Распределенные информационно-аналитические системы

Лекция № 10.

Определение и использование волновых алгоритмов

Профессор кафедры КБ-2: д.т.н. Шатовкин Р.Р.

Учебные цели:

Изучить основы научных знаний по определению волновых алгоритмов; элементарным результатам в волновых алгоритмах; распространению информации с обратной связью; синхронизации; вычислению функций инфинума.

Учебные вопросы:

1. Определение волновых алгоритмов.

2. Элементарные результаты о волновых алгоритмах.

3. Распространение информации с обратной связью.

4. Синхронизация.

5. Вычисление функций инфинума.

При разработке распределенных алгоритмов для различных приложений часто в качестве подзадач возникает несколько общих проблем для сетей процессов. Эти элементарные задачи включают в себя широковещательную рассылку информации – broadcasting (например, посылка начального или заключительного сообщения), достижение глобальной синхронизации между процессами, запуск выполнения некоторого события в каждом процессе, или вычисление функции, входные данные которой распределены между процессами. Эти задачи всегда решаются путем пересылки сообщений согласно некоторой, зависящей от топологии схеме, которая гарантирует участие всех процессов. Эти задачи настолько фундаментальны, что можно дать решения более сложных задач, таких как выбор, или взаимное исключение, в которых связь между процессами осуществляется только через эти схемы передачи сообщений.

Важность схем передачи сообщений, называемых ***волновыми алгоритмами***, оправдывает их рассмотрение отдельно от конкретного прикладного алгоритма, в который они могут быть включены.

В лекции формально определяются волновые алгоритмы и доказываются некоторые общие результаты о них.

Несмотря на то, что волновые алгоритмы обычно используются как подзадачи в более сложных алгоритмах, их полезно рассматривать отдельно.

Во-первых, введение новых понятий облегчает последующее рассмотрение более сложных алгоритмов, так как свойства их подзадач уже изучены.

Во-вторых, определенные задачи в распределенных вычислениях могут быть решены с помощью универсальных конструкций, в качестве параметров которых могут использоваться конкретные волновые алгоритмы. Тот же метод может использоваться для получения алгоритмов для различных сетевых топологий или для различных предположений о начальном знании процессов.

Сделаем ряд допущений.

Считается, если не указано обратное, что сетевая топология фиксирована (не происходит топологических перемен), не ориентирована (каждый канал передает сообщения в обоих направлениях) и связна (существует путь между любыми двумя процессами). Множество всех процессов обозначено через P, а множество каналов – через E.

Предполагается, что система использует асинхронную передачу сообщений и не существует понятия глобального или реального времени. Рассматриваемые в данном разделе алгоритмы также могут быть использованы в случае синхронной передачи сообщений (возможно с некоторыми изменениями во избежание тупиков) или с часами глобального времени, если они доступны. Однако некоторые более общие теоремы в этих случаях неверны.

1. Определение волновых алгоритмов

Распределенные алгоритмы обычно допускают большой набор возможных вычислений благодаря недетерминированности как в процессах, так и в подсистеме передачи.

Вычисление – это набор событий, частично упорядоченных отношением каузального (причинно-следственного) предшествования π. Количество событий в вычислении C обозначается через |C|, а подмножество событий, происходящих в процессе p, обозначается через C_p. Считается, что существует особый тип внутренних событий, называемый decide (принять решение); в рассматриваемых далее алгоритмах такое событие представляется просто утверждением decide . Волновой алгоритм обменивается конечным числом сообщений и затем принимает решение, которое каузально зависит от некоторого события в каждом процессе.

Определение 1. ***Волновой алгоритм*** – это распределенный алгоритм, который удовлетворяет следующим трем требованиям:

a) *Завершение*. Каждое вычисление конечно:

$\forall C : |C| < \infty$.

b) *Принятие решения*. Каждое вычисление содержит хотя бы одно событие decide:

$\forall C : \exists e \in C : e$ – событие decide.

c) *Зависимость*. В каждом вычислении каждому событию decide каузально предшествует какое-либо событие в каждом процессе:

$\forall C : \forall e \in C : (e$ – событие decide $\Rightarrow \forall q \in P \exists f \in C_q : f \pi e)$.

Вычисление волнового алгоритма называется ***волной***. Кроме того, в вычислении алгоритма различаются ***инициаторы***, также называемые ***стартерами***, и ***не-инициаторы***, называемые ***последователями***.

Процесс является инициатором, если он начинает выполнение своего локального алгоритма самопроизвольно, то есть при выполнении некоторого условия, внутреннего по отношению к процессу. Не-инициатор включается в алгоритм только когда прибывает сообщение и вызывает выполнение локального алгоритма процесса. Начальное событие инициатора – внутреннее событие или событие посылки сообщения, начальное событие не-инициатора – событие получения сообщения.

Существует множество волновых алгоритмов, так как они могут различаться во многих отношениях. Для обоснования алгоритмов и в качестве помощи в выборе одного из них для конкретной цели рассмотрим ***аспекты, отличающие волновые алгоритмы друг от друга***:

1) ***Централизация***. Алгоритм называется ***централизованным***, если в каждом вычислении должен быть ровно один инициатор, и ***децентрализованным***, если алгоритм может быть запущен произвольным подмножеством процессов.

Централизованные алгоритмы также называют *алгоритмами одного источника*, а *децентрализованные – алгоритмами многих источников*. Централизация существенно влияет на сложность волновых алгоритмов.

2) ***Топология***. Алгоритм может быть разработан для конкретной топологии, такой как кольцо, дерево, клика и т.д.

3) ***Начальное знание***. Алгоритм может предполагать доступность различных типов начального знания в процессах. Примеры требуемых заранее знаний:

(a) *Идентификация процессов*. Каждому процессу изначально известно свое собственное уникальное имя.

(b) *Идентификация соседей*. Каждому процессу изначально известны имена его соседей.

(c) *Чувство направления* (sense of direction).

4) ***Число решений***. Во всех волновых алгоритмах в каждом процессе происходит не более одного решения. Количество процессов, которые выполняют событие decide, может быть различным; в некоторых алгоритмах решение принимает только один процесс, в других – все процессы. В древовидном алгоритме решают ровно два процесса.

5) ***Сложность***. Меры сложности – это количество передаваемых сообщений (message complexity), количество передаваемых бит (bit complexity) и время, необходимое для одного вычисления.

Каждый рассматриваемый далее волновой алгоритм будет дан вместе с используемыми переменными и, в случае необходимости, с информацией, передаваемой в сообщениях.

Большинство этих алгоритмов посылают «пустые сообщения», без какой-либо реальной информации: сообщения передают причинную связь, а не информацию.

Некоторые алгоритмы используют сообщения для передачи нетривиальной информации, другие – используют различные типы сообщений; при этом требуется, чтобы каждое сообщение содержало 1–2 бита для указания типа сообщения.

Обычно при применении волновых алгоритмов в сообщение могут быть включены дополнительные переменные и другая информация.

Многие приложения используют одновременное или последовательное распространение нескольких волн. В этом случае в сообщение должна быть включена информация о волне, которой оно принадлежит.

Кроме того, процесс может хранить дополнительные переменные для управления волной, или волнами, в которых он в настоящее время активен.

Важный подкласс волновых алгоритмов составляют централизованные волновые алгоритмы, обладающие двумя дополнительными качествами: инициатор является единственным процессом, который принимает решение; и все события совершенно упорядочены каузальными отношениями. Такие волновые алгоритмы называются ***алгоритмами обхода***.

2. Элементарные результаты о волновых алгоритмах

Структурные свойства волн. Каждому событию в вычислении предшествует событие в инициаторе.

Лемма 1. Для любого события $e \in C$ существует инициатор p и событие f в C_p такое, что $f \pi e$.

Доказательство. Выберем в качестве f минимальный элемент в предыстории e, то есть такой, что $f \pi e$ и не существует $f' \pi f$. Такое f существует, поскольку предыстория каждого события конечна. Остается показать, что процесс p, в котором находится f, является инициатором.

Для начала, заметим, что f – это первое событие p, иначе более раннее событие p предшествовало бы f.

Первое событие не-инициатора – это событие получения сообщения, которому предшествовало бы соответствующее событие посылки сообщения, что противоречит минимальности f. Следовательно, p является инициатором.

Волна с одним инициатором определяет остовное дерево сети, где для каждого не-инициатора выбирается канал, через который будет получено первое сообщение.

Лемма 2. Пусть C – волна с одним инициатором p; и пусть для каждого не-инициатора q $father_q$ – это сосед q, от которого q получает сообщение в своем первом событии. Тогда граф $T = (P, E_T)$, где $E_T = \{qr: q \neq p \ \& \ r = father_q\}$ – остовное дерево, направленное к p.

Доказательство. Так как количество вершин T превышает количество ребер на 1, достаточно показать, что T не содержит циклов. Это выполняется, так как если e_q – первое событие в q, из того, что $qr \in E_T$ следует, что $e_r \ \pi \ e_q$, а p – отношение частичного порядка.

В качестве события f в пункте (c) Определения 1 может быть выбрано событие посылки сообщения всеми процессами q, кроме того, где происходит событие decide.

Лемма 3. Пусть C – волна, а $d_p \in C$ – событие decide в процессе p. Тогда

$\forall\ q \neq p: \exists\ f \in C_q : (\ f\ \pi\ d_p$ & f – событие send)

Доказательство. Так как C – это волна, существует событие $f \in C_q$, которое каузально предшествует d_p; выберем в качестве f последнее событие C_q, которое предшествует d_p. Чтобы показать, что f – событие send, отметим, что из определения каузальности следует, что существует последовательность (каузальная цепочка)

$$f = e_0, e_1, ..., e_k = d_p,$$

такая, что для любого $i < k$, e_i и e_{i+1} – либо последовательные события в одном процессе, либо пара соответствующих событий send-receive. Так как f – последнее событие в q, которое предшествует d_p, e_1 происходит в процессе, отличном от q, следовательно f – событие send.

Нижние границы сложности волн. Из Леммы 3 непосредственно следует, что нижняя граница количества сообщений, которые передаются в волне, равна N – 1. Если событие decide происходит в единственном инициаторе волны (что выполняется всегда в случае алгоритмов обхода), граница равна N сообщениям, а волновые алгоритмы для сетей произвольной топологии используют не менее |E| сообщений.

Теорема 1. Пусть C – волна с одним инициатором p, причем событие decide d_p происходит в p. Тогда в C передается не менее N сообщений.

Доказательство. В соответствии с Леммой 1 каждому событию в C предшествует событие в p, и, используя каузальную последовательность, как в доказательстве Леммы 3, нетрудно показать, что в p происходит хотя бы одно событие send. В соответствии с Леммой 3 событие send также происходит во всех других процессах, откуда количество посылаемых сообщений составляет не меньше N.

Теорема 2. Пусть A – волновой алгоритм для сетей произвольной топологии без начального знания об идентификации соседей. Тогда A передает не менее |E| сообщений в каждом вычислении.

Доказательство. Допустим, A содержит вычисление C, в котором передается менее |E| сообщений; тогда существует канал xy, по которому в C не передаются сообщения (рисунок 1).

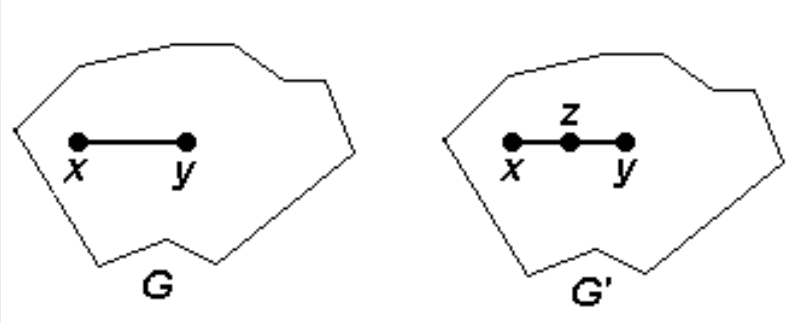

Рисунок 1 – Включение процесса в неиспользуемый канал

Рассмотрим сеть G′ , полученную путем включения одного узла z в канал между x и y. Так как узлы не имеют знания о соседях, начальное состояние x и y в G′ совпадает с их начальным состоянием в G. Это верно и для всех остальных узлов G. Следовательно, все события C могут быть применены в том же порядке, начиная с исходной конфигурации G′ , но теперь событию decide не предшествует событие в z.

3. Распространение информации с обратной связью

Рассмотрим применение волновых алгоритмов для случая, когда некоторая информация должна быть передана всем процессам и определенные процессы должны быть оповещены о завершении передачи.

Эта ***задача распространения информации с обратной связью (PIF – propagation of information with feedback)*** формулируется следующим образом: «Формируется подмножество процессов из тех, кому известно сообщение M (одно и то же для всех процессов), которое должно быть распространено, то есть все процессы должны принять M. Определенные процессы должны быть оповещены о завершении передачи; то есть должно быть выполнено специальное событие notify, причем оно может быть выполнено только когда все процессы уже получили сообщение M. Алгоритм должен использовать конечное количество сообщений».

Оповещение в PIF-алгоритме можно рассматривать как событие decide.

Теорема 3. Каждый PIF-алгоритм является волновым алгоритмом.

Доказательство. Пусть P – PIF-алгоритм. Из формулировки задачи каждое вычисление P должно быть конечным и в каждом вычислении должно происходить событие оповещения (decide). Если в некотором вычислении P происходит оповещение d_p, которому не предшествует никакое событие в процессе q, то существует выполнение P, в котором оповещение происходит до того, как q принимает какое-либо сообщение, что противоречит требованиям.

Теорема 4. Любой волновой алгоритм может использоваться как PIF-алгоритм.

Доказательство. Пусть A – волновой алгоритм. Чтобы использовать A как PIF-алгоритм, возьмем в качестве процессов, изначально знающих информацию M, стартеры (инициаторы) A. Информация M добавляется к каждому сообщению A. Это возможно, поскольку по построению стартеры A знают информацию M изначально, а последователи не посылают сообщений, пока не получат хотя бы одно сообщение, то есть пока не получат информацию M. Событию decide в волне предшествуют события в каждом процессе; следовательно, когда оно происходит, каждый процесс знает информацию M, и событие decide можно считать требуемым событием notify в PIF-алгоритме.

Построенный PIF-алгоритм имеет такую же сложность сообщений, как и алгоритм, описанный в первом учебном вопросе. Он обладает всеми другими качествами этого алгоритма, кроме битовой сложности.

Битовая сложность может быть уменьшена путем добавления информации M только к первому сообщению, пересылаемому через каждый канал. Если w – количество бит в информации M, битовая сложность полученного алгоритма превышает битовую сложность A на $w*|E|$.

4. Синхронизация

Рассмотрим применение волновых алгоритмов для случаев, когда должна быть достигнута глобальная синхронизация процессов.

Задача синхронизации (SYN) формулируется следующим образом: «В каждом процессе q должно быть выполнено событие a_q, и в некоторых процессах должны быть выполнены события b_p, причем все события a_q должны быть выполнены по времени раньше, чем будет выполнено какое-либо событие b_p. Алгоритм должен использовать конечное количество сообщений.

В SYN-алгоритмах события b_p будут рассматриваться как события decide.

Теорема 5. Каждый SYN-алгоритм является волновым алгоритмом.

Доказательство. Пусть S – SYN-алгоритм. Из формулировки задачи каждое вычисление S должно быть конечным, и в каждом вычислении должно происходить событие b_p (decide). Если в некотором вычислении S происходит событие b_p, которому каузально не предшествует a_q, тогда существует выполнение S, в котором b_p происходит ранее a_q.

Теорема 6. Любой волновой алгоритм может использоваться как SYN-алгоритм.

Доказательство. Пусть A – волновой алгоритм. Чтобы преобразовать A в SYN-алгоритм, потребуем, чтобы каждый процесс q выполнял a_q до того, как q пошлет сообщение в A или примет решение в A. Событие b_p происходит после события decide в p. Из Леммы 3, каждому событию decide каузально предшествует a_q для любого q.

Построенный SYN-алгоритм имеет такую же сложность сообщений, как и алгоритм, рассмотренный в первом учебном вопросе, и обладает всеми другими свойствами этого алгоритма.

5. Вычисление функций инфимума

Рассмотрим применение волновых алгоритмов для вычисления функций, значения которых зависят от входов каждого процесса.

В качестве представителей таких функций рассмотрим алгоритмы, вычисляющие инфимум по всем входам, которые должны быть извлечены из частично упорядоченного множества.

Если $(X, \leq)$ – частичный порядок, то c называют инфимумом a и b, если $c \leq a$, $c \leq b$ и $\forall\, d : (d \leq a \,\&\, d \leq b \Rightarrow d \leq c)$. Допустим, что X таково, что инфимум всегда существует; в этом случае инфимум является единственным и обозначается как $a \wedge b$. Так как $\wedge$ – бинарный оператор, коммутативный ($a \wedge b = b \wedge a$) и ассоциативный (то есть $a \wedge (b \wedge c) = (a \wedge b) \wedge c$), операция может быть обобщена на конечные множества:

$$\inf \{j_1, ..., j_k\} = j_1 \wedge ... \wedge j_k.$$

Задача вычисления инфимума формулируется следующим образом: «Каждый процесс q содержит вход j_q, являющийся элементом частично упорядоченного множества X. Потребуем, чтобы определенные процессы вычисляли значение $\inf \{j_q : q \in P\}$ и чтобы эти процессы знали, когда вычисление завершается. Они записывают результат вычисления в переменную out и после этого не могут изменять ее значение».

Событие write, которое заносит значение в out, рассматривается в INF-алгоритме как событие decide.

Теорема 7. Каждый INF-алгоритм является волновым алгоритмом.

Доказательство. Пусть I – INF-алгоритм. Предположим, что существует вычисление C алгоритма I с начальной конфигурацией Υ, в котором p записывает значение J в out_p и этому событию write не предшествует никакое событие в q. Рассмотрим начальную конфигурацию Υ' , которая аналогична Υ за исключением того, что q имеет другой вход j_q' , выбранный так, что $j_q' < J$.

Так как никакое применение входа q не предшествует каузально событию write процесса p в C, все события C, предшествующие событию write, применимы в том же порядке, начиная с Υ' . В полученном вычислении p записывает ошибочный результат J, так же как в C.

Теорема 8. Любой волновой алгоритм может быть использован для вычисления инфимума.

Доказательство. Допустим, что дан волновой алгоритм A. Назначим каждому процессу q дополнительную переменную v_q, которой придадим начальное значение j_q. Во время волны эти переменные переприсваиваются следующим образом. Всякий раз, когда процесс q посылает сообщение, текущее значение v_q включается в сообщение. Всякий раз, когда процесс q получает сообщение со значением v, v_q присваивается значение $v_q \wedge v$. Когда в процессе p происходит событие decide, текущее значение v_p заносится в out_p.

Теперь нужно показать, что в результат заносится правильное значение. Обозначим правильный ответ через J, то есть $J = \inf \{j_q : q \in P\}$. Для события a в процессе q обозначим через $v^{(a)}$ значение v_q сразу после выполнения a. Так как начальное значение v_q равно j_q, и в течение волны оно только уменьшается, неравенство $v_{(a)} \le j_q$ сохраняется для каждого события a в q. Из присваивания v следует, что для событий a и b, a p b $\Rightarrow v^{(a)} \ge v^{(b)}$. Кроме того, так как v всегда вычисляется как инфимум двух уже существующих величин, неравенство $J \le v$ выполняется для всех величин в течение волны.

Таким образом, если d – событие decide в p, значение $v^{(d)}$ удовлетворяет неравенству $J \le v^{(d)}$ и, так как событию d предшествует хотя бы одно событие в каждом процессе q, $v^{(d)} \le j_q$ для всех q.

Отсюда следует, что $J = v^{(d)}$.

Построенный INF-алгоритм обладает всеми свойствами алгоритма, рассмотренного в первом учебном вопросе, кроме битовой сложности, поскольку к каждому сообщению ранее рассмторенного алгоритма добавляется элемент X. Понятие функции инфимума может показаться довольно абстрактным, но фактически многие функции могут быть выражены через функцию инфимума.

Аксиома 1. (Теорема об инфимуме). Если * – бинарный оператор на множестве X, причем он:

(1) коммутативен, то есть a * b = b * a,

(2) ассоциативен, то есть (a * b) * c = a * (b * c),

(3) идемпотентен, то есть a * a = a,

то существует отношение частичного порядка ≤ на X такое, что * – функция инфимума.

Среди операторов, удовлетворяющих этим трем критериям – логическая конъюнкция и дизъюнкция, минимум и максимум целых чисел, наибольший общий делитель и наименьшее общее кратное целых чисел, пересечение и объединение множеств.

Заключение 1. &, ∨, min, max, НОД, НОК, ∩ и ∪ величин, локальных по отношению к процессам, могут быть вычислены за одну волну.

Выводы

В ходе лекции рассмотрены следующие вопросы:

- определение волновых алгоритмов;

- элементарные результаты о волновых алгоритмах;

- распространение информации с обратной связью;

- синхронизация;

- вычисление функций инфинума.

Задание на самостоятельную работу

1. Конспект лекций.

Вид и тема следующего занятия

Практическое занятие №10. Маршрутизация (ч. 2)