Ссылки

1. Нейман Дж. фон. Теория самовоспроизводящихся автоматов: пер. с англ. М.: Мир, 1971.

В оригинале: von Neumann J. Theory of Self-Reproducing Automata: ed. and compl. by A. Burks. University of Illinois Press, 1966.

2. Тоффоли Т., Марголус Н. Машины клеточных автоматов. М.: Мир. 1991.

УДК 004

Моделирование и анализ свойств протокола для p2p-сетей при помощи раскрашенных сетей Петри

С. А. Максимов, Д. Ю. Чалый

Ярославский государственный университет им. П. Г. Демидова
E-mail:maksimov.s.and@gmail.com, chaly@uniyar.ac.ru

В статье объектом исследования является протокол Chord, который моделируется при помощи раскрашенных сетей Петри и среды моделирования CPN Tools. Целью работы является проведение анализа семантических свойств протокола при помощи методов, разработанных для этого формализма.

Ключевые слова: протокол Chord, раскрашенные сети Петри, CPN Tools.

Введение

Исследование свойств протоколов передачи данных является важной и актуальной задачей. В работе смоделирован и проанализирован протокол Chord [7] — протокол получения данных из распределенной хэш-таблицы (distributed hash table, DHT), задающий алгоритм поиска узла, на котором содержится искомая порция данных. Chord служит для организации эффективного доступа к узлам распределенной системы.

Моделирование является одним из основных методов исследования протоколов в коммутируемых сетях. В настоящее время разработано большое количество инструментальных средств, ориентированных на анализ моделей, представленных с помощь сетей Петри. Существует много современных инструментов, работающих с сетями Петри, примером таких средств являются ARP tool, CoopnBuilder, CPN Tools, HISIm, Petri .NET Simulator, Petri Net Toolbo1, Petruchio, Snoop2. Среди перечисленных средств большими возможностями выделяется среда моделирования CPN Tools [6], которая и была выбрана для использования. В рамках работы была поставлена задача построить модель протокола Chord в системе CPN Tools и проанализировать работу модели на примере нескольких топологий коммутируемых сетей.

Принципы работы протокола Chord

Протокол Chord [7] — алгоритм для соединения равноправных узлов в локальной сети, использующий распределенную хэш-таблицу (DHT). Принцип DHT служит для организации эффективного доступа к большому числу блоков данных, разбросанных по нескольким серверам и хранящихся на них примерно в одном виде. Каждый из таких блоков данных должен обладать некоторым уникальным ключом.

Предоставляемые DHT методы работы с данными:

- get (key)
- delete (key)
- insert (key, value)

С помощью хеш-функции (HASH()) множество ключей блоков данных отображается в кольцо Chord Ring (последний и первый элемент этого упорядоченного множества хешей считаются связанными). Узлам (серверам) присваиваются ключи (nodekey), принадлежащие множеству Chord Ring, — узлы «распределяются» по кольцу.

Моделирование архитектуры протокола

В данной статье разобрана не вся модель, а только основные ее части. Модель построена с применением иерархии. Самой общей из них является «Top». На рис. 1 изображено разбиение задачи на три подзадачи: добавление или инициализация (Add), удаление (Delete), обновление finger table (RFT) и тестирование

(Test). Finger table каждого пира представляет собой список, элементы которого — это номера узлов, связанных с данным пиром. Пир характеризуется идентификатором и finger table. В позиции «Peers» осуществляется хранение информации о пирах.

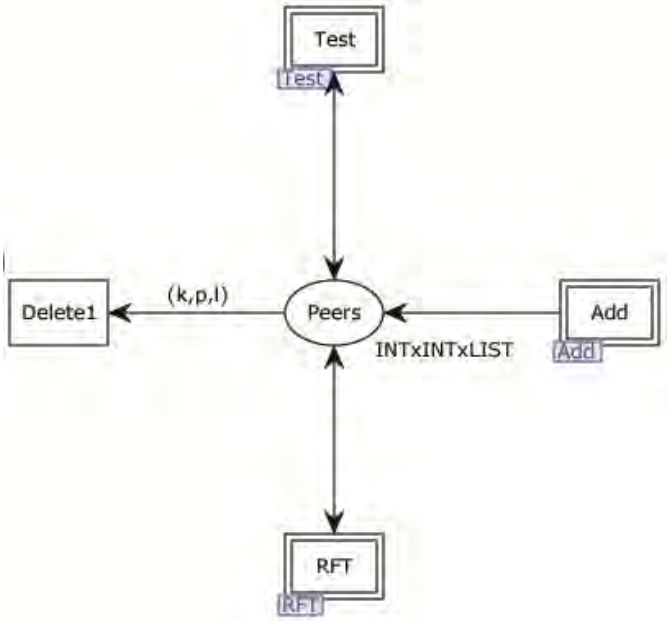

Рис. 1. Моделирование состояний системы

Моделирование обновления служебных таблиц

Второй самой важной частью модели является граф (рис. 2), в котором происходит заполнение (или обновление) finger table.

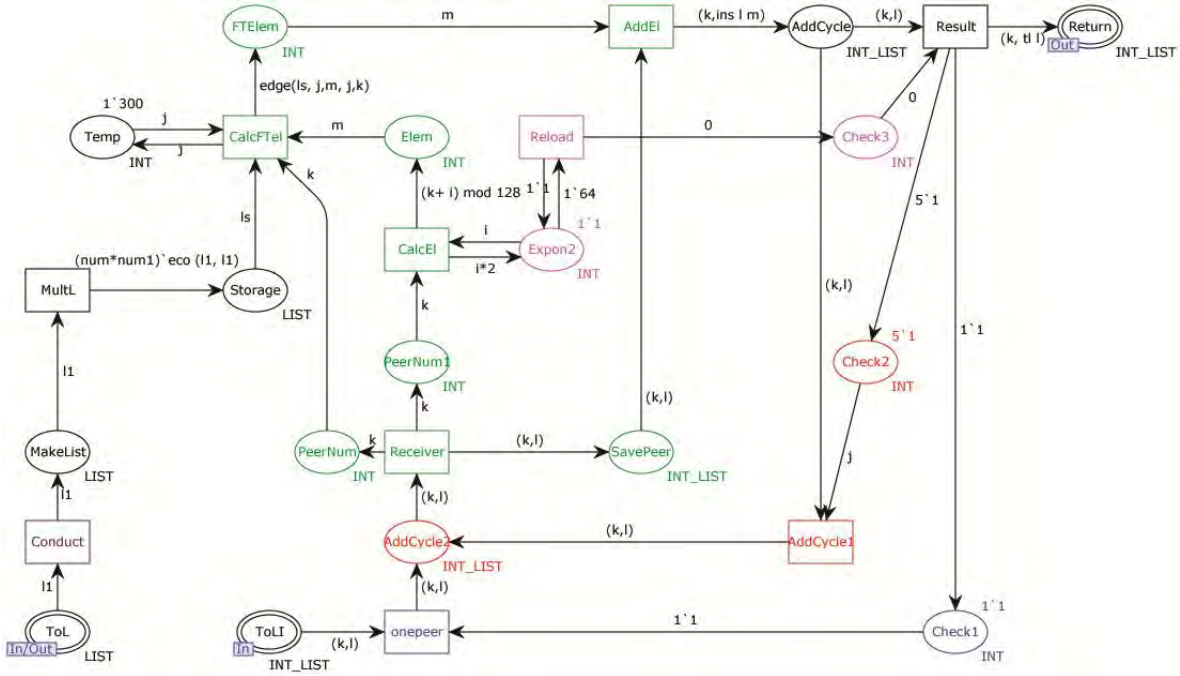

Рис. 2. Обновление таблицы

Далее будет описываться работа каждого перехода и позиции, некоторые из них работают вместе, поэтому будут объединены в группы. MultL создает столько списков всех пиров, сколько раз будет работать цикл. 1peer и Check1 «пропускают» в цикл только по одной метке. AddCycle, AddCycle1 и Check2 проверяют, чтобы таблица была полностью заполнена. CaclEl и Expon2 — цикл для расчета степеней двойки. CalcFTel при помощи рекурсивной функции edge() рассчитывает следующий элемент, который будет необходимо добавить в finger table. Когда в цикл перестанут входить данные, пиры возвращаются в Return.

Эксперименты

Рассмотрим работу модели на следующих примерах.

- Несколько примеров топологий кольца. Проверка работоспособности, модели протокола при разном количестве пиров в сети: 10, 7, 5.

- Запрос данных из системы. Запрос моделируется двумя параметрами, на каком узле он сейчас находится и куда он должен в итоге поступить.

- Динамический сценарий работы системы. Инициализация системы, удаление узла, обращение к удаленному узлу.

Рассмотрим один эксперимент. На рис. 3 приведена начальная разметка сети. Выполняется проверка работоспособности для сети из 5 пиров.

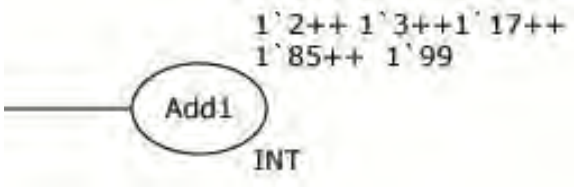

Рис. 3. Добавление фишек, соответствующих узлам

При таких входных данных исполнение модели сработало за 178 шагов. На рис. 4 виден результат работы, который показывает корректную инициализацию кольца DHT.

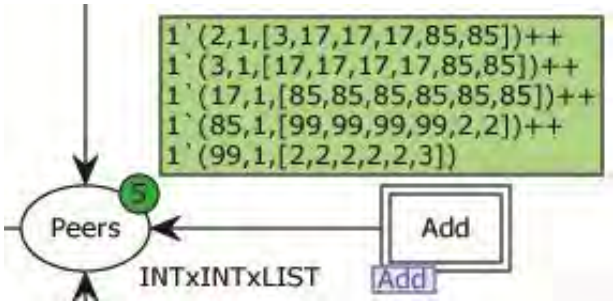

Рис. 4. Данные о пирах в DHT после выполнения сети Петри

В табл. 1 виден результат трех экспериментов, где производится выполнение модели с разными исходными данными.

Таблица 1

Количество шагов, требуемых для инициализации системы

Количество узлов	Количество шагов
5	178
7	248
10	353

Для более детального анализа проверим свойства графа достижимости. Для получения данных о графе достижимости был написан следующий программный код:

```
val q = NoOfNodes();
val w = NoOfArcs ();
val r = EntireGraphCalculated ();
val y = ListLiveTIs();
val u = ListDeadMarkings();
val i = st_Mark.Top'Peers 1 100;
UpperInteger (Mark.Top'Peers 1);
LowerInteger (Mark.Top'Peers 1);
```

Данный программный код позволяет исследовать свойства:

1. Статистический анализ графа достижимости.

- NoOfNodes() — количество вершин в графе достижимости;
- NoOfArcs()— количество дуг;
- EntireGraphCalculated() — функция проверяет, весь ли граф построен;
- st_Mark — определяет, какая разметка содержится в позиции with на странице Top 100-го узла графа достижимости.

2. Проверка графа на наличие тупиковых состояний.

- ListDeadMarkings()— возвращает список тупиковых состояний в графе достижимости. В нашей модели в любом тупиковом состоянии все пиры должны иметь обновленные DHT.

3. Проверка на наличие бесконечных циклов.

- ListLiveTIs()— выводит список бесконечных циклов в графе достижимости. Модель должна выполнить свою задачу за конечное число шагов, т. е. бесконечных циклов быть не должно.

4. Проверка ограниченности.

- UpperInteger — возвращает максимальное количество фишек в позиции with на страничке Top;

- LowerInteger — возвращает минимально количество фишек в позиции with на страничке Top. В модели должно быть минимальное количество 0, а максимальное равно количеству фишек в сети.

Построен граф достижимости для инициализации трех пиров с идентификаторами 2, 70, 120. Свойства графа достижимости показаны как результат срабатывания кода, разработанного для среды CPN Tools:

```
val q = 14180 : int
val w = 29441 : int
val r = true : bool
val y = [] : TI.TransInst list
val u = [14178,14177] : Node list
val i =
 "Top'Peers 1: 1`(2,1,[70,70,70,70,70,70])++\n1`(70,1,[120,120,120,120,12#"
 : string
val i =
 "Top'Peers 1: 1`(2,1,[70,70,70,70,70,70])++\n1`(70,1,[120,120,120,120,12#"
 : string
val it = 3 : int
val it = 0 : int
```

Как видим, всего вершин в графе 14 180, дуг — 29 441. Значение переменной r означает, что граф достижимости построен полностью. Если в переменной y хранится пустой список — в графе нет бесконечного цикла, что подтверждает корректность протокола, т. к. процесс инициализации должен проходить за конечное время. В переменную u записываются вершины, которые являются тупиковыми состояниями. Следующие 2 переменные как раз и проверяют, какие метки находятся в состояниях 14 178 и 14 177. Последние 2 переменные указывают на ограничение позиции Peers, которое равно от 0 до 3.

Заключение

В работе рассмотрены вопросы моделирования и анализа протокола Chord с помощью раскрашенных сетей Петри, описана работающая модель и модификации к ней. Добавлена возможность мониторинга выполнения запроса. Запросом является построение маршрута от одного узла к другому.

Эксперименты, проведенные над моделью протокола Chord, показали корректность протокола. А именно, при об-

ращении к только, что удаленному узлу ссылка на данный узел не будет получена.

Предложены подходы к анализу полученной модели. В дальнейшем предполагается проведение более детального анализа производительности протокола Chord.

Ссылки

1. Зайцев Д. А., Шмелева Т. Р. Моделирование телекоммуникационных систем в CPN Tools: учебное пособие по курсу «Математическое моделирование информационных систем». Одесса, 2008. 68 с.

2. Козюра В. Е., Непомнящий В. А., Новиков Р. М. Верификация раскрашенных сетей Петри методом проверки моделей: учебное пособие. Новосибирск, 2001. 26 с.

3. Башкин В. А. Функциональное программирование на языке SML: методические указания. Ярославль: ЯрГУ, 2007. 39 с.

4. Kulik J., Rabiner W., Balakrishnan H. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks // International Conference on Mobile Computing and Networking. Seattle, 1999.

5. Allavena A., Demers A., Hopcroft J. Correctness of a Gossip-based Membership Protocol // Symposium on Principles of Distributed Computing (PODC). Las Vegas, Nevada, 2005. P. 292–103.

6. Chaly D., Sokolov V. An Extensible Coloured Petri Net Model of a Transport Protocol for Packet Switched Networks // Lecture Notes in Computer Science Volume 2763, 2003. P. 66–75.

7. Chord: A scalable peer-to-peer lookup service for internet applications / I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan // ACM SIGCOMM Computer Communication Review. 2001.

8. Документация системы CPN Tools. URL: http://www.cpntools.org (дата обращения: 23.06.2015).