Протокол консенсуса в асинхронных сетях с неисправностями

М.О. Кривошеев, Ю.Н. Логвинов

Поиск неисправностей в сетевых вычислительных средах, надежность функционирования программного и аппаратного обеспечения является актуальной задачей, для решения которой используются формальные математические модели, позволяющие на этапах проектирования, тестирования и эксплуатации обнаружить и локализовать статические и динамические неисправности, как в структуре сетевой среды, так и в каналах связи [1, 2]. Эффективные модели вычислительных процессов дают возможность обнаружить не только отказ компонент вычислительной сети, но и сетевые атаки, а также аномальную активность в каналах связи [3].

Достижение консенсуса среди распределенных процессов является одной из наиболее фундаментальных задач в распределенных вычислениях и находится в центре многих алгоритмов для распределенной обработки данных, распределенного управления файлами и отказоустойчивости распределенных приложений. Для достижения надежности в распределенных системах необходимы протоколы, которые позволяют системе, как единому целому, функционировать, невзирая на отказ ограниченного числа ее компонентов.

Ключевым моментом здесь является не то, о *чем* договариваются процессы, а тот факт, что они все должны прийти к *одинаковому выводу*. Решение задачи консенсуса сильно зависит от того, какие делаются предположения о модели вычислений и видах неисправностей, которым подвержена распределенная система. В этой статье мы будет полагать фиксированное число процессов N. Считается, что протокол t-устойчивый, если он корректно работает при отказе не более чем t процессов до или в течении своего выполнения [4].

Мы рассматриваем два вида неисправностей процессов. *Отказ* (crash) происходит, когда процесс прекращает всю свою деятельность. Вплоть до

момента отказа, процесс работает корректно, и после он становится полностью неактивным. Протокол, который может выдержать вплоть до *t* отказавших процессов, обладает *t-устойчивостью к отказам*. В данной работе не рассматривается проблема восстановления неисправных процессов и повторным введением их обратно в систему.

Более «злокачественным» или разрушительным видом неисправности является, так называемая, Византийская неисправность, при которой не делается никаких предположений о поведении неисправного процесса. В частности, он может отправлять сообщения в неподходящее время или делать конфликтующие заявления, не выходить на связь в течение некоторого времени и затем возобновлять работу и так далее. Протокол, который может выдержать до *t* процессов, проявляющих Византийскую неисправность, обладает *t-устойчивостью к Византийским неисправностям* и иногда называется *Византийским* протоколом [5].

В работе считается, что система сообщений полностью надежна и что только процессы подвержены неисправностям. Мы также полагаем, что любой процесс может абсолютно надежно определить отправителя любого принимаемого сообщения и что любое сообщение доставляется без изменений и ошибок. Коммуникационная сеть является полностью связанным графом.

В реальной системе коммуникационные каналы, также как и процессы, подвержены неисправностям. Неисправность канала может быть представлена, как неисправность процесса на одном из его концов, значит, протокол обладающий *t*-устойчивостью, автоматически допускает вплоть до *t* неисправностей процессов и каналов.

Для разработки распределенных алгоритмов широко используются две модели распределенных коммуникационных систем — синхронная и асинхронная. Синхронная модель полагает наличие общую систему синхронизации для всех узлов и ограниченную задержку доставки сообщений. Время может быть представлено разделенным на слоты.

Передача сообщений всегда происходит в фиксированной позиции в слоте. Если некий узел передает сообщение в течение слота *i*, то это сообщение гарантировано будет принято (и обработано) всеми соседями к началу слота *i* + 1. Распределенные алгоритмы, которые работают в раундах или циклах, естественным образом удовлетворяют этой модели. Синхронная природа передачи сообщений гарантирует, что вся информация предыдущего цикла доступна некому узлу до отправки сообщений в следующих циклах.

Асинхронная модель полагает отсутствие общей возможности синхронизации и обычно полагает конечную, но не ограниченную задержку доставки сообщения. Следовательно, многие распределенные алгоритмы, разработанные для асинхронных сетей значительно менее эффективны в терминах времени и сложности сообщений, чем те, что разработаны для синхронных сетей [6].

Синхронная модель является более сильной, чем модель асинхронная. Между этими двумя моделями существует модель *асинхронной сети с ограниченной задержкой доставки сообщений* (Asynchronous Bounded Delay — ABD). Эта модель слабее, чем полностью синхронная, но сильнее, чем полностью асинхронная модель [7]. Предлагаемый нами алгоритм выполняет обработку сообщений в последовательности раундов, что позволяет отнести его к сетям типа ABD.

В предлагаемой модели вводится допущение о наличие в системе комбинации неисправностей смешанного типа – отказавших и процессов проявляющих Византийскую неисправность. Алгоритм консенсуса Брачи-Туэга *t*-устойчивый к Византийским процессам существует для предельной границы $t < N/3$ и *t*-устойчивый алгоритм консенсуса для отказавших процессов с предельной границей $t < N/2$ [8]. В предлагаемом ниже алгоритме предельное значение *t*-устойчивости находится в диапазоне (N/3 : N/2).

Любой процесс может принять участие в голосовании в каждом раунде, и решение принимается только по достижению достаточного количества

голосов за одинаковое значение. Алгоритм предполагает, что Византийский процесс всегда участвует в процедуре голосования, посылая, может быть, конфликтующие значения разным соседним процессам. Это более слабое определение Византийской неисправности, не принимающее во внимание «злонамеренную волю». Процесс, не присылающий ни одного сообщения в течение всего раунда голосования, считается отказавшим.

```
var value_p          : (0,1)    initial x_p ;
    round_p          : integer  initial 0 ;
    msgs_p [0..1]    : integer  initial 0 ;
    msgc_p           : integer  initial 0 ; (* счет сообщений от процесса *)
    crashc           : integer  initial 0 ; (* счет числа crash-процессов *)
    echos_p [P, 0..1] : integer  initial 0 ;
repeat forall v, q do begin msgs_p[v] := 0 ; echos_p[q, v] := 0 end ;
    broadcast { vote, initial, p, value_p, round_p } ;
    (* Теперь принимаем N – t голосов для текущего раунда *)
    while msgs_p[0] + msgs_p[1] < N – t do
        begin receive { vote, t, r, v, rn } from q ;
        (* увеличиваем счетчик числа полученных сообщений от q *)
            msgc_q := msgc_q + 1
            if { vote, t, r, *, rn } уже было принято от q
                then skip (* повтор, значит q византиец*)
            else if t = initial and q ≠ r
                then skip (* q лжет, значит он - византиец*)
            else if rn > round_p
                then (* Обработать позже, в следующем раунде *)
                    send { vote, t, r, *, rn } to p
            else (* Обработать или ретранслировать сообщение *)
                case t of
                    initial : broadcast { vote, echo, r, v, rn } ;
                    echo   : if rn = round_p then
                        begin echos_p [r, v] := echos_p [r, v] + 1 ;
                        if echos_p [r, v] = ⌊(N + t)/2⌋ + 1
```

then $msgs_p\,[v] := msgs_p\,[v] + 1$

end

else skip (* старое сообщение *)

esac

end;

(* накопление числа crash-процессов за текущий раунд *)

repeat forall q **do**

begin if $msgc_q = 0$ **then** $crashc := crashc + 1$; $msgc_q := 0$; **end**;

(* Выбор значения для следующего раунда *)

if $msgs_p[0] > msgs_p[1]$ **then** $value_p := 0$ **else** $value_p := 1$;

if $msgs_p[0] > 2(N - 2crashc)/3$ **then** $y_p := value_p$;

$round_p := round_p + 1$

until *false*

Представленный алгоритм выявляет в каждом раунде количество отказавших процессов равных *crachc*. Тогда процесс *p* будет учитывать бюллетень голосования после получения {**vote**, **echo**, *r*, *v*, *k*} от более чем 2(*N* — 2*crashc*)/3 процессов. Следовательно, наш алгоритм консенсуса устойчив к суммарному числу Византийских и отказавших процессов большему, чем предельный случай для чисто Византийских процессов в алгоритме консенсуса Брачи-Туэга [9, 10].

Литература:

1. Н.А. Бурякова А.В. Чернов. Классификация частично формализованных и формальных моделей и методов верификации программного обеспечения [Электронный ресурс] // «Инженерный вестник Дона», 2010, №4. – Режим доступа: http://ivdon.ru/magazine/archive/n4y2010/259 (доступ свободный) – Загл. с экрана. – Яз. Рус.
2. Ильичева, О.А. Технология логического моделирования и анализа сложных систем [Электронный ресурс] // «Инженерный вестник Дона», 2012, №4 (часть 2). – Режим доступа:

http://ivdon.ru/magazine/archive/n4p2y2012/1234 (доступ свободный) – Загл. с экрана. – Яз. Рус.

3. И.М.Ажмухамедов А.Н. Марьенков. Поиск и оценка аномалий сетевого трафика на основе циклического анализа [Электронный ресурс] // «Инженерный вестник Дона», 2012, №2. – Режим доступа: http://ivdon.ru/magazine/archive/n2y2012/742 (доступ свободный) – Загл. с экрана. – Яз. Рус.
4. M.J. Fischer, N.A. Lynch, M.S. Paterson. Impossibility of Distributed Consensus with One Faulty Process – Journal of the Association for Computing Machinery, Vol. 32, No. 2, April 1985, pp. 374-382.
5. M.J. Fischer. The Consensus Problem in Unreliable Distributed Systems (A Brief Survey) – Foundations of Computation Theory, volume 158 of Lecture Notes in Computer Science, pp 127–140, Spring-Verlag, 1983. Reissued February 2000
6. Chou, C. T., Cidon, I., Gopal, I. S., and Zaks, S. Synchronizing asynchronous bounded delay networks. IEEE Trans. Commun. 38, 1990, pp 144-147.
7. G. Tel, E. Korach, S. Zaks. Optimal Synchronization of ABD Networks – Department of Computer Science Utrecht University Technical Report RUU-CS-88-23, May 1998, The Netherlands.
8. G. Bracha, S. Toueg. Asynchronous Consensus and Broadcast Protocols – Journal of the Association for Computing Machinery, Vol. 32, No. 4, October 1985, pp. 824-840.
9. T. Herman, G. Tel. Advanced Distributed Algorithms – Department of Computer Science Utrecht University Technical Report RUU-CS-93-37, Novemer 1993, The Netherlands.
10. Тель Ж. Введение в распределенные алгоритмы [Текст]: Монография / Жерар Тель; перевод с англ. В.А. Захарова – М.: МЦНМО, 2009 г. – 616 с.: ил.; ISBN 978-5-94057-515-3