presence of faults // JACM. — 1980. — Vol. 27, N 2. — P. 228–234.
2. **Lamport L.** The byzantine generals problem // ACM Transactions on Programming Languages and Systems. — 1982. — Vol. 4, N 3. — P. 382–401.
3. **Schlichting R.** Fail-stop processors: an approach to designing fault-tolerant computing systems // ACM Transactions on Computing Systems. — 1983. — Vol. 1, N 3. — P. 222–238.
4. **Schneider F. B.** Synchronization in distributed programs // ACM Transactions on Programming Languages and Systems. — 1982. — Vol. 4, N 2. — P. 179–195.
5. **Perry K. J.** Distributed agreement in the presence of processor and communication faults: Tech. Rep. 84-610 / K. J. Perry: Cornell University, 1984.
6. **Burns, J. E.** The byzantine firing squad problem / J. E. Burns, N. A. Lynch.
7. The distributed firing squad problem / B. A. Coan, D. Dolev, C. Dwork, L. Stockmeyer // ACM Symposium on the Theory of Computing. — 1985. — P. 335–345.
8. **Hadzilacos V.** Connectivity requirement for byzantine agreement under restricted types of failures // Distributed Computing. — 1987. — Vol. 2, N 2. — P. 95–103.
9. **Lamport L.** Using time instead of timeouts for fault-tolerant distributed systems // ACM Transactions on Programming Languages and Systems. — 1984. — Vol. 6, N 22. — P. 254–280.
10. An efficient algorithm for byzantine agreement without authentication / D. Dolev, M. J. Fischer, R. Fowler, H. R. Strong // Information and Control. — 1982. — Vol. 52, N 3. — P. 257–274.
11. **Lamport L.** The weak general problems // JACM. — 1983. — Vol. 30, N 3. — P. 668–676.
12. Reaching approximate agreement in the presence of faults / D. Dolev, N. A. Lynch, S. S. Pinter et al. // JACM. — 1986. — Vol. 33, N 3. — P. 499–516.
13. **Lynch N. A.** A new fault-tolerant algorithm for clock synchronization // ACM Symposium on Principles of Distributed Computing. — 1984. — P. 75–88.

14. **Schneider S. M. F.** Inexact agreement: accuracy, precision, and graceful degradation // ACM Symposium on Principles of Distributed Computing. — 1985. — P. 237–249.
15. **Fischer M. J.** Easy impossibility proofs for distributed consensus problems // ACM Symposium on Principles of Distributed Computing. — 1985. — P. 59–70.
16. **Dolev D.** Bounds on information exchange for byzantine agreement // JACM. — 1985. — Vol. 32, N 1. — P. 191–204.
17. **Perry K. J.** An authenticated byzantine generals algorithm with early stopping: Tech. Rep. TR-84-620 / K. J. Perry, S. Toueg: Cornell University, 1984.
18. **Dolev D.** Authenticated algorithms for byzantine agreement // SIAM J. on Computing. — 1983. — Vol. 12, N 4. — P. 656–666.
19. Shifting gears: changing algorithms on the fly to expedite byzantine agreement / A. Bar-Noy, D. Dolev, C. Dwork, H. R. Strong // Information and Computation. — 1990. — Vol. 97, N 2. — P. 203–233.
20. **Moses Y.** Coordinated traversal: $(t + 1)$-round byzantine agreement in polynomial time // Annual Symposium on Foundations of Computer Science. — 1988. — P. 246–255.
21. **Garay J. A., Moses Y.** Fully polynomial byzantine agreement for n $>$ 3t processors in t+1 rounds. — SIAM J. of Computing. — 1998. — Vol. 27(1).
22. **Berman P., Garay J. A., Perry K. J.** Optimal early stopping in distributed consensus // Internat. Workshop on Distributed Algorithms. — 1992. — P. 221–237.
23. **Berman P., Garay J. A.** Cloture votes: $n/4$-resilient distributed consensus in $t+1$ rounds // Mathematical Systems Theory. — 1993. — Vol. 26. — P. 3–19.
24. **Berman P., Garay J. A.** Efficient distributed consensus with $n = (3 + \epsilon)t$ processors // Lect. Notes Comput. Sci. — 1991. — Vol. 579. — P. 129–142.
25. **Neiger G.** Distributed consensus revisited: Tech. Rep. GIT-CS-93/45 / G. Neiger: Georgia Institute of Technology, 1993.
26. **Fischer M. J., Lynch N. A., Paterson M. S.** Impossibility of distributed consensus with one faulty process // JACM. — 1985. — Vol. 32, N 2. — Pp. 374–382.
27. **Dolev D., Dwork C., Stockmeyer L.** On the minimal synchronism needed for distributed consensus // JACM. — 1987. — Vol. 34, N 1. — P. 77–97.
28. **Attiya G., Dolev D., Gil J.** Asynchronous byzantine consensus // ACM Symposium on Principles of Distributed Computing. — 1984. — P. 119–133.
29. **Dwork C., Lynch N. A., Stockmeyer L.** Consensus in the presence of partial synchrony // JACM. — 1988. — Vol. 35, N 2. — P. 288–323.
30. **Perry K. J.** Randomized byzantine agreement: Tech. Rep. TR-84-595 / K. J. Perry: Cornell University, 1984.
31. **Dolev D.** The byzantine generals strike again.: Tech. Rep. STAN-CS-81-846 / D. Dolev: 1981.
32. Fault tolerance in the network of bounded degree / C. Dwork, D. Peleg, N. Pippenger, E. Upfal // Annual ACM Symposium on Theory of Computing. — 1986. — P. 370–379.
33. **Berman P., Garay J. A.** Asymptotocally optimal distributed consensus // Lect. Notes Comput. Sci. — 1989. — Vol. 372. — P. 80–94.
34. **Berman P.** Fast consensus in networks of bounded degree // Dsitributed Computing. — 1993. — Vol. 7. — P. 67–73.
35. **Upfal E.** Tolerating linear number of faults in networks of bounded degree // Annual ACM Symposium on Principles of Distributed Computing. — 1992. — P. 83–89.

36. **Dolev D., Strong H. R.** Polynomial algorithm for multiple processor agreement// ACM symposium on Theory of computing. — 1982. — P. 401–407.
37. **Fischer M. J., Fowler R., Lynch N. A.** A simple and efficient byzantine generals algorithm // IEEE Symposium on Reliability in Distributed Software and Database Systems, IEEE CS 2, Wiederhold(ed), Pittsburgh PA, 1982.
38. **Srikanth, T. K.** Simulating authenticated broadcasts to derive simple fault-tolerant algorithms: Tech. Rep. TR-84-623 / T. K. Srikanth, S. Toueg: Cornell University, 1984.
39. **Coan B. A.** A communication-efficient canonical form for fault-tolerant distributed protocols // Annual ACM Symposium on Principles of Distributed Computing. — 1986. — P. 63–72.
40. **Coan B. A., Welch J. L.** Modular construction of nearly optimal byzantine agreement protocols // Annual ACM Symposium on Principles of Distributed Computing. — 1989. — P. 295–305.
41. **Berman P., Garay J. A., Perry K. J.** Toward optimal distributed consensus // Annual Symposium on Foundations of Computer Science. — 1989. — P. 410–415.
42. **Dolev D., Reischuk R., Strong H. R.** Early stopping in byzantine agreement // JACM. — 1990. — Vol. 37, N 4. — P. 720–741.
43. **Perry, K. J.** Fast distributed agreement: Tech. Rep. TR-84-621 / K. J. Perry, T. K. Srikanth, S. Toueg: Cornell University, 1984.
44. **Bar-Noy A., Dolev D.** Families of consensus algorithms // AWOC. — 1988. — P. 380–390.
45. **Coan B. A., Welch J. L.** Modular construction of an efficient 1-bit byzantine agreement protocol // **Mathematical Systems Theory**. — 1993. — Vol. 26. — P. 131–154.
46. **DeMillo R. D., Lynch N. A., Merritt M.** Cryptographic protocols // ACM Symposium on Theory of Computing. — 1982. — P. 383–400.
47. **Fischer, M. J.** Byzantine generals and transaction commit protocol: Tech. Rep. Opus 62 / M. J. Fischer, L. Lamport: SRI inc, 1982.
48. **Ben-Or M.** Another advantage of free choice: completely asynchronous agreement protocols // ACM symposium on Principles of distributed computing. — 1983. — P. 27–30.
49. **Bracha G., Toueg S.** Asynchronous consensus and broadcast protocols // JACM. — 1985. — Vol. 32, N 4. — P. 824–840.
50. **Bracha G.** An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol // ACM symposium on Principles of distributed computing. — 1984. — P. 154–162.
51. **Rabin M.** Randomized byzantine generals // Symposium on Foundations of Computer Science. — 1983. — P. 403–409.
52. **Toueg S.** Randomized byzantine agreement // ACM Symposium on Principles of Distributed Computing. — 1984. — P. 163–178.
53. **Fischer M. J.** The consensus problem in unreliable distributed systems (a brief survey): Tech. Rep. YALEU/DCS/TR-273 / M. J. Fischer: Yale University, 1983.