Теоретический минимум по курсу

«Распределённые алгоритмы и системы» (весна 2016)

*лектор – Захаров Владимир Анатольевич*

Составитель: avasite

**Оглавление**

[1. Распределенные системы и их характерные особенности. Архитектура распределенных систем. Алгоритмические проблемы организации вычислений распределенных систем. Особенности распределенных алгоритмов. 2](#_Toc453788827)

[2. Математическая модель. Системы переходов. Системы с синхронным и асинхронным обменом сообщениями. Свойство справедливости выполнений системы. Зависимые и независимые события. Причинно-следственный порядок событий. Эквивалентность выполнений. Вычисления. Логические часы. 3](#_Toc453788828)

[3. Коммуникационные протоколы. Ошибки, возникающие при передаче сообщений. Симметричный протокол раздвижного окна. Устройство протокола раздвижного окна. Принципы обоснования корректности протоколов Обоснование корректности протокола раздвижного окна. Модификации протокола. 5](#_Toc453788829)

[4. Коммуникационный протокол с таймерами. Допущения о времени. Таймеры. Устройство протокола с таймерами. Обоснование корректности протокола с таймерами. Модификации протокола. 6](#_Toc453788830)

[5. Задача маршрутизации. Маршрутизация на основе узлов-адресатов. Задача построения кратчайших путей для всех пар вершин. Алгоритм Флойда-Уоршалла. Алгоритм Туэга. Алгоритм Мерлина-Сигалла. Алгоритм Чанди-Мисры. 7](#_Toc453788831)

[6. Алгоритм маршрутизации Netchange. Описание алгоритма. Инварианты алгоритма Netchange. Корректность алгоритма Netchange. Сходимость алгоритма Netchange. Особенности реализации алгоритма. Другие виды маршрутизации: интервальная, префиксная, иерархическая. 9](#_Toc453788832)

[7. Волновые алгоритмы: определения, основные свойства, область применения. Древесный алгоритм. Алгоритм эха. 10](#_Toc453788833)

[8. Волновые алгоритмы. Фазовый алгоритм. Алгоритм Финна. Распределенные алгоритмы обхода: основные свойства. Алгоритм обхода Тарри. Распределенный обход в глубину. Алгоритмы Авербаха и Сидона. 12](#_Toc453788834)

[9. Задача избрания лидера. Основные определения и допущения. Волновые алгоритмы избрания лидера. Выборы лидера на дереве. Выборы в кольцах. Алгоритм Ле-Ланна. Алгоритм Ченя - Робертса. Алгоритм Петерсона/Долева-Клейва-Роде. Эффект угасания. 14](#_Toc453788835)

[10. Задача избрания лидера. Нижние оценки сложности. Оптимальные выборы. Алгоритм Галладжера-Хамблета-Спиры (GHS). Глобальное описание алгоритма GHS. Подробное описание алгоритма GHS. Алгоритм Корача-Каттена-Морана. 15](#_Toc453788836)

[11. Задача обнаружения завершения вычислений. Алгоритм Дейкстры-Шолтена. Алгоритм Шави-Франчеза. Алгоритм Сафры. Алгоритм возвращения кредитов. 17](#_Toc453788837)

[12. Задача сохранения моментального состояния. Алгоритм Чанди-Лампорта Алгоритм Лаи-Янга. Применение алгоритмов сохранения моментального состояния. 20](#_Toc453788838)

1. Распределенные системы и их характерные особенности. Архитектура распределенных систем. Алгоритмические проблемы организации вычислений распределенных систем. Особенности распределенных алгоритмов.

Распределенной системой называют всякую вычислительную систему, в которой несколько автономных компьютеров или программ вступают во взаимодействие друг с другом.

Взаимодействующие компьютеры, процессы или программы называют узлами распределенной системы.

Механизм организации связи между узлами называют коммуникационной подсистемой.

Распределенные алгоритмы - это алгоритмы, определяющие функционирование распределенных систем.

*Примеры распределенных систем*:

* глобальные коммуникационные вычислительные сети
* локальные вычислительные сети
* многопроцессорные компьютеры
* системы взаимодействующих процессов.

*Зачем нужны распределенные системы?*

1. Обмен информацией
2. Совместное использование ресурсов
3. Повышение надежности за счет дублирования
4. Повышение производительности за счет параллельного выполнения
5. Упрощение проектирования за счет специализации

Вычислительная сеть - это совокупность компьютеров, соединенных между собой при помощи механизмов коммуникации, позволяющих компьютерам сети обмениваться информацией. Этот обмен осуществляется путем отправления и приема сообщений.

*Основные отличия между глобальными и локальными сетями:*

1. Параметры надежности.
2. Время связи.
3. Однородность аппаратуры и программного обеспечения.
4. Взаимное доверие.

*Алгоритмические проблемы для глобальных сетей:*

1. Обеспечение надежного обмена данными по двухточечной линии связи.
2. Выбор каналов связи - задача маршрутизации пакетов.
3. Устранение перегрузки в сети.
4. Предотвращение блокировки (тупиков) - задача неблокируемой коммутации.
5. Обеспечение безопасности.

Связь между узлами локальной сети обеспечивает единый механизм - шина - к которой подключены все узлы.

*Алгоритмические проблемы для локальных сетей:*

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.
3. Обнаружение завершения.
4. Обеспечение корректного использования ресурсов.
5. Обнаружение и устранение тупиков.
6. Распределенное обслуживание файлов.

Многопроцессорный компьютер - это вычислительная машина, которая содержит несколько процессоров, размещенных в корпусе одного устройства.

Если многопроцессорный компьютер предназначен для ускорения вычислений, то его называют параллельным компьютером.

А если он предназначен для повышения надежности вычислений, то его называют вычислительной системой с дублированием.

*Алгоритмические проблемы для многопроцессорных компьютеров:*

1. Реализация системы передачи сообщений - маршрутизация, устранение перегрузки и блокировок.
2. Реализация виртуальной разделяемой памяти.
3. Уравновешивание нагрузки.
4. Устойчивость к необнаруживаемым сбоям.

*Алгоритмические проблемы для систем взаимодействующих процессов:*

1. Атомарность операций записи и считывания.
2. Синхронизация процессов.
3. Сборка мусора.
4. Обмен сообщениями.

Коммуникационная подсистема строится из отдельных модулей, каждый из которых служит для выполнения очень специальной функции при поддержке других модулей. Модули образуют строгую иерархию: каждый модуль может обращаться только к тем модулям, которые непосредственно предшествуют ему в этом иерархическом порядке. Эти модули называются уровнями.

Архитектурой сети называется совокупность всех модулей (уровней), вместе с определениями соответствующих интерфейсов и протоколов.

Чтобы выразить параллелизм достаточно определить несколько процессов, каждый из которых является последовательной процедурой.

В языке могут быть возможности для статического и для динамического определения процессов.

Коммуникация осуществляется посредством обмена сообщениями (обеспечивает как коммуникацию, так и синхронизацию) и/или при помощи совместно используемой памяти (обеспечивает только коммуникацию)

Недетерминизм выражается при помощи охраняемых команд.

Охраняемая команда - это список операторов, каждому из которых предшествует булево выражение (предохранитель). Процесс может продолжить свое выполнение, выбрав любой оператор, предохранитель которого принимает значение true.

Особенности распределенных алгоритмов:

1. Отсутствие сведений о глобальном состоянии.
2. Отсутствие глобальной шкалы времени.
3. Недетерминизм.
4. Математическая модель. Системы переходов. Системы с синхронным и асинхронным обменом сообщениями. Свойство справедливости выполнений системы. Зависимые и независимые события. Причинно-следственный порядок событий. Эквивалентность выполнений. Вычисления. Логические часы.

От модели требуется быть: точной, общей, компактной.

Распределенное вычисление обычно рассматривается в виде упорядоченной совокупности дискретных событий, каждое из которых представляет собой небольшое изменение конфигурации (глобального состояния всей системы). Правила изменения конфигураций задаются в виде системы переходов.

Система переходов – это тройка <конфигурация, отношение переходов, начальные конфигурации>

Выполнение системы переходов – максимальная последовательность, удовлетворяющая системе переходов.

Заключительная конфигурация (если из неё нету перехода).

Максимальное выполнение – выполнение, которое либо бесконечное, либо заканчивается заключительной конфигурацией.

Распределенная система состоит из совокупности процессов и коммуникационной подсистемы.

События бывают 3-х типов: внутренние, отправления и приёма.

Соглашение:

Термины переход и конфигурация будут использоваться применительно ко всей распределенной системе в целом.

Термины событие и состояние, обозначающие равносильные понятия, будут использоваться, если речь идет о процессах системы.

Локальный алгоритм процесса – это пятёрка <состояния, начальные состояния, внутренние действия, отправление и приём сообщения>

Распределённый алгоритм семейства процессов – совокупность локальных алгоритмов, каждый из которых в точности соответствует одному процессу.

Система переходов асинхронно взаимосвязанных процессов –

Система переходов синхронно связанных процессов –

Соглашение:

Для каждого сообщения существует единственный процесс, который может его получить, его мы назовём адресатом.

Каждое выполнение, заканчивающееся заключительной конфигурацией, является справедливым (как слабо, так и сильно)

Бесконечное выполнение слабо справедливо, если когда-нибудь то, что может выполниться – выполнится.

Бесконечное выполнение сильно справедливо, если событие после какого-то шага может случиться, то когда-нибудь будет так, что оно случится первым, как только станет возможным.

Пространственно-временная диаграмма.

Теорема 1.

Пусть есть 2 события в разных процессах, которые допустимы в данной конфигурации, тогда каждое событие допустимо в конфигурации после выполнения другого события, и после выполнения событий в любой очерёдности конфигурации системы будут равны.

Отношение причинно-следственного порядка – (задано выполнение E) – это наименьшее отношение, удовлетворяющее следующим условиям:

1. Если e и f – 2 разных события одного процесса, и е происходит прежде события f, то e <- f.
2. Если s – это событие отправления сообщения, а r – это события принятия этого сообщения, то s<-r.
3. «<-» - обладает свойством транзитивности

«<=» - отношение частичного порядка, которое определяется формулой (a<-b or a=b)

События, для которых не выполняются a<=b и b<=a – называются параллельными.

Теорема 2.

Пусть f – перестановка событий выполнения E, сохраняющая причинно-следственный порядок событий. Тогда f определяет единственное выполнение F, начинающееся с той же конфигурации, что и E. При этом в F происходит столько же событий, сколько и в E, и в том случае, если E конечное выполнение, то последняя конфигурация выполнения F будет точно такой же, как последняя конфигурация выполнения E.

Эквивалентные выполнения (~) – выполнения, в которых одно и тоже множество событий, в котором задан один и тот же причинно-следственный порядок событий.

Вычислением распределенного алгоритма называется класс эквивалентности выполнений алгоритма по отношению эквивалентности ~.

Часы – всякая функция, которая отображает множество событий в некоторое упорядоченное множество так, что при этом сохраняется отношение причинно-следственного порядка.

Логические часы Лэмпорта – по сути они тикают на +1 на каждом шаге в каждом процессе, но когда приходит сообщение от другого процесса, то там также указан момент времени, когда сообщение было послано, и если этот тик больше локального тика, то локальный тик увеличивается до него.

Определение формулируется через максимальную длину частично упорядоченной последовательности событий всех различных выполнений.

Асинхронные системы – менее безопасны, но более живучи.

*Свойства каналов связи*: надёжность, свойство очерёдности, пропускная способность каналов

*Осведомлённость процессов*: топологическая информация, отличительные признаки процессов, отличительные признаки соседей.

*Сложность распределённых алгоритмов*: по числу обменов сообщениями, битовая сложность, сложность по времени.

1. Коммуникационные протоколы. Ошибки, возникающие при передаче сообщений. Симметричный протокол раздвижного окна. Устройство протокола раздвижного окна. Принципы обоснования корректности протоколов Обоснование корректности протокола раздвижного окна. Модификации протокола.

*(лекция специфична, подробнее см. слайды)*

*Определение 3.1*

Пусть задана система переходов S = (C, ->, I) и рассматриваются некоторые свойства конфигураций P и Q.

Запись P(y), где y — конфигурация, будет обозначать логическое выражение (формулу), принимающее истинное значение в том случае, если утверждение P справедливо для конфигурации y, и ложное значение в противном случае. Запись {P} -> {Q} будет использоваться для обозначения того, что для каждого перехода y -> z в системе S истинность P(y) влечет истинность Q(z).

Утверждение P называется инвариантом системы S, если

1. P (y) истинно для каждой начальной кон фигурации y ∈ I и
2. {P} ^ {P}

Теорема 3.1

Если утверждение P является инвариантом системы переходов S, то доя любого выполнения E системы S утверждение P будет истинно в каждой конфигурации этого выполнения.

Теорема 3.2.

Если Q является инвариантом системы переходов S, и для каждой конфигурации y ∈ C выполняется Q(y) => P(y), то для любого выполнения системы S утверждение P будет истинно в каждой конфигурации выполнения.

*Определение 3.2.*

Пусть задан признак правильного завершения выполнений Р. Система S правильно завершает выполнения (или, иными словами, свободна от блокировки), если предикат (term => Р) всегда является истинным для системы S.

*Определение 3.3.*

Частично упорядоченное множество (W, <) называется фундированным, если из его элементов нельзя составить бесконечно убывающую последовательность.

*Определение 3.4.*

Пусть заданы система переходов S и утверждение P. Функция f, отображающая множество C в фундированное множество W, называется функцией нормировки (по отношению к P), если для каждого перехода y -> z, либо выполняется f(y) > f(z), либо P(z) принимает истинное значение.

Теорема 3.3.

Пусть заданы система переходов S и утверждение P. Если S правильно завершает выполнения, и существует функция нормировки f (по отношению к P), то для каждого выполнения S утверждение P становится истинным в некоторой конфигурации.

Теорема 3.4.

Утверждение P является инвариантом алгоритма раздвижного окна.

Теорема 3.5.

Симметричный протокол раздвижного окна удовлетворяет требованию безопасной доставки сообщений, т.е. в каждой достижимой конфигурации протокола выполняются соотношения outp[0..sp-1]=inq[0..sp-1], outq[0..sq-1]=inp[0..sq-1]

Лемма 3.1.

В любой достижимой конфигурации выполняются неравенства sp - lq <= aр <= sq <= aq + lp <= sp + lp

Лемма 3.2.

В любой достижимой конфигурации допустимо хотя бы одно из двух действий: отправление пакета (расk, inp[sq], sq) процессом р или отправление пакета (расk, inq[sp], sp) процессом q.

Теорема З.6.

Симметричный протокол раздвижного окна удовлетворяет требованию неизбежной доставки сообщений, т.е. для каждого целого числа к >= 0 , в ходе любого выполнения протокола будет достигнуты конфигурация, в которой sp >= k и sq >= k.

*Пусть L = lр + lq*

Лемма 3.3.

Отправление пакета (расk, w, i) процессом p допустимо только тогда, когда i < ap + L.

Лемма 3.4.

Если outp[i] ≠ udef, то выполняется неравенство i < sp + L.

Такой вариант алгоритма раздвижного окна называется протоколом чередующихся (альтернирующих) битов; он предназначен для односторонней передачи данных (L=1).

1. Коммуникационный протокол с таймерами. Допущения о времени. Таймеры. Устройство протокола с таймерами. Обоснование корректности протокола с таймерами. Модификации протокола.

Упрощённый протокол сквозной передачи сообщений:

1. ***Однонаправленность***. Передача данных идет только в одном направлении от процесса p к процессу q. Мы будем называть процесс p отправителем, а процесс q — получателем.
2. ***В окне приема — одно слово***. Поступивший пакет получатель доставляет по назначению только в том случае, когда его порядковый номер совпадает с ожидаемым номером.
3. ***Упрощенные допущения о времени***. Используется минимальное количество таймеров. Например, предполагается, что подтверждение может быть отправлено в любое время до тех пор, пока получатель поддерживает соединение открытым.
4. ***Пакеты состоят из одного слова***. В каждый пакет данных отправитель может поместить одно-единственное слово.

Таймеры — устройства, измеряющие физическое время

1. Глобальное время. Все процессы системы функционируют в рамках единой глобальной шкалы времени. Каждое событие происходит мгновенно, и процессы не могут регистрировать те моменты времени, в которые происходят события.
2. *Ограниченное время жизни пакета*
3. Таймер — это вещественная переменная Xt, значение которой со временем постоянно убывает (или присваивается этой переменной явным образом).

Утверждение:

Никакой протокол не может предоставить гарантии того, что слово будет доставлено по назначению за ограниченный срок времени. (т.е. невозможно одновременное отсутствие потерь и сохранение порядка)

<дальше шёл алгоритм, доказательство его корректности, безопасности, инварианта, …> см. слайды

Теорема 4.3. (об отсутствии потерь)

Каждое слово массива inp либо достигает процесса q, либо отмечается процессом p как утраченное в течение U + 2µ + R + λ единиц времени после поступления этого слова процессу р.

Теорема 4.4. (о сохранении порядка вручения)

Слова, доставленные процессом q потребителю, расположены в строго возрастающем порядке в массиве inp.

1. Задача маршрутизации. Маршрутизация на основе узлов-адресатов. Задача построения кратчайших путей для всех пар вершин. Алгоритм Флойда-Уоршалла. Алгоритм Туэга. Алгоритм Мерлина-Сигалла. Алгоритм Чанди-Мисры.

Из каждого узла пакеты информации могут непосредственно передаваться только некоторому подмножеству других узлов, которые называются соседями этого узла.

Маршрутизация — это процедура принятия решения о том, какому соседу (иногда не единственному) следует передать пакет, чтобы он в конце концов был доставлен по назначению. Цель, которая ставится при проектировании алгоритма маршрутизации, состоит в том, чтобы снабдить каждый узел процедурой, которая сможет выполнять эту функцию и гарантировать доставку каждого пакета по назначению.

*Задачу маршрутизации можно разбить на две алгоритмические составляющие*:

1. Вычисление таблиц. Таблицы маршрутизации должны быть вычислены при инициализации сети и должны обновляться при изменении топологии сети.
2. Продвижение пакета. Когда пакет пересылается по сети, то его продвижение осуществляется на основе таблиц маршрутизации.

*Критерии оценки качества методов маршрутизации учитывают следующие показатели*.

1. Корректность. Алгоритм должен доставлять каждый пакет, поступивший в сеть, в точности по назначению.
2. Эффективность. Алгоритм должен отправлять пакеты по «наилучшим» путям.
3. Сложность. Алгоритм вычисления таблиц должен использовать как можно меньше сообщений, расходовать как можно более экономно время и память.
4. Устойчивость. В случае изменения топологии (добавлении или удалении канала или вершины) алгоритм вносит изменения в таблицы маршрутизации, для того чтобы задачу маршрутизации можно было решить в модифицированной сети.
5. Адаптивность. Для того чтобы выровнять загруженность каналов и вершин, алгоритм приспосабливает к этому таблицы, избегая прокладывать пути через те каналы и вершины, которые чрезмерно загружены, и отдавая предпочтение тем каналам и вершинам, на которые в это время возложена наименьшая нагрузка.
6. Справедливость. Алгоритм должен обслуживать всех пользователей в равной мере.

*Несколько вариантов понятия «наилучший путь»*:

1. Минимальное количество звеньев.
2. Минимальная стоимость.
3. Минимальная задержка.

*Основные допущения о свойствах маршрутов*:

1. Стоимость отправления пакета по пути P не зависит от того, используются ли ребра пути P для продвижения других сообщений. Поэтому стоимость C(P) пути P — это функция, зависящая только от самого пути.
2. При последовательном соединении двух путей образуется новый путь, стоимость которого равна сумме стоимостей двух исходных путей, т.е. C({u0, u1, ..., uk}) = C({u0, ..., ui}) + C({ui, ..., uk}).

Стоимость пустого пути {u0} равна 0.

1. В графе нет циклов отрицательной стоимости.

Путь из вершины *u* в вершину *v* называется оптимальным если нет пути меньшей стоимости из *u* в *v*.

Лемма 5.1. (о простых путях).

Пусть u и v — вершины графа G. Если в графе G есть путь P из u в v, то имеется также и простой путь из u в v, который является при этом оптимальным.

Теорема 5.1. (о дереве оптимальных путей).

Для каждой вершины u ∈ V связного графа G существует такое дерево Tu = (V, Eu), Eu ⊆ E, в котором для любой вершины v ∈ V, единственный путь из v в u в дереве Tu является оптимальным путем из v в u в графе G.

Остовное дерево, корнем которого служит вершина u, называется входным деревом для u, а дерево, удовлетворяющее теореме о дереве оптимальных путей, называется оптимальным входным деревом.

Существование оптимального входного дерева означает, что алгоритм оптимальной маршрутизации может работать с использованием следующей локальной процедуры table\_lookupv.

Лемма 5.2. (об ациклических таблицах)

Механизм продвижения доставляет каждый пакет по назначению в том и только том случае, когда таблицы маршрутизации являются ациклическими.

S-путь – путь через вершины подмножества S всех вершин V.

Алгоритм Флойда-Уоршалла

<описание алгоритма Флойда-Уоршалла> см. слайды

Допущения:

1. Каждый цикл имеет положительный вес
2. Каждый процесс располагает информацией об отличительных признаках всех узлов системы
3. Каждый процесс знает кто его соседи (знает их отличительные признаки) и вес соединяющих их каналов

Теорема 5.3. (об алгоритме Флойда-Уоршалла)

Алгоритм Флойда-Уоршалла вычисляет расстояние между всеми парами вершин за θ(N3) шагов.

Алгоритм Туэга

В основу алгоритма Туэга (Toueg) положен централизованный алгоритм Флойда-Уоршалла.

<описание алгоритма Туэга (Toueg)> см. слайды

Допущения:

1. Каждый цикл имеет положительный вес
2. Каждый процесс располагает информацией об отличительных признаках всех узлов системы
3. Каждый процесс знает кто его соседи (знает их отличительные признаки) и вес соединяющих их каналов

Теорема 5.5. (корректности и сложности алгоритма Туэга)

Для каждой пары вершин u и v алгоритм Туэга вычисляет расстояние между u и v. Если это расстояние конечно, то он также определяет первый канал в кратчайшем пути.

По ходу работы алгоритма по каждому каналу проходит O(N) сообщений, O(N2W) битов информации. Таким образом, суммарно по ходу работы алгоритма передается O(N \*|Е|) сообщений и O(N3 \* W) битов информации. Кроме того в каждом узле используется память, объем которой составляет O(N \* W) битов.

Алгоритм Туэга:

Достоинства: Прост, имеет небольшую сложность, и строит оптимальные маршруты.

Недостатки:

1. Плохая устойчивость («робастость»): при изменении топологии сети все вычисления нужно проводить заново.
2. Согласованный выбор очередной опорной вершины (w) всеми узлами сети предполагает, что множество участвующих в алгоритме процессов заранее известно.
3. В алгоритме Туэга часто применяется неравенство треугольника d(u, v) < d(u, w) + d(w, v). Для вычисления правой части этого неравенства (в узле u) требуется «глобальная» информация о d(w, v), которой не обладает ни процесс u, ни его соседи. Зависимость от удаленных данных вынуждает нас организовать доставку этой информации удаленным вершинам.

Алгоритм Мерлина-Сигалла

<алгоритм Мерлина-Сигалла> см. слайды

Особенности:

1. Алгоритм может быть приспособлен к изменяющейся топологии.
2. Вычисления для разных вершин адресатов проводятся независимо.
3. Походу вычислений таблицы всегда находятся в ациклическом состоянии.

Теорема 5.6. (корректности и сложности алгоритма Мерлина-Сигалла)

Алгоритм Мерлина—Сигалла вычисляет таблицы маршрутизации по кратчайшим путям, совершая при этом обмен O(N2) сообщениями и передавая O(N2 \* W) битов информации по каждому каналу связи, имея, таким образом, коммуникационную сложность O(N2 \* |Е|) и битовую сложность O(N2 \* |E|W).

Алгоритм Чанди-Мисры вычисляет все кратчайшие пути к заданной вершине-адресату на основе принципа диффузных вычислений. Суть его такова: распределенное вычисление начинается в одной-единственной вершине, и другие узлы сети поочередно присоединяются к вычислению только после получения сообщения.

Подробнее см. слайды.

Сложность Чанди-Мисры совпадает со сложностью Мерлина-Сигалла.

*(я не вижу разницы между этими алгоритмами)*

1. Алгоритм маршрутизации Netchange. Описание алгоритма. Инварианты алгоритма Netchange. Корректность алгоритма Netchange. Сходимость алгоритма Netchange. Особенности реализации алгоритма. Другие виды маршрутизации: интервальная, префиксная, иерархическая.

В нем поддерживается дополнительная информация, чтобы в случае возникновения неисправности в канале или восстановления работоспособности канала уточнять таблицы путем их частичного перевычисления.

Допущения для алгоритма Netchange:

1. Каждый узел осведомлен о размере всей сети (N).
2. В каналах поддерживается очередность сообщений.
3. Узлы получают уведомления о возникновении неисправностей и восстановлении работоспособности примыкающих к ним каналов. (алгоритм может быть легко расширен для случая, если у каналов есть вес, и эти веса могут меняться (нужны лишь уведомления))
4. Стоимость пути равна количеству каналов в этом пути

Требования к алгоритму:

1. Если топология сети после конечного числа изменений остается далее неизменной, то алгоритм завершает работу после конечного числа шагов.
2. Теорема 1

Когда алгоритм завершает работу, таблицы Nbu[v] удовлетворяют следующим условиям:

* + если v = u , то Nbu[v] = local
  + если существует путь из вершины u в вершину v ≠ u, то Nbu[v] = w, где w — это первая вершина-сосед узла u, который следует за u в кратчайшем пути из u в v
  + если пути из вершины u в вершину v нет, то Nbu[v] = udef

<описание алгоритма Netchange> (начиная со слайда 14)

В постой модели алгоритма мы считаем, что fail и repair обрабатываются одновременно на обоих узлах от линка (но это не обязательно – слайд 74), и ещё мы просто ищем самый короткий путь, без учёта веса (это не влияет на общую организацию алгоритма) (слайд 75)

<доказательство инвариантов>

Доказательство завершаемости алгоритма Netchange происходит за счёт введения частично-упорядоченного множества (фундированное множество).

Теорема 2.

Если, начиная с какого-то момента топология сети остается неизменной, то спустя конечное число шагов алгоритм достигает стабильной конфигурации.

Другие виды маршрутизации:

1. Схема древесной разметки - Этот метод помечает вершины целыми числами от 0 до N-1 так, чтобы для каждого канала связи множество вершин-адресатов, сообщение с которыми осуществляется через этот канал, представляло собой интервал.

Достоинства

* + для задания одного циклического интервала можно обойтись 2 log N битами, указывая только границы интервала.

Недостатки:

* + Каналы, не принадлежащие дереву T, не используются, а это означает, что ресурсы сети расходуются неэкономно.
  + Весь трафик сосредоточен в дереве, а это приводит к перегрузкам в сети.
  + Всякая неисправность в одном канале дерева T приводит к разрыву сети.

1. Префиксная маршрутизация - в префиксной маршрутизации пометками вершин и каналов служат строки. Чтобы выбрать канал, по которому нужно продвигать пакет, алгоритм рассматривает все пометки каналов, являющиеся префиксом адреса вершины назначения. Выбирается наиболее длинная из таких пометок, и продвижение сообщения проводится по соответствующему каналу.
2. Иерархическая маршрутизация - иерархический метод маршрутизации предполагает разбиение сети на кластеры. (очень схоже с современным интернетом, в котором есть понятие «шлюза»)
3. Волновые алгоритмы: определения, основные свойства, область применения. Древесный алгоритм. Алгоритм эха.

Зачем нужны волновые алгоритмы:

При проектировании распределенных алгоритмов для разных приложений некоторые общие проблемы, которые нужно уметь решать, как промежуточные задачи. К ним относятся:

* широковещательное распространение информация (например, сообщения о начале или завершении вычисления), (PIF)
* установление глобальной синхронизации между процессами, (SYN)
* запуск выполнения некоторого действия в каждом процессе,
* вычисление функции при условии, что каждый из процессов содержит только часть входных данных, (INF)
* …

Эти задачи можно решать путем обмена сообщениями согласно некоторой предписанной схеме, которая зависит от топологии сети и позволяет задействовать все процессы.

Допущения для волновых алгоритмов

Соглашение о сетевой модели:

* имеет фиксированную топологию (никаких изменений топологии не происходит);
* является неориентированной (по каждому каналу сообщения могут передаваться в обоих направлениях);
* является связной (между любыми двумя процессами пролегает путь),
* является асинхронной,
* не имеет доступа к показаниям вещественных часов глобального времени.

Предполагается, что существуют внутренние события специального типа, которые называются событиями решения, такие события просто представлены оператором decide – «событие решения».

Волновой алгоритм, это алгоритм, который удовлетворяет следующим требованиям:

1. он конечен,
2. каждое вычисление содержит хотя бы одно событие *decide,*
3. В любом вычислении всякому событию решения предшествует в причинно-следственном отношении хотя бы одно событие в каждом из процессов.

При выполнении проводится различие между процессами инициаторами и неинициаторами.

Параметры волновых алгоритмов:

1. Централизация
2. Топология
3. Первоначальные сведения
4. Число решений
5. Сложность

*(формулировки утверждений, перефразированные на человеческий язык)*

Утверждение 1

Для каждого события решения есть инициатор и его событие, которые в любом выполнении происходят до этого события.

Утверждение 2

Если отслеживать граф передачи первых событий, то это остовное дерево

Утверждение 3

Для любого события решения в каждом другом процессе было событие до него (в любом выполнении), являющееся отправкой сообщения.

Утверждение 4

Пусть C — такая волна с одним инициатором p в распределенной системе, состоящей из N процессов, что в p происходит событие решения decide. Тогда в вычислении C происходит обмен по крайней мере N сообщениями.

Утверждение 5

Пусть A — волновой алгоритм для произвольной сети, в котором не используются первоначальные сведения об отличительных признаках соседей. Тогда A в каждом вычислении проводит не менее |Е| обменов сообщениями.

Волновые алгоритмы необходимы для широковещательного распространения информации с подтверждением о завершении выполнения (широковещательной связи Propagation of Information with Feedback, PIF).

Пусть имеется подмножество процессов, обладающих некоторым сообщением M. Нужно, чтобы все процессы приняли M. Некоторые выделенные процессы должны быть уведомлены о завершении широковещательной связи; это означает, что в них должно быть выработано специальное событие notify с тем условием, что событие notify произойдет только в том случае, когда все процессы уже получат сообщение M.

Утверждение 6.

Каждый PIF-алгоритм — это волновой алгоритм.

Утверждение 7.

Каждый волновой алгоритм может быть использован в качестве PIF-алгоритма.

Волновые алгоритмы необходимы для глобальной синхронизации (Synchronization, SYN) между процессами. Требование синхронизации таковы: в каждом процессе q должно быть выполнено событие aq, и в некоторых процессах событие bp должно быть выполнено так, чтобы выполнение aq произошло по времени ранее, нежели выполнение любого из событий bp.

В SYN-алгоритме события bp рассматриваются как события decide.

Утверждение 8.

Каждый волновой алгоритм может быть использован в качестве SYN-алгоритма.

Утверждение 9.

Каждый SYN-алгоритм может быть использован в качестве волнового алгоритма.

Волновые алгоритмы необходимы для вычисления ***некоторых функций***, зависящих от входных данных каждого процесса. Примером такими функциями могут служить функции точной нижней грани по всем входам.

**Примеры волновых алгоритмов** (подробное и формальное описание на слайдах):

1. Кольцевой алгоритм
2. Древесный алгоритм
3. Алгоритм эха

Утверждения: эти примеры алгоритмов являются волновыми алгоритмами.

1. Волновые алгоритмы. Фазовый алгоритм. Алгоритм Финна. Распределенные алгоритмы обхода: основные свойства. Алгоритм обхода Тарри. Распределенный обход в глубину. Алгоритмы Авербаха и Сидона.

Фазовый алгоритм является *децентрализованным* алгоритмом, пригодным для сетей с произвольной топологией.

Ограничения:

1. Его можно использовать в качестве волнового алгоритма для **ориентированных сетей**.
2. В этом алгоритме требуется, чтобы все процессы располагали сведениями о диаметре сети D. Алгоритм будет оставаться корректным (хотя и менее эффективным), если все процессы будут использовать вместо D константу D1, превышающую диаметр сети.
3. Процессы могут не уметь друг друга идентифицировать

Идея фазового алгоритма:

В фазовом алгоритме всякий процесс отправляет в точности D сообщений каждому соседу на выходе.

При этом (i + 1) -е сообщение отправляется каждому соседу на выходе только после того, как были получены i сообщений от каждого соседа на входе.

Как только от каждого соседа будет получено в точности D сообщений, процесс завершает алгоритм и принимает decision.

Лемма 7.1

Суть: даже если канал не является очередью (при волновом алгоритме), то соотношение взаимозависимости между событиями сохраняется.

Теорема о фазовом алгоритме.

Фазовый алгоритм – это волновой алгоритм.

Алгоритм Финна

Алгоритм Финна — это еще один волновой алгоритм, который можно использовать для произвольных ориентированных сетей.

Ограничения:

1. Процессы должны однозначно друг друга идентифицировать.
2. **Не** нужно знать диаметра сети
3. В сообщениях процессы обмениваются отличительными признаками, и это приводит к тому, что битовая сложность алгоритма становится достаточно большой.

Основная идея.

Процесс p формирует два множества отличительных признаков Incp и NIncp

* Incp — это множество таких процессов q, что некоторое событие в q предшествует (по отношению <=) самому последнему событию, случившемуся в p.
* NIncp — множество таких процессов q, что у каждого соседа г процесса q какое-нибудь событие в r предшествует самому последнему событию, случившемуся в p.

Множества Incp и NIncp формируются так:

1. Вначале Incp = {p} и NIncp = Ø.
2. Процесс p отправляет сообщения, содержащие Incp и NIncp всякий раз, когда одно из этих множеств расширяется.
3. Когда p получает сообщение с множествами Inc и NInc полученные отличительные признаки добавляются к мнoжecтвaм Incp и NIncp
4. Если p получил сообщения от всех своих соседей на входе, то отличительный признак p вставляется в NIncp
5. Если Incp = NIncp, то p принимает решение.

Это означает, что каков бы ни был процесс q , если некоторое событие в q предшествует dp, то у всякого соседа r процесса q также произошло какое-нибудь событие, предшествующее dp. Отсюда следует, что для нашего алгоритма требование зависимости соблюдено.

Теорема об алгоритме Финна:

Алгоритм Финна — это волновой алгоритм.

Особый интерес представляют централизованные волновые алгоритмы, в которых все события линейно упорядочены по отношению причинно-следственной зависимости, и решение принимает инициатор. Волновые алгоритмы такого рода называются алгоритмами обхода.

Алгоритмом обхода называется алгоритм, обладающий следующими тремя свойствами.

1. В каждом вычислении единственный инициатор запускает алгоритм, отправляя одно-единственное сообщение.
2. Всякий процесс после получения сообщения либо отправляет ровно одно сообщение, либо принимает решение.
3. Алгоритм завершает работу в инициаторе, и к тому моменту, когда это происходит, каждому процессу хотя бы один раз удалось отправить сообщение.

Алгоритм Тарри

Алгоритм обхода произвольных связных сетей был предложен Тарри в 1895. Этот алгоритм определяется двумя следующими правилами.

R1. Процесс не передает маркер по одному и тому же каналу дважды.

R2. He-инициатор передает маркер своей родительской вершине (соседу, от которого он впервые получил маркер) только в том случае, когда его невозможно передать по другим каналам согласно правилу R1.

Теорема об алгоритме Тарри: Алгоритм Тарри — это алгоритм обхода

Стягивающее ребро — это ребро, не принадлежащее остовному дереву.

Корневое остовное дерево T тети G называется деревом поиска в глубину, если для каждого стягивающего ребра pq выполняется соотношение q ∈ T [p] V q ∈ A[p] – т.е. либо q – вершина поддерева вершины p, либо q – вершина предшественник вершины p.

Классический алгоритм поиска в глубину получается из алгоритма Тарри, в котором свобода выбора очередного соседа для передачи ему маркера ограничена третьим правилом следующего вида.

R3. Всякий раз, когда процесс получает маркер, он возвращает его назад по тому же каналу, если это не противоречит правилам R1 и R2.

Теорема о классическом алгоритме обхода в глубину

Классический алгоритм поиска в глубину строит остовное дерево поиска в глубину, используя при этом 2|E| обменов сообщениями.

Алгоритм Авебаха

Когда маркер впервые достается процессу p (для инициатора — в момент запуска алгоритма), p оповещает каждого соседа r кроме родительской вершины, отправляя ему сообщение (vis).

Маркер не передается, до тех пор, пока p не получит сообщение (ack) от всех своих соседей. Это гарантирует, что каждый сосед r процесса p осведомлен к моменту отправления маркера процессом p о том, что этот маркер уже побывал у p. Когда позднее маркер достигнет r, процесс r уже не будет отправлять маркер процессу p , если только p не является родительской вершиной для r

Теорема об алгоритме Авербаха

Алгоритм Авербаха строит дерево поиска в глубину за 4N — 2 единиц времени, проводя при этом 4|E| обменов сообщениями.

Алгоритм Сидона.

Основная идея:

1. Один инициатор посылает токен в начальный момент времени
2. mrc – это обратный указатель к указателю father, т.е. указатель на потомка.
3. Сообщение vis шлётся всем, при первом получении узлом токена
4. Если какой-то узел получил сообщение vis от другого узла, то это направление запоминается и блокируется, чтобы туда нельзя было больше послать токен (ибо там токен уже побывал, его туда слать незачем)
5. Аналогично алгоритму поиска в глубину – мы пытаемся вернуть токен как можно быстрее тому, от кого последний раз его получили, чтобы в итоге наше дерево получилось *деревом поиска в глубину.*
6. Если так получится, что токен и vis встретятся в канале друг на встречу другу (такое теоретически может случиться (из-за долгой задержки vis в канале)), то такое отловится первыми «if» в функциях обработки второй и третей. В этом случае токен будет выкинут, а данный vis станет новым токеном и продолжет ходить по графу.

Теорема об алгоритме Сидона

Алгоритм Сидона строит DFS дерево за 2N — 2 единицы времени, используя 4|E| обменов сообщениями

1. Задача избрания лидера. Основные определения и допущения. Волновые алгоритмы избрания лидера. Выборы лидера на дереве. Выборы в кольцах. Алгоритм Ле-Ланна. Алгоритм Ченя - Робертса. Алгоритм Петерсона/Долева-Клейва-Роде. Эффект угасания.

Задача избрания лидера состоит в том, чтобы, из конфигурации, в которой все процессы пребывают в одном и том же состоянии, достичь такой конфигурации, в которой ровно один процесс будет находиться в состоянии leader , а все остальные процессы — в состоянии lost .

Алгоритмом избрания лидера называется алгоритм, который обладает следующими свойствами.

1. Каждый процесс наделен одним и тем же локальным алгоритмом.
2. Алгоритм является децентрализованным, т.е. вычисление может быть инициировано произвольным непустым подмножеством процессов.
3. Алгоритм достигает заключительной конфигурации в каждом вычислении, и в каждой заключительной конфигурации существует ровно один процесс, который находится в состоянии leader а все остальные процессы при этом пребывают в состоянии lost.

Допущения для задачи избрания лидера:

1. Система вполне асинхронна
2. Каждый процесс опознается по уникальному имени, и это имя изначально известно самому процессу.
3. Каждое сообщение может содержать O(w) бит.

Выборы лидера в дереве – сначала идёт процесс *побудки*, а потом все листья инициируют древесный волновой алгоритм.

Теорема 8.1.

Предложенный алгоритм решает задачу о выборах на древесных сетях с использованием O(N) обменов сообщениями, затрачивая при этом O(D) единиц времени.

Выбор лидера в кольцах:

1. Алгоритм Ле-Ланна – O(N2) обменов за O(N) времени – каждый инициатор вычисляет список отличительных признаков всех инициаторов, после чего лидером избирается инициатор с наименьшим признаком.
2. Алгоритм Ченя-Робертса – O(N\*logN) – в среднем и O(N2) – в худшем, если все процессы инициаторы, то в среднем ≈ 0.69\*N\*log N

Алгоритм Ченя и Робертса улучшает алгоритм Ле-Ланна за счет того, что из кольца изымаются все маркеры тех процессов, относительно которых уже становится ясно, что они проиграют выборы.

1. Алгоритм Петерсона/Долева-Клейва-Роде – O(N\*logN) – для однонаправленного кольца.

Суть в том, что признаки беруться среди вершины и её соседей, и выбирается тот, чей признак минимален, тем самым сокращение на каждом шаге – туре, происходит в 2 раза (экспоненциальное падение).

(лучше картинка со слайда 35)

1. Теоретическая нижняя оценка ≈ 0.34\*N\*log N

Теорема 8.6.

Если

* кольцо — однонаправленное,
* процессы не осведомлены о размерах кольца,
* в каналах поддерживается очередность сообщений,
* все процессы являются инициаторами,

то сложность в среднем всякого алгоритма избрания лидера N будет не меньше, чем N•НN, где HN = .

Теорема 8.7.

Всякий алгоритм избрания лидера на основе сравнения для произвольных сетей имеет сложность (и в среднем, и в наихудшем случае) не меньшую, чем Ω(|E| + N\*logN)

Следствие.

Всякий децентрализованный волновой алгоритм для произвольных сетей без предварительной осведомленности о соседях имеет сложность по числу обменов сообщениями, не меньшую чем Ω(|E| + N\*logN)

Эффект угасания

Алгоритм избрания лидера можно получить из произвольного волнового алгоритма при помощи конструкции, которая носит название “угасание”.

Каждый инициатор запускает отдельную волну. Чтобы отличаться от сообщений других волн, сообщения, движущиеся по волне, запущенной процессом p, должны быть помечены отличительным признаком процесса p.

Данный алгоритм гарантирует, что независимо от количества запущенных волн только одна волна приведет к решению, а именно, та волна, которую запустил инициатор с самым младшим отличительным признаком. Все остальные волны будут прерваны еще до того, как будет принято решение.

Суть в том, что каждый элемент графа пропускает только ту волну через себя, которая наилучшая из тех, что он видел, любую другую он через себя не пропустит, в итоге только та волна закончит своё действие, которую пропустили все элементы графа, т.е. наилучшая, остальные волны никогда не завершатся.

1. Задача избрания лидера. Нижние оценки сложности. Оптимальные выборы. Алгоритм Галладжера-Хамблета-Спиры (GHS). Глобальное описание алгоритма GHS. Подробное описание алгоритма GHS. Алгоритм Корача-Каттена-Морана.

Утверждение

Алгоритм избрания лидера на дереве решает задачу о выборах на древесных сетях с использованием O(N) обменов сообщениями.

Из этой теоремы следует, что CE < CT + O(N)

Если в нашем распоряжении есть лидер, то остовное дерево можно построить с использованием 2|E| обменов сообщениями при помощи централизованного алгоритма обхода сети.

Поэтому справедливо неравенство CT < CE + 2|E|

**Таким образом**, для оптимального выбора лидера необходимо и достаточно уметь оптимально строить остовное дерево.

Алгоритм Галладжера-Хамблета-Спиры (GHS)

Допущения:

1. Каждому ребру приписан уникальный вес ш(в) . Все веса ребер линейно упорядочены.
2. Все узлы пребывают первоначально в состоянии оцепенения и пробуждаются перед началом выполнения алгоритма. Некоторые узлы пробуждаются самопроизвольно, другие могут получить сообщение по ходу работы алгоритма, еще пребывая в оцепенении. При этом узел, получивший сообщение, вначале выполняет процедуру локальной инициализации, а затем приступает к обработке этого сообщения.

Вес остовного дерева T в графе G полагается равным сумме весов всех N — 1 ребер, входящих в состав T. При этом T называется минимальным остовным деревом, или сокращенно MST, если ни одно остовное дерево не имеет вес меньший, чем T.

Утверждение 9.1.

Если все веса ребер попарно различны, то существует только одно MST.

Фрагмент - произвольное поддерево MST. Ребро e называется исходящим ребром фрагмента F, если один из концов e принадлежит F, а другой нет.

Утверждение 9.2.

Если F является фрагментом, и e - ребро наименьшего веса, исходящее из F, то F U {е} также является фрагментом.

Глобальное описание алгоритма GHS:

1. Формируется такое семейство фрагментов, что объединение их содержит все узлы сети.
2. Первоначально это семейство состоит из всех узлов сети, каждый из которых рассматривается как граф с одним узлом.
3. Узлы всякого фрагмента вступают во взаимодействие с целью выявления исходящего из фрагмента ребра с наименьшим весом.
4. Как только будет определено исходящее из фрагмента ребро с наименьшим весом, данный фрагмент соединяется с другим фрагментом путем добавления этого исходящего ребра, которое строится в результате взаимодействия этих двух фрагментов.

Для эффективной реализации нужны:

1. Имя фрагмента
2. Соединение старшего и младшего фрагментов (ранг результирующего фрагмент равен рангу большего из фрагментов, если фрагменты имели одинаковый ранг, то новый ранг будет на 1 больше)
3. Ранги фрагментов

Утверждение 9.3.

Если соблюдать указанные выше правила соединения фрагментов, то суммарно во всех процессах перемена имени или ранга случится не более N\*log N раз.

Подробное описание алгоритма GHS <см. слайды с 41 лекции 10>

Теорема.

Алгоритм GHS вычисляет минимальное остовное дерево, используя при этом не более (5N\*log N + 2|E|) обменов сообщениями.

Алгоритм Корача-Каттена-Морана – алгоритм метода угасания, но с оптимизациями.

Чтобы «гасить» обходы, алгоритм оперирует на разных уровнях, подобно тому как алгоритм Петерсона/Долев-Клейва-Роде разбивает свое вычисление на туры.

Если запущены по крайней мере два обхода, то один из маркеров рано или поздно достигнет того процесса, в котором уже успел побывать другой. Тогда обход, который проводится при помощи вновь прибывшего маркера, будет прерван.

Цель работы алгоритма состоит в том, чтобы привести две маркера вместе в один процесс, где они будут подавлены, и после этого может начаться новый обход.

1. Задача обнаружения завершения вычислений. Алгоритм Дейкстры-Шолтена. Алгоритм Шави-Франчеза. Алгоритм Сафры. Алгоритм возвращения кредитов.

Вычисление завершается явно, если все процессы пребывают в заключительном состоянии. Тогда говорят, что произошло завершение работы алгоритма.

Вычисление завершается неявно, если все каналы пусты, но при этом не все процессы пребывают в заключительном состоянии, т.е. некоторые процессы ожидают поступления сообщений. Тогда говорят, что произошло завершение обмена сообщениями.

Упрощения для задачи обнаружения завершения вычислений.

1. Активный процесс становится пассивным только после осуществления внутреннего события.
2. Процесс всегда становится активным после получения сообщения.
3. Внутренние события, приводящие к тому, что процесс p становится пассивным, — это единственно возможные внутренние события в процессе p.

Теорема 1

Будем полагать, что предикат term (Y) обращается в истину на всякой конфигурации y, на которой не может произойти ни одного события в базовом вычислении.

term(Y) <=> (Для любого p принадлежащего P: statep = passive) && (Для любого pq принадлежащего E: Mpq не содержит сообщения <mes>)

Задача обнаружения вычисления заключается в присоединении к системе контрольного алгоритма, который приведет все процессы в заключительные состояния, после того как базовое вычисление достигнет заключительной конфигурации.

Контрольный алгоритм состоит из алгоритма обнаружения завершения вычисления и алгоритма оповещения о завершении вычисления.

Требования к алгоритму обнаружения завершения вычисления:

1. Невмешательство
2. Живость - если выполняется условие term, то алгоритм оповещения Announce должен быть вызван спустя конечное число шагов.
3. Безопасность - если вызван алгоритм Announce, то конфигурация должна удовлетворять условию term.

Теорема 2.

Для всякого алгоритма обнаружения завершения вычисления существует такое базовое вычисление, осуществляющее M обменов базовыми сообщениями, для обнаружения завершения которого рассматриваемый алгоритм совершает не менее M обменов контрольными сообщениями.

Теорема 3.

Для обнаружения завершения децентрализованного базового вычисления в худшем случае требуется совершить обмен не менее чем W контрольными сообщениями, где W — коммуникационная сложность волнового алгоритма.

Алгоритм Дейкстры-Шолтена

Условия применения:

* базовый алгоритм централизованный;
* топология сети произвольная;
* сеть неориентированная;
* контрольный алгоритм централизованный и инициируется в том же процессе, что и базовый алгоритм.

Алгоритм обнаружения завершения строит и постоянно обновляет дерево вычислений T = (VT, ET), которое обладает следующими двумя свойствами.

1. Граф T либо является пустым, либо представляет собой ориентированное дерево, корнем которого является вершина р0
2. Множество VT включает все активные процессы и все базовые сообщения, находящиеся на этапе пересылки.

Инициатор р0 вызывает процедуру Announce, если р0 yt принадлежит VT; согласно первому свойству граф T в таком случае является пустым, а согласно второму свойству это означает, что вычисление завершилось.

Когда процесс p отправляет базовое сообщение <mes>, это сообщение добавляется к дереву в качестве вершины <mes>, причем родительской вершиной для нее становится процесс p.

Когда процесс p, не входящий в состав дерева, становится активным после получения сообщения от процесса q, вершина q становится родительской вершиной для процесса p.

Чтобы обозначить явно отправителя сообщения, всякое базовое сообщение <mes>, отправленное процессом q, будет обозначаться записью cmesq.

Вершины из дерева T удаляются по двум причинам:

1. Базовое сообщение удаляется после его получения.
2. Чтобы контрольный алгоритм сработал, дерево должно исчезнуть после завершения базового алгоритма. Сообщения — листья дерева T; в каждом процессе есть переменная для подсчета числа вершин-последователей этого процесса в дереве T. Удаление из дерева вершины-последователя процесса p осуществляется в другом процессе q в результате того, что вершина-последователь
   1. либо соответствует сообщению, достигшему процесса q
   2. либо соответствует самому процессу q

Чтобы в процессе p не нарушался учет вершин-последователей, ему отправляется сигнальное сообщение sigp при удалении вершины-последователя процесса p. Это сообщение замещает удаленную вершину-последователя процесса p, а удаление этой вершины из дерева осуществляется самим процессом p после получения указанного сигнала; при этом p уменьшает значение счетчика последователей на 1.

Теорема 1.

Алгоритм Дейкстры-Шолтена является корректным алгоритмом обнаружения завершения вычисления; в нем используется M обменов контрольными сообщениями.

(доказывается через инвариант)

Алгоритм Шави—Франчеза

Обобщает алгоритм Дейкстры-Шолтена для случая децентрализованных базовых вычислений.

В алгоритме Шави-Франчеза граф вычислений является лесом, каждое дерево которого имеет в качестве корня один из инициаторов базового вычисления.

Для обозначения дерева, корнем которого служит процесс p будем использовать запись Tp.

В алгоритме Шави-Франчеза строится такой граф F = (VF, EF), что

1. либо F пуст, либо F представляет собой лес, каждое дерево которого имеет один из инициаторов в качестве корня,
2. множество VF содержит все активные процессы и все базовые сообщения, пребывающие на этапе пересылки.

Завершение базового вычисления будет обнаружено, когда указанный граф становится пустым.

Проверка того, что все деревья исчезли, проводится при помощи одной-единственной волны.

Лес F строится так, чтобы всякое дерево Tp, ставшее пустым, оставалось пустым и в дальнейшем. Это не препятствует процессу p вновь становиться активным, но если p становится активным после исчезновения его собственного дерева, то он заносится в дерево другого инициатора.

Каждый процесс принимает участие в распространении волны только тогда, когда его дерево исчезает; и если волна приводит к тому, что принимается решение, то вызывается процедура оповещения Announce.

Переменная scp используется для подсчета числа последователей вершины p в дереве T.

Булева переменная emptyp принимает значение true тогда и только тогда, когда дерево, в котором содержится вершина p становится пустым.

Примечания.

1. В приведенном описании волновой алгоритм в явном виде не выделен.
2. Волновой алгоритм запускают только инициаторы базовых вычислений
3. Все процессы проводят параллельное выполнение волнового алгоритма, причем отправление сообщений или принятие решения разрешается только тем процессам p, у которых переменная emptyp имеет значение true
4. При осуществлении события decide вызывается процедура Announce.

Теорема 2.

Алгоритм Шави-Франчеза является корректным алгоритмом обнаружения завершения вычисления; в нем используется M + W обменов контрольными сообщениями.

(доказывается через инвариант)

Алгоритм Сафры

Условия применения:

1. Базовый алгоритм децентрализованный.
2. Каналы связи однонаправленные.
3. В сети существует гамильтонов контур, по которому распространяются контрольные сообщения; базовые сообщения могут передаваться по любым каналам.
4. Контрольный алгоритм — централизованный волновой алгоритм в однонаправленном кольце.

Принцип работы:

1. Каждый процесс может иметь некоторый цвет, который задается переменной color; эта переменная может иметь одно из двух значений white или black.
2. Контрольные сообщения (маркеры) также окрашиваются в один из двух цветов white или black.
3. Маркер передается только пассивными процессами.
4. Инициализатор контрольного алгоритма запускает маркер цвета white
5. После приема базового сообщения процесс окрашивается в цвет black
6. Процесс цвета black окрашивает маркер в цвет black
7. После передачи маркера процесс окрашивается в цвет white
8. Если инициализатор принимает маркер цвета white, то он объявляет о завершении базового алгоритма.
9. Каждый процесс p снабжен счетчиком базовых сообщений mcp, который увеличивается на 1 при отправлении процессом базового сообщения и уменьшается на 1 при приеме базового сообщения.
10. Маркер при прохождении через процессы суммирует показания их счетчиков.
11. Завершение работы базового алгоритма регистрируется, если сумма собранных маркером показаний счетчиков равна 0.

Теорема 3.

Алгоритм Сафры является корректным алгоритмом обнаружения завершения вычисления.

(доказывается через инвариант)

Алгоритм возвращения кредита

Условия применения:

Алгоритм обнаруживает завершение централизованного вычисления при условии, что каждый процесс способен отправить сообщение непосредственно самому инициатору вычисления (т. е., сеть содержит подграф, имеющий форму звезды, в центре которой расположен инициатор).

Принцип работы:

В этом алгоритме каждому сообщению и каждому процессу вручается некоторый *кредит*; его значением служит вещественное число, расположенное между 0 и 1 (включительно). Алгоритм стремится сохранить инвариантность следующих утверждений.

1. Суммарный кредит, выданный сообщениям и процессам, равен 1.
2. Всякому базовому сообщению вручается положительный кредит.
3. Всякому активному процессу вручается положительный кредит.

Процессы, имеющие положительный кредит и не подпадающие под перечисленные законы (т. е. пассивные процессы), возвращают кредит инициатору. Инициатор поступает подобно банку и собирает все кредиты, которые ему отправляют, в переменной ret (возвращённые кредита).

Когда инициатор владеет всеми кредитами, то он сообщает о завершении базового алгоритма.

Теорема 4.

Алгоритм возвращения кредита является корректным алгоритмом обнаружения завершения вычислений.

(доказательство через инвариант)

Алгоритм Раны

Для решения задачи обнаружения завершения вычисления можно использовать штемпели времени. Предполагается, что все процессы снабжены часами; для этой цели пригодны как аппаратные таймеры, так и логические часы Лампорта.

В основу алгоритма положен локальный предикат quiet(p), определенный для каждого процесса p так, что:

quiet(p) 🡪 statep = passive && никакое базовое сообщение, отправленное p не пребывает на этапе пересылки,

откуда следует импликация (Для любого p quiet(p)) => term

Условие quiet определяется следующим соотношением: quiet(p) := (statep = passive || unackp = 0).

Цель алгоритма — проверить, верно ли, что в определенный момент времени t все процессы удовлетворяют условию quiet.

Процесс p, перейдя в состояние quiet, сохраняет в памяти тот момент времени qtp, когда произошло это событие, и запускает волну, чтобы проверить, все ли процессы уже перешли в состояние quiet к моменту времени qtp

Если это так, то завершение вычисления считается обнаруженным. В противном случае существует процесс, который перейдет в состояние quiet позже, и он запустит новую волну.

Теорема 5.

Алгоритм Раны является корректным алгоритмом обнаружения завершения вычислений.

(доказательство через инвариант)

1. Задача сохранения моментального состояния. Алгоритм Чанди-Лампорта Алгоритм Лаи-Янга. Применение алгоритмов сохранения моментального состояния.

Конфигурация системы — это набор конфигураций составляющих ее процессов плюс содержание каналов связи.

Необходимо уметь сохранять конфигурации согласованно.

Множество реализуемых конфигураций не является инвариантом отношения эквивалентности, и поэтому оно не определяется вычислением однозначно. Таким образом, мы будем считать разумным всякий результат работы алгоритма, если он приводит к построению какой-либо конфигурации, которая окажется возможной для заданного вычисления (т. е. может случится в некоторой реализации этого вычисления).

Для чего необходимо сохранение моментального состояния системы (snapshot)

1. Проверка свойств после в режиме offline над snapshot-ом

Свойство P конфигураций называется устойчивым, если справедливо соотношение P(Конф.1) && (Конф.1 -> Конф.2) => P(Конф.2)

1. Использование snapshot вместо начальных конфигураций.
2. Использование snapshot для отладки распределённой программы.

Допущение:

1. Слабая справедливость вычислений.
2. Сеть – сильно связна

Распределённая система C.

Множество процессов P.

Множество событий вычисления Ev.

На множестве событий процесса p вводится причинно-следственный порядок <<p

Очевидным образом вводятся значения словосочетаний «моментальное локальное состояние процесса p», «предмоментальное событие», «постмоментальное событие».

Моментальное состояние системы состоит из

1. Моментальные состояния всех процессов
2. Моментальные состояния всех каналов (могут быть случаи, когда сообщение находится на этапе пересылки, а могут быть случаи, когда один из процессов уже зарегистрировал получение сообщения, а другой из процессов не зарегистрировал отправки этого сообщения)

**Осуществимое** моментальное состояние: если для всех пар соседних процессов p и q, rsvdpq подмножество sentpq

Сечением называется совокупность событий вычисления, замкнутая влево относительно отношения локальной причинно-следственной зависимости.

Сечение L2 считается более поздним, чем сечение L1, если верно включение L1 вложено в L2.

Сечение называется **согласованным**, если оно замкнуто влево относительно отношения причинно-следственной зависимости.

Моментальное состояние S\* называется **значимым** в вычислении C, если существует такая реализация E ∈ C, что Y\* является конфигурацией E.

Теорема 1.

Пусть S\* — моментальное состояние системы и L — сечение, порожденное S\*. Тогда равносильны следующие три утверждения:

1. S\* осуществимо;
2. L согласовано;
3. S\* значимо.

(доказывается 1 => 2 => 3 => 1)

Алгоритм Чанди-Лампорта

По теореме 1 достаточно скоординировать локальные моментальные состояния, чтобы выполнялись следующие два свойства:

1. выбранные в каждом процессе локальные моментальные состояния должны быть реализованы.
2. получение постмоментального сообщения не может составлять предмоментальное событие.

Сообщения самого вычисления – базовые сообщения.

Сообщения алгоритма построения моментального состояния – контрольные сообщения.

Допущения:

1. Сильно связная сеть
2. Каналы могут быть однонапавленными
3. Каналы сохраняют очерёдность собщений

Идея алгоритма:

В алгоритме Ченди—Лэмпорта процессы сообщают друг другу о построении моментальных состояний путем отправления специальных сообщений (маркеров) (mkr) по каждому каналу.

Каждый процесс, запечатлевая свое локальное состояние, отправляет маркер в точности один раз по каждому примыкающему к нему каналу; маркеры рассматриваются как контрольные сообщения.

Получение сообщения (mkr) каким-либо процессом, который еще не запечатлел свое моментальное состояние, приводит к тому, что этот процесс запечатлевает свое моментальное состояние и также отправляет маркер (mkr), который выполняется параллельно с вычислением C.

Лемма

Если хотя бы один процесс запустит алгоритм Чанди-Лампорта, то спустя конечный промежуток времени все процессы запечатлеют свое локальное моментальное состояние.

Теорема 2

Алгоритм Ченди—Лэмпорта вычисляет значимое моментальное состояние системы.

Алгоритм Лаи-Янга

Допущения:

1. Каналы – **не** обязательно очередь

Идея алгоритма:

Каждое базовое сообщение снабжается пометкой, которая позволяет понять, является ли это сообщение предмоментальным или постмоментальным.

Для этого процесс p, отправляя сообщение по ходу вычисления C, добавляет к нему значение takenp . Так как содержание сообщений, относящихся к вычислению C, не играет здесь никакой роли, мы будем обозначать такие сообщения просто (mes, c), где c — это значение переменной, включенное в состав сообщения процессом-отправителем. Алгоритм построения моментального состояния проверяет поступающие сообщения и записывает моментальное локальное состояние, и в дальнейшем ко всем своим сообщения приписывает takenp. Следует считать, что принятое сообщение с поднятым флагом takenp – это постмоментальное сообщение и записывать его в конфигурацию не нужно.

Недостатки алгоритма:

1. В алгоритме Лаи-Янга нет обмена контрольными сообщениями, и он не может гарантировать того, что каждый процесс рано или поздно запишет свое состояние.

Теорема 3.

Алгоритм Лая—Янга вычисляет только значимые моментальные состояния системы.

Примеры устойчивых свойств, которые можно проверять на записанных конфигурациях:

1. Завершение вычисления
2. Наличие тупиков
3. Потеря маркеров
4. Наличие «мусора»