УДК 681.3

# Метод уменьшения глубины каскадных откатов при распределенном оптимистическом моделировании алгоритмов маршрутизации

Ладыженский Ю.В., Мирецкий А.В.
Донецкий национальный технический университет
ly@cs.dgtu.donetsk.ua, a.miretsky@gmail.com

***Abstract***

***Ladyzhensky Y.V., Miretsky A.V. Method for decreasing depth of cascade rollbacks in distributed optimistic simulation of routing algorithms.*** *A new method for decreasing depth of cascade rollbacks in distributed optimistic simulation of routing algorithms is described in this article. Cascade rollbacks are moved from the level of logical processes to the level of their components in this method. This allows to decrease time depth, space depth and execution time of rollbacks.*

## *Введение*

В современном мире компьютерные сети широко используются во многих сферах нашей деятельности: в бизнесе, в промышленности, в науке и т.д. Это обусловливает постоянный рост сложности сетей, количества сетевых сервисов и нагрузки на сеть. Важные требования к сетям: высокая пропускная способность, устойчивость к отказам, гарантированная доставка информации. Эти показатели во многом зависят от выбора маршрутов потоков данных в сети. Правильно организованная маршрутизация позволяет значительно повысить эти показатели.

Важным этапом при проектировании сетей и разработке алгоритмов маршрутизации является моделирование. Компьютерные сети – это сложный объект, поэтому чаще всего применяется имитационное моделирование. Для представления компьютерных сетей используются дискретные модели, что позволяет применять метод событийного продвижения времени. Известно, что выполнение дискретно-событийного моделирования больших сетей на одном процессоре обычно требует много времени. Ускорить этот процесс можно, если выполнить декомпозицию модели на логические процессы и распределить эти процессы между процессорами для параллельного запуска, но при этом появляется необходимость в синхронизации моделирующих логических процессов.

Существующие алгоритмы синхронизации слабо учитывают неоднородности потоков сообщений между компонентами моделируемых систем. Это приводит к выполнению избыточных откатов состояний. Исключение таких откатов позволяет ускорить моделирование. Следовательно, использование свойств потоков сообщений является перспективным направлением повышения эффективности распределенного моделирования.

Первые алгоритмы синхронизации логических процессов при распределённом моделировании появились в начале 80-х годов XX века. Консервативный протокол синхронизации впервые описали в своих работах Chandy, Misra [1, 2, 3, 4] и Bryant [5], а оптимистический протокол синхронизации (Time Warp) впервые описал Jefferson [6].

Эти методы имеют ряд недостатков, негативно влияющих на скорость моделирования [7, 8]. Консервативный алгоритм всегда синхронизирует логические процессы, даже если те временно не взаимодействуют друг с другом. Используемые при этом нуль-сообщения нагружают вычислительную сеть лишней работой. Оптимистический алгоритм позволяет логическим процессам выполнять моделирование параллельно, независимо друг от друга, без синхронизации до тех пор пока один из процессов не получит отставшее сообщение. Выполняемый при этом откат возвращает логический процесс в его модельное прошлое. Постоянное выполнение каскадных откатов, при которых в прошлое возвращается более одного процесса, может значительно замедлить время моделирования из-за затрат на возврат состояния, отмену выполненных и запланированных событий и передачу по вычислительной сети множества анти-сообщений.

Цель работы состоит в разработке новых методов уменьшения стоимости (глубины и времени выполнения) каскадных откатов, учитывающих особенности обмена сообщениями между компонентами моделируемых систем.

## *Моделирование маршрутизации*

Моделирование компьютерных сетей и маршрутизации в них позволяет получить оценки основных характеристик сетей: пропускная способность, уровень потерь пакетов, загрузка

узлов сети. Часто возникают задачи проверки сетей или маршрутизаторов (алгоритмов маршрутизации) на устойчивость к отказам в обслуживании. При этом имитируются различные сбои в работе узлов, обрывы линий связи, перегрузка отдельных участков сети [9, 10, 11]. В этих условиях становятся видны недостатки алгоритмов маршрутизации и выбираемых ими маршрутов.

При распределенном моделировании описанных событий могут возникать ситуации, в которых узлы сети, находящиеся в рамках одного логического процесса, перестают взаимодействовать друг с другом или наоборот начинают интенсивно взаимодействовать, т.е. возникают неоднородности потоков событий в сетевой модели.

В оптимистическом алгоритме, при получении логическим процессом отставшего сообщения, выполняется откат состояния всего логического процесса. Однако, как показано выше, компоненты логического процесса могли не взаимодействовать в откатываемом интервале времени, вследствие временного отсутствия линий связи между ними. Следовательно, нет необходимости выполнять полный откат всего логического процесса. Этот подход позволяет уменьшить время выполнения отката и ускорить распределенное оптимистическое моделирование.

### *Метод локальных каскадных откатов*

В работах [12, 13, 14] предложен метод локальных каскадных откатов (ЛКО), в котором откаты впервые перенесены с уровня логических процессов на уровень их компонентов. Такая детализация позволяет уменьшить глубину отката в пространстве и во времени, т.к. учитывает обмен сообщениями не только между логическими процессами, но и между их компонентами, т.е. внутри логических процессов.

Модель сети представляется в виде графа $G=(C, L)$, где $C$ – множество компонентов, $L$ – множество линий связи между компонентами. В методе ЛКО описание события как вектора расширено двумя дополнительными измерениями: $e=(\sigma,\tau,c,h)$, где $\sigma$ – момент модельного времени, когда было порождено сообщение; $\tau$ – момент времени, когда событие должно произойти; $c$ – компонент, породивший сообщение; $h$ – компонент, которому предназначено сообщение.

У каждого компонента есть свое собственное локальное виртуальное время $LVT^{(c)}$, где $c \backsim C$ – компонент. Локальное виртуальное время логического процесса равно временной метке последнего обработанного события.

Все структуры данных, используемые в оптимистическом алгоритме Time Warp [7] распределены между компонентами:

– $SE$ – общий список событий, предназначенный для синхронизации компонентов внутри логического процесса;
– $SE^{(c)}$ – список событий компонента $c$, предназначен для ускорения доступа к событиям, запланированным для выполнения в $c$;
– $HE^{(c)}$ – список событий, обработанных компонентом $c$;
– $AM(c, c_1)$ – список анти-сообщений для сообщений отправленных из $c$ в смежный с ним $c_1$.

На рисунке 1 представлено концептуальное описание алгоритма отката компонента $c$ до отметки $\tau$.

**Если** $\tau<LVT^{(c)}$ **Тогда**
  ***1.*** Откат компонента c до отметки τ;
  $LVT^{(c)} := \tau$;
  ***2.*** **Для всех** смежных компонентов c`
    **Если** c` принадлежит другому логическому процессу **Тогда**
      Отправить все антисообщения для него по сети;
    **Иначе**
      **Если** есть анти-сообщение в AM(c, c`) **Тогда**
        Выполнить рекурсивный откат c` до минимальной метки анти-сообщения;
        Удалить антисообщения;
      **КонецЕсли**
    **КонецЕсли**
  **КонецДля**
  **3.** Удалить все события e` из HE для которых $\sigma`>LVT^{(c`)}$;
  Переместить события e` из HE в SE для которых $\sigma`<=LVT^{(c`)}$ и $\tau`>\tau$;
  **4.** Удалить все события e` из SE для которых $\sigma`>LVT^{(c`)}$;
**КонецЕсли**

Рисунок 1 – Алгоритм ЛКО

На этапе 1 выполняется откат состояния компонента в его прошлое.

На этапе 2 рекурсивно вызывается процедура отката для всех компонентов, связанных с текущим компонентом.

На этапе 3 выполняется отмена обработанных компонентом сообщений. Сообщения, которые были созданы до того момента времени, до которого откатилось состояние компонента их породившего, заново помещаются в список запланированных событий.

На этапе 4 выполняется отмена запланированных событий, которые были созданы позднее того момента времени, до которого откатилось состояние компонента их породившего.

На рисунке 2 представлен пример модели сети, распределенной между 5-ю процессорами. Процесс $LP_3$ присылает в процесс $LP_1$ событие, которое становится отставшим. В процессе $LP_1$ начинается откат состояния и отмена событий.

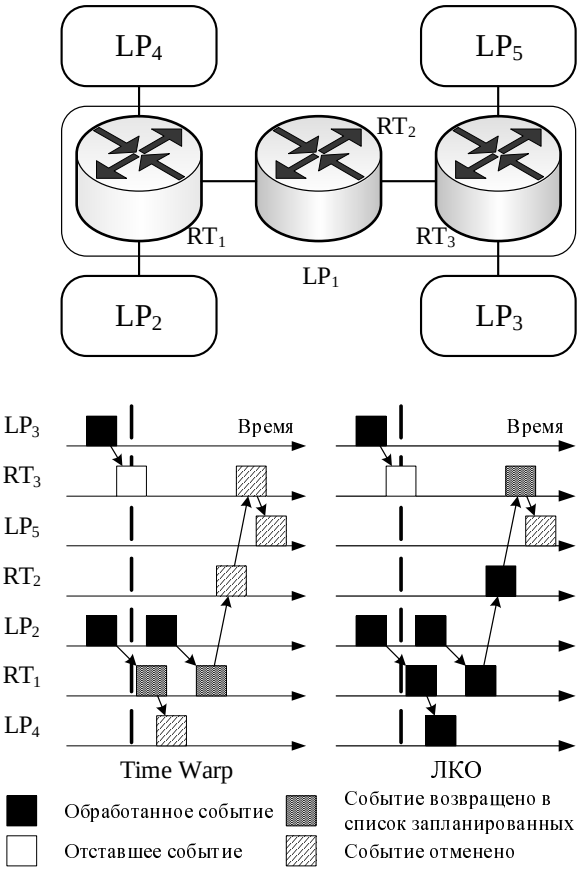


Рисунок 2 – Сравнение результатов процедуры отката в алгоритмах Time Warp и ЛКО.

Алгоритм Time Warp выполняет откат всего логического процесса, поэтому вызывает отмену событий, отправленных в $LP_4$ и $LP_5$, что в свою очередь может привести к каскадному откату логических и этих процессов.

Алгоритм ЛКО выполняет отмену только событий компонента $RT_3$, что вызывает отмену события, отправленного в $LP_5$. В рассмотренном примере, все остальные события не отменяются алгоритмом ЛКО. Откат процесса $LP_4$ не происходит.

Алгоритм ЛКО позволяет уменьшить временную глубину отката, а, следовательно, и количество отменяемых событий, и количество анти-сообщений, отправляемых по сети, и количество каскадных откатов удаленных логических процессов.

### *Модель процедуры отката метода ЛКО*

Модель метода ЛКО построена в среде MathCAD. Она позволяет рассчитать вероятности отката компонентов и среднее количество компонентов, принимающих участие в откате.

В модели вероятность отката компонента $c$ вычисляется по формуле:

$$P_c = \sum_{i=1}^{Nc} \left[ \left( \prod_{j=1}^{i-1} \left(1 - P_{c_j,c}(LVT^{(c_j)} - \tau_{st}^{(c_j)}) \cdot P_{c_j} \right) \right) \cdot \right.$$
$$\left. \cdot P_{c_i,c}(LVT^{(c_i)} - \tau_{st}^{(c_i)}) \cdot P_{c_j} \right]$$

где $LVT^{(c)}$ - локальное виртуальное время компонента $c$; $\tau_{st}^{(c)}$ - временная метка, до которой выполнен откат в $c$; $P_{c_j,c}(LVT^{(c_j)} - \tau_{st}^{(c_j)})$ - вероятность того, что в интервале $\left[LVT^{(c_j)} - \tau_{st}^{(c_j)}, LVT^{(c_j)}\right]$ у компонента $c_j$ будут анти-сообщения для компонента $c$; $Nc$ – количество компонентов, смежных с $c$, из которых может прийти каскадный откат.

На основе рассчитанных вероятностей можно получить количество компонентов, принимающих участие в откате: $K = \sum_{\forall c} P_c \cdot$

Моделирование выполнено на замкнутой сети с очередями, т.к. такие сети можно рассматривать как упрощенные модели компьютерных сетей. В модели использована сеть, состоящая из 5 рядов серверов FCFS по 3 в каждом ряду. Эти сервера имеют бесконечные очереди. Передача заявок между рядами FCFS осуществляется с помощью переключателей Switch с заданными вероятностями переключения. Сеть разрезана на два логических процесса. Первый содержит 3 ряда, а второй – 2 ряда серверов. На рисунке 3 показан логический процесс, содержащий 3 ряда. В модели рассматривается случай, когда из второго процесса $LP_2$ приходит отставшее сообщение в первый процесс в сервер FCFS 0.

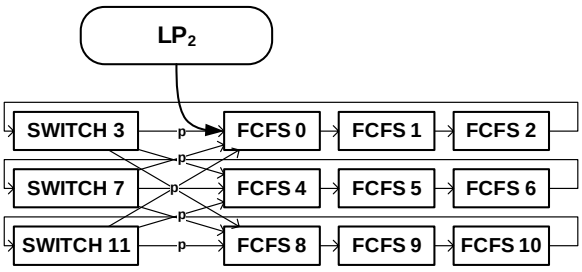


Рисунок 3 – Замкнутая сеть с очередями

На рисунке 3 числами обозначены номера компонентов. Величина $p$ – это вероятность передачи пакета переключателем Switch в одно из устройств FCFS, p=const=1/N, где N – количество рядов серверов FCFS.

На рисунке 4 показано среднее количество компонентов, принимающих участие в откате, в зависимости от глубины отката компонента $c_0$ (FCFS 0), который получает отставшее сообщение из логического процесса $LP_1$. Параметры сети с очередями, использовавшиеся при вычислении,

такие: время задержки пакетов в Switch – 0.05; время обработки пакетов серверами FCFS равномерно распределено в интервале [0.05, 0.15]. Вероятность наличия антисообщений у сервера FCFS для следующего FCFS или Switch выбрана равной:

$$P_{FCFS}(\Delta\tau) = \begin{cases} 0, \Delta\tau < 0.05 \\ \dfrac{\Delta\tau - 0.05}{0.15 - 0.05}, \Delta\tau \in [0.05, 0.15] \\ 1, \Delta\tau > 0.15 \end{cases}$$

Для переключателя вероятность наличия антисообщений равна $P_{Switch}=P_{FCFS}\cdot p$, где $p=0.2$.

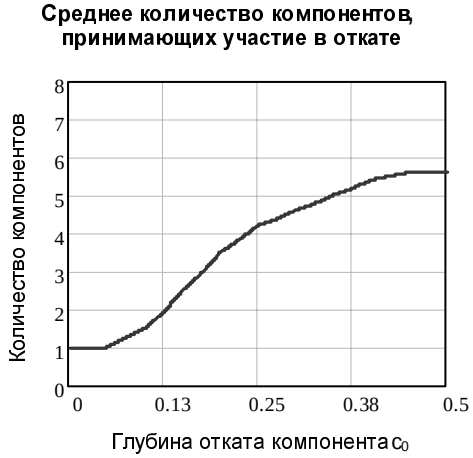


Рисунок 4– Зависимость количества компонентов, принимающих участие в откате от глубины отката компонента $c_0$.

По графику видно, что среднее количество компонентов, принимающих участие в откате, возрастает с ростом глубины отката, однако после значения 0.45 рост прекращается и достигает отметки 5.6, т.е. максимально возможное число компонентов (12) не достигается. Это связано с тем, что каскадный откат может быть передан в те же компоненты, что уже возвращены в прошлое, либо в компоненты логического процесса, инициировавшего откат. В этих приведенных случаях процесс выполнения каскадного отката останавливается, и количество затронутых им компонентов действительно не достигнет максимума.

Важно отметить, что алгоритм Time Warp в рассматриваемом примере всегда откатывал бы все компоненты.

### *Экспериментальное исследование ЛКО*

Разработана система для моделирования маршрутизации в компьютерных сетях AVMSim, позволяющая использовать любые методы синхронизации логических процессов и исследовать их свойства.

В системе моделирования AVMSim выполнено исследование поведения алгоритма ЛКО при моделировании замкнутых сетей с очередями разного размера. Для задания размера сети используется обозначение X×Y, где X – количество переключателей в сети (фактически, это количество строк в сети), а Y – количество компонентов FCFS в каждой строке. Количество компонентов в модели можно вычислить по формуле X·(Y+1). Все модели разрезались на два логических процесса. Размеры сетей и соответствующие им минимальные, средние и максимальные глубины отката показаны в таблице 1.

Таблица 1. Результаты экспериментального исследования

| Размер сети X×Y | Кол-во компо-нентов | Мин. $C_{min}$ | Среднее $C_{avg}$ | Макс. $C_{max}$ | Коэф. умень-шения, % |
|---|---|---|---|---|---|
| 12x12 | 156 | 1 | 2,5 | 21 | 73,1 |
| 12x24 | 300 | 1 | 2,4 | 16 | 89,3 |
| 24x12 | 312 | 1 | 2,4 | 25 | 84,0 |
| 12x36 | 444 | 1 | 2,3 | 13 | 94,1 |
| 36x12 | 468 | 1 | 2,1 | 31 | 86,8 |
| 24x24 | 600 | 1 | 2,3 | 34 | 88,7 |
| 24x36 | 888 | 1 | 2,3 | 34 | 92,3 |
| 36x24 | 900 | 1 | 2,3 | 35 | 92,2 |
| 36x36 | 1332 | 1 | 2,2 | 42 | 93,7 |

Видно, что при увеличении размера модели, максимальное количество откатываемых компонентов возрастает. Причем, для малых сетей характерны скачки, а для больших сетей, закономерность имеет более выраженный характер.

Интересным результатом является тот факт, что при увеличении размера модели, среднее количество откатываемых компонентов практически не меняется. Это связано с тем, что в промоделированных сетях с большей вероятностью происходят неглубокие по времени откаты, т.е. компоненты чаще получают отставшие сообщения с небольшим временем отставания. А при неглубоких откатах, количество откатываемых компонентов невелико.

Рассчитаем коэффициент уменьшения глубины отката при применении метода ЛКО в сравнении с алгоритмом Time Warp. Количество компонентов в каждом логическом процессе равно X·(Y+1)/2. В алгоритме Time Warp они все принимают участие в откате. Тогда коэффициент уменьшения равен

$$K_{кол\text{-}во\ комп.},\% = 1\text{-}2\cdot C_{max}/(X\cdot(Y+1))\cdot 100,$$

где $C_{max}$ – максимальное количество компонентов, принявших участие в откате в алгоритме ЛКО. В таблице 1 представлены коэффициенты уменьшения для разных сетей.

Алгоритм ЛКО позволил уменьшить максимальную глубину отката на ~73-94%, в зависимости от размера сети.

### *Заключение*

Разработан метод локальных каскадных откатов (ЛКО), учитывающий неоднородности потоков событий внутри логических процессов. Он уменьшает глубину отката в модельном времени и в пространстве моделирующих компонентов, за счет чего уменьшаются время на возврат состояния логического процесса в прошлое, количество отправляемых по сети антисообщений и вероятность возникновения вторичных откатов логических процессов.

Методом моделирования показано, что алгоритм ЛКО позволяет выполнять откат не всех, а только части компонентов логического процесса.

Выполнено экспериментальное исследование работы алгоритма ЛКО при моделировании замкнутой сети с очередями, которое показало, что алгоритм ЛКО позволяет уменьшить максимальную глубину отката в пространстве моделирующих компонентов на ~73-94% в зависимости от размера сети.

### *Литература*

1. K.M. Chandy, J. Misra “Asynchronous Distributed Simulation via a Sequence of Parallel Computations” in Comms. ACM, Vol. 24, No.11, pp. 198-206, 1981.
2. K.M. Chandy, J. Misra, L. Haas “Distributed Deadlock Detection” in ACM Trans. on Computer Systems, Vol. 1, No.2, pp. 144-156, May 1983.
3. Chandy K.M., Misra J. Distributed simulation: A case study in Design and Verification of Distributed Programs / IEEE Transactions on Software Engineering, SE-5(5). – September, 1979. – pp. 440-452.
4. Tay S.C., Teo Y.M. Performance optimization of throttled time-warp simulation // Proceedings of the 34th Annual Simulation Symposium (SS’01). – National University of Singapore, Department of computer science. – Singapore, 2001.
5. Bryant R. E. A Switch-Level Model and Simulator for MOS Digital Systems, IEEE Transactions on Computers, c-33(2):166-177, February 1984.
6. D. Jefferson “Virtual Time” in ACM Trans. Prog. Lang. and Sys. Vol.7 No.3, pp. 404-425, July 1985.
7. Fujimoto R.M. Parallel and distributed simulation systems, Wiley Interscience, 2000
8. Samir R. Das. Adaptive protocols for parallel discrete event simulation, Proceedings of the 28th conference on Winter simulation. 1996. pp. 186-193.
9. Ладыженский Ю.В., Мирецкий А.В., Мирецкая В.А. Моделирование мультиагентных алгоритмов маршрутизации сообщений в компьютерных сетях// Моделирование и компьютерная графика: Материалы 1-й международной научно-технической конференции, г Донецк, 04-07 октября 2005 г. — Донецк, ДонНТУ, Министерство образования и науки Украины, 2005. — 285 с., с. 122-126.
10. Ладыженский Ю.В., Мирецкий А.В. Моделирование алгоритмов динамической маршрутизации // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей Міжнародної науково-практичної конференції м. Запоріжжя, 13-15 квітня 2006 року / Під заг. Ред. Д.М. Пізи. – Запоріжжя: ЗНТУ. 2006. С. 165-167.
11. Ладыженский Ю.В., Мирецкий А.В. Программная система для моделирования алгоритмов динамической маршрутизации // Інтернет-освіта-наука-2006, п’ята міжнародна конференція ІОН-2006, 10-14 жовтня, 2006. Збірник матеріалів конференції. Том 2. – Вінниця: УНІВЕРСУМ-Вінниця. 2006. С. 372-374.
12. Ладыженский Ю.В., Мирецкий А.В. Оптимистический алгоритм синхронизации с внутрикомпонентными откатами для распределенного моделирования // Моделирование и компьютерная графика: Материалы второй международной научно-технической конференции, г. Донецк, 10-12 октября 2007 года – Донецк, ДонНТУ, Министерство образования и науки Украины, 2007. – 358с. // стр.187 – 192
13. Ладыженский Ю.В., Мирецкий А.В. Метод локальных каскадных откатов для распределенного моделирования// Сборник трудов конференции МОДЕЛИРОВАНИЕ-2008 Том 2. – Киев.: ИПМЭ им. Г.Е. Пухова. – с. 594-597
14. Ладыженский Ю.В., Мирецкий А.В. Метод локальных каскадных откатов и организация списков событий для распределенного моделирования компьютерных систем // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2008). Выпуск 9(132) – Донецк: ДонНТУ. – 2008. – 316с.

*Поступила в редколлегию 10.03.2009*