УДК 681.3

АЛГОРИТМ ВЗАИМНОГО ИСКЛЮЧЕНИЯ В ПИРИНГОВЫХ СИСТЕМАХ

В.В. Губарев, А.А. Обейдат

Новосибирский государственный технический университет
E-mail: atefob@hotmail.com

Предложен алгоритм взаимного исключения одновременного доступа разных процессов к одному и тому же объекту в динамических П2П системах, ориентированный на уменьшение служебного трафика. Основная идея его – передача сообщений между запросными узлами и координатором. Информация n дубликатов объекта R_j публикуется в n узлах, (в координаторе и его кандидатах). 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-приложения.

1. Введение и постановка задачи

Одно из последних направлений распределенных информационных систем и сетей – пиринговые сети – переживает в настоящее время активное становление [1–5]. В связи с молодостью и перспективностью в теории их создания и практического применения появляются все новые и новые фундаментальные задачи, требующие неотложного решения. Одной из таких является задача взаимного исключения – устранения возможности одновременного доступа разных пользователей к одному и тому же объекту сети в тех критических ситуациях, когда одновременное выполнение запросов этим объектом не допускается. Хотя эта задача решается в традиционных средствах параллельных вычислений, специфика П2П не всегда позволяет эффективно использовать известные решения. Существующие для аналогичных П2П решения удовлетворяют не всем требованиям П2П-систем.

В связи с этим цель настоящей работы – разработка алгоритма взаимного исключения в П2П, отвечающего требованиям эффективности, надежности и масштабируемости. Для достижения поставленной цели в работе решаются следующие задачи:

1. Описание модели рассматриваемых систем.
2. Рассмотрение операций процесса тиражирования дубликатов объекта.
3. Описание и исследование нового алгоритма взаимного исключения, отличающегося отсутствием недостатков, присущих существующим алгоритмам, и удовлетворяющего требованиям упорядоченности, масштабируемости, децентрализованности и отказоустойчивости сети.
4. Имитационное сравнение предложенного алгоритма с существующими алгоритмами по масштабируемости 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. Обозначим множество узлов через $\{p_i, i=\overline{0,N-1}\}$, где N – общее количество узлов в системе, а p_i – узел с идентификатором *id*. Каждый узел может быть клиентом (получателем) и в то же время сервером (поставщиком) объектов. Узел, выступающий как сервер, т. е. имеющий дубликат совместно используемого объекта (данных) и готовый представить его в ответ на запрос, назовем серверным узлом (СУ).
2. Каждому доступному пользователям объекту R_j (реальному файлу данных или вычислительному объекту) соответствует определенный набор узлов, содержащих дубликаты запрашиваемых пользователями объектов. Набор объектов обозначим через $\Omega=\{R_j, j=\overline{0,m-1}\}$, где m – количество объектов, а набор дубликатов (реплик) r_{jl}, соответствующих объекту R_j, обозначим через $\Gamma_j=\{r_{jl}, l=\overline{0,n-1}\}$, где n – количество дубликатов. Каждый дубликат r_{jl} объекта R_j содержится в определенном узле. Мы условно назовем узел, который имеет дубликат l объекта, копией (репликой или дубликатом) l.
3. Дубликаты всегда доступны, но их внутреннее состояние может быть случайно сброшено вследствие сбоя. После обнаружения сбоя дубликат вновь восстанавливается в сети с прежним значением идентификатора узла *id*.
4. Каждый объект R_j имеет уникальный двоичный m-битный идентификатор (ключ) K_j, который вычисляется как хэш текстового имени объекта (например, X). Идентификатор K_j может быть вычислен с использованием конфликтоустойчивой (не допускающей повторения) хэш-функции (например, SHA-1 [6]), так же, как идентификатор узлов. Узел v_j системы с идентификатором, совпадающим с идентификатором K_j объекта R_j, выступает в качестве координатора набора Γ_j дубликатов объекта.

3. Тиражирование дубликатов объекта

Одной из важнейших задач П2П является тиражирование дубликатов, обеспечение их аутентичности. Существующие методы тиражирования дубликатов на все узлы являются громоздкими. Поэ-

тому предлагается новый метод наборного (не поголовного, а востребованного, текущего) тиражирования.

Основная идея метода – тиражировать дубликаты изменяемых объектов не во всех узлах сети, а только в активных, т.е. по мере необходимости обращения узлов к дубликатам. Для этого предлагается организация набора заинтересованных узлов, объединяемых координатором. Предусматривается, что собственники дубликатов некоторого объекта организованы в набор Γ_j. Каждое множество Γ_j имеет уникальный идентификатор K_j. Идентификатор K_j набора присваивается с использованием хеширования имени объекта. В качестве координатора для соответствующего набора выступает узел v_j с идентификатором, являющимся численно наиболее близким к идентификатору набора K_j.

3.1. Операции метода

Соединение: чтобы получать обновление дубликата, узел, содержащий дубликат, который надо поддерживать в современном состоянии, инициирует операцию «Соединение» (т. е. присоединения к набору).

Разъединение: если узлу не требуется поддерживать свой дубликата в современном состоянии, он инициирует операцию «разъединение» (т. е. отсоединения от набора), тем самым оставив без внимания текущие модификации дубликата, экономя ресурсы (свои и сети).

Восстановление: при восстановлении и возвращении в систему, узел инициирует операцию «восстановление», по выполнении которой получает современную версию дубликата объекта.

3.1.1. Операция «соединение». На рис. 1, *а*, приведена схема операции *соединение* для собственника дубликата объекта.

Шаг 1. Узел p_i – собственник дубликата объекта просит оверлей вычислить хешированный идентификатор K_j его объекта R_j.

Шаг 2. p_i просит оверлей послать сообщение *соединение M=(id, соединение, K_j)* узлу v_j с идентификатором *id*, наиболее близким к хешированному идентификатору K_j.

Шаг 3. p_i получает ответ от координатора, что он стал членом набора Γ_j.

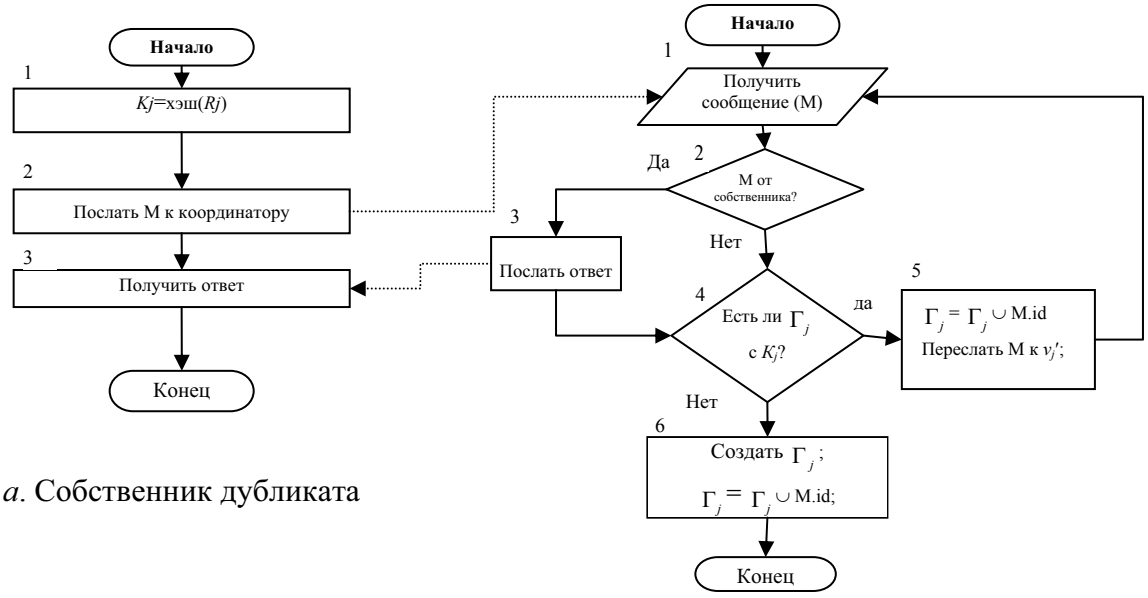
При получении координатором сообщения соединение он начнет выполнять следующие шаги (см. рис. 1, *б*):

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

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 Γ_j с идентификатором K_j или нет.

Шаг 5. Если набор существует, это означает, что узел v_j уже является координатором множества Γ_j. В этом случае v_j добавляет информацию о собственнике дубликата в список собственников и отправляет сообщение *соединение* узлу v_j', чей идентификатор наиболее близок к нему, чтобы этот узел стал новым кандидатом координатора v_c. Число кандидатов вместе с координатором равно числу дубликатов.



а. Собственник дубликата

б. Координатор

Рис. 1. *Схема операции «соединение»*

Шаг 6. Если набор не существует, источник сообщения – собственник дубликата осознает, что он является координатором v_j для нового набора собственников дубликатов Γ_j, поэтому должен создать список Γ_j и добавить отправителя сообщения в свой список; в противном случае он будет кандидатом v_c для координатора.

3.1.2. Операция «*разъединение*». Когда узлу p_i не требуется поддерживать свой дубликата в современном состоянии, он инициирует операцию разъединения, чтобы выйти из набора. Координатор, получая это сообщение, удаляет информацию о дубликате узла p_i из своего списка и информирует об этом своих кандидатов.

3.1.3. Операция «воссоединение». Шаги операции восстановления следующие.

Шаг 1. Когда любой узел p_i желает получить современную версию дубликата объекта, он посылает своему v_j сообщение *ВОССТАНОВЛЕНИЕ* с указанием имеющейся в узле версии дубликата.

Шаг 2. При получении сообщения *ВОССТАНОВЛЕНИЕ*, v_j определяет различие между последней версией дубликата, находящегося в системе, и версией дубликата узла p_i.

Шаг 3. Если отличия в дубликатах нет, v_j включает узел p_i в набор. Если дубликаты отличаются, v_j отправляет сообщение *ОБНОВЛЕНИЕ* узлу p_i, содержащее список модификаций *OPS*.

Шаг 4. При получении сообщения *ОБНОВЛЕНИЕ* узел p_i выполняет модификации в списке *OPS* и приводит свой дубликат объекта в соответствие с текущей версией объекта. После этого узел p_i включается в набор Γ_j.

4. Алгоритм взаимного исключения одновременного доступа для изменения объекта

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

Предложенный алгоритм основан на организации набора узлов и передаче сообщений только между запросными узлами и координатором. Поэтому назовем его условно алгоритмом *узел-координатор* (У2К). Запросные узлы посылают запрос координатору, чтобы получить доступ к объекту. При получении запроса на использование объекта R_j координатор v_j уведомляет об этом всех своих кандидатов и посылает ответ запросному узлу. Ответ содержит информацию о текущем владельце, работающем с объектом, и очереди ожидающих узлов. После завершения работы с объектом узел посылает координатору сообщение ОСВОБ. Когда v_j получает сообщение ОСВОБ от текущего владельца объекта, он уведомляет об этом все запросные узлы набора, включая и кандидатов координатора, и, если только что завершивший работу с объектом узел производил модификацию объекта, сообщает им о новой версии объекта, а при необходимости тиражирует ее.

Полномочие доступа к объекту предоставляется при одновременном соблюдении следующих условий:

a) Допустимо множество операций чтения в одно и то же время.
b) Разрешается только одна операция записи в одно то же время.
c) 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*;

$\boldsymbol{c_j}$: идентификатор клиента, владеющего объектом R_j; $\boldsymbol{t_{cj}}$: время получения полномочий (разрешения) на доступ к объекту R_j; $t_{во}$: временная отметка, переменная состояния, всегда хранящаяся в p_i, изначально ноль;

$\boldsymbol{pQ_j}$: частичный перечень ожидающих запросов, который содержит сведения о запросах всех клиентов, упорядоченных по времени; $\boldsymbol{Q_j}$: полная очередь запросов, изначально пустая, Q_j используется только координатором;

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

$\boldsymbol{R_j}$: идентификатор объекта *j*;

M: сообщения между узлами, которые обычно содержат: $\boldsymbol{s}$ – идентификатор источника сообщения, запросного узла p_s; $t_{во}$ – временная отметка запроса; $\boldsymbol{pQ_j}$ – частичная очередь запросов; $\boldsymbol{d}$ – пункт доставки сообщения; ($\boldsymbol{c_j}$,$\boldsymbol{t_{cj}}$) – текущий собственник и временная отметка; Тип сообщения: (ЗАПРОС, ОСВОБОЖДЕНИЕ (ОСВОБ), ОТВЕТ).

Шаг 1. Когда узел p_i хочет получить доступ к определенному объекту R_j, он посылает координатору сообщение *ЗАПРОС*($s=id$, $t_{во}$, $d=K_j$). В качестве координатора рассматривается узел системы с идентификатором, наиболее близким к хешированному идентификатору K_j ассоциированного объекта R_j.

Шаг 2. Когда узел v_j, который выступает в качестве координатора для набора дубликатов, получает запрос на доступ к объекту R_j от узла p_i, он определяет, доступен объект R_j или нет. Если да, то он блокирует объект R_j для узла p_i (т. е. $\boldsymbol{c_j}=p_i$, $\boldsymbol{t_{cj}}=M\cdot t_{во}$), посылает сообщение ОТВЕТ «Да» запросному узлу и информирует об этом его кандидатов. В противном случае он добавляет запрос в очередь**.**

Шаг 3. Когда узел p_i получает сообщение ОТВЕТ «да», он проверяет, является ли он сам владельцем объекта или нет. Если да, он может войти в критическую секцию (КС) и начать использовать

объект R_j. Когда он заканчивает свою работу в КС, он посылает v_j сообщение ОСВОБ. В противном случае, если он получает сообщение ОТВЕТ «нет», ему требуется ждать в течение определенного времени ожидания $t_w = l \cdot t_{kc}$, где l – число узлов в очереди pQ_j, t_{kc} – время нахождения узла в КС.

Шаг 4. При получении координатором сообщения ОСВОБ от текущего владельца КС, он проверяет, пуста ли очередь запросов. Если нет, он исключает из очереди нового владелица, посылая ему ОТВЕТ «да», и рассылает сообщение ОСВОБ всем остальным узлам очереди Q_j. Если координатор не получит от текущего узла сообщение ОСВОБ в течение прогнозного времени (T_l), он поймет, что владелец объекта вышел из строя. Тогда он даст полномочие на доступ к объекту следующему узлу в очереди. Если очередь закончилась, координатор ожидает, новые запросы.

Шаг 5. При получении от координатора сообщения ОСВОБ, узел p_i ждет свою очередь. Если узел p_i не получает сообщение ОТВЕТ или ОСВОБ в течение прогнозного времени t_w, он вновь (шаг 1) посылает сообщение ЗАПРОС узлу в качестве координатора.

Итак, шаги 1–5 повторяются по мере того, как узлы хотят получить доступ к совместно используемому объекту R_j.

5. Свойства У2К

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

Первое свойство – **упорядоченность.** Прежде всего отметим, что У2К соответствует требованию **упорядоченности**, т. к. запросы удовлетворяются в порядке «первым пришёл, первым обслужен (FCFS)». Действительно, только координатор v_j содержит Q_j и принимает решение относительно доступа к объекту в соответствии со своей информацией. После получения сообщения (ЗАПРОС или ОСВОБ) координатор проверяет состояние объекта. Если он свободен, v_j посылает сообщение «ДА» следующему в очереди узлу. Это сообщение содержит разрешение на использование объекта, доступ в его КС. По его получении запросный узел входит в КС. Только v_j может предоставить полномочие на доступ к общему объекту. Таким образом У2К обладает свойством упорядоченности.

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

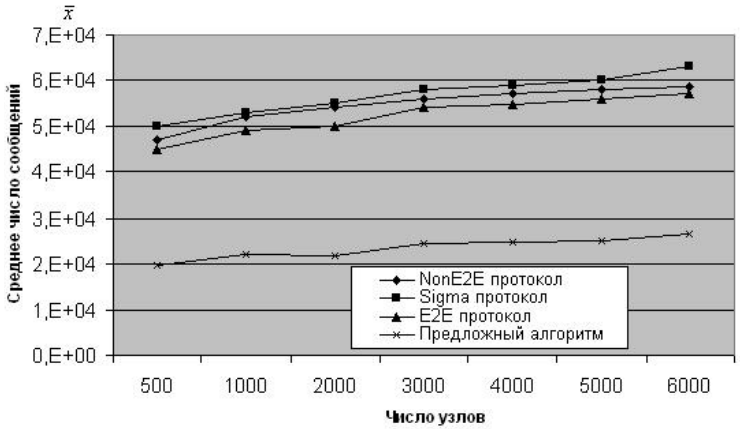
Свойство **децентрализации**. У2К достигает лучших показателей децентрализации и **отказоустойчивости**, т. к. Q_j хранится в v_j и его кандидатах, а частичные очереди – во всех запросных узлах. Поэтому все узлы локально определяют следующего владельца объекта при получении сообщения ОСВОБ или по окончании времени аренды объекта. Несмотря на то, что Q_j хранится в координаторе и кандидатах, У2К гарантирует взаимное исключение, но с другой стороны У2К **отказоустойчив**: когда координатор и кандидаты 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 координатор выходит из строя, запросные узлы вновь посылают свои запросы с частичными очередями (pQ_j) новому координатору. Новый координатор объединит все полученные pQ_j в одну полную очередь Q_j.

Свойство **аутентичности.** Требование аутентичности удовлетворяется, если в любой момент времени содержимое всех востребованных дубликатов объекта одинаково (т. е. удовлетворяется условие тождественности всех копий одной). Это требование можно обеспечить разными путями. Первый – одновременной заменой всех старых дублей измененного объекта новой редакцией (тиражированием новых дублей) сразу же по окончании изменения объекта до выполнения следующего по очереди запроса. Второй путь, описанный в п. 2, выдачей последней редакции дублей только тем узлам, которые за ними сейчас обращаются. Ясно, что при этом, чтобы обеспечить аутентичность дубликатов последней версии объекта, У2К, использующий метод отслеживания активных дубликатов объекта, в случае остановки, сбоя или тиражирования нового дубликата объекта вначале должен обновить содержимое старого дубликата объекта.

6. Результаты имитационного моделирования У2К

Для сравнения У2К и других известных [7, 8] алгоритмов взаимного исключения применим подход имитационного экспериментирования с алгоритмами. Программную реализацию предложенного и существующих алгоритмов взаимного исключения выполним на основе системы FreePastry [9, 10]. Разработанная программа, стартуя, создает набор узлов, расставленных согласно сетевой топологии Pastry. Цели моделирования – проверить масштабируемость и загрузку сети служебным трафиком (т. е. определить число сообщений).

Одним из главных качеств алгоритма взаимного исключения является его пригодность к **масштабируемости** сети, т. е. к увеличению количества узлов сети. Для экспериментального компьютерного исследования масштабируемости У2К по сравнению с другими, хорошими по этому показателю, алгоритмами, выполняем их одинаковую программную реализацию. Для этого следует реализовать на ЭВМ компьютерные модели сетей и запустить на компьютере несколько раундов работы смоделиро-



Отношение $\hat{\sigma} / \bar{x}$

NON-E2E (9.9-17.6)%
Sigma (11.2-16.0)%
E2E (14.4-18.2)%
Предложный алгоритм (7.2-16.4)%

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

ванных сетей. Выполняем несколько раундов моделирования (в среднем свыше десяти) и находим средние значения экспериментальных результатов. В каждом раунде каждая сеть (точнее её модель) проделывает весь тот объем работы, который проделывают узлы в реальной сети. При этом от раунда к раунду может меняться число узлов сети, а также число запросов доступа к объектам, посылаемых множеством случайно выбранных узлов. Делается это таким образом, что одинаковые показатели 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 алгоритмами Sigma [7], E2E [8] и NONE2E [8] по среднему 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 E2E алгоритме. Следовательно, У2К имеет лучшие свойства по **масштабируемости**. В правой части рис. 2 представлены отношения выборочного среднеквадратического отклонение $\hat{\sigma}$ к среднему $\bar{x}$ числу сообщений. Как видно из приведенных данных, представленные на рисунке зависимости характеризуются сравнительно малой стохастичностью (отношения $\hat{\sigma}/\bar{x}$ – единицы либо один, два десятка %), т. е. близки к функциональным. Это подтверждает справедливость выводов по результатам имитации.

Выводы

Описаны новые алгоритмы тиражирования дубликатов и взаимного исключения в П2П. Обоснована актуальность разработки и приведены модель системы, общее описание алгоритмов, их операции и свойства. Полученные результаты имитации алгоритма взаимного исключения выявили, что он обладает намного лучшей эффективностью, чем существующие алгоритмы, и хорошо масштабируем. Таким образом, алгоритм способен поддерживать большее количество узлов. Предложенные алгоритмы инвариантны к его разной сетевой реализации 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-Molina H. Canon in G major: Designing DHTS with hierarchical structure // ICDCS. – Tokyo, 2004. – P. 263–272.
4. Kumar A. Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data // IEEE Transactions on Computers. – 1991. – V. 40. – № 9. – P. 996–1004.
5. Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications // Proc. of ACM SIGCOMM. – 2001 [Электронный ресурс]. – Режим доступа: – URL: http://iptps04.cs.ucsd.edu/ papers/karger-subgroup.pdf/. – 15.05.2007.
6. FIPS 180-1, «Secure hash standard», Tech. Rep. Publication 180-1, Federal Information Processing Standard (FIPS), National Institute of Standards and Technology, US Department of Commerce, Washington, D.C., April, 1995.
7. Shiding Lin, Qiao Lian, Ming Chen, Zheng Zhang. A practical distributed mutual exclusion protocol in dynamic peer-to-peer systems // 3rd International Workshop on Peer-to-Peer Systems (IPTPS'04). – San Diego, CA, USA, Feb., 2004. – P. 1–10.
8. Muhammad M., Cheema A.S., Gupta I. Efficient mutual exclusion in peer-to-peer systems // 6th IEEE/ACM International Conference on Grid Computing. – 2005. – P. 296–299.
9. Hoye J. Freepastry [Электронный ресурс]. – Режим доступа: http://freepastry.rice.edu/FreePastry/. – 15.01.2007.
10. Rowstron A., Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems // Proceedings of Middleware, Nov. 2001 [Электронный ресурс]. – Режим доступа: research.microsoft.com/~antr/PAST/pastry.pdf. – 15.01.2007.

Поступила 03.04.2009 г.