Известия Томского политехнического университета. 2009. Т. 314. № 5



УДК 681.3

**АЛГОРИТМ ВЗАИМНОГО ИСКЛЮЧЕНИЯ В ПИРИНГОВЫХ СИСТЕМАХ**

В.В. Губарев, A.A. Обейдат

Новосибирский государственный технический университет

E

*Предложен алгоритм взаимного исключения одновременного доступа разных процессов к одному и тому же объекту в динами ческих П2П системах, ориентированный на уменьшение служебного трафика. Основная идея его – передача сообщений между запросными узлами и координатором. Информация n дубликатов объекта Rj публикуется в n узлах, (в координаторе и его канди датах). Узлы посылают запрос координатору собственников дубликатов, чтобы получить доступ к объекту. В работе описывается актуальность, существо алгоритма, экспериментально, путем имитации, оценивается его масштабируемость и эффективность.*

***Ключевые слова:***

*Распределенные системы, пиринговые системы, взаимное исключение, Интернет*

**1. Введение и постановка задачи**

Одно из последних направлений распределенных информационных систем и сетей – пиринговые сети

– переживает в настоящее время активное становле ние [1–5]. В связи с молодостью и перспективностью в теории их создания и практического применения появляются все новые и новые фундаментальные за дачи, требующие неотложного решения. Одной из та ких является задача взаимного исключения – устра нения возможности одновременного доступа разных пользователей к одному и тому же объекту сети в тех критических ситуациях, когда одновременное выпол нение запросов этим объектом не допускается. Хотя эта задача решается в традиционных средствах парал лельных вычислений, специфика П2П не всегда по зволяет эффективно использовать известные реше ния. Существующие для аналогичных П2П решения удовлетворяют не всем требованиям П2П

* + связи с этим цель настоящей работы – разра ботка алгоритма взаимного исключения в П2П, от вечающего требованиям эффективности, надежно сти и масштабируемости. Для достижения поста вленной цели в работе решаются следующие задачи:

1. Описание модели рассматриваемых систем.
2. Рассмотрение операций процесса тиражирова ния дубликатов объекта.
3. Описание и исследование нового алгоритма взаимного исключения, отличающегося отсут ствием недостатков, присущих существующим алгоритмам, и удовлетворяющего требованиям упорядоченности, масштабируемости, децен трализованности и отказоустойчивости сети.
4. Имитационное сравнение предложенного алго ритма с существующими алгоритмами по мас штабируемости и эффективности.

**2. Модель рассматриваемых систем**

Рассматриваемые в данной работе алгоритмы основаны на следующей модели, описывающей динамические пиринговие системы.

1. Система состоит из равноправных узлов. Обоз

начим множество узлов через {*pi*,*i*=0,*N* –1}, где

*N* – общее количество узлов в системе, а *pi* – узел с идентификатором *id*. Каждый узел может быть клиентом (получателем) и в то же время сервером (поставщиком) объектов. Узел, высту пающий как сервер, т. е. имеющий дубликат совместно используемого объекта (данных) и готовый представить его в ответ на запрос, на зовем серверным узлом (СУ).

1. Каждому доступному пользователям объекту *Rj* (реальному файлу данных или вычислительно му объекту) соответствует определенный набор узлов, содержащих дубликаты запрашиваемых

пользователями объектов. Набор объектов

Ω

обозначим через ={*Rj*,*j*= 0,*m*–1}, где *m* – коли чество объектов, а набор дубликатов (реплик) *rjl*, соответствующих объекту *R* , обозначим через

*j*

Г*j*={*rjl*,*l*= 0,*n* –1}, где *n* – количество дубликатов. Каждый дубликат *rjl* объекта *Rj* содержится в определенном узле. Мы условно назовем узел, который имеет дубликат *l* объекта, копией (ре пликой или дубликатом) *l*.

1. Дубликаты всегда доступны, но их внутреннее состояние может быть случайно сброшено вследствие сбоя. После обнаружения сбоя ду бликат вновь восстанавливается в сети с преж ним значением идентификатора узла *id*.
2. Каждый объект *Rj* имеет уникальный двоичный

*m Kj*, который вычисляется как хэш текстового имени объекта

(например, *Х*). Идентификатор *Kj* может быть

вычислен с использованием конфликтоустой

чивой (не допускающей повторения) хэш

функции (например, SHA

идентификатор узлов. Узел *vj* системы с иденти

фикатором, совпадающим с идентификатором

*Kj* объекта *Rj*, выступает в качестве координато

ра набора Г*j* дубликатов объекта.

**3. Тиражирование дубликатов объекта**

Одной из важнейших задач П2П является тира жирование дубликатов, обеспечение их аутентич ности. Существующие методы тиражирования ду бликатов на все узлы являются громоздкими. Поэ

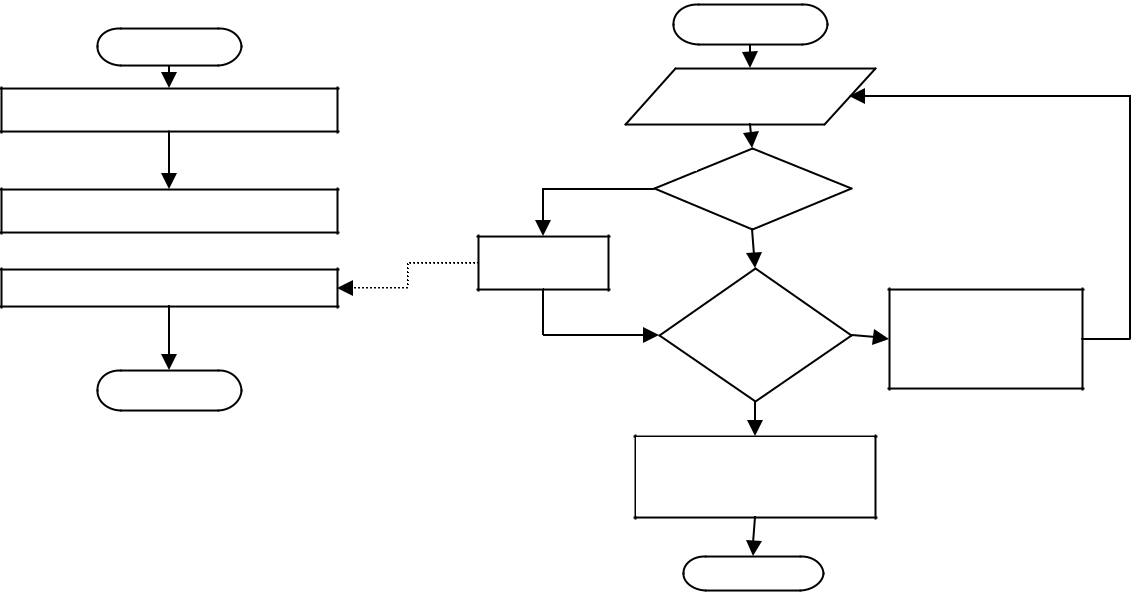
32

Управление, вычислительная техника и информатика

тому предлагается новый метод наборного (не по головного, а востребованного, текущего) тиражи рования.



Основная идея метода – тиражировать дублика ты изменяемых объектов не во всех узлах сети, а только в активных, т.е. по мере необходимости об ращения узлов к дубликатам. Для этого предлагает ся организация набора заинтересованных узлов, объединяемых координатором. Предусматривает ся, что собственники дубликатов некоторого объекта организованы в набор Г*j*. Каждое множе ство Г*j* имеет уникальный идентификатор *Kj*. Иден тификатор *Kj* набора присваивается с использова нием хеширования имени объекта. В качестве ко ординатора для соответствующего набора выступа ет узел *vj* с идентификатором, являющимся числен но наиболее близким к идентификатору набора *Kj*.



**3. 1.1. Операция «cоединение».** На рис. 1, *а* , при ведена схема операции *cоединение* для собственни ка дубликата объекта.

**Шаг 1.** Узел *p i* – собственник дубликата объекта просит оверлей вычислить хешированный иденти фикатор *K j* его объекта *R j .*

**Шаг 2.** *p i* просит оверлей послать сообщение *со единение М=(id, соединение, K j )* узлу *v j* с идентифи катором *id* , наиболее близким к хешированному идентификатору *K j* .

**Шаг 3.** *p i* получает ответ от координатора, что он стал членом набора Г *j* .

При получении координатором сообщения сое динение он начнет выполнять следующие шаги (см. рис. 1, *б* ):

**Шаг 1** . Координатор *v j* или кандидат получает сообщение *M* ( *соединение* ) .

3.1. Операции метода

*Соединение*: чтобы получать обновление дубли ката, узел, содержащий дубликат, который надо поддерживать в современном состоянии, иници ирует операцию «Cоединение» (т. е. присоедине ния к набору).

*Разъединение*: если узлу не требуется поддержи вать свой дубликата в современном состоянии, он инициирует операцию «разъединение» (т. е. отсое динения от набора), тем самым оставив без внима ния текущие модификации дубликата, экономя ре сурсы (свои и сети).

*Восстановление*: при восстановлении и возвра щении в систему, узел инициирует операцию «вос становление», по выполнении которой получает современную версию дубликата объекта.

**Шаг 2.** Координатор *v j* проверяет, пришло ли это сообщение от собственника дубликата или нет. Если да, он (на **Шаге 3** ) отправляет собственнику *ответ*, содержащий текущую версию дубликата объекта*.*

**Шаг 4**. Координатор *v j* проверяет, существует ли набор Г*j* с идентификатором *K j* или нет.

**Шаг 5**. Если набор существует, это означает, что узел *vj* уже является координатором множества Г *j* . В этом случае *v j* добавляет информацию о собствен нике дубликата в список собственников и отпра вляет сообщение *соединение* узлу *v j '* , чей идентифи катор наиболее близок к нему, чтобы этот узел стал новым кандидатом координатора *v c* . Число канди датов вместе с координатором равно числу дубли катов.

**ɇɚɱɚɥɨ**

**ɇɚɱɚɥɨ**

1

*Kj*=ɯɷɲ(*Rj*)

2

|  |  |
| --- | --- |
| ɉɨɫɥɚɬɶ M ɤ ɤɨɨɪɞɢɧɚɬɨɪɭ | 3 |

3

 ɉɨɫɥɚɬɶ ɨɬɜɟɬ ɉɨɥɭɱɢɬɶ ɨɬɜɟɬ

Ʉɨɧɟɰ

1

ɉɨɥɭɱɢɬɶ

ɫɨɨɛɳɟɧɢɟ (Ɇ)

Ⱦɚ 22

* ɨɬ

ɫɨɛɫɬɜɟɧɧɢɤɚ?

ɇɟɬ

1. ɞɚ

ȿɫɬɶ ɥɢ \**j*

* + *Ʉj*?

ɇɟɬ

6

5

* *j*= \**j*  M.id

ɉɟɪɟɫɥɚɬɶ Ɇ ɤ *vj*ƍ;

*ɚ* . ɋɨɛɫɬɜɟɧɧɢɤ ɞɭɛɥɢɤɚɬɚ

***Рис.*** ***1.*** *Схема* *операции* *«соединение»*

ɋɨɡɞɚɬɶ \* *j* ;

* *j*= \**j*  M.id;

Ʉɨɧɟɰ

*ɛ*. Ʉɨɨɪɞɢɧɚɬɨɪ

33

Известия Томского политехнического университета. 2009. Т. 314. № 5



**Шаг 6**. Если набор не существует, источник со общения – собственник дубликата осознает, что он является координатором *vj* для нового набора соб ственников дубликатов Г*j*, поэтому должен создать список Г*j* и добавить отправителя сообщения в свой список; в противном случае он будет кандидатом *vc* для координатора.

**3.1.2. Операция «*разъединение»*.** Когда узлу *pi* не требуется поддерживать свой дубликата в совре менном состоянии, он инициирует операцию разъединения, чтобы выйти из набора. Координа тор, получая это сообщение, удаляет информацию

* дубликате узла *pi* из своего списка и информирует об этом своих кандидатов.

**3.1.3. Операция «воссоединение».** Шаги опера ции восстановления следующие.

Шаг 1. Когда любой узел *pi* желает получить со временную версию дубликата объекта, он посылает своему *vj* сообщение *ВОССТАНОВЛЕНИЕ* с указа нием имеющейся в узле версии дубликата.

Шаг 2. При получении сообщения *ВОССТАНО ВЛЕНИЕ*, *vj* определяет различие между последнейверсией дубликата, находящегося в системе, и вер сией дубликата узла *pi*.

Шаг 3. Если отличия в дубликатах нет, *vj* вклю чает узел *pi* в набор. Если дубликаты отличаются, *vj* отправляет сообщение *ОБНОВЛЕНИЕ* узлу *pi*, со держащее список модификаций *OPS* .

Шаг 4. При получении сообщения *ОБНОВЛЕ НИЕ* узел *pi* выполняет модификации в списке *OPS*

* приводит свой дубликат объекта в соответствие с текущей версией объекта. После этого узел *pi* вклю чается в набор Г*j*.

1. **Алгоритм взаимного исключения одновременного доступа для изменения объекта**

Вторая важная задача – это взаимное исключе ние доступа к одному и тему же объекту или его ду бликатам, когда одновременный доступ недопу стим, нежелателен (один узел хочет читать содер жимое объекта, другой записывать или изменять его либо оба узла хотят изменить содержимое объекта). Недостатки существующих методов – в трудности их масштабирования.

Предложенный алгоритм основан на организа

ции набора узлов и передаче сообщений только меж

ду запросными узлами и координатором. Поэтому

назовем его условно алгоритмом *узел*

(У2К). Запросные узлы посылают запрос координа

тору, чтобы получить доступ к объекту. При получе

нии запроса на использование объекта *Rj* координа

тор *vj* уведомляет об этом всех своих кандидатов и по

сылает ответ запросному узлу. Ответ содержит ин

формацию о текущем владельце, работающем с

объектом, и очереди ожидающих узлов. После завер

шения работы с объектом узел посылает координа

тору сообщение ОСВОБ. Когда *vj* получает сообще

ние ОСВОБ от текущего владельца объекта, он уве

домляет об этом все запросные узлы набора, включая

* кандидатов координатора, и, если только что за вершивший работу с объектом узел производил мо дификацию объекта, сообщает им о новой версии объекта, а при необходимости тиражирует ее.

Полномочие доступа к объекту предоставляется при одновременном соблюдении следующих усло вий:

1. Допустимо множество операций чтения в одно и то же время.
2. Разрешается только одна операция записи в од но то же время.
3. Отсутствие (запрет) пары или поле операций «чтение – запись» в одно и то же время.

Шаги алгоритма У2К следующие. Входящими данными алгоритма являются: *id*: идентификатор узла *i*;

***cj***: идентификатор клиента, владеющего объек том *Rj*; ***tcj***: время получения полномочий (разреше ния) на доступ к объекту *Rj*; *tво*: временная отметка, переменная состояния, всегда хранящаяся в *pi*, из начально ноль;

***pQj***: частичный перечень ожидающих запросов, который содержит сведения о запросах всех клиен тов, упорядоченных по времени; ***Qj***: полная очередь запросов, изначально пустая, *Qj* используется толь ко координатором;

*Tl*: *время аренды*, полномочие на доступ объекта предоставляется на некоторое время *Tl*. Координа тор отслеживает время окончания блокировки и снимает ее, когда время *Tl* истекает;

***Rj***: идентификатор объекта *j*;

***M***: сообщения между узлами, которые обычно содержат: ***s*** – идентификатор источника сообще ния, запросного узла *ps*; *tво* – временная отметка за проса; ***pQj*** – частичная очередь запросов; ***d*** – пункт доставки сообщения; (***cj***,***tcj***) – текущий собственник

* временная отметка; Тип сообщения: (ЗАПРОС,

ОСВОБОЖДЕНИЕ (ОСВОБ), ОТВЕТ).

**Шаг 1.** Когда узел *pi* хочет получить доступ к определенному объекту *Rj*, он посылает координа тору сообщение *ЗАПРОС*(*s*=*id*, *tво*, *d=Кj*). В качестве координатора рассматривается узел системы с идентификатором, наиболее близким к хеширо ванному идентификатору *Кj* ассоциированного объекта *Rj*.

**Шаг 2***.* Когда узел *vj*, который выступает в каче стве координатора для набора дубликатов, получа ет запрос на доступ к объекту *Rj* от узла *pi*, он опре деляет, доступен объект *Rj* или нет. Если да, то он

блокирует объект *Rj* для узла *pi* (т. е. ***cj***=*pi*, ***tcj***=*M*·*tво*), посылает сообщение ОТВЕТ «Да» запросному узлу

* информирует об этом его кандидатов. В против ном случае он добавляет запрос в очередь**.**

**Шаг 3**. Когда узел *pi* получает сообщение ОТ ВЕТ «да», он проверяет, является ли он сам вла дельцем объекта или нет. Если да, он может войти в критическую секцию (КС) и начать использовать

34

Управление, вычислительная техника и информатика



объект *Rj*. Когда он заканчивает свою работу в КС, он посылает *vj* сообщение ОСВОБ. В противном случае, если он получает сообщение ОТВЕТ «нет», ему требуется ждать в течение определенного вре мени ожидания *tw=l.tkc*, где *l* – число узлов в очере ди *pQj*, *tkc* – время нахождения узла в КС.

**Шаг 4**. При получении координатором сообще ния ОСВОБ от текущего владельца КС, он проверяет, пуста ли очередь запросов. Если нет, он исключает из очереди нового владелица, посылая ему ОТВЕТ «да»,

* рассылает сообщение ОСВОБ всем остальным уз лам очереди *Qj*. Если координатор не получит от теку щего узла сообщение ОСВОБ в течение прогнозного времени (*Tl*), он поймет, что владелец объекта вышел из строя. Тогда он даст полномочие на доступ к объекту следующему узлу в очереди. Если очередь за кончилась, координатор ожидает, новые запросы.

**Шаг 5.** При получении от координатора сооб щения ОСВОБ, узел *pi* ждет свою очередь. Если узел *pi* не получает сообщение ОТВЕТ или ОСВОБ

* течение прогнозного времени *tw*, он вновь (шаг 1) посылает сообщение ЗАПРОС узлу в качестве ко ординатора.

Итак, шаги 1–5 повторяются по мере того, как узлы хотят получить доступ к совместно использу емому объекту *Rj*.

**5. Свойства У2К**

Теперь рассмотрим, какие свойства имеет У2К и, следовательно, каким требованиям он удовле творяет.

Первое свойство – **упорядоченность.** Прежде всего отметим, что У2К соответствует требованию **упорядоченности**, т. к. запросы удовлетворяются впорядке «первым пришёл, первым обслужен (FCFS)». Действительно, только координатор *vj* со держит *Qj* и принимает решение относительно до ступа к объекту в соответствии со своей информа цией. После получения сообщения (ЗАПРОС или ОСВОБ) координатор проверяет состояние объек та. Если он свободен, *vj* посылает сообщение «ДА» следующему в очереди узлу. Это сообщение содер жит разрешение на использование объекта, доступ

* его КС. По его получении запросный узел входит
* КС. Только *vj* может предоставить полномочие на доступ к общему объекту. Таким образом У2К обла дает свойством упорядоченности.

Второе важное свойство – **степень загрузки сети служебным трафиком**. Как нетрудно убедиться, онабудет ниже, чем в существующих алгоритмах, по скольку У2К использует меньшее количество опе раций, чем другие алгоритмы. На самом деле обмен сообщениями в алгоритме осуществляется только между запросными узлами и координатором вме сто обмена сообщениями со всеми узлами как в су ществующих алгоритмах. Например, сообщение ЗАПРОС посылается только координатору, а не всем узлам, хранящим *n* дубликатов, как в других алгоритмах, основанных на кворуме [7, 8].

Свойство **децентрализации**. У2К достигает луч ших показателей децентрализации и **отказоустойчи вости**, т. к. *Qj* хранится в *vj* и его кандидатах, а частичные очереди – во всех запросных узлах. Поэтому все узлы локально определяют следующего владельца объекта при получении сообщения ОСВОБ или по окончании времени аренды объекта. Несмотря на то, что *Qj* хранится в координаторе и кандидатах, У2К гарантирует взаимное исключение, но с другой сто роны У2К **отказоустойчив**: когда координатор и кан дидаты выходят из строя одновременно, запросные узлы могут и узнают об этом, не получая сообщение ОСВОБ от текущего владельца объекта в пределах времени аренды. Это означает, что когда координа тор выходит из строя, запросные узлы вновь посыла ют свои запросы с частичными очередями (*pQj*) но вому координатору. Новый координатор объединит все полученные *pQj* в одну полную очередь *Qj*.

Свойство **аутентичности.** Требование аутентич ности удовлетворяется, если в любой момент вре мени содержимое всех востребованных дубликатов объекта одинаково (т. е. удовлетворяется условие тождественности всех копий одной). Это требова ние можно обеспечить разными путями. Первый – одновременной заменой всех старых дублей изме ненного объекта новой редакцией (тиражировани ем новых дублей) сразу же по окончании измене ния объекта до выполнения следующего по очере ди запроса. Второй путь, описанный в п. 2, выдачей последней редакции дублей только тем узлам, ко торые за ними сейчас обращаются. Ясно, что при этом, чтобы обеспечить аутентичность дубликатов последней версии объекта, У2К, использующий метод отслеживания активных дубликатов объекта,

* случае остановки, сбоя или тиражирования ново го дубликата объекта вначале должен обновить со держимое старого дубликата объекта.

**6. Результаты имитационного моделирования У2К**

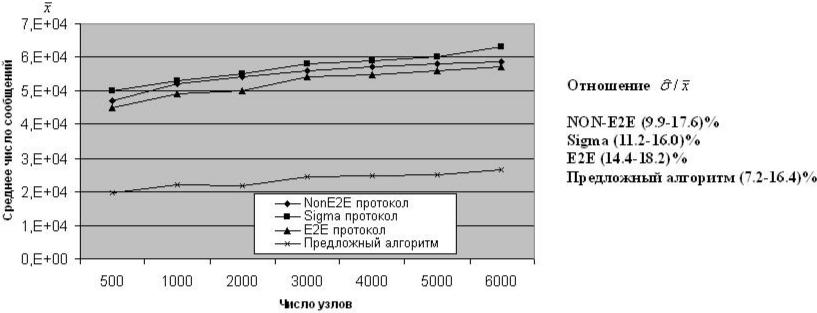
Для сравнения У2К и других известных [7, 8] алгоритмов взаимного исключения применим под ход имитационного экспериментирования с алго ритмами. Программную реализацию предложен ного и существующих алгоритмов взаимного ис ключения выполним на основе системы FreePastry [9, 10]. Разработанная программа, стартуя, создает набор узлов, расставленных согласно сетевой топо логии Pastry. Цели моделирования – проверить масштабируемость и загрузку сети служебным тра фиком (т. е. определить число сообщений).

Одним из главных качеств алгоритма взаимного исключения является его пригодность к **масштаби руемости** сети, т. е. к увеличению количества узловсети. Для экспериментального компьютерного ис следования масштабируемости У2К по сравнению

* другими, хорошими по этому показателю, алго ритмами, выполняем их одинаковую программную реализацию. Для этого следует реализовать на ЭВМ компьютерные модели сетей и запустить на компьютере несколько раундов работы смоделиро

35

Известия Томского политехнического университета. 2009. Т. 314. № 5



***Рис. 2.*** *Сравнение алгоритмов по среднему количеству сообщений*

ванных сетей. Выполняем несколько раундов мо делирования (в среднем свыше десяти) и находим средние значения экспериментальных результатов.

* каждом раунде каждая сеть (точнее её модель) проделывает весь тот объем работы, который про делывают узлы в реальной сети. При этом от раун да к раунду может меняться число узлов сети, а так же число запросов доступа к объектам, посылае мых множеством случайно выбранных узлов. Дела ется это таким образом, что одинаковые показате ли параметров сети остаются неизменными для всех сравниваемых в том же раунде алгоритмов. А именно, в каждом раунде узел наугад освобождает 50 % удерживаемых им объектов. Число объектов, дубликатов и запросов в каждом раунде постоянно и составляет 65, 10 и 20 соответственно.

На рис. 2 представлено сравнение У2К с алго ритмами Sigma [7], E2E [8] и NONE2E [8] по сред нему числу сообщений. Как видно из рисунка, все зависимости почти линейны относительно числа узлов. В У2К загрузка сети служебным трафиком в среднем в 2 раза меньше, чем в сквозном Е2Е алго ритме. Следовательно, У2К имеет лучшие свойства

по **масштабируемости**. В правой части рис. 2 пред

ставлены отношения выборочного среднеквадра

*σ* *–*

тического отклонение к среднему *x* числу сооб щений. Как видно из приведенных данных, пред ставленные на рисунке зависимости характеризу

ются сравнительно малой стохастичностью (отно *σ –*

шения /*x* – единицы либо один, два десятка %),

* е. близки к функциональным. Это подтверждает справедливость выводов по результатам имитации.

**Выводы**

Описаны новые алгоритмы тиражирования ду бликатов и взаимного исключения в П2П. Обосно вана актуальность разработки и приведены модель системы, общее описание алгоритмов, их операции

* свойства. Полученные результаты имитации ал горитма взаимного исключения выявили, что он обладает намного лучшей эффективностью, чем существующие алгоритмы, и хорошо масштаби руем. Таким образом, алгоритм способен поддер живать большее количество узлов. Предложенные алгоритмы инвариантны к его разной сетевой реа лизации в структурированных П2П.

**СПИСОК ЛИТЕРАТУРЫ**

1. Мартин Дж. Системный анализ передачи данных. – М.: Мир, 1975. – Т. 1. – 256 с.
2. Обейдат А.А., Губарев В.В. Обзор алгоритмов распределенного взаимного исключения в динамических пиринговых системах // Сб. научных трудов НГТУ. – 2007. – № 2 (48). – С. 63–68.
3. Ganesan P., Gummadi P.K., Garcia

Designing DHTS with hierarchical structure // ICDCS. – Tokyo, 2004. – P. 263–272.

1. Kumar A. Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data // IEEE Transactions on Computers. – 1991. – V. 40. – № 9. – P. 996–1004.
2. Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H. Chord: A scalable peer

tions // Proc. of ACM SIGCOMM. – 2001 [Электронный ре сурс]. – Режим доступа: – URL: http://iptps04.cs.ucsd.edu/ pa pers/karger

1. FIPS 180ech. Rep. Publication 180

Federal Information Processing Standard (FIPS), National Institu

te of Standards and Technology, US Department of Commerce, Washington, D.C., April, 1995.

1. Shiding Lin, Qiao Lian, Ming Chen, Zheng Zhang. A practical di stributed mutual exclusion protocol in dynamic peer

stems // 3rd International Workshop on Peereer Systems (IPTPS’04). – San Diego, CA, USA, Feb., 2004. – P. 1–10.

1. Muhammad M., Cheema A.S., Gupta I. Efficient mutual exclusion

in peer th IEEE/ACM International Conference on Grid Computing. – 2005. – P. 296–299.

1. Hoye J. Freepastry [Электронный ресурс]. – Режим доступа: http://freepastry.rice.edu/FreePastry/. – 15.01.2007.
2. Rowstron A., Druschel P. Pastry: Scalable, distributed object location and routing for large

Middleware, Nov. 2001 [Электронный ресурс]. – Режим доступа: research.microsoft.com/~antr/PAST/pastry.pdf. – 15.01.2007.

*Поступила 03.04.2009 г.*

36