УДК 004.02(004.75+004.021)
DOI: 10.25559/SITITO.18.202201.54-61

Научная статья

Верификация алгоритма забияки для распределенных систем средствами TLA+ и PlusCal

А. С. Поляков, Е. А. Нигодин*, Е. Е. Полупанова, П. Е. Усов
ФГБОУ ВО «Кубанский государственный университет», г. Краснодар, Российская Федерация
Адрес: 350040, Российская Федерация, Краснодарский край, г. Краснодар, ул. Ставропольская, д. 149
* apostolje@gmail.com

Аннотация
Данная работа посвящена верификации алгоритма забияки для распределенных систем средствами TLA+ и PlusCal. В работе обзорно приводится основная информация о распределенных системах, далее приводится краткая информация о распределенных алгоритмах выбора координатора, и, затем приводится подробный разбор распределенного алгоритма забияки на простом примере распределенной системы из семи узлов. Затем, в работе демонстрируется созданная модель распределенного алгоритма забияки на языках спецификации TLA+ и PlusCal и приводится описание основных частей данной модели. Далее приводится верификация созданной модели. Верификация производится при помощи инструмента TLC – это специализированное средство проверки моделей и симулятор для TLA+ спецификаций. Далее в работе продемонстрирован результат верификации. В результате верификации распределенного алгоритма забияки удалось установить, что заявленные свойства надежности и живучести выполняются в полной мере для всех возможных состояний системы.

Ключевые слова: алгоритм выбора координатора, алгоритм выбора лидера, алгоритм забияки, распределенные вычисления, формальные методы верификации, языки спецификации, темпоральные логики, проверка моделей, TLA+, PlusCal, TLC, LTL

Авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Поляков А. С., Нигодин Е. А., Полупанова Е. Е., Усов П. Е. Верификация алгоритма забияки для распределенных систем средствами TLA+ и PlusCal // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 1. С. 54-61. doi: https://doi.org/10.25559/SITITO.18.202201.54-61

Original article

Verification of the Bully Election Algorithm for Distributed Systems Using TLA+ and PlusCal

A. S. Polyakov, E. A. Nigodin*, E. E. Polupanova, P. E. Usov
Kuban State University, Krasnodar, Russian Federation
Address: 149 Stavropolskaya St., Krasnodar 350040, Russian Federation
* apostolje@gmail.com

Abstract

This article is devoted to verification of the bully election algorithm for distributed systems with TLA+ and PlusCal. In this work, we show an overview of the basic information about distributed systems, then we show definition of election algorithms for distributed systems, after that we provide a full description of the bully election algorithm for distributed systems. Later in this article, we show the model of the distributed algorithm created with TLA+ and PlusCal. Then we describe the main parts of this model. Next, we illustrate results of verification of this model. The verification was done using TLC – a model checker and simulator for executable TLA+ specifications. As a result of the verification, it was possible to establish that the declared properties of safety and liveness are fully satisfied for all possible states of the system.

Keywords: coordinator election algorithm, leader election algorithm, Bully algorithm, distributed systems, distributed computing, formal verification, specification language, temporal logic, model checking, TLA+, PlusCal, TLC, LTL

The authors declare no conflict of interest.

For citation: Polyakov A.S., Nigodin E.A., Polupanova E.E., Usov P.E. Verification of the Bully Election Algorithm for Distributed Systems Using TLA+ and PlusCal. *Sovremennye informacionnye tehnologii i IT-obrazovanie = Modern Information Technologies and IT-Education.* 2022; 18(1):54-61. doi: https://doi.org/10.25559/SITITO.18.202201.54-61

Введение

Верификация моделей распределенных программных систем и алгоритмов является одной из наиболее значимых технологий, позволяющих убедится в корректности распределенных систем и избежать ошибок при их разработке.
Объектом данного исследования является метод верификации модифицированного алгоритма забияки для распределенных вычислений с помощью средств TLA+ и PlusCal.
Перед тем, как перейти к верификации распределенного алгоритма забияки, необходимо указать терминологию распределенных вычислений и систем.

Основные термины распределенных систем

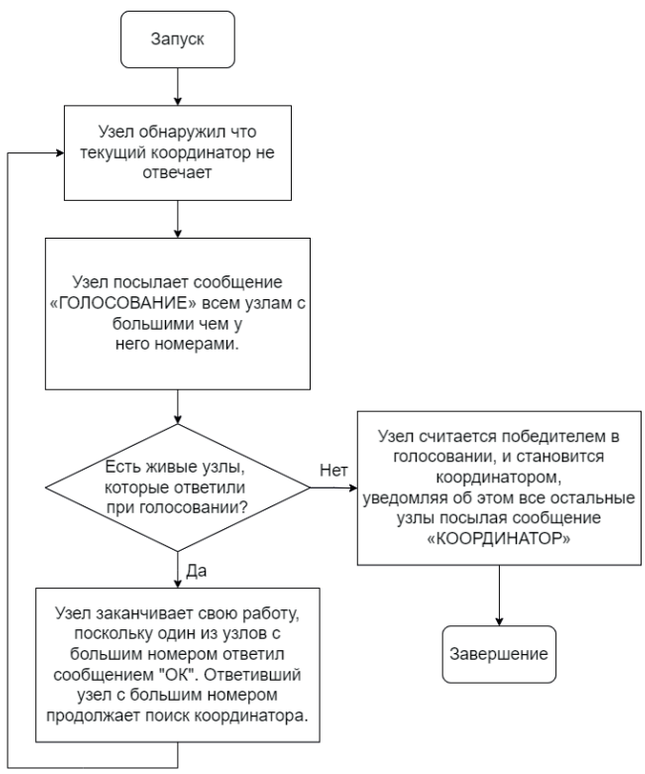
Распределённые вычисления – это способ решения трудоёмких вычислительных задач с использованием нескольких компьютеров (узлов), чаще всего объединённых в некоторую распределенную систему [1].
Распределенная система – это набор независимых (узлов), представляющийся их пользователям единой объединенной системой [2], [10; 11]. Распределенными системами являются:
- пиринговые (peer-to-peer) сетевые системы, к примеру система кооперативного обмена файлами «BitTorrent», полностью децентрализованная файлообменная сеть «Gnutella» и т.п.,
- анонимные распределенные системы: анонимная сеть виртуальных туннелей на основе луковой маршрутизации «Tor» (The Onion Router), децентрализованная компьютерная сеть «I2P» (invisible internet project) и т.п.,
- грид-вычисления и облачные вычисления, например коммерческие облачные сервисы «Amazon Web Services», «Microsoft Azure» «Google Cloud Platform» и многое другое.
Распределенные системы основаны на большом наборе технологий, который включает в себя бизнес-логику, технологи промежуточного уровня (middleware) и распределенные алгоритмы.
Алгоритмы распределенных вычислений являются важнейшими компонентами распределенных систем [3]. Часть из этих алгоритмов называют алгоритмами выбора узла-координатора, или алгоритмами голосования [4], [14].

Алгоритмы выбора координатора

Многие распределенные алгоритмы требуют, чтобы один из узлов был координатором, инициатором или выполнял другую специальную роль. Обычно не важно, какой именно узел выполняет эти специальные действия, главное, чтобы он вообще существовал [5], [16].
Если все узлы абсолютно одинаковы и не имеют отличительных характеристик, способа выбрать один из них не существует. Соответственно, будем считать, что каждый узел имеет уникальный номер, например сетевой адрес. В общем, алгоритмы голосования пытаются найти узел с максимальным номером и назначить его координатором.
Каждый узел знает номера всех остальных узлов. Чего узлы не знают, так это то, какие из них в настоящее время работают, а какие нет. Алгоритм голосования должен гарантировать, что если голосование началось, то оно, рассмотрев все узлы, решит, кто станет новым координатором. Одним из таких алгоритмов выбора координатора в распределенной системе является алгоритм забияки [18-25].
Алгоритм забияки (Bully algorithm) – это метод распределённых вычислений для динамического выбора координатора или лидера из группы распределённых вычислительных узлов [1], [6], [19]. В данном алгоритме узел с наивысшим ID среди живущих (не упавших) узлов выбирается в качестве координатора [1], [7].

Классический алгоритм забияки

На рисунке 1 приведем алгоритм забияки в виде блок-схемы:



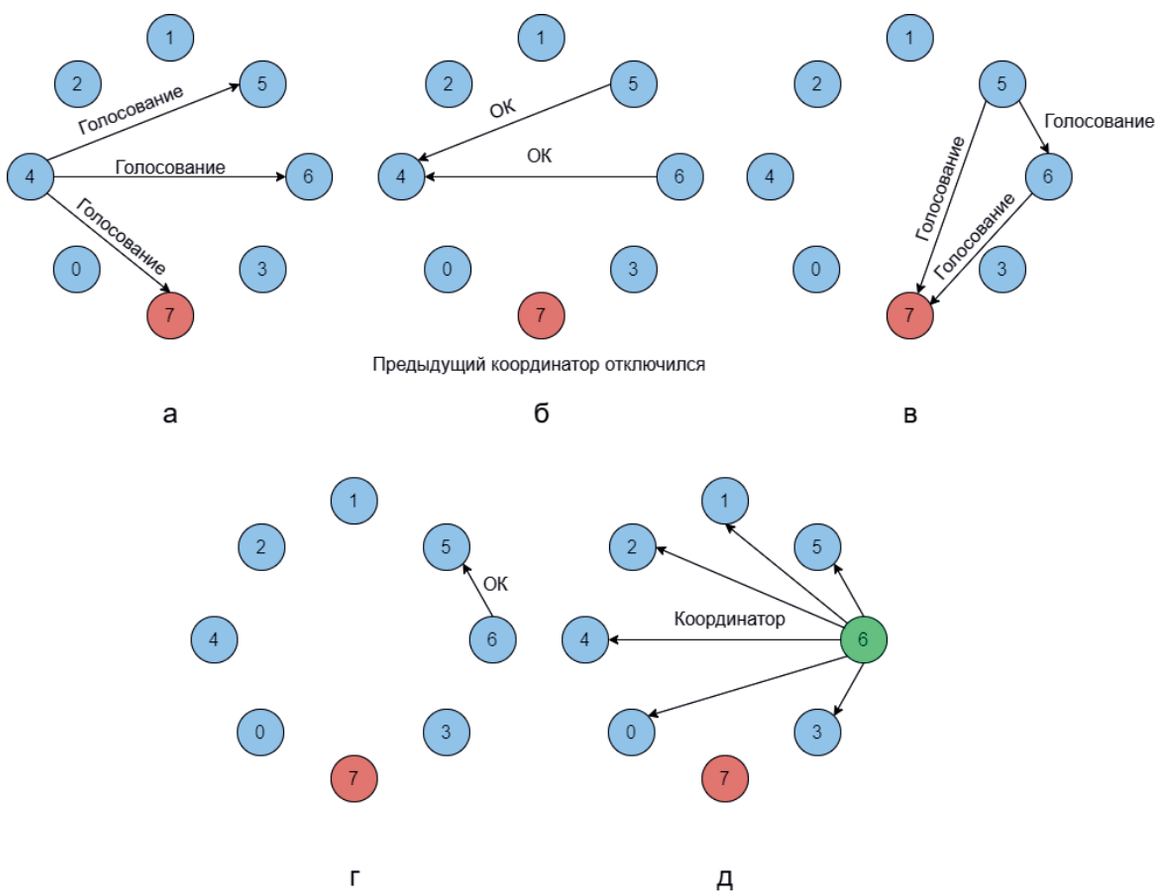
Р и с. 1. Блок-схема алгоритма забияки
F i g. 1. Block diagram of the Bully Algorithm

Когда один из узлов замечает, что координатор больше не отвечает на запросы, он инициирует голосование:
1) Узел P посылает сообщение «ГОЛОСОВАНИЕ» всем узлам с большими чем у него номерами.
2) Если нет ни одного ответа, то P считается победителем в голосовании, и становится координатором, уведомляя об этом все остальные узлы посылая сообщение «КООРДИНАТОР».
3) Если один из узлов с большим номером отвечает сообщением «ОК», то узел P заканчивает свою работу. Затем ответивший узел становится новым инициатором голосования и повторяет действие из пункта 1.
Изложенный выше алгоритм повторяется до тех пор, пока не возникнет ситуации, описанной в пункте 2 – в этом случае нет

живых узлов с большим номером. Тогда алгоритм завершает свою работу.

Другими словами, узел может в любой момент получить сообщение ГОЛОСОВАНИЕ от одного из своих коллег с меньшим номером. По получении этого сообщения получатель посылает отправителю сообщение ОК, показывая, что он работает и готов стать координатором. Затем получатель сам организует голосование. В конце концов, все узлы, кроме одного, отпадут, этот последний и будет новым координатором. Он уведомит о своей победе посылкой всем узлам сообщения, гласящего, что он новый координатор и приступает к работе. Если узел, который находился в нерабочем состоянии, начинает работать, он организует голосование. Если он оказывается узлом с самым большим из работающих узлов номером, он выигрывает голосование и берет на себя функции координатора.

На рис. 2 приведен пример работы алгоритма забияки. Группа состоит из восьми узлов, пронумерованных от 0 до 7. Ранее координатором был узел 7, но он завис. Узел 4 первым замечает это и посылает сообщение ГОЛОСОВАНИЕ всем узлам с номерами больше, чем у него, то есть узлам 5, 6 и 7, как показано на рис. 2(а). Узлы 5 и 6 отвечают ОК, как показано на рис. 2(б). После получения первого из этих ответов узел 4 понимает, координатором будет один узлов с более высоким номером. Он ожидает, кто станет победителем.



Р и с. 2. Голосование по алгоритму забияки

F i g. 2. Bully Algorithm Voting

На рис. 2(в) показано, как оба оставшихся узла, 5 и 6, продолжают голосование. Каждый посылает сообщения только тем узлам, номера у которых больше их собственных. На рис. 2(г) узел 6 сообщает узлу 5, что голосование будет вести он. В это время 6 понимает, что узел 7 мертв, а значит, победитель – он сам. Если информация о состоянии сохраняется на диске или где-то еще, откуда ее можно достать, когда с прежним координатором что-нибудь случается, узел 6 должен записать все что нужно на диск. Готовый занять свою должность узел 6 заявляет об этом путем рассылки сообщения КООРДИНАТОР всем работающим узлам как показано на рис. 2(д). Когда 4 получит это сообщение, он продолжит работу с той операции, которую пытался выполнить, когда обнаружил, что узел 7 мертв, используя теперь в качестве координатора узел 6. Таким образом, мы обошли сбой в узле 7, и работа продолжается.

Если узел 7 запустится снова, ему будет достаточно послать всем остальным сообщение КООРДИНАТОР и вынудить их подчиниться.

Верификация алгоритма забияки средствами TLA+ и PlusCal

TLA+ является языком спецификаций, используется в основном для описания и верификации последовательных и распределенных алгоритмов [8; 9], [12], [15]. TLA+ основан на теории множеств, логике первого порядка и темпоральной логике действий (TLA). В TLA+ доступна возможность автоматического доказательство теорем и многое другое.

Основным разработчиком TLA+ является Лесли Лэмпорт – исследователь теории распределённых систем, темпоральной логики и вопросов синхронизации процессов во взаимодействующих системах.

Средства TLA+ и PlusCal были выбраны для верификации алгоритма забияки ввиду их гибкости, простоты и современности с другими средствами верификации. К примеру, проверка моделей на TLA+ и PlusCal значительно проще и нагляднее, чем моделей на верификаторе SPIN.

Приведем краткое описание разработанной модели алгоритма забияки:

1) PeersAmount – число узлов в распределенной системе, их должно быть не менее двух;
2) IDS – множество идентификаторов узлов;
3) failed_leader – идентификатор отказавшего узла-координатора;
4) initiator – идентификатор узла, который раньше всех заметил, что координатор не отвечает на запросы и инициировал алгоритм выбора нового координатора;
5) *n* – число отказавших узлов помимо координатора;
6) others_who_failed – множество идентификаторов узлов, которые отказали;
7) channels – каналы связи между узлами;
8) leader – функция которая возвращает ID текущего лидера для конкретного узла.

Стоит отметить, что инициатором алгоритма выбора нового координатора может быть любой узел кроме отказавшего координатора.

С помощью LTL формулы в алгоритме забияки проверяются свойства надежности и живучести данного алгоритма [17]. На рис. 3 отображена LTL формула, используемая для верификации.

```
FailedIDS ≜ others_who_failed ∪ {failed_leader}

WorkingIDS ≜ IDS \ FailedIDS

IDThatShouldBecomeNewLeader ≜
    CHOOSE new_leader ∈ WorkingIDS :
    ∀ id ∈ WorkingIDS \ {new_leader} : new_leader > id

AllWorkingIDSAreCoordinatedByNewLeader ≜
    ∀ id ∈ WorkingIDS : leader[id] = IDThatShouldBecomeNewLeader

EventuallySolved ≜ □◇AllWorkingIDSAreCoordinatedByNewLeader
```

Р и с. 3. LTL формула, используемая в модели алгоритма забияки
F i g. 3. LTL formula used in the Bully Algorithm Model

На рис. 4, 5, 6 приведена формальная спецификация модели алгоритма забияки написанная на TLA+ и PlusCal. Для удобства она разбита на три части. Соответственно, на рис. 4 показана первая часть модели, на рис. 5 вторая, а на рис. 6 третья.

```
MODULE bully
EXTENDS TLC, Integers, FiniteSets, Randomization

CONSTANT PeersAmount

ASSUME PeersAmount ∈ Nat \ {0, 1}

IDS ≜ 1 .. PeersAmount

  --algorithm bully
variables
    failed_leader = PeersAmount,

    initiator ∈ IDS \ {failed_leader},

    n ∈ 0 .. Cardinality(IDS \ {failed_leader, initiator}),
    others_who_failed = RandomSubset(n, IDS \ {failed_leader, initiator}),

    channels = [sender ∈ IDS ↦ [receiver ∈ IDS \ {sender} ↦ ""]],

    leader = [id ∈ IDS ↦ failed_leader] ;

define
    IDSBiggerThan ≜ [id_1 ∈ IDS ↦ {id_2 ∈ IDS : id_2 > id_1}]

    IDSSmallerThan ≜ [id_1 ∈ IDS ↦ {id_2 ∈ IDS : id_2 < id_1}]

    IDSBiggerThanExceptFailedLeader ≜
        [id ∈ IDS ↦ IDSBiggerThan[id] \ {failed_leader}]

    DoesNotReceiveAnyResponse(id) ≜
        IDSBiggerThanExceptFailedLeader[id] \ others_who_failed = {}

    NewLeaders(receiver) ≜
        {sender ∈ IDS \ {receiver} : channels[sender][receiver] = "Leader"}

    DoesNotReceiveOKResponseFromNewLeaders(receiver) ≜
        LET old_leader ≜ leader[receiver] IN
        ∃ new_leader ∈ IDSBiggerThanExceptFailedLeader[receiver] :
        ∧ new_leader ∉ others_who_failed
        ∧ IF old_leader = failed_leader THEN new_leader > old_leader ELSE TRUE

    ElectionInitiators(receiver) ≜
        {sender ∈ IDS \ {receiver} : channels[sender][receiver] = "Election"}

    MessageSenders(receiver) ≜
        {sender ∈ IDS \ {receiver} : channels[sender][receiver] ≠ ""}

    FailedIDS ≜ others_who_failed ∪ {failed_leader}

    WorkingIDS ≜ IDS \ FailedIDS
```

Р и с. 4. Модель алгоритма забияки, часть 1
F i g. 4. Bully Algorithm Model, Part 1

```
    IDThatShouldBecomeNewLeader ≜
        CHOOSE new_leader ∈ WorkingIDS :
        ∀ id ∈ WorkingIDS \ {new_leader} : new_leader > id

    AllWorkingIDSAreCoordinatedByNewLeader ≜
        ∀ id ∈ WorkingIDS : leader[id] = IDThatShouldBecomeNewLeader

    EventuallySolved ≜ □◇AllWorkingIDSAreCoordinatedByNewLeader
end define ;

fair process Peer ∈ IDS
begin
Initialize:
    if self ∈ FailedIDS then
        goto Failed ;
    elsif self = initiator then
        goto BecomeLeaderOrStartElection ;
    else
        goto NormalExecution ;
    end if ;

BecomeLeaderOrStartElection:
    if IDSBiggerThanExceptFailedLeader[self] = {} then
        leader[self] := self ||
        channels[self] :=
        [receiver ∈ DOMAIN channels[self] ↦
            IF receiver ∈ IDSSmallerThan[self] THEN "Leader" ELSE ""] ;
```

```
      goto NormalExecution;
    else
      channels[self] :=
      [receiver ∈ DOMAIN channels[self] ↦
        IF receiver ∈ IDSBiggerThan[self] THEN "Election" ELSE ""];
    end if;

CheckElectionTimeout:
    if DoesNotReceiveAnyResponse(self) then
      leader[self] := self ||
      channels[self] :=
      [receiver ∈ DOMAIN channels[self] ↦
        IF receiver ∈ IDSSmallerThan[self] THEN "Leader" ELSE ""];
      goto NormalExecution;
    end if;

CheckOkTimeout:
    if DoesNotReceiveOKResponseFromNewLeaders(self) then
      goto NormalExecution;
```

Р и с. 5. Модель алгоритма забияки, часть 2

F i g. 5. Bully Algorithm Model, Part 2

```
    end if;
AcceptNewLeader:
    with new_leader ∈ NewLeaders(self) do
      leader[self] := new_leader ||
      channels[new_leader][self] := "";
      goto CheckOkTimeout;
    end with;

NormalExecution:
    with sender ∈ MessageSenders(self) do
      if channels[sender][self] = "Election" then
        channels[self][sender] := "OK" ||
        channels[sender][self] := "";
        goto BecomeLeaderOrStartElection;
      elsif channels[sender][self] = "Leader" then
        leader[self] := sender ||
        channels[sender][self] := "";
        goto NormalExecution;
      else
        channels[sender][self] := "";
      end if;
    end with;

Failed:
    skip;
end process;

end algorithm
```

Р и с. 6. Модель алгоритма забияки, часть 3

F i g. 6. Bully Algorithm Model, Part 3

В результате верификации, проиллюстрированной выше модели, проверены свойства надежности и живучести алгоритма.

На рис. 7 проиллюстрирован результат верификации модели при помощи специального средства для проверки моделей TLC (TLC также является симулятором для TLA+ спецификаций) для 5 узлов с заданными проверяемыми LTL – свойствами (они были отображены на рисунке 3), приведено общее число состояний и число уникальных состояний для каждого действия в модели, время верификации модели [13]. Сообщение «Success» на данной иллюстрации говорит о том, что все свойства модели, заданные при помощи LTL, успешно выполняются.

TLA+ model checking

Status Check again Full output

Checking bully.tla / bully.cfg

Success : Fingerprint collision probability: 2.8E-9

Start: 15:57:18 (Jan 8), end: 15:57:46 (Jan 8)

States

Time	Diameter	Found	Distinct	Queue
00:00:00	0	16	16	16

Coverage

Module	Action	Total	Distinct
bully	Init	16	16
bully	Initialize	53 158	43 920
bully	BecomeLeaderOrStartElection	84 987	8 919
bully	CheckElectionTimeout	62 058	1 778
bully	CheckOkTimeout	96 693	4 624
bully	AcceptNewLeader	28 098	2 883
bully	NormalExecution	111 672	20 606
bully	Failed	88 999	46 335

Р и с. 7. Результат верификации модели алгоритма забияки

F i g. 7. The result of the verification of the Bully Algorithm Model

На рис. 8 показана конфигурация проверяемой модели.

bully.cfg

```
1  SPECIFICATION Spec
2  \* Add statements after this line.
3
4  CHECK_DEADLOCK FALSE
5
6  CONSTANTS
7      PeersAmount = 5
8
9  PROPERTIES
10     EventuallySolved
11
```

Р и с. 8. Конфигурация проверяемой модели

F i g. 8. The configuration of the tested model

Разработка модели велась с использованием системы контроля версий git[1].

Заключение

В ходе проведенной работы была реализована модель распределенного алгоритма забияки средствами TLA+ и PlusCal. Полученная модель была верифицирована для проверки свойств живучести и надежности с использованием LTL свойств.

В результате верификации свойств надежности и живучести распределенного алгоритма забияки удалось выяснить, что данные свойства выполняются в полной мере для всех возможных состояний системы.

[1] Ссылка на репозиторий: Polyakov A. Bully election algorithm model written in TLA+ and PlusCal [Электронный ресурс] // GitHub, Inc., 2022. URL: https://github.com/polikow/bully_election (дата обращения: 26.01.2022).

References

[1] Tanenbaum A.S., van Steen M. Distributed Systems: Principles and Paradigms. 2nd ed. Prentice-Hall, Inc., USA; 2006. 704 p. (In Eng.)
[2] Akhtar S., Zahoor E. Formal Specification and Verification of MQTT Protocol in PlusCal-2. *Wireless Personal Communications*. 2021; 119(2):1589-1606. (In Eng.) doi: https://doi.org/10.1007/s11277-021-08296-4
[3] Shkarupylo V.V., Blinov I.V., Chemeris A.A., Dusheba V.V., Alsayaydeh J.A.J. On Applicability of Model Checking Technique in Power Systems and Electric Power Industry. In: Zaporozhets A. (ed). *Systems, Decision and Control in Energy III. Studies in Systems, Decision and Control.* Vol. 399. Springer, Cham; 2022. p. 3-21. (In Eng.) doi: https://doi.org/10.1007/978-3-030-87675-3_1
[4] Akhtar S., Zahoor E., Perrin O. Formal Verification of Authorization Policies for Enterprise Social Networks Using PlusCal-2. In: Romdhani I., Shu L., Takahiro H., Zhou Z., Gordon T., Zeng D. (eds.). *Collaborative Computing: Networking, Applications and Worksharing. CollaborateCom 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Vol. 252. Springer, Cham; 2018. p. 530-540. (In Eng.) doi: https://doi.org/10.1007/978-3-030-00916-8_49
[5] Resch S., Paulitsch M. Using TLA+ in the Development of a Safety-Critical Fault-Tolerant Middleware. *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW).* IEEE Press, Toulouse, France; 2017. p. 146-152. (In Eng.) doi: https://doi.org/10.1109/ISSREW.2017.43
[6] Lamport L. The PlusCal Algorithm Language. In: Leucker M., Morgan C. (eds.). *Theoretical Aspects of Computing – ICTAC 2009. ICTAC 2009. Lecture Notes in Computer Science.* Vol. 5684. Springer, Berlin, Heidelberg; 2009. p. 36-60. (In Eng.) doi: https://doi.org/10.1007/978-3-642-03466-4_2
[7] Akhtar S., Merz S., Quinson M. A High-Level Language for Modeling Algorithms and Their Properties. In: Davies J., Silva L., Simao A. (eds.). *Formal Methods: Foundations and Applications. SBMF 2010. Lecture Notes in Computer Science.* Vol. 6527. Springer, Berlin, Heidelberg; 2011. p. 49-63. (In Eng.) doi: https://doi.org/10.1007/978-3-642-19829-8_4
[8] Selvaratnam D., Cantoni M., Davoren J.M., Shames I. Sampling polynomial trajectories for LTL verification. *Theoretical Computer Science*. 2022; 897:135-163. (In Eng.) doi: https://doi.org/10.1016/j.tcs.2021.10.024
[9] Rubio R., Martí-Oliet N., Pita I., Verdejo A. Model checking strategy-controlled systems in rewriting logic. *Automated Software Engineering*. 2022; 29(1). (In Eng.) doi: https://doi.org/10.1007/s10515-021-00307-9
[10] Kovatsch M. CoAP for the web of things: From tiny resource-constrained devices to the web browser. *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication. UbiComp'13 Adjunct.* New York, NY: ACM; 2013. p. 1495-1504. (In Eng.) doi: https://doi.org/10.1145/2494091.2497583
[11] Zahoor E., Ikram A., Akhtar S., Perrin O. Authorization Policies Specification and Consistency Management within Multi-cloud Environments. In: Gruschka N. (ed). *Secure IT Systems. NordSec 2018. Lecture Notes in Computer Science*. Vol. 11252. Springer, Cham; 2018. p. 272-288. (In Eng.) doi: https://doi.org/10.1007/978-3-030-03638-6_17
[12] Kuppe M.A., Lamport L., Ricketts D. The TLA+ Toolbox. *5th Workshop on Formal Integrated Development Environment, F-IDE 2019. EPTCS 310.* Porto, Portugal; 2019. p. 50-62. (In Eng.) doi: https://doi.org/10.4204/EPTCS.310.6
[13] Shkarupylo V.V., Tomičić I., Kasian K.M. The investigation of TLC model checker properties. *Journal of Information and Organizational Sciences*. 2016; 40(1):145-152. (In Eng.) doi: https://doi.org/10.31341/jios.40.1.7
[14] Park J.S., Sandhu R., Ahn G.-J. Role-based access control on the web. *ACM Transactions on Information and System Security*. 2001; 4(1):37-71. (In Eng.) doi: https://doi.org/10.1145/383775.383777
[15] Yu Y., Manolios P., Lamport L. Model checking TLA+ specifications. In: Pierre L., Kropf T. (eds.). *CHARME 1999. Lecture Notes in Networks and Systems*. Vol. 1703. Springer, Heidelberg; 1999. p. 54-66. (In Eng.) doi: https://doi.org/10.1007/3-540-48153-2_6
[16] Dutertre B., Easwaran A., Hall B., Steiner W. Model-based analysis of Timed-Triggered Ethernet. *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC).* IEEE Press, Williamsburg, VA, USA; 2012. p. 9D2-1-9D2-11. (In Eng.) doi: https://doi.org/10.1109/DASC.2012.6382445
[17] Bauer A., Leucker M., Schallhart C. Runtime Verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*. 2011; 20(4):14. (In Eng.) doi: https://doi.org/10.1145/2000799.2000800
[18] Arghavani A., Ahmadi E., Haghighat A.T. Improved bully election algorithm in distributed systems. *ICIMU 2011: Proceedings of the 5th international Conference on Information Technology & Multimedia*. IEEE Press, Kuala Lumpur, Malaysia; 2011. p. 1-6. (In Eng.) doi: https://doi.org/10.1109/ICIMU.2011.6122724
[19] Numan M., Subhan F., Khan W.Z., Assiri B., Armi N. Well-Organized Bully Leader Election Algorithm for Distributed System. *2018 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET).* IEEE Press, Serpong, Indonesia; 2018. p. 5-10. (In Eng.) doi: https://doi.org/10.1109/ICRAMET.2018.8683916
[20] Soundarabai P.B., Sahai R., Thriveni J., Venugopal K.R., Patnaik L.M. Improved Bully Election Algorithm for Distributed Systems. arXiv:1403.3255. 2014. Available at: https://arxiv.org/abs/1403.3255 (accessed 26.01.2022). (In Eng.)
[21] Gholipour M., Kordafshari M.S., Jahanshahi M., Rahmani A.M. A New Approach for Election Algorithm in Distributed Systems. *2009 Second International Conference on Communication Theory, Reliability, and Quality of Service.* IEEE Press, Colmar, France; 2009. p. 70-74. (In Eng.) doi: https://doi.org/10.1109/CTRQ.2009.32
[22] Park S.H. A Probabilistically Correct Election Protocol in Asynchronous Distributed Systems. In: Zhou X., Xu M., Jähnichen S., Cao J. (eds.). *Advanced Parallel Processing Technologies. APPT 2003. Lecture Notes in Computer Science.* Vol. 2834. Springer, Berlin, Heidelberg; 2003. p. 177-185. (In Eng.) doi: https://doi.org/10.1007/978-3-540-39425-9_23

[23] Dolev D. A simple model for agreement in distributed systems. In: Simons B., Spector A. (eds.). *Fault-Tolerant Distributed Computing. Lecture Notes in Computer Science.* Vol. 448. Springer, New York, NY; 1990. p. 42-50. (In Eng.) doi: https://doi.org/10.1007/BFb0042324

[24] EffatParvar M., Yazdani N., EffatParvar M., Dadlani A., Khonsari A. Improved algorithms for leader election in distributed systems. *2010 2nd International Conference on Computer Engineering and Technology*. IEEE Press, Chengdu, China; 2010. p. V2-6-V2-10. (In Eng.) doi: https://doi.org/10.1109/ICCET.2010.5485357

[25] Johansson B., Rågberger M., Papadopoulos A.V., Nolte T. Heartbeat Bully: Failure Detection and Redundancy Role Selection for Network-Centric Controller. *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society.* IEEE Press, Singapore; 2020. p. 2126-2133. (In Eng.) doi: https://doi.org/10.1109/IECON43393.2020.9254494

Поступила 26.01.2022; одобрена после рецензирования 25.02.2022; принята к публикации 05.03.2022.
Submitted 26.01.2022; approved after reviewing 25.02.2022; accepted for publication 05.03.2022.

Об авторах:

Поляков Алексей Сергеевич, магистрант кафедры вычислительных технологий, факультет компьютерных технологий и прикладной математики, ФГБОУ ВО «Кубанский государственный университет» (350040, Российская Федерация, Краснодарский край, г. Краснодар, ул. Ставропольская, д. 149), **ORCID: https://orcid.org/0000-0001-6765-7769**, superpolikow@gmail.com

Нигодин Елисей Алексеевич, магистрант кафедры вычислительных технологий, факультет компьютерных технологий и прикладной математики, ФГБОУ ВО «Кубанский государственный университет» (350040, Российская Федерация, Краснодарский край, г. Краснодар, ул. Ставропольская, д. 149), **ORCID: https://orcid.org/0000-0002-0898-7335,** apostolje@gmail.com

Полупанова Елена Евгеньевна, доцент кафедры вычислительных технологий, факультет компьютерных технологий и прикладной математики, ФГБОУ ВО «Кубанский государственный университет» (350040, Российская Федерация, Краснодарский край, г. Краснодар, ул. Ставропольская, д. 149), кандидат технических наук, **ORCID: https://orcid.org/0000-0002-0364-1132**, jienka@mail.ru

Усов Павел Евгеньевич, магистрант кафедры вычислительных технологий, факультет компьютерных технологий и прикладной математики, ФГБОУ ВО «Кубанский государственный университет» (350040, Российская Федерация, Краснодарский край, г. Краснодар, ул. Ставропольская, д. 149), **ORCID: https://orcid.org/0000-0003-3774-4903,** lyova-pavel.usov@yandex.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Aleksey S. Polyakov, Master degree student of the Chair of Computational Technologies, Faculty of Computer Technologies and Applied Mathematics, Kuban State University (149 Stavropolskaya St., Krasnodar 350040, Russian Federation), **ORCID: https://orcid.org/0000-0001-6765-7769**, superpolikow@gmail.com

Elisey A. Nigodin, Master degree student of the Chair of Computational Technologies, Faculty of Computer Technologies and Applied Mathematics, Kuban State University (149 Stavropolskaya St., Krasnodar 350040, Russian Federation), **ORCID: https://orcid.org/0000-0002-0898-7335,** apostolje@gmail.com

Elena E. Polupanova, Associate Professor of the Chair of Computational Technologies, Faculty of Computer Technologies and Applied Mathematics, Kuban State University (149 Stavropolskaya St., Krasnodar 350040, Russian Federation), Cand.Sci. (Tech.), **ORCID: https://orcid.org/0000-0002-0364-1132**, jienka@mail.ru

Pavel E. Usov, Master degree student of the Chair of Computational Technologies, Faculty of Computer Technologies and Applied Mathematics, Kuban State University (149 Stavropolskaya St., Krasnodar 350040, Russian Federation), **ORCID: https://orcid.org/0000-0003-3774-4903,** lyova-pavel.usov@yandex.ru

All authors have read and approved the final manuscript.