УДК 004.75
DOI 10.21685/2072-3059-2018-4-4

В. Н. Дубинин, И. А. Беззатеев, А. С. Войнов, И. В. Сенокосов

МОДЕЛИРОВАНИЕ И РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННЫХ СЕМАФОРОВ В АРХИТЕКТУРЕ IEC 61499 НА ОСНОВЕ ПРОТОКОЛОВ ДЛЯ ДОСТИЖЕНИЯ КОНСЕНСУСА В СЕТИ НЕНАДЕЖНЫХ ПРОЦЕССОВ

Аннотация.

Актуальность и цели. Международный стандарт IEC 61499, предназначенный для проектирования распределенных компонентно-базированных систем мониторинга и управления промышленными процессами, не позволяет напрямую строить эти системы со сложными видами синхронизации и взаимодействий, большинство из которых основано на концепции семафора. Целью представленной работы является моделирование, верификация и сравнительный анализ производительности распределенных семафоров на основе протоколов *Paxos* и *Raft*, а также их функционально-блочная реализация на основе стандарта IEC 61499.

Материалы и методы. Для построения распределенных семафоров использовались протоколы *Paxos* и *Raft* для достижения консенсуса в сети ненадежных процессоров. Для их формального описания и моделирования выбраны раскрашенные сети Петри, положенные в основу инструментальной системы моделирования *CPN Tools*. Реализация распределенных семафоров производилась на основе функциональных блоков стандарта IEC 61499 в среде *NxtStudio*.

Результаты. Разработаны сетевые модели алгоритмов выполнения операций открытия и закрытия распределенных семафоров (основанных на алгоритмах *Paxos* и *Raft*), которые реализованы в системе *CPN Tools* и позволяют производить как оценку производительности системы, так и ее верификацию. Проведены имитационные эксперименты с этими сетевыми моделями, дан сравнительный анализ подходов на основе *Paxos* и *Raft* и рекомендации по их использованию. Разработана библиотека функциональных блоков для реализации протоколов достижения консенсуса *Paxos* и *Raft*, а также распределенных семафоров на их основе, позволяющая существенно расширить возможности проектирования распределенных управляющих приложений со сложными видами взаимодействий в архитектуре IEC 61499.

Выводы. Результаты имитационных экспериментов и программная реализация показали работоспособность и корректность функционирования распределенных семафоров на основе алгоритмов для достижения консенсуса. Использование протокола *Raft* является предпочтительным как с точки зрения простоты реализации и масштабируемости, так и надежности.

Ключевые слова: распределенная управляющая система, синхронизация, коммуникационный протокол, *Paxos*, *Raft*, раскрашенные сети Петри, функциональный блок.

© Дубинин В. Н., Беззатеев И. А., Войнов А. С., Сенокосов И. В., 2018. Данная статья доступна по условиям всемирной лицензии Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), которая дает разрешение на неограниченное использование, копирование на любые носители при условии указания авторства, источника и ссылки на лицензию Creative Commons, а также изменений, если таковые имеют место.

V. N. Dubinin, I. A. Bezzateev, A. S. Voynov, I. V. Senokosov

MODELLING AND IMPLEMENTATION OF DISTRIBUTED SEMAPHORES IN THE IEC 61499 ARCHITECTURE ON THE BASIS OF PROTOCOLS FOR SOLVING CONSENSUS IN A NETWORK OF UNRELIABLE PROCESSES

Abstract.

Background. The international standard IEC 61499, intended for the design of distributed component-based industrial process measurement and control systems, does not allow the direct building of ones with complex types of synchronization and interactions, most of which are based on the concept of a semaphore. The purpose of the presented work is modeling, verification and comparative analysis of the performance of distributed semaphores based on the Paxos and Raft protocols, as well as their IEC 61499 function block-based implementation.

Materials and methods. To design distributed semaphores, Paxos and Raft protocols for solving consensus in a network of unreliable processors were used. For their formal description, modeling and simulation, the colored Petri nets and the supporting CPN Tools were chosen. The distributed semaphores were implemented on the basis of IEC 61499 function blocks in the NxtStudio.

Results. 1) Net models of algorithms for performing decrementing/incrementing distributed semaphores based on Paxos and Raft were developed in the CPN Tools allowing to conduct both performance evaluation and verification; 2) simulation experiments were carried out using these net models, comparative analysis of the approaches based on Paxos and Raft, as well as recommendations for their use were given; 3) a library of function blocks for implementation of Paxos and Raft, as well as distributed semaphores based on them were developed, which makes it possible to essentially expand capabilities for design of distributed control applications with complex interactions in the IEC 61499 architecture.

Conclusions. The results of the simulation experiments and software implementation showed the operability and correctness of the functioning of distributed semaphores on the basis of the two protocols for solving consensus. The use of the Raft protocol is preferred both in terms of ease of implementation and scalability, as well as reliability.

Keywords: distributed control system, synchronization, communication protocol, Paxos, Raft, coloured Petri nets, function block.

Введение

Для построения распределенных реконфигурируемых компонентно-базированных систем управления технологическим оборудованием был разработан международный стандарт IEC 61499 [1]. Данный стандарт привлек пристальное внимание исследователей, поскольку предлагал четкий путь перехода от устаревших централизованных систем управления к распределенным системам нового поколения на основе множества контроллеров, взаимодействующих в сети. Основным артефактом проектирования в стандарте является функциональный блок (ФБ), управляемый событиями. Для взаимодействия между ресурсами и устройствами системы в данном стандарте используется механизм передачи сообщений посредством коммуникационных ФБ CLIENT/SERVER и PUBLISH/SUBSCRIBE. Данный механизм является единственным и довольно низкоуровневым, что не позволяет напрямую строить

распределенные системы управления со сложными видами синхронизации и взаимодействий, включая «критические секции», «рандеву», «производитель–потребитель», «читатели–писатели» и т.д., которые во многом основаны на использовании концепции семафора [2].

Семафор – это особый тип разделяемой переменной, которая обрабатывается только двумя неделимыми операциями – *P* и *V* (закрытие и открытие семафора). Простой, но в то же время показательный пример использования семафора в управляющих системах – это взаимоисключающая работа нескольких устройств IEC 61499 с общим (разделяемым) технологическим оборудованием, данными или коммуникационной сетью.

Существует семантический разрыв между классической концепцией семафора, ориентированной на общую память, и его реализацией в распределенной системе. В системах с разделяемой общей памятью реализация семафора обычно основывается на машинной команде *Test&Set*. В работе [3] предлагается описание взаимного исключения в виде недетерминированного автомата и его реализация на программируемых логических интегральных схемах. Можно выделить два подхода к реализации семафоров в распределенной системе – централизованный [4, 5] и децентрализованный. В работе [4] была предложена клиент-серверная реализация семафора на уровне приложений для сети рабочих станций, работающих под *Unix*. В работе [5] похожий подход был применен в архитектуре IEC 61499. Однако в данных случаях семафор реализован как сервер, что фактически является централизованным решением, имеющим недостатки, присущие всем централизованным системам – низкую надежность и производительность, плохую масштабируемость. В работе [2] приводится децентрализованное решение, основанное на широковещательной рассылке сообщений. Каждый процесс имеет локальную переменную для представления значения семафора. Синхронность операций над семафором в каждом процессе обеспечивается строгой упорядоченностью запросов (сообщений) от процессов на операцию над семафором по меткам времени. Ограничением метода является предположение о надежной передаче сообщений. В работе [6] рассматривается децентрализованное управление распределенными данными на основе протокола двухфазной блокировки, используемого в распределенных базах данных. На основе данного метода предлагаются синхропримитивы для работы с разделяемыми ресурсами. Недостатком данного метода является длительное выполнение операций над семафорами. Каждая операция по изменению значения семафора требует нескольких фаз, каждая из которых предполагает передачу *N* сообщений, где *N* – число узлов в сети. В работе [7] для организации распределенного семафора требуется кластеризация узлов системы в двухуровневую структуру, причем размер кластера зависит от значения семафора. К недостаткам метода можно отнести его сложность и плохую масштабируемость.

В нашей работе мы будет использовать более современный подход на основе применения алгоритмов (протоколов) для решения задач консенсуса в сети ненадежных процессов – *Paxos* [8] и *Raft* [9]. Консенсус – процесс получения согласованного результата группой участников, основная проблема – ненадежная среда передачи и сами процессы. Протоколы данного типа в последнее время стали широко использоваться крупнейшими *IT*-компаниями, например *Google*, *Microsoft* и *Facebook*. Следует особо отметить, что в распределенном сервисе конфигурирования и синхронизации *Apache Zookeeper*

[10] механизм распределенного семафора реализован на основе протокола *Zab*, который близок к протоколу *Paxos*.

Целью представленной работы является моделирование, верификация и сравнительный анализ производительности распределенных семафоров на основе протоколов *Paxos* и *Raft*, а также их функционально-блочная реализация на основе стандарта IEC 61499.

1. Подход к реализации распределенных семафоров на основе протоколов для достижения консенсуса

Протоколы семейства *Paxos* гарантируют три следующих показателя: нетривиальность, консистентность и живучесть. В протоколе *Paxos* узлы распределенной системы имеют следующие роли: *Client*, *Proposer*, *Acceptor*, *Learner* [8]. Протокол *Raft* разрабатывался с учетом недостатков предыдущего протокола *Paxos*. При выборе ключевых идей предпочтение отдавалось более простым и практичным решениям. Тем не менее, несмотря на относительную простоту, *Raft* обеспечивает безопасную и эффективную реализацию машины состояний поверх кластерной вычислительной системы. К особенностям *Raft* относятся:

1) четкое разделение фаз;
2) наличие явно выделенного лидера;
3) возможность динамического изменения размера кластера [9].

Ниже предлагается подход к реализации распределенных семафоров на основе протоколов *Paxos* и *Raft*. При разработке распределенного семафора основная задача заключается в обеспечении идентичного состояния семафора на всех узлах распределенной системы. При использовании протокола *Paxos* считаем, что задача, решаемая его компонентами, состоит в определении текущего состояния семафора (после запроса на его изменение), которое должно быть идентично для всех узлов сети, на которых располагаются компоненты *Paxos*, запоминающие принятое решение. В классическом *Paxos* принятое решение запоминает только компонент *Learner*. Для реализации распределенного семафора на основе *Paxos* предлагается запоминать принятое решение также в компоненте *Proposer*. Такой подход позволяет снизить время принятия решения распределенной системы по запросам, которые не могут быть выполнены (например, запрос на захват семафора, который уже занят другим пользователем). Компонент *Proposer*, зная текущее состояние семафора, может сразу проанализировать возможность выполнения полученного запроса. Если был получен запрос на захват семафора, а он уже занят другим пользователем, то компонент *Proposer* сразу отправляет отрицательный ответ на полученный запрос, не задействуя компоненты *Acceptor* и *Learner*, тем самым уменьшая время ответа на такие запросы.

Протокол *Raft* изначально учитывает в себе вышеперечисленные проблемы, возникающие на пути использования протокола *Paxos*. Компоненты *Raft* объединяют в себе упрощенный функционал компонентов *Paxos*, сосредотачиваясь на задачах репликации данных (состояния семафора) в распределенной системе и действиях при выходе из строя компонентов лидера (в *Paxos* был *Proposer*) и ведомых компонентов. При реализации распределенного семафора с помощью протокола *Raft* в качестве данных, подлежащих репликации на все необходимые компоненты распределенной системы, выступают данные о состоянии семафора. Решение о состоянии семафора будет

принимать только компонент-лидер, на остальные компоненты принятое решение (текущее состояние семафора) будет реплицироваться. Компоненты, ведомые лидером, содержат в себе также функционал компонента-лидера, чтобы в случае аппаратного отказа лидера, занять его место. В каждом компоненте необходим функционал, реализующий опрос компонента-лидера ведомыми компонентами при отсутствии запросов в течение заданного времени (по таймеру).

2. Моделирование распределенных семафоров с использованием временных раскрашенных сетей Петри

Наиболее подходящим методом исследования сложных параллельных систем является имитационное моделирование. В рамках этого метода с точки зрения формализации наиболее подходящими являются модели на основе сетей Петри, позволяющие описывать параллельные процессы со сложными видами взаимодействий. Поэтому для формального описания и моделирования распределенных семафоров на основе протоколов *Paxos* и *Raft* были выбраны временные цветные (раскрашенные) сети Петри [11], положенные в основу инструментальной системы моделирования *CPN Tools* [12].

Имитационная модель семафора должна обеспечивать следующий функционал:

а) генерацию запросов на захват или освобождение семафора от нескольких клиентов;

б) обработку и анализ запросов клиентов в соответствии с протоколом;

в) отправку клиенту ответа на его запрос;

г) сбор статистических данных при проведении моделирования.

Ввиду того что сетевые модели *Paxos* и *Raft* довольно громоздкие, их описание приводится ниже лишь в ограниченном объеме.

Разработанная сетевая модель системы «Распределенный семафор на основе протокола *Paxos*» имеет иерархическую структуру и содержит в себе следующие подсети:

– *System* – сеть самого верхнего уровня;

– *Client* 1 и *Client* 2 – подсети, имитирующие генерацию запросов на открытие/закрытие семафора от соответствующих клиентов и прием ответов;

– *SubLearner* – подсеть, которая обрабатывает пакеты, приходящие от клиентов, анализирует их и выносит решение об отправке пакета в подсеть *Proposer* или отправляет клиенту ответ о невозможности выполнения его запроса;

– *Proposer*, *Acceptor* 1, *Acceptor* 2, *Acceptor* 3, *Learner* – подсети, имитирующие работы соответствующих ролей протокола *Paxos*.

Сеть *System*, фрагменты которой представлены на рис. 1 и 2, содержит в себе две подсети *Client*, одну подсеть *SubLearner*, одну подсеть *Proposer*, три подсети *Acceptor* и одну подсеть *Learner*. Содержимое макропереходов, входящих в эти сети, не представлено ввиду ограниченности объема данной статьи. В случае использования протокола *Raft* сеть *System* содержит в себе две подсети *Client*, одну подсеть *Leader_RAFT* и три подсети *Sub_RAFT*.

С разработанными сетевыми моделями была проведена серия имитационных экспериментов, в ходе которых верифицировалась правильность функционирования систем, а также собирались статистические данные.

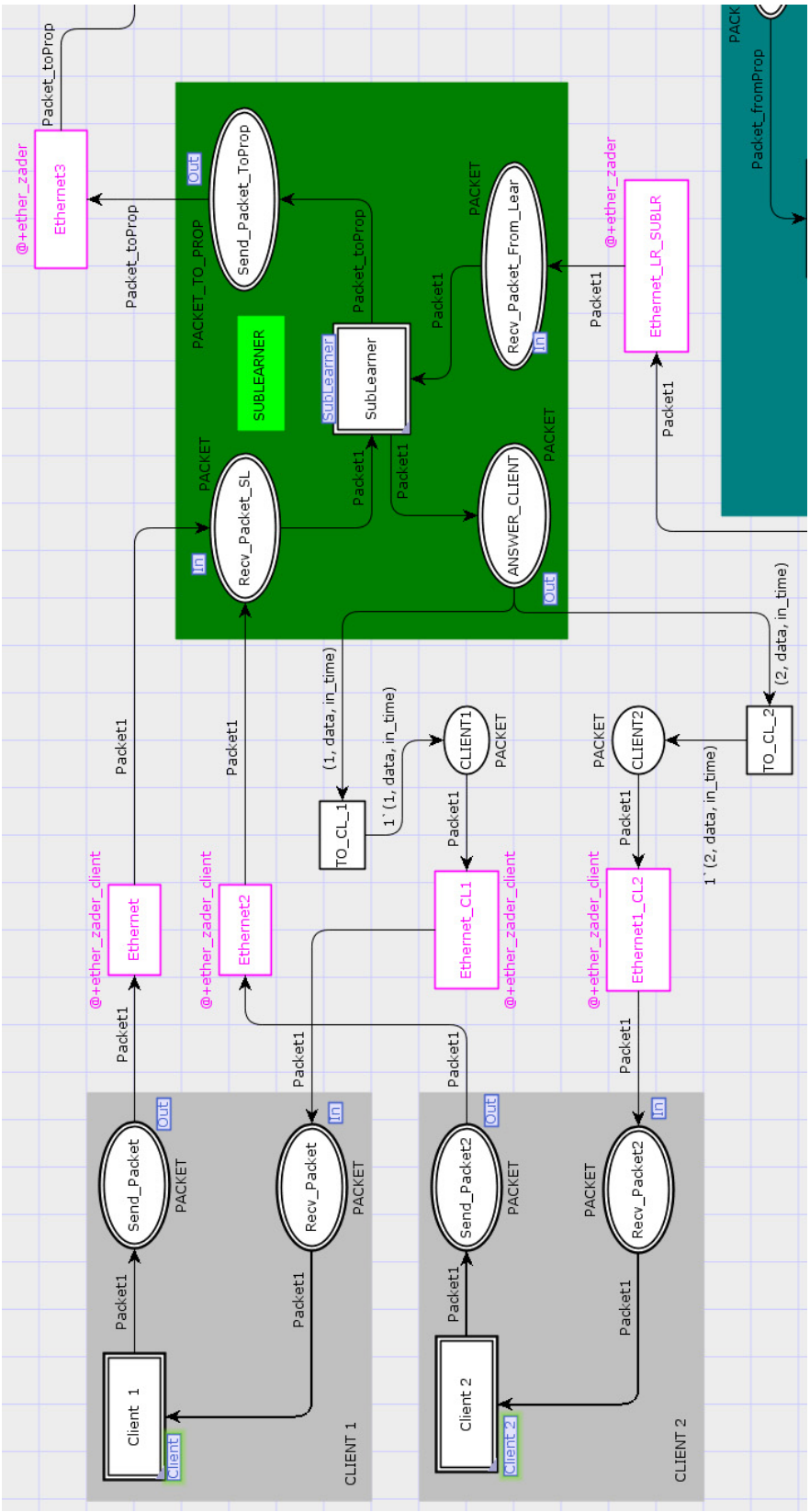

Рис. 1. Фрагмент сетевой модели верхнего уровня, содержащий подсети *Client* 1, *Client* 2 и *Sublearner*

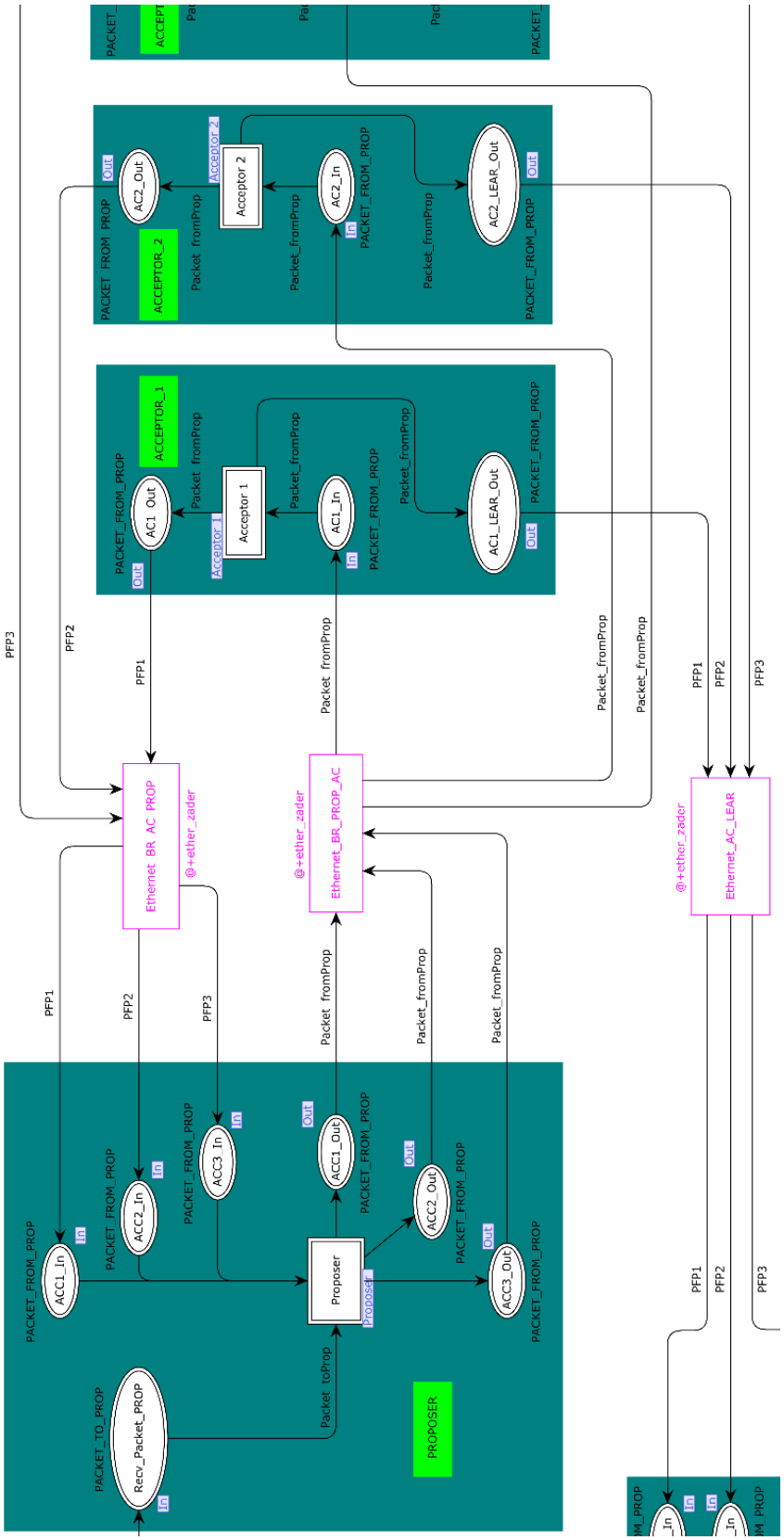

Рис. 2. Фрагмент сетевой модели верхнего уровня, содержащий подсети *Acceptor* 1, *Acceptor* 2 и *Proposer*

Моделирование осуществлялось при определенных временных параметрах, критерием выбора которых являлась их пригодность для сравнительной оценки двух подходов. Задержка между отправкой запросов, сгенерированных клиентом, была выбрана 10 мс, задержка при передаче данных по коммуникационной сети – 1 мс.

Значения времени, за которое каждая из подсетей (*SubLearner, Acceptor, Proposer, Learner, Leader_RAFT, Sub_RAFT*) выполняет обработку данных, были выбраны близкими к реальным, полученным в результате конкретных измерений в ходе отладки проекта. При проведении имитационного моделирования осуществлялся сбор следующих статистических данных:

а) минимальное, среднее, максимальное время выполнения операции захвата или освобождения семафора;

б) минимальное, среднее, максимальное время получения клиентом ответа на отправленный запрос;

в) количество запросов, пришедших от клиентов, в модули;

г) информация о состоянии семафора после успешного выполнения клиентского запроса.

В ходе анализа данных, полученных в процессе имитационного моделирования, были построены соответствующие таблицы и гистограммы. В табл. 1 представлены данные, полученные в результате анализа информационных пакетов, генерируемых при успешном выполнении операции захвата или освобождения семафора.

Таблица 1

Время успешного выполнения операций захвата/освобождения семафора

Количество величин в заданном диапазоне	PAXOS	RAFT
3–5 мс (шт.)	–	5836
5–8 мс (шт.)	5418	866
8–14 мс (шт.)	1158	0
более 14 мс (шт.)	97	0
Минимальное время	6,89 мс	3,13 мс
Среднее время	7,846 мс	3,781 мс
Максимальное время	18,12 мс	5,26 мс

На рис. 3 в качестве примера представлена гистограмма «Время получения клиентом ответа на выполненный запрос», построенная на основе результатов имитационного моделирования.

Исходя из результатов, полученных при имитационном моделировании моделей распределенного семафора на основе протоколов *Paxos* и *Raft*, можно сделать следующие выводы:

а) в сети на основе *Raft* клиенты получали ответ на свой невыполненный запрос в среднем на 21 % быстрее, чем в сети на основе *Paxos*;

б) в сети на основе *Raft* клиенты получали ответ на свой выполненный запрос в среднем на 46 % быстрее, чем в сети на основе *Paxos*;

в) в сети на основе *Raft* запрос на захват или освобождение семафора выполнялся в среднем на 52 % быстрее, чем в сети на основе *Paxos*;

г) обе сети успешно выполняли обработку запросов на захват или освобождение семафора от нескольких клиентов;

д) в обеих сетях в ходе тестирования была показана работоспособность механизма решения проблемы отказа компонента системы.

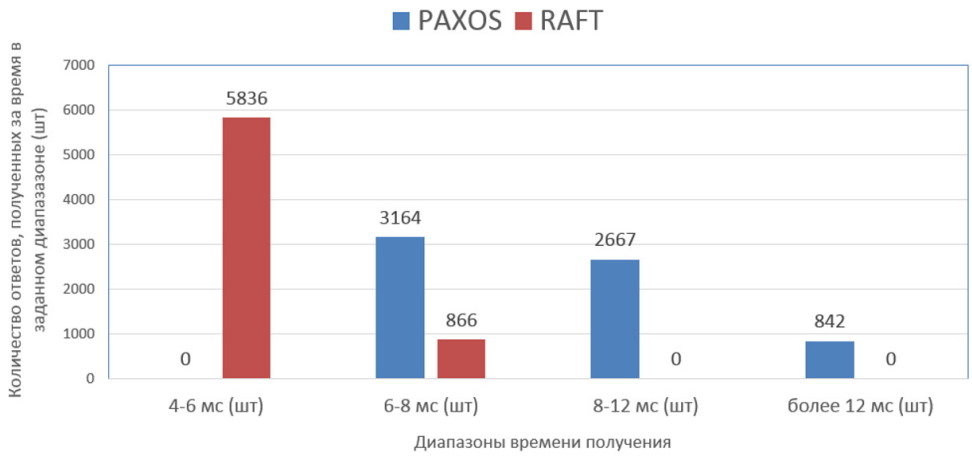

Рис. 3. Гистограмма «Время получения клиентом ответа на выполненный запрос»

Система «Распределенный семафор», реализованная с использованием алгоритма *Raft*, обладает следующими преимуществами:

а) простота алгоритма и его реализации;

б) возможность масштабирования системы с минимальными трудозатратами;

в) более высокие оценки производительности.

Можно утверждать, что использование алгоритма *Raft* при построении системы распределенного семафора является более простым и эффективным способом, чем использование протокола *Paxos*. Тем не менее недостатком *Raft* является то, что сохранение согласованности между репликами затруднено при наличии асинхронности, сбоев сети и сбоев узлов.

3. Функционально-блочная реализация распределенных семафоров на основе протоколов *Paxos* и *Raft*

Функционально-блочная реализация системы «Распределенный семафор на основе протокола *Paxos*» включает 14 ФБ, разделенных на следующие группы:

1) составные ФБ (CLIENT), реализующие функции по генерации клиентом запросов на взятие/освобождение семафора;

2) функциональные CAT-блоки для организации человеко-машинного интерфейса;

3) составные ФБ (PROPOSER, ACCEPTOR, LEARNER), реализующие соответствующие роли протокола *Paxos*.

Составной ФБ PROPOSER включает сеть ФБ, представленную на рис. 4.

Реализованная в среде *NxtStudio* система «Распределенный семафор на основе *Paxos*» корректно функционирует при обработке запросов на захват/освобождение семафора от нескольких клиентов.

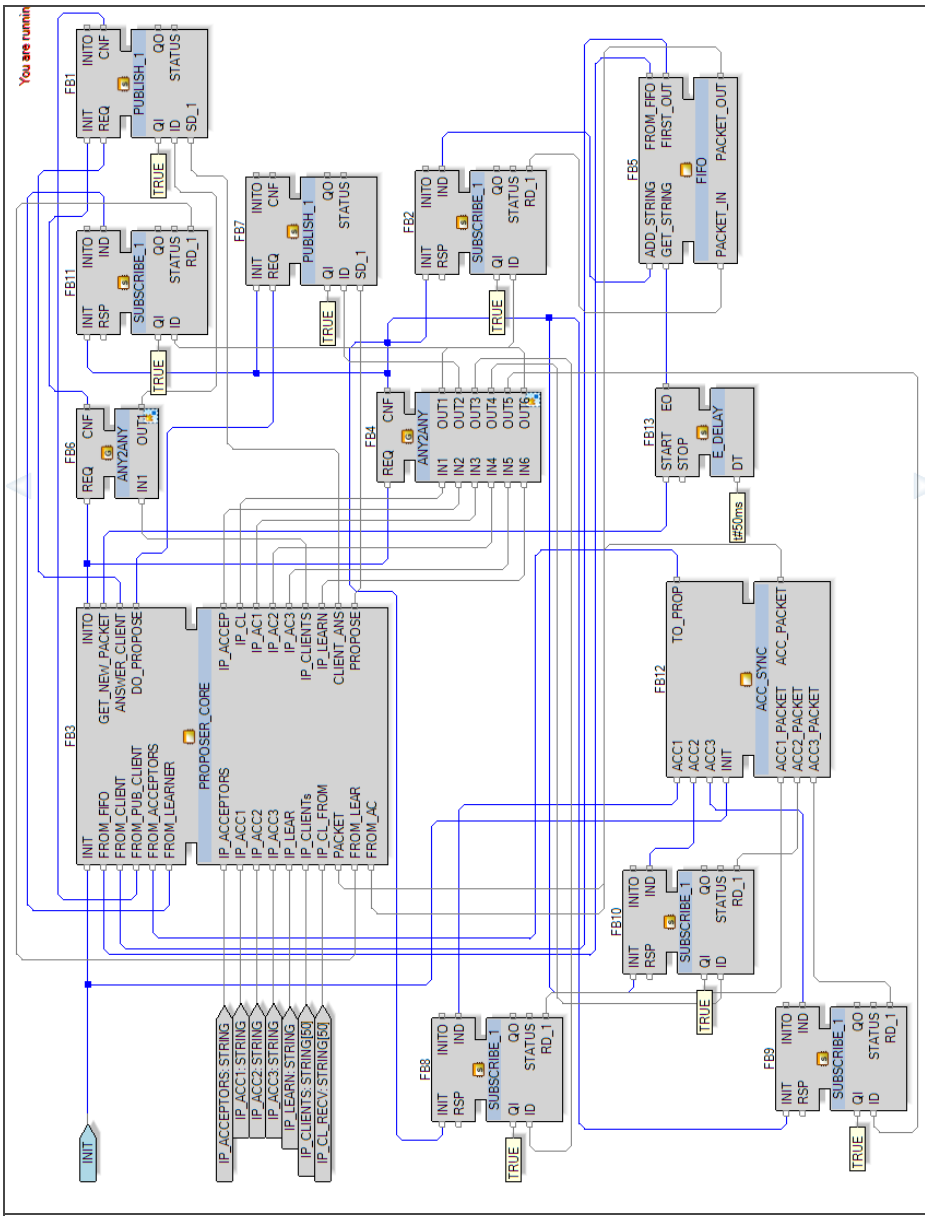

Рис. 4. Структура составного ФБ PROPOSER

При полном отказе модуля *Acceptor* система сохраняет свою работоспособность за счет особенностей протокола *Paxos*, если при этом сохраняется кворум. Описанные результаты подтверждены путем тестирования проекта на эмуляторе *SoftPLC*, входящем в среду исполнения *NxtStudio* [13].

Функционально-блочная реализация системы «Распределенный семафор на основе протокола *Raft*» включает 20 ФБ, разделенных на следующие группы:

1) составные ФБ (CLIENT), реализующие функции по генерации клиентом запросов на взятие/освобождение семафора;

2) функциональные CAT-блоки для организации человеко-машинного интерфейса;

3) составные ФБ (RAFT), реализующие функционал модулей протокола *Raft*.

Структура составного ФБ RAFT представлена на рис. 5.

Реализованная в среде *NxtStudio* система «Распределенный семафор на основе *Raft*» корректно функционирует при обработке запросов на захват/освобождение семафора от нескольких клиентов.

При полном отказе нескольких *Raft*-модулей (для функционирования системы требуется минимум два модуля) система сохраняет свою работоспособность за счет заложенных в алгоритм *Raft* репликации данных и алгоритма выбора нового лидера. Описанные результаты подтверждены путем тестирования проекта на эмуляторе *SoftPLC*.

Ниже приведено краткое сравнение реализаций *Paxos* и *Raft*. Протокол *Paxos* подразумевает наличие как минимум трех ролей с различным функционалом, в то время как в алгоритме *Raft* количество ролей сведено к одной, которая функционирует в двух режимах. Для сохранения работоспособности в системе на основе *Paxos* необходимо наличие одного модуля *Proposer*, двух модулей *Acceptor* и одного модуля *Learner*. Система на основе *Raft* функционирует при наличии одного модуля *Raft*, работающего в режиме лидера, и одного модуля *Raft*, работающего в режиме подчиненного. Таким образом, итоговая сеть ФБ на основе *Paxos*, получается более громоздкой, чем аналогичная сеть на основе *Raft*. В результате разработки системы в среде *NxtStudio* весь функционал протокола *Raft* вместился в составной ФБ RAFT.

При рассмотрении ситуации отказа компонентов системы также более привлекательным вариантом остается протокол *Raft*. В него изначально заложен механизм голосования при отказе модуля, работающего в режиме лидера. В результате голосования один из модулей, работающих в режиме подчиненного, становится новым лидером. Протокол *Paxos* тоже предполагает смену ролей при отказе компонента, например, с ролью *Proposer*, однако четко не определяет, модуль с какой ролью должен занять его место.

Предложенная в данной работе функционально-блочная реализация протоколов *Paxos* и *Raft* позволяет не только строить на их основе распределенные семафоры, но и обеспечивает корректную работу с распределенными данными и, таким образом, показывает путь реализации распределенных хранилищ данных в системах управления на основе ФБ.

Заключение

В данной работе были представлены следующие научные и практические результаты.

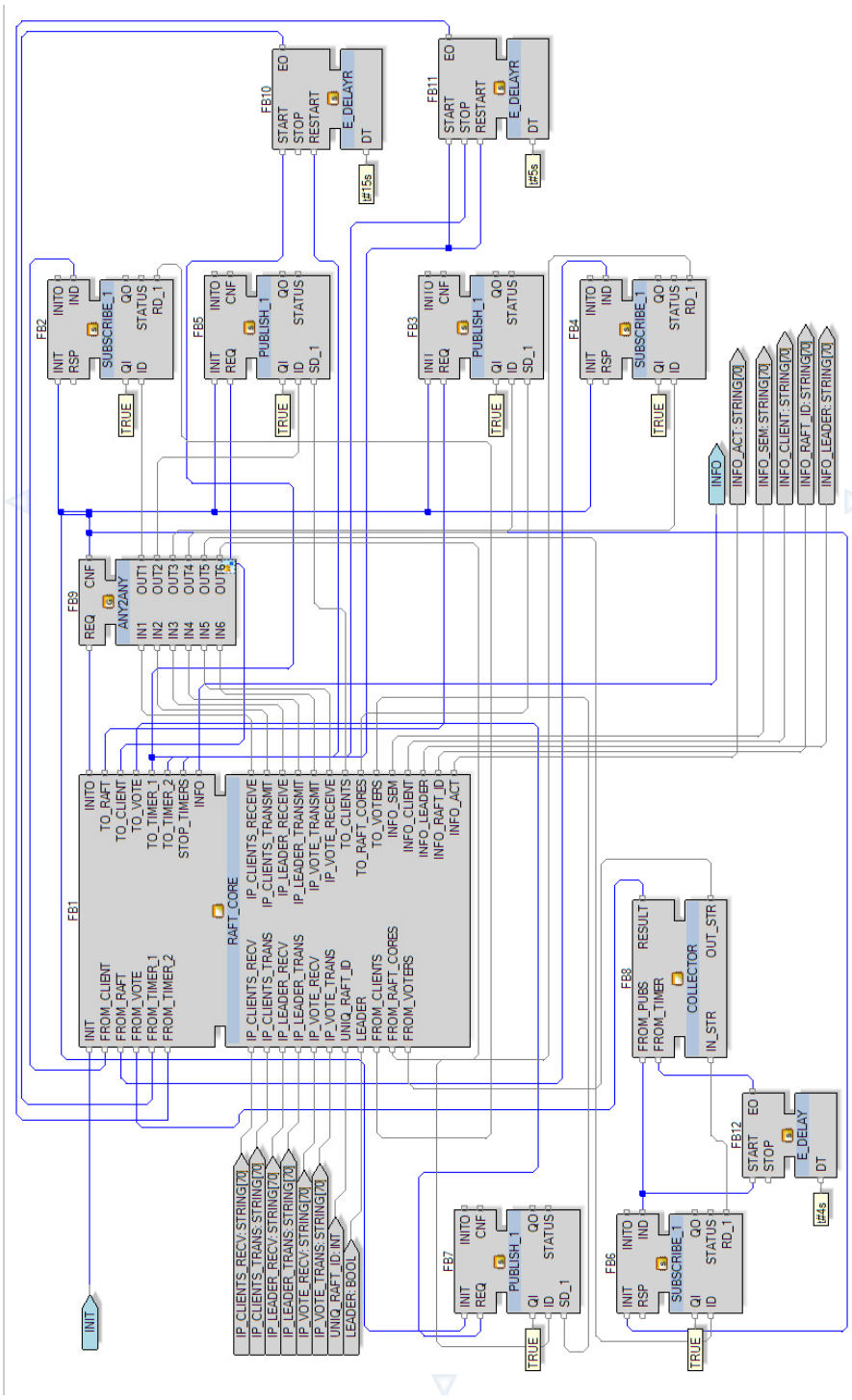

Рис. 5. Структура составного ФБ RAFT

1. Разработаны сетевые имитационные модели алгоритмов выполнения операций открытия и закрытия распределенных семафоров, основанных на алгоритмах *Paxos* и *Raft*, реализованные в системе *CPN Tools*, позволяющие производить как оценку производительности системы, так и ее верификацию.

2. Проведены имитационные эксперименты с сетевыми моделями распределенных семафоров на основе протоколов *Paxos* и *Raft*, дан сравнительный анализ и рекомендации по их использованию. Исследование обеих моделей показало корректность функционирования распределенных семафоров. Семафор на основе *Raft* оказался существенно производительнее семафора на основе *Paxos*.

3. Разработана библиотека функциональных блоков для реализации протоколов достижения консенсуса *Paxos* и *Raft*, а также распределенных семафоров на их основе, позволяющая существенно расширить возможности проектирования распределенных управляющих приложений со сложными видами взаимодействий в архитектуре IEC 61499. Данная библиотека реализована в системе *NxtStudio*.

Направлением дальнейших исследований является функционально-блочная реализация других механизмов синхронизации и взаимодействий, применимых в распределенных управляющих системах, например, механизма мониторов, задачи «производитель–потребитель», задачи «читатели–писатели» и т.д.

Библиографический список

1. **Дубинин, В. Н.** Модели функциональных блоков IEC 61499, их проверка и трансформации в проектировании распределенных систем управления : монография / В. Н. Дубинин, В. В. Вяткин. – Пенза : Изд-во ПГУ, 2012. – 348 с.
2. **Эндрюс, Г. Р.** Основы многопоточного, параллельного и распределенного программирования : пер. с англ. / Г. Р. Эндрюс. – М. : Вильямс, 2003. – 505 с.
3. **Вашкевич, Н. П.** Аппаратная реализация функций синхронизации параллельных процессов при обращении к разделяемому ресурсу на основе ПЛИС / Н. П. Вашкевич, Р. А. Бикташев, Е. И. Гурин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 2. – С. 3–12.
4. **Yuan, S.-M.** Design and implementation of a distributed semaphore facility / S.-M. Yuan, C.-J. Wu, H.-M. Lien, I.-N. Chen // Third Workshop on Future Trends of Distributed Computing Systems. – Taiwan, 1992. – P. 180–184.
5. **Дубинин, В. Н.** Реализация некоторых видов взаимодействий в системах, основанных на стандарте IEC 61499 / В. Н. Дубинин // Вычислительные системы и технологии обработки информации : межвуз. сб. науч. тр. – Пенза : Изд-во ПГУ, 2005. – Вып. 3 (29). – С. 65–71.
6. **Дубинин, В. Н.** Децентрализованное управление распределенными данными в локальной вычислительной сети: спецификация, моделирование, реализация и применение в распределенных вычислениях / В. Н. Дубинин / Пенз. гос. техн. ун-т. – Пенза, 1997. – 78 с. – Деп. В ВИНИТИ 10.06.97, № 1946-В97.
7. **Ramachandran, M.** Distributed semaphores. / M. Ramachandran, M. Singhal // Technical Report OSU-CISRC-6/94-TR34. – Ohio : The Ohio State University Computer and Information Science Research Center, 1994. – 20 p.
8. **Lamport, L.** The Part-Time Parliament // ACM Transactions on Computer Systems. – 1998. – Vol. 16, iss. 2. – P. 133–169.
9. **Ongaro, D.** In Search of an Understandable Consensus Algorithm / D. Ongaro, J. Ousterhout // Proc. USENIX Annual Technical Conference. – Philadelphia, 2014. – P. 305–320.

10. **Junqueira, F. P.** ZooKeeper: Distributed Process Coordination / F. P. Junqueira, B. Reed. – Sebastopol, United States : O'Reilly Media, 2013. – 246 p.
11. **Jensen, K.** Coloured Petri Nets. Modelling and Validation of Concurrent Systems / K. Jensen, L. M. Kristensen. – New York : Springer Publishing Company, 2009. – 384 p.
12. CPN Tools. – URL: http://cpntools.org/
13. NxtStudio. – URL: http://www.nxtcontrol.com/

References

1. Dubinin V. N., Vyatkin V. V. *Modeli funktsional'nykh blokov IEC 61499, ikh proverka i transformatsii v proektirovanii raspredelennykh sistem upravleniya: monografiya* [Models of functional blocks IEC 61499, their control and transfromations in the design og distributed control systems: monograph]. Penza: Izd-vo PGU, 2012, 348 p.
2. Endryus G. R. *Osnovy mnogopotochnogo, parallel'nogo i raspredelennogo programmirovaniya: per. s angl.* [Fundamentals of multithreaded, parallel and distributed programming: translation from English]. Moscow: Vil'yams, 2003, 505 p.
3. Vashkevich N. P., Biktashev R. A., Gurin E. I. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2007, no. 2, pp. 3–12.
4. Yuan S.-M., Wu C.-J., Lien H.-M., Chen I.-N. *Third Workshop on Future Trends of Distributed Computing Systems*. Taiwan, 1992, pp. 180–184.
5. Dubinin V. N. *Vychislitel'nye sistemy i tekhnologii obrabotki informatsii: mezhvuz. sb. nauch. tr.* [Computing systems and technologies of information processing: interuniversity collected papers]. Penza: Izd-vo PGU, 2005, iss. 3 (29), pp. 65–71.
6. Dubinin V. N. *Detsentralizovannoe upravlenie raspredelennymi dannymi v lokal'noy vychislitel'noy seti: spetsifikatsiya, modelirovanie, realizatsiya i primenenie v raspredelennykh vychisleniyakh* [Decentralized control of distributed data in a local computing network: specification, simulation, realization and application in distributed computations]. Penza, 1997, 78 p. Dep. V VINITI 10.06.97, no. 1946-V97.
7. Ramachandran M., Singhal M. *Technical Report OSU-CISRC-6/94-TR34*. Ohio: The Ohio State University Computer and Information Science Research Center, 1994, 20 p.
8. Lamport L. *ACM Transactions on Computer Systems*. 1998, vol. 16, iss. 2, pp. 133–169.
9. Ongaro D., Ousterhout J. *Proc. USENIX Annual Technical Conference*. Philadelphia, 2014, pp. 305–320.
10. Junqueira F. P., Reed B. *ZooKeeper: Distributed Process Coordination*. Sebastopol, United States: O'Reilly Media, 2013, 246 p.
11. Jensen K., Kristensen L. M. *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. New York: Springer Publishing Company, 2009, 384 p.
12. *CPN Tools*. Available at: http://cpntools.org/
13. *NxtStudio*. Available at: http://www.nxtcontrol.com/

Дубинин Виктор Николаевич
доктор технических наук, профессор, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dubinin.victor@gmail.com

Dubinin Viktor Nikolaevich
Doctor of engineering sciences, professor, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Беззатеев Игорь Андриянович
инженер, Пензенский научно-исследовательский электротехнический институт (Россия, г. Пенза, ул. Советская, 9)

Bezzateev Igor' Andriyanovich
Engineer, Penza Electrotechnical Research Institute (9 Sovetskaya street, Penza, Russia)

E-mail: apocalypse94@yandex.ru

Войнов Артем Сергеевич
магистрант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

Voynov Artem Sergeevich
Master's degree student, Penza State University (40 Krasnaya street, Penza, Russia)

E-mail: voj49@yandex.ru

Сенокосов Илья Владимирович
магистрант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

Senokosov Il'ya Vladimirovich
Master's degree student, Penza State University (40 Krasnaya street, Penza, Russia)

E-mail: senokosov.i@yandex.ru

УДК 004.75
Дубинин, В. Н.

Моделирование и реализация распределенных семафоров в архитектуре IEC 61499 на основе протоколов для достижения консенсуса в сети ненадежных процессов / В. Н. Дубинин, И. А. Беззатеев, А. С. Войнов, И. В. Сенокосов // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2018. – № 4 (48). – С. 39–53. – DOI 10.21685/2072-3059-2018-4-4.