МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Пензенский государственный университет» (ПГУ)
Российский фонд фундаментальных исследований

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ

СБОРНИК НАУЧНЫХ СТАТЕЙ
XIV Международной научно-технической
конференции,
посвященной 70-летию кафедры «Вычислительная техника»
и 30-летию кафедры «Системы автоматизированного проектирования»

г. Пенза, 22–24 ноября 2017 г.

Proceedings of the Fourteenth International Conference
of Science and Technology

NEW INFORMATION TECHNOLOGIES AND SYSTEMS

dedicated to the 70th anniversary of the Department
of Computer Engineering ,
and the 30th anniversary of Computer Aided Design Department

Penza, Russia, November 22–24, 2017

Пенза
Издательство ПГУ
2017

УДК 004.4
Н72

Новые информационные технологии и системы : сб. науч. ст. Н72 XIV Междунар. науч.-техн. конф., посвящ. 70-летию кафедры «Вычислительная техника» и 30-летию кафедры «Системы автоматизированного проектирования» (г. Пенза, 22–24 ноября 2017 г.). – Пенза : Изд-во ПГУ, 2017. – 420 с.

ISBN 978-5-906975-95-9

Сборник статей подготовлен на основе докладов, сделанных на XIV Международной научно-технической конференции «Новые информационные технологии и системы» (НИТиС–2017), посвященной 70-летию кафедры «Вычислительная техника» и 30-летию кафедры «Системы автоматизированного проектирования». Конференция проведена Министерством образования и науки РФ и Пензенским государственным университетом при поддержке Российского фонда фундаментальных исследований, проект 17-07-20569.

Доклады охватывают широкий спектр проблем в области проектирования и администрирования информационных систем и систем автоматизированного проектирования. Рассматриваются вопросы управления в социальных, экономических и технических системах. Представлены современные технологии хранения и обработки данных, информационные технологии в образовании. Обсуждаются вопросы математического моделирования в научных исследованиях, а также вопросы разработки программного и аппаратного обеспечения информационно-вычислительных систем и сетей и их информационной безопасности.

Издание предназначено для инженеров и научных работников, а также обучающихся старших курсов, интересующихся вопросами современных информационных технологий.

УДК 004.4

ISBN 978-5-906975-95-9

АППАРАТНО-ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СИСТЕМ УПРАВЛЕНИЯ

Д. Н. Дроздов[1,2], С. Патил[2], В. Н. Дубинин[1]

[1] Пензенский государственный университет, Пенза, Россия
[2] Технический университет Люлео, Люлео, Швеция

ФОРМАЛЬНАЯ ВЕРИФИКАЦИЯ КИБЕРФИЗИЧЕСКИХ СИСТЕМ УПРАВЛЕНИЯ СО СЛУЧАЙНЫМИ ВРЕМЕННЫМИ ЗАДЕРЖКАМИ

Рассматривается способ моделирования и формальной верификации распределенных киберфизических систем управления промышленной автоматикой. Для моделирования используется язык SMV с применением определенного класса абстракций с целью сокращения пространства состояний модели.

В настоящее время в области промышленной автоматики происходит сдвиг парадигмы с централизованных систем управления к распределенным. Такие системы управления содержат множество «умных» компонентов, таких как датчики, приводы и миниатюрные контроллеры, соединенных между собой посредством проводных и беспроводных сетей передачи данных, и образующих цельную систему совместно с управляемым оборудованием. В подобных системах важной характеристикой программного обеспечения является устойчивость к различным возмущениям физической среды, одним из наиболее часто наблюдаемых результатов которых является возникновение случайных задержек пакетов в беспроводных сетях передачи данных.

Данная проблема изначально рассматривалась в рамках теории автоматического управления [1, 2], однако, в данных работах не принимается во внимание конкретная реализация управляющего программного обеспечения (ПО). В качестве идеи, призванной охватить одновременно и процессы, физической системы и особенности программной реализации, в работе [3] предлагается концепция киберфизическог агностицизма (Cyber-Physical Agnosticism, CPA). Данная концепция рассматривается как свойство ПО проявлять всегда предсказуемое поведение, не зависимое от различных возмущений в рассматриваемой киберфизической системе.

Как было продемонстрированно в работах [4–6], распределенные киберфизические системы с сервисно-ориентированной архитектурой могут эффективно моделироваться с использованием функциональных блоков стандарта IEC 61499, а в работе [7] демонстрируется использование веб-сервисов в приложениях на базе IEC 61499.

В то время, как киберфизическая система может быть промоделирована в системе Ptolemy II с применением гетерогенной комбинации дискретного и непрерывного доменов (с использованием семантики pTides), реализация подобных систем требует соответствующей модификации языка, описываемого стандартом IEC 61499, за счет расширения понятия «событие» добавлением временных меток.

В данной статье рассматривается упрощенная реализация временных меток для IEC 61499. Каждому событийному входу или выходу функционального блока, ставится в соответствие дополнительный информационный вход/выход типа TIME, предназначенный для передачи временной метки соответствующего события между функциональными блоками [8]. Использование подобной методики позволяет выполнять моделирование и даже тестирование в реальных условиях без необходимости внесения изменений в среду исполнения и разработки ПО.

Для управления событиями с переменными метками, в модель добавлен инкрементный глобальный таймер, а чтобы ограничить пространство состояний при верификации модели, рассматривается только интервал времени от 0 до $T_{\max}$. Изменение глобальной переменной времени происходит в соответствии со следующим правилом:

$$T_{Global} < T_{max} \vee \gamma \vee (\wedge(Do_i)) \Rightarrow T_{Global} = T_{Global} + D_{min}.$$

Кроме того, правила изменения локального времени блоков E_DELAY/E_CYCLE, приведенные в [9], были модифицированы таким образом, чтобы останавливать выполение при достижении глобального времени $T_{\max}$.

Предложенный подход протестирован на примере цилиндра с PID-регулятором, приведенном в работе [3]. На рис. 1 приведено схематичное изображение данного примера. Контроллер пневматического цилиндра с возвратной пружиной реализован в виде управляющего приложения на базе функциональных блоков.

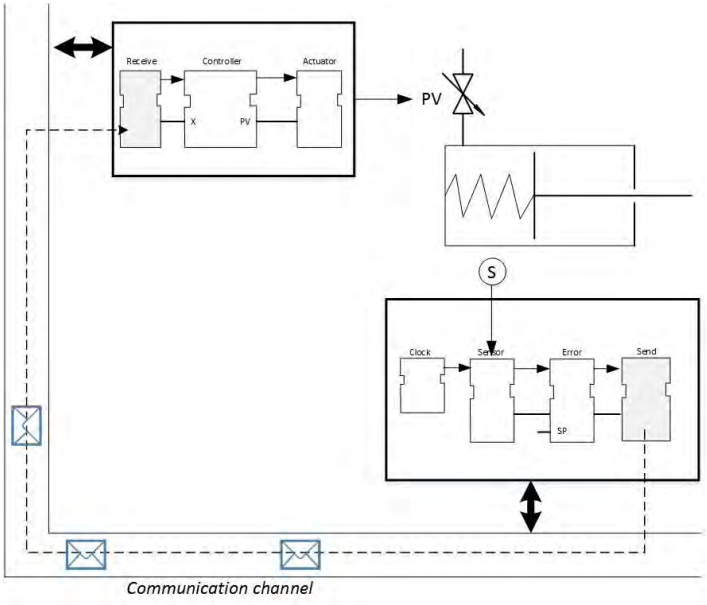
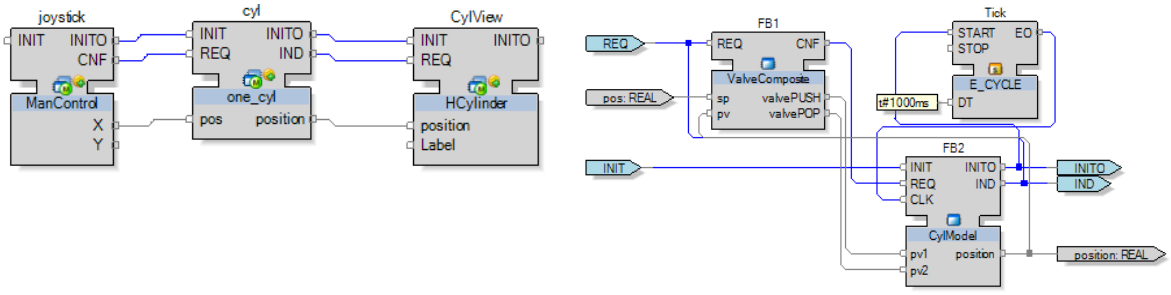


Рис. 1. Демонстрационный пример: цилиндр с PID-регулятором

На рис. 2,*а* показана модель верхнего уровня в системе nxtStudio, состоящая из джойстика, модели цилиндра с контроллером, и визуализации цилиндра, активного в режиме имитационного моделирования. На рис. 2,*б* изображена замкнутая модель цилиндра с контроллером.

Для формальной верификации данной системы использовался верификатор nuSMV. Основа SMV-модели контроллера сгенерирована с помощью утилиты fb2smv из управляющего ПО и модифицирована и дополнена как показано ниже.



(*а*) (*б*)

Рис. 2. Система верхнего уровня в среде nxtStudio (*а*)
и замкнутая модель цилиндра с контроллером (*б*)

В первую очередь, в имитационной модели в среде nxtStudio, были созданы два CAT-блока: GetTimestamp и TimeDiff, отвечающие за генерацию временной метки события и вычисления разницы между временной меткой и текущим временем системы, а так же добавлен блок E_DELAY, эмулирующий задержку передачи данных в сети. Эмуляция случайной задержки в беспроводных сетях передачи данных обеспечивается за счет подачи на вход блока E_DELAY случайной величины задержки при подаче события на вход.

Полученная после генерации SMV-кода система дополняется SMV-моделью оборудования и модифицированными временным диспетчерами. Для сокращения пространства состояний модели, непрерывное изменение положения поршня в управляемом цилиндре дискретизируется по времени с шагом в 500 единиц модельного времени. График изменения положения поршня, полученный из контрпримера, сгенерированного системой nuSMV, приведен на рис. 3.

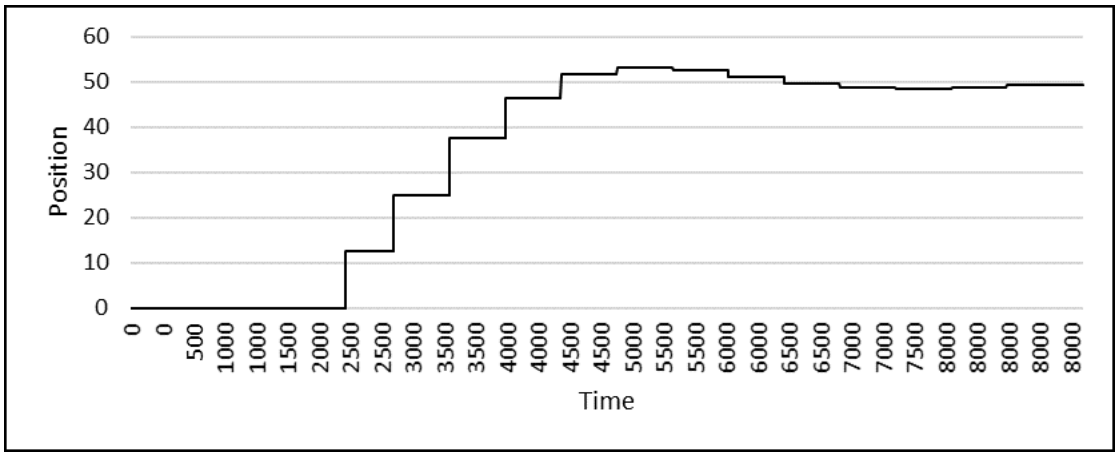


Рис. 3. График положения поршня в цилиндре, извлеченный из контрпримера nuSMV

Библиографический список

1. Nilsson, J. Analysis of real-time control systems with time delays / J. Nilsson, B. Bernhardsson // Decision and Control : Proceedings of the 35th IEEE Conference on. – IEEE, 1996. – Т. 3. – С. 3173–3178.

2. Nilsson, J. Real-time control systems with delays / J. Nilsson et al. ; Lund Institute of Technology. – Lund, 1998. – 138 p.

3. Vyatkin, V. Towards cyber-physical agnosticism by enhancing IEC 61499 with ptides model of computations / V. Vyatkin, C. Pang, S. Tripakis // Industrial Electronics Society : IECON 2015-41st Annual Conference of the IEEE. – IEEE, 2015. – С. 001970–001975.

4. Dai, W. Bridging service-oriented architecture and IEC 61499 for flexibility and interoperability / W. Dai et al. // IEEE Transactions on Industrial Informatics. – 2015. – Т. 11, № 3. – С. 771–781.

5. Dai, W. Service-oriented distributed control software design for process automation systems / W. Dai et al. // Systems, Man and Cybernetics (SMC) : 2014 IEEE International Conference on. – IEEE, 2014. – С. 3637–3642.

6. Dai, W. Function block implementation of service oriented architecture: Case study / W. Dai et al. // Industrial Informatics (INDIN) : 2014 12th IEEE International Conference on. – IEEE, 2014. – С. 112–117.

7. Demin, E. Automation services orchestration with function blocks: Web-service implementation and performance evaluation / E. Demin et al. // Service Orientation in Holonic and Multi-Agent Manufacturing. – Springer International Publishing, 2016. – С. 213–221.

8. Drozdov, D. Formal Modelling of Distributed Automation CPS with CP-Agnostic Software / D. Drozdov, S. Patil, V. Vyatkin // Service Orientation in Holonic and Multi-Agent Manufacturing. – Springer, Cham, 2017. – С. 35–46.

Д. Н. Дроздов[1,2], С. Патил[2]

[1] Пензенский государственный университет, Пенза, Россия
[2] Технический университет Люлео, Люлео, Швеция

ПОСТРОЕНИЕ РАСПРЕДЕЛЕННЫХ СИСТЕМ УПРАВЛЕНИЯ ПРОМЫШЛЕННОЙ АВТОМАТИКОЙ НА ОСНОВЕ СТАНДАРТА IEC 61499: ДЕКОМПОЗИЦИЯ И ОТЛАДКА

Активно развиваемый в настоящее время стандарт для построения систем управления промышленной автоматикой IEC 61499 предлагает новый подход к построению распределенных динамически реконфигурируемых систем управления промышленной автоматикой. Данная статья на реальном примере показывает возможности декомпозиции сложного управляющего автомата на множество простых функциональных блоков в рамках стандарта IEC 61499, а также касается вопросов отладки подобных распределенных систем.

Активно развиваемый в настоящее время стандарт для построения систем управления промышленной автоматикой IEC 61499 предлагает новый подход к построению распределенных динамически реконфигурируемых систем управления промышленной автоматикой [1], а с применением последних решений, – интеграцию идеологии интернета вещей для построения гибких производственных киберфизических систем. Кроме того, ряд работ направлен на создание и адаптацию различных методов верификации подобных систем [2–4].

Однако, старый подход к разработке централизованного управляющего ПО за десятки лет доминирования стандарта IEC 61131-3 прочно укоренился в головах разработчиков. Данная статья на реальном примере показывает возможности декомпозиции сложного управляющего автомата на множество простых функциональных блоков в рамках стандарта IEC 61499, а также преимущество данного подхода для отладки и построения реконфигурируемых систем.

В качестве исходного примера для демонстрации подхода был взят модуль DistributionStation учебной производственной системы FESTO MPS 500. Внешний вид станции приведен на рис. 1. Станция состоит из магазина заготовок с выталкивателем и руки-маятника с присоской, выполняющей захват заготовки и передачу ее к следующей станции, а также ряда двигателей и датчиков.

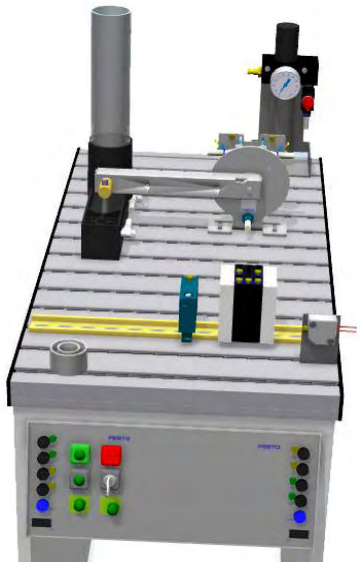


Рис. 1. Внешний вид (3D-модель) модуля DistributionStation

Управляющее ПО данной станции представлено как компонент общей системы управления сборочной линии FESTO MPS 500. Данный компонент, помимо прочего, осуществляет коммуникацию с тестирующей станцией (которой передается заготовка) по линиям передачи данных. На рис. 2 изображена диаграмма функциональных блоков верхнего уровня управляющего ПО всей системы.

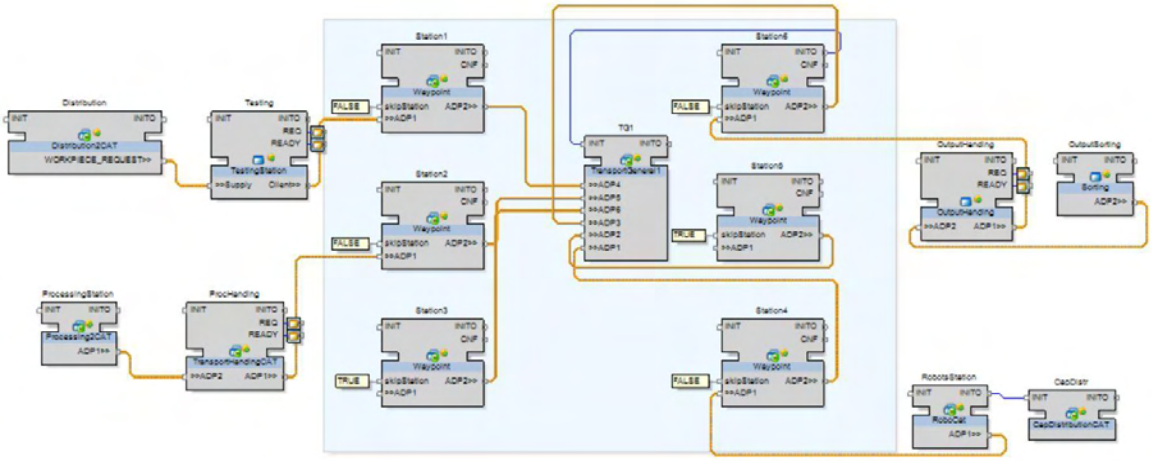


Рис. 2. Диаграмма функциональных блоков управляющего ПО системы MPS 500

Классический подход к созданию системы управления отдельным модулем основывается на создании одного управляющего автомата с множеством состояний. Такой автомат продемонстрирован на рис. 3. Данный подход позволяет быстро создать прототип управляющего ПО, однако, дальнейшая отладка и модификация оказывается затруднительной.

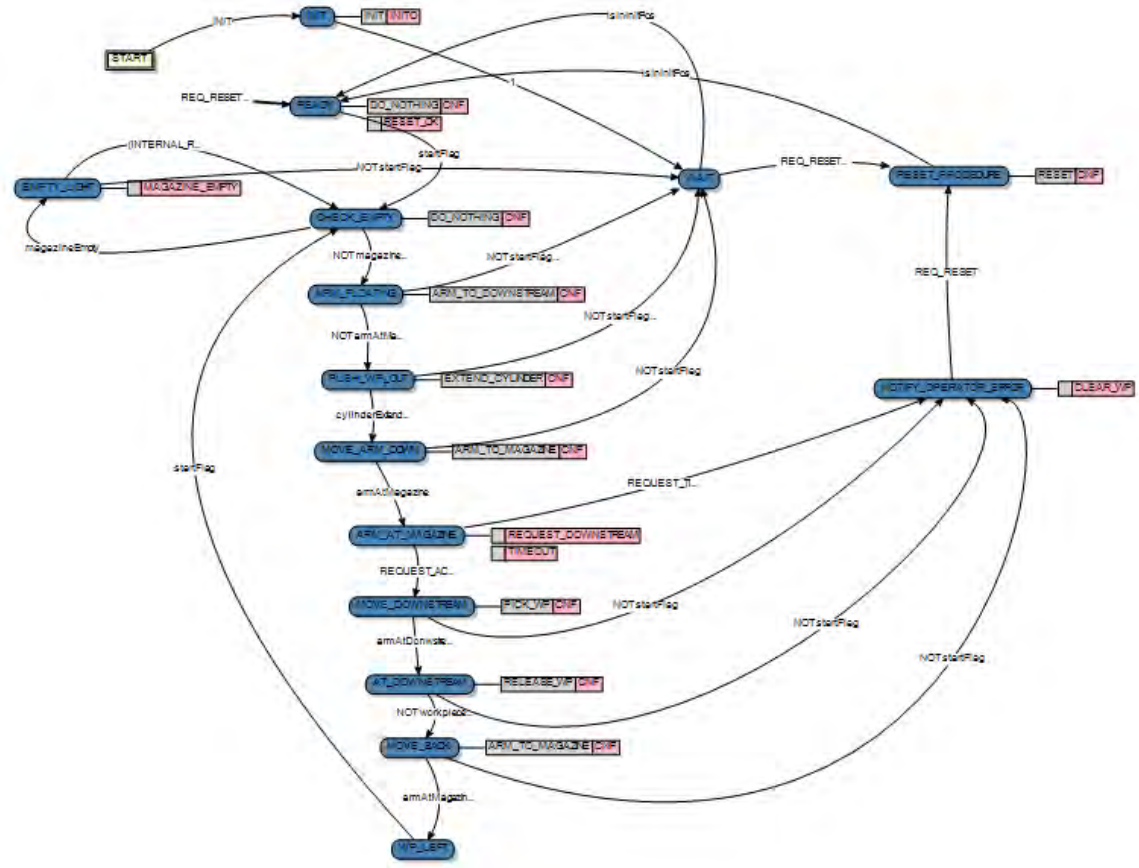


Рис. 3. Управляющий автомат (в форме диаграммы ECC) модуля DistributionStation

Так как все реализации стандарта IEC 61499 предоставляют развитые средства для отладки распределенной блочной событийно-управляемой архитектуры, целесообразной оказывается декомпозиция управляющего блока на множество отдельных, событийно-управляемых блоков, выполняющих простейшие операции.

Первое, что необходимо выделить при такой декомпозиции – модули преобразования управляющих событийных сигналов в логические уровни на портах ввода-вывода контроллера (условно назовем их драйверами механических устройств). В рассматриваемом примере было выделено три таких драйвера, названных: CustomStopActuator, SingleActingCylinder и ArmReset. Фактически, оба драйвера ArmReset и CustomStopActuator управляют одним и тем же элементом оборудования – рукой-маятником, однако, драйвер ArmReset выполняем специфическую процедуру установки первоначальной позиции, поэтому его было решено реализовать как отдельный блок. На рис. 4 изображены вышеописанные блоки драйверов с соответствующими управляющими автоматами.

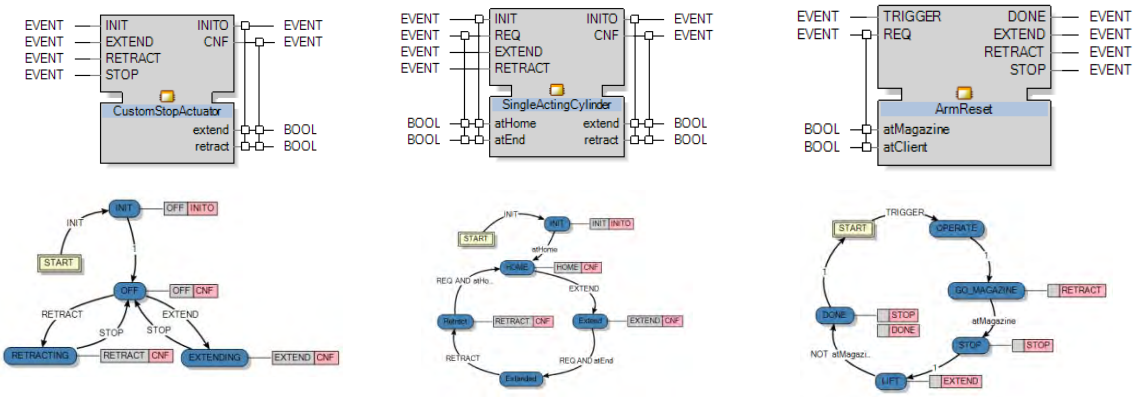


Рис. 4. Блоки «драйверов» механических устройств

Основная управляющая программа была разбита на серию простейших действий, каждое из которых выполняется специальным блоком-агентом, активизирующемся при поступлении сигнала на событийный вход TRIGGER и выдающим событийный сигнал DONE после окончания исполнения. Главная управляющая последовательность модуля DistributionStation изображена на рис. 5 (соединительные линии входных и выходных сигналов от оборудования/драйверов были удалены, чтобы не загромождать рисунок). Как можно увидеть, управляющая последовательность состоит из шести действий: GetWP (получить заготовку из магазина), ToMagazine (переместить руку-маятник в сторону магазина), Grab (захватить заготовку используя управляемую присоску), ToClient (переместить руку-маятник в пул для отгрузки заготовку), Release (используя присоску, отпустить заготовку в пул отгрузки), Reset (вернуть руку-маятник в первоначальное положение).

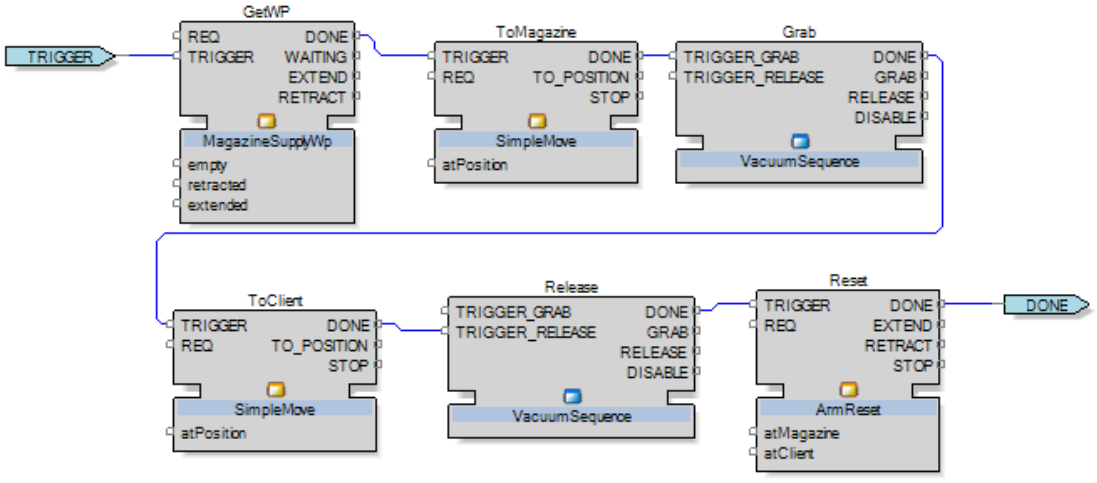


Рис. 5. Главная управляющая последовательность модуля DistributionStation

Каждый из блоков в данной последовательности имеет в своей основе простой управляющий автомат наподобие изображенных на рис. 4. Подобный подход, за счет контроля входных и выходных событий блоков, позволяет с легкостью выполнять отладку сначала отдельных простейших действий, а позже и комбинированных цепочек. Каждая цепочка действий, наподобие приведенной на рис. 5, заключается внутрь составного функционального блока и такой блок уже сам может выступать как реализация отдельного действия в более сложной комбинированной программе.

Библиографический список

1. Vyatkin, V. IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review / V. Vyatkin // IEEE transactions on industrial informatics. – 2011. – Т. 7, № 4. – С. 768–781.
2. Дубинин, В. Н. Модели функциональных блоков IEC 61499, их проверка и трансформации в проектировании распределенных систем управления : моногр. / В. Н. Дубинин, В. В. Вяткин. – Пенза : Изд-во ПГУ, 2012. – 348 с.
3. Patil, S. Cloud-Based Framework for Practical Model-Checking of Industrial Automation Applications / S. Patil, D. Drozdov, V. Dubinin, V. Vyatkin // DoCEIS. – 2015. – С. 73–81.
4. Drozdov, D. Formal verification of cyber-physical automation systems modelled with timed block diagrams / D. Drozdov et al. // Industrial Electronics (ISIE) : IEEE 25th International Symposium on. – IEEE, 2016. – С. 316–321.

Д. Н. Дроздов[1,2], С. Патил[2]

[1] Пензенский государственный университет, Пенза, Россия
[2] Технический университет Люлео, Люлео, Швеция

РАСПРЕДЕЛЕННАЯ СИСТЕМА УПРАВЛЕНИЯ ПРОИЗВОДСТВЕННОЙ ЛИНИЕЙ FESTO MPS-500

В рамках развития систем управления промышленной автоматикой важную позицию занимает переход от классических централизованных систем к распределенным. Серьезный толчок в данном направлении продиктован концепцией Промышленности 4.0. В данной статье описывается пример распределенной системы управления, разработанной для учебной производственной линии FESTO MPS-500 на базе лаборатории AIC[3] Технического Университета Люлео (Швеция).

Современные тенденции развития в области промышленной автоматики, продиктованные концепцией Промышленности 4.0 [1] требуют создания совершенно новых интеллектуальных систем управления [2]. Стандарт IEC 61499 для создания распределенного программного обеспечения (ПО) систем промышленной автоматики предлагает частичное решение этой проблемы на программном уровне. Однако, новая парадигма разработки ПО накладывает определенные требования к аппаратной платформе. Системы управления, созданные на базе предыдущего стандарта IEC 61131-3 были в высокой степени централизованным и жестко программировались под один сценарий поведения, при этом возможность быстрой реконфигурации системы никогда не рассматривалась и в аппаратной части было достаточно ПЛК, жестко «привязанного» к оборудованию и инфраструктуре. Новые системы управления на базе IEC 61499 предлагают беспрецедентные возможности [3], позволяющие, после изменения конфигурации производственного оборудования, выполнять реконфигурацию уже на программном уровне всего в пару щелчков мыши. Это становится возможным благодаря использованию множества «умных» аппаратных компонентов со встроенным контроллером, заранее запрограммированным для выполнения базовых операций.

Реконфигурируемые системы промышленной автоматики, которые должны работать в рамках новых «умных» фабрик, подразумевают возможность быстрой перекомпоновки системы. На программном уровне, как уже было сказано выше, данный вопрос решается созданием управляющего ПО с использованием распределенных стандартов программирования, например, IEC 61499, и внедрением динамических сценариев реконфигурации. Аппаратная составляющая управляющей системы, в то же время, требует декомпозиции на максимально автономные управляющие блоки, способные перемещаться вместе с управляемым оборудованием при физической реконфигурации системы.

Структура разработанного прототипа такого управляющего блока приведена на рис. 1. Он состоит из программируемого логического контроллера (ПЛК) со встроенными блоками цифрового ввода-вывода (БВВ), блока питания (БП), WiFi адаптера и дополнительного блока цифрового ввода-вывода, соединенного с ПЛК посредством интерфейса Ethercat.

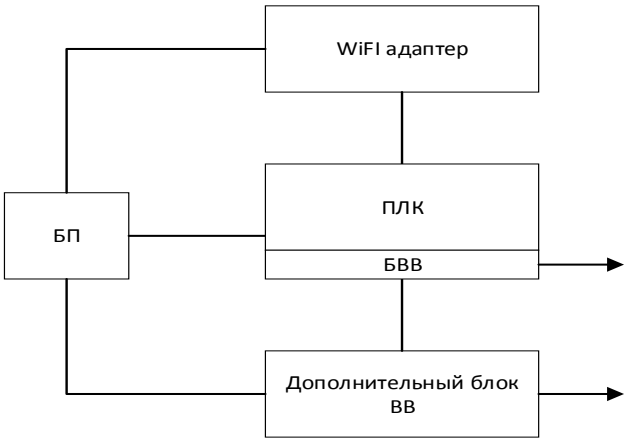


Рис. 1. Общая структура управляющего блока

В качестве программируемого логического контроллера использован миниатюрный ПЛК nxtControl DCSmini, поддерживающий управляющие программы на на базе стандарта IEC 61499. Данный ПЛК имеет сетевой интерфейс Ethernet для загрузки программ и соединения с другими ПЛК и иными компонентами управляющей системы, однако для поддержки большей мобильности, в управляющий блок включен адаптер Ethernet->WiFi, обеспечивающий подключение к беспроводным сетям передачи данных. Внешний вид собранного блока приведен на рис. 2.

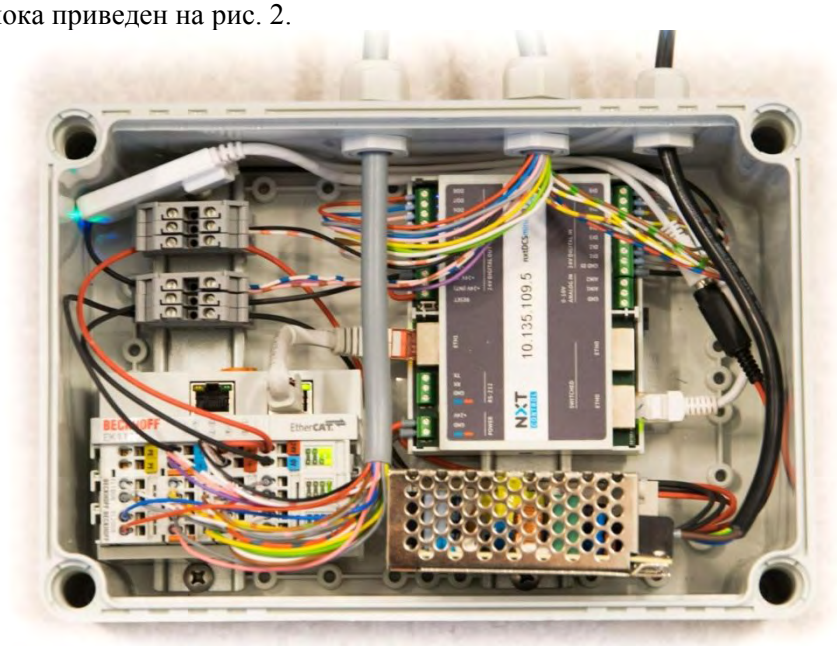


Рис. 2. Внешний вид управляющего блока

Данные блоки использованы для создания системы управления учебной производственной линией FESTO MPS-500, являющейся одной из подсистем лаборатории AIC[3][4] Технического университета Люлео (Швеция). Тестовая система, изображенная на рис. 3, состоит из конвейера, станции распределения заготовок, станции тестирования заготовок, двух манипуляторов, станции обработки деталей, станции сортировки деталей, станции взаимодействия со сборочным роботом, беспроводного маршрутизатора и рабочего местом программиста и оператора.

Рис. 3. Схема учебной производственной системы

Каждая станция была оснащена собственным управляющим блоком, а также для конвейера был создан отдельный управляющий блок без дополнительного блока ввода-вывода, но с более мощным ПЛК. В общей сложности, распределенная система управления состоит из 8 управляющих блоков, соединенных по сети WiFi между собой и с рабочим местом программиста/оператора, на которые было развернуто программное обеспечение, созданное по стандарту IEC 61499. На рис. 4 приведена диаграмма функциональных блоков верхнего уровня для управляющего ПО сборочной линии. В случае применения классической централизованной схемы управления с применением только проводных интерфейсов, при перемещении станций сборочной линии для изменения сценария работы, потребовалась бы трудоемкая работа по переподключению множества соединений и изменению управляющего ПО, однако с применением нового подходя, требуется лишь переключение питания станций и минимальные изменения в программе.

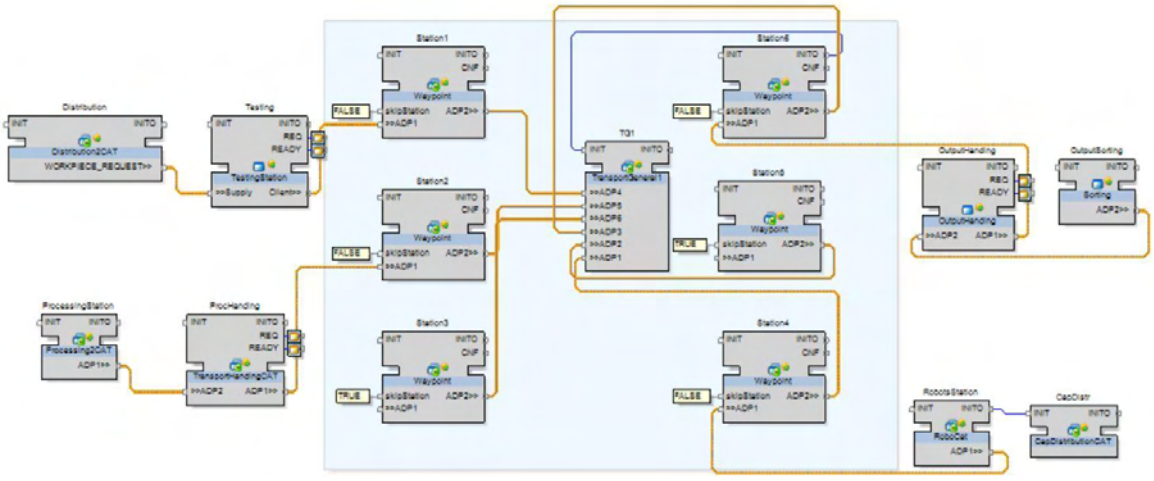


Рис. 4. Диаграмма функциональных блоков управляющего ПО сборочной линии

Заключение

В данной статье представлен прототип управляющего блока с беспроводным интерфейсом для применения в распределенных системах управления промышленной авто-

матикой с программным обеспечением на базе стандарта IEC 61499. Действенность подхода продемонстрирована на примере лабораторной сборочной линии.

Современные тенденции развития систем управления промышленной автоматикой требуют создания малогабаритных контроллеров с низким энергопотреблением. Множество наработок из области встроенной техники, однако, до сих пор не нашли применения в системах промышленной автоматики. Использование малогабаритных беспроводных контроллеров в то же время, позволит распределенным системам управления выйти на совершенно новый уровень – интернета вещей в области промышленной автоматики, где «умные» компоненты оборудования могут собираться в целые производственные линии и фабрики. Последующая работа направлена на разработку подобных малогабаритных контроллеров с беспроводными интерфейсами и поддержкой программ стандарта IEC 61499.

Библиографический список

1. Lasi, H. Industry 4.0 / H. Lasi et al. // Business & Information Systems Engineering. – 2014. – Т. 6, №. 4. – С. 239–242.

2. Monostori, L. Cyber-physical production systems: Roots, expectations and R&D challenges / L. Monostori // Procedia CIRP. – 2014. – Т. 17. – С. 9–13.

3. Vyatkin, V. IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review / V. Vyatkin // IEEE transactions on industrial informatics. – 2011. – Т. 7, № 4. – С. 768–781.

4. AIC^{3}lab: Automation, Industrial Computing, Communication and Control. – Luleå University of Technology, LTU. – URL: https://www.ltu.se/AICcube

Р. А. Бикташев, Н. П. Вашкевич

Пензенский государственный университет, Пенза, Россия

СТРУКТУРНАЯ РЕАЛИЗАЦИЯ И МОДЕЛИРОВАНИЕ АЛГОРИТМОВ ПЛАНИРОВАНИЯ/ДИСПЕТЧЕРИЗАЦИИ В ПАРАЛЛЕЛЬНЫХ СИСТЕМАХ НА ОСНОВЕ ЛОГИКИ НЕДЕТЕРМИНИРОВАННЫХ АВТОМАТОВ

Предложена структура аппаратного планировщика/диспетчера задач в составе многопроцессорной системы, его модели на языке недетерминированных автоматов и языке VHDL. Проведены моделирование и оценка результатов.

Известно, что системы реального времени требуют высокой реакции на запрос, а также гарантированного времени ответа. Это обстоятельство вызывает необходимость повышения производительности операционной системы, особенно систем жесткого реального времени.

Структурная реализация функций многопроцессорных операционных систем заключается в перенесении трудоемких компонент планирования и диспетчеризации задач на уровень специализированного устройства управления (процессора). Такой перенос позволяет освободить программную часть ОС от функций управления переключением задач и прерываниями. В результате сокращается время выполнения соответствующих API-функций, а тщательная отладка аппаратуры современными средствами проектирования ПЛИС гарантирует надежность и безопасность операционной системы. Аппаратный подход позволяет также уменьшить разброс времени (джиттер) выполнения процессов по сравнению с программной реализацией, что значительно повышает гарантийные сроки выполнения приложений реального времени.

Существуют два подхода к реализации планировщиков/диспетчеров задач: с глобальной очередью и с локальными очередями в каждом процессорном узле. Аппаратная

реализация первого из них достаточно подробно рассмотрена в [1], структурная реализация второго представлена на рис. 1.

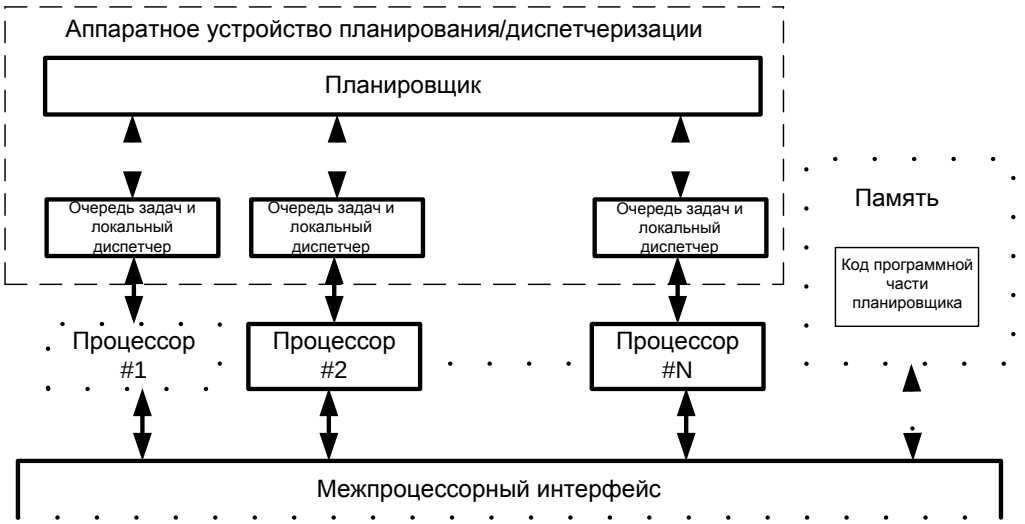


Рис. 1. Схема включения планировщика/диспетчера задач в состав многопроцессорной системы

В аппаратной части можно выделить два основных типа блока: собственно планировщик задач и локальную очередь задач с диспетчером. Первый принимает извне (например, из внешней памяти) новую задачу, помещает ее в одну из очередей, после чего в работу включается локальный диспетчер.

Интерфейс взаимодействия устройства локального диспетчера с каждым из процессоров и с планировщиком задач представлен на рис. 3.

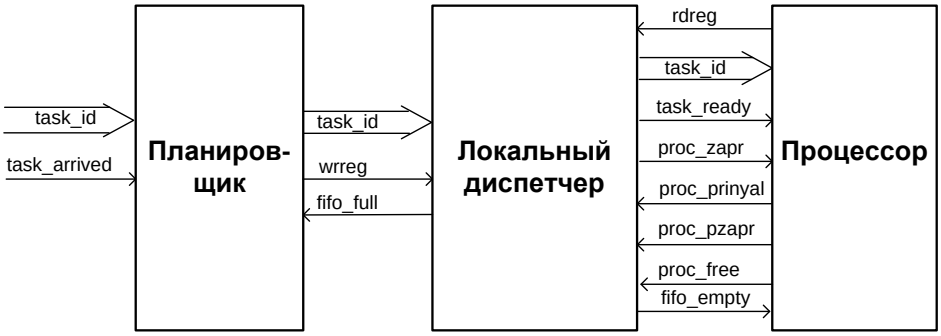


Рис. 2. Интерфейс взаимодействия локального диспетчера с планировщиком задач и процессором

Формализация алгоритма планирования задач

Алгоритм планирования/диспетчеризации основан на классической задаче «спящий парикмахер». Поступившая задача находится в планировщике до тех пор, пока не получит необходимые ей ресурсы, в число которых входит и место в очереди к одному из процессоров. Число мест в очереди ограничено. Если имеется свободное место в одной из очередей, то планировщик помещает новую задачу на это место. Принятая на обслуживание задача находится в очереди до тех пор, пока не поступит на выполнение в процессор. Если используется режим квантования, то незавершенная задача по окончании текущего кванта помещается в конец той же очереди, где она находилась ранее, иначе результат выдается пользователю, а в очереди освобождается одно место. При завершении задачи локальный диспетчер просматривает очередь, и если в ней имеются заявки на обслуживание, то назначается на выполнение процесс, стоящий в голове списка (очередь типа *FIFO*). Если очередь пуста, процессор переходит в режим ожидания [2].

Синхронизация процессов принятия задач в очередь и назначения очередной задачи процессору реализуется методом «рандеву» [3]. Формальное описание алгоритма взаимодействия процессов в данной задаче базируется на использовании моделей недетерминированных автоматов (НДА). Модели представляются в виде графа и/или систем канонических уравнений (СКУ), описывающих все реализуемые события управляющего алгоритма.

Основные частные события алгоритма представлены в табл. 1. В последней колонке показано их соответствие сигналам на схеме устройства управления процессами.

Обозначение события	Описание частного события	Сигналы на схеме (рис. 2)
S_I^t	Ожидание поступления новой задачи	*S_task_arrived*
S_U^{Qj}	Функция состояния очереди (количество задач в очереди *j*-го процессора)	*S_use [5…0]*
S_{FQ}^t	Все очереди заполнены	*S_fifo_full*
$\overline{S_{FQj}^t}$	В *j*-й локальной очереди имеются свободные места	$\overline{\text{S\_fifo\_fullj}}$
S_{SL}^{Qj}	Выбор очереди с минимальным числом задач и занесение новой задачи в выбранную очередь	*S_wrreg*
S_{Qj}^t	Задача в очереди *j*-го процессорного узла	*S_task_in_queue*
S_{ZPj}^t	Запрос *j*-го процессора локальный диспетчером	*S_proc_zapr_j*
S_{GPj}^t	Задача готова к обслуживанию в *j*-м процессоре	*S_task_ready_j*
S_{PZ}^{pj}	Подтверждение запроса *j*-м процессором	*S_proc_pzapr_j*
S_S^{pj}	Свободен *j*-й процессор	*S_proc_free*
S_{PT}^{pj}	Задачу *j*-й процессор принял	*S_proc_prinyal_j*
S_A^{pj}	Выполняет обслуживание задачи *j*-й процессор	Аппаратно не реализовано
S_O^{Qj}	Ожидание поступления задачи в очередь *j*-го процессора	Внутренний сигнал
S_{OF}^t	Удалить задачу из очереди	--------\\--------
S_{TO}^t	Снять задачу с исполнения	--------\\--------
S_E^{pj}	Выполнение задачи закончено	--------\\--------
S_{RT}^{pj}	Выдача результата выполнения текущей задачи	--------\\--------
S_{SZ}^{pj}	Процессор в режиме ожидания	--------\\--------

На основании словесного алгоритма планирования/диспетчеризации и введенных событий, система канонических уравнений будет иметь вид:

– для процесса «клиент», реализуемого локальным диспетчером до момента рандеву:

$$S_I^t(t+1) = x_Z \vee S_I^t \overline{S_{FQ}^t}\ ;\ S_O^{Qj}(t+1) = S_{FQ}^t \vee S_O^{Qj} \overline{S_{SL}^{Qj}}\ ;$$

$$S_{SL}^{Qj}(t+1) = \bigvee_{\forall j \in N} S_O^{Qj} S_U^{Qj}\ ;\ S_Q^t(t+1) = S_{SL}^{Qj} \vee S_Q^t \overline{S_{SZ}^{pj}}\ ; \qquad (1)$$

$$S_{ZPj}^t(t+1) = S_Q^t S_{SZ}^{pj} \vee S_{ZPj}^t \overline{S_{PZ}^{pj}}\ ;\ S_{GPj}^t(t+1) = S_{ZPj}^t S_{PZ}^{pj} \vee S_{GPj}^t \overline{S_{PT}^{pj}}\ ,$$

где x_Z означает наличие запроса задачи на постановку в очередь.

– для процесса «сервер», реализуемого процессором до момента рандеву:

$$S_{SZ}^{pj}(t+1) = S_S^{pj} \vee S_{SZ}^{pj} \overline{S_Q^t}\ ;\ S_{PZ}^{pj}(t+1) = S_{SZ}^{pj} S_Q^t \vee S_{PZ}^{pj} \overline{S_{ZPj}^t}\ ;$$

$$S_{PT}^{pj}(t+1) = S_{PZ}^{pj} S_{ZPj}^{t} \vee S_{PT}^{pj} \overline{S_{ZPj}^{t}}. \quad (2)$$

СКУ, описывающие события после рандеву:

$$S_A^{pj}(t+1)=S_{PT}^{pj} S_{GPj}^{t} \vee S_A^{pj} \overline{(S_E^{pj} \vee S_{PK}^{pj})};\ S_{TO}^{t}(t+1)=S_A^{pj} S_E^{pj} \vee S_{TO}^{t} \overline{S_{RT}^{pj}};$$

$$S_{RT}^{pj}(t+1)=S_A^{pj} S_E^{pj} \vee S_{RT}^{pj} \overline{S_{TO}^{t}};\ S_{OF}^{t}(t+1)=S_{RT}^{t} S_{TO}^{pj}; \quad (3)$$

$$S_S^{pj}(t+1)=S_{RT}^{t} S_{OF}^{pj}.$$

Следует иметь в виду, что выбор очереди для записи в нее вновь поступившей задачи выполняется в соответствии с выражением:

$$S_{SL}^{Qj} = \min_{(\forall j \in N)} \left\{ S_U^{Qj} \right\}. \quad (4)$$

Выражение (4) означает взятие функции минимального количества занятых мест из всего массива локальных очередей, где N – число процессоров (очередей) в многопроцессорной системе.

Структура устройства планирования с локальными очередями задач

Аппаратно устройство планирования представлено в виде трех типов блоков: блок назначения задач; N блоков очередей со схемами управления, где N – число процессоров в многопроцессорной системе; блок формирования очередей;

Блок очереди предназначен для хранения готовых к обслуживанию задач. Число этих блоков равно числу процессоров в системе. Запись в очередь производит диспетчер, по сигналам *S_wrreq,* подаваемым из блока планирования. Считывание содержимого регистра очереди производится по сигналу *S_rdreq*, поступающему от каждого процессора по индивидуальным линиям.

Входная шина данных формируется блоком планирования (*V_task_id_xz[7...0]*) и является общей для всех блоков очереди. Каждый процессор управляет только своей очередью, другими словами он может читать содержимое только из одного блока. Также каждый процессор постоянно сканирует свою очередь (семафорный сигнал *S_fifo_empty*) на предмет наличия идентификаторов задач. Если в регистрах имеется идентификатор хотя бы одной задачи, процессор читает его, если нет, то переходит в режим ожидания.

Выходная шина данных из каждого блока очереди заводится в индивидуальный процессор. Семафорный сигнал *S_fifo_full* используется блоком назначения задач для указания на заполнение очереди блока очереди. По этому сигналу диспетчер прекращает прием нового идентификатора в данную очередь.

Блок формирования очередей выбирает из всех N очередей блок очереди, в котором находится минимальное количество задач (содержащий наименьшее количество записей). Номер очереди с наименьшим числом записей передается в блок планирования по 3-х разрядной выходной шине *sel [2..0]*.

Блок назначения задач принимает новый идентификатор задачи и под действием сигналов *S_wrreq* помещает его в наименее заполненную очередь *FIFO*. Кроме того, на данный блок может возлагаться функция контроля состояния всех процессоров и функция перераспределения задач по очередям для выравнивания нагрузки на процессоры.

Моделирование алгоритма планирования задач на языке VHDL и анализ результатов

Для моделирования работы планировщика с локальными очередями была применена схема, показанная на рис. 3. Была смоделирована ситуация, когда в систему может поступить одновременно до восьми задач, причем любой процессор обслуживает одну задачу на протяжении случайного периода времени (от 6 до 11 тактов).

Частота некоторых запросов выбрана таким образом, чтобы возникали конфликты за доступ к процессору. Например, второй запрос xz[0] поступил в момент времени, когда процессор занят обслуживанием предыдущего запроса. В состоянии конфликта вновь поступившая задача устанавливается планировщиком в локальную очередь, время ожидания

в которой зависит от производительности процессора и трудоемкости задачи. Оценка таких задержек является предметом дальнейших исследований. Для этих целей авторами разрабатываются соответствующие модели массового обслуживания.

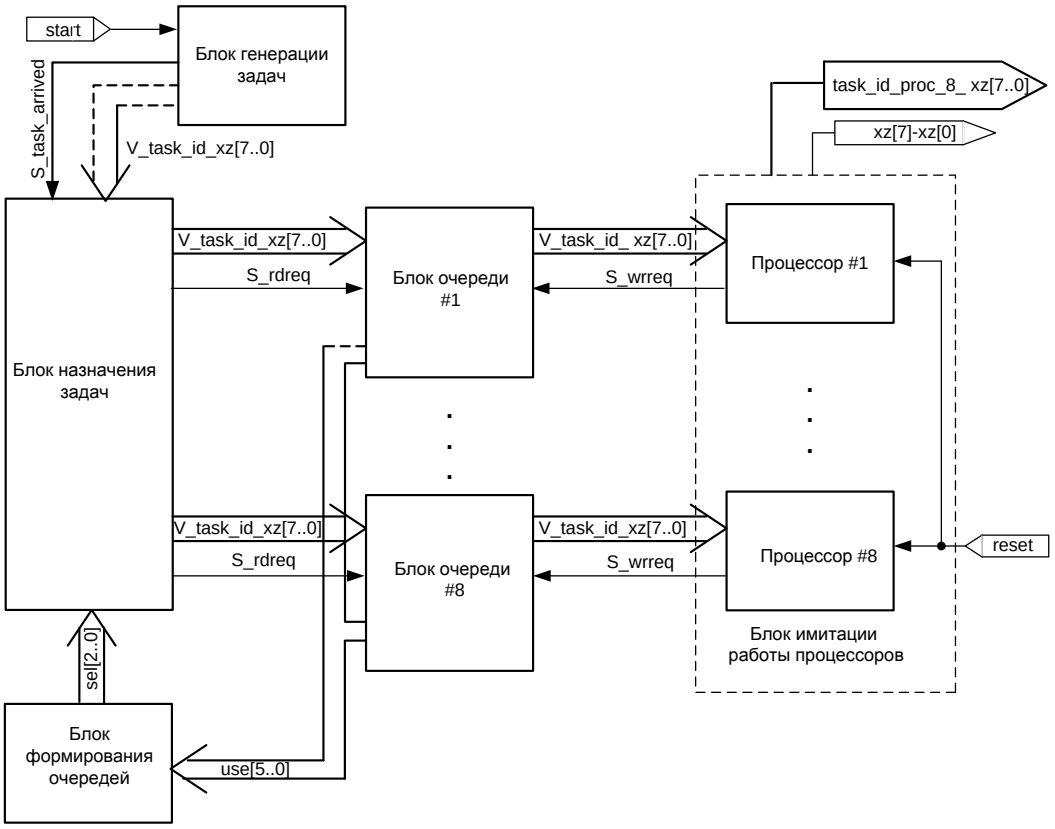


Рис. 3. Схема для проведения экспериментальных исследований

На рис. 4 приняты следующие обозначения: *xz*[7] – *xz*[0] – идентификаторы задач, поступающие на вход блоков очереди *FIFO*; *sa*[7] – *sa*[0] – обработка процессорами пользовательских задач.

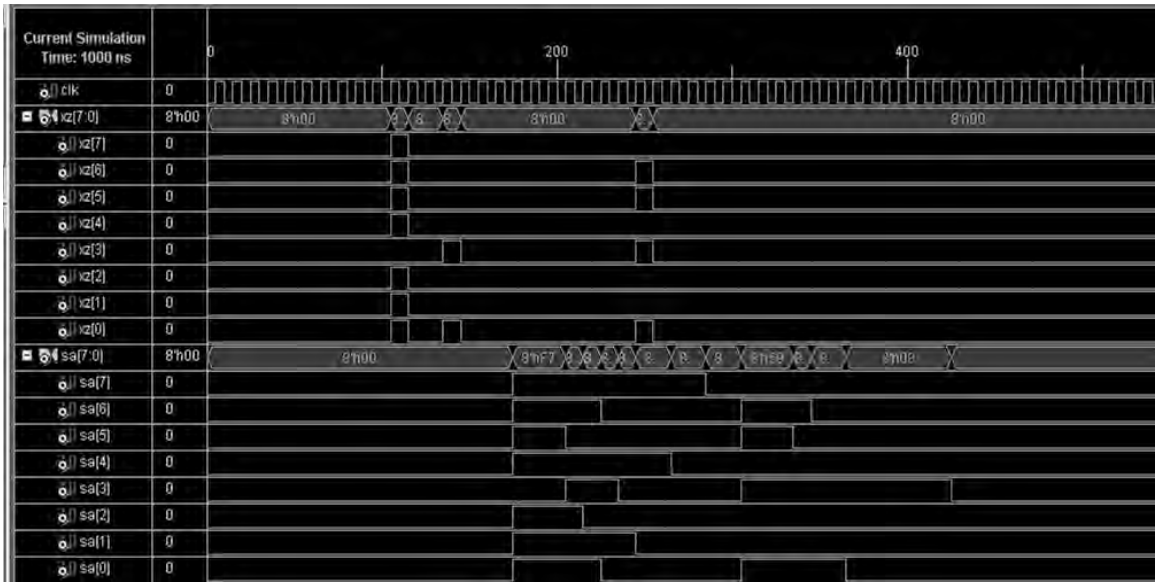


Рис. 4. Результаты моделирования

Из анализа временных диаграмм можно сделать следующие выводы. Задержка выполнения задач в условиях бесконфликтности (латентность) составляет всего 5–7 процессорных тактов. В современных ОСРВ, реализованных программно, задержка (обработка запроса) обычно составляет 5–10 тыс. процессорных тактов с учетом обработки прерывания по кванту.

Недостаток планировщика с локальными очередями заключается в отсутствии автоматической балансировки загрузки процессоров, что может приводить к неравномерности загрузки процессоров.

Библиографический список

1. Вашкевич, Н. П. Аппаратная поддержка диспетчера задач с глобальной очередью в многопроцессорных системах / Н. П. Вашкевич, Р. А. Бикташев, А. И. Меркурьев // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2011. – № 3. – С. 3–14.

2. Вашкевич, Н. П. Недетерминированные автоматы и их использование для реализации систем параллельной обработки информации : моногр. / Н. П. Вашкевич, Р. А. Бикташев. – Пенза : Изд-во ПГУ, 2016. – 394 с.

3. Вашкевич, Н. П. Формализация алгоритмов управления параллельными процессами на основе использования механизма рандеву / Н. П. Вашкевич, Р. А. Бикташев // XXI век: итоги прошлого и проблемы настоящего плюс. – 2011. – № 3 (03). – С. 115–124.

Р. А. Бикташев

Пензенский государственный университет, Пенза, Россия

МОДЕЛЬ УПРАВЛЕНИЯ ПРОЦЕССАМИ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ НА ОСНОВЕ СТОХАСТИЧЕСКИХ СЕТЕЙ МАССОВОГО ОБСЛУЖИВАНИЯ

Представлена аналитическая модель для оценки показателей производительности функций планирования и диспетчеризации многопроцессорных операционных систем.

В многопроцессорных системах (МПС) существует проблема деградации производительности вызываемой трудоемкими функциями управления процессами. Эти функции выполняют планировщики/диспетчеры задач, которые в МПС запрашиваются процессорами гораздо чаще, чем в однопроцессорных системах. При проектировании специализированных многопроцессорных операционных систем необходимо прогнозировать показатели производительности, которые чаще всего выражается через время ответа системы на запрос.

Планирование зачастую предполагает использование глобальной очереди готовых задач, которая представлена в многопроцессорной системе (МПС) в виде таблицы процессов, хранящейся в общей памяти. При переключениях процессов выбираемая из очереди задача поступает на обработку в любой свободный процессор, причем не обязательно в тот, в котором она выполнялась ранее. Поэтому в таких МПС существует явление перезагрузки кэш, когда прерванная задача с высокой вероятностью может быть направлена для продолжения обслуживания в другой процессор. Названное явление увеличивает частоту кэш-промахов и неизбежно приводит к снижению производительности многопроцессорной системы. Кроме того велика вероятность возникновения конфликтов, когда множество свободных процессоров состязаются между собой за доступ к единственной очереди задач. На разрешение конфликтов затрачивается процессорное время, что увеличивает потери производительности процессоров МПС.

Пусть МПС состоит из n статистически идентичных процессоров, планировщика, основная функция которого формирование очереди готовых задач и диспетчера, переключающего процессы (потоки). Завершенные задачи снимаются планировщиком с исполнения, поэтому считается, что планировщик выступает в качестве источника и поглотителя заявок.

Задача, заставшая все процессоры занятыми обслуживанием, ставится в очередь, иначе поступает на обслуживание в любой свободный процессор. Переключение контекста производит диспетчер, в результате чего текущая задача, если ее выполнение не закончилось, направляется в очередь, на то место, которое определяется алгоритмом планирования. Поскольку заявки назначаются на любой свободный процессор, то процессорный блок представляется в виде многоканальной СМО.

Диспетчер являются общим (разделяемым) ресурсом для всех запрашивающих процессоров, поэтому модель диспетчера представлена в виде одноканальной СМО.

Структура сетевой модели многопроцессорной системы с планировщиком/ диспетчером задач на основе глобальной очереди представлена на рис. 1. В модели принято, что запросы задач на обслуживание генерируются планировщиком с интенсивностью λ_0. Заявки назначаются в любой свободный процессоров и выполняются в течение случайного времени со средним значением $v_{ср} = 1/\mu_{ср}$, где $\mu_{ср}$ – интенсивность обслуживания в каждом из n процессоров.

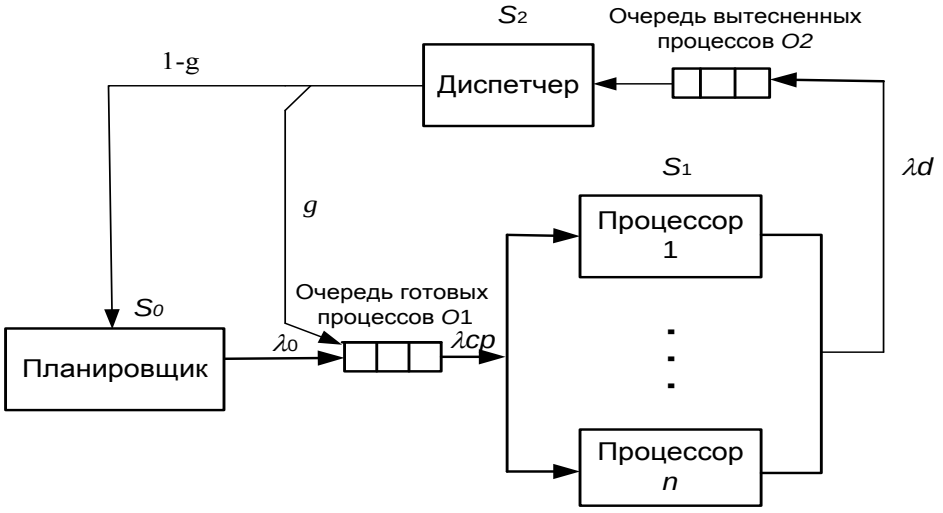


Рис. 1. Схема модели планировщика с общей очередью задач в составе многопроцессорной системы

Таким образом, обслуживание в процессорном блоке (S_1) в соответствии с нотацией Кендалла [1] представляется в виде многоканальной СМО типа $M/M/n/m$, с числом обслуживающих каналов n и ограничением длины очереди O_1 числом мест m.

Процессорный блок формирует поток запросов λ_d к диспетчеру на переключение процесса. Если диспетчер свободен, то прерывает выполнение текущей задачи в i-м процессоре, и с вероятностью g ставит ее в очередь O_1, (при наличии свободных мест). Далее диспетчер назначает i -му процессору новую задачу из очереди O_1, если она не пуста.

Если диспетчер занят, то процесс ставится в очередь (приостанавливается). В реальной системе приостановка осуществляться двумя вариантами: путем перевода процессора в спящий режим, либо путем выполнения процессором холостых команд в цикле (спиннинг).

Задача, получившая полное обслуживание, с вероятностью $(1–g)$ удаляется из сети после последнего переключения контекста, выполняемого диспетчером. Удаление проис-

ходит перемещением ее в фиктивную СМО S_0. На обслуживание в диспетчере затрачивается среднее время $v_d = 1/\mu_d$, где μ_d – интенсивность обслуживания.

Исходя из условия баланса, которое заключается в равенстве средних значений входящих и выходящих потоков для каждого узла сети, интенсивность потока задач, поступающих в очередь O_2, составляет $\lambda_{cp} = \lambda_d = \lambda_0 + g\lambda_d$.

При переключениях контекста текущая задача помещается в очередь в соответствии с алгоритмом планирования. Когда прерванной задаче очередной раз будет предоставлен процессорное время, она с вероятностью *f* будет направлена в процессор, в котором она обслуживалась ранее, а с вероятностью (1– *f*) в другой свободный процессор. Тогда с вероятностью (1– *f*) произойдет перезагрузка кэш. Если принять равновероятное распределение задач по процессорам, то *f* = 1/*n*. Если через t_c обозначить время, необходимое для перезагрузки кэш (случайная величина), а среднее время (математическое ожидание) выполнения задачи между двумя переключениями контекста (случайная величина) в процессоре – через t_t, то время обслуживания в *i*-м процессоре составит $v_{\text{ср}} = t_t + ft_c = t_t + t_c / n$.

Время обслуживания в диспетчере составит $v_d = (t_d + \tau)$, где t_d – среднее время работы программы диспетчеризации, τ – среднее время, необходимое для переключения контекста. Под временем работы программы диспетчеризации t_d, в данном случае, понимается время, затрачиваемое на выполнение системного вызова, вычисление приоритета процесса, на перераспределение очереди готовых процессов в таблице процессов, на передачу процессору адреса (идентификатора) нового процесса.

При переключениях контекста по кванту время обслуживания в *i*-м процессоре составит [2] $v_{\text{ср}} = (t_q + t_c)$, где t_c – время, необходимое для перезагрузки кэш (случайная величина); t_q – длительность кванта (постоянная величина). Время обслуживания одного процесса в диспетчере определяется выражением $v_d = (t_d + \tau)$.

Модель процессорного блока. Рассмотрим модель процессорного блока, представленная в стохастической сети многоканальной СМО S_1. СМО имеет единственную очередь задач типа *FIFO*, т.е. обслуживание происходит в порядке их поступления. Задачи, не успевшие выполниться за отведенный этап обслуживания между двумя переключениями контекста или за квант процессорного времени, вновь устанавливаются в ту же очередь на дообслуживание. Задача, поступившая в момент, когда все каналы (процессоры) заняты, становится в очередь и ожидает обслуживания. Число задач (процессов), которые одновременно выполняются в процессорном блоке $z = \lambda_{\text{ср}} / \mu_{\text{ср}}$, и может быть представлено как среднее число процессоров, занятых обслуживанием. Очередь имеет ограниченное число мест, поэтому если она заполнена, то новые задачи не принимаются на обслуживание. Этот факт моделируется потерей заявок, однако эта особенность на характеристики СМО не влияет. Обозначим число мест в очереди через *m*, тогда СМО S_1 не может вместить более *n* + *m* задач, из которых *n* находится на обслуживании, а *m* ожидают.

Определив параметры задач и параметры устройств достаточно легко можно найти единичное время пребывания задачи в процессорном блоке $u_{\text{ср}}$ [1].

Модель диспетчера. Заявки на обслуживание в диспетчере формируются процессами, выполняющимися в процессорном блоке. Если в момент поступления заявки диспетчер занят, то выполнение *i*-го процесса приостанавливается, причем приостановка моделируется очередью O_2. Если диспетчер свободен, он обслуживает задачу, на что уходит среднее время v_d.

Модель диспетчера представляется замкнутой одноканальной СМО (S_2) с ограниченным числом источников заявок на обслуживание, что по классификации Кендала представляется как *M/G*/1/*n*.

Выше определено, что в многоканальной СМО, моделирующей обслуживание в процессорном блоке, занято обслуживанием в среднем z из общего числа *n* процессоров, где z – число запущенных задач, т.е. задач, получающих процессорное время (квант про-

цессорного времени). Следовательно, z процессов (процессоров, которые обслуживают указанные процессы) являются источниками заявок для обслуживания диспетчером задач (СМО S_2). Если диспетчер свободен, то он принимает задачу (процесс) на обработку, если занят, то задача ставится в очередь. В реальной МПС очередь состоит из приостановленных процессов, которые обратились к диспетчеру задач в момент его занятости.

Пусть $\bar{z}$ – среднее число приостановленных задач (процессоров) из общего числа работающих z, тогда $(z-\bar{z})$ процессов являются источником заявок к диспетчеру, средняя интенсивность которого составляет $\lambda_d = (z-\bar{z})\cdot\lambda_1$, где λ_1 интенсивность потока задач, формируемым одним процессором. Вследствие баланса потоков на входе и выходе СМО S_2, получим: $\left(z-\overline{z}\right)\cdot\lambda_{ср} = (1-\pi_0)\cdot\mu_d$, где μ_d – интенсивность обслуживания задач диспетчером, π_0 – вероятность отсутствия запросов к СМО S_2.

Зная параметры диспетчера задач можно легко определить единичное время пребывания u_d заявки в СМО S_2 по формулам, приведенным в [1].

С учетом коэффициентов передачи сети $\varphi_{ср} = \eta_{ср} / \lambda_0$ и $\varphi_d = \lambda_d / \lambda_0$ время пребывания заявки в сети время ответа системы () определяется суммой времен пребывания заявки в процессорном блоке и в диспетчере $U = \varphi_{ср}\cdot u_{ср} + \varphi_d \cdot u_d$, где $u_{ср}$, u_d – единичные времена ответа СМО S_1 и S_2 соответственно [2]. Последнее выражение является временем ответа системы.

Библиографический список

1. Таха, Х. А. Введение в исследование операций / Х. А. Таха. – М. : Вильямс, 2005. – 912 с.

2. Бикташев, Р. А. Математическое моделирование функций управления взаимодействующими процессами и ресурсами многопроцессорных систем / Р. А. Бикташев, А. И. Мартышкин, Н. Г. Востоков // Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации. Распознавание – 2013 : сб. ст. XI Междунар. науч.-техн. конф. – Курск : ЮЗГТУ, 2013. – С. 218–220.

Р. Н. Федюнин

Пензенский государственный университет, Пенза, Россия

СПЕЦИАЛИЗИРОВАННЫЕ АРИФМЕТИКО-ЛОГИЧЕСКИЕ УСТРОЙСТВА НА БАЗЕ КОНВЕЙЕРНЫХ МАТРИЧНЫХ ВЫЧИСЛИТЕЛЕЙ

Итеративно-битовые процессоры [1, 2] ориентированы на реализацию двуместных арифметических операций и отношений равенства и неравенства значений данных относительно некоторого критерия. В ряде случаев функциональные возможности процессорных элементов обеспечивают также возможность выполнения и операций преобразования структур данных – операций сжатия двоичных векторов, формирования векторов-масок и т.д.

Концептуально организация арифметико-логического блока процессоров на базе итеративно-битовых процессорных пространств определяется функциональными возможностями процессорных модулей многомерной вычислительной среды, топологией межпроцессорных связей, способами управления настройкой и вычислениями, количеством каналов и способами ввода и вывода информации из вычислительного пространства. В связи

с этим общая организация АЛУ процессора представима структурой, представленной на рис. 1.

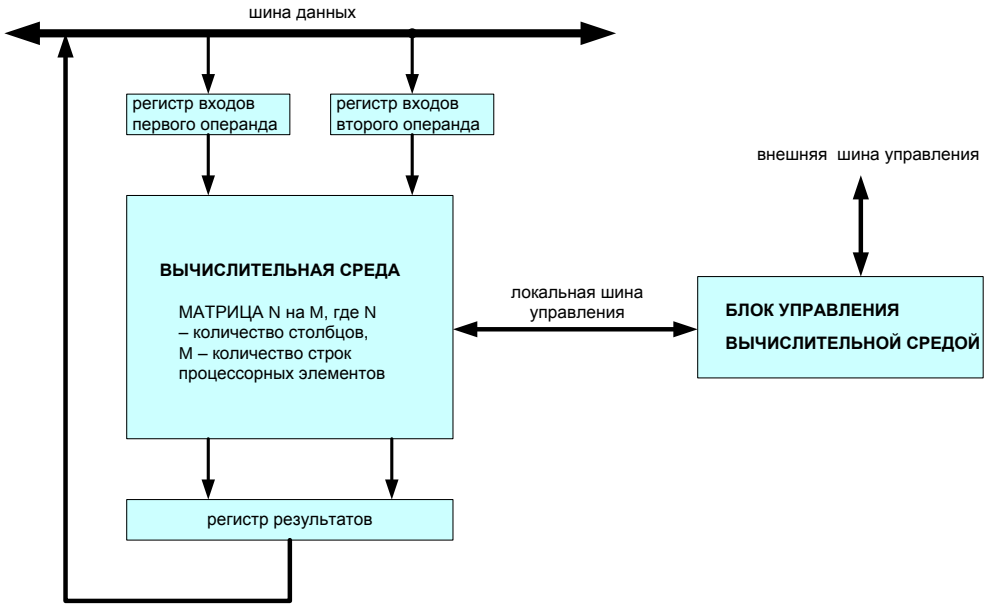


Рис. 1. Организация АЛУ итеративно-битового процессора

Вычислительная среда или как ее еще называют однородная вычислительная среда (ОВС) (рис. 1) – является операционной частью процессора и организована как матрица процессорных элементов (ПЭ). Иллюстрация организации операционной части процессора приведена на рис. 2, где 1,1–1, *m*, 2,1–2, *m*, …, *n*,1–*n*,*m* – где ячейки процессорных элементов, p_0–p_n, b_0–b_m – информационные входы, c_0–c_n, s_0–s_m – информационные выходы, PRG – управляющий вход, соединенный с управляющими входами prg, каждого процессорного элемента.

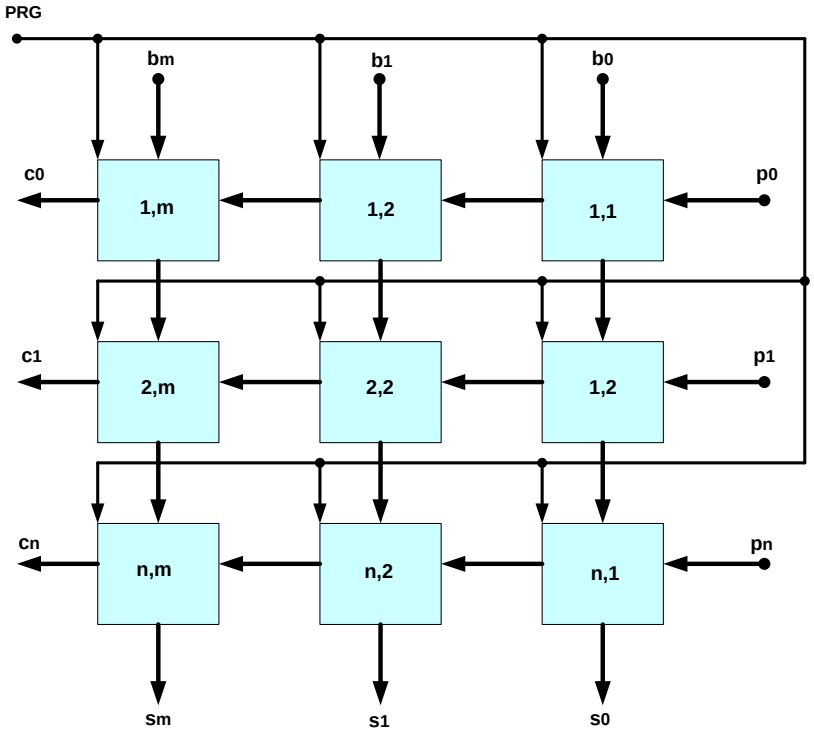


Рис. 2. Организация операционной части АЛУ процессора

Анализ методов организации вычислений в многомерных арифметико-логических процессорах показывает, что для выполнения массивных операций наиболее эффективна топология информационных межпроцессорных каналов типа «решетка».

Такая топология межпроцессорных связей обеспечивает возможность реализации арифметико-логических операций на базе простого одноразрядного процессорного модуля.

В качестве процессорного модуля вычислительного пространства для выполнения арифметико-логических вычислений над данными, представляемыми n-разрядными двоичными векторами, достаточно эффективен процессорный элемент (ПЭ) [3, 4], организация которого приведена на рис. 3.

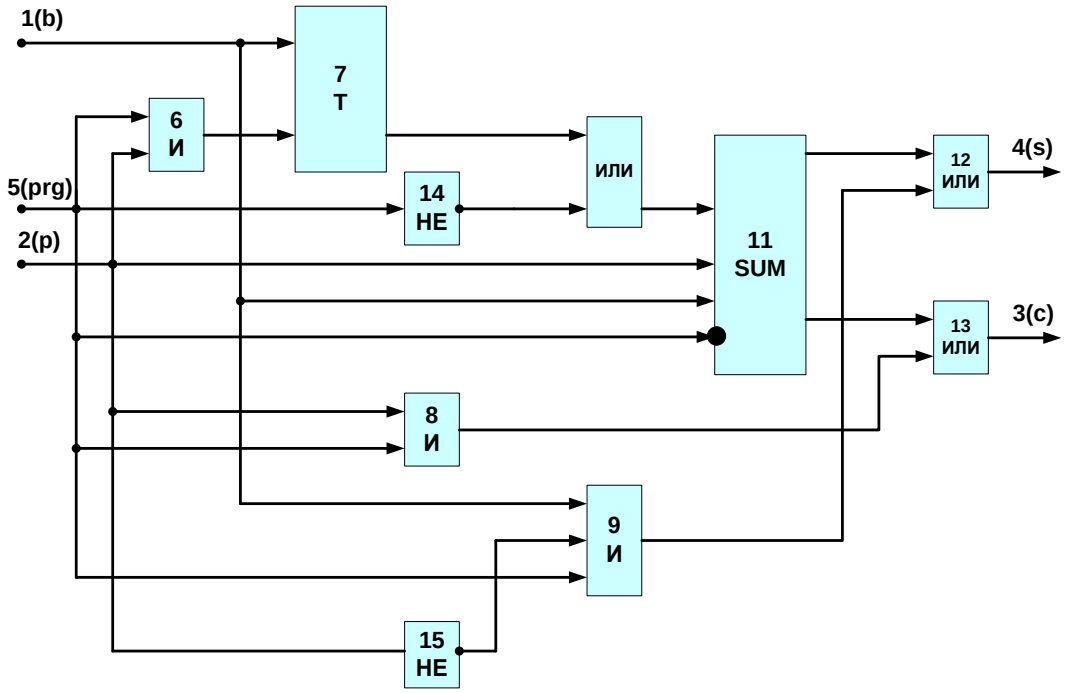


Рис. 3. Организация процессорного элемента АЛУ итеративно-битового процессора

Процессорный элемент реализует следующую систему логических функций:

$$\begin{aligned} Q(t) &= prg \cdot b \cdot p \\ s &= \overline{prg} \cdot b \cdot p \cdot Q(t) \vee prg \cdot \overline{p} \cdot b \\ c &= \overline{prg} \cdot b \cdot p \cdot Q(t) \vee prg \cdot p \end{aligned} \qquad (1)$$

где **b** и **p** – сигналы, соответственно подаваемые на информационные входы 1 и 2 ячейки;

prg – сигнал, формируемый соответственно на управляющем входе 5;

c и **s** – сигналы, формируемые соответственно на информационных выходах 3 и 4 ячейки;

Q(t) – состояние триггера 7 в момент времени *t.*

Однородная вычислительная структура позволяет выполнять построчную загрузку кода с группы информационных входов b_0–b_m в триггеры ячеек однородной структуры и реализовывать режим вычисления.

Загрузка кода, для реализации арифметической операции, осуществляется с группы входов b_0, b_1,..., b_m ОВС в триггеры ПЭ соответствующей строки по каналам коммутации b_i–s_i.

Настройка ячеек ОВС на работу в режиме загрузки кода осуществляется подачей исходного вектора на группу входов b_0, b_1, ..., b_m ОВС, при этом младший разряд исходного вектора b_0 подается на первый информационный вход первой ячейки первого столбца, первой строки (на вход b_0 ОВС), при этом на управляющий вход PRG однородной структуры подается сигнал логической единицы, а на группу входов p_0, p_1, ..., p_n ОВС подается управ-

ляющий вектор, единичный бит которого указывает строку, в которую необходимо загрузить код вектора данных. В результате вектор с группы входов b_0, b_1, ..., b_m ОВС фиксируется в триггерах ПЭ нужной строки. Заметим, что загрузка кода в ячейки ОВС осуществляется в следующем порядке: первыми загружаются ячейки самой последней строки, затем предпоследней и т.д., последними будут загружены ячейки первой строки.

ПЭ в данном режиме реализует следующую систему логических функций:

$$\begin{aligned} s &= prg \cdot \overline{p} \cdot b \\ c &= prg \cdot p \\ Q(t) &= prg \cdot p \cdot b \end{aligned} \qquad (2)$$

Пример работы однородной вычислительной структуры в режиме загрузки кода приведен на рис. 4. На рис. 4 – показана загрузка кода 100_2 в ячейки самой последней строки. Для чего на группу входов p_0-p_2 подается управляющий вектор 100_2 , в результате чего ячейки последней строки настраиваются на выполнение операции загрузки кода, а ячейки всех остальных строк – на выполнение операции трансляции кода, при подаче на управляющий вход PRG ОВС активного сигнала, равного уровню логической единицы вектор 100_2 с группы входов b_0, b_1, b_2 загружается в ячейки самой последней строки.

Затем, по аналогии, на рис. 4 – иллюстрируется загрузка кода 0112 в ячейки третьей строки, для чего на группу входов p0-p2 подается управляющий вектор 0102 а на управляющий вход PRG ОВС подается активный сигнал, равный уровню логической единицы. Далее во вторую строку ОВС загружается битовый вектор 0112, для чего на группу входов p0-p2 подается управляющий вектор 0102 а на управляющий вход PRG ОВС подается активный сигнал, равный уровню логической единицы, а в первую строку загружается вектор – 0102, при подаче на группу входов p0-p2 управляющего вектора 0012 а на управляющий вход PRG ОВС активного сигнала, равного уровню логической единицы В данном примере видно, что заполнение ПЭ кодами векторов осуществляется построчно и снизу вверх, т.е. первый вектор загружается в самую последнюю строку однородной структуры, и т.д. до полного заполнения структуры или исчерпания векторных массивов.

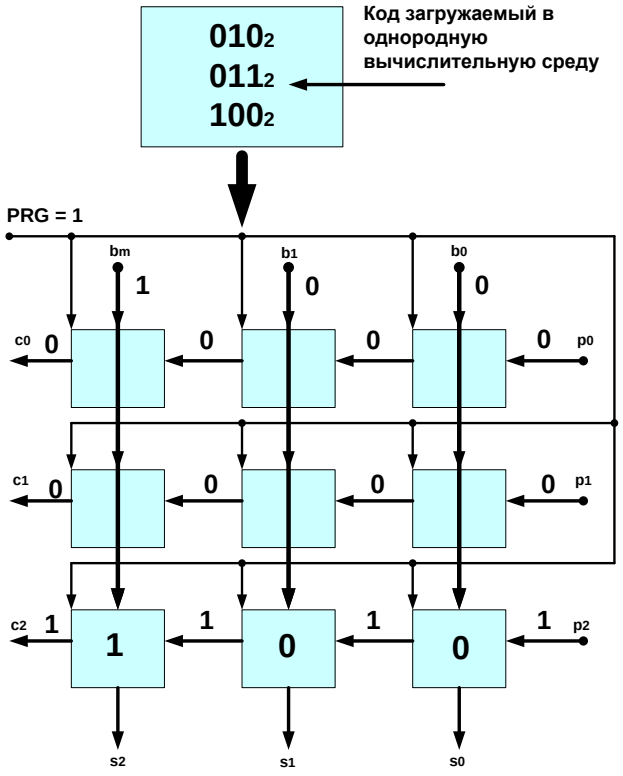


Рис. 4. Загрузка данных и процесс вычисления

После реализации подготовительного режима, устройство выполняет функцию сумматора, складывая вектора, загруженные в ПЭ ОВС в режиме загрузки.

Настройка ячеек ОВС на работу в режиме вычисления осуществляется подачей на управляющий вход PRG однородной структуры сигнала логического нуля. В результате чего в каждой ЯОС ОВС включаются цепи суммирования и реализуется следующая система логических функций:

$$\begin{aligned} s &= b \cdot p \cdot Q(t), \\ c &= b \cdot p \cdot Q(t), \end{aligned} \tag{3}$$

т.е. в каждой ячейке ОВС происходит сложение данных, хранящихся в триггере данной ячейки, с данными, поступающими в данную ячейку по входам **b** и **p**. Результат операции фиксируется на группе входов s_0–s_m.

Пример работы однородной вычислительной структуры в режиме вычислений приведен на рис. 5. На все информационные входы ПЭ ОВС подается сигнал логического нуля. В данном примере предполагается, что загрузка в ячейки ОВС необходимых для сложения векторов уже выполнена. Для начала выполнения операции сложения, на управляющий вход PRG подается сигнал равный нулю, в результате каждая ЯОС ОВС выполняет систему функций 4. Результат формируется по столбцам, и из примера видно, что если в столбце четное количество единиц то результат суммы столбца 0, иначе, если в столбце нечетное число единиц, то результат суммы столбца 1, при этом также учитывается формирование переносов из младших в старшие разряды числа. Результат вычислений фиксируется на группе информационных выходов s_0–s_6.

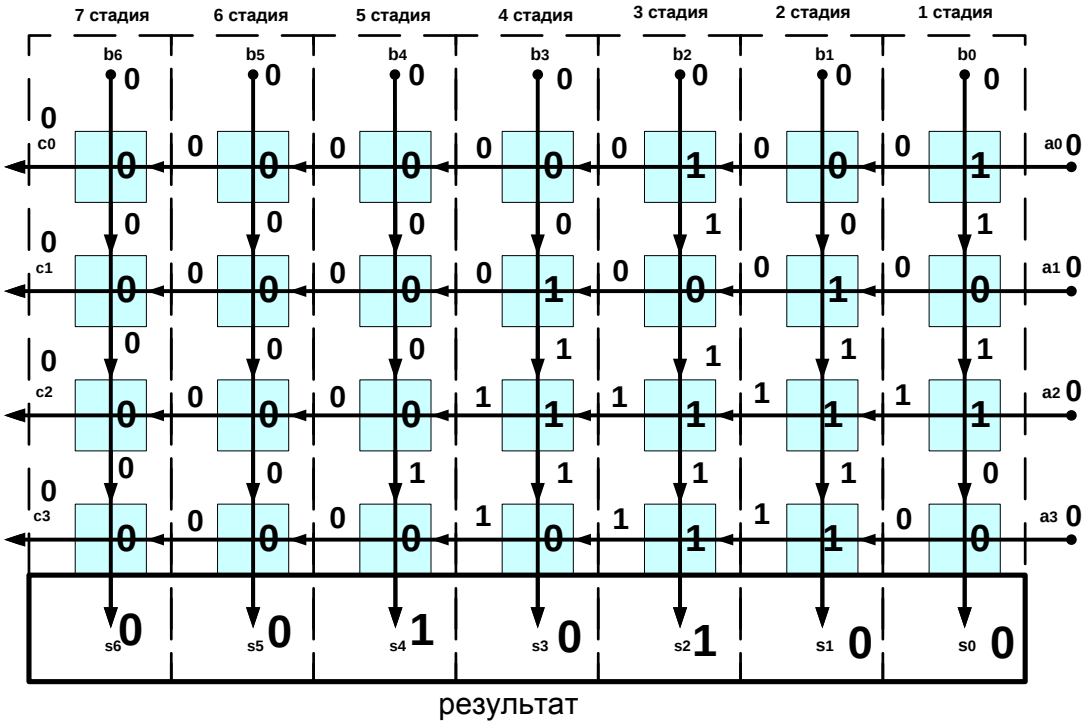


Рис. 5. Формирование результата

Таким образом, рассмотрев частный пример построения однородной вычислительной структуры, можно сделать следующие выводы.

Итеративно-битовая архитектура вычислительного устройства является решением, агрессивно эксплуатирующим параллелизм «конвейерного» типа. Несмотря на принципиальную дороговизну решения (применение специализированных аппаратных решений), систолы (ПЭ) являются одним из самых эффективных методов решения специализированных задач, так как часто обеспечивают на конкретной задаче быстродействие, недостижимое

для большинства микропроцессорных архитектур. Значительное удешевление программируемых логических матриц [5, 6] сводит на нет экономическую составляющую вопроса применения и пробуждает интерес исследователей к применению ОВС в сложных вычислительных задачах.

Итеративно-битовые системы являются очень специализированными вычислителями и производятся в основном под конкретную задачу. Фактически, задача построения вычислителя сводится к построению аппаратного конвейера, имеющего достаточно большое время получения результата (т.е. большое количество ступеней) но при этом сравнительно маленькое время между последовательной выдачей результатов, так как значительное количество промежуточных значений обрабатывается на разных ступенях конвейера.

Библиографический список

1. Фрумкин, А. И. Систолические вычисления / А. И. Фрумкин. – М. : Высш. шк., 1988. – 457 с.
2. Федюнин, Р. Н. Устройства конвейерно-параллельной обработки информации / Р. Н. Федюнин. – М. : Научное обозрение, 2005. – С. 72–80.
3. Пат. РФ № 2004136518/09 Ячейка однородной вычислительной среды / Федюнин Р. Н. Князьков В. С. ; от 12 мая 2006 г.
4. Пат. РФ № 2004136518/09 Ячейка однородной среды / Князьков В. С., Федюнин Р. Н. ; от 15 мая 2006 г.
5. URL: www.xilinx.com
6. URL: www.altera.com

И. А. Беззатеев[1], В. Н. Дубинин[1], Д. А. Уваров[1], В. В. Вяткин[2]

[1] Пензенский государственный университет, Пенза, Россия
[2] Технический университет Aalto University, г. Хельсинки, Финляндия

РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННЫХ СЕМАФОРОВ В АРХИТЕКТУРЕ IEC 61499 НА ОСНОВЕ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ КОНСЕНСУСА В СЕТИ НЕНАДЕЖНЫХ ПРОЦЕССОВ

Предложен подход к реализации распределенных семафоров в архитектуре IEC 61499 на основе алгоритмов для решения задач консенсуса в сети ненадежных процессов, что позволяет разрабатывать управляющие функционально-блочные приложения со сложными видами взаимодействий, основанных на разделении общих ресурсов.

Международный стандарт IEC 61499 определяет концепцию и системную модель для построения распределенных управляющих и измерительных систем нового поколения [1]. Для взаимодействия между ресурсами и устройствами системы в данном стандарте используется механизм передачи сообщений посредством коммуникационных функциональных блоков (ФБ) CLIENT/SERVER и PUBLISH/SUBSCRIBE. Данный механизм является единственным и довольно низкоуровневым, что не позволят напрямую строить распределенные системы со сложными видами взаимодействий, включая критические секции, «производитель-потребитель», «читатели-писатели» и т.д., которые во многом ориентированы на взаимодействие через общую память и основаны на использовании семафоров [2]. Простой, но в то же время показательный пример использования семафора в управляющих системах – это взаимоисключающая работа нескольких ресурсов (или устройств) IEC

61499 с общим технологическим оборудованием, данными или коммуникационной сетью.

Можно выделить два подхода к реализации семафоров в распределенной системе – централизованный и децентрализованный. В работе [3] была предложена клиент-серверная реализация семафора в архитектуре IEC 61499. Однако, данная реализация, где сам семафор реализован как сервер, фактически является централизованным решением, имеющим недостатки, присущие всем централизованным системам – низкую надежность и производительность, плохую масштабируемость.

В последнее время все большее внимание исследователей привлекают методы групповых взаимодействий в распределенных системах, в частности алгоритмы (протоколы) достижения консенсуса между компонентами распределенной сети – Paxos [4] и Raft [5]. Семейство протоколов Paxos используется в распределенной службе блокировок Chubby компании Google, используемой, в свою очередь, в BigTable (проприетарная высокопроизводительная база данных). Microsoft использует Paxos в своей системе автоматического управления кластерами. Алгоритм Raft относительно молод, но уже находит применение в крупных IT-проектах. Facebook использует NoSQL базу данных HydraBase в механизме сообщений, в основе которой лежит кластер, поверх которого развернут Raft.

Основываясь на факте, что семафор это фактически разделяемая булева или целочисленная переменная с определенными на ней операциями, можно предположить, что протоколы Paxos и Raft могут вполне успешно использоваться при реализации распределенных семафоров. Немногочисленные публикации подтверждают это (например, [6]).

Протоколы семейства Paxos гарантируют три следующих показателя: нетривиальность, консистентность и живучесть. При этом рассматриваются варианты решения задачи консенсуса в зависимости от количества вычислителей, количества задержек для получения результата, активности участников, количества отправленных сообщений и типов отказов. В протоколе Paxos узлы распределенной системы имеют следующие роли: Client, Proposer, Acceptor, Learner. Базовый алгоритм Paxos состоит из следующих этапов: Prepare («предложение»), Promise («обещание»), Accept («принять»), Accepted («принято»).

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. Каждый узел кластера Raft имитирует копию конечного автомата, а цель протокола – поддерживать согласованность между копиями. Репликация позволяет системе продолжать обслуживать клиентов всякий раз, когда доступно большинство машин. Однако сохранение согласованности между репликами затруднено при наличии асинхронности, сбоев сети (падений пакетов, дублирования и переупорядочения) и сбоев узлов (сбоев и перезагрузок). К особенностям Raft относятся: 1) четкое разделение фаз; 2) наличие явно выделенного лидера; 3) возможность динамического изменения размера кластера.

В данной работе разработана библиотека функциональных блоков (ФБ) для реализации алгоритмов достижения консенсуса Paxos и Raft, позволяющая существенно расширить возможности проектирования распределенных управляющих приложений со сложными видами взаимодействий. Данная библиотека реализована в системе NxtStudio [7]. Ввиду того, что функционально-блочная реализация протоколов Paxos и Raft довольно объемная, описание этой реализации приводится лишь в ограниченном объеме.

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. 1.

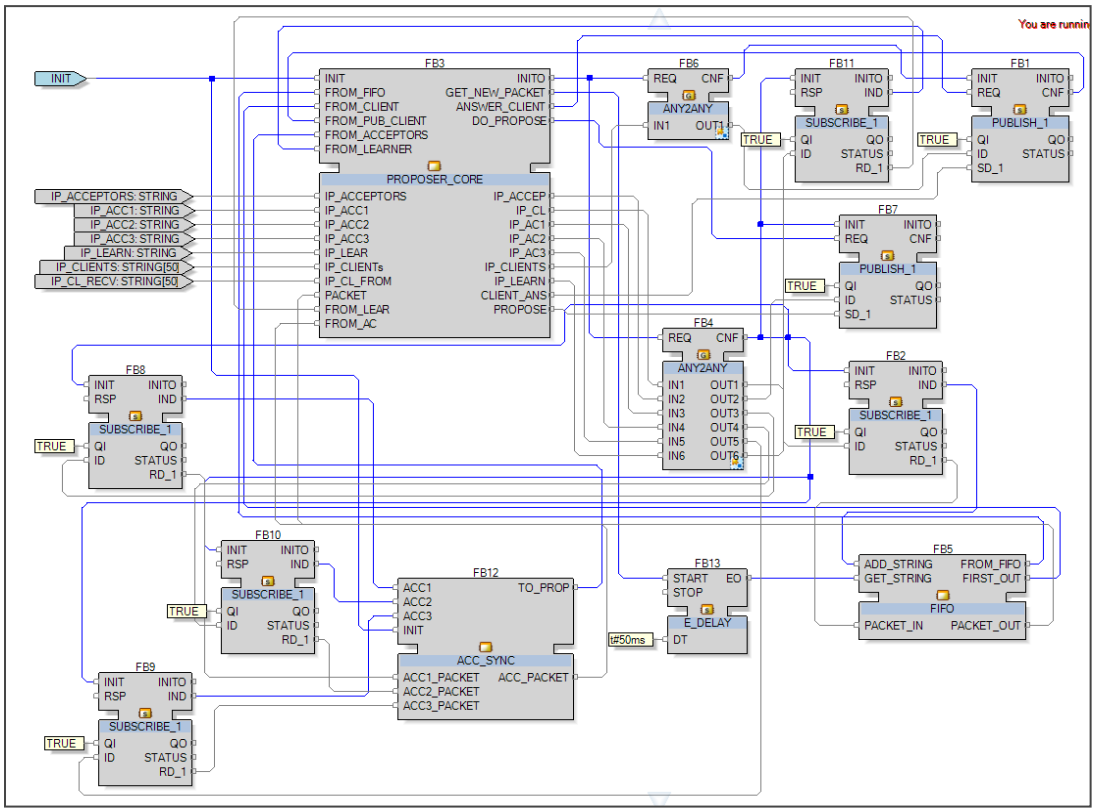


Рис. 1. Структура составного ФБ PROPOSER

Спроектированная в среде NxtStudio система «Распределенный семафор на основе Paxos» корректно функционирует при обработке запросов на захват/освобождение семафора от нескольких клиентов. 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, входящим в среду исполнения NxtStudio.

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. 2.

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.

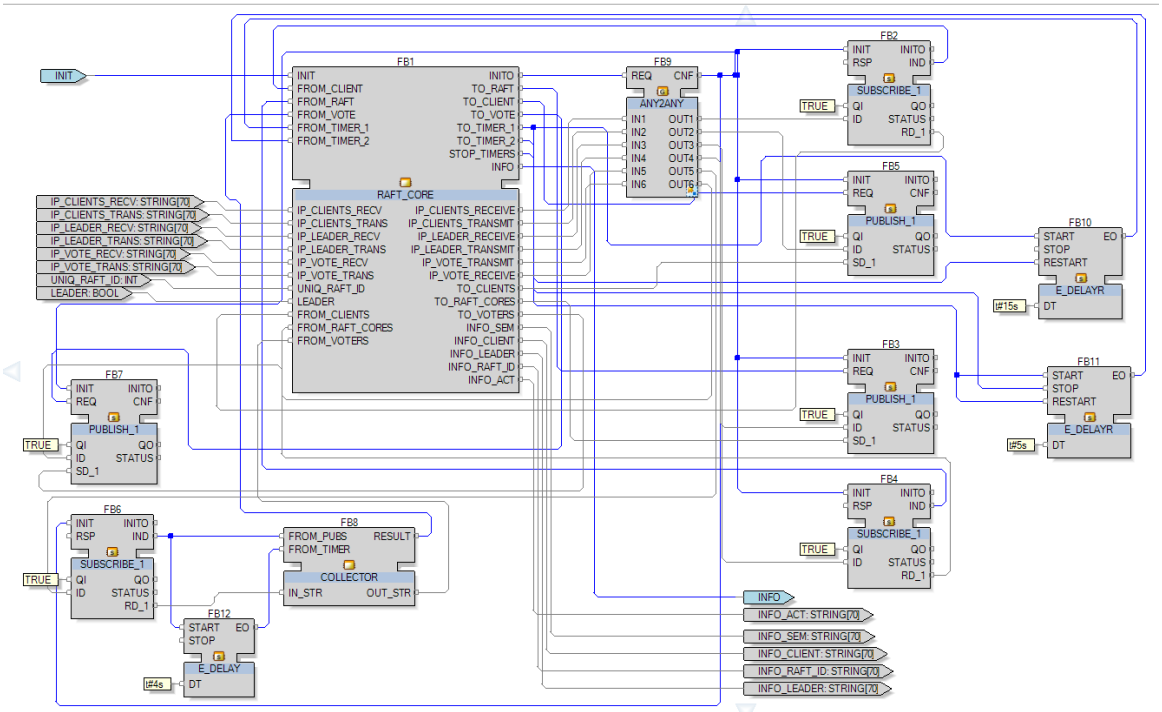


Рис. 2. Структура составного ФБ RAFT

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.

Таким образом, распределенный семафор, реализованный на основе протокола Raft, обладает следующими преимуществами: 1) простота алгоритма и его реализации; 2) возможность масштабирования системы с минимальными трудозатратами; 3) более высокая производительность.

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 баз данных в системах управления на основе ФБ.

Следует также отметить, что были построены сетевые модели распределенных семафоров на основе Paxos и Raft с использованием цветных временных сетей Петри, а также проведена серия имитационных экспериментов в системе CPN Tools. Направлением дальнейших исследований является функционально-блочная реализация других механизмов синхронизации и взаимодействий, применимых в распределенных управляющих системах.

Библиографический список

1. Vyatkin V. IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design, 3rd edn. / V. Vyatkin. – Pittsburgh : Instrumentation Society of America, 2016. – 261 p.

2. Грегори, Р. Э. Основы многопоточного, параллельного и распределенного программирования : пер. с англ. / Р. Эндрюс Грегори. – М. : Вильямс, 2003. – 505с.

3. Dubinin V. N. Implementation of some kinds of interactions and communications in IEC 61499 architecture / V. N. Dubinin // Новые информационные технологии и системы (НИТиС'2004) : тр. VI Междунар. науч.-техн. конф. (Пенза, Россия, 17–19 июня). – Пенза, 2004. – Ч. 2. – С. 84–92.

4. Lamport, L. Fast Paxos / L. Lamport // Distributed Computing. – 2006. – № 19 (2). – P. 103.

5. Planning for Change in a Formal Verification of the Raft Consensus Protocol / D. Woos, J. R. Wilcox, S. Anton, Z. Tatlock, M. D. Ernst, T. Anderson. – University of Washington, 2016. – 12 с.

6. Yethadka, S. A Distributed Lock Manager Using Paxos: Design and Implementation of Warlock, a Consensus Based Lock Manager / S. Yethadka. – Master thesis, Uppsala University, Department of Information Technology, 2013. – 82 p.

7. nxtStudio. – URL: http://www.nxtcontrol.com

Д. В. Сафронов, А. В. Дубравин

Пензенский государственный университет, Пенза, Россия

АВТОМАТИЗАЦИЯ ПОИСКА ИЗБЫТОЧНЫХ ФУНКЦИОНАЛЬНЫХ ОБЪЕКТОВ В ИСПОЛНИМЫХ ФАЙЛАХ ДЛЯ ПЛАТФОРМЫ MICROSOFT .NET

Рассмотрены вопросы поиска избыточных функциональных объектов в исполнимых файлах для платформы Microsoft .NET, проводимого в рамках исследования программного обеспечения на наличие недекларированных и опасных возможностей.

Руководящий документ ФСТЭК РФ [1] предписывает проверку на наличие избыточных функциональных объектов всего программного обеспечения, используемого на защищенных рабочих местах при обработке информации, содержащей гостайну, и устранение найденной избыточности. Устранение избыточности с целью повышения защищенности информации актуально также и для информационных систем предприятий, не связанных с гостайной, но в которых обрабатывается информация, составляющая коммерческую тайну, и конфиденциальная информация.

Таким образом, актуальной является автоматизация поиска неиспользуемых функциональных объектов в исполнимых файлах для платформы Microsoft .NET.

Под неиспользуемыми функциональными объектами будем понимать методы управляемых классов, вызовов которых (прямых, виртуальных, косвенных, посредством отражения) не содержится в некоторой совокупности исполнимых файлов, которую назовем исследуемым ПО. К неиспользуемым функциональным объектам отнесем также управляемые классы, от которых не наследуется ни одного другого класса, которые не инстанциируются в процессе выполнения исследуемого ПО, и у которых не вызывается ни одного статического метода.

Неиспользуемые функциональные объекты (ФО) могут образовываться в программном коде следующими путями:

– добавление классов и методов для полноты программного интерфейса;

– классы и методы, используемые в специальных сборках (например, отладочных);

– классы и методы, которые использовались ранее.

При компиляции в машинный код во многих случаях избыточный код может быть удален компилятором на стадии оптимизации. При компиляции под управляемые платформу сохраняются все функциональные объекты. При этом сохраняется также информация о типах и сигнатурах этих объектов – метаданные. Это позволяет в процессе выполнения разобрать состав исполнимого файла, выделить определенные типы и методы (например, реализующие заданный интерфейс, имеющие заданное название либо сигнатуру) и вызвать метод либо создать объект. Такие обращения могут осуществляться не только из самого исполнимого файла, но и из других файлов, в том числе, создаваемых позднее. В сре-

де Microsoft .NET механизм, позволяющий разбирать состав исполнимого файла во время выполнения и осуществлять вызовы, называется отражением (Reflection).

Обращения могут производится также из исполнимых файлов, не входящих в состав программного продукта. Так, например, присутствующим в исполнимом файле отладочным кодом могут воспользоваться вредоносные программы (malware).

На данный момент имеется ряд программных продуктов, предназначенных для восстановления и анализа структуры и алгоритмов исполнимых файлов для платформы Microsoft .NET (сборок).

- Дизассемблер ILDasm, разработанный Microsoft, позволяет изучать сборки в интерактивном режиме, представляя структуру файла и алгоритмы отдельных методов, восстановленные до уровня промежуточного языка (MSIL). Навигация возможна только по структуре файла, возможности анализа не предусмотрены. Имеется возможность сохранения всего дизассемблированного кода сборки. Одновременная работа возможна только с одним файлом.
- Декомпиляторы RedGate.NET Reflector и JetBrains dotPeek восстанавливают алгоритмы методов до записи на языке высокого уровня, поддерживают одновременную работу с несколькими файлами. Содержат средства навигации по коду на основе связей между сущностями, интерактивные средства анализа использования функциональных и информационных объектов (переменных, аргументов и полей данных). Декомпиляторы позволяют сохранить восстановленный код в виде проекта, пригодного – как правило, после некоторых доработок – для повторной компиляции.

Имеющиеся средства ориентированы в первую очередь на ручной анализ. Только .NET Reflector позволяет расширять свою функциональность посредством дополнительных модулей – плагинов.

Таким образом, программные средства, позволяющие автоматизировано анализировать состав сборок в соответствии с заданными критериями, отсутствуют.

Функциональный объект-метод класса является используемым, если в коде присутствуют обращения к этому методу одного или нескольких следующих видов:

- прямой вызов статического либо экземплярного метода, в том числе виртуального;
- вызов виртуального метода базового класса (интерфейса), если в коде создается объект рассматриваемого класса и этот объект может появляться в месте вызова;
- получение делегата для метод.

Функциональный объект-класс является используемым, если в коде присутствует:

- создание объекта этого класса;
- вызов статического метода этого класса;
- некоторый метод, принимающий либо возвращающий объект этого класса;
- некоторый тип либо метод, обобщенным параметром (generic) которого является данный класс;
- унаследованный от него другой класс.

Функциональные объекты, не подходящие ни под один из этих критериев, являются неиспользуемыми. Также неиспользуемыми считаются функциональные объекты, используемые только в таких функциональных объектах, которые в свою очередь являются неиспользуемыми. Таким образом, процесс выявления неиспользуемых – избыточных – функциональных объектов является итеративным.

Специальным видом использования является создание объектов и вызов методов посредством механизма отражения (Reflection). При этом, если используемые функциональные объекты идентифицируются именами, факт такого использования может быть детектирован автоматически. Если же функциональные объекты выбираются из некоторого множества в соответствии с определенными критериями, связь их с объявлением в коде не может быть установлена автоматически. В этом случае для обозначения неизбыточности функционального объекта требуется использование дополнительных маркеров. В сборках

для платформы .NET в качестве таких маркеров могут выступать настраиваемые атрибуты (Custom Attributes) специального типа.

В соответствии с вышеизложенным, разработано программное средство автоматизированного анализа исполнимых файлов для платформы Microsoft .NET, которое применяется при проведении тематических исследований.

В качестве входных данных используется группа исполнимых файлов исследуемого ПО, а результатом является перечень избыточных функциональных объектов.

ПО учитывает возможность использования функционального объекта только в модулях, внешних по отношению к тому, в котором этот объект определяется. Такая ситуация характерна, в частности, для библиотек программного кода.

На основании полученного перечня разработчик может удалить избыточные сущности либо провести иную реструктуризацию кода.

В дальнейшем планируется добавление модуля анализа зависимостей, что позволит упростить работу аналитика и существенно сократить объем ручного анализа.

Библиографический список

1. Руководящий документ. Защита от несанкционированного доступа к информации. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей // ФСТЭК России. – 1999. – Ч. 1. – URL: http://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty/114-spetsialnye-normativnye-dokumenty/382-rukovodyashchij-dokument-prikaz-predsedatelya-gostekhkomissii-rossii-ot-4-iyunya-1999-g-n-114

Н. П. Вашкевич, П. Ю. Мясин, С. Ю. Поздняков

Пензенский государственный университет, Пенза, Россия

СТРУКТУРА МНОГОПРОЦЕССОРНОЙ СИСТЕМЫ С АППАРАТНЫМ ПЛАНИРОВЩИКОМ

Рассмотрим многопроцессорную систему, в которой используется планировщик с разделением времени, диспетчер задач на основе глобальной очереди.

Рассмотрим многопроцессорную систему, в которой используется планировщик с разделением времени [2], который использует глобальную очередь задач типа FIFO, разделяемую всеми процессорами (рис. 1). Вновь поступившая задача (процесс) помещается планировщиком в конец очереди, а задачи, находящиеся в начале очереди, являются кандидатами на выполнение. Когда один из процессоров освобождается от текущей работы, он обращается к диспетчеру, который выбирает из «головы» очереди готовую к выполнению задачу и работает с ней до ее завершения или до момента блокирования, например, вследствие необходимости выполнения операции ввода-вывода.

Алгоритм планирования с разделением времени и глобальной очередью задач представляет наиболее простой и одновременно эффективный способ планирования, поскольку обладает рядом достоинств [1]:

– загрузка распределяется равномерно между процессорами, обеспечивая отсутствие простоев процессоров при наличии готовых к выполнению задач;

– простота функционирования, заключающаяся в том, что, когда процессор освобождается, он вызывает функцию назначения задач (диспетчер) из операционной системы.

Эти достоинства обеспечивают преимущественное применение такого способа планирования в многопроцессорных системах.

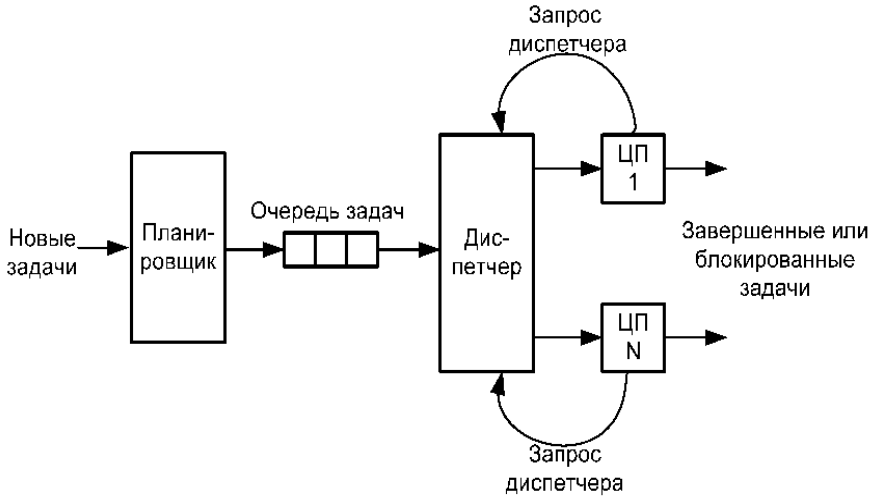


Рис. 1. Модель многопроцессорной системы, реализующей планировщика и диспетчера с глобальной очередью задач

Традиционно планировщик и диспетчер задач выполняются программно и реализуются в SMP-системах методом вызова этих функций из общей памяти, в которой резидентно хранится программа операционной системы. Практическая программная реализация предполагает два варианта:

– в пространстве пользователя;

– в пространстве ядра.

Программная реализация в пространстве пользователя хотя и является быстрой, однако вызывает затруднения при программировании, поскольку для выполнения сопряженной с диспетчеризацией процедуры синхронизации требуется три семафора: один – счетчик – для подсчета числа мест, занятых готовыми для выполнения процессами; другой – счетчик – для подсчета числа процессоров, занятых обслуживанием; и третий – мьютекс – для реализации взаимного исключения, предотвращающего одновременный доступ нескольких свободных процессоров к единственной очереди, которая в данном случае играет роль общего ресурса. Эти обстоятельства могут привести к трудно выявляемым ошибкам непредсказуемым последствиям при работе операционной системы, особенно реального времени.

Очевидным выходом из этого положения является реализация процедуры планирования/диспетчеризации в пространстве ядра [2, 3], однако возрастающие при этом временные затраты на выполнение системных вызовов резко уменьшают производительность многопроцессорной системы.

Одним из путей повышения производительности многопроцессорной операционной системы является аппаратная реализация функций планирования/диспетчеризации [4–6], которая не только снимает проблему временных потерь, но и позволяет увеличить надежность операционной системы, что особенно важно для систем реального времени.

Предлагаемый подход заключается в том, что планировщик и диспетчер задач выполняются в виде независимого аппаратного устройства (специализированного процессора) в составе многопроцессорной системы (рис. 2).

Планировщик создает очередь, состоящую из идентификаторов задач, в соответствии с заданным правилом (FIFO, LIFO, приоритетная очередь) и осуществляет выборку задачи из очереди. Идентификатор задачи однозначно указывает на адрес задачи в памяти вычислительной системы. Традиционно очередь процессов хранится в виде блоков управления процессами (БУП) или *Process Control Block (PCB)* в общей памяти системы по виртуальному адресу либо в специализированной памяти устройства управления процессами (задачами).

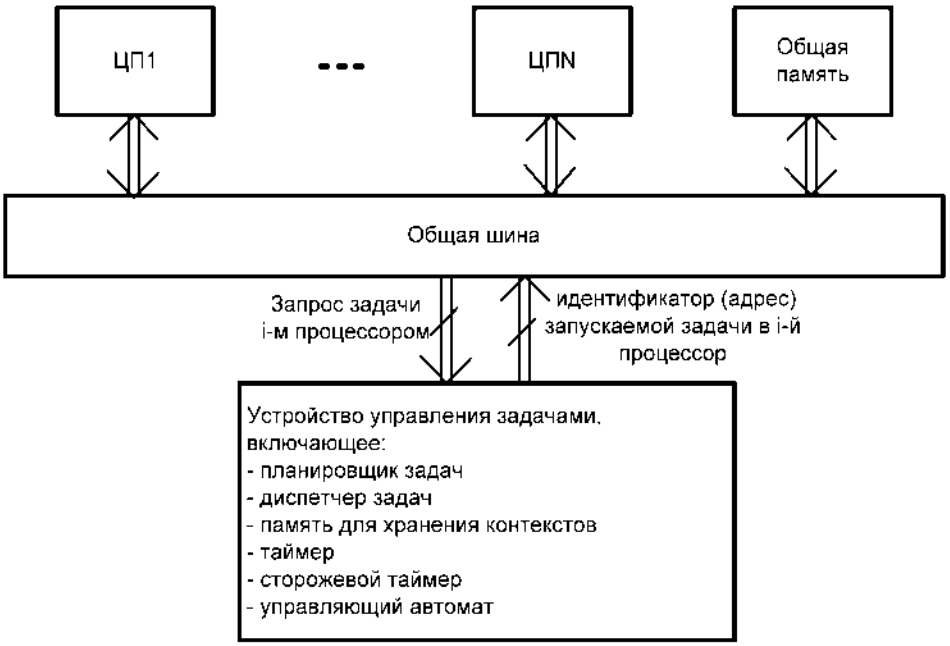


Рис. 2. Схема включения устройства управления задачами в многопроцессорную систему

Диспетчер осуществляет переключение задач (процессов) при прерываниях программ, например, от таймера. Результатом обработки прерывания является сохранение контекста текущей задачи и восстановление контекста задачи, выбираемой из очереди в соответствии с заданным ее типом (FIFO, LIFO, приоритетная). Традиционно контекст хранится в оперативной памяти в виде сегмента состояния процесса (ССП или TSS) и включается в состав БУП.

Таймер предназначен для формирования отметок времени, т.е. квантов процессорного времени, выделяемых каждой задаче из очереди готовых процессов, по которым производится переключение процессов.

Сторожевой таймер отслеживает задачи, которые занимают процессоры на неопределенное время, например, вследствие зацикливания (взаимной блокировки процессов или клинча).

В разработанном аппаратном устройстве управления процессами очередь, создаваемая планировщиком, представляет собой набор из 256 регистров, каждый из которых хранит идентификатор процесса (потока), приоритет процесса, счетчик времени и другую информацию (аналог БУП), необходимую для планирования и диспетчеризации. Каждая процесс имеет собственный идентификатор, который однозначно указывает на адрес задачи в памяти вычислительной системы. Такой подход позволяет значительно ускорить переключение процессов по окончании выделенного кванта времени.

Библиографический список

1. Дейтел, Х. М. Операционные системы. Основы и принципы / Х. М. Дейтел, П. Дж. Дейтел, Д. Р. Чофнес. – М. : Бином-Пресс, 2011. – Ч. 1. – 1024 с.

2. Таненбаум, Э. Современные операционные системы / Э. Таненбаум. – 3-е изд. – СПб. : Питер, 2010. – 1120 с.

3. Эндрюс, Г. Р. Основы многопоточного и распределенного программирования : пер. с англ. / Г. Р. Эндрюс. – М. : Вильямс, 2003. – 512 с.

4. Kuacharoen, P. A configurable hardware scheduler for Real-Time systems / P. Kuacharoen, M. A. Shalan, V. J. Mooney III // In Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms. – Las Vegas, 2003. – P. 96–101.

5. Kohout, P. Hardware support for realtime operating systems / P. Kohout, B. Ganesh, B. Jacob // In Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. – Newport Beach, CA, USA : ACM, 2003. – P. 45–51.

6. Бикташев, Р. А. Аппаратная поддержка диспетчеризации задач в многопроцессорных системах / Р. А. Бикташев, А. И. Меркурьев // Современные методы и средства обработки пространственно-временных сигналов : сб. ст. VIII Всерос. науч.-техн. конф. – Пенза : Приволжский Дом знаний, 2010. – С. 112.

7. Вашкевич, Н. П. Недетерминированные автоматы и их использование для реализации систем параллельной обработки информации : моногр. / Н. П. Вашкевич, Р. А. Бикташев. – Пенза : Изд-во ПГУ, 2016. – 394 с.

А. В. Калачев

Пензенский государственный университет, Пенза, Россия

РАЗРАБОТКА БАЗОВОГО ФУНКЦИОНАЛЬНОГО БЛОКА УПРАВЛЕНИЯ ДЛЯ СИСТЕМЫ ПЕРЕМЕЩЕНИЯ БАГАЖА В СИСТЕМЕ NXTCONTROL

Разработан базовый блок в соответствии с международным стандартом IEC 61499 для системы управления перемещением багажа с двумя выталкивателями и датчиками состояний, описана диаграмма управления выполнением для разработанного блока, описан его интерфейс взаимодействия с другими компонентами системы, разработаны алгоритмы управления выходными сигналами блока в среде автоматизированного проектирование nxtControl.

В современном мире множество рутинных операций заменяется автоматизированными системами с частичным участием человека или полностью управляемыми компьютерами и роботами.

Оборудование с каждым днем становится быстрее, надежнее, наделяется интеллектуальными функциями управления, благодаря которым система самостоятельно способна оценивать окружающую обстановку и принимать соответствующие решения о взаимодействии с другими объектами и собственной реконфигурации.

Важной особенностью модернизации и замены устаревшего оборудования на более новое, является его совместимость с предыдущими поколениями. На сегодняшний день не все отрасли выработали и перешли на единые стандарты промышленного оборудования обратной совместимости.

В результате модернизации оборудования снижаются издержки производства, затраты на энергетические и экономические ресурсы, что делает целые отрасли более эффективными чем это было ранее. Одной из сфер применения функциональных блоков международного стандарта IEC 61499 являются разработка систем управления при транспортировке и перемещении различных объектов на производстве [1].

К сожалению, не все существующие системы придерживаются международных стандартов, которые позволяют эффективно заменять устаревшее оборудование на более современное т.к. они не используют механизм обратной совместимости. Что приводит к огромным финансовым и трудовым потерям при реконфигурации и замене целых комплексов систем вместо того, чтобы заменить нескольких устаревших модулей на более новые.

В связи с этим актуальным считается разработка автоматизированной системы управления перемещением багажа с использованием международного стандарта IEC 61499 [2], которая в дальнейшем может использоваться в транспортных компаниях, аэропортах, железнодорожных станциях и других областях.

Целью научного исследования является разработка базового функционального блока в соответствии с международным стандартом IEC 61499 для системы управления перемещением багажа с двумя выталкивателями и датчиками состояний, описание ECC диа-

граммы управления выполнением (execution control charts) для базового блока управления, описание его интерфейса взаимодействия с другими компонентами системы, реализация алгоритмов управления выходными сигналами блока в среде автоматизированного проектирование nxtControl.

Полная система управления заданной системой не может быть описана в данной работе, поэтому на примере создания базового блока управления движением двух выталкивателей и функционирования набора датчиков, отвечающих за определение состояний оборудования будет рассмотрен процесс проектирования и реализации других базовых компонентов интеллектуальной системы управления.

Создание базового функционального блока начинается с того, что в созданный проект системы управления на один из ресурсов, например, “RES1”, устройства “Panel_DEV2” добавляется стандартный базовый блок стандарта IEC 61499. Его внешний вид в среде разработки nxtControl представлен на рис. 1 [3].

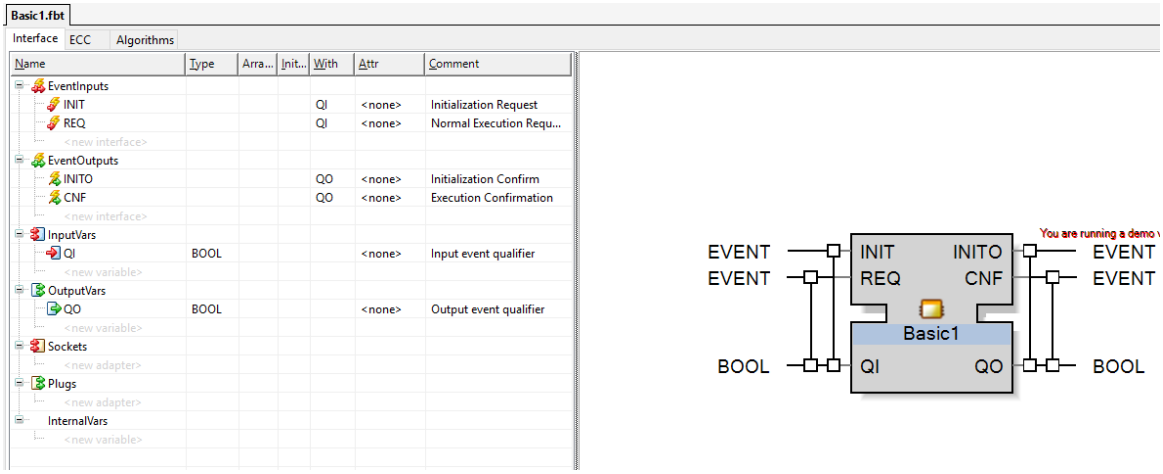


Рис. 1. Внешний вид базового блока и его интерфейс по международному стандарту IEC 61499 в среде nxtControl

Интерфейс стандартного базового блока включает в себя: событийные входы “INIT” и “REQ”, связанные с входной переменной “QI”; событийные выходы “INITO” и “CNF”, связанные с выходной переменной “QO”; входную переменную “QI”; выходную переменную “QO”.

Также у базового блока имеется простая ECC диаграмма управления выполнением. Данная диаграмма позволяет подавать выходные управляющие сигналы в соответствии с логикой работы блока и зависит от входных событийных сигналов, и значений переменных. По умолчанию, в состав ECC диаграммы входят следующие состояния:

- “START” – с него начинается работа функционального блока;
- “INIT” – инициализация параметров блока, при переходе в которое выполняется действие (action) алгоритма с названием “INIT” и подается управляющее воздействие на событийный выход “INITO”;
- “REQ” – нормальное выполнение функций блока, при переходе в которое выполняется действие (action) алгоритма с названием “REQ” и подается управляющее воздействие на событийный выход “CNF”.

А также ECC диаграмма содержит переходы из описанных состояний:

- переход “START” – “INIT” по условию “INIT”;
- переход “INIT” – “START” по условию “1” т.е. всегда выполняется переход;
- переход “START” – “REQ” по условию “REQ”;
- переход “REQ” – “START” по условию “1”.

Созданные по умолчанию алгоритмы “REQ” и “INIT” не содержат в себе специальной логики функционирования блока.

После создания заготовки базового блок управления был переименован в “ControlMove” и выполнена его реконфигурация в соответствии его логикой функционирования [4]. Обновленный интерфейс представлен на рис. 2.

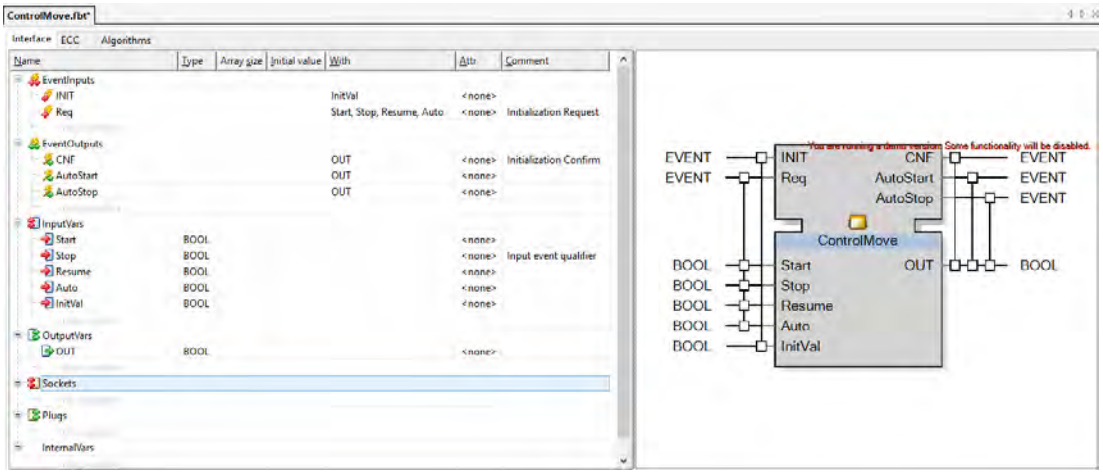


Рис. 2. Внешний вид интерфейса базового блока управления “ControlMove”

Интерфейс базового блока управления **“**ControlMove” включает в себя: событийный вход “INIT”, связанный с входной переменной “InitVal”; событийный вход “Req”, связанный с входными переменными “Start”, “Stop”, “Resume”, “Auto”; событийные выходы “CNF”, “AutoStart” и “AutoStop”, связанные с выходной переменной “OUT”; входную переменную “Start”; входную переменную “Stop”; входную переменную “Resume”; входную переменную “Auto”; выходную переменную “OUT”.

Для базового блока управления двумя выталкивателями и датчиками состояния была разработана ECC диаграмма. Данная диаграмма позволяет осуществлять управляющие воздействия: поднятие багажа, передача его дальнейшему оборудованию, режим блокировки, работа блока управления в ручном и автоматическом режиме в зависимости от входных событийных сигналов и значений переменных. В состав данной ECC диаграммы входят следующие состояния:

- “START” – с него начинается работа функционального блока;
- “INIT” – при переходе в данное состояние выполняется действие (action) алгоритма с названием “INIT” и подается управляющее воздействие на событийный выход “CNF”;
- “Begin” – при переходе в данное состояние выполняется действие (action) алгоритма с названием “StartMove” и подается управляющее воздействие на событийный выход “CNF”;
- “Stop” – при переходе в данное состояние выполняется действие (action) алгоритма с названием “StopMove” и подается управляющее воздействие на событийный выход “CNF”;
- “Resume” – при переходе в данное состояние выполняется действие (action) алгоритма с названием “StartMove” и подается управляющее воздействие на событийный выход “CNF”;
- “AutoOn” – при переходе в данное состояние выполняется действие (action) алгоритма с названием “StartMove” и подается управляющее воздействие на событийный выход “AutoStart”;
- “AutoOff” – при переходе в данное состояние выполняется действие (action) алгоритма с названием “StopMove” и подается управляющее воздействие на событийный выход “AutoStop”.

А также ECC диаграмма содержит переходы из описанных состояний: переход “START” – “INIT” по условию “INIT”; переход “INIT” – “START” по условию “1; переход

“START” – “Begin” по условию “Start”; переход “Begin” – “START” по условию “1”; переход “START” – “Stop” по условию “Stop”; переход “Stop” – “Resume” по условию “Resume”; переход “Resume” – “Begin” по условию “Auto=FALSE”; переход “Resume” – “AutoOn” по условию “Auto=TRUE”; переход “Start” – “AutoOn” по условию “Auto=TRUE”; переход “AutoOn” – “Stop” по условию “Stop”; переход “AutoOn” – “AutoOff” по условию “Auto=FALSE”; переход “AutoOff” – “Start” по условию “1”.

ECC диаграмма и ее алгоритмы для функционирования логики управления выходными сигналами базового блока представлены на рис. 3.

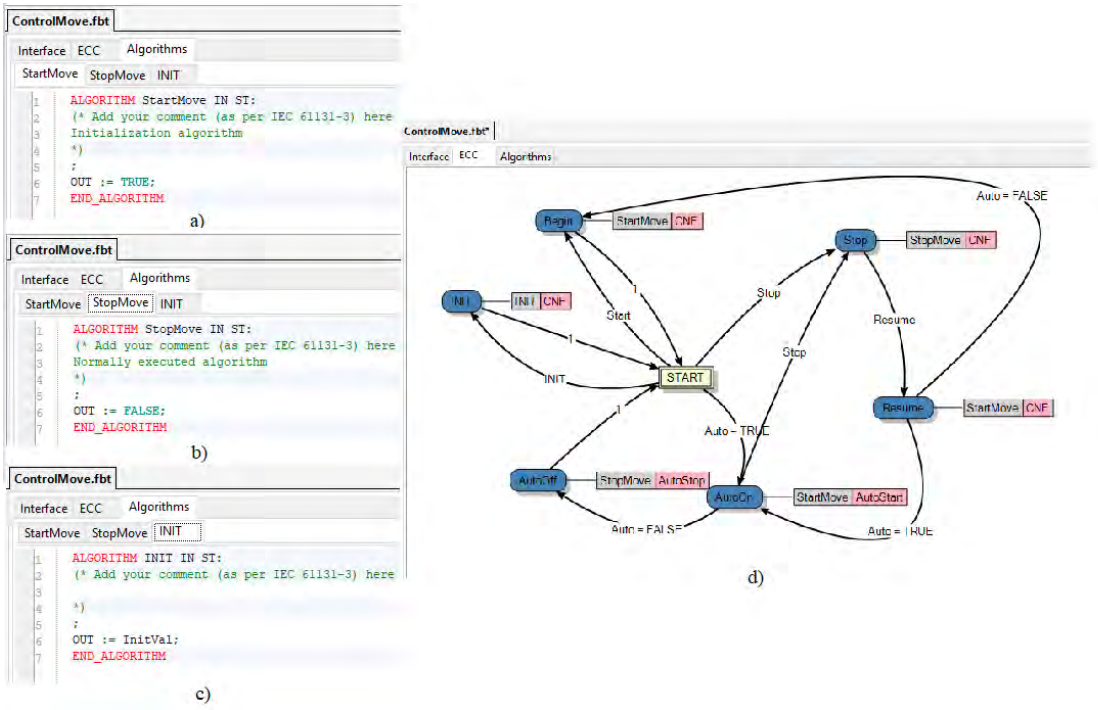


Рис. 3. Внешний вид алгоритмов управления выходными состояниями функционального блока управления “ControlMove”: *a* – StartMove; *b* – StopMove; *c* – INIT; *d* – ECC диаграммы базового блока управления “ControlMove”

Таким образом, была продемонстрирована разработка базового блока в соответствии с международным стандартом IEC 61499 для системы управления перемещением багажа с двумя выталкивателями и датчиками состояний, разработана ECC диаграмма управления для разработанного блока, разработан интерфейс взаимодействия с другими компонентами системы, реализованы алгоритмы управления выходными сигналами блока в среде автоматизированного проектирование nxtControl.

В результате получен базовый блок управления, который является универсальным и в дальнейшем будет встроен в различные системы управления.

Библиографический список

1. Vyatkin, V. IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design / V. Vyatkin // Instrumentation Society of America (ISA) and o^{3}neida. – Auckland University, New Zealand, 2011. – 297 p.

2. Dubinin, V. Formal modeling and verification of IEC 61499 function blocks on the basis of transition systems / V. Dubinin, V. Vyatkin, A. Shalyto // International Siberian Conference on Control and Communications (SIBCON'2016). – M., 2016. – P. 1–4.

3. Интернет-ресурс среды разработки NxtControl. Базовые функции среды разработки. – URL: http://www.nxtcontrol.com/fileadmin/main/download/tutorial_basic_concept_ demo_en.pdf

4. Интернет-ресурс EasyVeep. Система поднятия багажа. – URL: www.easyveep.com

Д. А. Уваров

Пензенский государственный университет, Пенза, Россия

ВСТРАИВАНИЕ БАЗ ЗНАНИЙ В SOA

Рассматриваются процесс управления и встраивания баз знаний, а также реализация сервисно-ориентированной архитектуры с помощью данного процесса.

1. Введение

В настоящем документе предлагается основа для внедрения баз знаний в сервис-ориентированной архитектуре. Есть архитектурная основа для управления бизнес-процессами в сервис-ориентированных системах и набор бизнес-процессов, которые интегрируют управление знаниями с SOA (сервис-ориентированной архитектурой). Важной новизной структуры является то, что она позволяет автоматизировать эволюцию уровня знаний в базе знаний и интегрировать эволюцию базы знаний концептуального уровня с общей сервис-ориентированной системой. В рамках этой рамки решается ряд других вопросов в области управления знаниями, таких как интеграция в повседневную работу и сбор информации из разных ресурсов. Фокусировка происходит на онтологических системах управления знаниями, которые стали одной из важнейших технологий внедрения систем управления знаниями [1].

2. Автоматизация бизнес-процессов в SOA

В последнее десятилетие сервис-ориентированная архитектура стала наиболее признанным подходом к автоматизации бизнес-процессов. В нем основное внимание уделяется тому, как информационные технологии должны поддерживать бизнес-процессы организации с ее основной мотивацией для увеличения возможностей организации в краткосрочной перспективе решать новые бизнес-требования за счет повторного использования существующей бизнес-логики и моделей данных, что приведет к минимальным затратам, ресурсам и накладным расходам времени, минимизируя риски, особенно по сравнению с переписыванием целых прикладных систем. Для этого он добавляет новый уровень обслуживания между уровнем бизнес-процессов и прикладным уровнем предприятия. Сервисный уровень состоит из сервисов, единиц логики, которые должны основываться на сервис-ориентированных принципах, таких как свободное соединение, контракт на обслуживание, автономность, абстракция, многоразовость, возможность компоновки и открытость. Одним из преимуществ уровня сервиса является то, что возможность компоновки позволяет дополнительно структурировать сервисы на разных уровнях. Рекомендуется, чтобы более низкие уровни обслуживания состояли из сервисов приложений и что верхний уровень состоит из сервисов процесса. Сервисы процессов выполняют согласование сервисов нижнего уровня, чтобы их операции выполнялись в определенном порядке. Таким образом, происходит автоматизация бизнес-процесса, что является важной частью достижения лучшей гибкости бизнес-системы.

В настоящее время сервис-ориентированная архитектура основана на технологии веб-сервисов. Для организации веб-сервисов требуется модель, с помощью которой бизнес-процессы можно описать с точки зрения подключения веб-сервисов. Эта модель предоставляется BPEL (язык исполнения бизнес-процессов), язык на основе XML, предназначенный главным образом для поддержки автоматизированных бизнес-процессов на основе веб-сервисов.

3. Управление знаниями и онтологии

Важная часть подходов к управлению знаниями основана на онтологиях. Онтология – это общее понимание некоторой области интереса. Это как средство обеспечения обмена

информацией и знаниями путем сбора общего понимания терминов, которые могут использоваться как людьми, так и программами. Существует язык веб-онтологии OWL. Он состоит из отдельных лиц, свойств и классов. Отдельные экземпляры представляют объекты в интересующем домене. Свойства – это двоичные отношения между отдельными лицами. Классы OWL интерпретируются как наборы, содержащие людей. Они определяются с использованием формальных описаний, которые точно определяют требования к членству в классе. Документ OWL может использоваться для представления двух аспектов организации: концептуального уровня (онтологии) и уровня экземпляра.

Решения по управлению знаниями основаны на организационной памяти, которая объединяет неформальные, полуформальные и формальные знания для облегчения доступа, совместного использования и повторного использования членами бизнес-системы для решения их индивидуальных или коллективных задач [2]. Концепция базы организационных знаний (OOKB), относится к хранимой информации из истории организации и поддерживает совместное использование и повторное использование индивидуальных и корпоративных знаний и уроков.

4. Встраивание баз знаний через автоматизацию бизнес-процессов

Встраивание баз знаний способствует захвату, хранению и распространению знаний с использованием информационных технологий. Встраивание баз знаний рассматривается как непрерывный процесс, который создает знания как побочный продукт работы, интегрирует знания в открытый и развивающийся репозиторий и распространяет его среди других в организации, когда он имеет отношение к их работе. Предлагаемая структура основывается на этих наблюдениях. В контексте бизнес-процесса они переходят к следующему: это непрерывный процесс, который создает знания как побочный продукт выполнения бизнес-процесса, интегрирует знания в открытый и развивающийся репозиторий и распространяет его среди других в организации, когда этого требуют процессы.

С распространением SOA и связанных с SOA технологий наблюдаются несколько возможностей:

1. Автоматический перевод данных процесса в OOKB. Вся бизнес-деятельность для данной бизнес-системы может быть описана бизнес-процессами. Таким образом, каждый информационный компонент, релевантный и известный бизнес-системе, обрабатывается по меньшей мере в одном из бизнес-процессов. OOKB входит в информационную систему и что информационная система автоматизирует один или несколько бизнес-процессов в отношении принципов сервис-ориентирования. В контексте технологических процессов данные процесса имеют определенный смысл и поэтому должны соответствовать информации о случаях в OOKB. Следовательно, существует трансляция между данными процесса и уровнем экземпляра OOKB. В SOA-системе данные уровня экземпляра могут быть захвачены посредством выполнения бизнес-процесса, если такие переводы определены.

Одним из важных видений SOA является автоматизация значительной части бизнес-процессов таким образом, чтобы обеспечить гибкое, экономичное и эффективное внедрение изменений. В полностью реализованной SOA-системе выполнение всех бизнес-процессов будет автоматизировано с помощью механизма выполнения бизнес-процессов, организующего различные сервисы. В такой системе каждый экземпляр OOKB будет иметь соответствующий источник в бизнес-процессе и может быть автоматически переведен в OOKB.

Данные являются основой для информации. Они представляют факты, символы, которые описывают некоторый контент, без более широкого контекста. Информация представляет обработанные данные, которые имеют определенное добавленное значение для пользователя. В контексте выполнения бизнес-процесса данные процесса являются основой для входной информации, необходимой для выполнения действий процесса. Чтобы использовать их вместе с информацией в OOKB и дать возможность обосновать их, нужно привести их в правильный контекст. Этот контекст определяется моделью бизнес-процессов бизнес-системы. Кроме того, в разных экземплярах процесса и активности разные данные могут представлять различную информацию в зависимости от субъекта, который выполняет деятельность. Это особенно важно в случае человеческих задач. Контекст

также зависит от субъекта, ответственного за деятельность, например, индивидуума, отдела, бизнес-системы в целом или другого бизнес-объекта.

Данные процесса соответствуют уровню экземпляра OOKB. Чтобы поместить данные процесса в правильный контекст, определяются переводы в OOKB. Целью является автоматическое получение информации о экземплярах OOKB во время выполнения бизнес-процесса. Возможно, потребуется подтверждение и подтверждение. В этом случае перевод является полуавтоматическим.

Структура данных процесса определяется схемами XML. Таким образом, основой для определения перевода является XML-схема данных процесса и уровень онтологии OOKB. Все данные процесса не всегда релевантны для выполнения процесса. Можно ссылаться на данные процесса, которые представляют входные данные в качестве носителей информации процесса. Информационный носитель процесса может быть любой частью данных процесса. Он может быть представлен одним XML-элементом, появлением элемента внутри другого элемента, комбинацией нескольких элементов и т.д.

2. Сбор информации из разных источников с использованием SOA-подхода к интеграции: сервис-ориентированная интеграция использует решения на основе совместимых сервисов. Из-за нейтральной для поставщиков инфраструктура связи, основанная на SOA, основанной на веб-сервисах, потенциал для предприятий заключается в реализации высоко стандартизированных описаний услуг и структур сообщений. Таким образом, сбор информации из разных источников в SOA осуществляется в отношении принципов ориентации на обслуживание и использования одной и той же интеграционной платформы. Например, предполагается, что информационные системы, содержащие информацию, необходимую для действия, выставляют веб-сервис с операцией, которая возвращает требуемую информацию. В этом случае сбор информации может быть реализован с помощью процесса BPEL, вызывающего соответствующие сервисы. Необходимая информация будет получена через выход сервиса.

Подход SOA к интеграции может также использоваться для сбора информации из разных источников, а не для сбора информации из разных источников таким образом, чтобы обеспечить плотную связь с используемой системой. Для этого сам OOKB может быть представлен как веб-сервис. Можно обеспечить большую гибкость, когда исходные информационные системы меняются во времени, а также независимость конкретной используемой системы. Поэтому предлагаемая структура основывается на принципах ориентации на сервисы и состоит из нескольких слабо связанных сервисов, которые соответствуют этим принципам.

3. Автоматические запросы базы знаний OWL. В SOA-системе бизнес-процессы реализуются как процессы BPEL, которые выполняют различные сервисы, необходимые для выполнения процесса. Для этого необходимо определить точные модели бизнес-процессов и данные процесса. В случае деятельности крайне непредсказуемого характера все компоненты данных будут трудно определить заранее. Однако во многих случаях может быть идентифицирована деятельность бизнес-процесса, для которой требуется OOKB вместе с необходимыми компонентами данных. Для этих действий требуемый запрос OOKB может быть указан статически или динамически. Во время выполнения они могут выполняться автоматически.

4. Интеграция в повседневную работу. Формализация бизнес-процессов позволяет идентифицировать значительную часть мероприятий, требующих OOKB, вместе с требуемым информационным компонентом. Это особенно полезно в случае действий, выполняемых человеческими задачами. При определении человеческой задачи часто можно определить, какие компоненты данных понадобятся для ее выполнения. Поэтому запросы OOKB могут быть частью определения человеческой задачи. Перед тем, как экземпляр задачи будет представлен лицу, выполняющему задачу, запрос может быть выполнен. Полученная информация может быть представлена владельцу задачи с заданием. Таким образом, для владельца задачи использование OOKB является прозрачным.

Когда деятельность по эволюции онтологии требует изменений в онтологии, то они должны реализовываться. Эти действия являются частью бизнес-процесса эволюции онто-

логии. Поскольку они требуют человеческой работы, они могут быть реализованы как человеческие задачи. Когда необходимо внести изменения в OOKB, задача человека запускается и назначается. События, которые инициируют эти действия и как они должны выполняться, определяются с помощью выбранной методологии. Развитие и эволюция онтологии управляются XML-схемой.

5. Автоматизация определенных человеческих задач, основанных на знаниях. Если знания о действиях захватываются способом, который может быть автоматически обработан, и если доступны необходимые входы активности, задача может быть выполнена автоматически. Действия, требующие проведения логического процесса, редко автоматизируются, поскольку они требуют человеческих рассуждений и логических способностей. Однако часто рассуждающие действия требуют аргументации, которая носит повторяющийся характер - разные экземпляры деятельности требуют одного и того же или аналогичного процесса рассуждения. В бизнес-системах владельцы человеческих задач также ограничены политикой и правилами, которые они должны соблюдать. Для повторяющихся типов методов рассуждений существуют методы, которые могут быть использованы для захвата этого типа знаний, например, правил принятия решений, бизнес-правил, логических инструкций и т.д. На основании этого требуемый ввод для аргументации может быть извлечен из соответствующего источника, если он доступен. Поэтому, если знания, необходимые для аргументации, фиксируются в OOKB, задача анализа может быть автоматизирована путем выполнения запроса и аргументации с помощью OOKB. Это означает, что при соответствующем планировании, выборе соответствующих методов и применении их к системе, основанной на онтологии, как части общей системы SOA, может быть достигнут более высокий уровень автоматизации бизнес-процессов.

Библиографический список

1. Ontologies for Enterprise Knowledge Management / A. Maedche, B. Motik, L. Stojanovic, R. Studer, R. Volz // IEEE Intelligent Systems. – 2003. – № 18 (2). – P. 26–33.

2. Dieng, R. Methods and tools for corporate knowledge management / R. Dieng, O. Corby, A. Giboin, M. Ribiere // International Journal of Human Computer Studies. – 1999. – № 51 (3). – P. 567–598.

И. А. Беззатеев, В. Н. Дубинин, Д. А. Уваров

Пензенский государственный университет, Пенза, Россия

МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕННЫХ СЕМАФОРОВ НА ОСНОВЕ ПРОТОКОЛОВ PAXOS И RAFT В СИСТЕМЕ CPN TOOLS

Разработаны сетевые имитационные модели (в виде цветных временных сетей Петри) распределенных семафоров, основанных на алгоритмах Paxos и Raft для решения задач консенсуса в сети ненадежных процессов, проведены имитационные эксперименты в системе CPN Tools, даны сравнительный анализ и рекомендации по их использованию. Делается вывод, что семафор на основе Raft существенно производительнее семафора на основе Paxos.

Одним из наиболее важных примитивов синхронизации и взаимодействий процессов в параллельных системах является семафор. Семафор – это особый тип разделяемой переменной, которая обрабатывается только двумя неделимыми операциями – P и V (за-

крытие и открытие семафора) [1]. На основе семафоров строятся другие, более сложные примитивы и задачи синхронизации. Например, задача «производитель-потребитель» реализуется с использованием трех семафоров. По своей сути семафор относится к методам взаимодействий процессов через общую (разделяемую) память, поэтому существует семантический разрыв между концепцией семафора и концепцией распределенных систем, где взаимодействия осуществляются посредством передачи сообщений.

Существует несколько подходом к реализации распределенных семафоров. Один из подходов использует двухфазный протокол блокировки в распределенных базах данных с дублированием копий [2]. Однако в представленной работе будет использоваться более современный подход на основе использования алгоритмов (протоколов) для решения задач консенсуса в сети ненадежных процессов – Paxos [3] и Raft [4]. Консенсус – процесс получения согласованного результата группой участников, основная проблема – наличие помех в среде передачи данных. Алгоритм Raft разрабатывался с учетом недостатков алгоритма Paxos. При выборе ключевых идей предпочтение отдавалось более простым и практичным решениям.

Целью данной работы является моделирование распределенных семафоров на основе протоколов Paxos и Raft с целью верификации, а также для их сравнительного анализа по производительности. Наиболее подходящим методом исследования является имитационное моделирование. В рамках этого метода с точки зрения формализации наиболее подходящими являются модели на основе сетей Петри, позволяющие описывать параллельные взаимодействующие процессы. В работе [5] был предложен сетевой формализм (СФ), позволяющий строить формальные модели на основе расширенных сетей Петри, разработана система сетевого имитационного моделирования (СИМС), однако отсутствие версии СИМС на современных компьютерах не позволяет использовать СФ на практике. Поэтому для формального описания протоколов Paxos и Raft были выбраны цветные (раскрашенные) временные сети Петри [6], положенные в основу инструментальной системы моделирования CPN Tools [7]. Цветные временные сети Петри и система CPN Tools хорошо зарекомендовали себя для моделирования коммуникационных протоколов [8].

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 (время захвата или освобождения семафора, время получения клиентом ответа на запрос, данные пакетов, полученных клиентом). Ввиду того, что сетевые модели Paxos и Raft довольно громоздкие, их описание приводится ниже лишь в ограниченном объеме.

Разработанная модель системы «Распределенный семафор на основе протокола Paxos» включает собой раскрашенную сеть Петри, имитирующую работу протокола Paxos для достижения консенсуса при одновременном доступе клиентов к критическому ресурсу, она имеет иерархическую структуру и содержит в себе следующие подсети: System – сеть самого верхнего уровня, включает в себя все нижеперечисленные подсети; Client, 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, которая представлена на рис. 1, является самой верхней в иерархической структуре и содержит в себе остальные подсети.

Имитационные эксперименты с разработанными сетевыми моделями в системе CPN Tools показали работоспособность и эффективность алгоритмов Paxos и Raft для решения задачи достижения консенсуса в процессе взаимодействия клиентов с критическим ресурсом. В ходе моделирования были получены статистические данные о работе модели. На основе их анализа были выявлены следующие преимущества и недостатки разработанных систем (на основе Paxos и Raft): а) обе сети успешно выполняют обработку запросов на захват или освобождение семафора от нескольких клиентов; б) в обеих сетях преду-

смотрен механизм решения проблемы отказа компонента системы и в ходе тестирования разработанных систем была доказана его работоспособность.

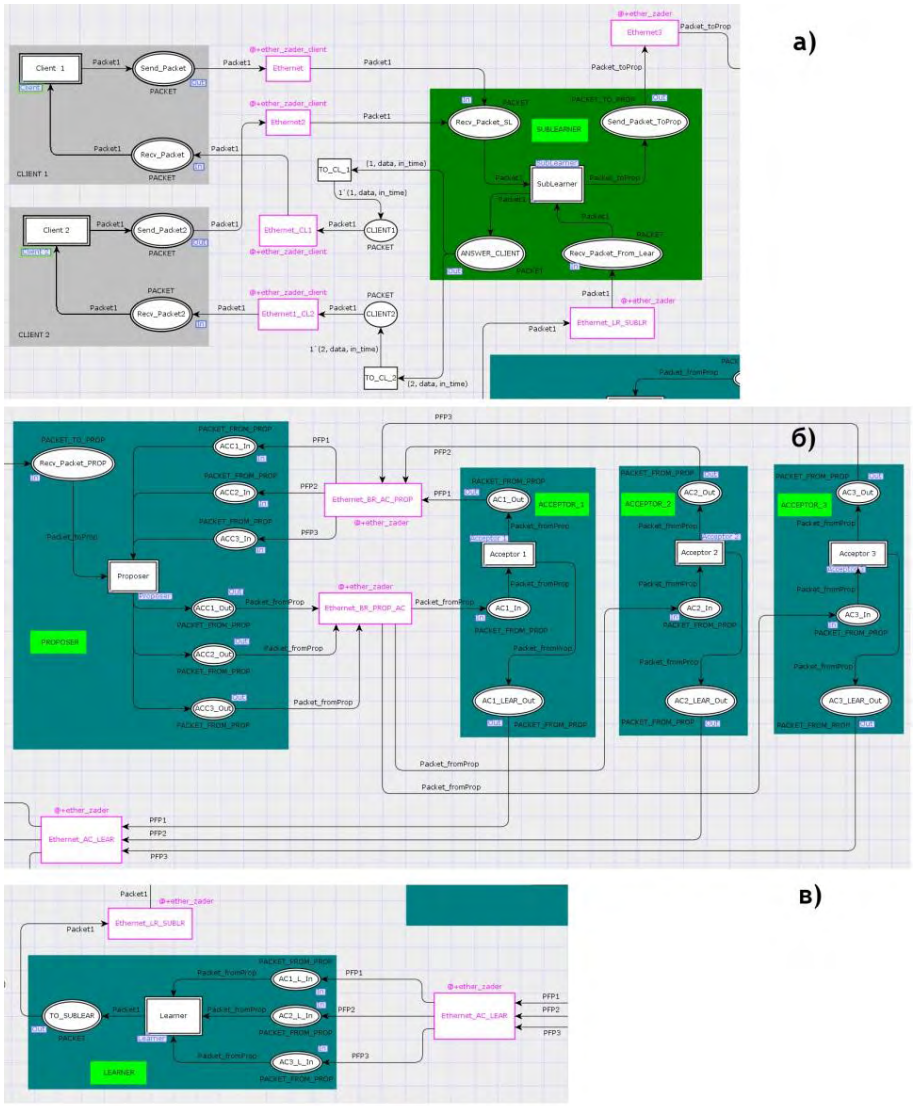


Рис. 1. Сетевая модель верхнего уровня (протокол Paxos)

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 моделей распределенного семафора на основе протоколов Paxos и Raft, были построены соответствующие таблицы и гистограммы. В приведенной таблице представлены данные, полученные в результате анализа информационных пакетов, генерируемых при успешном выполнении операции захвата или освобождения семафора (подсети SubLearner, Leader_RAFT).

На рис. 2 представлены следующие гистограммы (в порядке следования): «Время получения клиентом ответа на свой невыполненный запрос», «Время получения клиентом ответа на свой выполненный запрос», «Время выполнения операции захвата/освобождения семафора»).

Время выполнения операции захвата/освобождения семафора		
Количество величин в заданном диапазоне	PAXOS	RAFT
3–5 мс (шт.)	0	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

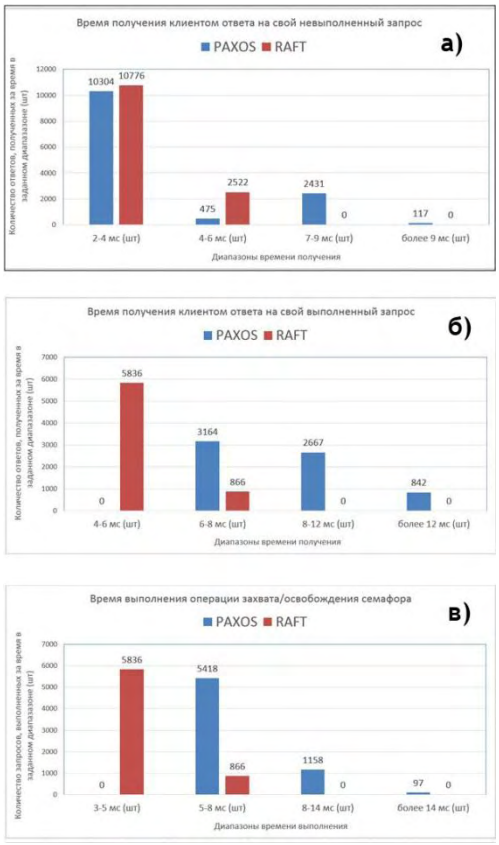


Рис. 2. Результаты имитационного моделирования

Из рис. 2,а видно, что основное количество невыполненных запросов, обрабатываемых сетями Paxos и Raft, обрабатываются за время 2–4 мс. Однако в сети Paxos часть запросов была обработана за время большее чем 7 мс, а в сети Raft данная величина не превышала 6 мс. На рис. 2,б аналогичная тенденция просматривается с количеством выполненных запросов. На рис. 2,в показано, что и сама операция захвата или освобождения семафора, в целом, выполняется быстрее в сети на основе алгоритма Raft.

Исходя из результатов, полученных при имитационном моделировании моделей распределенного семафора на основе протоколов Paxos и Raft в среде CPNTools, можно сделать следующие выводы: а) в сети на основе Raft клиенты получали ответ на свой невыполненный запрос в среднем на 21,1 % 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 51,8 % быстрее, чем в сети на основе Paxos; г) обе сети успешно выполняют обработку запросов на захват или освобождение семафора от нескольких клиентов.

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) система справляется с поставленной задачей быстрее. В общем, использование алгоритма Raft при построении системы распределенного семафора является более простым и эффективным способом, чем использование протокола Paxos.

Направлением дальнейших исследований является построение распределенных семафоров на основе других принципов, алгоритмов и протоколов, их моделирование и сравнительный анализ.

Библиографический список

1. Эндрюс, Г. Р. Основы многопоточного, параллельного и распределенного программирования : пер. с англ. / Г. Р. Эндрюс. – М. : Вильямс, 2003. – 505 с.

2. Дубинин, В. Н. Инструментальные средства для проектирования, верификации и реализации распределенных вычислительных систем на основе локальной сети / В. Н. Дубинин // Вычислительная техника в автоматизированных системах контроля и управления : Межвуз. сб. науч. тр. – Пенза : Изд-во ПГТУ, 1996. – Вып. 24. – С. 43–49.

3. Алгоритм Paxos. – URL: https://ru.wikipedia.org/wiki/Алгоритм_Паксос

4. Алгоритм Raft. – URL: https://ru.wikipedia.org/wiki/Алгоритм_Raft

5. Дубинин, В. Н. Сетевые модели распределенных систем обработки, хранения и передачи данных : моногр. / В. Н. Дубинин, С. А. Зинкин. – Пенза : Приволжский Дом знаний, 2013. – 452 с.

6. Jensen, K. Coloured Petri Nets. Modelling and Validation of Concurrent Systems / K. Jensen, L. M. Kristensen. – Springer Publishing Company, 2009. – 384 p.

7. CPN Tools. – URL: http://cpntools.org/

8. Зайцев, Д. А. Моделирование телекоммуникационных систем в CPN Tools / Д. А. Зайцев, Т. Р. Шмелева. – Одесса : ОНАС, 2008. – 68 с.

И. В. Сенокосов

Пензенский государственный университет, Пенза, Россия

ФУНКЦИОНАЛЬНО-БЛОЧНАЯ РЕАЛИЗАЦИЯ МАГАЗИННЫХ АВТОМАТОВ

Предлагается метод реализации магазинных автоматов на основе функциональных блоков международного стандарта МЭК 61499. Предложенный реализационный подход может быть использован в проектировании систем управления выборкой деталей в производственных системах.

В настоящее время в связи с усложнением технологических процессов и требованием к их быстрой перенастройке возникает потребность как в формализации техпроцессов, так и в разработке на основе формальных описаний методов реализации систем управления, поддерживающих данные техпроцессы. Наиболее перспективный подход к построению реконфигурируемых распределенных систем управления предлагает международный стандарт МЭК 61499 [1]. Наибольшее распространение при построении систем управления технологическим оборудованием нашли конечные автоматы и регулярные языки. В работе [2] предложен подход к реализации недетерминированных конечных автоматов на основе функциональных блоков (ФБ) стандарта МЭК 61499. Следует отметить, что конечные автоматы являются достаточно простыми моделями и не могут описать сложные процессы, например, представляемые контекстно-свободными (КС) языками. Для распознавания КС-языков используются магазинные автоматы (МА) [3]. Формальное определение МА можно найти, например, в [4]. В отличие от конечных автоматов, в МА используется магазинная память (стек, магазин).

В данной работе для простоты будут использоваться только детерминированные МА. Пример графического представления МА в человеко-машинном интерфейсе, реализованным в *NxtStudio* [5], приведен на рис. 1. Пометка каждого перехода показывает входной символ, вершину стека, а также чем она будет заменена. Следует заметить, что сам стек в данном представлении не фигурирует. Приведенный на рис. 1 МА распознает КС-язык, заданный следующей формальной грамматикой:

$$S \rightarrow S + T \mid T$$

$$T \rightarrow (S) \mid a.$$

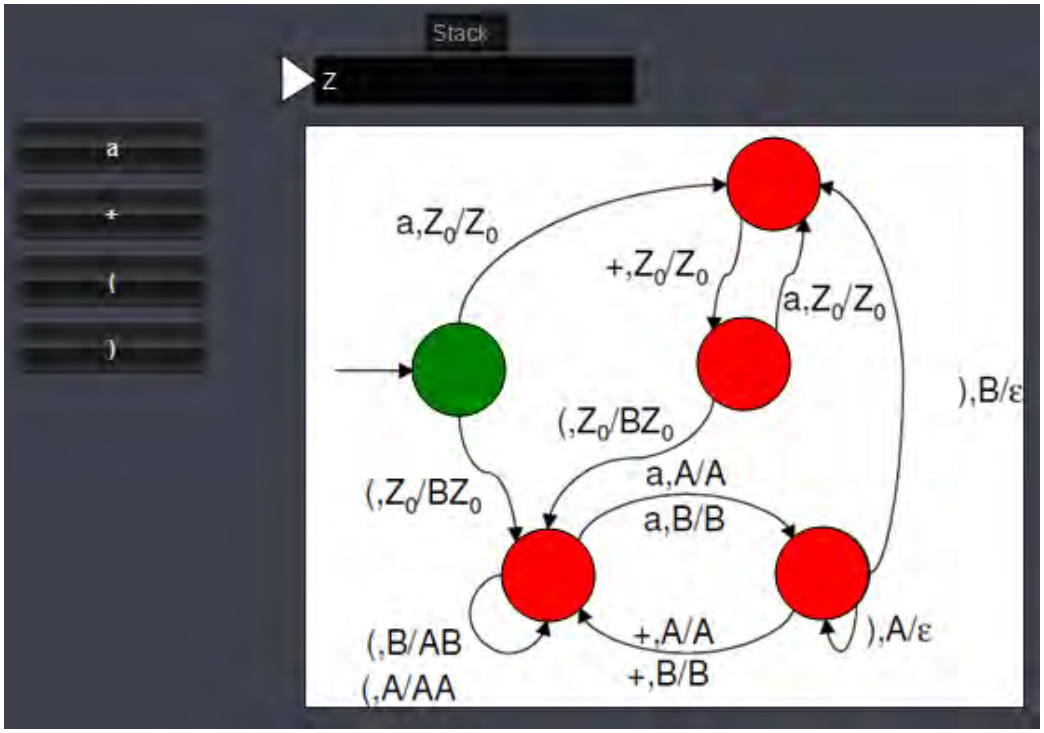


Рис. 1. Графический интерфейс для моделирования работы МА

Данная грамматика определяет язык скобочных арифметических выражений с операцией сложения над операндом *a*.

Рассмотрим подход к реализации магазинных автоматов на основе функциональных блоков (ФБ) международного стандарта МЭК 61499 [1]. Данный подход предполагает реализацию МА, когда переходы представляются в виде ФБ. При данном подходе ФБ, представляющий переход, одного и того же типа и отличается только некоторыми параметрами при инициализации. Можно также представлять состояния в виде ФБ, но каждый ФБ будет отличаться числом входных/выходными сигналов и переменных, и сложность реализации каждого отдельного блока-состояния будет довольно большой.

Для моделирования функционирования МА используется механизм маркеров. Маркер является динамическим объектом, который может быть передан из одного ФБ-перехода в другой ФБ-переход. Если ФБ имеет маркер, то он может принимать входные символы. Упрощенная схема реализации представлена на рис. 2,а. Полная схема реализации примера из рис. 1 не приведена из-за большого объема – в работающей схеме тринадцать ФБ-переходов. Первый блоков типа *StateMA* реализует один из переходов автомата. ФБ *Switch* управляет потоками данных от ФБ-переходов в функциональный блок стека. ФБ *Stack* отвечает за все операции связанные со стеком (магазином). Реализация МА функционирует следующим образом: приходит некоторый входной сигнал, срабатывает один из ФБ-переходов или несколько ФБ, если ФБ принимают один и тот же входной сигнал и в них установлены маркеры. Далее сигналы через ФБ *Switch* направляются в магазин. Если

верхний магазинный символ совпал с магазинным символом перехода, то сработавший ФБ-переход передает свой выходной сигнал на один из выходов $S_0 \ldots S_z$.

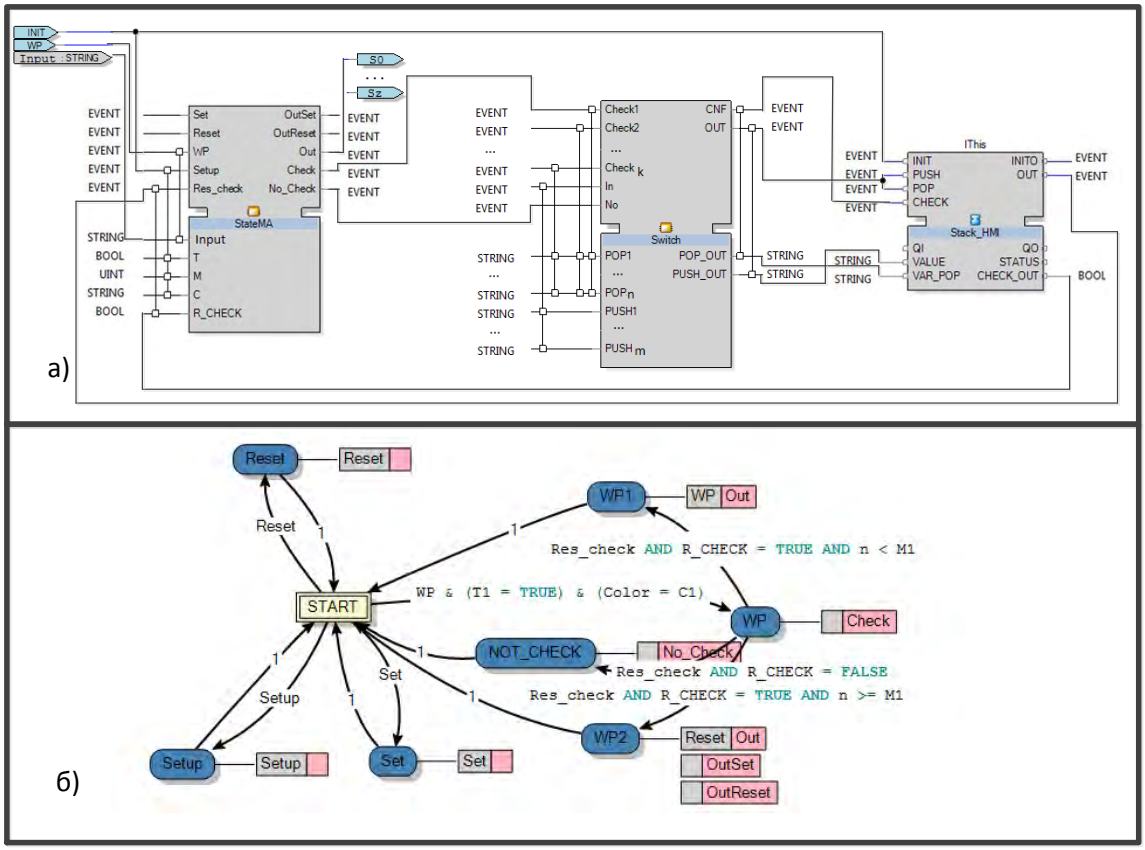


Рис. 2. а – сеть ФБ, реализующая МА; б – диаграмма ECC ФБ StateMA

Рассмотрим реализацию ФБ *StateMA* подробнее. Интерфейс данного базисного ФБ состоит из следующих групп параметров:

Входные сигналы:

Set – сигнал установки маркера в данном ФБ;

Reset – сигнал удаления маркера из данного ФБ;

WP – сигнал поступления входного символа;

Setup – сигнал, установки (сброса) начальных параметров ФБ;

Res_Check – сигнал проверки верхнего символа магазина с магазинным символом ФБ.

Выходные сигналы:

OutSet – сигнал, устанавливающий маркеры в других ФБ;

OutReset – сигнал, сбрасывающий маркеры в других ФБ;

Out – сигнал, означающий, что ФБ сработал;

Check – сигнал проверки верхнего символа магазина;

No_Check – сигнал, который выдается, если верхний символ магазина не совпал с магазинным символом ФБ.

Входные переменные:

Input – входной символ;

C – входной символ, который может принять ФБ;

T – флаг наличия маркера в ФБ;

M – максимальное количество деталей заданного цвета, которые могут быть обработаны данным ФБ перед передачей маркера на другой (другие) ФБ.

R_CHECK – результат проверки верхнего символа магазина.

Диаграмма управления выполнением (иначе, диаграмма ECC) базисного ФБ *State-MA* представлена на рис. 2,б. и состоит из шести состояний:

1) состояние *Reset*, в котором выполняется сброс маркера ФБ;

2) состояние *Setup*, в котором устанавливаются начальные значения внутренних переменных;

3) состояние *Set*, в котором происходит установка маркера ФБ;

4) состояние *WP*, в которое автомат переходит при выполнении следующего условия: *WP & (T1 = TRUE) & (Color = C1)* WP&(T1 = TRUE)&(Color = C1). Переход в данное состояние произойдет, если в реализацию МА поступит входной символ, воспринимаемый этим ФБ, и данный ФБ имеет маркер, при переходе в данное состояние выдается сигнал *Check*;

5) состояние *WP1*, в котором происходит увеличение счетчика количества прохождений по петле. В данное состояние может попасть ФБ, *R_CHECK = TRUE* и *M > 1*. Кроме того, в данном состоянии выдаются выходные сигналы *Out* и *OutReset*;

6) состояние *WP2*, в котором происходит удаление маркера и выдача всех требуемых выходных сигналов. В данное состояние попадает ФБ, имеющий *R_CHECK= = TRUE* и *M = 0,* или ФБ, уже прошедший *M* раз по петле;

7) состояние *NOT_CHECK*, в которое автомат переходит, если не принят стеком (*R_CHECK = FALSE*). Кроме того, в данном состоянии выдаются выходной сигнал *No_Check*.

Интерфейс ФБ *Stack* состоит из следующих групп параметров:

Входные сигналы:

INIT – инициализация стека;

PUSH – положить значение в стек;

POP – извлечь значение из стека;

CHECK – проверка значения входной переменной *VAR_POP* с верхним значением стека, результат данной проверки выдается в выходной переменной *CHECK_OUT*.

Выходные сигналы:

INTO – сигнал, означающий, что ФБ инициализирован;

OUT – сигнал, сбрасывающий в результате проверки верхнего значения стека;

Входные переменные:

VAR_POP – переменная, с которой происходит сравнение верхнего символа стека;

VALUE – переменная, значение которой добавляется в стек;

Выходные переменные:

OUT_VALUE – переменная – результат проверки входной переменной *VAR_POP* с верхним значением стека (*TRUE* – значения равны, *FALSE* – не равны).

ФБ *Switch* необходим для перенаправления значений переменных, содержащих магазинные символы, в стек, ввиду того, что в среде *NxtStudio* нельзя подсоединить несколько выходных переменных ФБ к одной входной. ФБ *Switch* в зависимости от определенного сигнала проверки (*Check1…Check*$_k$) сопоставляет соответствующий магазинный символ с вершиной стека и, если символы совпали, то записывает необходимую цепочку в магазин.

Интерфейс данного ФБ состоит из следующих групп параметров:

Входные сигналы:

Check1..Check13 – проверка стека от определенного ФБ-перехода;

In – втолкнуть значение в стек;

No – значение в стек не вталкивать;

Выходные сигналы:

CNF – сигнал, означающий проверку верхнего значения в стеке;

OUT – сигнал, по которому извлекается верхнее значение стека и переменная *PUSH_OUT,* которая содержит необходимую цепочку, которая добавляется в стек;

Входные переменные:

POP1..POP3 – переменные (алфавит стека) с которыми сравнивается верхний магазинный символ;

PUSH0..PUSH6 – переменные – цепочки, которые добавляются в стек;

Выходные переменные:

POP_OUT – переменная, значение которой необходимо сравнить с верхним символом стека.

PUSH_OUT – переменная, значение которой вталкивается в стек.

В представленной работе предложен и протестирован метод реализации детерминированных магазинных автоматов на основе ФБ. Направлением дальнейших исследований является разработка метода реализации селектирующих сетей Петри на основе ФБ МЭК 61499.

Библиографический список

1. Vyatkin, V. IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design / V. Vyatkin. – 3-rd edn. – Pittsburgh: Instrumentation Society of America, 2016. – 261 p.

2. Функционально-блочная реализация недетерминированных конечных автоматов / В. Н. Дубинин, И. В. Сенокосов, А. С. Войнов, Д. Н. Дроздов, В. В. Вяткин // Современные информационные технологии : тр. Междунар. науч.-техн. конф. – Пенза, 2017. – Вып. 25. – С. 5–18.

3. Дубинин, В. Н. Использование моделей магазинных автоматов в проектировании технологических процессов сортировки и сборки изделий / В. Н. Дубинин, И. В. Сенокосов, Л. П. Климкина, В. В. Вяткин // Современные технологии в науке и образовании : сб. тр. II Междунар. науч.-техн. и науч.-метод. конф. (СТНО-2017). – Рязань, 2017. – Т. 1. – С. 10–14.

4. Хопкрофт, Дж. Введение в теорию автоматов, языков и вычислений / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман. – М. : Вильямс, 2002. – 528 с.

5. Сайт nxtStudio (nxtControl). – URL: http://www.nxtcontrol.com/

А. С. Бождай, Ю. И. Евсеева, А. А. Гудков

Пензенский государственный университет, Пенза, Россия

ОПТИМИЗАЦИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА АДАПТИВНЫХ СИСТЕМ С ПОМОЩЬЮ ТЕХНОЛОГИИ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Изложена оригинальная технология самооптимизации и пользовательского интерфейса адаптивных информационных систем. В основе предлагаемой технологии лежат алгоритмы интеллектуального анализа данных.

Поскольку создание пользовательского интерфейса программной системы представляет собой задачу, решаемую человеком (разработчиком программного обеспечения или дизайнером), нет абсолютной гарантии того, что разработанный интерфейс будет оптимальным для всех категорий пользователей. Чем выше сложность программной системы, тем сложнее учесть все факторы, связанные с ее предметной областью и функционированием. Определенные закономерности в работе сложной системы могут быть недоступны пониманию разработчиков на этапе проектирования. Такие закономерности, как правило, становятся выявляемыми только после определенного количества времени эксплуатации программного обеспечения [1, 2].

На рис. 1 представлена схема пространства интерфейсов. Как видно из рисунка, пространство интерфейсов организовано иерархическим образом. Сплошными линиями изображены отношения агрегации между элементами, пунктирными – отношения вызова. Примеры отношения вызова – отображение дочернего окна при нажатии на пункт меню, переход по гиперссылке на другую страницу веб-сайта.

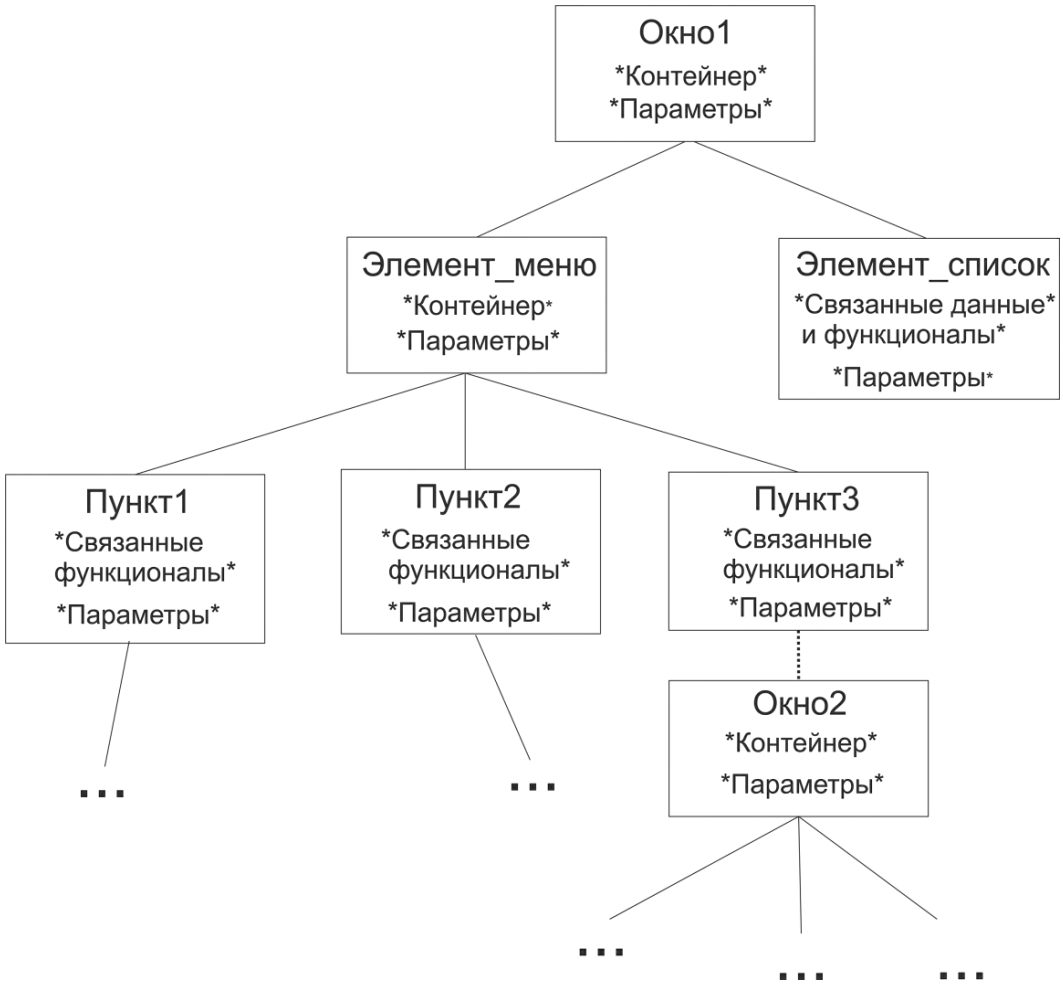


Рис. 1. Пример рис. в тексте доклада

Для оценки оптимальности создаваемых интерфейсов следует воспользоваться вычислением и анализом пользовательских шаблонов навигации. Пользовательский шаблон навигации представляет собой одну из наиболее часто используемых траекторий перемещения пользователя программной системы в пространстве интерфейсов. Для сбора информации о траекториях можно использовать запись пользовательских действий в лог-файл, для анализа и выявления навигационных шаблонов – методы нахождения последовательных шаблонов.

Для анализа поведенческой продукции системы будем использовать алгоритм AprioriALL [3]. На вход алгоритму AprioriALL поступает лог-файл, протоколирующий действия пользователя с системой. В качестве выходных данных выступает список пользовательских шаблонов навигации.

Предлагаемый алгоритм самомодификации пользовательского интерфейса, основанный на технологии интеллектуального анализа данных [4], содержит следующие шаги:

1. Считывание списка исходного набора данных из файла.
2. Формирование списка пользовательских шаблонов навигации помощью алгоритма AprioriALL. Список шаблонов навигации будет иметь вид

$$A = \{A_1, A_2, ..., A_z\},\ z = 1,... f,$$
$$A_i = \{A_{i1}, A_{i2} ..., A_{ik}\},\ A_{ij} = (id_j, idF_j, idD_j, v_j) \forall j = 1,..., k_i,$$

где f – число шаблонов; k_i – число элементов i-го шаблона; $id_j \in P_j$ – идентификатор интерфейса j-го элемента i-го шаблона; $idF_j \in FB_j$ – идентификатор задействованного

в операции над *j*-м интерфейсом *i*-го шаблона функционала; $idD_j \in DB_j$ – идентификатор задействованного в операции над *j*-м интерфейсом *i*-го шаблона объекта данных; $v_j \in V_I$ – связанная с воздействием на *j*-й интерфейс *i*-го шаблона информация (пользовательские действия).

3. Цикл по шаблонам, $s = 1,..., f$.

4. Внутренний цикл по шаблонам, $r = s + 1..., f$.

5. Просмотр очередного шаблона A_r и проверка на то, является ли он схожим с шаблоном A_s . Для определения схожести шаблонов определяются совпадающие в обоих шаблонах подпоследовательности и оценивается порядок их следования. В схожих шаблонах порядок следования идентичных подпоследовательностей должен совпадать. Шаблоны можно считать достаточно схожими, если число элементов, не входящих в идентичные подпоследовательности, не превышает определенного порога.

6. Если A_s и A_r являются схожими шаблонами, то осуществляется их «слияние» – формирование на первом общем для шаблонов интерфейсе-контейнере нового дочернего интерфейса, реализующего совпадающую функциональность шаблонов и использующего общие данные. В процессе выполнения операций, связанных с новым интерфейсом, осуществляется обращение к интерфейсам, входящим в неидентичные подпоследовательности, для получения необходимых данных. Если A_s не является схожим шаблоном, то осуществляется переход к пункту 8.

7. Поиск элементов из совпадающих подпоследовательностей в других шаблонах списка. Если какой-либо элемент не встречается в других шаблонах, то принимается решение о том, что данный элемент является редко используемым. Редко используемый элемент становится скрытым элементом.

8. Завершение внутреннего цикла по шаблонам. Если еще есть непросмотренные шаблоны, переход к пункту 5, если нет – к пункту 9.

9. Если для шаблона A_s не были найдены схожие шаблоны, осуществляется оптимизация интерфейса на основе информации, содержащейся в A_s . На первом интерфейсе-контейнере шаблона формируется новый дочерний интерфейс, реализующий функциональность и работающий с данными всего шаблона.

10. Завершение внешнего цикла по шаблонам.

11. Завершение работы алгоритма.

Предлагаемый алгоритм, разработанный в рамках представлений о рефлексивной адаптации программных систем, позволяет решить 2 основные проблемы, возникающие в процессе создания и эксплуатации пользовательских интерфейсов: проблему недостаточности пространства интерфейсов (отсутствия элементов управления для наиболее часто выполняемых операций) и проблему избыточности (наличие элементов, реализующих «несамостоятельную» функциональность).

Библиографический список

1. Keromytis, A. Characterizing Software Self-healing Systems / A. Keromytis // International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security. – Berlin : Springer, 2007. – P. 22–23.

2. Эванс, Э. Предметно-ориентированное проектирование: структуризация сложных систем / Э. Эванс. – М. : Вильямс, 2011. – 444 с.

3. Анализ данных и процессов / А. Барсегян, М. Куприянов, И. Холод, М. Тесс, С. Елизаров. – СПб. : БХВ-Петербург, 2009. – 314 с.

4. Бутчер, С. Программы лояльности и клубы постоянных клиентов / С. Бутчер. – М. : Вильямс, 2004. – 272 с.

Ю. И. Евсеева

Пензенский государственный университет, Пенза, Россия

МНОГОУРОВНЕВАЯ АРХИТЕКТУРА АДАПТИВНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Рассматриваются проблемы создания адаптивных информационных систем. Помимо разработки специализированных методов синтеза программного обеспечения такого типа, существенной частной проблемой является проблема создания универсальной архитектуры адаптивной системы. Для ее решения в данной статье предложен оригинальный архитектурный шаблон.

В настоящее время среди специалистов отсутствует единое определение понятию адаптивности программного обеспечения. Ситуация такова, что для разных типов программных систем адаптивность может определяться различным образом. Взгляды на реализацию процесса адаптации могут также различаться и в пределах рассмотрения одного класса систем и сильно зависеть от выбранной методологии проектирования [1–3].

Тем не менее, несмотря на кажущиеся существенные различия в определениях и подходах, можно выделить одно общее свойство всех систем, называемых адаптивными [4, 5]: способность к самомодификации (на уровне структуры системы в целом либо на уровне отдельных компонентов).

Любая программная система построена из компонентов различных типов. Хорошим стилем в проектировании программных систем считается минимизация количества зависимостей между отдельными компонентами (модульный подход). Многие совре-менные способы проектирования программного обеспечения (например, сервис-ориентиро-ванная архитектура) предполагают также наличие унифицированных интерфейсов для взаимодействия модулей друг с другом и системой в целом. Как показывает практика, гибкость архитектуры оказывает огромное влияние на способность программы адаптироваться к изменениям и усложнениям. Гибкая архитектура – это, как правило, результат детального моделирования и тщательного выбора проектных решений.

Построение универсального шаблона гибкой архитектуры для абсолютно всех возможных в реализации программных систем вряд ли является выполнимой задачей. Однако, если рассматривать организацию системы только с точки зрения интересующих свойств (например, адаптивности), можно создать архитектурный шаблон, позволяющий проектировать программы, оптимальные именно с позиции рассматриваемого свойства.

На рис. 1 приведена упрощенная архитектурная модель адаптивной системы. Основная идея предлагаемой модели заключается в том, что компоненты каждого из существующих архитектурных уровней будут собственным образом реагировать на необходимость в реорганизации.

Интерфейсный уровень архитектуры содержит объекты-интерфейсы, предназначенные для взаимодействия с конечным пользователем. Уровень данных объединяет используемую в работе системы информацию (например, информацию из баз данных). Уровень функционалов включает в себя 3 уровня: уровень функционалов предметной области (программные операции, выполняемые отдельными объектами предметной области); уровень операционных функционалов (распределяют задачи между объектами предметной области); уровень инфраструктурных функционалов (обеспечивают непосредственную техническую поддержку для верхних уровней).

Рис. 1. Обобщенная архитектура адаптивной информационной системы

Наибольший интерес представляют модели предметной области. Благодаря специализированным предметно-ориентированным языкам, конечный пользователь системы видит их как цельные понятия предметной области. Однако окружающей архитектурной средой эти понятия представляют собой комбинацию компонентов 2 смежных уровней – уровня данных и уровня функционалов. С этой точки зрения, модели предметной области являются аналогами классов в объектно-ориентированном проектировании, однако в предлагаемой модели разделение между функциональным уровнем и уровнем данных акцентируется и в том случае, когда речь идет о цельном понятии предметной области. Это связано с тем фактом, что функционалы и данные различным образом реагируют на необходимость в изменениях.

Для рассмотренной ориентированной на самомодификацию архитектуры механизм адаптации определяется следующим соотношением:

$$S_A = A(S, L(S, V)), \tag{1}$$

где S_A – модифицированная структура системы; A – функция анализа поведенческой продукции системы и модификации ее первоначальной структуры; L – функция вычисления поведенческой продукции системы; S – исходная структура системы (включает совокупность объектов адаптируемых уровней архитектуры); V – совокупность внешних воздействий на адаптируемые объекты.

Применительно к S также справедливо

$$S = \{I, D, F_A, M\},$$

где I – множество объектов уровня интерфейсов; D – множество объектов уровня данных; $F_A = F \setminus F_I = \{F_D, F_O\}$ – множество адаптируемых функционалов; M – множество объектов предметной области. К адаптируемым функционалам относятся функционалы уровня предметной области (F_D) и функционалы операционного уровня (F_O). Функционалы уровня инфраструктуры (F_I) не являются адаптируемыми объектами.

Множество объектов предметной области представляет собой неориентированный гиперграф

$$M = \{W_D, W_F, E\},$$

где W_D – множество вершин, отображающих используемые структуры данных (точки на уровне данных рис. 2); W_F – множество вершин, отображающих функционалы предметной области; E – семейство непустых подмножеств множества $W_D \bigcup W_F$ (цельных объектов предметной области).

Следует отметить, что адаптация уровня целостных объектов предметной области не сводима к адаптации отдельных компонентов этих объектов (функционалов и данных). Помимо оптимизации функционалов и данных, она включает в себя также изменение структуры целостного объекта. Изменение структуры объектов реализуется через операции над гиперграфом *M*.

Библиографический список

1. Keromytis, A. D. Characterizing Software Self-healing Systems / A. D. Keromytis // Computer Network Security. – Berlin : Springer, 2007. – P. 22–33.

2. Shin, M. E. Connector-Based Self-Healing Mechanism for Components of a Reliable System / M. E. Shin, D. Cooke // Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005). – New Yourk : ACM, 2005. – P. 1–7.

3. Smart Sparrow. – URL: https://www.smartsparrow.com (дата обращения: 22.09.2017).

4. Fuzzy control-based software self-adaptation: A case study in mission critical systems / Q. Yang, J. Lü, J. Xing, X. Tao, H. Hu, Y. Zou // IEEE 35th Annual Computer Software and Applications Conference Workshops (COMPSACW). – Washington : IEEE Computer Society, 2011. – P. 13–18.

5. Liu, L. Requirements cybernetics: elicitation based on user behavioural data / L. Liu, Q. Zhou, J. Liu, Z. Cao // J. Syst. Software. – Amsterdam : Elsevier, 2016.

И. Ю. Балашова, С. В. Шибанов, А. Ю. Афонин

Пензенский государственный университет, Пенза, Россия

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ БЛОКЧЕЙН В ПОСТРОЕНИИ ДЕЦЕНТРАЛИЗОВАННЫХ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Выделены группы блокчейнов в соответствии с доступом к данным. Описаны преимущества применения эксклюзивных блокчейнов в построении децентрализованных распределенных информационных систем.

Технология блокчейна представляет собой совокупность инновационных решений в области организации, хранения и обработки распределенных данных. Блокчейн – это распределенная база данных для обработки транзакций [1]. Одно из главных преимуществ блокчейна по сравнению с другими моделями распределенных баз данных – интеграция обработки данных, обеспечения корректности и безопасности в единый протокол, реализуемый алгоритмически и минимизирующий человеческий фактор.

Блокчейн-система работает по следующим правилам.

1) новые транзакции рассылаются всем узлам;

2) каждый узел объединяет пришедшие транзакции в блок;

3) каждый узел пытается подобрать хэш блока, удовлетворяющий текущей сложности;

4) как только такой хэш найден, блок отправляется в сеть;

5) узлы принимают этот блок, только если все транзакции в нем корректны и не используют уже потраченные средства;

6) свое согласие с новыми данными узлы выражают, начиная работать над следующим блоком и используя хэш предыдущего в качестве новых исходных данных.

Основным принципом функционирования технологии блокчейна является прозрачность совершаемых операций с невозможностью их изменения лицами, не имеющими к ней санкционированного доступа. Все транзакции в блокчейне хранятся в едином реестре. Поскольку транзакции полностью упорядочены по времени, текущее состояние системы

определяется исключительно этим реестром транзакций. Хранение полной истории изменений состояния системы имеет свои преимущества, например, возможность определить состояние системы в произвольный момент времени, просто «проиграв» заново соответствующие транзакции.

В идеальном случае, обработка транзакций в рамках блокчейн-технологии удовлетворяет следующим свойствам [2]:

- транзакции согласованы с текущим состоянием системы;
- транзакции авторизованы;
- транзакции неизменяемы: после того, как транзакция записана в реестр, ее должно быть невозможно изменить;
- транзакции конечны: после того, как транзакция записана в реестр, ее будет невозможно удалить оттуда;
- устойчивость к цензуре: если транзакция удовлетворяет всем правилам блокчейна, она должна быть, в конце концов, в него добавлена.

Соответствие текущему состоянию системы удовлетворяется за счет проверки транзакции совместно с этим состоянием, хранящимся в защищенной от злоумышленников памяти. Поскольку текущее состояние системы можно восстановить при помощи блокчейна, предположение защищенности не сужает безопасность системы в целом. Это предположение вводит ограничение на блокчейн, которое заключается в организации хранения транзакций таким образом, чтобы надежная верификация транзакций занимала не слишком много времени.

Проблема авторизации решается за счет использования криптографии с открытым ключом. Каждому пользователю системы выдается пара из секретного и открытого ключа; открытый ключ может быть без проблем опубликован для определения цифровой личности пользователя, так как секретный ключ невозможно вывести из открытого. Использование цифровых подписей решает не только проблему авторизации, но также и проблему изменяемости транзакций. Если цифровые подписи используются для всех транзакций в блокчейне, злоумышленник, получивший внутренний доступ к системе, не может изменить эти транзакции.

Неизменяемость и конечность транзакций в системе на основе блокчейна достигается при помощи разделения транзакций в блоки, упорядоченные во времени, и расчете криптографической хэш-функции для каждого из блоков.

При использовании блокчейн-технологии состояние системы, которое является общим для всех узлов – это сам блокчейн; новые транзакции изначально не принадлежат ни одному из блоков, т.е. являются неподтвержденными. Протокол блокчейна определяет правила по созданию новых блоков; когда новый блок создан, он распространяется по сети, и множество неподтвержденных транзакций на всех узлах системы меняется в соответствии с новым состоянием (в частности, противоречащие новому состоянию транзакции удаляются). Это правило гарантирует, что, в то время как неподтвержденные транзакции могут противоречить друг другу, транзакции в блокчейне всегда соответствуют протоколу его построения. Адекватный протокол минимизирует вероятность того, что несколько узлов одновременно добавят «свои» блоки в блокчейн. Например, если в системе существует ограниченное количество известных узлов, они могут создавать блоки по очереди с интервалом, который в несколько раз превышает интервал распространения блока по системе.

Выделяют четыре вида блокчейнов в соответствии с доступом к данным [4]:

- открытый блокчейн – это блокчейн, в котором не существует ограничений на чтение данных блоков (при этом данные могут быть зашифрованы) и ограничений на отсылку транзакций для включения в блокчейн;
- закрытый блокчейн – это блокчейн, в котором прямой доступ к данным и к отправке транзакций ограничен определенным узким кругом организаций;
- общедоступный (инклюзивный) блокчейн – это блокчейн, в котором не существует ограничений на личность обработчиков транзакций (т.е., пользователей, которые могут создавать блоки транзакций);

• эксклюзивный блокчейн – это блокчейн, в котором обработка транзакций осуществляется определенным списком субъектов с установленными личностями.

В эксклюзивных блокчейнах существует несколько уровней доступа к данным блокчейна, например:

• чтение транзакций из блокчейна, возможно, с определенными ограничениями (например, у клиента может быть доступ только к транзакциям, которые затрагивают его);

• предложение транзакций для включения в блокчейн;

• создание новых блоков транзакций и добавление блоков в блокчейн.

В то время как третий уровень доступа в эксклюзивных блокчейнах предоставляется ограниченному кругу учреждений (например, лицензированным обработчикам транзакций), доступ к блокчейну не обязан ограничиваться этими учреждениями. Напротив, учреждения, управляющие эксклюзивным блокчейном, теоретически могут

• предоставить доступ на чтение транзакций и заголовков блоков (возможно, ограниченный) для клиентов с целью обеспечения технологичного, прозрачного и надежного способа проверки сохранности средств клиентов;

• предоставить полный доступ на чтение данных блокчейна для регулирующих учреждений для соответствия законам;

• предоставить всем лицам и учреждениям к доступом к данным исчерпывающее описание протокола блокчейна, который включает в себя объяснение возможных взаимодействий с данными блокчейна.

Эти шаги могут облегчить независимый аудит и проверку целостности данных блокчейна, например, регулирующими учреждениями.

Если база данных блокчейна полностью непрозрачна для клиентов (т.е. клиенты не имеют доступ к данным блокчейна), то безопасность блокчейна уменьшается. Поскольку встроенный протокол авторизации транзакций является одним из ключевых моментов блокчейн-технологии, его игнорирование в пользу централизованных решений может отрицательно сказаться на защищенности системы. Кроме того, так как транзакции в этом случае доступны ограниченному числу компьютеров, возникают риски вмешательства человеческого фактора в функционирование этих компьютеров, причем клиенты не смогут выявить это вмешательство. Таким образом, непрозрачный дизайн блокчейнов по своей сути подрывает два ключевых аспекта блокчейн-технологии: децентрализацию и опору на алгоритмически заложенные правила обработки транзакций без вмешательства человека. В целом, любое взаимодействие с данными блокчейна в обход его протокола вводит в систему уязвимости.

Блокчейн соединяет в себе преимущества нескольких технологий: децентрализованных сетей, распределенных баз данных и криптографических алгоритмов. Главным достоинством технологии блокчейн является предельная достоверность и защищенность от изменения записей, что достигается за счет применения специальных криптографических алгоритмов и азмещения полной версии базы данных на всех узлах сети. Дополнительным преимуществом блокчейна является возможность реализовать в его среде обмен любыми электронными активами без посредников с сохранением приватности.

Юридические и технические ограничения, риски мошенничества и угрозы безопасности делают невозможным применение открытых блокчейнов в распределенных информационных системах финансовых и государственных учреждений. Эксклюзивные блокчейны привлекательны для учреждений, работающих с реестрами и финансовыми системами учета, поскольку большинство систем законодательства предусматривает необходимость регистрации обработчиков транзакций. Эти блокчейны могут сформировать более контролируемую и прогнозируемую среду, чем общедоступные блокчейны.

Библиографический список

1. Siraj Raval, Decentralized Applications: Harnessing Bitcoin’s Blockchain Technology. – Published by O’Reilly Media, Inc., 2016. – 104 p.

2. Distributed Ledger Technology: beyond block chain. A report by the UK Government Chief Scientific Adviser. Available at. – URL: https://www.gov.uk/government/uploads/system/uploads/attach-ment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf (accessed 4 April 2017).

3. Unlocking economic advantage with blockchain: a guide for asset managers. Available at: – URL: http://www.oliverwyman.com/our-expertise/insights/2016/jul/unlocking-economic-advantage-with-blockchain.html (accessed 7 April 2017).

Д. В. Ломизов, А. В. Дубравин

Пензенский государственный университет, Пенза, Россия

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ

Рассмотрен вопрос автоматизации процесса тестирования программно-аппаратных комплексов. В качестве средства автоматизации описан модуль на базе платформы Arduino.

В своей работе в отделе тестирования ООО НТП «Криптософт» мы часто сталкиваемся с задачей проверки функционирования программно-аппаратных комплексов. Многие из них реализованы на основе аппаратных средств, имеющих собственную консоль управления. Тестирование функционала данных комплексов, выполняющегося под управлением операционной системы можно достаточно легко автоматизировать, используя как мощные интегрированные средства тестирования (Ranorex, TestComplete), так и отдельные платные или бесплатные фреймворки. В то же время, проверка функционирования консолей управления долгое время оставалась задачей исключительно для ручного тестирования.

Таким образом, актуальным является создание средства, позволяющего автоматизировать данный вид работ частично или полностью.

Рассмотрим преимущества и недостатки автоматического тестирования:

- повторяемость тестов. Тест будет выполняться абсолютно однообразно раз за разом, что позволит во-первых без лишних усилий проверить исправление найденной в процессе такого тестирования ошибки, а во-вторых создать постоянно расширяемый набор тестов для регрессионного тестирования;
- высвобождение рабочего времени сотрудников для проведения тестов, требующих участия человека, например оценка эргономики интерфейса, выработка рекомендаций для разработчиков и т.п.
- возможность автоматически собирать и анализировать информацию о реакции системы на тест путем вывода и анализа логов;
- возможность проводить длительные тесты без участия человека, в том числе в нерабочее время, что повышает эффективность использования оборудования и снижает общие временные затраты на проведение полного цикла тестирования комплекса;
- возможность проведения интенсивных нагрузочных тестов, в том числе нагрузочное тестирование отдельных узлов аппаратуры, к примеру – контроллера Ethernet или USB;
- единожды созданные тесты могут многократно использоваться повторно, как для тестирования текущего проекта, так и для вновь возникающих задач, как в адаптированном, так и в исходном виде. При накоплении достаточно большого массива тестов, время на подготовку к тестированию нового устройства будет сокращаться за счет использования подобных заготовок;

- при проведении автоматического тестирования могут быть пропущены ошибки, на которые мог бы «попутно» обратить внимание тестировщик и, выполнив дополнительные действия, локализовать ошибку;
- для тестирования аппаратных средств необходима разработка как аппаратных же модулей, взаимодействующих с тестируемым устройством, так и программного обеспечения, использующегося для управления процессом тестирования, сбора и анализа информации, полученной в результате. Это влечет за собой дополнительные финансовые и временные затраты, а также увеличивает требования к необходимой квалификации сотрудников;
- при анализе существующих средств тестирования не найдено готовых бесплатных или коммерческих решений, направленных на тестирование аппаратной составляющей при функционировании до загрузки операционной системы.

Сформулируем требования, предъявляемые к необходимому нам средству автоматизированного тестирования.

При исследовании функционирования аппаратных средств, нас интересует реакция на следующие виды воздействий:

- срабатывание различных внешних датчиков;
- аппаратные сбои в работе ПК;
- действия пользователя по вводу информации.

Кроме того, важной задачей является получение информации о действиях, выполняемых программным обеспечением комплекса.

На этапах отладки и тестирования получение и журналирование информации о выполняемых ПО комплекса действиях осуществляется путем вывода данной информации в serial-порт, которых на платах, входящих в состав комплекса, может быть несколько. Данные с этих портов поступают на ПК тестировщика, где хранятся и обрабатываются. При этом в процессе разработки аппаратных прошивок контроллеров возможно изменять и расширять состав выводимой информации.

Для реализации вышеперечисленных требований было использовано устройство на основе платформы Arduino. Управление данным контроллером осуществляется с ПК тестировщика через serial-интерфейс. Обратная связь с тестируемым устройством в настоящее время отсутствует. Информация о выполнении тестов получается напрямую с тестируемого устройства на ПК тестировщика.

Рассмотрим основные функции разработанного устройства.

Сигналы внешних датчиков, используемых в нашей аппаратуре, чаще всего сводятся к замкнутому/разомкнутому контакту, либо наличию логического уровня на выходе датчика. При тестировании к разъемам датчиков подключаются шлейфы, на которые подается сигнал, инициированный контроллером. Предусмотрены команды «замкнуть контакты», «разомкнуть контакты», «послать 0», «послать 1», а также имитация «дребезга контактов», при которой контакты шлейфа многократно замыкаются и размыкаются через случайные промежутки времени в интервале от 20 до 40 миллисекунд в течение заданного отрезка времени.

Подобным образом реализована и эмуляция аппаратных сбоев. Контроллер управляет прерывателями, включенными в разрыв кабелей питания и/или информационных шлейфов. Реализованы команды «замкнуть цепь», «разомкнуть цепь».

Эмуляция действий пользователя по вводу информации реализована следующим образом. USB-контроллер на плате Arduino прошит модифицированной прошивкой, которая позволяет при подключении к ПК, на котором установлен тестируемый программно-аппаратный комплекс, эмулировать USB HID-устройство. Для управления виртуальной клавиатурой используются команды «клавиша нажата», «клавиша отпущена» (эмулируется соответственно нажатие и отпускание клавиши), а так же «клик» по клавише, при котором эмулируется кратковременное (на случайный интервал от 30 до 75 миллисекунд) нажатие и последующее отпускание клавиши.

Использование разработанного устройства позволяет заметно сократить время, затрачиваемое сотрудником на проведение рутинных проверок и оставляет возможность тестировать комплекс в «свободном поиске».

В качестве дальнейшего развития предполагается дополнить устройство USB-хабом с управляемым подключением/отключением к хосту включенных в хаб устройств.

Библиографический список

1. Борисова, Т. М. Особенности автоматизации тестирования программно-аппаратных СЗИ / Т. М. Борисова. – URL: http://www.okbsapr.ru/borisova_2013_1.html

О. С Дорофеева, Б. В. Казаков, И. А. Казакова

Пензенский государственный университет, Пенза, Россия

ПРИМЕНЕНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПОДХОДА ПРИ РАЗРАБОТКЕ АВТОМАТНОЙ МОДЕЛИ ПРОГРАММНЫХ СИСТЕМ

Рассматриваются вопросы применения объектно-ориентированного подхода при разработке автоматной модели программных систем.

В формулировке цели разработки программной системы участвуют предметы и понятия реального мира, имеющие отношение к разрабатываемой программной системе. При объектно-ориентированном подходе эти предметы и понятия заменяются их моделями, т.е. определенными формальными конструкциями, представляющими их в программной системе.

Формальный характер моделей позволяет получить формальную модель разрабатываемой программной системы как композицию формальных моделей ее компонентов.

Таким образом, объектно-ориентированный подход помогает справиться с такими сложными проблемами, как

– уменьшение сложности программного обеспечения;

– повышение надежности программного обеспечения;

– обеспечение возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов;

– обеспечение возможности повторного использования отдельных компонентов программного обеспечения.

Применение объектно-ориентированного подхода позволяет разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы.

Автоматная модель программной системы представляет собой граф, вершины которого называются состояниями и изображают ситуации (или классы ситуаций), в которых может находиться система в различные моменты времени. Ребра графа могут иметь метки, изображающие действия, которые может исполнять система.

Функционирование системы изображается в данной модели переходами по ребрам графа от одного состояния к другому. Если проходимое ребро имеет метку, то эта метка изображает действие системы, исполняемое при переходе от состояния в начале ребра к состоянию в его конце.

Построение модели системы начинается с анализа предметной области. Стандартными средствами [1] строится статическая модель системы и, согласно автоматному подходу, в статической модели выделяются объекты и управляющие ими автоматы.

Автоматная методология описывает динамическую модель системы с помощью так называемой *модели управления*. Построение этой модели состоит из следующих этапов:

– в статической модели выделяются автоматы, все возможные объекты, являющиеся источниками событий, и объекты управления, которые также могут являться источниками входных воздействий;

– все объекты компонуются на схеме связей автомата, которая представляется с помощью диаграммы классов *UML*;

– каждое входное и выходное воздействие реализуется вручную в соответствии с его функциональностью;

– для каждого автомата описывается поток управления, представленный в виде графа переходов, заданного диаграммой состояний *UML*, в которой на дугах указываются события, входные и выходные воздействия;

– с использованием подходящего инструмента разработки программного обеспечения диаграмма классов и графы переходов могут быть автоматически преобразованы в исходный код на целевом языке или интерпретированы специальным интерпретатором.

Основными составляющими функциональной модели являются диаграммы потоков данных [2], которые показывают потоки значений от внешних входов через операции и внутренние хранилища данных к внешним выходам.

Помимо диаграмм потоков данных, функциональная модель описывает смысл операций объектной модели и действий динамической модели, а также ограничения на объектную модель.

Согласно функциональной модели, все операции можно разделить на три категории: *запросы*, *действия* и *активности*. Запросом называется операция без побочных эффектов над видимым извне объекта его состоянием. Примером запроса может служить входное воздействие автоматной модели. Действием называется операция, имеющая побочные эффекты, которые могут влиять на целевой объект и на другие объекты системы, которые достижимы из целевого объекта. Примером действия является выходное воздействие.

Все операции в модели должны быть специфицированы. Обычно спецификация операции содержит ее характеристики (имя, количество, порядок и типы параметров, количество, порядок и типы возвращаемых значений) и описание ее эффекта (действия, преобразования). Операции автоматной модели не имеют параметров. Выходные воздействия к тому же не имеют возвращаемых значений. Однако и они, и входные воздействия, могут получать и изменять данные в памяти автомата. Запрашиваемые действием элементы данных называются *входными данными* (или параметрами). Сохраняемые в память автомата данные называются *выходными данными действия*. Кроме того, события в автоматной модели также несут с собой данные – свои параметры, которые называются *выходными данными события*.

Итак, для автоматной модели целесообразно описывать имена операций (действий и событий), входные и выходные данные, а также эффект операций. Для описания эффекта операций в функциональной модели могут использоваться:

– математические формулы;

– табличные функции: таблицы, сопоставляющие выходные значения входным;

– уравнения, связывающие входные и выходные значения;

– аксиоматическое определение пред- и постусловий;

– таблицы принятия решений;

– псевдокод;

– естественный язык.

Таким образом, для спецификации воздействий и событий автоматной модели достаточно указания списков входных и выходных данных, а также описания эффекта этих действий на естественном языке.

Библиографический список

1. Гайсарян, С. С. Объектно-ориентированные технологии проектирования прикладных программных систем / С. С. Гайсарян. – URL: http://www.citforum.ru/programming/oop_rsis/index.shtml

2. Казаков, Б. В. Объектно-ориентированная технология разработки программных систем и диаграммы потоков данных / Б. В. Казаков, И. А. Казакова // Информационные технологии в науке и образовании. Проблемы и перспективы : сб. науч. ст. IV Ежегодной межвуз. науч.-практ. конф. (МНПК – 2017), (г. Пенза, 15 марта 2017 г.). – Пенза : Изд-во ПГУ, 2017. – С. 121–124.

А. А. Доломанов

Пензенский государственный университет, Пенза, Россия

ДИСТАНЦИОННОЕ УПРАВЛЕНИЕ СРЕДСТВАМИ ИЗМЕРЕНИЙ НА ПРИМЕРЕ SMB100 ROHDE&SCHAWZ

Рассматриваются вопросы автоматизации поверки СИ, что исключает человеческий фактор, ускоряет производство и автоматизирует заполнение документации, при этом не исключая режим ручного управления.

Актуальность автоматизации поверки средств измерений (СИ) заключается в возможности исключить человеческий фактор и ускорить производство и автоматизирует заполнение документации, при этом не исключая режим ручного управления. Данная статья содержит информацию:

– о настройке генератора сигналов для работы в режиме дистанционного управления;

– об общих принципах дистанционного управления программируемыми измерительными приборами, включая описание структуры команд и синтаксиса применительно к стандарту SCPI, а так же описание выполнения команд и содержимого регистров состояния.

Прибор имеет следующие интерфейсы для дистанционного управления:

– интерфейс шины IEC/IEEE, соответствующий стандарту IEC 625.1/EEE 488.2;

– сетевой интерфейс LAN: в сетевой карте используется протокол 100МГц Ethernet IEEE 802.3u; этот протокол основывается на стандарте VXI-11.

– интерфейс USB: интерфейс B-типа (USB-устройство) используется для дистанционного управления.

Разъемы дистанционного управления расположены на задней стенке прибора, контроллер дистанционного управления может подключаться к ним на прямую или через локальную сеть (LAN).

Необходимым условием для дистанционного управления прибором по сетевому или USB-интерфейсу является установка VISA-библиотек на стороне контроллера. VISA – это стандартизированная библиотека программируемого интерфейса, содержащая функции ввода и вывода для взаимодействия с приборами.

Для дистанционного управления используются команды стандарта SCPI (Standard Commands for Programmable Instruments), имеющие вид сообщений. Команды, не входящие в стандарт SCPI, тем не менее, соответствуют синтаксическим правилам этого стандарта. Прибор поддерживает версию стандарта SCPI от 1999 г. Сам же стандарт SCPI основывается на стандарте IEEE 488.2 и предназначен для стандартизации команд конкретных устройств, обработки ошибок и регистров состояния.

Изложенная ниже простая и короткая последовательность действий позволит быстро привести прибор в рабочее состояние и установить его основные функции. При этом

адрес прибора на шине IEC/IEEE (при выпуске прибора установлен адрес 28) менять не следует.

– соединить прибор с контроллером через кабель шины IEC/IEEE, затем включить их;

– составить и запустить на контроллере следующую программу:

```
CALL IBFIND (“DEV1”,                ‘Открыть порт для обмена данных с прибором
generator %)
CALL IBPAD (generator %, 28)        ‘Сообщить контроллеру адрес прибора
CALL IBWRT (generator %,            ‘Сброс прибора
“*RST; *CLS”)
CALL IBWRT (generator %,            ‘Установить частоту 50 МГц
“FREQ 50MHz ”)
CALL IBWRT (generator %,            ‘Установить уровень сигнала выходе – 7,3 дБм
“POW – 7. 3dBm”)
CALL IBWRT (generator %,            ‘Включить выход ВЧ
“OUTP : STAT ON”)
CALL IBWRT (generator %,            ‘Установить источник амплитудной модуляции LFGEN
“AM : SOUR INT”)
CALL IBWRT (generator %,            ‘Установить частоту модуляции 15 кГц
“LFO : FREQ 15kHz”)
CALL IBWRT (generator %,            ‘Установить глубину амплитудной модуляции 30 %
“AM 30PCT”)
CALL IBWRT (generator %,            ‘ Включить амплитудную модуляцию
“AM : STAT ON”)
```

В результате выполнения этих команд на ВЧ-выход генератора сигналов будет выводиться амплитудно-модулированный сигнал.

Чтобы вернуться в режим ручного управления, нажмите клавишу LOCAL на передней панели прибора.

При включении питания прибор всегда устанавливается в режим ручного управления, при этом управления может производится или органами управления на передней панели прибора, или мышью и внешней клавиатурой.

При дистанционном управление по шине IEC/EEE прибор устанавливается в режим дистанционного управления (состояние REMOTE) по средствам любой адресной команды.

При дистанционном управлении через сеть Ethernet режим дистанционного управления посредствам команды не устанавливается. Прибор должен быть предварительно установлен именно в состояние REMOTE, например, посылкой команды &GTR (перейти в состояние REMOTE).

В состоянии REMOTE управление прибором органами управления на передней панели прибора или мышью и внешней клавиатурой невозможны. Однако возможно открытие меню (например, для проверки настроек). Кнопки и области настроек не выделены и активированы быть не могут. В строке состояния отображается сообщение REMOTE.

Прибор остается в режиме дистанционного управления REMOTE до тех пор, пока не будет задан режим ручного управления – вручную (клавишей LOCAL на передней панели прибора) или командой &GTR через интерфейс дистанционного управления.

Переход от дистанционного управления прибором к ручному и наоборот каких-либо других настроек прибора не затрагивает. В режиме дистанционного управления клавиши на передней панели, а также мышь и внешняя клавиатура (если они подключены) могут быть заблокированы командой SYST : KLOC ON. В этом случае меню открыть нельзя, а переход не ручное управление возможен только посредствам соответствующей команды дистанционного управления. Клавиша LOCAL в этом случае не действует.

Действие этой клавиши в отдельности может быть заблокировано командой &LLO.

Для осуществления дистанционного управления по интерфейсу USB компьютер и прибор должны быть соединены через USB-интерфейс типа B.

Для осуществления USB-соединения на контроллере должна быть установлена библиотека VISA. При установлении USB-соединения VISA автоматически обнаружит и сконфигурирует генератор R&S SMB. При этом ввода адресной строки и установки отдельного драйвера не требуется.

Используемая адресная строка USB:

USB::<vendor Id>::<product Id>::<серийный номер>

Для Rohde&Schwarz строка “vendor Id” – это 0x0AAD, строка “product Id” – 0x54, а серийный номер – это индивидуальный серийный номер, нанесенный на заднюю стенку прибора.

Пример:

“USB::0x0AAD::0x0054::100001::INSTR”

Таким образом, представленными выше вариантами дистанционного управления СИ, оптимальными для вывода., можно реализовать осуществлять автоматическую поверку и калибровку новых источников питания, поддерживающих дистанционное управление. Имея протоколы поверки, с внесенными данными о пределах погрешности, достаточно всего лишь подключить современный источник питания к электронной нагрузке (АКИП-1303) и запустить программу дистанционного управления. Оба средства измерения подключаются к ДУ через ПК посредствам одного из вариантов представленных выше, т.е. по USB-интерфейсу либо по шине IEC/IEEE. Программа автоматически проводит поверку, используя пределы погрешности, с последующим заключением о поверке/калибровке и выводом на экран для печати протокола поверки. Что исключает человеческий фактор, ускоряет производство и автоматизирует заполнение документации, но не исключает ручного управления.

В. А. Калашников, К. Д. Чиркин

Пензенский государственный университет, Пенза, Россия

РЕАЛИЗАЦИЯ СИСТЕМЫ ТРАНСФОРМАЦИИ БУЛЕВЫХ ФОРМУЛ НА ЯЗЫКЕ PROLOG

Рассматриваются принципы построения системы трансформации и упрощения формул булевой алгебры через основные тождества с помощью средств логического программирования Prolog. Подчеркиваются преимущества языка Prolog по отношению к традиционным языкам программирования при решении некоторых задач. Приводятся примеры работы программы преобразования булевых формул.

При проектировании интеллектуальных систем программист сталкивается с рядом сложностей, возникающих в связи с использованием императивного метода программирования. При традиционном программировании проблема представлена через описание метода ее решения непосредственно процедурно-ориентированным языком. Другими словами, *решением* служит описание алгоритма действий, и *решение* всецело зависит от точности описания этого алгоритма.

Язык Prolog сосредоточен вокруг небольшого набора основных механизмов: сопоставление с образцом, древовидное представление структур данных и автоматический перебор с возвратами. Тем самым Prolog избавляет программиста от утомительного написания сложных алгоритмов анализа и обработки информации.

Prolog является декларативным языком программирования: логика программы выражается в терминах отношений, представленных в виде фактов и правил. Для того чтобы

инициировать вычисления, выполняется специальный запрос к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь». Для обобщенных запросов с переменными в качестве аргументов созданная система Prolog выводит конкретные данные в подтверждение истинности обобщенных сведений и правил вывода.

В данной статье представлена программа, осуществляющая символьное преобразование формул булевой алгебры. Программа использует основные законы булевой алгебры. Пример трансформации выражения с поэтапным выводом дан на рис. 1.

```
[2] 20 ?- exp_transform((x v y) =>((~(x)=>y)& ~(~(x)&y))).

x v y=> (~x=>y)& ~ (~x&y)
~ (x v y)v (~x=>y)& ~ (~x&y)
~x& ~y v (~x=>y)& ~ (~x&y)
~x& ~y v (~ (~x)v y)& ~ (~x&y)
~x& ~y v (x v y)& ~ (~x&y)
~x& ~y v (x v y)& (~ (~x)v~y)
~x& ~y v (x v y)& (x v ~y)
~x& ~y v x v y& ~y
~x& ~y v x v false
~x& ~y v x
~y v x
true.
```

Рис. 1. Пример работы программы преобразователя

При получении выражения от пользователя программа обращается к базе фактов, являющейся набором основных свойств и тождеств булевой алгебры. Цикл работы программы представлен следующим образом: механизм Prolog сопоставляет входное выражение с образцом и в случае успеха выводит преобразованное выражение. В новом цикл посылается уже преобразованное выражение. Цикл длится до тех пор, пока Prolog не сможет найти соответствие в базе фактов. Преобразование основывается на принципе «от большего к меньшему», для этого используются удаление тавтологии и законы поглощения и склеивания. Блок-схема алгоритма трансформации представлена на рис. 2.

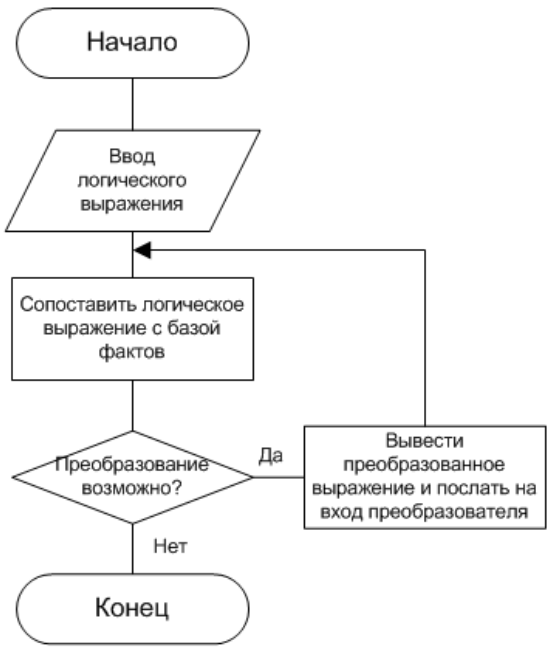


Рис. 2. Блок-схема алгоритма трансформации булевых формул

Программа дает возможность сравнить две любые булевые формулы между собой. Перед тем как сравнить, программа пытается преобразовать выражения в простую форму. Когда преобразование становится невозможным, программа начинает сравнение, опираясь на базу фактов, которая состоит из основных тождеств булевой алгебры. Результат работы программы представлен на рис. 3.

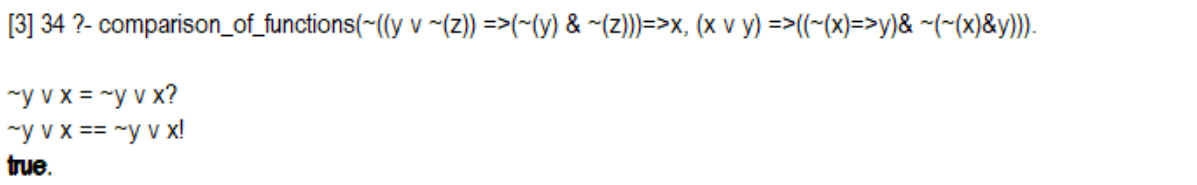


Рис. 3. Пример работы программы преобразования и сравнения

От программиста не требуется написания сложного алгоритма, по которому выражение разделяется на подвыражения, которые сопоставляются по образцу, и в случае соответствия выполняется преобразование. Prolog делает это сам благодаря некоторым особенностям работы его механизмов.

В качестве наглядного примера представлен участок кода программы преобразователя для демонстрации работы Prolog-машины.

```
exp_transform(Term, Result_Term):-
      transform(Term, Result_Term),!,
      nl, write(Term),
      exp_transform(Result_Term, New_Result_Term).

exp_transform(Term, _ ):-
      nl, write(Term).
```

Программа имеет два одноименных предиката – это своего рода аналог If…Then конструкции. Первый предикат осуществляет трансформацию выражения, второй выводит финальное выражение. Механизм Prolog читает код программы по порядку и при обращении к предикату пытается доказать первый вариант в списке. Если Prolog не может достичь цели (трансформация не возможна), то он начинает поиск других вариантов в коде программы. Необходимо подчеркнуть, что переменные в языке Prolog могут хранить любую информацию, в том числе сложные выражения.

В качестве наглядного примера представлен участок кода базы фактов программы преобразователя для демонстрации работы Prolog-машины.

```
      transform((X v Z) & Z, Z).
      transform(X & Y, X1 & Y):-
            transform(X, X1).
```

Пользователь отправляет в качестве аргумента выражение (a v b) & b, после этого Prolog сопоставляет выражение с образцом. В первом случае переменная X = a , Z = b. Если первый факт не будет доказан, то Prolog перейдет ко второму варианту и разделит выражение на подвыражение с помощью рекурсивного вызова предиката. В нем X = (a v b), Y = b. Из-за этих особенностей механизмов языка Prolog код программы, в сравнении с традиционными языками, занимает малый объем.

Библиографический список

1. Клоксин, У. Программирование на языке Пролог / У. Клоксин, К. Меллиш. – URL: https://www.e-reading.club/book.php?book=96404

2. Братко, И. Алгоритмы искусственного интеллекта на языке PROLOG / И. Братко. – М. : Вильямс, 2004. – 640 с.

Е. В. Трусов

Пензенский государственный университет, Пенза, Россия

QT QUICK CONTROLS

Рассказано о библиотеке qt quick controls. Описаны ее возможности, положительные стороны и недостатки.

Qt известна благодаря своей кроссплатформенности позволяющая делать гибкие GUI приложения, с удобным интерфейсом. В него встроен редактор форм с большим набором встроенных компонентов. Для стандартного приложения их вполне достаточно, но порой разработчикам хочется добавить свой собственный компонент, с индивидуальной графикой и логикой. Qt quick controls с qml позволяет это сделать.

Еще в qt 4.7 появилась возможность создавать пользовательский интерфейс (qt quick controls). На протяжении долгого времени разработчики занимались исправлением недочетов и доработками. Но все время не хватало некоторых возможностей: правильного масштабирования на High DPI экранах; компонентов, специфичных для мобильных устройств и стилизации уже существующих компонентов.

Но первые две проблемы решались разработчиками при помощи умножения размеров на коэффициент и написанием собственных компонентов соответственно. Позже в qt quick controls появились готовые пользовательские компоненты в открытом доступе, но это не сильно спасло ситуацию.

Начиная с версии qt 5.7 quick controls потерпел многие положительные изменения и стал называться quick controls 2. В новой версии многое было переписано практически с нуля, были решены проблемы первой версии. Что касается стилизации, то она потерпела большие переработки: отлов событий вынесли в c++, избавились от некоторых ненужных классов, потребляющих много ресурсов устройства (для фона и надписи, теперь вместо них можно использовать что угодно), время запуска приложений сократилось в разы (в 1-й версии для создания слайдера требовалось ~400 мс, во 2й меньше 20).

Qt quick controls может с успехом использоваться в играх со сложным UI. С его помощью создание собственной библиотеки для отображения и инструментов для удобного редактирования может упроститься в разы. Можно и по старинке использовать такие библиотеки как scaleform, coherent UI, awesomium, но у них всех есть очень большой недостаток: огромные тормоза, особенно это заметно на мобильных устройствах.

Что касается причины из-за чего в играх не используют qt widgets – они конфликтуют с такими библиотеками как openGL или directX, Каждый из них хочет быть хозяином главной нити, а потому их совместное использование невозможно.

Но qt quick отлично справится с задачей отрисовки: все его компоненты рисуются ускоренно, а возможность динамического изменения UI программно добавляет скорости, так как не требуется полностью перерисовывать всю модель. Это отлично подходит для динамического изменения внешнего вида игры, быстрой настройки и т.п.

Принципы работы интерфейса, написанного на qml, в следующем: вначале вы описываете различные объекты, располагаете их относительно друг друга (в qml такой же динамически изменяемый интерфейс, как в WPF, а это значит, что координаты фиксировать не нужно). Дальнейшую отрисовку и отображение qt берет на себя.

Помимо этого, qml тесно интегрирован с JavaScript, в результате в компоненты очень удобно встраивать логику: привязки событий, таймеры и просто вспомогательные функции. Все описания можно делать как внутри qml-файлов, так и выносить в отдельные js-файлы.

Что касается конструктора форм, то он не совершенен, т.к. не позволяет редактировать все свойства объекта. Но эти проблемы можно преодолеть, описывая недостающие свойства при помощи текстового редактора.

Так же у qt есть существенный минус: если вы думаете, что после того, как вы написали свой код, нарисовали красивый UI и упаковали ресурсы, у вас все готово к релизу – то это не совсем так.

Стоит напомнить, что qt – это огромное количество dll и qml-модулей. Для того, чтобы иметь возможность переносить программу с пк на пк, все задействованные в программе модули нужно переносить на устройство (именно при помощи их обеспечивается кроссплатформенность). Но чтобы их таскать, сначала нужно найти, а они находятся в разных местах огромной установочной директории. При использовании qt creator, при сборке release версии необходимые модули будут скопированы в папку с приложением, а при использовании другой IDE, придется самому искать все нужные dll и модули.

При разрабатываете игры под pc, или создания крупного мобильного приложения 25–40 Мб библиотек не будут проблемой. Если же вы создаете игру под android, с рекомендованными 50 Мб на приложение, то стоит подумать, нужно ли связываться с данной библиотекой.

Если вам для проекта нужна готовая UI библиотека, а написать свою нет времени, то qtQuick/qml станет хорошим выбором, так как выигрывает по производительности у конкурентов.

При интеграции в проект qt могут возникнуть проблемы: придется изменить инициализацию и логику основного цикла. При создании нового проекта это можно пережить, а быстро сменить UI с одного на qtQuick/qml вряд ли получится.

К qtQuick/qml прилагается подробная документация расписаны большинство функции, есть пояснения с примерами, иногда этого недостаточно. В подобных случаях, есть возможность заглянуть в исходный код (он полностью открыт).

Если сравнивать qml с чем-то похожим, например, xaml, то можно выделить явные преимущества: он позволяет создавать кроссплатформенные программы и очень тесно завязан с JavaScript. Создание новых компонентов происходит очень просто, всю логику можно поместить в него и основной код будет не захламлен. В итоге, я хочу сказать, что плюсы данная библиотека очень хороша, хоть и не сильно известна, она имеет сильные стороны, перевешивающие ее минусы.

Библиографический список

1. Официальная документация по библиотеке qt quick controls. – URL: http://doc.qt.io/qt-5/qtquickcontrols-index.html

2. Шлее, М. Qt 4.8 Профессиональное программирование на C++ / М. Шлее. – СПб : БХВ-Петербург, 2013.

Д. А. Родионов

Пензенский государственный университет, Пенза, Россия

АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОНТРОЛЯ РАБОТОСПОСОБНОСТИ НА БАЗЕ СБИС

Заводы и предприятия, выпускающие радиодетали и микросхемы, после изготовления, но до отправки их на склад, подвергают их контролю на работоспособность. Однако радиодетали и микросхемы, пройдя отдел технического контроля (ОТК) на заводе, имеют некоторый процент брака в процессе транспортировок, монтажа или эксплуатации. Это влечет дополнительные затраты рабочего времени, средств для их выявления и замены (причем большую часть времени занимает именно выявление неисправных деталей).

Особенно важна полная работоспособность комплектующих деталей при сборке ответственных узлов управляющих систем, когда неисправность какого-либо одного элемен-

та микросхемы может повлечь за собой выход из строя других элементов, узлов, а возможно, и всего комплекса в целом.

Объединение аналоговых, цифровых и радиочастотных цепей в одной интегральной микросхеме и увеличение плотности размещения элементов микросхемы приводит к росту числа выводов в новых поколениях микросхем. В результате усложняется процедура тестирования таких микросхем и увеличивается время их выполнения. Это влияет как на стоимость готовых изделий, так и на длительность производственного цикла. Таким образом, производители интегральных микросхем (ИМС) постоянно решают задачу повышения производительности тестирующей аппаратуры, при этом требования, которые предъявляются к ней, порой оказываются на грани технических возможностей. Например, при увеличении числа одновременно тестируемых элементов увеличивается число измерительных и питающих блоков. В то же время стандартные шины, посредством которых обеспечивается синхронизация и необходимый порядок проведения измерений, ограничивают информационный обмен как по скорости, так и по количеству участников.

Здесь хорошую помощь могло бы оказать устройство, которое позволяет очень быстро проверять работоспособность микросхем. Неплохо было бы иметь возможность проверки как новых (подготовленных для монтажа), так и уже демонтированных из платы ИМС. Проверять микросхему очень удобно, когда на плате предусмотрен разъем. Это позволяет проводить достаточно быструю проверку деталей, уменьшить время производства аппаратуры. Это позволит свести к минимуму риск ее выхода из строя, так как в этом случае полностью исключается ее нагрев и различные механические повреждения при монтаже/демонтаже.

В настоящее время на заводах очень большую популярность получили персональные IBM-совместимые компьютеры. Так как задача тестирования ИМС требует наличия высокоинтеллектуального устройства для выполнения алгоритмов тестирования, целесообразно проектировать именно приставку к компьютеру, которая будет подключаться через внешний порт, а не отдельное самостоятельное устройство. Это обусловлено наличием в стандартном комплекте IBM-совместимого компьютера многих компонент, которые необходимы при решении таких задач (микропроцессора, составляющего основу компьютера; жесткого диска, предназначенного для хранения информации; внешних портов ввода-вывода – последовательных COM1, COM2 и параллельного LPT и порта USB; клавиатуры и дисплея – для ввода и вывода информации соответственно).

Автором данной работы разработана относительно недорогая автоматизированная система контроля (АСК), подключаемая к IBM-совместимому компьютеру, предназначенная для тестирования электронных модулей на базе СБИС. АСК должна формировать управляющие сигналы, предназначенные для тестирования электронных модулей.

Проверка производится следующим образом:

К компьютеру посредством кабеля подключается АСК. В разъем, выведенного на его корпус, вставляется испытуемый модуль. На компьютере запускается управляющая программа. Она формирует управляющие сигналы в порт, которые в свою очередь поступают на входы модуля. Далее программа считывает сигналы с выходов проверяемого модуля, анализирует считанные данные и выводит на дисплей результат тестирования.

Устройство предназначено для:

1. Формирования управляющих сигналов, предназначенных для тестирования электронных модулей на базе СБИС, согласно электрической принципиальной схемы на проверяемый модуль;

В современных сложных цифровых печатных платах (ПП), в которых содержаться ПЛИС, имеется JTAG-интерфейс, который подключается к микросхемам.

Идея тестирования микросхем с помощью JTAG-интерфейса – это внутрисистемное тестирование или, как его еще называют, граничное сканирование (Boundary Scan). Суть метода заключается в тестировании ИМС через специальный небольшой разъем при помощи 4-проводного интерфейса, закрепленного стандартом IEEE 1149.1 еще в 1990 году. Этот интерфейс также часто называют JTAG-интерфейсом по названию группы специалистов, которые занимались его разработкой в период с 1985 по 1990 годы (Joint Test Access

Group). Имея в составе устройства одну или несколько микросхем, поддерживающих стандарт IEEE 1149.1, можно тестировать не только цепи, которые связанны с этими микросхемами, но и остальные элементы этой микросхемы, включая память, логику, резисторы и внешние разъемы. При переходе в режим граничного сканирования JTAG-компоненты отключают свою основную логику и переходят в режим тестирования, позволяя внешнему оборудованию управлять своими выводами и тестировать цепи. Инструменты, которые предназначены для тестирования при помощи JTAG-интерфейса, позволяют также провести внутрисистемное программирование Flash-памяти и ПЛИС. Огромное количество микросхем, используемых в современных разработках, поддерживают стандарт IEEE 1149.1 – это требование современной электронной индустрии.

Следует отметить, что JTAG-сканирование можно осуществлять и на системном уровне. Несколько устройств, которые объединены в одну плату, могут иметь общую JTAG-шину граничного сканирования и тестироваться в составе сложного изделия через один единственный разъем. Такое решение часто используется при производстве телекоммуникационного оборудования. Многие предприятия, в особенности оборонного и аэрокосмического приборостроения, практикуют проведение испытаний изделий в экстремальных условиях (критические температуры, повышенная влажность и т.п.). В этом случае граничное сканирование также является незаменимым помощником, так как позволяет проводить тестирование и определять «слабые» места на плате непосредственно в камере для испытаний.

Порт тестирования (TAP–TestAccessPort) представляет собой четыре или пять выделенных линий микросхемы: TCK, TMS, TDI, TDO и TRST(опционально).

Функциональное назначение этих линий:

- **TDI**(«вход тестовых данных») – вход последовательных данных периферийного сканирования. Команды и данные поступают в микросхему с этого вывода по переднему фронту сигнала линии TCK;
- **TDO**(«выход тестовых данных») – выход последовательных данных. Команды и данные выводятся из микросхемы с этого вывода по заднему фронту сигнала TCK;
- **TCK**(«тестовое тактирование») – тактирует работу встроенного автомата управления периферийным сканированием. Максимальная частота сканирования периферийных ячеек зависит от используемой аппаратной части и на данный момент ограничена 25…40 МГц;
- **TMS**(«выбор режима тестирования») – обеспечивает переход микросхемы в/из режима тестирования и переключение между различными режимами тестирования.
- **TRST**(«тестовый сброс») – данный сигнал добавляется в определенных случаях для инициализации порта тестирования, что необязательно, так как инициализация возможна путем подачи определенной последовательности сигналов на вход линии TMS.

Состояния автомата определяются сигналами линий TDI и TMS порта тестирования. Определенное сочетание сигналов TMS и TCK обеспечивает ввод команды для автомата и ее исполнение.

При тестировании сложных цифровых устройств граничное сканирование является наиболее оптимальным методом, учитывая относительно невысокую стоимость оборудования, низкие трудозатраты, а также то, что возможность тестирования уже заложена во многие цифровые микросхемы.

2. Автономной проверки электронных модулей (без ПК);

Для выполнения в большом количестве простых однотипных операций, не требующих наличие ПК, таких как тестирование кабелей, программирование ИМС, предусмотрен режим автономной проверки.

3. Входного контроля ИМС.

Входной контроль компонентов является неотъемлемой частью обеспечения качества продукции в производстве электроники. С повышением уровня сложности производимых изделий, повышается уровень ответственности компонента в изделии.

Проведение входного контроля необходимо для любых типов компонентов, от резисторов и до ИМС. Очень важно на этапе входного контроля оценить работоспособность ИМС.

Обычно, производители электроники на входном контроле ограничиваются визуальным осмотром и инструментальным контролем геометрии микросхем.

Однако, при отсутствии внешних повреждений корпуса и соответствии чертежам, могут наблюдаться дефекты ИМС, выявляемые лишь тестированием компонента. При проведении тестирования ИМС, ее серия и тип известны, и микросхема считается исправной, при условии соответствия всех контролируемых входных и выходных сигналов для данной ИМС нормативным требованиям технических условий.

АСК реализовывает следующие функции:

1) независимое управление цифровыми каналами (ЦК);
2) управление АСК с ПК;
3) автономная работа системы.

АСК удовлетворяет следующим требованиям:

1) напряжение питания – 5В;
2) количество независимых ЦК – 160;
3) управление по интерфейсу USB.

В результате исследований были выявлены следующие аналоги АСК:

1) PCI-1753 является 96-канальной платой цифрового ввода/вывода с шиной PCI. Для организации 96 каналов цифрового ввода/вывода каналы разделены на двенадцать 8-битных портов. Каждый порт может быть запрограммирован либо как порт ввода либо как порт вывода. Цена PCI-1753 составляет 21 700 руб. Недостатком этого устройства является невозможность одновременной побитовой настройки портов на прием/передачу сигналов. Цена разрабатываемого устройства при этом составляет 21 000 руб.;

2) USB-4750 – это 32-канальный модуль цифрового ввода/вывода с USB интерфейсом. Настройка работы данного модуля не вызывает затруднений, так как каждый его канал ввода/вывода соотносится с битом в порте ввода/вывода. Цена устройства составляет 21 500 руб. Недостатком этого устройства является малое количество ЦК;

3) NI 6581 представляет собой 54-канальный модуль цифрового ввода/вывода с различными вариантами напряжения – 1,8 В, 2,5 В, 3,3 В. Цена устройства составляет 82 000 руб. и это является его недостатком. Также для работы NI 6581 необходимо дополнительное дорогостоящее оборудование.

В табл. 1 отображены все положительные и отрицательные качества рассмотренных ранее аналогов по отношению к АСК.

Таблица 1

Сравнение аналогов разрабатываемого устройства

Наименование	Цена	Кол-во ЦК	Простота организации структуры
PCI-1753	+	–	–
USB-4750	+	–	–
NI 6581	–	–	–
АСК	+	+	+

Из таблицы видно, что АСК обладает большим преимуществом относительно его аналогов. Это достигается за счет использования другой, более простой элементной базы, реализации управления АСК при помощи USB-интерфейса и формирования управляющих сигналов различной формы.

Библиографический список

1. Ключев, А. О. Использование интерфейса JTAG для отладки встраиваемых систем / А. О. Ключев, Т. А. Коровьякова, А. Е. Платунов // Изввестия вузов. Приборостроение. – 2014. – 341с.

2. Алексеенко, А. Г. Проектирование радиоэлектронной аппаратуры на микропроцессорах: Программирование, типовые решения, методы отладки / А. Г. Алексеенко, А. А. Галицын, А. Д. Иванников. – М. : Радио и связь, 2014. – 272 с.
3. URL: http://kit-e.ru/articles/measure/2005_6_230.php (дата обращения: 14.09.2017).
4. URL: http://www.studfiles.ru/preview/1544154/ (дата обращения: 13.09.2017).

С. В. Шибанов, И. Ю. Балашова, П. П. Макарычев

Пензенский государственный университет, Пенза, Россия

ПЕРСПЕКТИВЫ И ПРОБЛЕМЫ ПРИМЕНЕНИЯ БЛОКЧЕЙНОВ ДЛЯ ПОСТРОЕНИЯ РАСПРЕДЕЛЕННЫХ КОРПОРАТИВНЫХ СИСТЕМ

Рассмотрены проекты на основе блокчейнов. Выделены проблемы распределенных блокчейн-систем и предложены подходы к их решению.

Используемые в настоящее время в финансовых и государственных учреждениях информационные технологии морально устарели. Ориентация на максимальное сближение с клиентом требует передачи права на доступ к данным третьим лицам и приложениям при существующем по умолчанию недоверии между участниками операций. Ограниченные возможности интеграции данных из различных систем не обеспечивают доступность множества услуг большому количеству различных клиентов при сохранении (а в идеале, и повышении) уровня надежности и безопасности систем. Сложность внутренней организации обуславливает высокие расходы на эксплуатацию и сопровождение. Вышеперечисленные проблемы вынуждают в процессе развития корпоративных информационных систем ориентироваться, прежде всего, на использование децентрализованной распределенной архитектуры.

Впервые о технологии блокчейн заговорили в контексте криптовалюты биткойн, которая в настоящее время занимает первое место в рейтинге ведущих блокчейн-систем. Для биткойна блокчейн реализован в виде распределенного аналога гроссбуха (ledger), в котором регистрируются все новые транзакции с биткойном, после чего их изменение или отмена становятся невозможными. После многолетнего успешного функционирования в качестве программной основы криптовалюты биткойн, технология блокчейн заинтересовала разработчиков как возможное эффективное решение для реализации многих не связанных с виртуальными деньгами проектов. Многие крупные финансовые организации и международные корпорации, такие, как IBM, Microsoft, Goldman Sachs, Barclysm BNYMellon и другие, начали инвестировать в исследования и разработку блокчейн-технологии. В первую очередь данная технология привлекла внимание банков и других финансовых организаций, поскольку она повышает защищенность от кибератак, гарантирует неизменность данных, снижает операционные расходы. В январе 2017 г. семь крупнейших европейских банков (Deutsche Bank, HSBC, KBC, Natixis, Rabobank, Societe Generale и UniCredit) объединились в создании общей финансовой платформы для трансграничной торговли на основе блокчейн-технологии. Qiwi публично объявила о регистрации товарного знака «битрубль». К интересным финансовым проектам можно отнести «Ripple», совмещающий в себе распределенную платежную систему и валютную биржу.

Предпринимаются попытки создания на основе блокчейна систем электронного документооборота в пределах организаций или территориальных образований. Сбербанк анонсировал запуск «Digital Ecosystem» – альтернативной системы электронного документооборота на основе технологий блокчейн, ключевым отличием которой является отсутствие единого провайдера и открытость для всех участников. Использовать «Digital

Ecosystem» на первом этапе помимо крупнейшего российского банка планируют Федеральная антимонопольная служба (ФАС), «Аэрофлот», ОАК, «Русский уголь» и «ФортеИнвест». Корпорации Mahindra Group и IBM объявили о создании блокчейн-решения для финансовой логистики, позволяющего повысить безопасность, прозрачность и качество операционных процессов в цепочке поставок. Имеются опробованные блокчейн-системы, эксплуатирующие такое преимущество блокчейна, как возможность проведения операций обмена без посредников. Это всевозможные децентрализованные биржи (NXTAsset Exchange, Bitshares, OpenLedger), краудфандинговые площадки, системы p2p кредитования.

Другие направления развития применения блокчейна – это единые учетные реестры, отчетность, системы голосования и опросы, нотариальные документы, оформление товаров (например, на уровне таможенных документов), государственное финансирование и многосторонние услуги. Известно, что Национальный расчетный депозитарий РФ завершил разработку и успешно протестировал прототип системы электронного голосования для собраний владельцев облигаций «Электронное каскадное голосование» (E-proxy-voting) на основе технологии блокчейн. В конце января 2017 г. стало известно о создании нового блокчейн-консорциума, в задачи которого входит повышение безопасности и улучшение использования Интернета вещей. В организацию вошли Cisco, Bosch, Foxconn, Gemalto, банк BNY Mellon, а также блокчейн-стартапы Consensus Systems, BitSE и Chronicled. Совместными усилиями компании будут разрабатывать на основе технологии блокчейн протокол безопасного обмена информацией между всевозможными IoT-устройствами (стиральные машины, транспортированные контейнеры, датчики и т.п.)

Важный шаг в развитии технологии блокчейн был сделан проектом «Ethereum», предложившим универсальную блокчейн-платформу для создания приложений на основе смарт-контрактов – программных модулей, выполняющих алгоритм сделки с произвольными ифровыми активами. Главными преимуществами смарт-контрактов по сравнению с обычными контрактами являются безопасность, отсутствие необходимости в посредниках, нижение транзакционных издержек и, конечно же, автоматичность выполнения. На платформе «Ethereum» реализован сервис децентрализованных автономных организаций ДАО). ДАО представляют собой систему смарт-контрактов, полностью или частично обеспечивающую работу коммерческих и некоммерческих организаций. Возможности Ethereum могут быть применяются для разработки, децентрализации, и защиты решений в самых разных областях человеческой деятельности: голосования, доменные имена, финансовые биржи, краудфандинг, управление компаниями, большинство разновидностей контрактов и договоров, интеллектуальные права и даже умная собственность, в случае интеграции с аппаратными решениями. Microsoft совместно с компанией ConsenSys предоставила в Azure средства разработки для платформы «Ethereum» по модели «Ethereum Blockchain as a Sevice» (EBaaS). EBaaS позволяет клиентам быстро развернуть и протестировать свои решения на базе технологии блокчейна, освобождая их от каких-либо капитальных затрат на этапе апробации новой технологии. Готовое решение может развертываться в любых публичных облаках, в частных облаках, на виртуальных и физических машинах. Корпорация IBM прилагает усилия к развитию корпоративного применения технологии блокчейн, в том числе предлагая блокчейн в качестве услуги разработчикам. Новый набор связанных с блокчейн услуг в облаке IBM Cloud помогает разработчикам создавать и управлять блокчейн-сетями. Разработчики могут генерировать цифровые активы и сопутствующую бизнес-логику, способствующую более безопасной и конфиденциальной передаче активов между участниками тестовой сети блокчейн с защищенными правами доступа.

Опубликованные государственные и негосударственные аналитические исследования констатируют, что использование технологии блокчейн при разработке распределенных корпоративных информационных систем снизит издержки финансовых операций, одновременно повысив их защищенность от мошенничества, а также даст неоспоримые преимущества при организации контроля над всевозможными активами и обеспечении конфиденциальности данных. Однако необходимо отметить ряд основных технических

проблем, с которыми сталкиваются блокчейн-системы: плохая масштабируемость (рост объема данных в каждом узле), тенденция к централизации. проблема доверия к данным.

В данный момент в разработке находится проект, целью которого является создание платформы для построения высокоэффективных и безопасных децентрализованных распределенных информационных систем на основе технологии блокчейн. В рамках данного проекта проблему больших объемов хранения данных на всех узлах распределенной сети и, как следствия – плохой масштабируемости, предлагается решать за счет лимитирования длины блокчейна и введения промежуточного узла доверенного подтверждения хронологически отдаленных от текущего момента транзакций. Данные узлы являются гарантами достоверности предыдущих транзакций, при этом на узлах распределенной сети хранятся блокчейны только с транзакциями за лимитированный промежуток времени относительно текущего момента. Блокчейн при этом строится от фиксированного слепка цепочки, хранящегося на узлах доверенного подтверждения. Достоверность слепка подтверждается пересчетом на всех узлах сети и голосованием, после чего он фиксируется и не подлежит изменениям. Таким образом, откат и разрешение коллизий для удаленных хронологически от текущего момента транзакций становится невозможным, но при этом снижается вычислительная нагрузка при выполнении достоверности текущих транзакций.

Проблему стремления к централизации системы как следствия растущей ресурсоемкости предлагается решать путем применения процедуры подтверждения достоверности данных за счет динамического перераспределения весов узлов с использованием нейросетевых алгоритмов.

Любые данные, размещаемые в блокчейн-системах, обладают не большей достоверностью, чем исходная оффлайн-информация. То есть использование технологии блокчейн обеспечивает лишь сохранность данных, защиту их от модификации после их ввода в систему и уверенность в однозначности транзакций с ними внутри блокчейна, но не гарантию их полной достоверности. Такого же рода ограничения существуют и при работе со смарт-контрактами – они могут работать только при условии, что исполняемый код имеет прямой доступ к написанным в них объектам, то есть могут оперировать только данными, сохраненными в реализующей смарт-контракты блокчейн-платформе. Поэтому контракты, функционирующие только внутри замкнутой информационной среды конкретного блокчейна, не могут широко применяться государственными, финансовыми и бизнес-институтами. Для обеспечения максимального доверия к вводимым в систему внешним данным предлагается использовать оффлайн-протоколов на основе математически доказуемых формализмов с использованием логики предикатов.

Опыт коммерческого и общественного использования технологии блокчейн в разных странах показывает ее явные преимущества (простота, прозрачность, дешевизна, надежность, скорость) по сравнению с традиционными процедурами. Технологии блокчейн лучше всего применяются в тех сферах, где требуется хранение, постоянная обработка и обновление больших объемов защищенных данных.

Библиографический список

1. Мазонка, О. Хэш Цепочки Технологии Hasq / О. Мазонка, В. Попов. – Hasq Technology Pty Ltd, Австралия, 2014.

2. BigchainDB: A Scalable Blockchain Database / T. McConaghy, R. Marques, A. Muller, D. De Jonghe, T. Troy McConaghy, G. McMullen, R. Henderson, S. Bellemare, A. Granzotto. – ascribe GmbH, Berlin, Germany, 2016.

3. Distributed Ledger Technology: beyond block chain. A report by the UK Government Chief Scientific Adviser. Available at:. – URL: https://www.gov.uk/government/uploads/system/uploads/ attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf (accessed: 4 April 2017).

4. Unlocking economic advantage with blockchain: a guide for asset managers. Available at. – URL: http://www.oliverwyman.com/our-expertise/insights/2016/jul/unlocking-economic-advantage-with-blockchain.html (accessed: 7 April 2017).

С. В. Шибанов, А. В. Тарасеев, Е. С. Кикта

Пензенский государственный университет, Пенза, Россия

КЛАССИФИКАЦИЯ ПРАВИЛ В АКТИВНЫХ БАЗАХ ДАННЫХ

Рассматриваются активные правила как механизм реагирования на внутренние и внешние события в активных базах данных. Исследуются основные свойства активных правил, приводится классификация правил по наиболее важным критериям.

Активные базы данных (АБД) поддерживают механизмы, позволяющие им автоматически отвечать на события, которые происходят как внутри, так и снаружи системы. Для реагирования на события в АБД используются так называемые активные правила, которые содержат три основных компонента – событие, условие и действие. Событие (event) описывает то, что должно произойти, и на что правило должно быть способно реагировать. Условие (condition) проверяет контекст, при котором событие произошло. Действие (action) описывает процедуру, которая должна быть выполнена правилом, если соответствующее событие произошло и условие оказалось истинным [1].

На сегодняшний день существует большое количество различных моделей активных правил. При решении конкретной задачи, которая подразумевает применение какой-либо модели правил, перед разработчиком встает вопрос: «Какая именно концепция активных правил будет наиболее эффективна в данном случае?». Чтобы облегчить столь сложный выбор, необходимо классифицировать активные правила, предварительно определив критерии классификации.

Прежде всего, необходимо отметить, что в АБД описываются и обрабатываются два основных типа событий:

- примитивное (простое, элементарное) событие, которое порождается одиночным низкоуровневым действием;
- сложное (составное, композитное) событие, которое порождается комбинацией примитивных или сложных событий, используя ряд операторов, которые составляют алгебру событий [2].

Правила, реагирующие на примитивные события, можно классифицировать с учетом источника возникновения события на следующие виды:

- структурные, которые порождаются операцией над некоторой частью структуры (например, вставить кортеж, изменить атрибут, получить доступ к кортежу);
- поведенческие, которые порождаются выполнением некой операции, описанной пользователем;
- транзакционные, которые порождаются командами транзакций (например, abort, commit, begin-transaction);
- абстрактные, которые позволяют сигнализировать о возникновении события явно, за счет механизма программирования (например, в ответ на некоторую информацию, введенную пользователем);
- срабатывающие на исключения, которые порождаются в результате некоторого возникшего исключения (например, была сделана попытка получить доступ к данным без соответствующей авторизации);
- временные, которые возникают в некоторый момент времени. Часто их подразделяют на абсолютные (например, 13 Ноября 1998 в 15:00), относительные (например, 10 дней после того, как акции проданы) и периодические (например, в первый день каждого месяца);
- внешние, которые порождаются чем-то извне (например, измеряемая температура выше 30 градусов).

По масштабу события (Event Granularity) правила могут быть определены:

- для всего набора элементов (например, все экземпляры класса);
- для заданных поднаборов (например, все штатные сотрудники, кроме профессоров);
- для некоторых элементов набора (например, чтобы предотвратить неавторизованный доступ к некоторым экземплярам, или чтобы сделать возможным обновление тех объектов виджета, которые сейчас находятся на экране) [3].

Набор операция, с помощью которых описываются сложные события, как комбинации примитивных или других сложных событий называется алгеброй событий. В [2] предлагается ряд операторов, на основании которых можно классифицировать сложные события:

- использующие оператор «дизъюнкция» – E1 или E2 возникает, когда возникает либо E1, либо E2;
- использующие оператор «конъюнкция» – E1 и E2 возникает, когда возникают оба события E1 и E2 в любом порядке;
- использующие оператор «последовательность» – seq(El,E2) возникает, когда E1 появляется раньше E2;
- использующие оператор «закрытие» – closure E in Int возникает только один раз, когда появляется E, и блокирует последующие появления E в течение интервала Int;
- использующие оператор «история» – times(n,E) in Int возникает, когда E появляется n раз в течение временного интервала Int;
- использующие оператор «не» – not El in Int обнаруживается, если событие El не появлялось в течении интервала Int;
- использующие связки нескольких операторов [1].

Существуют два основных режима связывания правил компонентов правила «событие-условие» и «условие-действие». Первый определяет, когда условие вычисляется относительно события, инициировавшего правило. Режим связывания условие-действие показывает, когда действие должно быть выполнено относительно вычисления условия. На основании данных режимов выделяют следующие виды правил:

- имеющие непосредственный (immediate) режим связывания, в случае которого условие (действие) вычисляется (выполняется) сразу после события (условия);
- имеющие отложенный (deferred) режим связывания, в случае которого условие (действие) вычисляется (выполняется) в той же транзакции, что и событие (условие) правила, но не обязательно в первую очередь. Однако, в некоторых моделях описывается режим связывания, осуществляемый пользователем, когда условие (действие) вычисляется (выполняется) в описанное пользователем время после того, как событие (условие) просигналит (вычислится). Похожий эффект также поддержан в системе Starburst, где пользователи могут вызвать обработку правил внутри транзакции специальными командами: команды «обработать правила», «обработать список правил S» и «обработать правило R» для обработки всех правил, заданного набора или одного правила, соответственно;
- имеющие отдельный (detached) режим связывания, в случае которого условие (действие) вычисляется (выполняется) в другой транзакции, относительно события (условия). Выполнение действия может зависеть или не зависеть от подтверждения транзакции, в которой было событие или вычислялось условие [4].

Когда несколько правил вызываются в одновременно в качестве реакции на некоторое событие, возникает проблема выбора следующего правила для запуска. На основании этого выделяют:

- динамические правила, в которых назначение приоритетов правилам основано либо на давности обновления (то есть на времени появления события), либо на сложности условия;
- статичные правила, в которых приоритеты часто назначаются либо системой (т.е. на основе времени создания правила), либо пользователем как атрибут правила. Правила могут быть расположены по порядку, используя числовую схему, в которой каждому правилу дается абсолютное значение, являющееся приоритетом, или указывая относительные приоритеты правил явным определением того, что данное правило должно быть запущено перед другим, когда оба они вызываются одновременно.

При исполнении сложных правил в одно и то же время не менее важен выбор числа правил, которые нужно запустить. Поэтому их можно разделить на те, в которых:

- запуск всех экземпляров правил осуществляется последовательно;
- запуск всех экземпляров правил происходит параллельно;
- есть возможность запустить все экземпляры конкретного правила перед любыми другими;
- есть возможность запустить только один или несколько экземпляров правил [5, 6].

Таким образом, на основании данной классификации, среди множества существующих типов активных правил каждый разработчик сможет найти такую модель, которая наиболее оптимально подойдет при решении конкретной задачи.

Библиографический список

1. Paton, N. Active Database Systems / N. Paton, O. Diaz // ACM Computing Surveys. – March 1999.

2. Dayal, U. Active Database Systems / U. Dayal, E. Hanson, J. Widom // Interoperability, and Beyond. – Addison-Wesley, Reading, Massachusetts, Sep. 1994.

3. Шибанов, С. В. Графовые модели для анализа взаимодействий правил в активных базах данных / С. В. Шибанов, А. Б. Зудов // Модели, системы, сети в экономике, природе и обществе. – Пенза, 2016. – № 1 (17). – С. 344–354.

4. Шибанов, С. В. Формальное представление правил в активных базах данных как последовательных взаимодействующих процессов / С. В. Шибанов, А. Б. Зудов // Модели, системы, сети в экономике, природе и обществе. – 2016. – № 1 (17). – С. 335–343.

5. Шибанов, С. В. Формализация ECA и SECA-моделей активных правил / С. В. Шибанов, Э. В. Лысенко, А. А. Скоробогатько // Прикладная математика, управление и информатика : сб. тр. Междунар. молодеж. конф. : в 2 т. – Белгород : ИД «Белгород», 2012. – Т. 1. – С. 319–322.

6. Шибанов, С. В. Платформа для разработки приложений активных баз данных / С. В. Шибанов, А. А. Скоробогатько, А. Б. Зудов, П. В. Вишняков // Научный сервис в сети Интернет: поиск новых решений : тр. Междунар. суперкомпьютерной конф. – М. : Изд-во МГУ, 2012. – С. 591–599.

А. Р. Бородулин

Пензенский государственный университет, Пенза, Россия

ЯЗЫК COLAMO И КОМПЛЕКС СРЕДСТВ ПРОГРАММИРОВАНИЯ МНОГОКРИСТАЛЬНЫХ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Обсуждаются возможности современных инструментальных средств для проектирования аппаратных средств, ускоряющих работу вычислительных систем при реализации вычислительно трудоемких алгоритмов.

Одним из перспективных способов достижения высокой реальной производительности вычислительной системы является адаптация ее архитектуры под структуру решаемой задачи и создание вычислительного устройства, одинаково эффективно реализующего как структурные, так и процедурные фрагменты вычислений. Поэтому отечественные и зарубежные производители вычислительной техники все чаще включают в состав разрабатываемых вычислительных систем программируемые логические интегральные схемы (ПЛИС) для ускорения работы при реализации вычислительно трудоемких алгоритмов. Создаются как отдельные ускорители с одним-двумя кристаллами ПЛИС, так и целые вычислительные комплексы. Некоторые выпускают ряд ускорителей и несущих плат с не-

большим числом (до четырех) кристаллов ПЛИС, которые используются в создании серверов и гибридных кластерных систем фирмами HP и IBM.

В настоящее время существует множество различных систем проектирования, позволяющих создавать структурные решения прикладных задач на ПЛИС. Наиболее популярными как в качестве отдельного средства разработки, так и в составе комплексов, являются программы- синтезаторы, предоставляемые фирмами-изготовителями ПЛИС: ISE и Vivado фирмы Xilinx, Quartus II фирмы Altera и Actel Libero IDE фирмы Actel. Данные средства, по- мимо непосредственно среды проектирования цифровых устройств, включают в себя ряд вспомогательных утилит: анализаторы временных характеристик, редакторы размещения, модули программирования ПЛИС, системы моделирования цифровых устройств и др. Благодаря широкому инструментарию эти системы проектирования обеспечивают полный цикл разработки цифровых устройств в пределах одного кристалла ПЛИС: создание исходного описания проекта, синтез, моделирование, размещение, трассировку, конфигурирование кристалла.

В связи с постоянным ростом емкости кристаллов ПЛИС проектирование решений прикладных задач в кристаллах ПЛИС с помощью языков описания аппаратуры (VHDL, AHDL, Verilog и др.), а также разработка цифровых устройств в графических редакторах становится все более трудоемкой. Поэтому в настоящее время ведущие производители ПЛИС и реконфигурируемых вычислителей ориентируются на языки высокого уровня. Так, в новой среде разработки Vivado фирмы Xilinx добавлен новый инструмент проектирования Vivado HLS, основанный на языке высокого уровня, а компания Altera для своих ПЛИС предлагает инструментарий разработки Altera SDK (http://www.altera.com/literature/lit-opencl-sdk.jsp, дата обращения 12.12.2014) для нового стандарта параллельного программирования гетерогенных систем OpenCL. В данных решениях используются трансляторы С-подобных языков, генерирующие из программы на аналоге языка программирования высокого уровня С код на языках описания аппаратуры на уровне регистровых передач (RTL-уровень, трансляторы C-to-RTL).

Несмотря на схожесть синтаксисов данных С-подобных языков с самим языком С, подобный подход вовсе не означает, что исходный код на языке С, написанный под персональный компьютер или кластерную вычислительную систему, будет понят трансляторами C-to-RTL. Выбор языка С в качестве основы обусловлен широкой распространенностью данного языка, что существенно упрощает освоение новых инструментариев разработки решений прикладных задач для ПЛИС.

Также при использовании трансляторов C-to-RTL весь код программы либо явно указанные процедуры транслируются в RTL-описания отдельных кристаллов ПЛИС. В подобных системах разработки отсутствует инструментарий, обеспечивающий автоматические разбиения параллельной программы на множество связанных кристаллов ПЛИС.

В Vivado HLS проект разрабатывается в рамках одного кристалла ПЛИС, и если программисту недостаточно аппаратного ресурса кристалла, то он вынужден самостоятельно распределять вычисления между несколькими проектами для каждой ПЛИС и синхронизировать управляющие и информационные потоки между ними.

Стандарт OpenCL используется компанией Nallatech (разработчик реконфигурируемых вычислителей) и подразумевает использование нескольких ПЛИС в одном проекте. Программирование решений в кристаллах ПЛИС в данном случае осуществляется с помощью вызова функций из библиотек инструментария разработки Altera SDK, и в каждом задействованном кристалле ПЛИС выполняются вычисления, описанные отдельным участком кода. Таким образом, программа, написанная в стандарте OpenCL, представляет собой основной код, написанный под традиционные процессоры, и отдельные участки кода, написанные под ПЛИС, задействованные как сопроцессоры. В данном случае задача синхронизации данных также возлагается на самого программиста.

Еще одним известным средством программирования ПЛИС является комплекс, разработанный компанией Mitrionics, состоящий из виртуального процессора Mitrion Virtual Processor (MVP), программирование которого осуществляется на языке высокого уровня Mitrion-C, и библиотеки функций для построения хост-программ MithalAPI, входящих

в пакет разработки 195 Mitrion SDK. Разрабатываемая на языке Mitrion-C программа должна быть полностью реализована на одном виртуальном процессоре MVP, что не позволяет программировать многокристальные РВС, и, следовательно, существенно снижает эффективность использования программного комплекса компании Mitrionics. Для программирования многокристальных РВС со связями между ПЛИС программисту необходимо самому реализовать интерфейс (протокол) обмена данными между ПЛИС и решить вопросы, связанные с синхронизацией потоков данных. В этом случае программа для РВС вырождается в программу для кластера (множество процессоров MVP), реализованного на ПЛИС, что существенно снижает эффективность реали- зации задач на многокристальных РВС.

Альтернативный подход к программированию РВС предлагается в НИИ МВС ЮФУ, занимающемся разработкой многокристальных реконфигурируемых вычислительных систем раз-личных архитектуры и конфигурации уже более 15 лет. Опыт решения различных классов задач в НИИ МВС ЮФУ показал, что для эффективного решения современных трудоемких задач программисту необходимы средства программирова-ния, обеспечивающие:

- программирование на языке высокого уровня;
- поддержку многокристального программирования;
- обеспечение высокой частоты работы ПЛИС;
- высокую плотность заполнения кристалла ПЛИС;
- поддержку конвейерной и макро конвейерной организации вычислений.

В НИИ МВС ЮФУ разработан и широко используется программный комплекс, состоящий из:

- транслятора языка программирования COLAMO, осуществляющего трансляцию исходного кода на COLAMO в информационный граф параллельной прикладной программы;
- синтезатора масштабируемых схемотехнических решений на уровне логических ячеек ПЛИС Fire!Constructor, осуществляющего отображение полученного от транслятора языка программирования COLAMO информационного графа на архитектуру РВС, размещение отображенного решения по кристаллам ПЛИС и автоматическую синхронизацию фрагментов информационного графа в разных кристаллах ПЛИС;
- библиотеки IP-ядер, соответствующих операторам языка COLAMO (функционально-законченных структурно-реализованных аппаратных устройств) для различных предметных областей, и интерфейсов для согласования скорости обработки информации и связи в единую вычислительную структуру;
- средств отладки и программ доступа и мониторинга состояния РВС.

Язык высокого уровня COLAMO предназначен для описания реализации параллельного алгоритма и создания на основе принципов структурно-процедурной организации вычислений [1, 8, 9] в архитектуре РВС специализированной вычислительной структуры, которая предполагает последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов.

Таким образом, приложение (прикладная задача) для РВС состоит из структурной составляющей, представленной в виде аппаратно реализованных фрагментов вычислений, и процедурной составляющей, представляющей собой единую для всех структурных фрагментов управляющую программу последовательной смены вычислительных структур и организации потоков данных.

Для представления такой организации вычислений в языке используется понятие «кадр». Кадром является программно-неделимый компонент, представляющий собой совокупность операторов, которые реализуются в виде арифметико-логических команд и команд чтения/записи, выполняемых на различных функциональных устройствах, соединенных между собой в соответствии с информационной структурой алгоритма.

В языке COLAMO отсутствуют явные формы описания параллелизма, а распараллеливание достигается с помощью объявления типов доступа к переменным и индексации элементов массивов, что характерно для языков потока данных. Для обращения к данным используется два основных метода доступа: параллельный доступ (задаваемый типом

Vector) и последовательный доступ (задаваемый типом Stream). Степень параллелизма определяется по минимальному значению параметра распараллеливания. Для типа доступа Stream степень параллелизма равна 1, а для типа доступа Vector определяется наименьшим значением векторной составляющей каждого массива, участвующего в вычислениях. Для параллельного типа доступа возможна одновременная обработка всех размерностей массивов, заданных типом Vector, при этом повышается аппаратный ресурс на обработку, но снижается время обработки.

Многомерные массивы данных могут иметь множество измерений, каждое из которых может иметь последовательный или параллельный тип доступа, задаваемый ключевым словом Stream или Vector соответственно. Изменение типа доступа позволяет достаточно просто управлять как степенью распараллеливания вычислений на уровне описания структур данных, так и скоростью обработки, и занимаемым ресурсом, что позволяет программисту описывать различные виды параллелизма в достаточно сжатом виде.

Помимо типа доступа, для переменной в языке COLAMO определены также и типы ее хранения: мемориальный (Mem), регистровый (Reg) и коммутационный (Com).

Мемориальной переменной называется величина, хранящаяся в ячейке распределенной памяти и, следовательно, сохраняющая свое значение до очередного переприсваивания. Для мемориальной переменной возможно одновременное выполнение только одного процесса. Поэтому в семантике языка COLAMO для мемориальной переменной в теле кадра действуют правило однократного присваивания и правило единственной подстановки. Правило однократного присваивания указывает на то, что мемориальная переменная в кадре изменяет свое значение только один раз. Правило единственной подстановки определяет, что переменная в кадре может использоваться только для одного процесса чтения или записи.

Для описания связей между элементами информационного графа задачи в языке COLAMO предназначена коммутационная переменная. Поскольку коммутационная переменная описывает информационные связи, то она не требует никакого вычислительного аппаратного ресурса для своей реализации. Доступ к значению коммутационной переменной после выполнения кадра невозможен. Коммутационная переменная необходима транслятору для указания информационных зависимостей при построении вычислительной структуры задачи. На коммутационную переменную, как и на мемориальную переменную, действует правило однократного присваивания, но для нее не выполняется правило единственной подстановки. Использование коммутационных переменных позволяет легко разветвлять и дублировать потоки данных, но не позволяет реализовать рекурсию.

Для организации рекурсии в языке COLAMO используется регистровая переменная, которая представляет собой регистр на аппаратном уровне и используется для хранения промежуточных данных, полученных в процессе вычислений. Единственным ограничением для регистровой переменной в теле кадра является правило однократного присваивания.

Трансляция программы на языке высокого уровня COLAMO состоит в создании схемотехнической конфигурации вычислительной системы (структурной составляющей) и параллельной программы, управляющей потоками данных (потоковой и процедурной составляющих). Создание структурной составляющей заключается в построении вычислительного графа, соответствующего описанным на COLAMO информационным зависимостям между результатами вычислений.

При этом для каждой используемой в тексте программы операции подставляется специализированный вычислительный блок в зависимости от типа доступа к переменным, типа данных, их разрядности и т.д. Синтезированный информационный граф задачи передается в синтезатор Fire!Constructor для отображения на аппаратный ресурс многокристальной РВС.

Задача автоматического отображения параллельной программы на аппаратный ресурс многокристальной РВС состоит из трех подзадач: разбиения информационного графа на непересекающиеся подграфы, размещения сформированных подграфов в ПЛИС РВС и трассировки внешних связей размещенных подграфов в системе коммутаций РВС.

Результатом работы синтезатора Fire!Constructor являются файлы VHDL-описаний и файлы временных и топографических ограничений (User Constraints Files). Файлы VHDL описывают структурные реализации фрагментов параллельной программы. На основании этих файлов и библиотеки схемотехнических элементов формируются проекты для синтезатора ISE под каждую отдельную ПЛИС. Далее с помощью синтезатора ISE формируются конфигурационные файлы ПЛИС. Полученные конфигурационные файлы загружаются в РВС.

Программа на языке COLAMO разрабатывается в едином проекте и может быть транслирована на любую РВС, описание которой и соответствующие библиотеки присутствуют в комплексе программирования РВС. В отличие от других существующих систем разработки решений прикладных задач на ПЛИС пользователю не требуется в тексте программы указывать, какие фрагменты программы в каких ПЛИС будут выполняться. Синтезатор Fire!Constructor обеспечивает автоматическое разбиение вычислительной структуры программы на языке COLAMO на несколько проектов в синтезаторе Xilinx ISE, при этом обеспечивается синхронизация информационных потоков как внутри кристалла ПЛИС, так и между ними.

Библиографический список

1. Каляев, И. А. Реконфигурируемые мультиконвейерные вычислительные структуры / И. А. Каляев, И. И. Левин, Е. А. Семерников, В. И. Шмойлов ; под общ. ред. И. А. Каляева. – 2-е изд., перераб. и доп. – Ростов н/Д : Изд-во ЮНЦ РАН, 2009. – 344 с.

2. Левин, И. И. Реконфигурируемые вычислительные системы с открытой масштабируемой архитектурой / И. И. Левин // Параллельные вычисления и задачи управления : тр. V Междунар. конф. PACO'2010. – М. : Учреждение Российской академии наук Институт проблем управления им. В. А. Трапезникова РАН, 2010. – С. 83–95.

А. В. Николаев[1], Б. В. Цыпин [2], Д. А. Ярославцева[1]

[1]АО «Научно-исследовательский институт физических измерений», Пенза, Россия
[2] Пензенский государственный университет, Пенза, Россия

ИСПОЛЬЗОВАНИЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ АНАЛИЗА СТРУКТУРНЫХ СХЕМ ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫХ И УПРАВЛЯЮЩИХ СИСТЕМ

Рассмотрены особенности построения информационно-измерительных и управляющих систем (ИИС) и целесообразность применения имитационного моделирования на начальных стадиях разработки ИИС. Представлена типовая структурная схема ИИС и приведены измеряемые физические параметры. Определен состав общего и специального программного обеспечения ИИС для автоматизированного рабочего места (АРМ) оператора, сервера и станции сбора данных. По результатам анализа типовой схемы и программного обеспечения предложен алгоритм построения и испытаний имитационной модели ИИС. Разработан алгоритм достижения цели операции от события к событию в заданный срок, в режиме реального времени. Получены адекватные оценки метрологических характеристик измерительных каналов, составных частей и ИИС в целом.

Основная тенденция развития измерений в автоматизированном производстве – это переход к машинному контролю по адаптивным моделям, к применению более сложных управляющих и информационно-измерительных систем (ИИС) [1].

Современный этап развития ИИС характеризуется переходом к созданию сложных иерархических систем, в которых осуществляется децентрализованная обработка информации, а отдельные части ИИС распределены в пространстве. Наблюдается тенденция к все большему использованию стандартных аппаратных и программных средств сопряжения на всех уровнях иерархии, что дает возможность создавать ИИС непосредственно у пользователя, исходя из его специфических требований.

ИИС можно определить как совокупность функционально связанных устройств, обеспечивающих измерение, сбор, вычислительную обработку и распределение измерительной информации в системе управления предприятием [2, 3].

Отличительной чертой ИИС является наличие в системе программируемой электронно-вычислительной машины (ЭВМ), которая используется не только для обработки результатов измерений, но и для управления процессом измерения, а также для управления воздействием (если это необходимо) на объект.

На рис. 1 представлена типовая структурная схема ИИС.

Рис. 1. Типовая структурная схема ИИС

Измерители первичные и нормализующие преобразователи должны осуществлять сбор и первичную обработки следующих физических параметров:

- давление (абсолютное, избыточное, динамическое);
- температуру (от минус 100 до 350 °С);
- силу (до 500 кН);
- перемещение (от 0,5 до 1250 мм);
- ускорение (от 0,5 до 10000 м/с^2, диапазон частот от 0,5 до 10000 Гц);
- расход (до 350 м^3/с);
- ток (до 200 А).

Станция сбора данных осуществляет сбор и хранение измерительной информации, поступающей от измерителей первичных и нормализующих преобразователей, и предоставляет данные на верхний уровень – АРМ оператора, сервер, обеспечивающий хранение данных и информации. С АРМ оператора информация поступает в локальную сеть предприятия [4].

Уровень развития ИИС определяется программным обеспечением (ПО) – совокупностью программ, используемых при подготовке и решении задач, управления и проверки работоспособности ИИС (рис. 2) [5, 6].

В соответствии с функциями, выполняемыми его компонентами, ПО ИИС состоит из двух видов – общего и специального ПО.

Общее ПО включает в себя ПО управления – совокупность программ, предназначенных для управления функционированием ИИС в процессе реализации прикладных измерительных программ и разработки новых программ.

Специальное ПО обычно реализуется в виде пакетов прикладных программ. Пакет прикладных программ – совокупность программ реализации стандартных методов (математической статистики, вычислительной математики, интеллектуального анализа данных) при решении различных прикладных задач.

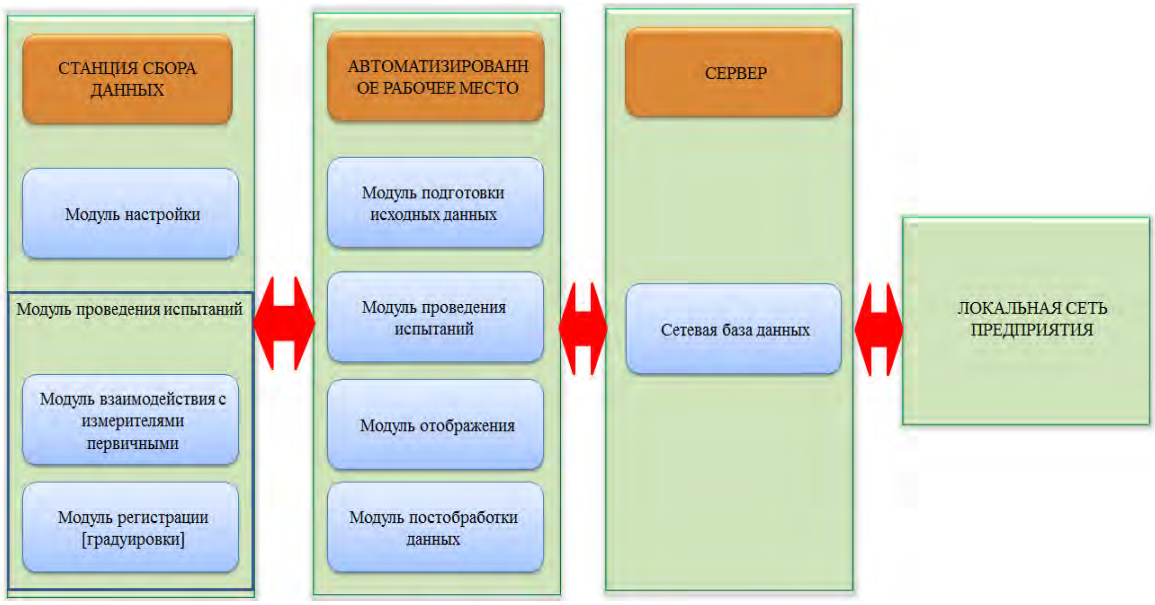


Рис. 2. Состав программного обеспечения ИИС

В настоящее время отработка технических решений на начальных стадиях разработки ИИС на натурных макетах является длительным дорогостоящим процессом. В связи с развитием вычислительной техники все чаще используются методы имитационного моделирования, которое представляет собой процесс построения и испытания моделирующего алгоритма, имитирующего поведение и взаимодействие исследуемой ИИС с учетом случайных входных воздействий и внешней среды.

Имитационное моделирование целесообразно применять в тех случаях, когда:

- нет аналитических методов решения сформулированной задачи;
- аналитические методы есть, но не удовлетворяют метрологическим требованиям точности и достоверности;
- аналитические методы есть, но их вычислительные процедуры сложны даже для ЭВМ;
- реализация известных процедур сталкивается с недостаточной математической подготовкой исследователя;
- исследователю нужно знать не только оценки искомых характеристик, но и динамику всего процесса.

Одной из основных целей имитационного моделирования является определение показателей эффективности различных операций. Показатели эффективности могут выступать в виде оценок характеристик случайных величин, процессов или вероятностей исхода операций. В первом случае – это время, расход ресурсов, расстояния и т.п., во втором случае показатель эффективности выступает в качестве вероятности, например, достижения цели операции в заданный срок, исправного состояния техники и т.д. [7].

Исходными данными для определения метрологических характеристик ИИС служат метрологические характеристики компонентов: либо нормируемые в соответствии с [8], либо полученные экспериментально. Обычно известны (например, из паспорта на средство измерения) нормируемые метрологические характеристики, которые характеризуют воздействие факторов на результат измерительных процедур.

При учете воздействующих факторов приходится довольствоваться рядом упрощений и ограничений [9, 10]:

- ИИС рассматривается как совокупность независимых измерительных каналов;
- имитационными моделями отдельных составляющих погрешности являются либо детерминированные, либо случайные величины с равномерным или нормальным законами распределения, а также кусочно-линейные функции от них;
- функции распределения компонентов линейны;

• имитационными моделями измеряемых величин являются либо детерминированные величины, либо случайные процессы, удовлетворяющие условиям стационарных и динамических процессов;

• композиция составляющих погрешности имеет в общем случае аддитивно-мультипликативный характер;

• «алгоритмические» компоненты рассматриваются автономно, не в составе измерительных каналов;

• объектом рассмотрения (при определении метрологических характеристик средств измерений, но не погрешностей результата) служат лишь процедуры прямых измерений.

Эти обстоятельства обусловливают необходимость применения подходов, позволяющих снять подобные ограничения и получить более адекватные оценки метрологических характеристик [11].

В результате имитационного моделирования были решены следующие вопросы:

• разработана типовая структурная схема ИИС;

• проведен выбор общего и специального программного обеспечения;

• разработан алгоритм построения испытаний имитационной модели ИИС;

• разработан алгоритм достижения цели операции от события к событию в заданный срок в режиме реального времени;

• получены адекватные оценки метрологических характеристик составных частей и ИИС в целом.

Библиографический список

1. Парахуда, Р. Н. Информационно-измерительные системы : Письменные лекции / Р. Н. Парахуда, Б. Я. Литвинов. – СПб. : СЗТУ, 2002. – 74 с.

2. Древс, Ю. Г. Применение ПЭВМ в системах измерения и обработки данных. Ч. 2 Технические средства сбора и обработка информации в измерительно-вычислительных комплексах : учеб. пособие / Ю. Г. Древс, А. Л. Зорин. – М. : МИФИ, 2001.

3. Цапенко, М. П. Измерительные информационные системы: Структуры и алгоритмы, системотехническое проектирование : учеб. пособие для вузов / М. П. Цапенко. – 2-е изд., перераб. и доп. – М. : Энергоатомиздат, 1995. – 439 с.

4. Горшков, П. С. Функциональная модель процесса разработки автоматизированной системы контроля состояния авиационного оборудования / П. С. Горшков, А. В. Потемкин // Труды Международного симпозума. Надежность и качество. – 2016. – Т. 1. – С. 178–183.

5. Дмитриенко, А. Г. Распределенная интеллектуальная система мониторинга состояния объектов РКТ / А. Г. Дмитриенко, А. В. Блинов, А. В. Николаев // XXI век: итоги прошлого и проблемы настоящего. – 2011. – С. 15–21.

6. Козлов, В. В. Применение методов цифрового спектрального оценивания в задаче измерения параметров сигнала / В. В. Козлов, Б. В. Цыпин, М. Г. Мясникова, С. В. Ионов // Измерительная техника. – 2010. – № 10. – С. 16–20.

7. Дмитриенко, А. Г. Обобщенная процедура структурно-параметрического синтеза информационных моделей сложных систем / А. Г. Дмитриенко, М. Ю. Михеев, Т. В. Жашкова // XXI век: итоги прошлого и проблемы настоящего плюс. – 2012. – № 4. – С. 143–152.

8. ГОСТ 8.009–84. Нормирование и использование метрологических характеристик средств измерений.

9. МИ 222–80. Методика расчета метрологических характеристик измерительных каналов информационно-измерительных систем по метрологическим характеристикам компонентов.

10. РД 153-34.0-11.201–97. Методика определения обобщенных метрологических характеристик измерительных каналов ИИС и АСУ ТП по метрологическим характеристикам агрегатных средств измерений.

11. Климентьев, К. Е. Имитационное моделирование программно-управляемого процесса измерений / К. Е. Климентьев // Измерение. Мониторинг. Управление. Контроль. – 2012. – № 1. – С. 10–14.

А. В. Николаев, А. А. Папко, А. В. Поспелов

АО «Научно-исследовательский институт физических измерений», Пенза, Россия

МЕТОДЫ ПОВЫШЕНИЯ ВИБРАЦИОННОЙ ИНТЕЛЛЕКТУАЛЬНОЙ УСТОЙЧИВОСТИ

Представлены результаты разработки интеллектуальной системы устройств измерения углов тангажа и крена (УИТК), предназначенной для горизонтирования объектов, эксплуатируемых на подвижных шасси. Рассмотрены условия возникновения эффекта наложения спектров при эксплуатации УИТК и методы его исключения.

Горизонтирование при установке объектов различного назначения на подвижные шасси осуществляется с помощью систем с аналогичным названием.

Неотъемлемой частью систем горизонтирования являются инерциальные системы, предназначенные для измерения углов крена, тангажа и формирования управляющих воздействий для опорно-поворотных устройств установки объектов в заданное положение. При этом эффективность работы систем напрямую связана с точностью измерения углов тангажа и крена.

В качестве измерителей углов в наземных системах горизонтирования, как правило, используются акселерометры уравновешивающего преобразования [5]. В отличие от традиционных датчиков угла, использование акселерометров не требует механической связи между чувствительным элементом и подвижным основанием. Акселерометры измеряют углы наклона конструкций относительно вектора гравитационного ускорения g на основе энергетического взаимодействия с ним. При измерении углов тангажа θ и крена γ два акселерометра устанавливаются на платформу измерительными осями перпендикулярно. При этом они измеряют проекции вектора гравитационного ускорения, равные $g\theta = g \cdot \sin\theta$ и $g\gamma = g \cdot \sin\gamma$. Для каждого из них выходной сигнал без учета смещения нуля $U_{\text{вых}}(\theta)$ будет равен

$$U_{\text{вых}(\theta)} = \text{К}_{g\theta} \cdot \arcsin\frac{g_\theta}{g}\ , U_{\text{вых}(\gamma)} = \text{К}_{g\gamma} \cdot \arcsin\frac{g_\gamma}{g}\ ,$$

где К$g\theta$, К$g\gamma$ – коэффициенты преобразования акселерометров каналов θ и γ по гравитационному ускорению.

Особенностями создания акселерометров для систем горизонтирования является выполнение двух противоречивых требований:

– высокой точности, реализуемой за счет увеличения глубины уравновешивания и, как следствие, увеличения собственной частоты;

– устойчивости или сохранения метрологических характеристик при воздействии виброускорений в широком частотном диапазоне, возникающих при работе двигателя шасси, опорно-двигательных и технологических устройств объекта, например вентиляторов. При этом следует иметь в виду, что для систем горизонтирования с использованием акселерометров вибрационные помехи являются влияющей величиной, однородной с измеряемым параметром.

Выполнение указанных требований достигается путем формирования частотного диапазона измерений (ЧДИ), верхняя граница которого определяется контролируемой динамикой объекта и скоростью затухания амплитудно-частотной характеристики в первой октаве за верхней границей ЧДИ до 30 дБ/окт. При этом необходимость выполнения требований осложняется также тем, что параметры влияющих вибраций, как любой реальный сигнал, имеют неограниченный спектр, а требования к их частотному диапазону, как правило, задаются весьма приближенно.

В настоящее время для формирования ЧДИ акселерометров эффективно использование ФНЧ высокого порядка на переключаемых конденсаторах (на ПК-фильтрах). Для микросхем ПК-фильтров частота среза (верхняя граница ЧДИ) и частота генерации встроенного генератора, под которой следует понимать частоту дискретизации, связаны соотношением 50:1.

Структурная схема системы с учетом включения ПК-фильтров на входах АЦП микроконтроллера показана на рис. 1.

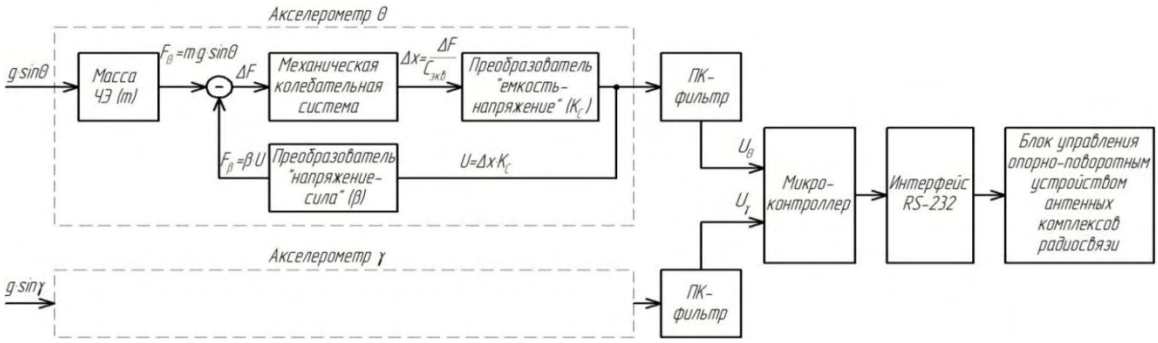


Рис. 1. Структурная схема системы горизонтирования

На рисунке обозначено:

g – модуль вектора гравитационного ускорения;

θ, γ – углы тангажа и крена соответственно;

m – масса инерционного элемента;

Δx – смещение инерционного элемента под действием измеряемой и уравновешивающей сил $F\theta$ ($F\gamma$), $F\beta$;

$C_{экв}$ – жесткость подвеса, формируемая электромеханической обратной связью.

Согласно структурной схеме проекции гравитационного ускорения на измерительные оси акселерометров каналов θ и γ преобразовываются в них в выходные напряжения, которые подаются на ПК-фильтры и входы АЦП микроконтроллера.

В настоящее время в АО «НИИФИ» разработана система горизонтирования типа УИТК. Ее отличительной особенностью является использование акселерометров собственной разработки с применением отечественной комплектации, включая ПК-фильтр, микроконтроллер и приемопередатчик, а также высокая стабильность метрологических характеристик, нормируемая в виде среднеквадратического значения абсолютной погрешности, которое в условиях эксплуатации не превышает ±20 угловых секунд (табл. 1). Для примера, эталонные средства горизонтирования типа поплавковых и электронных уровней имеют сопоставимую погрешность, но при неизменной температуре окружающей среды и отсутствии влияющих ускорений.

Таблица 1

Диапазоны измерений, град.	от ±3 до ±90
ЧДИ, Гц	0–4
Среднеквадратическое значение абсолютной погрешности в условиях эксплуатации, угл. с	±20
Время непрерывной работы, час, не более	15
Представление выходной информации	кодировка ASCII
Интерфейс	RS-232
Напряжение питания, В	23–34
Ток потребления, мА, не более	100
Температура окружающей среды, °С	–50 – +70
Виброускорение в направлениях измерительных осей, м/с^2/Гц	0,5/5–500
Виброускорение в поперечном направлении, м/с^2/Гц	5/5–500
Масса с корпусом из стали, кг	1,7
Габаритные размеры, мм	100×150×54

Несмотря на положительные результаты разработки и высокий уровень экспериментальной отработки УИТК, при ее пробной эксплуатации на подвижном шасси автомобиля «Тигр-М», системой записан спектр низкочастотного сигнала частотой до 2 Гц, который невозможно интерпретировать в качестве записей вибрации, воспроизведенной работой двигателя автомобиля. В режиме холостого хода автомобиля возможно наложение вибропроцессов от работы различных узлов двигателя, но их нижняя частотная граница, как правило, не бывает менее 300 Гц. Анализ вибрационных составляющих, возникающих при работе двигателя внутреннего сгорания [4, 6], показал, что причиной появления низкочастотной составляющей в спектре УИТК не может являться биение сигналов с незначительно отличающимися частотами, в связи с тем, что биение как линейный процесс не может быть источником новых частотных компонент.

Дальнейшие исследования показали, что появление в спектре выходного сигнала УИТК низкочастотной составляющей объясняется эффектом наложения спектров вибрационного сигнала и частоты дискретизации примененного ПК-фильтра. Указанный эффект характеризуется следующими признаками [1–3]:

– нарушением условия нахождения анализируемого сигнала (в рассматриваемом случае вибрационного) в зоне Найквиста или нарушением критерия дискретизации Котельникова, согласно которому частота дискретизации должна быть как минимум вдвое больше частоты исследуемого или влияющего сигнала;

– формированием в ПК-фильтре паразитного сигнала или низкочастотного образа реального сигнала, отображаемого в частотном диапазоне измерений системы и воспринимаемого АЦП и микроконтроллером в качестве измеряемого угла наклона (рис. 2).

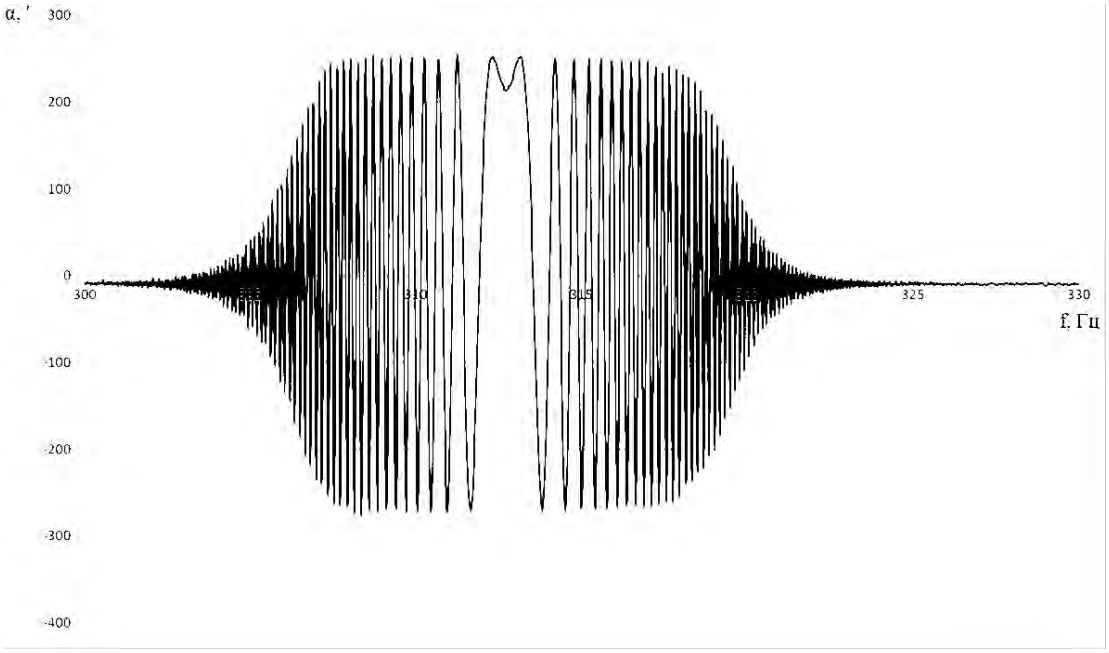


Рис. 2. Запись выходного сигнала УИТК при наложении спектров
с частотой воздействия вибрации, близкой к частоте дискретизации

В этой связи фактические измерения низкочастотной составляющей при установке УИТК на автомобиль «Тигр-М» объясняются отображением частот выше частоты дискретизации ПК-фильтра в полосу частот полезного сигнала, находящуюся в диапазоне частот от 0 до верхней границы ЧДИ.

Из анализа признаков наложения спектров, а также учитывая, что эффект наложения в цифровом сигнале полностью производится на первом этапе цифро-аналогового преобразования, а в данном случае – в ПК-фильтре, и то, что для любого реального сигнала спектр является бесконечным, следует, что для его исключения при работе УИТК необходимо принятие специальных мер.

К их числу относятся:

– увеличение частоты дискретизации ПК-фильтра, что связано с нежелательным расширением верхней границы ЧДИ;

– использование более предпочтительного решения в виде ограничения спектра измеряемого сигнала путем дополнительной аналоговой фильтрации выходного сигнала акселерометров.

Реализация этого решения и его экспериментальное подтверждение позволили решить достаточно сложную проблему наложения спектра включением пассивного интегрирующего звена первого порядка на вход ПК-фильтра с частотой среза, превышающей верхнюю границу ЧДИ в (5−10) раз.

Таким образом, для правильной интерпретации результатов измерений систем горизонтирования на основе акселерометров в процессе эксплуатации необходимо не только фильтровать анализируемый сигнал до начала дискретизации, но и обращать особое внимание на достоверность представления частотного диапазона вибрационных помех, возникающих при работе шасси, опорно-поворотных и технологических устройств объектов горизонтирования.

Кроме этого представляется целесообразным нормирование частоты наложения спектров в КД системы на основе экспериментального исследования при воздействии влияющих вибраций в интервале частот, верхняя граница которого превышает на порядок максимальную частоту вибраций, установленную в техническом задании.

Библиографический список

1. Автомобили семейства «Тигр». Виброакустический метод диагностики подшипников коленвала двигателей внутреннего сгорания. Вибрация двигателя на холостых // Техническая документация.

2. ГОСТ 32108−2013 Вибрация. Измерения вибрации, передаваемой машиной через упругие изоля-торы. Двигатели внутреннего сгорания поршневые высокоскоростные и среднескоростные. – URL: docs.cntd.ru

3. ГОСТ Р ИСО 13373-2-209 Контроль состояния и диагностика машин. Вибрационный контроль состояния машин. Ч. 2. Обработка, анализ и представление результатов измерений вибрации. – URL: docs.cntd.ru

4. Куприянов, М. С. Цифровая обработка сигналов / М. С. Куприянов, Б. Д. Матюшкин. – 2-е изд. – СПб. : Политехника, 2000. – 592 с.

5. Мокров, Е. А. Статико-динамические акселерометры для ракетно-космической техники / Е. А. Мокров, А. А. Папко. – Пенза : ПАИИ, 2004. – 164 с.

6. Лайонс, Р. Цифровая обработка сигналов / Ричард Лайонс. – 2-е изд. – М. : Бином-Пресс, 2006.

В. Н. Тарасов, Н. Ф. Бахарева, Л. В. Липилина

Поволжский государственный университет телекоммуникаций
и информатики, Самара, Россия

МОДЕЛИРОВАНИЕ ТЕЛЕТРАФИКА В СЛУЧАЕ ШИРОКОГО ДИАПАЗОНА ИЗМЕНЕНИЯ ЕГО ПАРАМЕТРОВ НА ОСНОВЕ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ

Представлено исследование систем массового обслуживания (СМО) G/G/1 с произвольными входными распределениями с коэффициентами вариаций интервала между поступлениями требований и времени обслуживания меньшими и большими 1, т.е. перекрывающими весь интервал (0, ∞). Для этого рассмотрены системы M/M/1 с запаздыванием во времени и $H_2/H_2/1$ с гиперэкспоненциальными входными распределениями.

Введение. Статья посвящена исследованию систем массового обслуживания (СМО) G/G/1 с произвольными входными распределениями с коэффициентами вариаций интервала между поступлениями требований и времени обслуживания меньшими и большими 1, т.е. перекрывающими весь интервал (0, ∞). Для этого рассмотрены системы M/M/1 с запаздыванием во времени [1] и $H_2/H_2/1$ с гиперэкспоненциальными входными распределениями [2–4] с целью получения решения для среднего времени ожидания требований в очереди. Для этого использован метод спектрального разложения решения интегрального уравнения Линдли. Для практического применения полученных результатов использован метод моментов. В работе [1] показано, что коэффициент вариации случайной величины, распределенной по экспоненциальному закону с запаздыванием во времени, меньше 1.

Известно, что распределенная по гиперэкспоненциальному закону H_2 случайная величина имеет коэффициент вариации *c* больше 1. Учитывая тот факт, что распределение H_2 является трехпараметрическим, в статье приведен механизм аппроксимации произвольных законов распределений гиперэкспоненциальным как на уровне двух первых моментов, так и на уровне трех первых моментов.

Решение для системы M/M/1 с запаздыванием (случай $c_\lambda < 1, c_\mu < 1$)

Рассмотрим систему массового обслуживания (СМО), на вход которой поступают требования, случайные интервалы между которыми распределены по закону [1]:

$$a(t) = \begin{cases} \lambda e^{-\lambda(t-t_0)}, & t \ge t_0 \\ 0, & 0 \le t < t_0 \end{cases}. \qquad (1)$$

Аналогично распределено и время обслуживания:

$$b(t) = \begin{cases} \mu e^{-\mu(t-t_0)}, & t \ge t_0 \\ 0, & 0 \le t < t_0 \end{cases}. \qquad (2)$$

Функции плотности (1) и (2) являются сдвинутыми вправо от нулевой точки на величину t_0 экспоненциальными распределениями с двумя параметрами (λ, t_0) и (μ, t_0), причем $\lambda < \mu$. Таким образом, имеем СМО с запаздыванием во времени на величину [1].

В работе [1] приведены преобразование Лапласа функции плотности времени ожидания, полученное методом спектрального разложения решения интегрального уравнения Линдли: $W^*(s) = s\Phi_+(s) = \dfrac{(1-\lambda/\mu)(\mu+s)}{(s+\mu-\lambda)}$, и полученное через него среднее время ожидания

$$\bar{W} = \frac{\lambda/\mu}{\mu-\lambda}. \tag{3}$$

Для определения неизвестных параметров (3) получена система уравнений:

$$\begin{cases} \lambda^{-1} + t_0 = \bar{\tau}_\lambda \\ (1+\lambda t_0)^{-1} = c_\lambda \\ \mu^{-1} + t_0 = \bar{\tau}_\mu \\ (1+\mu t_0)^{-1} = c_\mu \end{cases}, \tag{4}$$

где числовые характеристики в правых частях системы: средние значения интервалов $\bar{\tau}_\lambda$ и $\bar{\tau}_\mu$, коэффициенты вариации c_λ и c_μ меньшие единицы при $t_0, \lambda, \mu > 0$, будут входными параметрами для определения неизвестных параметров. В связи с этим мы имеем немарковскую модель массового обслуживания. Среднее время ожидания требования в очереди в такой системе будет меньше, чем в системе M/M/1 при одинаковом коэффициенте загрузки. Кроме того, использование функций плотности (1) и (2) позволяет аппроксимировать исходные входные распределения на уровне двух первых моментов, в отличие от классической системы M/M/1.

В качестве примера рассмотрим случай высокой нагрузки при параметрах: $\bar{\tau}_\mu = 1$, $\bar{\tau}_\lambda = 10/9$ и $c_\mu = 0.5$. В этом случае получим загрузку $\rho = 0{,}9$, значение $c_\lambda = 0{,}55$, а из (4) значения $\lambda = 18/11$ и $\mu = 2$.Тогда среднее время ожидания по формуле (3) $\bar{W} = \dfrac{18/11/2}{2-18/11} = \dfrac{18\cdot 11}{4\cdot 22} = \dfrac{9}{4}$.

Система M/M/1 при той же нагрузке дает $\bar{W} = \dfrac{0.9/1}{1-0.9} = 9$, т.е. в 4 раза большую задержку. Таким образом, рассмотренная система массового обслуживания с запаздыванием позволяет рассчитать ее характеристики при коэффициентах вариации интервалов между поступлениями требований c_λ и времени обслуживания c_μ, меньших 1 при некоторых ограничениях на входные параметры системы.

Решение для системы $H_2/H_2/1$ (случай $c_\lambda \geq 1, c_\mu \geq 1$).

В работах [2, 4] авторами найдено преобразование Лапласа-Стилтьеса для функции плотности времени ожидания для системы $H_2/H_2/1$:

$$W^*(s) = \frac{s_1 s_2 (s+\mu_1)(s+\mu_2)}{\mu_1\mu_2 (s+s_1)(s+s_2)}, \tag{5}$$

где $-s_1$ и $-s_2$ отрицательные вещественные части корней кубического уравнения

$$s^3 - c_2 s^2 - c_1 s - c_0 = 0. \tag{6}$$

Коэффициенты этого уравнения: $c_0 = a_0 b_1 - a_1 b_0 - a_0(\mu_1+\mu_2) + b_0(\lambda_1+\lambda_2)$, $c_1 = -a_1 b_1 - a_0 - b_0 + (\lambda_1+\lambda_2)(\mu_1+\mu_2)$, $c_2 = \lambda_1+\lambda_2-\mu_1-\mu_2$, а параметры: $a_0 = \lambda_1\lambda_2$, $a_1 = p\lambda_1 + (1-p)\lambda_2$, $b_0 = \mu_1\mu_2$, $b_1 = q\mu_1 + (1-q)\mu_2$. Величины $p, \lambda_1, \lambda_2, q, \mu_1, \mu_2$ являются параметрами гиперэкспоненциальных распределений с функциями плотностей

$$a(t) = p\lambda_1 e^{-\lambda_1 t} + (1-p)\lambda_2 e^{-\lambda_2 t}, \quad (7)$$

$$b(t) = q\mu_1 e^{-\mu_1 t} + (1-q)\mu_2 e^{-\mu_2 t}. \quad (8)$$

соответственно для системы $H_2/H_2/1$.

Среднее время ожидания в очереди равно значению производной от функции преобразования Лапласа (5) со знаком минус в точке s=0. Окончательно, среднее время ожидания в очереди для СМО $H_2/H_2/1$:

$$\overline{W} = \frac{1}{s_1} + \frac{1}{s_2} - \frac{1}{\mu_1} - \frac{1}{\mu_2}. \quad (9)$$

Выражение (5) на основе свойств преобразования Лапласа, позволяет также определить моменты высших порядков для времени ожидания. Например, начальный момент 2-го порядка времени ожидания равен значению второй производной от преобразования (5) в точке $s = 0$ и он будет иметь вид:

$$\overline{W^2} = \frac{d^2 W^*(s)}{ds^2} = \frac{2}{\mu_1\mu_2} - \frac{2}{s_1 s_2} - \frac{2(s_1+s_2)}{\mu_1\mu_2 s_1^2 s_2^2} \times [s_1 s_2(\mu_1+\mu_2) - \mu_1\mu_2(s_1+s_2)]. \quad (10)$$

В свою очередь, начальный момент 2-го порядка позволяет определить дисперсию времени ожидания. Учитывая определение джиттера в телекоммуникациях как разброс времени ожидания от его среднего значения, тем самым получим возможность определения джиттера через дисперсию. Это является важным результатом для анализа трафика, чувствительного к задержкам.

Для практического применения результатов (9) и (10) необходимо определить входящие в них параметры. Определение этих неизвестных параметров рассматривается ниже.

Аппроксимация законов распределений на уровне двух первых моментов

Воспользуемся свойством преобразования Лапласа воспроизведения моментов и запишем начальные моменты до второго порядка для распределения (7):

$$\overline{\tau}_\lambda = \frac{p}{\lambda_1} + \frac{(1-p)}{\lambda_2}, \quad (11)$$

$$\overline{\tau_\lambda^2} = \frac{2p}{\lambda_1^2} + \frac{2(1-p)}{\lambda_2^2}. \quad (12)$$

Аналогично, выписывая начальные моменты до 3-го порядка включительно, можно выполнить аппроксимацию гиперэкспоненциального закона распределения на уровне трех моментов, что очевидно, является более сложной задачей. Рассматривая равенства (11) и (12) как уравнения метода моментов, найдем неизвестные параметры распределения (7) λ_1, λ_2, p. Для этого запишем связующее условие в виде выражения для квадрата коэффициента вариации

$$c^2 = \frac{\overline{\tau_\lambda^2} - (\overline{\tau}_\lambda)^2}{(\overline{\tau}_\lambda)^2}. \quad (13)$$

Заметим, что система уравнений (11) и (12) с тремя неизвестными является недоопределенной. Исходя из вида уравнения (11) положим

$$\lambda_1 = 2p/\overline{\tau}_\lambda, \quad \lambda_2 = 2(1-p)/\overline{\tau}_\lambda \quad (14)$$

и потребуем выполнения условия (13). Подставив выражения (11), (12) и (14) в (13) и решив квадратное уравнение относительно параметра p, получим для него два значения:

$p = \frac{1}{2}(1 \pm \sqrt{\frac{c_\lambda^2 - 1}{c_\lambda^2 + 1}})$. При этом можно воспользоваться любым из них. Аналогично поступим с распределением (8). Таким образом, гиперэкспоненциальный закон расп-ределения может определяться полностью двумя первыми моментами и перекрывать весь диапазон изменения коэффициента вариации от 1 до ∞.

Рассмотрим пример. Пусть коэффициент загрузки СМО $\rho = \bar{\tau}_\mu / \bar{\tau}_\lambda = 0{,}9$, где $\bar{\tau}_\lambda$ и $\bar{\tau}_\mu$ средние значения интервалов между поступлениями и времени обслуживания. Рассмотрим случай нормированного обслуживания $\bar{\tau}_\mu = \mu^{-1} = 1$. Тогда средний интервал между поступлениями $\bar{\tau}_\lambda = 10/9.$ Пусть коэффициенты вариаций случайных величин – интервалов между поступлениями и времени обслуживания $c_\lambda = c_\mu = 2$. Аппроксимация на уровне двух первых моментов дает: $p \approx 0{,}887$, $\lambda_1 \approx 1{,}597$, $\lambda_2 \approx 0{,}203$, $q \approx 0{,}887$, $\mu_1 \approx 1{,}775$, $\mu_2 \approx 0{,}225$. Таким образом, неизвестные параметры распределений (7) и (8) однозначно определены. Теперь воспользуемся результатом для системы $H_2/H_2/1$ (9), приведенным выше. Коэффициенты кубического уравнения (6) в этом примере равны: $c_0 \approx 0{,}014$; $c_1 \approx 0{,}572$; $c_2 = -0{,}20$. Найдем корни кубического уравнения (8) с помощью пакета Mathcad: $-s_1 \approx -0{,}852$, $-s_2 \approx -0{,}025$, $s_3 \approx 0{,}677$. Тогда среднее время ожидания равно $\overline{W} \approx 36{,}20$.

Заключение

Полученные результаты приводят к следующим выводам.

1. В том случае, когда коэффициенты вариации интервалов поступления и времени обслуживания меньше 1, можно использовать систему M/M/1 с запаздыванием.

2. В случае, если коэффициенты вариации интервалов поступления и времени обслуживания больше 1, можно использовать систему $H_2/H_2/1$.

3. Рабочие формулы для расчета задержки в обоих случаях, приведены выше.

Библиографический список

1. Тарасов, В. Н. Анализ и расчет системы массового обслуживания с запаздыванием / В. Н. Тарасов, Н. Ф. Бахарева, И. А. Блатов // Автоматика и телемеханика. – 2015. – № 11. – С. 51–59.

2. Тарасов, В. Н. Исследование систем массового обслуживания с гиперэкспоненциальными входными распределениями / В. Н. Тарасов // Проблемы передачи информации. – 2016. – № 1. – С. 16–26.

3. Тарасов, В. Н. Математическая модель трафика с тяжелохвостным распределением на основе системы массового обслуживания H_2/M/1 / В. Н. Тарасов, Н. Ф. Бахарева, Г. А. Горелов // Инфокоммуникационные технологии. – 2014. – № 3. – С. 36–41.

4. Тарасов, В. Н. Определение среднего времени ожидания требований в управляемой системе массового обслуживания $H_2/H_2/1$ / В. Н. Тарасов, И. В. Карташевский // Системы управления и информационные технологии. – 2014. – Т. 57, № 3. – С. 92–96.

5. Тарасов, В. Н. Анализ входящего трафика на уровне трех моментов распределений временных интервалов / В. Н. Тарасов, Н. Ф. Бахарева, Г. А. Горелов, С. В. Малахов // Информационные технологии. – 2014. – № 9. – С. 54–59.

Mustafa Sadeq Jaafar, Sergey A. Zinkin, Dmitry V. Pashchenko

Penza State University, Penza, Russia

RECONFIGURABLE NETWORK MODELS FOR DISTRIBUTED COMPUTING SYSTEMS

The article is devoted to the development of customizable software for distributed computing systems. In order to develop such intermediate-level (middleware) software, it is suggested to use modifiable, or changeable, models based on Petri nets. The basis for constructing reconfigurable Petri nets (RPN – reconfigurable, or changeable, Petri nets) is the ordinary, lively and safe (1-bounded) Petri nets, which have no configuration restrictions and the multiplicity of arcs in which does not exceed one. The proposed approach is illustrated by an example of a simple protocol for peer-to-peer computing systems, where the roles of clients and servers can change during the process of the distributed system work.

1. Introduction

The size and complexity of the hardware and software of computing systems are noticeably increasing. The design of computer networks, distributed computing systems, including cloud and grid systems, wireless mobile networks and other such objects, forces developers to look for new approaches to the design of continuously modifiable hardware and software.

The newest technologies for the design of distributed systems are based, essentially, on the concept of "network is a computer", promoted by Sun Microsystems and later used by Oracle Corporation. This concept was further developed in the form of a paradigm for cloud applications that have the properties of “multilease” and “live” database migration. A particular interest in this concern is the development of a middleware for large reconfigurable clustered server systems as part of supporting the “big data” concept. Therefore, there was considerable interest in designing a reconfigurable and parametrically tuned functional architecture of distributed computing systems. In some cases, the implementation of this approach can increase the efficiency and reduce the cost of large software and hardware systems.

2. Reconfigurable and binary controlled Petri nets and some of their interpretations

The basis for constructing reconfigurable Petri nets (RPN – reconfigurable, or changeable, Petri nets) is the ordinary, lively and safe (1-bounded) Petri nets, which have no configuration restrictions and the multiplicity of arcs in which does not exceed one.

The basic information about liveliness, safety and good formation of the Petri nets are consistent with those adopted in the basic research papers [1, 2]. The interpretation of the RPN in the future will be based for the production and logical-algebraic representations.

In [2] it was stated that Petri nets are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. The fundamental properties of Petri nets were also considered in paper [3]. Problems and exercises related to reachability graph, boundedness, coverability graph, liveness, reversibility, persistence and fairness, conservativeness, P-invariants and T-invariants, concurrency and conflict, analysis of Petri net properties, structural properties of Petri nets, are given in article [4].

Article [5] deals with the supervisory control of concurrent systems based on a Petri net structural approach. This article addresses the formal development of tools that design supervisors automatically, given a Petri net model and a specification. The article also addresses the problems of deadlock prevention and liveness properties of Petri nets.

In paper [6], the authors review the concepts of modification in Petri nets with respect to their capabilities of modeling self-organizing, mobile communication networks and further they stated that only a few approaches, however, reflect the ability to adopt their structure to their current dynamics; this property is becoming more and more important regarding modern communication networks. Self-modifying nets (SMnets) in [7] basically are place-transition nets (PTNs) extended by marking-dependent arc-cardinality. Colored Petri nets with changeable structure were considered in [8]. The authors of this work considered Temporized Colored Petri nets with changeable structure (TCPN-CS). A TCPNCS consists of a sequence of TCPNs along with a mechanism to alter structure according to this sequence.

Controlled Petri nets (CtlPNs) considered in [9] are a class of Petri nets with external enabling conditions called control places which allow an external controller to influence the progression of tokens in the net. CtlPNs were first introduced by Krogh [10] and Ichikawa and Hiraishi [11].

The authors of article [12] proposed the similar concept of the modified binary Petri net. The idea is to follow the traditional Petri net firing mechanisms, but with a restriction that no places can be marked with more than one token. An algorithm was designed to capture transition firings based on the state equation and the priority assignment. But in the considered networks, information and inhibitory arcs are not used.

Some varieties of binary Petri nets are also considered in [13]. Article [14] confirms that modeling the adaption of a system to a changing environment gets more and more important. Application areas cover e.g. computer supported cooperative work, multi agent systems, dynamic process mining or mobile networks. The main idea of Petri net transformations in [14] is to extend the classical theory of Petri nets by a rule-based technique that allows to model the changes of the Petri net structure.

As an illustrative example, a simple Petri net (figures 1, 2), which describes well-known interactions of the nodes in the computer network at the applied and transport levels, is next chosen. Such an illustrative example was considered earlier in a number of researches, for example, in [2, 14, 15, 16].

However, in this paper, this example was used to construct a special type of Petri net – RPN, whose graphical representation contains duplicate positions. The work of such a Petri net is typical for peer-to-peer systems.

3. Reconfigurable network models for distributed computing systems

The construction of the proposed kind of Petri nets is also based on a structural and reconfigurable approach, differing, however, taking into account the structural links and supervisory control. Here figure 1 and figure 2 show the same RPN network that simulates client-server interactions in the computer network. Especially note that both figures represent the same network. In each part of this network, the same positions can be located. A feature of this RPN network is that it allows one of two implementations (defined and used in the present work, the RPN networks in principle admit a set of implementations that can be established dynamically, during the operation of the network, for example, after the operation of certain transitions).

In the implementation of the model shown in figure 1, the RPN is configured so that the initiator of message exchanges is a client application described by the positions a_1, a_2, a_3, a_4, a_5 and transitions t_1, t_2, t_3, t_4, t_5. The server application is represented in the model by the positions b_1, b_2, b_3, b_4, b_5 and the transitions u_1, u_2, u_3, u_4, u_5.

Intermediate positions represent message-sending and message-receiving buffers mes_1 (from client to server) and mes_2 (from server to client). The remaining intermediate positions act as buffers for ack_1 acknowledgment messages (from the server to the client) and ack_2 (from the client to the server).

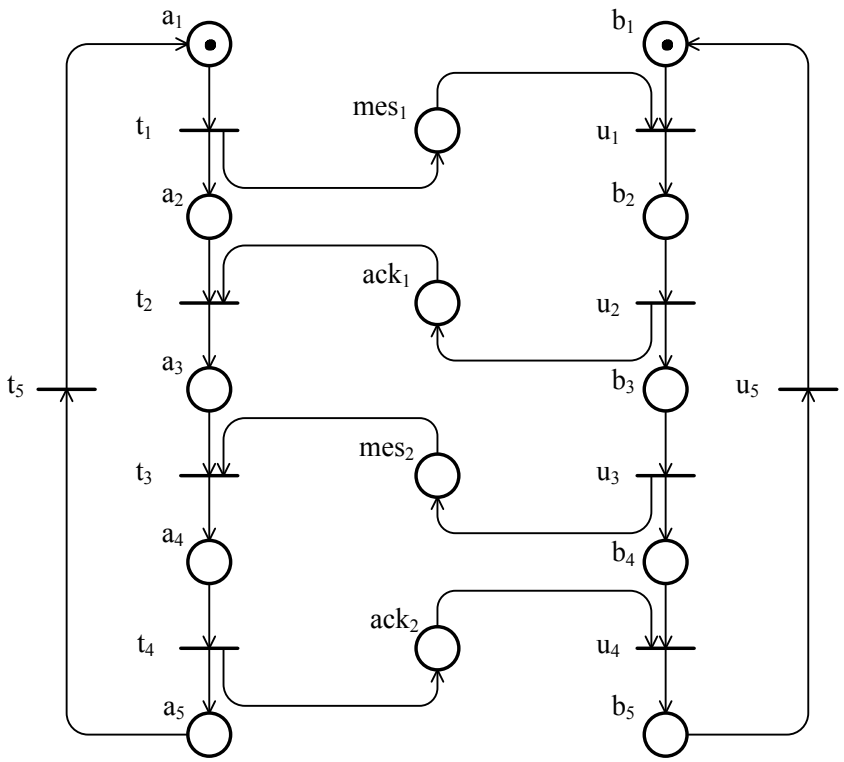


Fig. 1. Reconfigurable Petri net (RPN: part 1): implementation for the case where the initiator of the dialog is the client application

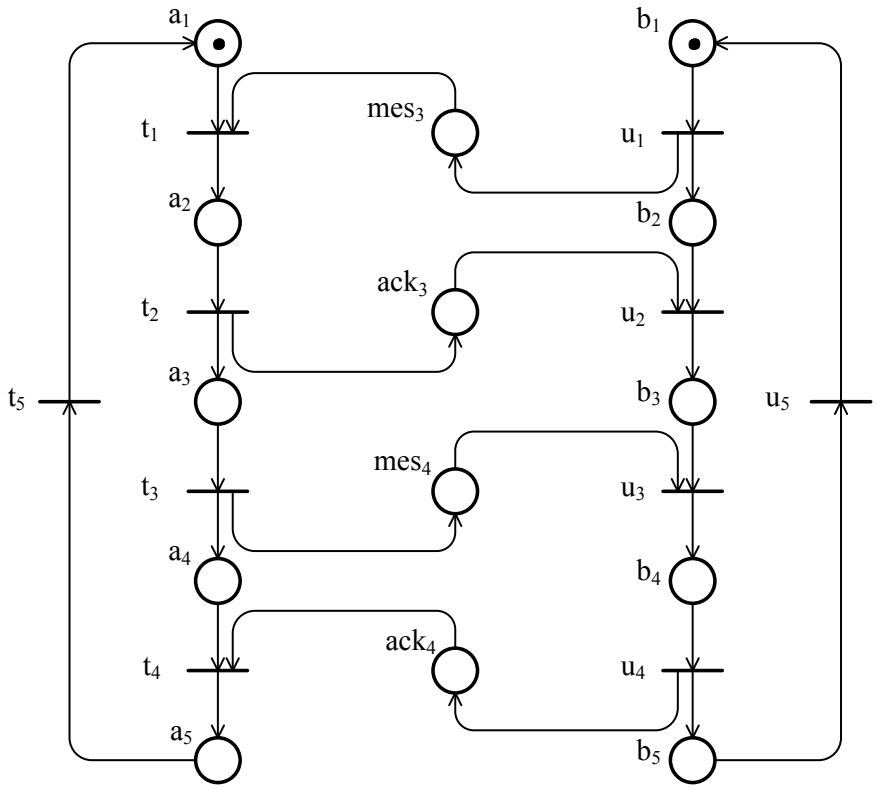


Fig. 2. Reconfigurable Petri net (RPN: part 2): implementation for the case where the initiator of the dialog is the application server

Suppose that after changing the operation mode of the client-server system, the initiator of the messaging should be the server. Such changes in the operation modes of clients and servers are typical, for example, for peer-to-peer subsystems in computer networks. The work of the system in this case is determined by the implementation of the RPN model shown in figure 2. This work assumes that the structure of both the living and safe RPN under consideration without multiple arcs is determined by two predicates $F(p, t)$ and $H(t, p)$ known in Petri net theory:

$$F: P \times T \to \{\textbf{true}, \textbf{false}\};$$
$$H: T \times P \to \{\textbf{true}, \textbf{false}\}.$$

For example, the truth of the statement $F(p_i, t_i)$ means that there is an arc that joins the position p_i with the transition t_i , and the truth of the statement $H(t_m, p_n)$ means that there is an arc connecting the transition t_m to the position p_n.

Depending on the given predicates, the RPN can be configured to one of two implementations, corresponding to the selected messaging mode in the client-server system.

The liveliness and safety of RPN can be established by known methods using some tools for analyzing Petri nets.

Now the task of designing a parametrically customizable application in a client-server architecture, in which the client and the server can change roles. One approach to solve this problem in which two pairs of independent client-server applications are created. One pair of applications "client" and "server" can correspond to the Petri net in figure 1, and the second, independent from the first one, can correspond to the Petri net in figure 2.

In this paper, a different approach based on the reconfiguration of one Petri net will be used. Noticing that in figure 1 and figure 2 which are showing two implementations of the RPN network, in which only the positions mes_1, ack_1, mes_2, ack_2, or positions mes_3, ack_3, mes_4, ack_4 are used. This means that the client and the server can change roles in the computer network. This approach is beneficial in the case when the majority of the particular Petri net during the reconfiguration remains unchanged.

4. Deadlock avoidance for Petri net models

Matlab based analysis of Petri nets designed in Pmediteur in [16] is done by the procedure of deadlock avoidance for a Petri Net model like in figures 1, 2 of two communicating computers in deadlock situation. Mathematical methods for calculating invariants in Petri nets and the graph of reachable markings are used for analyzing the properties of bounded net. Functions of Matlab Toolbox 1.0 can be downloaded from web site http://dce.felk.cvut.cz/cak/research/PN/Matlab_Toolbox.

Figures 3, 4 are showing Petri Net models (similar to the pictures in [16]) of two communicating computers in a deadlock situation; the problem is that both computers are ready to receive but none is ready to send. For the Petri net in figure. 3 the deadlock situation occurs when the statements $P(a_2)$ and $P(b_1)$ are true. Similarly, for the Petri net in figure. 3 the deadlock situation occurs when the statements $P(a_3)$ and $P(b_4)$ are true. Here P: $Pos \to \{\textbf{true}, \textbf{false}\}$ is an unary marking function (unary predicate) defined for a 1-bounded fragment of the Petri net, Pos is a finite set of places.

Through the growing popularity of using formal tools based on Petri nets similar to [16], when creating prototypes of distributed computing systems on the hardware and software platforms of both the local and global networks, the implementation of such tools has found wide distribution, as was indicated by an overview of known tools from the site http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick [17].

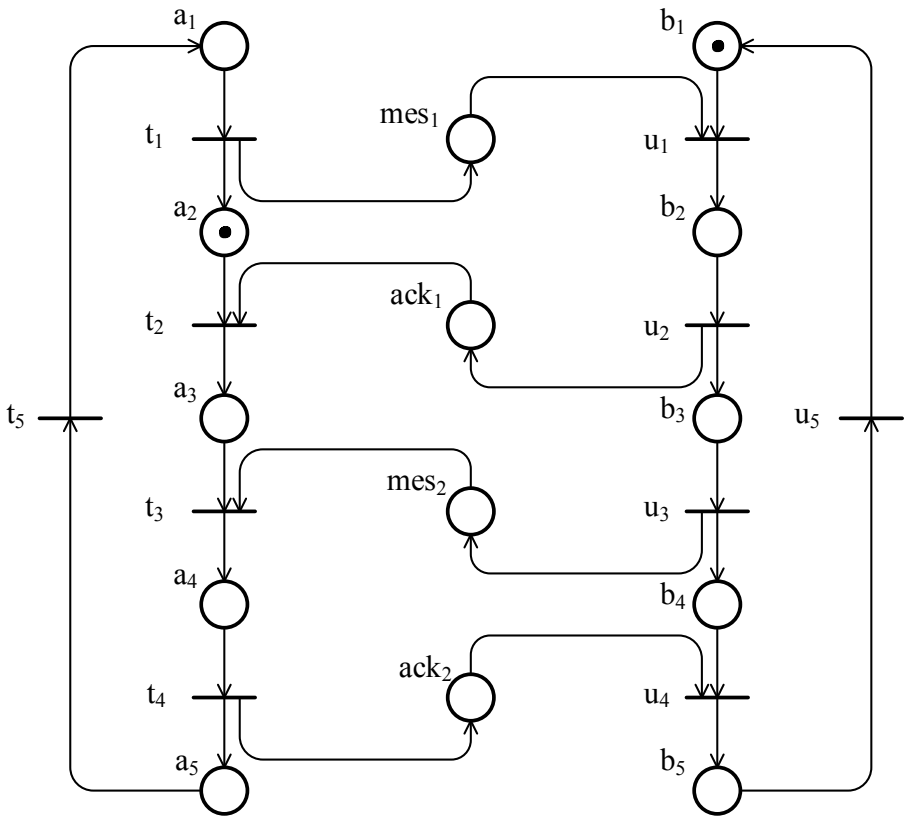


Fig. 3. Deadlock situation for two communicating computers (case 1)

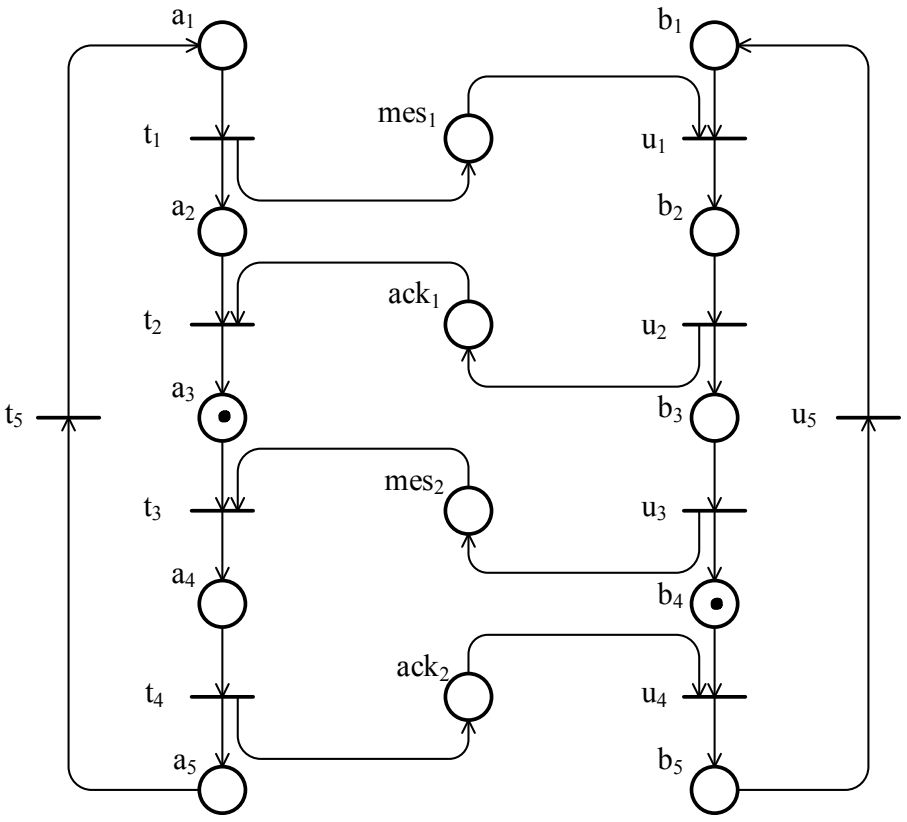


Fig. 4. Deadlock situation for two communicating computers (case 2)

5. Parametric adjustment of the production rules

Synthesis of distributed applications is proposed to be carried out on the basis of formalized specifications, which are used as logical production rules for firing transitions. In contrast to the known product rules, in new rules the statements of the form $F(p_i, t_i)$ and $H(t_m, p_n)$ reflecting the structure of the Petri net will be used. It becomes possible, when changing the truth of such statements, to translate the Petri net from one mode of functioning to another. At the same time, considerable flexibility of the transformations is provided, they do not affect the production rules, and the choice of a rule is determined by the changeable structural links of the transitions and positions in the particular Petri net. For the first mode of operation (it corresponds to the implementation of the RPN network in figure 1), the "parametrically tunable" R_1 transition rules, or the rules of logical inference, are presented in the following form:

System of parametrically tunable product rules R_1:

t_1: $mt_1 = F(a_1, t_1)\&H(t_1, a_2)\&H(t_1, mes_1)\&P(a_1) \to \neg P(a_1)\&P(a_2)\&P(mes_1)$;

u_1: $mu_1 = F(b_1, u_1)\&F(mes_1, u_1)\&H(u_1, b_2)\&P(mes_1)\&P(b_1) \to \neg P(mes_1)\& \neg P(b_1)\&P(b_2)$;

t_2: $mt_2 = F(a_2, t_2)\&F(ack_1, t_2)\&H(t_2, a_3)\&P(a_2)\&P(ack_1) \to \neg P(a_2)\&\neg P(ack_1)\&P(a_3)$;

u_2: $mu_2 = F(b_2, u_2)\&H(u_2, ack_1)\&H(u_2, b_3)\&P(b_2) \to \neg P(b_2)\&P(ack_1)\&P(b_3)$;

t_3: $mt_3 = F(a_3, t_3)\&F(mes_2, t_3)\&H(t_3, a_4)\&P(a_3)\&P(mes_2) \to \neg P(a_3)\&\neg P(mes_2)\&P(a_4)$;

u_3: $mu_3 = F(b_3, u_3)\&H(u_3, mes_2)\&H(u_3, b_4)\&P(b_3) \to \neg P(b_3)\&P(mes_2)\&P(b_4)$;

t_4: $mt_4 = F(a_4, t_4)\&H(t_4, a_5)\&H(t_4, ack_2)\&P(a_4) \to \neg P(a_4)\&P(a_5)\&P(ack_2)$;

u_4: $mu_4 = F(ack_2, u_4)\&F(b_4, u_4)\&H(u_4, b_5)\&P(b_4)\&P(ack_2) \to \neg P(b_4)\&\neg P(ack_2)\&P(b_5)$;

t_5: $mt_5 = F(a_5, t_5)\&H(t_5, a_1)\&P(a_5) \to \neg P(a_5)\&P(a_1)$;

u_5: $mu_5 = F(b_5, u_5)\&H(u_5, b_1)\&P(b_5) \to \neg P(b_5)\&P(b_1)$.

For example, a meaningful description of the production rule mt_1 for the transition t_1 would be like this: "if the position a_1 is the input for the transition t_1, the positions a_2 and mes_1 are the outputs for the same transition, and at position a_1 there is a marker, then from the position a_1, this marker is deleted, and in each of the positions a_2 and mes_1 appears one marker".

Speaking in a formal language, if before the firing of the transition t_1, a compound statement below was true:

$F(a_1, t_1)\&H\,(t_1, a_2)\&H(t_1, mes_1)\&P(a_1)$,

Then after firing this transition, the following compound statement will be true:

$\neg P\,(a_1)\&P(a_2)\&P(mes_1)$.

In a real client-server system, firing of t_1 corresponds to the transmission of message mes_1 from the client application to the server application through the corresponding logical and physical I/O ports. The production rules for firing transitions are interpreted, and the corresponding software modules are then executed in a networked environment.

For the second mode of operation (it corresponds to the implementation of the RPN network in figure 2), the "parametrically tunable" transition rules R_2, or the rules of logical inference, are presented below. The asterisks here indicate alternative rules for the firing of RPN network transitions:

The system of parametrically tunable product rules R_2:

t_1: $mt_1^* = F(a_1, t_1)\&F(mes_3, t_1)\&H(t_1, a_2)\&P(a_1)\&P(mes_3) \to \neg P\,(a_1)\&\neg P\,(mes_3)\&P(a_2)$;

u_1: $mu_1^* = F(b_1, u_1)\&H(u_1, mes_3)\&H(u_1, b_2)\&P(b_1) \to \neg P(b_1)\&P(mes_3)\&P(b_2)$;

t_2: $mt_2^* = F(a_2, t_2)\&H(t_2, a_3)\&H(t_3, ack_3)\&P(a_2) \to \neg P\,(a_2)\&P(a_3)\&P(ack_3)$;

u_2: $mu_2^* = F(b_2, u_2)\&F(ack_3, u_2)\&H(u_2, b_3)\&P(ack_3)\&P(b_2) \to \neg P(ack_3)\&\neg P(b_2)\&P(b_3)$;

t_3: $mt_3^* = F(a_3, t_3)\&H(t_3, a_4)\&H(t_3, mes_4)\&P(a_3) \to \neg P(a_3)\&P(a_4)\&P(mes_4)$;

u_3: $mu_3^* = F(b_3, u_3)\&F(mes_4, u_3)\&H(u_3, b_4)\&P(mes_4)\&P(b_3) \to \neg P(mes_4)\&\neg P(b_3)\&P(b_4)$;

t_4: $mt_4^* = F(a_4, t_4)\&F(ack_4, t_4)\&H(t_4, a_5)\&P(a_4)\&P(ack_4) \to \neg P(a_4)\&\neg P(ack_4)\&P(a_5)$;

u_4: $mu_4^* = F(b_4, u_4)\&H(u_4, ack_4)\&H(u_4, b_5)\&P(b_4) \to \neg P(b_4)\&P(ack_4)\&P(b_5)$;

t_5: $mt_5^* = F(a_5, t_5)\&H(t_5, a_1)\&P(a_5) \to \neg P(a_5)\&P(a_1)$;

u_5: $mu_5^* = F(b_5, u_5)\&H(u_5, b_1)\&P(b_5) \to \neg P(b_5)\&P(b_1)$.

Depending on the structural links established in the network of RPN between positions and transitions, one or another implementation of the network is established. In this case, for each RPN transition, two fire rules are defined and, with the established structure, for each transition, only one rule is chosen for possible execution.

For the network shown in figure 1 and figure 2, an example of the RPN network reconfiguration will be given, which makes it possible to switch from the network in figure 1 to the network in figure 2, that is, when the reconfiguration, the transition from an exchange of the type "Client → Server" to the exchange of the type "Server → Client".

1. All links of the positions mes_1, ack_1, mes_2, ack_2 with the transitions (figure 1) are broken:

$H(t_1, mes_1) \leftarrow false$; $F(mes_1, u_1) \leftarrow false$; $H(u_2, ack_1) \leftarrow false$;
$F(ack_1, t_2) \leftarrow false$; $H(u_3, mes_2) \leftarrow false$; $F(mes_2, t_3) \leftarrow false$;
$H(t_4, ack_2) \leftarrow false$; $F(ack_2, u_4) \leftarrow false$.

(It is assumed that these statements were true before the operations were performed).

2. New links are established for the positions mes_3, ack_3, mes_4, ack_4 with transitions (figure 2):

$H(u_1, mes_3) \leftarrow true$; $F(mes_3, t_1) \leftarrow true$; $H(t_2, ack_3) \leftarrow true$;
$F(ack_3, u_2) \leftarrow true$; $H(t_3, mes_4) \leftarrow true$; $F(mes_4, u_3) \leftarrow true$;
$H(u_4, ack_4) \leftarrow true$; $F(ack_4, t_4) \leftarrow true$.

(It is assumed that these statements were false before the operations were performed).

The advantage of this approach to the construction of RPN networks is that the structural links between positions and transitions are specified by propositional functions, or predicates, F and H, which substantially simplifies the parametric adjustment of the network to the required mode of operation.

6. Simplification of production rules

The process of reconfiguring networks of the RPN type may in many cases not involve all structural links between positions and transitions, therefore, some of the true statements of the form $F(p_i, t_i)$ and $H(t_m, p_n)$ involved in the conjunctions can be simply omitted. Then the system of simplified production rules R_1', obtained by excluding constantly true statements from the system R_1, will take the following form:

The simplified production rules system R_1':

t_1: $mt_1 = H(t_1, mes_1)\&P(a_1) \to \neg P(a_1)\&P(a_2)\&P(mes_1)$;
u_1: $mu_1 = F(mes_1, u_1)\&P(mes_1)\&P(b_1) \to \neg P(mes_1)\& \neg P(b_1)\&P(b_2)$;

t_2: $mt_2 = F(ack_1, t_2)\&P(a_2)\&P(ack_1) \to \neg P(a_2)\&\neg P(ack_1)\&P(a_3)$;

u_2: $mu_2 = H(u_2, ack_1)\&P(b_2) \to \neg P(b_2)\&P(ack_1)\&P(b_3)$;

t_3: $mt_3 = F(mes_2, t_3)\&P(a_3)\&P(mes_2) \to \neg P(a_3)\&\neg P(mes_2)\&P(a_4)$;

u_3: $mu_3 = H(u_3, mes_2)\&P(b_3) \to \neg P(b_3)\&P(mes_2)\&P(b_4)$;

t_4: $mt_4 = H(t_4, ack_2)\&P(a_4) \to \neg P(a_4)\&P(a_5)\&P(ack_2)$;

u_4: $mu_4 = F(ack_2, u_4)\&P(b_4)\&P(ack_2) \to \neg P(b_4)\&\neg P(ack_2)\&P(b_5)$;

t_5: $mt_5 = P(a_5) \to \neg P(a_5)\&P(a_1)$;

u_5: $mu_5 = P(b_5) \to \neg P(b_5)\&P(b_1)$.

Similarly, based on R_2, a system of simplified product rules R_2' is formed:

The simplified production rules system R_2':

t_1: $mt_1^* = F(mes_3, t_1)\&P(a_1)\&P(mes_3) \to \neg P\ (a_1)\&\neg P\ (mes_3)\&P(a_2)$;

u_1: $mu_1^* = H(u_1, mes_3)\&P(b_1) \to \neg P\ (b_1)\&P(mes_3)\&P(b_2)$;

t_2: $mt_2^* = H(t_3, ack_3)\&P(a_2) \to \neg P\ (a_2)\&P(a_3)\&P(ack_3)$;

u_2: $mu_2^* = F(ack_3, u_2)\&P(ack_3)\&P(b_2) \to \neg P(ack_3)\&\neg P\ (b_2)\&P\ (b_3)$;

t_3: $mt_3^* = H(t_3, mes_4)\&P(a_3) \to \neg P\ (a_3)\&P(a_4)\&P(mes_4)$;

u_3: $mu_3^* = F(mes_4, u_3)\&P(mes_4)\&P(b_3) \to \neg P(mes_4)\&\neg P(b_3)\&P(b_4)$;

t_4: $mt_4^* = F(ack_4, t_4)\&P(a_4)\&P(ack_4) \to \neg P(a_4)\&\neg P(ack_4)\&P(a_5)$;

u_4: $mu_4^* = H(u_4, ack_4)\&P(b_4) \to \neg P\ (b_4)\&P(ack_4)\&P(b_5)$;

t_5: $mt_5^* = P(a_5) \to \neg P(a_5)\&P(a_1)$;

u_5: $mu_5^* = P(b_5) \to \neg P\ (b_5)\&P(b_1)$.

7. Logical-algebraic operational expressions for reconfigurable Petri nets

Direct programming of distributed applications with the help of production expressions is difficult, because of the operational semantics of product rules is not obvious. Therefore, for an explicit representation of the procedural and operational semantics of models originally represented by RPN networks, it is advisable to move from the system of production rules to logical-algebraic operational expressions (LAOE) describing the functioning of the RPN. Therefore, the transition happens from the systems of product rules R_1 and R_2 to LAOEs L_1 and L_2, respectively.

The expressions for the LAOEs form are constructed on the basis of the superposition principle of α-disjunctions introduced by V. M. Glushkov in systems of algorithmic algebras.

The α-disjunction operation allows to represent the operator *if*: $[\alpha](A_1 \vee A_2)$ in an analytic form. Here, depending on the true or false value of the condition α, the operator A_1 or the operator A_2, respectively, is executed. Further, the superposition of two operations of α-disjunction is also used: $[\alpha_1]([\alpha_2](A_1 \vee A_2) \vee A_3)$, where among the operators there can be an empty or identical operator E. In addition, the modification predicate P operators are used, for example, $P(a_1)\leftarrow$**false**, $P(a_2)\leftarrow$**true**. This presentation has clear operational semantics and allows the creation of modular network software.

The system of logical-algebraic operational expressions L_1:

t_1: $Lmt_1 = [F(a_1, t_1)\&H(t_1, a_2)\&H(t_1, mes_1)]([P(a_1)](\{P(a_1)\leftarrow$**false**, $P(a_2)\leftarrow$**true**, $P(mes_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_1: $Lmu_1 = [F(b_1, u_1)\&F(mes_1, u_1)\&H(u_1, b_2)]$

$([P(mes_1)\&P(b_1)](\{P(mes_1)\leftarrow$**false**, $P(b_1)\leftarrow$**false**, $P(b_2)\leftarrow$**true**$\} \vee E) \vee E)$;

t_2: $Lmt_2 = [F(a_2, t_2)\&F(ack_1, t_2)\&H(t_2, a_3)]([P(a_2)\&P(ack_1)](\{P(a_2)\leftarrow$**false**, $P(ack_1)\leftarrow$**false**, $P(a_3)\leftarrow$**true**$\} \vee E) \vee E)$;

u_2: $Lmu_2 = [F(b_2, u_2)\&H(u_2, ack_1)\&H(u_2, b_3)]([P(b_2)](\{P(b_2)\leftarrow$**false**, $P(ack_1)\leftarrow$**true**, $P(b_3)$ $\leftarrow$ **true**$\} \vee E) \vee E)$;

t_3: $Lmt_3 = [F(a_3, t_3)\&F(mes_2, t_3)\&H(t_3, a_4)]([P(a_3)\&P(mes_2)](\{P(a_3)\leftarrow$**false**, $P(mes_2)\leftarrow$**false**, $P(a_4)\leftarrow$**true**$\} \vee E) \vee E)$;

u_3: $Lmu_3 = [F(b_3, u_3)\&H(u_3, mes_2)\&H(u_3, b_4)]([P(b_3)](\{P(b_3)\leftarrow$**false**,

$P(mes_2)\leftarrow$ **true**, $P(b_4)\leftarrow$**true**$\} \vee E) \vee E)$;

t_4: $Lmt_4 = [F(a_4, t_4)\&H(t_4, a_5)\&H(t_4, ack_2)]([P(a_4)](\{P(a_4)\leftarrow$**false**, $P(a_5)\leftarrow$**true**, $P(ack_2)\leftarrow$ **true**$\} \vee E) \vee E)$;

u_4: $Lmu_4 = [F(ack_2, u_4)\&F(b_4, u_4)\&H(u_4, b_5)]([P(b_4)\&P(ack_2)](\{P(b_4)\leftarrow$**false**, $P(ack_2)\leftarrow$ **false**, $P(b_5)\leftarrow$**true**$\} \vee E) \vee E)$;

t_5: $Lmt_5 = [F(a_5, t_5)\&H(t_5, a_1)]([P(a_5)](\{P(a_5)\leftarrow$**false**, $P(a_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_5: $Lmu_5 = [F(b_5, u_5)\&H(u_5, b_1)]([P(b_5)](\{P(b_5)\leftarrow$**false**,

$P(b_1)\leftarrow$**true**$\} \vee E) \vee E)$.

The system of logical-algebraic operational expressions L_2:

t_1: $Lmt_1^* = [F(a_1, t_1)\&F(mes_3, t_1)\&H(t_1, a_2)]([P(a_1)\&P(mes_3)](\{P(a_1)\leftarrow$**false**,

$P(mes_3)\leftarrow$**false**, $P(a_2)\leftarrow$**true**$\} \vee E) \vee E)$;

u_1: $Lmu_1^* = [F(b_1, u_1)\&H(u_1, mes_3)\&H(u_1, b_2)]([P(b_1)](\{P(b_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**true**, $P(b_2)\leftarrow$**true**$\} \vee E) \vee E)$;

t_2: $Lmt_2^* = [F(a_2, t_2)\&H(t_2, a_3)\&H(t_3, ack_3)]([P(a_2)](\{P(a_2)\leftarrow$**false**, $P(a_3)\leftarrow$**true**, $P(ack_3)\leftarrow$**true**$\} \vee E) \vee E)$;

u_2: $Lmu_2^* = [F(b_2, u_2)\&F(ack_3, u_2)\&H(u_2, b_3)]$

$([P(ack_3)\&P(b_2)](\{P(ack_3)\leftarrow$**false**, $P(b_2)\leftarrow$**false**, $P(b_3)\leftarrow$**true**$\} \vee E) \vee E)$;

t_3: $Lmt_3^* = [F(a_3, t_3)\&H(t_3, a_4)\&H(t_3, mes_4)]([P(a_3)](\{P(a_3)\leftarrow$**false**, $P(a_4)\leftarrow$**true**, $P(mes_4)\leftarrow$**true**$\} \vee E) \vee E)$;

u_3: $Lmu_3^* = [F(b_3, u_3)\&F(mes_4, u_3)\&H(u_3, b_4)]$

$([P(mes_4)\&P(b_3)](\{P(mes_4)\leftarrow$**false**, $P(b_3)\leftarrow$**false**, $P(b_4)\leftarrow$**true**$\} \vee E) \vee E)$;

t_4: $Lmt_4^* = [F(a_4, t_4)\&F(ack_4, t_4)\&H(t_4, a_5)]([P(a_4)\&P(ack_4)](\{P(a_4)\leftarrow$**false**, $P(ack_4)\leftarrow$**false**, $P(a_5)\leftarrow$**true**$\} \vee E) \vee E)$;

u_4: $Lmu_4^* = [F(b_4, u_4)\&H(u_4, ack_4)\&H(u_4, b_5)]([P(b_4)](\{P(b_4)\leftarrow$**false**, $P(ack_4)\leftarrow$**true**, $P(b_5)\leftarrow$**true**$\} \vee E) \vee E)$;

t_5: $Lmt_5^* = [F(a_5, t_5)\&H(t_5, a_1)]([P(a_5)](\{P(a_5)\leftarrow$**false**,

$P(a_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_5: $Lmu_5^* = [F(b_5, u_5)\&H(u_5, b_1)]([P(b_5)](\{P(b_5)\leftarrow$**false**,

$P(b_1)\leftarrow$**true**$\}\vee E) \vee E)$.

For the given structural relationship between the transitions and positions that remain unchanged in the reconfiguration of the CPR, the systems of expressions L_1 and L_2 were simplified. As a result, expression systems L_1' and L_2' were obtained:

The system of reduced logical-algebraic operating expressions L_1':

t_1: $Lmt_1 = [H(t_1, mes_1)]([P(a_1)](\{P(a_1)\leftarrow$**false**, $P(a_2)\leftarrow$**true**,
$P(mes_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_1: $Lmu_1 = [F(mes_1, u_1)]([P(mes_1)\&P(b_1)](\{P(mes_1)\leftarrow$**false**, $P(b_1)\leftarrow$**false**, $P(b_2)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_2: $Lmt_2 = [F(ack_1, t_2)]([P(a_2)\&P(ack_1)](\{P(a_2)\leftarrow$**false**, $P(ack_1)\leftarrow$**false**, $P(a_3)\leftarrow$**true**$\} \vee E) \vee E)$;

u_2: $Lmu_2 = [H(u_2, ack_1)]([P(b_2)](\{P(b_2)\leftarrow$**false**, $(ack_1)\leftarrow$**true**,

$P(b_3) \leftarrow$ **true**$\} \vee E) \vee E)$;

t_3: $Lmt_3 = [F(mes_2, t_3)]([P(a_3)\&P(mes_2)](\{P(a_3)\leftarrow$**false**, $P(mes_2)\leftarrow$**false**, $P(a_4)\leftarrow$**true**$\} \vee E) \vee E)$;

u_3: $Lmu_3 = [H(u_3, mes_2)]([P(b_3)](\{P(b_3)\leftarrow$**false**, $P(mes_2)\leftarrow$ **true**,

$P(b_4)\leftarrow$**true**$\} \vee E) \vee E)$;

t_4: $Lmt_4 = [H(t_4, ack_2)]([P(a_4)](\{P(a_4)\leftarrow$**false**, $P(a_5)\leftarrow$**true**,

$P(ack_2) \leftarrow$ **true**$\} \vee E) \vee E)$;

u_4: $Lmu_4 = [F(ack_2, u_4)]([P(b_4)\&P(ack_2)](\{P(b_4)\leftarrow$**false**, $P(ack_2)\leftarrow$ **false**, $P(b_5)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_5: $Lmt_5 = [P(a_5)](\{P(a_5)\leftarrow$**false**, $P(a_1)\leftarrow$**true**$\} \vee E)$;

u_5: $Lmu_5 = [P(b_5)](\{P(b_5)\leftarrow$**false**, $P(b_1)\leftarrow$**true**$\} \vee E)$.

The system of reduced logical-algebraic operating expressions L_2':

t_1: $Lmt_1^* = [F(mes_3, t_1)]([P(a_1)\&P(mes_3)](\{P(a_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**false**, $P(a_2)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

u_1: $Lmu_1^* = [H(u_1, mes_3)]([P(b_1)](\{P(b_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**true**,

$P(b_2) \leftarrow$**true**$\} \vee E) \vee E)$;

t_2: $Lmt_2^* = [H(t_3, ack_3)]([P(a_2)](\{P(a_2)\leftarrow$**false**, $P(a_3)\leftarrow$**true**,

$P(ack_3)\leftarrow$**true**$\} \vee E) \vee E)$;

u_2: $Lmu_2^* = [F(ack_3, u_2)]([P(ack_3)\&P(b_2)](\{P(ack_3)\leftarrow$**false**, $P(b_2)\leftarrow$**false**, $P(b_3)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_3: $Lmt_3^* = [H(t_3, mes_4)]([P(a_3)](\{P(a_3)\leftarrow$**false**, $P(a_4)\leftarrow$**true**,

$P(mes_4) \leftarrow$**true**$\} \vee E) \vee E)$;

u_3: $Lmu_3^* = [F(mes_4, u_3)]([P(mes_4)\&P(b_3)](\{P(mes_4)\leftarrow$**false**, $P(b_3)\leftarrow$**false**, $P(b_4)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_4: $Lmt_4^* = [F(ack_4, t_4)]([P(a_4)\&P(ack_4)](\{P(a_4)\leftarrow$**false**, $P(ack_4)\leftarrow$**false**, $P(a_5)\leftarrow$**true**$\} \vee E) \vee E)$;

u_4: $Lmu_4^* = [H(u_4, ack_4)]([P(b_4)](\{P(b_4)\leftarrow$**false**, $P(ack_4)\leftarrow$**true**,

$P(b_5)\leftarrow$**true**$\} \vee E) \vee E)$;

t_5: $Lmt_5^* = [P(a_5)](\{P(a_5)\leftarrow$**false**, $P(a_1)\leftarrow$**true**$\} \vee E)$;

u_5: $Lmu_5^* = [P(b_5)](\{P(b_5)\leftarrow$**false**, $P(b_1)\leftarrow$**true**$\} \vee E)$.

8. External control of the reconfigurable Petri nets

Considering the following way of managing the reconfiguration of the RPN, based on external management of the network by another, control, or supervisory network. In the simplest case, this control may be organized by using only one position (figure 5). To select one of the two

firing rules for each transition, the *Contr* entry is entered, which is connected with the transitions of the reconfigurable network by inhibitory (with non-blackened circles at the ends) or information arcs (with black circles at the ends). The inhibitor link of the *Contr* position with some transition blocks its operation if there is a label in it, and the information link blocks the transition when there is no label in the *Contr* position.

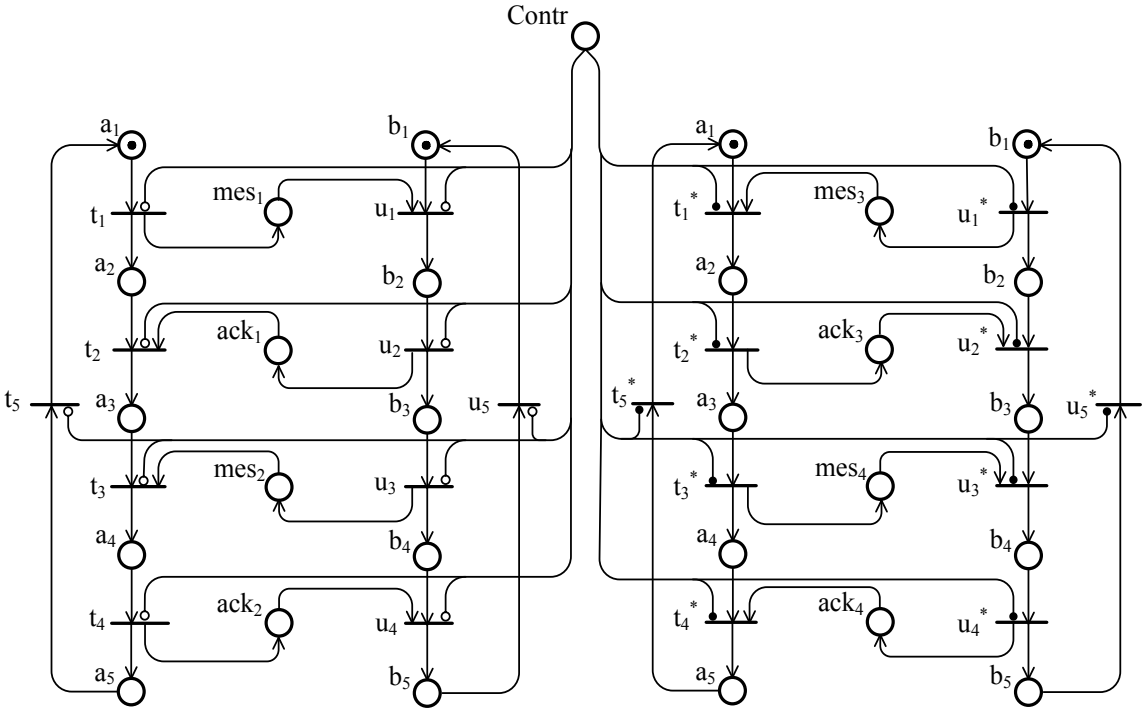


Fig. 5. Reconfigurable Petri net with external control from the Contr position of the supervisory network

Depending on the type of these links, one of the two firing rules is selected for each transition. The rules for firing transitions are presented below in the form of logical-algebraic operational expressions. Here $P(Contr)$ is a statement that is true when there is a marker in the control position *Contr*, *P* is an unary predicate markup symbol for the RPN network positions.

***The system of logical-algebraic operating expressions C_1 (external control of the RPN)*:**

t_1: $Cmt_1 = [\neg P(Contr)]([P(a_1)](\{P(a_1)\leftarrow$**false**, $P(a_2)\leftarrow$**true**,
$P(mes_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_1: $Cmu_1 = [\neg P(Contr)]([P(mes_1)\&P(b_1)](\{P(mes_1)\leftarrow$**false**, $P(b_1)\leftarrow$**false**, $P(b_2)\leftarrow$**true**$\} \vee E) \vee E)$;

t_2: $Cmt_2 = [\neg P(Contr)]([P(a_2)\&P(ack_1)](\{P(a_2)\leftarrow$**false**, $P(ack_1)\leftarrow$**false**, $P(a_3)\leftarrow$**true**$\} \vee E) \vee E)$;

u_2: $Cmu_2 = [\neg P(Contr)]([P(b_2)](\{P(b_2)\leftarrow$**false**, $(ack_1)\leftarrow$**true**,
$P(b_3) \leftarrow$ **true**$\} \vee E) \vee E)$;

t_3: $Cmt_3 = [\neg P(Contr)]([P(a_3)\&P(mes_2)](\{P(a_3)\leftarrow$**false**, $P(mes_2)\leftarrow$**false**, $P(a_4)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

u_3: $Cmu_3 = [\neg P(Contr)]([P(b_3)](\{P(b_3)\leftarrow$**false**, $P(mes_2)\leftarrow$ **true**,
$P(b_4)\leftarrow$**true**$\} \vee E) \vee E)$;

t_4: $Cmt_4 = [\neg P(Contr)]([P(a_4)](\{P(a_4)\leftarrow$**false**, $P(a_5)\leftarrow$**true**,
$P(ack_2) \leftarrow$ **true**$\} \vee E) \vee E)$;

u_4: $Cmu_4 = [\neg P(Contr)]([P(b_4)\&P(ack_2)](\{P(b_4)\leftarrow$**false**, $P(ack_2)\leftarrow$ **false**, $P(b_5)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_5: $Cmt_5 = [\neg P(Contr)]([P(a_5)](\{P(a_5)\leftarrow$**false**, $P(a_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_5: $Cmu_5 = [\neg P(Contr)]([P(b_5)](\{P(b_5)\leftarrow$**false**, $P(b_1)\leftarrow$**true**$\} \vee E) \vee E)$.

***The system of logical-algebraic operating expressions C_2 (external control of the RPN)*:**

t_1: $Cmt_1^* = [P(Contr)]([P(a_1)\&P(mes_3)](\{P(a_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**false**,
$P(a_2)\leftarrow$**true**$\} \vee E) \vee E)$;

u_1: $Cmu_1^* = [P(Contr)]([P(b_1)](\{P(b_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**true**,
$P(b_2) \leftarrow$**true**$\} \vee E) \vee E)$;

t_2: $Cmt_2^* = [P(Contr)]([P(a_2)](\{P(a_2)\leftarrow$**false**, $P(a_3)\leftarrow$**true**,
$P(ack_3)\leftarrow$**true**$\} \vee E) \vee E)$;

u_2: $Cmu_2^* = [P(Contr)]([P(ack_3)\&P(b_2)](\{P(ack_3)\leftarrow$**false**, $P(b_2)\leftarrow$**false**, $P(b_3)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_3: $Cmt_3^* = [P(Contr)]([P(a_3)](\{P(a_3)\leftarrow$**false**, $P(a_4)\leftarrow$**true**,
$P(mes_4) \leftarrow$**true**$\} \vee E) \vee E)$;

u_3: $Cmu_3^* = [P(Contr)]([P(mes_4)\&P(b_3)](\{P(mes_4)\leftarrow$**false**, $P(b_3)\leftarrow$**false**, $P(b_4)\leftarrow$**true**$\} \vee \vee E) \vee E)$;

t_4: $Cmt_4^* = [P(Contr)]([P(a_4)\&P(ack_4)](\{P(a_4)\leftarrow$**false**, $P(ack_4)\leftarrow$**false**, $P(a_5)\leftarrow$**true**$\} \vee E) \vee E)$;

u_4: $Cmu_4^* = [P(Contr)]([P(b_4)](\{P(b_4)\leftarrow$**false**, $P(ack_4)\leftarrow$**true**,
$P(b_5)\leftarrow$**true**$\} \vee E) \vee E)$;

t_5: $Cmt_5^* =[P(Contr)]([P(a_5)](\{P(a_5)\leftarrow$**false**, $P(a_1)\leftarrow$**true**$\} \vee E) \vee E)$;

u_5: $Cmu_5^* =[P(Contr)]([P(b_5)](\{P(b_5)\leftarrow$**false**, $P(b_1)\leftarrow$**true**$\} \vee E) \vee E)$.

To explain the choice of the rule for firing a transition, for example, t_1. For this transition, one of the following two firing rules can be selected:

$Cmt_1 = [\neg P(Contr)]([P(a_1)](\{P(a_1)\leftarrow$**false**, $P(a_2)\leftarrow$**true**,
$P(mes_1)\leftarrow$**true**$\} \vee E) \vee E)$
or
$Cmt_1^* = [P(Contr)]([P(a_1)\&P(mes_3)](\{P(a_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**false**,
$P(a_2)\leftarrow$**true**$\} \vee E) \vee E)$.

If there is no marker in the *Contr* position, then the statement $\neg P(Contr)$ is true and further, if the statement $P(a_1)$ is true, the actions in the curly brackets $\{P(a_1)\leftarrow$**false**, $P(a_2)\leftarrow$**true**, $P(mes_1)\leftarrow$**true**$\}$ are performed, modeling the removal of a marker from position a_1 and the appearance of markers in positions a_2 and mes_1, that is, the transition t_1 is firing in accordance with the rule Cmt_1.

If there is a marker in the *Contr* position, the statement $P(Contr)$ is true and further, if the statements $P(a_1)$ and $P(mes_3)$ are true, the actions in the curly brackets $\{P(a_1)\leftarrow$**false**, $P(mes_3)\leftarrow$**false**, $P(a_2)\leftarrow$**true**$\}$, modeling the removal of markers from positions a_1 and mes_3 and the appearance of a marker in position a_2, that is, the transition t_1 is firing in accordance with the rule Cmt_1^*. Similarly, the firing rules for the other transitions are selected.

Conclusion and future work

In RPN networks, if necessary, more complex management for the reconfiguration of the structure can be organized by using several external positions. The construction of the proposed kind of Petri nets is also based on a structural and reconfigurable approach, differing, however, taking into account the structural links and supervisory control. In addition, dynamic management for the reconfiguration of RPN can be organized, for example, after certain transitions are fired. For this, it will be necessary to define the links of these transitions with the supervisory positions. In this paper, the examples were used to construct a special type of Petri nets – RPNs, whose graphical representation contains duplicate positions. The work of such a Petri net is typical for peer-to-peer and other computer systems.

References

1. Peterson, J. L. Petri Net Theory and the Modeling of Systems / J. L. Peterson. – N. Y. : Prentice-Hall, 1981. – 288 p.

2. Murata, T. Petri nets: properties, analysis and applications / T. Murata // Proceedings of the IEEE. – 1989. – № 77 (4). – P. 541–580.

3. W. M. P. van der Aalst. A class of Petri nets for modeling and analyzing business processes // Computing Science Reports 95/26. – Eindhoven University of Technology, Eindhoven, 1995. – P. 1–25.

4. Hruz, B. Modeling and Control of Discrete-event Dynamic Systems with Petri Nets and Other Tools. Springer-Verlag / B. Hruz, M. C. Zhou. – London, 2007. – 351 p.

5. Iordache, M. V. Supervisory Control of Concurrent Systems. A Petri Net Structural Approach / M. V. Iordache, P. J. Antsaklis. – Boston : Birkhauser, 2006. – 281 p.

6. Jahns, R. On Modification in Petri Nets / R. Jahns, M. Becker, T. Bessey, H. Szczerbicka // Journal Proc. Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS). – 2002. – 5 p.

7. Valk, R. Self-modifying Nets / R. Valk // Report Bericht Nr. 34, IFI-HH-B-34/77, from the Institute of Informatics. – University of Hamburg, West Germany, 1977. – July. – 36 p.

8. Jiang, Z. Temporized Colored Petri Nets with Changeable Structure (TCPN-CS) for the Performance Modeling of Dynamic Production Systems / Z. Jiang, M. J. Zuo, R. Y. K. Fung, P. Y. Tu. // International Journal of Production Research. – 2000. – № 8. – P. 1917–1945.

9. Holloway, L. E. Survey of Petri Net Methods for Controlled Discrete Event Systems / L. E. Holloway, B. H. Krogh, A. A. Giua // Discrete Event Dynamic Systems: Theory and Applications. – Kluwer Academic Publishers, Boston : Manufactured in The Netherlands, 1997. – P. 151–190.

10. Krogh, B. H. Controlled Petri nets and maximally permissive feedback logic / B. H. Krogh // Proc. 25th Annual Allerton Conference. – University of Illinois, Urbana, 1987. – P. 317–326.

11. Ichikawa, A. Analysis and control of discrete event systems represented by Petri nets / A. Ichikawa, K. Hiraishi // Discrete Event Systems: Models and Applications, Lecture Notes in Control and Information Sciences / Varaiya, P. and Kurzhanski A. B., eds. – New York : Springer Verlag, 1988. – Vol. 103. – P. 115–134.

12. Omar, Yaqub. Modeling and Analysis of Connected Traffic Intersections Based on Modified Binary Petri Nets / Yaqub Omar, Li. Lingxi // Hindawi Publishing Corporation, International Journal of Vehicular Technology. – 2013. – 10 p. – URL: http://dx.doi.org/10.1155/2013/

13. Jianxin, L An Inter-Working Petri Net Model between SIMPLE and IMPS for XDM Service / Liao Jianxin, Yuting Zhang, Xiaomin Zhu // Petri Net, Theory and Applications. – 2008. – P. 73–90. – URL: www.intechopen.com

14. Petri Net Transformations / H. Ehrig, K. Hoffmann, J. Padberg, C. Ermel, U. Prange, E. Biermann, T. Modica // Chapter from the book Petri Net, Theory and Applications. – 2008. – P. 1–16. – URL: http://www.intechopen.com/books/petri_net_theory_and_applications

15. Petri Nets Fundamental Models // Verification and Applications / ed.by Michel Diaz. – John Wiley & Sons, 2009. – 585 p.

16. Svadova, M. Matlab Toolbox for Petri Nets / M. Svadova, Z. Hanzalek. – URL: https://simonline.akademitelkom.ac.id/dosen/riset/smartgrid/Bukan/Hanzalek01a.pdf

17. Petri Nets Tools Database Quick Overview. – URL: www.informatik.uni-hamburg.de

Mustafa Sadeq Jaafar, Sergey A. Zinkin

Penza State University, Penza, Russia

THE IMPLEMENTATION OF GLOBAL COMPUTING THROUGH THE MAPPING OF OBJECT-ORIENTED PETRI NETS INTO THE ARCHITECTURE OF DISTRIBUTED COMPUTING SYSTEMS

The article deals with the problems of mapping Petri nets to the system and functional architectures of distributed computing systems at the middleware level. In particular, the problem of mapping Petri nets to the TCP/IP network is solved experimentally. The need to solve this problem came from the fact that various methods for mapping algorithms to the architecture of parallel and distributed computing systems are known, but only a few of them are oriented to the development of middleware. The importance of the problem is explained by the fact that the definition of middleware overlaps with many existing technologies, such as Cloud Computing, Grid Computing, Utility Computing, Services Computing, and Distributed Computing in general. It is shown that different interpretations of its work for the same Petri net can be used, which significantly affect the quality of the middleware class network software. An example and detailed comments for the distributed Java program are given.

1. Introduction

Cloud Computing is defined as a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet. This definition of Cloud Computing overlaps with many existing technologies, such as Grid Computing, Utility Computing, Services Computing, and Distributed Computing in general [1]. The problems in this area cover progressively the hardware and software architectures that underpin modern massively parallel computer systems; the concepts and technologies that enable cloud and distributed computing; and advanced topics in distributed computing, including grid, peer-to-peer, and the Internet of Things [2].

The main areas related to the development of Cloud Computing and Grid Computing cover the following topics [3]:

– optimal online deterministic algorithms and adaptive heuristics for energy and performance;

– efficient dynamic consolidation of virtual machines in cloud data centers;

– a check-pointing enabled and resource-aware Java Virtual Machine (VM) for efficient and reliable e-science applications in grid environments;

– a bi-criteria scheduling process with Class of Services support on Grids and Clouds;

– a component-based middleware for hybrid Grid/Cloud Computing platforms;

– a peer-to-peer scheduling mechanism for workflows in Grid Computing;

– a provenance management system for scientific workflows;

– an adaptive parallel execution strategy for cloud-based scientific workflows etc.

When building systems similar to this distributed “Cloud” or “Grid” systems, a convenient toolkit, based on formalized specifications of their system and functional architecture is needed. Data processing technologies based on distributed object-oriented Petri nets (DOOPN) allow building real distributed architectures. Despite the growing popularity of the use of formal tools based Petri nets, when creating prototypes of distributed computing systems on the hardware and

software platforms of local and global networks, the implementation of such tools has not yet found wide distribution, as indicated by an overview of known tools from the site http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick [4]. A Java-program is proposed to implement one of the variants of such DOOPN networks. It is also possible to implement this program in the form of an applet, but in this case serious limitations are imposed on the configuration of DOOPN, due to the fact that the applet can establish a connection on the network only with the node from which it was downloaded. The technology of the DOOPN networks is closely related to networked computer technologies.

2. Interpretation of object-oriented Petri nets for applications to computer networks

First of all, some known information on Petri nets were given from [5, 6]. Petri nets were introduced in 1962 by Dr. Carl Adam Petri [6]. Additional information on Petri nets can be found in [7, 8] and others. A further description of conventional Petri nets is based on the citations from [5]: "A Petri net is a particular kind of bipartite directed graphs populated by three types of objects. These objects are *places*, *transitions*, and *directed arcs*. Directed arcs connect places to transitions or transitions to places. In order to study the dynamic behavior of a Petri net modeled system in terms of its states and state changes, each place may potentially hold either none or a positive number of *tokens*. Tokens are a primitive concept for Petri nets in addition to places and transitions. The presence or absence of a token in a place can indicate whether a condition associated with this place is true or false, for instance.

A Petri net is formally defined as a 5-tuple $N = (P, T, F, H, M_0)$, where

(1) $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;

(2) $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions, $P \cup T \neq \varnothing$, and $P \cap T = \varnothing$;

(3) F: $P \times T \rightarrow N$ is an *input function* that defines directed arcs from places to transitions, where N is a set of nonnegative integers;

(4) H: $T \times P \rightarrow N$ is an *output function* that defines directed arcs from transitions to places;

(5) M_0: $P \rightarrow N$ is the *initial marking*.

A *marking* in a Petri net is an assignment of tokens (or markers) to the places of a Petri net. Tokens reside in the places of a Petri net. The number and position of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net.

The execution of a Petri net is controlled by the number and distribution of tokens in a Petri net. By changing the distribution of tokens in places, it may reflect the occurrence of events or execution of operations, for instance, it could help in studying the dynamic behavior of the modeled system. A Petri net executes by *firing* transitions. Now the enabling rule and firing rule of a transition, which govern the flow of tokens would be introduced in the following points:

(1) *Enabling Rule*: A transition t is said to be *enabled* if each input place p of t contains at least the number of tokens equals to the weight of the directed arc connecting p to t, i.e., $M(p) \geq F(p, t)$ for any p in P.

(2) *Firing Rule*: Only enabled transitions can be fired. The firing of an enabled transition t removes from each input place p the number of tokens equals to the weight of the directed arc connecting p to t. It also deposits in each output place p the number of tokens equals to the weight of the directed arc connecting t to p. Mathematically, firing t at M yields a new marking

$$M'(p) = M(p) - F(p, t) + H(t, p) \text{ for any } p \text{ in } P.$$

Notice that since only enabled transitions can fire, the number of tokens in each place always remains non-negative when a transition is fired. Firing transitions would never try to remove a token that is not there." On this, citing the description of the most famous Petri nets from [5] is finished.

Petri nets used in this work differ in that markers can have parameters in them, and input and output arcs can be weighted by conditions.

In this paper, a formalized method was proposed for creating network applications of the "middleware" class by embedding the Petri net into the architecture of computer networks for global computing. First, consider embedding to a computer network of a Petri net fragment containing one transition t_1, three inputs (p_1, p_2, p_3) and one output (p_4) positions. The rule r_1 for firing the transition t_1 is represented in the form of a logical-algebraic expression:

$$r_1 = [Mark(p_1)\&Mark(p_2)\&Mark(p_3)](\{Mark(p_1)\leftarrow\textbf{false}, Mark(p_2)\leftarrow\textbf{false}, Mark(p_3)\leftarrow\textbf{false}, Mark(p_4)\leftarrow\textbf{true}\} \vee E), \quad (1)$$

where $Mark$: $P \rightarrow \{\textbf{true}, \textbf{false}\}$ is an unary marking function defined for a 1-bounded fragment of the Petri net, $\alpha = Mark(p_1)\&Mark(p_2)\&Mark(p_3)$ – compound statement, the truth of which allows the firing of the transition t_1. Updating $Mark(p_i)\leftarrow\textbf{true}$ of the function $Mark$ means placing the marker in the position p_i, and updating $Mark(p_j)\leftarrow\textbf{false}$ of the same function – removing the marker from the position p_j. As soon as the condition α becomes true, the following updates to the function $Mark$ are performed: $\{Mark(p_1)\leftarrow\textbf{false}, Mark(p_2)\leftarrow\textbf{false}, Mark(p_3)\leftarrow\textbf{false}, Mark(p_4)\leftarrow\textbf{true}\}$. The curly brackets here have an important meaning: all the operations indicated in them are considered to be consistent. These operations can be executed in an arbitrary order or in parallel. The symbol E denotes an empty operator, which is satisfied in the case when the compound statement α is false.

First of all, it's necessary to consider the interpretation of Petri nets for applications to computer networks. The testing of the condition α can have a cyclic character, therefore, the operator E is replaced by the operator $Ret_{\tau,n}$:

$$r'_1 = [Mark(p_1)\&Mark(p_2)\&Mark(p_3)](\{Mark(p_1)\leftarrow\textbf{false}, Mark(p_2)\leftarrow\textbf{false}, Mark(p_3)\leftarrow\textbf{false}, Mark(p_4)\leftarrow\textbf{true}\} \vee Ret_{\tau,n}). \quad (2)$$

The operator $Ret_{\tau,n}$ returns control to check condition $\alpha = Mark(p_1)\&Mark(p_2)\&Mark(p_3)$ until it is true, or the number of re-attempts to fire the transition will be exhausted. Each repetition of an attempt to fire a transition t_1 is carried out after a certain time intervalτ.

In the general case, the expression

$$A;[\alpha](B \vee Ret_{\tau,n});C \quad (3)$$

corresponds to the following algorithm shown in figure 1. The symbol ";" here means the sequential execution of operators.

As it was stated above that the operations $\{Mark(p_1)\leftarrow\textbf{false}, Mark(p_2)\leftarrow\textbf{false}, Mark(p_3)\leftarrow\textbf{false}, Mark(p_4)\leftarrow\textbf{true}\}$ can be executed in an arbitrary order or in parallel one. However, during the interpretation of expressions (1) and (2) in computer networks and microprocessor systems, the order of operations is important. Consider, for example, the following expression:

$$r''_1 = [Mark(p_1)\&Mark(p_2)\&Mark(p_3)](\{(Mark(p_1)\leftarrow\textbf{false}, Mark(p_2)\leftarrow\textbf{false}, Mark(p_3)\leftarrow\textbf{false}); Mark(p_4)\leftarrow\textbf{true}\} \vee Ret_{\tau,n}). \quad (4)$$

In a sequence of statements enclosed in braces in (3), the symbols "," and ";" mean, respectively, arbitrary (possibly parallel) and sequential execution of actions.

Similarly, when implemented in computer networks, there can have a meaning of an order for verification of statements $Mark(p_1)$, $Mark(p_2)$ and $Mark(p_3)$, since this can be related to the order in which messages are sent and received in the computer network.

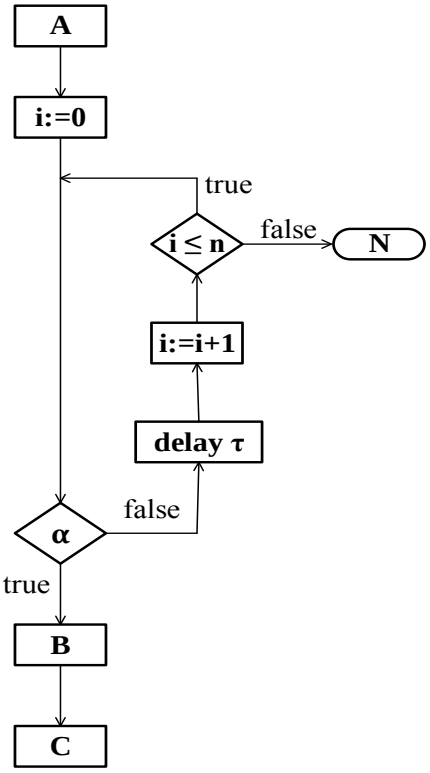


Fig. 1. Algorithm for the network implementation of the transition firing rule

Considering the case where the positions p_1, p_2, p_3, p_4 and the transition t_1 are placed on the nodes n_{p1}, n_{p2}, n_{p3}, n_{p4} and n_{t1} of the computer TCP/IP network, respectively (figure 2). Two unary functions f_{Node} and $f_{Address}$ were introduced:

$$f_{Node}: P \cup T \to Node;$$
$$f_{Address}: Node \to IPaddress,$$

where *Node* is a set of nodes that belongs to a computer network, *IPaddress* – a set of IP-addresses in a TCP/IP network. For example, the expression $f_{Address}(f_{Node}(p_1))$ allows you to determine the IP address 129.44.192.1 of the node $n_{p1} = f_{Node}(p_1)$ on which position p_1 is implemented.

Three variants for the implementation of the transition t_1 firing rule in a TCP/IP network were considered in this paper:

1. For the implementation of the rule r'_1 for firing the transition t_1, the node n_{t1} on which the transition t_1 is implemented makes a request for the markers to be present at the nodes n_{p1}, n_{p2} and n_{p3} and then initiates an independent and possibly a parallel transfer of their markers to the node n_{t1}. Then, the node n_{t1} sends a new marker to node n_{p4}, where the output position p_4 is implemented. The function updates $Mark(p_1) \leftarrow$ **false**, $Mark(p_2) \leftarrow$ **false** and $Mark(p_3) \leftarrow$ **false** correspond to sending acknowledgments to the node-positions that have been freed from the markers. The function update $Mark(p_4) \leftarrow$ **true** corresponds to the sending of the marker from the node-transition n_{t1} to the output node-position n_{p4}. The implementation of the rule r''_1 differs in that, the output marker is sent to the output node-position n_{p4} at the very final stage.

2. The second variant for the implementation of the firing rules is that, when polling, the markers in the input node-positions are blocked after their finding. Node-transition n_{t1} will then fire only if all markers will be present in the input positions (after receiving all of the markers-messages). Then, the output marker is sent from the node-transition n_{t1} to the output node-position n_{p4}.

3. In the third case, all the markers-messages from the input node-positions n_{p1}, n_{p2} and n_{p3} are immediately and independently transferred to the node-transition n_{t1}. Then, the output marker is sent from node-transition n_{t1} to the output node-position n_{p4}.

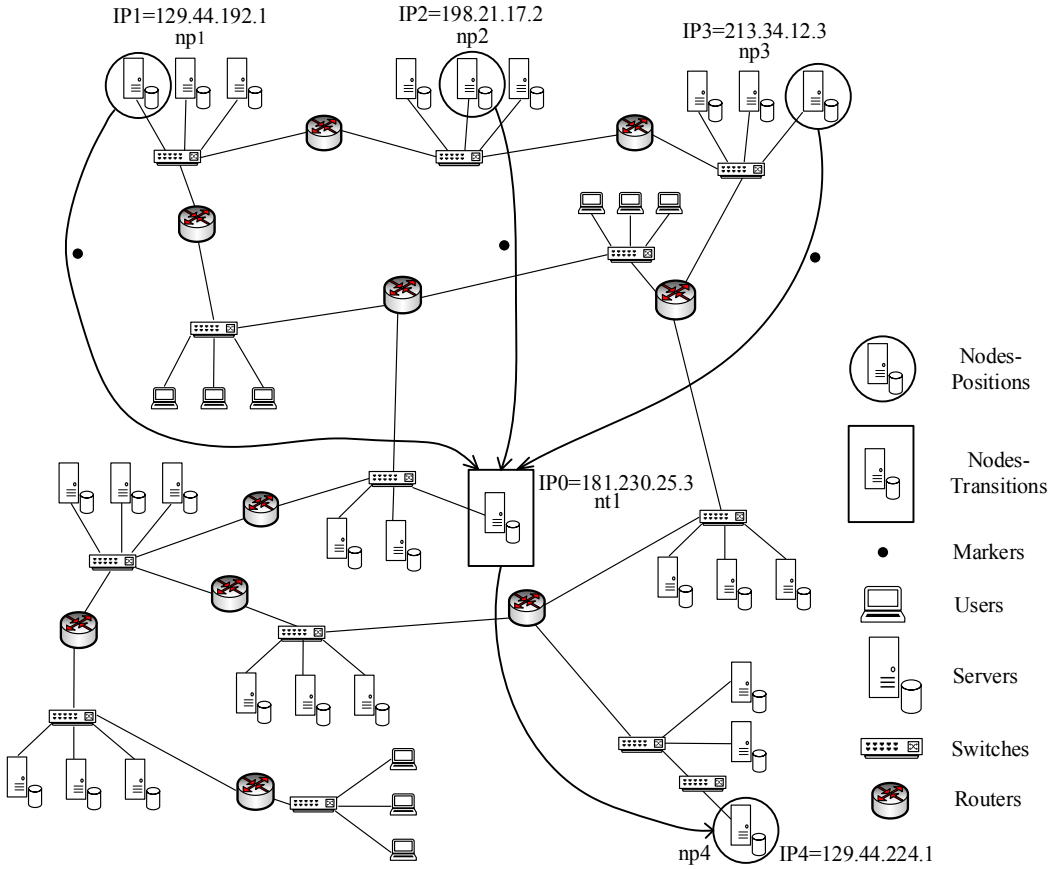


Fig. 2. An example of mapping object-oriented Petri nets into the architecture of distributed computing systems based on TCP/IP networks

Other implementations of the rules for firing transitions when Petri nets are embedded in the architecture of a computer network are also possible, so when developing middleware, the developer must choose an acceptable option.

3. The principles of creating a program for mapping object-oriented Petri nets into the architecture of distributed computing systems

Direct mapping of object-oriented Petri nets into the architecture of distributed computing systems is the most accessible to application software developers, as well as to developers of intermediate-type middleware software. A description of the demo version of the program, on the basis of which complex distributed applications can be created with a structure represented as a Petri net, is given below.

The object-oriented Petri net is represented by the classes Place and Transition. Between the objects of these classes, a marker which is implemented in the Mark class and transmitted over the TCP/IP network. To work with the TCP/IP network, the java.net package was used, in particular, the classes InetAddress, Socket and ServerSocket.

The Mark class describes a marker transmitted over the Petri net:

```
class Mark { …
```

Each marker can have parameters stored in the array parameter. The number of parameters in the markers is fixed:

```
final int MAX_PARAM = 3;
byte[] parametr = new byte[MAX_PARAM];
```

The marker constructor sets the parameter values:

```
Mark(){
    for (int i=0; i<MAX_PARAM; i++)
          parametr[i]=0;
    }
}
```

The Place class implements the network position:

```
class Place {
    private final int MAX_OUT = 1;
```

The out_adr and out_port arrays store information about arcs emerging from the position:

```
private InetAddress[] out_adr =  new netAddress[MAX_OUT];
    private int out_port[] = new int[MAX_OUT];
    private int in_port;
```

In this implementation, the position can contain only one marker:

```
    private Mark m;
```

The sendMark method dispatches markers from the position:

```
void sendMark() throws IOException{
          for(int i=0; i<MAX_OUT; i++){
               Socket s = new Socket(out_adr[i], out_port[i]);
            s.getOutputStream().write(m.parametr);
            s.close();
          }
    }
```

The position constructor creates a node with the given input and output links:

```
Place(int i_port, InetAddress[] o_adr, int[] o_port) {
          for(int i=0; i<MAX_OUT; i++){
              out_adr[i] = o_adr[i];
               out_port[i] = o_port[i];
          };
          in_port = i_port;
    }
}
```

The Transition class represents the transition of the Petri net:

```
class Transition {
    private final int MAX_OUT = 1;
    private final int COUNT_IN_MARK = 3;
```

The addresses of the output nodes of the network:

```
    private InetAddress[] out_adr = new InetAddress[MAX_OUT];
    private int out_port[] = new int[MAX_OUT];
    private int in_port;
```

The marker to be sent to the output nodes:

```
private Mark m_out_mark;
```

Markers needed to run the transition:

```
private Mark[] m_in_mark = new Mark[COUNT_IN_MARK];
    private static int n=0;
```

The transition constructor sets up the input and output links:

```
  Transition(int i_port, InetAddress[] o_adr, int[] o_port) {
        for(int i=0; i<MAX_OUT; i++){
            out_adr[i] = o_adr[i];
            out_port[i] = o_port[i];
        }
        in_port = i_port;
    }
```

The firing method contains the actions that are performed when the transition is started. If necessary, you can include more complex processes of marker parameters, as well as increase the number of outgoing markers for the implementation of conditional branching:

```
  void fire() {
        // transition's events
        m_out_mark = m_in_mark[0];
    }
```

The enable method checks for the presence of the tokens needed to start the transition:

```
boolean enable() {
        for(int i=0; i<COUNT_IN_MARK; i++)
            if (m_in_mark[i] == null)
                return false;
        return true;
    }
```

The sendMark method dispatches markers from the transition. Here, a conditional transfer of a marker can also be implemented:

```
void sendMark() throws IOException{
        int i;
        for(i=0; i<MAX_OUT; i++){
            Socket s = new Socket(out_adr[i], out_port[i]);
            s.getOutputStream().write(m_out_mark.parametr);
            s.close();
        }
        n=0;
        for(i=0; i<COUNT_IN_MARK; i++)
             m_in_mark[i] = null;
    }
```

The recvMark method accepts tokens:

```
void recvMark() throws IOException{
        ServerSocket ss = new ServerSocket(in_port);
        Socket s = ss.accept();
        InputStream ins = s.getInputStream();
        ins.read(m_in_mark[n++].parametr);
        s.close();
        ss.close();
    }
    void work() throws IOException{
        while (!enable())
            recvMark();
        fire();
        sendMark();
    }
}
```

The creation of a Petri net can be performed as follows, The PlaceNode class implements three node-positions that are connected to one transition:

```
class PlaceNode {
  public static void main(String[] args) throws Exception{
        // Positions creation
        Place place1, place2, place3;
        InetAddress[] outAdr = new InetAddress[1];
        int[] outPort = new int[1];
        outAdr[0] = InetAddress.getByName("127.0.0.1");
        outPort[0] = 12100;
```

```
            // Linking positions with a transition
            place1 = new Place(12001, outAdr, outPort);
            place2 = new Place(12002, outAdr, outPort);
            place3 = new Place(12003, outAdr, outPort);
            place1.sendMark();
            place2.sendMark();
            place3.sendMark();
            System.out.println(" All Marks sended.");
  }
}
```

The TransitionNode class implements a junction node with three input arcs and one output:

```
class TransitionNode {
  public static void main(String[] args) throws Exception{
            InetAddress[] outAdr = new InetAddress[1];
            int[] outPort = new int[1];
            outAdr[0] = InetAddress.getByName("127.0.0.1");
            outPort[0] = 12004;
            Transition tNode1 = new Transition(12100, outAdr, outPort);
            tNode1.work();
            System.out.println(" Transition fired.");
  }
}
```

An alternative way to implement the launching of both the PlaceNode and TransitionNode classes is to use lightweight processes. In this case, the specified classes are performed within the same process. This option is useful if several positions or transitions in the full text of the program of a Petri net fragment look like this, are implemented in one node of the network.

4. A program for mapping object-oriented Petri nets into the architecture of distributed computing systems

The full text of the program for the Petri net fragment of the DOOPN class looks like this:

```
import java.net.*;
import java.io.*;
class Mark {
      final int MAX_PARAM = 3;
      byte[] parametr = new byte[MAX_PARAM];

      Mark(){
            for (int i=0; i&ltMAX_PARAM; i++)
                  parametr[i]=0;
      }
}
```

```
class Place {
    private final int MAX_OUT = 1;
    private InetAddress[] out_adr = new InetAddress[MAX_OUT];
    private int out_port[] = new int[MAX_OUT];
    private int in_port;
    private Mark m;
    void sendMark() throws IOException{
        for(int i=0; i&ltMAX_OUT; i++){
            Socket s = new Socket(out_adr[i], out_port[i]);
            s.getOutputStream().write(m.parametr);
            s.close();
        }
    }
    Place(int i_port, InetAddress[] o_adr, int[] o_port) {
        for(int i=0; i&ltMAX_OUT; i++){
            out_adr[i] = o_adr[i];
            out_port[i] = o_port[i];
        };
        in_port = i_port;
    }
}
class Transition {
    private final int MAX_OUT = 1;
    private final int COUNT_IN_MARK = 3;
    private InetAddress[] out_adr = new InetAddress[MAX_OUT];
    private int out_port[] = new int[MAX_OUT];
    private int in_port;
    private Mark m_out_mark;
    private Mark[] m_in_mark = new Mark[COUNT_IN_MARK];
    private static int n=0;
    Transition(int i_port, InetAddress[] o_adr, int[] o_port) {
        for(int i=0; i&ltMAX_OUT; i++){
            out_adr[i] = o_adr[i];
            out_port[i] = o_port[i];
        }
        /*     for(int i=0; i&ltCOUNT_IN_MARK; i++)
            m_in_mark[i] = i_mark[i];*/
        in_port = i_port;
    }
    void fire() {
    // transition's events
        m_out_mark = m_in_mark[0];
    }
```

```
        boolean enable() {
            for(int i=0; i&ltCOUNT_IN_MARK; i++)
                if (m_in_mark[i] == null)
                    return false;
            return true;
        }
        void sendMark() throws IOException{
            int i;
            for(i=0; i&ltMAX_OUT; i++){
                Socket s = new Socket(out_adr[i], out_port[i]);
                s.getOutputStream().write(m_out_mark.parametr);
                s.close();
                n=0;
                for(i=0; i&ltCOUNT_IN_MARK; i++)
                    m_in_mark[i] = null;
            }
        }
        void recvMark() throws IOException{
            ServerSocket ss = new ServerSocket(in_port);
            Socket s = ss.accept();
            InputStream ins = s.getInputStream();
            ins.read(m_in_mark[n++].parametr);
            s.close();
            ss.close();
        }
        void work() throws IOException{
             while (!enable())
                  recvMark();
             fire();
             sendMark();
        }
}
class PlaceNode {
        public static void main(String[] args) throws Exception{
            Place place1, place2, place3;
            InetAddress[] outAdr = new InetAddress[1];
            int[] outPort = new int[1];
            outAdr[0] = InetAddress.getByName("127.0.0.1");
            outPort[0] = 12100;
            place1 = new Place(12001, outAdr, outPort);
            place2 = new Place(12002, outAdr, outPort);
            place3 = new Place(12003, outAdr, outPort);
            place1.sendMark();
```

```
            place2.sendMark();
            place3.sendMark();
            System.out.println(" All marks sended.");
        }
    }
    class TransitionNode {
        public static void main(String[] args) throws Exception{
            InetAddress[] outAdr = new InetAddress[1];
            int[] outPort = new int[1];
            outAdr[0] = InetAddress.getByName("127.0.0.1");
            outPort[0] = 12004;
            Transition tNode1 = new Transition(12100, outAdr, outPort);
            tNode1.work();
            System.out.println(" Transition fired.");
        }
    }
```

Conclusion

The given example is suitable for the realization in computer networks and demonstrates the mapping of a Petri net into the architecture of a distributed computing system. The realization of particular cases in some specific Petri networks of the DOOPN class requires the completion of this program, a specific interpretation of Petri net can take into account the priorities in triggering the transitions, the presence of additional conditions on the input and output arcs, the presence of inhibitory and information arcs, the multiplicity of arcs, the number of marks in positions, tag parameters and other properties.

References

1. Foster, Ian. Cloud Computing and Grid Computing 360-Degree Compared / Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu. – URL: https://www.researchgate.net/publication/23716795_Cloud_Computing_and_Grid_Computing_360-Degree_Compared

2. Hwang, Kai. Distributed and Cloud Computing: From Parallel Processing to the Internet of Things / Kai Hwang, Geoffrey C. Fox, and Jack J. Dongarra. – Elsevier (Singapore) Pte Ltd, 2012. – 630 p.

3. Schulze, B. Middleware for Clouds and e-Science. Editorial / B. Schulze, R. Buyya, F. Porto // Concurrency Computation: Practice and Experience. – John Wiley & Sons, Ltd., 2012.

4. Petri Nets Tools Database Quick Overview. – URL: http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick

5. Wang, J. Petri Nets for Dynamic Event-Driven System Modeling / J. Wang. – Department of Software Engineering. Monmouth University, West Long Branch, NJ 07764. – URL: http://bluehawk.monmouth.edu

6. Petri, C. A. Kommunikation mit Automaten / C. A. Petri. – New York, 1962.

7. Peterson, J. L. Petri Net Theory and the Modeling of Systems / J. L. Peterson. – New York : Prentice-Hall, 1981. – 288 p.

8. Murata, T. Petri nets: properties, analysis and applications / T. Murata // Proceedings of the IEEE. – 1989. – № 77(4). – P. 541–580.

Г. В. Мартяшин, И. С. Пышкина, Д. А. Трокоз

Пензенский государственный университет, Пенза, Россия

ОСОБЕННОСТИ ПОСТРОЕНИЯ КАРТ ПОКРЫТИЯ В СИСТЕМАХ ПАССИВНОГО МОНИТОРИНГА ТЕХНИЧЕСКИХ ПАРАМЕТРОВ СЕТЕЙ СОТОВОЙ СВЯЗИ

Обсуждается вопрос пассивного мониторинга технических параметров сетей сотовой связи. Предлагается методология построения карт покрытия для систем, основанных на пассивном мониторинге.

Диагностика и мониторинг являются неотъемлемыми составляющими предоставления услуг сотовой связи. Это подразумевает выполнение измерений, которые осуществляются с помощью специализированных измерительных комплексов. В настоящее время сотовые операторы преимущественно используют измерительные комплексы, разработанные зарубежными производителями, которые отличаются высокой стоимостью и при этом требуют постоянного нахождения высококвалифицированного инженера непосредственно рядом с аппаратурой комплекса для управления процессом измерения [1]. Решение этой проблемы заключается в использовании пассивного способа мониторинга, который позволяет производить автоматизированный сбор данных и отказаться от целенаправленных измерений. Использование современного смартфона, обладающего возможностью определения местоположения с помощью GPS и доступом к сети интернет, в качестве основы измерительного комплекса легко позволяет реализовать пассивный мониторинг. Этим была продиктована необходимость разработки программного комплекса диагностики и мониторинга сетей сотовой связи на базе смартфона, использующего пассивный подход к мониторингу [2].

Сотовая связь имеет в своей основе ключевую особенность, которая состоит в том, что общая зона покрытия делится на части, каждая из которых называется сотой и обслуживается совей базовой станцией, обеспечивающей зону покрытия соты. В идеальных условиях ровной поверхности и отсутствия каких-либо помех зона покрытия базовой станции представляет собой круг, однако, так как с целью обеспечения непрерывного покрытия зоны покрытия соседних сот частично перекрывают друг на друга, соты традиционно рассматриваются как граничащие шестиугольные ячейки [3]. Таким образом, сеть сотовой связи представляет собой множество территориально разнесенных базовых станций, обеспечивающих непрерывную зону покрытия, и коммутационное оборудование, связывающее соты в единое целое с телефонными сетями общего пользования.

Результаты мониторинга технических параметров сетей сотовой связи заключаются в построении актуальной, полученной эмпирическим путем, карты покрытия, то есть карты, представляющей графическую проекцию сигнала базовых станций на местность с градацией уровня сигнала. Такая карта крайне интересна по той причине, что в силу виляния различных факторов, таких как особенности ландшафта, наличие построек, наличие источников достаточно сильных радиоволн, реальное покрытие может отличаться от теоретического. Помимо построения общей актуальной карты покрытия еще один немаловажным результатом мониторинга является построение локальных зон покрытия, соответствующих сотам. Подобная информация позволяет оценить соответствие реальной зоны покрытия, обеспечивающейся сотой, и теоретической зоны покрытия, что может поспособствовать обнаружению некорректности режима работы или расположения базовой станции.

Важным моментом является тот факт, что используемая для определения координат точки измерения система геопозиционирования GPS имеет погрешность, суммарное среднеквадратичное значение которой составляет 13,1 м [4]. Таким образом, под точкой измерения здесь понимается круглая область, радиус которой составляет 13,1 м, внутри которой находится реальная точка измерения.

Еще одним немаловажным замечанием является то, что все смартфоны, производящие измерения, в зависимости от марки могут иметь различные антенны и радиомодули, что непосредственно виляет на определение уровня принимаемого сигнала тем или иным смартфоном в одной и той же точке. Таким образом, значение измеренного уровня сигнала в точке измерения является случайной величиной, на которую влияют конструктивные особенности смартфона. Тогда, исходя из центральной предельной теоремы, которая утверждает, что сумма достаточно большого количества слабо зависимых случайных величин, имеющих примерно одинаковые масштабы (ни одно из слагаемых не доминирует, не вносит в сумму определяющего вклада), имеет распределение, близкое к нормальному, можно сделать вывод, что уровень сигнала в определенной точке, измеренный с помощью смартфона является случайной величиной, распределенной по нормальному закону [5]. Тогда выборочное среднее (выборкой в данном случае является совокупность результатов измерений) при точечной оценке математического ожидания нормально распределенного значения уровня сигнала можно считать состоятельной и несмещенной оценкой [6]. Итак, для оценки уровня сигнала в точке измерения, используется выборочное среднее, иначе говоря, среднее арифметическое совокупности результатов измерений уровня сигнала в этой точке.

При построении карт покрытия для наглядности градации уровня сигнала в той или иной области используются цвета, соответствующие ранговой оценке уровня сигнала в данной области.

Для того, чтобы по полученным данным построить карту покрытия с градацией уровня сигнала, предлагается использовать дискретную модель местности. Для этого на территорию, карта покрытия которой строится, накладывается сетка, разбивающая ее на одинаковые ячейки квадратной формы. Оптимальные размеры ячейки должны быть определены эмпирическим путем, однако, размеры ячейки должны быть соизмеримы с размерами области, в которой находится точка измерения, то есть круга с радиусом 13,1 м, для того, чтобы можно было сделать допущение о том, что в рамках квадрата уровень сигнала имеет примерно одинаковое значение, равное среднему значению уровней сигнала в точках, принадлежащих квадрату. В то же время размеры ячейки должен быть на порядок меньше размеров соты, чтобы возможно было отследить границы соты и изменение уровня сигнала внутри соты. Минимальный размер имеет сота сети типа UMTS (3G) в городских условия, для обеспечения оптимального режима работы ее радиус должен составлять в среднем 1–1,5 км. Исходя из описанных требований, размер дискретной ячейки карты покрытия определен как 30*30 м, однако, это предварительное допускаемое значение, которое, возможно, будет получено эмпирическим путем в дальнейшем для обеспечения эффективного построения карты покрытия.

Средний уровень сигнала, характеризующий ячейку, поддается ранговой оценке, в результате чего определяется соответствующий градации уровня сигнала цвет, в который на карте покрытия будет окрашена ячейка. Распределение точек измерения по дискретным ячейкам карты покрытия проиллюстрировано рис. 1. Так как точка измерения представляется в виде круглой области, достаточно велика вероятность попадания этой области на границу ячеек, в таком случае сложно сделать вывод о принадлежности точки к той или иной ячейке, поэтому делается допущение, что точка принадлежит сразу к нескольким ячейкам, и ее значение уровня сигнала в этой точке соответственно участвует в вычислении средних уровней всех ячеек, к которым она принадлежит.

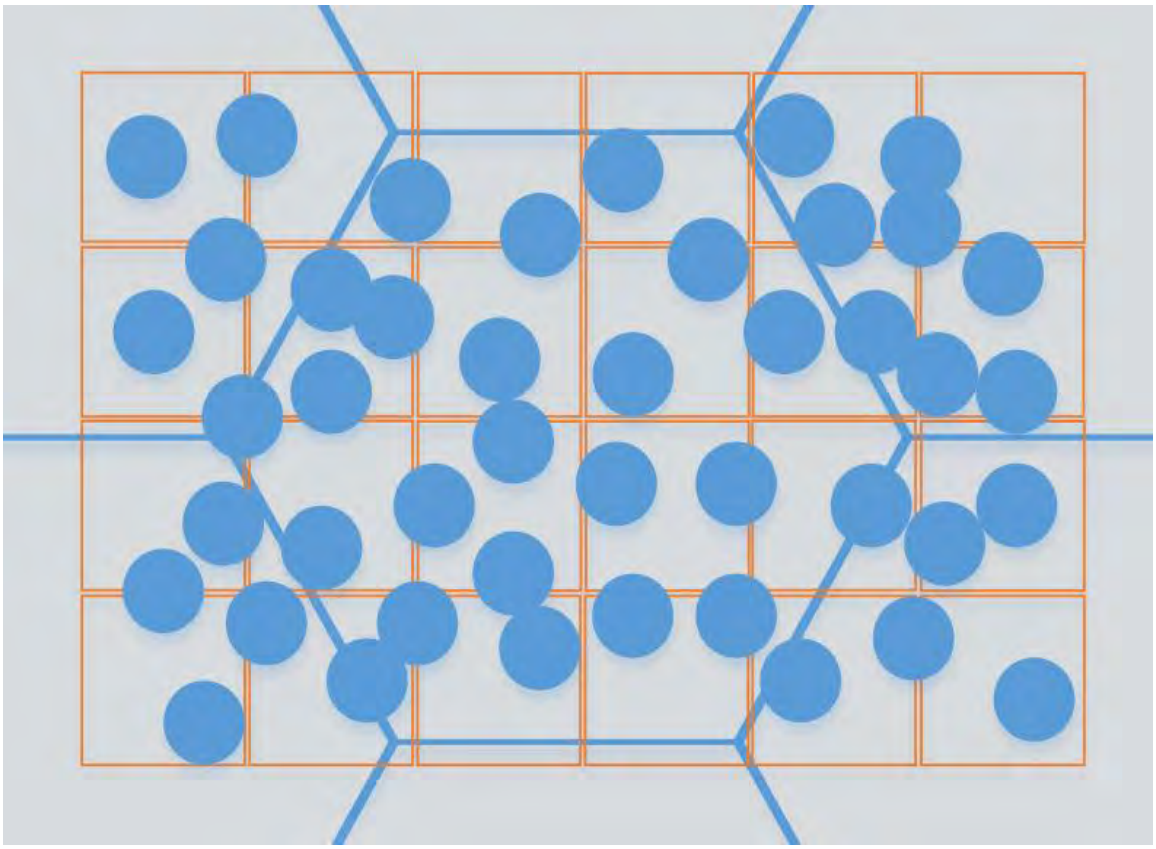


Рис. 1. Точки измерения технических параметров, распределенные
по дискретным ячейкам карты покрытия

Таким образом, реальная карта покрытия представляется как совокупность дискретных ячеек, с рассчитанными средними значениями уровня сигнала, а актуализация карты покрытия осуществляется за счет периодического обновления усредненных значений уровня сигнала в ячейках, полученных на основе новых данных измерений, присылаемых абонентами. В настоящий момент в рамках проекта программы «УМНИК» ведется разработка программного комплекса, использующего описанную методологию построения карт покрытия, и данная работа также написана в рамках этого проекта.

Библиографический список

1. Ханин, И. В. Информационно-измерительная система для мониторинга сетей мобильной связи : дис. ... канд. техн. наук (05.11.16. Пензенский государственный университет) / Ханин И. В. – Пенза, 2012.

2. Мартяшин, Г. В. Проблемы диагностики и мониторинга технических параметров сетей сотовой связи / Г. В. Мартяшин, И. С. Пышкина, А. В. Калачев // Взаимодействие науки и общества: проблемы и перспективы : сб. ст. Междунар. науч.-практ. конф. : в 4 ч. (г. Казань, 8 июня 2017 г.). – Казань, 2017. – Ч. 3. – С. 74–76.

3. Principles of cellular frequency reuse // JPL's Wireless Communication Reference Website [сайт]. – URL: http://www.wirelesscommunication.nl/reference/chaptr04/cellplan/reuse.htm (дата обращения: 16.10.2016).

4. Анучин, О. Н. Интегрированные системы ориентации для морских подвижных объектов / О. Н. Анучин, Г. И. Емельянцев ; под ред. В. Г. Пешехонова. – 2-е изд. – СПб. : ГНЦ РФ-ЦНИИ «Электроприбор», 2003. – С. 160–161.

5. Вентцель, Е. С. Теория вероятностей : учеб. для вузов / Е. С. Вентцель. – 6-е изд., стер. – М. : Высш. шк., 1999. – 576 с.

6. Левин, Б. Р. Теоретические основы статистической радиотехники / Б. Р. Левин. – 3-е изд., перераб. и доп. – М. : Радио и связь, 1989. – 656 с.

Г. В. Мартяшин, И. С. Пышкина

Пензенский государственный университет, Пенза, Россия

АНАЛИТИЧЕСКИЙ ПРОГРАММНЫЙ ИНСТРУМЕНТ ПЛАНИРОВАНИЯ ЭКСПЕРИМЕНТА

Обсуждается вопрос разработки составов строительных смесей путем планирования эксперимента с помощью специальных программных средств. Рассматривается разрабатываемое аналитическое программное средство, которое является более простой и удобной альтернативой существующим решениям.

В настоящий момент активно развивается сфера разработки отечественных строительных материалов, о чем можно судить и по достаточно большому количеству научных работ на подобные темы, включающих публикации, патенты и диссертации.

Во многом это определяется повышением спроса на строительные материалы по причине увеличения объемов строительства в России [1]. Наибольшее распространение на отечественном рынке имеют импортные строительные материалы или отечественными строительными материалами, которые основаны на импортных составляющих. В то же время, использование отечественных аналогичных строительных материалов, особенно в контексте импортозамещения, имеет ряд существенных преимуществ по сравнению с импортными. Во-первых, использование отечественных материалов делает рынок менее импортозависимым, укрепляя тем самым российскую экономику. Во-вторых, отечественные материалы обладают более низкой себестоимостью, за счет колоссального сокращения издержек на транспортировку при использовании местных сырьевых ресурсов.

Строительные материалы как правило имеют рецептуры, включающие большое количество компонентов, причем конченые свойства материала определяется соотноше-нием массовых долей компонентов, их химическим составом, режимом синтеза. Например, сухие строительные смеси (ССС), занимающие немалую часть рынка, имеют рецептуры, в которые входят вяжущие, заполнитель, минеральные, пластифицирующие, гидрофобизирующие и полимерные добавки [2].

Разработка строительных материалов, таким образом, подразумевает решение задачи поиска оптимального соотношения каких-либо количественных параметров для получения требуемых свойств конечного продукта. Такими параметрами могут быть, например, массовые доли компонентов смеси и температура твердения. Это означает, что подобного рода исследование опирается на эксперимент, как на основной метод. Так как проведение экспериментов требует дополнительных материальных и временных ресурсов (продолжительность эксперимента может достигать нескольких месяцев), количество необходимых экспериментов всегда требуется сводить к минимальному. Для этого прибегают к такому методу, как планирование эксперимента. Метод позволяет построить математическую модель зависимости между оптимизируемыми свойствами смеси и количественно выраженными факторами, а затем, после проведения минимального необходимого количества экспериментов, опровергнуть или же подтвердить и уточнить модель. Затем определяется оптимальное решение (оптимальные значения количественных факторов) на основе поиска экстремумов модели [3].

В настоящие время планирование эксперимента, как и любая обработка результатов экспериментов без использования компьютеров и пакетов прикладных программ практически не применяется в силу своей рутинности. Наиболее универсальным из подобных средств являются табличные процессоры с различными математическими надстройками и математические программные пакеты, например, MATLAB. Однако, в силу их универсальности подобных средства не ориентированы на конкретные сценарии, и их использование предполагает знание определенных особенностей и наличия некоторых навыков програм-

мирования, что делает их применение весьма проблематичным для тех исследователей, область научных интересов которых далека от информационных технологий. Этим продиктована необходимость разработки более простых в использовании узкоспециализированных средств [4].

В настоящий момент в этих целях на завершающих этапах находится разработка аналитического программного средства для планирования эксперимента и обработки результатов экспериментов, ориентированное для применения в сфере разработки строительных материалов. В качестве программной платформы для разработки выбрана .NET Framework.

Разрабатываемое программное средство может быть проиллюстрировано UML-диаграммой вариантов использования, которая представлена на рис. 1.

Рис. 1. Диаграмма вариантов использования

Из данной диаграммы видно, что присутствуют все прецеденты, необходимые для выполнения планирования эксперимента. В качестве начальных вариантов использования выступает «планирование эксперимента», «сохранение результатов планирования» и «загрузка сохраненных результатов планирования». Вариант использования «планирование эксперимента», соответствующий началу нового планирования эксперимента, связан отношениями включения и расширения с другими вариантами использования, относящимися к более глубокому уровню детализации модели. Так, «указание факторов и типа модели», «формирование матрицы планирования эксперимента», «ввод полученных экспериментальных значений» являются прецедентами, включаемыми в состав начала нового планирования эксперимента. Отношение включения подразумевает обязательное наличие данных вариантов использования в рамках начала нового планирования эксперимента, как неотъемлемых его частей. Варианты использования «расчет математических моделей» и «формирование графиков» связаны с началом нового планирования эксперимента отношением расширения, так как выступают как действия, выполняющиеся на последующих этапах планирования эксперимента, то есть в начале планирования являются необязательными.

Сохранение результатов может быть выполнено на любом этапе планирования и не связано с какими-либо отношениями с другими вариантам использования.

Загрузка сохраненных результатов планирования эксперимента расширяется такими вариантами использования, как «ввод полученных экспериментальных значений», «расчет математической модели» и «формирование графиков». Отношение расширения объясняется тем, что сохраненные результаты планирования могут соответствовать любому этапу

планирования, т.е. это может быть начальный этап, на котором только определены факторы и тип модели, и сгенерирована матрица планирования, так и завершающий этап, на котором уже построена математическая модель и оценена ее адекватность. Так сохраненные результаты планирования могут соответствовать разным этап, все дальнейшие варианты использования рассматриваются как необязательные.

Итак, типичный сценарий работы с разрабатываемым программным инструментом имеет следующий вид:

- начинается новое планирование эксперимента, после получения матрицы планирования результаты сохраняются;
- по мере выполнения экспериментов в соответствии с матрицей планирования периодически производится загрузка результатов, ввод полученных в результате экспериментов значений и сохранение результатов;
- после проведения всех необходимых экспериментов строится математическая модель и оценивается ее адекватность, а также производится формирование графиков, иллюстрирующих полученную математическую модель.

Таким образом, была рассмотрена проблема планирования эксперимента и обработки экспериментальных данных при разработке строительных материалов. Были предложено решение в виде аналитического программного средства, решающего поставленную проблему и удовлетворяющего поставленным требованиям. В дальнейшем планируется завершение разработки описанного в данной статье программного инструмента.

Библиографический список

1. Савченко, Е. Ю. Рынок строительных материалов: тенденции развития / Е. Ю. Савченко // Экономика, управление и право: инновационное решение проблем : сб. ст. победителей VI Междунар. науч.-практ. конф. / под общ. ред. Г. Ю. Гуляева. – Пенза : МЦНС «Наука и Просвещение», 2017. – С. 149–153.

2. Оптимизация режима синтеза добавки на основе гидросиликатов для известковых сухих строительных смесей / В. И. Логанина, И. С. Пышкина, С. А. Толушов, С. А. Болдырев // Региональная архитектура и строительство. – 2015. – № 4-1 (25). – С. 48–51.

3. Логанина, В. И. Разработка технологии синтеза добавки на основе гидросиликатов кальция для сухих строительных смесей / В. И. Логанина, И. С. Пышкина // Известия высших учебных заведений. Строительство. – 2014. – № 7 (667). – С. 46–49.

4. Мартяшин, Г. В. Программный комплекс планирования эксперимента и анализа экспериментальных данных / Г. В. Мартяшин, И. С. Пышкина // Новые информационные технологии и системы (НИТиС – 2015) : сб. науч. ст. XII Междунар. науч.-техн. конф. – Пенза : Изд-во ПГУ, 2015. – С. 63–65.

С. В. Тумасов, Е. И. Гурин

Пензенский государственный университет, Пенза, Россия

АНАЛИЗАТОР КАЧЕСТВА СВЯЗИ НА БАЗЕ ШУМОПОДОБНОГО СИГНАЛА

Представлено программно-аппаратное решение для оценки качества связи на базе шумоподобного сигнала.

Беспроводные технологии в настоящее время применяются во многих прикладных областях, таких как системы обеспечения безопасности и обороны, управления энергосбережением, охранные системы, мониторинг состояния человека. Особо важной становится проблема организации беспроводной сети в сложных условиях, например, в лесу. Одним из вариантов решения этой проблемы является применение технологии шумоподобного сиг-

нала. Она позволяет увеличить дальность связи по сравнению с узкополосными сигналами. При этом важно выполнить оценку качества связи [1].

В системах LPWAN (энергоэффективная сеть дальнего радиуса действия) широко применяется стандарт LoRaWAN. Технология LoRa реализует физический уровень системы, представляет собой схему модуляции с расширением спектра, в которой данные кодируются широкополосными ЛЧМ-импульсами с частотой, увеличивающейся или уменьшающейся на некотором временном интервале (метод CSS). Метод CSS обеспечивает больший бюджет канала, защищен от эффектов многолучевого распространения, эффекта Допплера. При этом в LoRa применяются коды Грея, расширение спектра путем выполнения операции «ИСКЛЮЧАЮЩЕЕ ИЛИ» над данными и псевдослучайной последовательностью, используются коды Хэмминга прямой коррекции ошибки [2].

В настоящей работе рассматривается анализатор качества связи для LoRa. Он представляет собой макеты передатчика и приемника LoRa. Макет приемника имеет в своем составе одноплатный компьютер Raspberry Pi, приемник/передатчик на базе SX1278, GPS-приемник, LCD-экран. Макет передатчика включает в себя одноплатный компьютер Raspberry Pi, приемник/передатчик на базе SX1278. Было разработано программное обеспечение для передатчика и приемника, реализующее функционал анализатора качества связи. Структурная схема макетов приемника и передатчика представлена ниже.

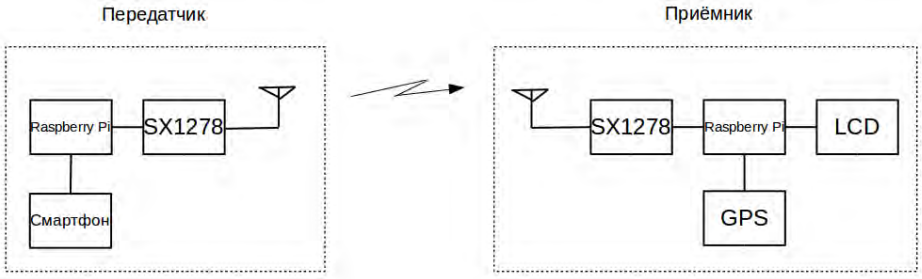


Рис. 1. Структурная схема макетов приемника и передатчика

В качестве используемого диапазона для работы рассматривались LPD433 (433 МГц), 868 (UHF)(868 МГц), 2400(S) (2400 МГц). Был выбран диапазон LPD433, потому что он обеспечивает наибольшую возможную дальность работы при одной и той же мощности [3].

В ходе эксперимента передатчик создавал точку доступа Wi-Fi. К этой точке доступа подключался смартфон. Далее через терминал устанавливалась требуемая длина псевдослучайной последовательности, скорость передачи [4]. Для экспериментов использовались две комбинации параметров:

1) для передачи SMS устанавливалась комбинация: 7,8 КГц – скорость передачи, 4096 чипов – длина псевдослучайной последовательности;

2) для передачи голоса устанавливалась комбинация: 500 КГц – скорость передачи, 128 чипов – длина псевдослучайной последовательности.

После чего Raspberry Pi в макете передатчика формировал пакеты, а передатчик/приемник на базе SX1278 передавал их. В макет приемника вводились такие же установки с помощью LCD-экрана с тачскрином. Если параметры введены верно, приемник обнаруживал передаваемый от передатчика сигнал. Для анализа качества связи было решено подключить к Raspberry Pi в макете приемника GPS-приемник для записи треков. GPS-приемник отправляет координаты местоположения макета приемника каждую секунду Raspberry Pi. При успешном получении пакета от передатчика в макете приемника Raspberry Pi записывает координаты в лог-файл в определенном формате. В ходе эксперимента этот лог-файл заполняется значениями. Для каждой комбинации параметров создается собственный лог-файл. После эксперимента полученный лог-файл транслируется в kml-файл, который позволяет посмотреть трек в программе Google Earth. Треки анализируются и выбирается максимальная возможная дальность связи.

Фотография макетов передатчика и приемника представлена ниже.

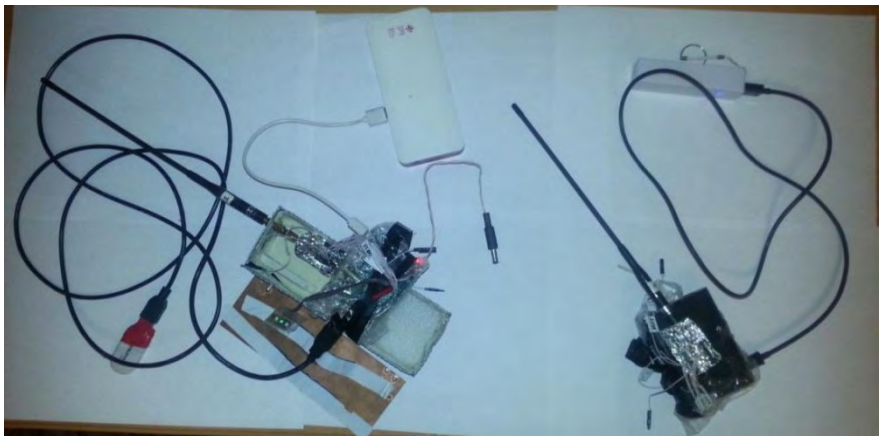


Рис. 2. Макеты приемника и передатчика

Эксперимент проводился в лесистой местности. На передатчике запускалась передача и он находился в статическом состоянии, а приемник перемещался и записывал логфайл. При комбинации параметров для передачи SMS максимальная возможная дальность связи составила 758 м, а для голоса она составила 342 м.

Предложенные авторами программно-аппаратные решения позволят проанализировать качество связи на базе шумоподобного сигнала, измеряя максимально возможную дальность связи.

Библиографический список

1. Тумасов, С. В. Макет сенсорной сети с использованием технологии SDR / С. В. Тумасов, Е. К. Таранцев // Информационные технологии в науке и образовании : сб. ст. II Междунар. науч.-техн. конф. – Пенза : Изд-во ПГУ, 2015. – С. 304–308.

2. Matt Knight Reversing LoRa : Exploring next-generation wireless. – Bastille Networks, 2016. – 128 с.

3. Таранцев, Е. К. Сеть радиомаяков с расширенным спектром / Е. К. Таранцев, С. В. Тумасов // XXI век: итоги прошлого и проблемы настоящего плюс. – 2016. – № 3 (31). – С. 161–166.

4. Miguel, Elias M. Campista, Marcelo G. Rubinstein Advanced Routing Protocols for Wireless Networks / Miguel Elias M. Campista, Marcelo G. – UK : Wiley, 2014.

К. И. Никишин

Пензенский государственный университет, Пенза, Россия

АППАРАТНАЯ КОНФИГУРАЦИЯ ПЛИС ТИПА FPGA С ИСПОЛЬЗОВАНИЕМ ИНТЕРФЕЙСА ETHERNET ДЛЯ ОБМЕНА ИНФОРМАЦИЕЙ

Рассматривается аппаратная конфигурация ПЛИС типа FPGA с использованием интерфейса Ethernet для обмена информацией на основе системы на кристалле (System on a Chip) в среде Xilinx. Конфигурация ПЛИС использует процессорное ядро MicroBlaze, через которое подключаются все необходимые периферийные устройства.

В настоящее время технология Ethernet является самой распространенной технологией в мире. При разработке новых типов конфигурации Ethernet, новых алгоритмов дис-

петчеризации [1] в коммутаторе большое внимание уделяется к параметрам быстродействия коммутатора, минимизации аппаратных затрат, легкости встраиваемости системы.

Для корректной отладки интерфейса Ethernet используют различные программные пакеты, позволяющие моделировать различные аспекты поведения сложных телекоммуникационных систем [2]. К такому программному пакету относится CPN Tools [3]. Для проверки реализованной модели в CPN Tools на аппаратуре можно использовать программируемые логические интегральные схемы (ПЛИС) типа FPGA в среде Xilinx [4].

Конфигурация ПЛИС использует процессорное ядро *MicroBlaze*, через которое подключается все необходимые периферийные устройства. В данном случае подключается интерфейс Ethernet для обмена информацией на основе системы на кристалле (System on a Chip) [5]. В статье используется отладочная плата Spartan 3E Starter Kit (рис. 1) для проверки аппаратных и программных особенностей [6].

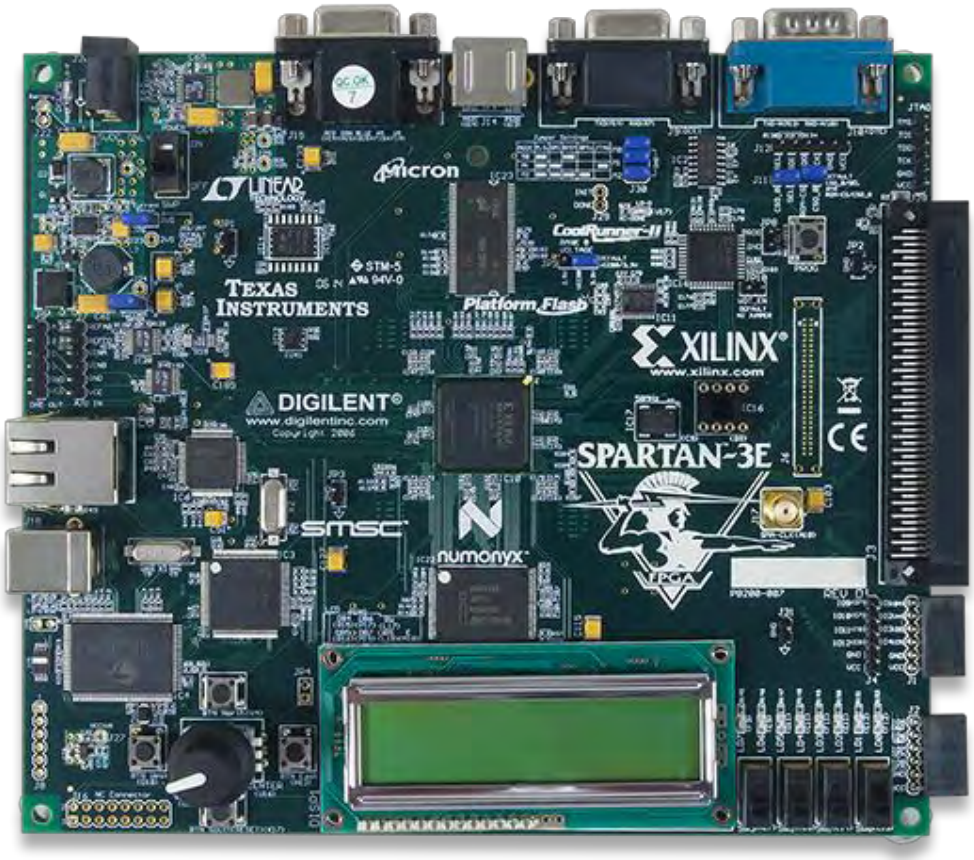


Рис. 1. Отладочная плата Spartan 3E Starter Kit

При создании проекта в XPS (Xilinx Platform Studio) задавался следующий тип микросхемы ПЛИС семейства Spartan 3E: *XC3S500E-4FG320*, полярность сброса (Reset polarity – Active HIGH), выбиралось процессорное ядро *MicroBlaze* и настраивались его необходимые характеристики (в частности частота процессорного ядра). В проекте использовались следующие внешние периферийные устройства: переключатели DIP_Switches на 4 разряда, светодиоды LEDs на 8 разрядов, интерфейс RS232 для вывода информации через COM-порт и интерфейс Ethernet для проверки обмена информации.

После создания проекта должны создаться основные аппаратные спецификации для обеспечения работы отладочной платы. Спецификация аппаратной платформы встраиваемой микропроцессорной системы MHS (*Microprocessor Hardware Specification*) представляет собой файл с расширением MHS, в котором в текстовом формате описывается конфигурация и параметры аппаратных средств проектируемой системы [7].

Дополнительная информация, используемая программами синтеза, размещения и трассировки проекта аппаратной части микропроцессорной системы, задается в форме файла временных и топологических ограничений *User Constraints File* (UCF) [8]. После окончания работы мастера BSB на экране появляется основное окно XPS (Xilinx Platform Studio), верхняя часть которого показана на рис. 2.

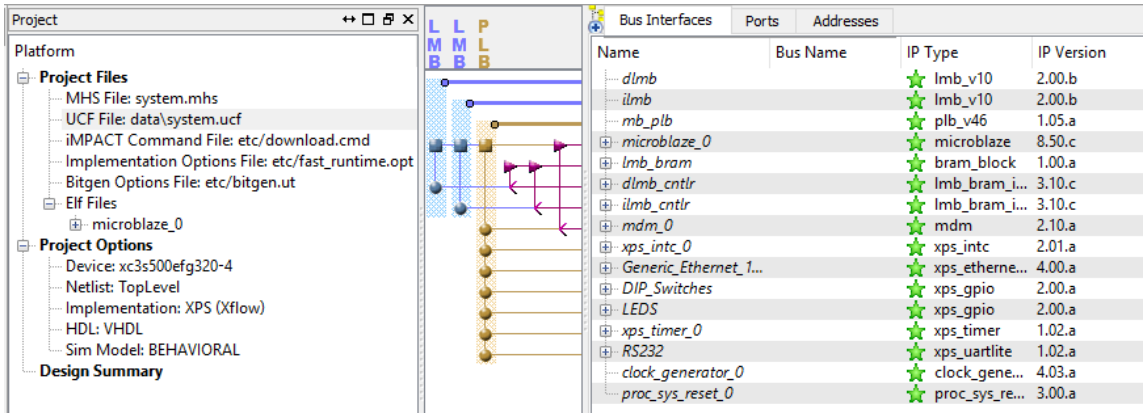


Рис. 2. Основное окно XPS (Xilinx Platform Studio)
после установки необходимых параметров проекта

Компонент интерфейса Ethernet, включаемый в состав спецификации аппаратной платформы, выглядит следующим образом:

BEGIN xps_ethernetlite
PARAMETER INSTANCE = Generic_Ethernet_10_100
PARAMETER HW_VER = 4.00.a
PARAMETER C_BASEADDR = 0x81000000
PARAMETER C_HIGHADDR = 0x8100ffff
BUS_INTERFACE SPLB = mb_plb
PORT PHY_tx_clk = fpga_0_Generic_Ethernet_10_100_PHY_tx_clk_pin
PORT PHY_rx_clk = fpga_0_Generic_Ethernet_10_100_PHY_rx_clk_pin
PORT PHY_crs = fpga_0_Generic_Ethernet_10_100_PHY_crs_pin
PORT PHY_dv = fpga_0_Generic_Ethernet_10_100_PHY_dv_pin
PORT PHY_rx_data = fpga_0_Generic_Ethernet_10_100_PHY_rx_data_pin
PORT PHY_col = fpga_0_Generic_Ethernet_10_100_PHY_col_pin
PORT PHY_rx_er = fpga_0_Generic_Ethernet_10_100_PHY_rx_er_pin
PORT PHY_rst_n = fpga_0_Generic_Ethernet_10_100_PHY_rst_n_pin
PORT PHY_tx_en = fpga_0_Generic_Ethernet_10_100_PHY_tx_en_pin
PORT PHY_tx_data = fpga_0_Generic_Ethernet_10_100_PHY_tx_data_pin
PORT PHY_MDC = fpga_0_Generic_Ethernet_10_100_PHY_MDC_pin
PORT PHY_MDIO = fpga_0_Generic_Ethernet_10_100_PHY_MDIO_pin
END

Значение базового адреса компонента определяется с помощью параметра *C_BASEADDR*. Верхняя граница адресного пространства памяти, используемого компонентом, указывается в виде значения параметра *C_HIGHADDR*. Диапазон адресов можно настроить и на вкладе проекта адресов «*Addresses*». Пользователь может сам установить необходимый диапазон вручную или нажать на кнопку автоматической генерации адресов «*Generate Addresses*».

Внешние порты компонента: *PHY_tx_clk* – порт синхросигнала передатчика; *PHY_rx_clk* – порт синхросигнала приемника; *PHY_crs* – порт чувствительности; *PHY_dv* – порт корректности принимаемой информации; *PHY_rx_data* – порт принимаемой информации; *PHY_col* – порт определения коллизий; *PHY_rx_er* – порт ошибки приема; *PHY_rst_n* – порт сброса; *PHY_tx_en* – порт разрешения передачи; *PHY_tx_data* – порт передаваемой

информации; *PHY_MDC* – синхросигнал управления; *PHY_MDIO* – управление входной/выходной информацией.

Важной частью проекта является файл ограничений, который позволяет установить соответствие между внешними цепями проектируемой системы и номерами выводов ПЛИС. Для создания файла временных и топологических ограничений и внесения в него информации можно использовать встроенный текстовый редактор управляющей оболочки Xilinx Platform Studio.

Ограничения для интерфейса Ethernet выглядят следующим образом:

Net fpga_0_Generic_Ethernet_10_100_PHY_tx_clk_pin LOC=T7;
Net fpga_0_Generic_Ethernet_10_100_PHY_rx_clk_pin LOC=V3;
Net fpga_0_Generic_Ethernet_10_100_PHY_crs_pin LOC=U13;
Net fpga_0_Generic_Ethernet_10_100_PHY_dv_pin LOC=V2;
Net fpga_0_Generic_Ethernet_10_100_PHY_rx_data_pin<0> LOC=V8;
Net fpga_0_Generic_Ethernet_10_100_PHY_rx_data_pin<1> LOC=T11;
Net fpga_0_Generic_Ethernet_10_100_PHY_rx_data_pin<2> LOC=U11;
Net fpga_0_Generic_Ethernet_10_100_PHY_rx_data_pin<3> LOC=V14;
Net fpga_0_Generic_Ethernet_10_100_PHY_col_pin LOC=U6;
Net fpga_0_Generic_Ethernet_10_100_PHY_rx_er_pin LOC=U14;
Net fpga_0_Generic_Ethernet_10_100_PHY_tx_en_pin LOC=P15;
Net fpga_0_Generic_Ethernet_10_100_PHY_tx_data_pin<0> LOC=R11;
Net fpga_0_Generic_Ethernet_10_100_PHY_tx_data_pin<1> LOC=T15;
Net fpga_0_Generic_Ethernet_10_100_PHY_tx_data_pin<2> LOC=R5;
Net fpga_0_Generic_Ethernet_10_100_PHY_tx_data_pin<3> LOC=T5;

Таким образом, видно соответствие наименований портов в спецификации аппаратной платформы и в файле ограничений. В файле ограничений указывается еще и разрядность, порты принимаемой/передаваемой информации являются четырехразрядными.

В статье рассмотрена аппаратная конфигурация ПЛИС типа FPGA с использованием интерфейса Ethernet для обмена информацией на основе системы на кристалле (System on a Chip) в среде Xilinx. Все основные компоненты (периферийные устройства) ПЛИС подключались через процессорное ядро *MicroBlaze*. Рассмотрены спецификация аппаратной платформы (MHS) и файл временных и топологических ограничений (UCF) для интерфейса Ethernet на плате Spartan 3E Starter Kit.

Библиографический список

1. Никишин К. И. Механизм управления трафиком реального времени в коммутаторе Ethernet / К. И. Никишин // Вестник компьютерных и информационных технологий. – 2015. – Вып. 10. – С. 32–37.

2. Scheduling queues in the Ethernet switch, considering the waiting time of frames / Evgeniy Kizilov, Nikolas Konnov, Kirill Nikishin, Dmitry Pashchenko, Dmitriy Trokoz // MATEC Web of Conferences . – 2016. – Vol. 44. – P. 01011-p.1-01011-p.

3. Описание программы CPN Tools. – URL: http://cpntools.org/

4. Описание работы с программными средами фирмы Xilinx. – URL: http://xilinx.com/

5. Никишин, К. И. Разработка выходного порта коммутатора Ethernet на базе ПЛИС / К. И. Никишин // Информационные технологии. Проблемы и решения : сб. материалов III Междунар. науч.-практ. конф. – Уфа : Восточная печать, 2015. – Т. 2. – С. 201–204.

6. Руководство к плате Spartan-3E Start Kit Board User Guide UG230 2006.

7. Зотов, В. Формирование спецификации аппаратной платформы микропроцессорной системы на основе ядра MicroBlaze, реализуемой в ПЛИС семейств FPGA фирмы Xilinx / В. Зотов // Компоненты и технологии. – 2004. – № 7-8.

8. Зотов, В. Реализация аппаратной платформы микропроцессорной системы, проектируемой на основе ядра MicroBlaze, в ПЛИС семейств FPGA фирмы Xilinx / В. Зотов // Компоненты и технологии. – 2004. – № 9.

К. И. Никишин

Пензенский государственный университет, Пенза, Россия

ОБМЕН ИНФОРМАЦИЕЙ С ИСПОЛЬЗОВАНИЕМ ИНТЕРФЕЙСА ETHERNET НА ОСНОВЕ ПЛИС ТИПА FPGA

Рассматривается обмен информацией с использованием интерфейса Ethernet на основе программируемых логических интегральных схем ПЛИС типа FPGA. Интерфейс Ethernet подключается через процессорное ядро MicroBlaze.

В настоящее время технология Ethernet является самой распространенной технологией в мире. Большое внимание при разработке улучшений Ethernet уделяется быстродействию, повышению качества обслуживания сети. Моделирование всей системы Ethernet позволяет избежать многих ошибок и выявить закономерности в обработки системы [1, 2]. Используются различные алгоритмы диспетчеризации очередей в коммутаторе Ethernet и выявляют особенности обработки трафика [2].

Конфигурация ПЛИС использует процессорное ядро *MicroBlaze*, через которое подключается все необходимые периферийные устройства. В данном случае подключается интерфейс Ethernet для обмена информацией на основе системы на кристалле (System on a Chip) [3]. В статье используется отладочная плата Spartan 3E Starter Kit для проверки аппаратных и программных особенностей [4].

Обмен данными по каналу Ethernet осуществляется между ПК и отладочной платой Spartan 3E Starter Kit. Для работы платы Spartan 3E Starter Kit на уровне интерфейса Ethernet в системе Xilinx SDK был создан проект для данной схемы, в котором помимо стандартных периферийных устройств был подключен контроллер Ethernet [5]. Формат кадра Ethernet представлен на рис. 1.

Рис. 1. Кадр Ethernet

Рассмотрим реализацию обмена данных с использованием интерфейса Ethernet в среде SDK [6–7]. Вначале работы с контроллером Ethernet устанавливается указатель на драйвер контроллера.

*XEmacLite *EmacLiteInstancePtr = &EmacLiteInstance;*

После установления указателя на котроллер необходимо проинициализировать Ethernet. За инициализацию отвечает функция *XEmacLite_Initialize*. Функция возвращает статус успешности или неудачи инициализации контроллера.

*int XEmacLite_Initialize(XEmacLite *InstancePtr, u16 DeviceId)*, где *InstancePtr* – указатель на контроллер, *DeviceId* – идентификатор устройства.

После инициализации контроллера необходимо установить MAC-адрес отладочной платы через функцию *XEmacLite_SetMacAddress*, входными параметрами функции являются указатель на контроллер и необходимое значение MAC-адреса.

После выполняется функция очистки буферов приема. Через функцию *XEmacLite_TxBufferAvailable* определяются, заполнены ли буферы передачи и доступны ли

они для дальнейшей обработки. Если возвращается истинное значение, значит, отладочная плата успешно приняла информацию от ПК.

Функция получения кадра (поступившей информации) выглядит следующим образом:

*u16 XEmacLite_Recv(XEmacLite *InstancePtr, u8 *FramePtr)*, где первым параметром выступает указатель на контроллер, второй параметр – указатель на буфер, который будет хранить кадр. Функция возвращает 0 в случае, если нет информации в буфере.

Для двустороннего взаимодействия программа работы интерфейса Ethernet была сделана таким образом, что при получении кадра по адресу 169.254.255.255 на ПК отправлялся ответный пакет, подтверждающий прием команды PING. Для фиксации входящего трафика использовался сниффер Microsoft Network Monitor.

```
if((*(Xuint8 *)(RxFrame + 30) == 169) &&
(*(Xuint8 *)(RxFrame + 31) == 254) &&
(*(Xuint8 *)(RxFrame + 32) == 255) &&
(*(Xuint8 *)(RxFrame + 33) == 255)) {}
```

После успешного сравнения полученного ответа по заданному адресу выполняется алгоритм отправки кадра. Сначала устанавливается адрес назначения (адрес ПК), затем записывается адрес источника (адрес отладочной платы) в кадр передачи. Устанавливается поле длины кадра и затем заполняется поле данных кадра передачи. После установки всех необходимых полей кадра передачи вызывается функция *XEmacLite_Send*.

*int XEmacLite_Send(XEmacLite *InstancePtr, u8 *FramePtr, unsigned ByteCount)*, где *InstancePtr* - указатель на контроллер, *FramePtr* – указатель на буфер передачи, *ByteCount* – размер кадра в байтах.

После отправки команды PING по сети передавались пакеты. Во время обработки данных, осуществлялась отправка ответного сообщения. Об отправке пакета данных свидетельствует мигание диода выходного интерфейса Ethernet.

Принятые пакеты фиксируются сниффером Microsoft Network Monitor (рис. 2). Как видно из рис. 2 в сеть отправляется сообщение с адресом 169.254.255.255 в ответ, на которое приходит сообщение, отправленное отладочной платой.

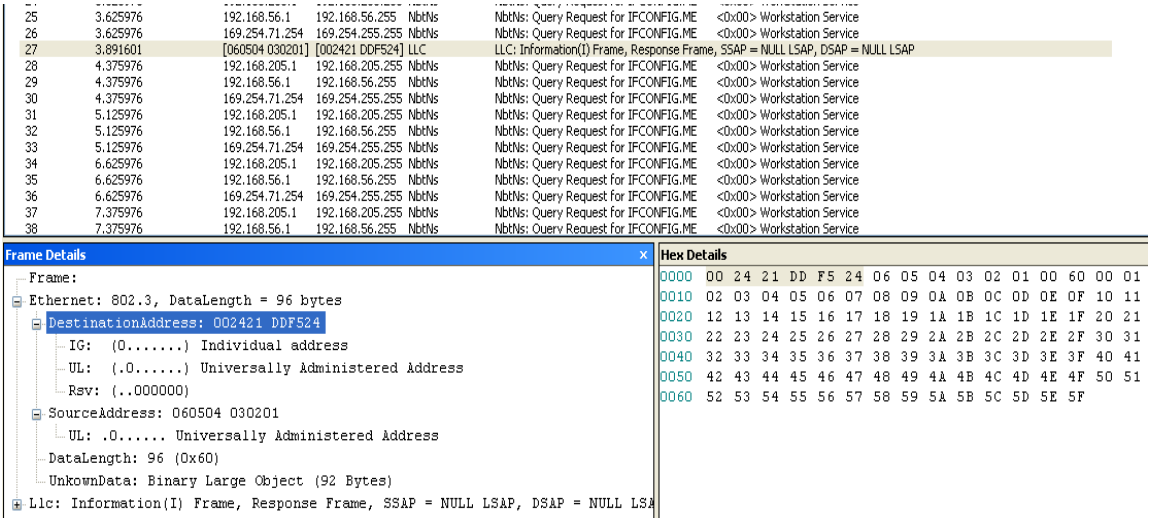


Рис. 2. Входящий трафик

В статье рассмотрен программный обмен информацией на основе системы на кристалле (System on a Chip) с использованием интерфейса Ethernet в среде Xilinx. Интерфейс Ethernet подключался через процессорное ядро MicroBlaze. Рассмотрены основные функции обработки интерфейса Ethernet, функции работы с кадрами принятой и отправленной информации на плате Spartan 3E Starter Kit.

Библиографический список

1. Scheduling queues in the Ethernet switch, considering the waiting time of frames / Kizilov Evgeniy, Konnov Nikolas, Nikishin Kirill, Pashchenko Dmitry, Trokoz Dmitriy // MATEC Web of Conferences. – 2016. – Vol. 44. – P. 01011-p.1-01011-p.
2. Никишин, К. И. Механизм управления трафиком реального времени в коммутаторе Ethernet / К. И. Никишин // Вестник компьютерных и информационных технологий. – 2015. – Вып. 10. – С. 32–37.
3. Никишин, К. И. Разработка выходного порта коммутатора Ethernet на базе ПЛИС / К. И. Никишин // Информационные технологии. Проблемы и решения : сб. материалов III Междунар. науч.-практ. конф. – Уфа : Восточная печать, 2015. – Т. 2. – С 201–204.
4. Руководство к плате Spartan-3E Start Kit Board User Guide UG230 2006.
5. Описание работы с программными средами фирмы Xilinx. – URL: http://xilinx.com/
6. Зотов, В. Разработка программного обеспечения микропроцессорной системы на основе ядра MicroBlaze, реализуемой в ПЛИС семейств FPGA фирмы Xilinx / В. Зотов // Компоненты и технологии. – 2005. – № 2.
7. Зотов, В. Разработка встраиваемых микропроцессорных систем на основе ядра MicroBlaze, реализуемых в ПЛИС семейств FPGA фирмы Xilinx, с помощью «мастера» Base System Builder Wizard / В. Зотов // Компоненты и технологии. – 2005. – № 3-4.

А. О. Семенов

Пензенский государственный университет, Пенза, Россия

УСТРОЙСТВО, РЕАЛИЗУЮЩЕЕ АЛГОРИТМ ДИСПЕТЧЕРИЗАЦИИ НА БАЗЕ АДАПТИВНЫХ RAND-ПРИНЦИПОВ

Приводится описание особенностей построения модели алгоритма стохастической диспетчеризации очередей в составе сетевых коммутаторов с использованием сетей Петри.

В настоящей работе предлагается описание структуры сетей Петри, реализующих различные модификации стохастического алгоритма диспетчеризации, основанного на стохастическом подходе к выбору очередей на обслуживание с адаптацией к длинам очередей и текущему трафику в отдельных классах QoS.

При разработке устройства, обеспечивающего работу заданного алгоритма диспетчеризации, в составе сетевого коммутатора следует разработать, прежде всего, архитектуру построения коммутатора, в том числе исходя из понятий модульности, расширяемости и др. [1–4] С этой целью в качестве инструмента создания модели сетевого коммутатора взято средство создания иерархических сетей Петри.

В работе [2] рассматривалась разновидность алгоритма, на базе принципов хаотического выбора очередей, при котором каждая очередь имеет пакеты различной длины. Также был предложен вариант имитационной модели устройства, моделирующего стохастический алгоритм. В рамках данной работ

В состав устройства стохастической диспетчеризации очередей сетевого коммутатора с поддержкой качества обслуживания, содержит: генератор псевдослучайных чисел; узел хранения постоянных величин (ПЗУ); узел подсчета вероятностей выбора очередей на обслуживание Узел сравнения величин; узел подсчета номера обслуживаемой очереди. На рис. 1 показана подсеть диспетчера, принимающая на свои входы длины очередей, а на выходы поступает единственная метка обслуживания очереди.

В состав модели подсети, изображенной на рис. 1 входят узел сбора (надзора) входной информации (1, 3), генератор псевдослучайных чисел (2, 11); узел хранения постоянных величин (ПЗУ) (4); узел подсчета вероятностей выбора очередей на обслуживание (5, 6), узел сравнения величин (7–13); узел подсчета номера обслуживаемой очереди (14, 15).

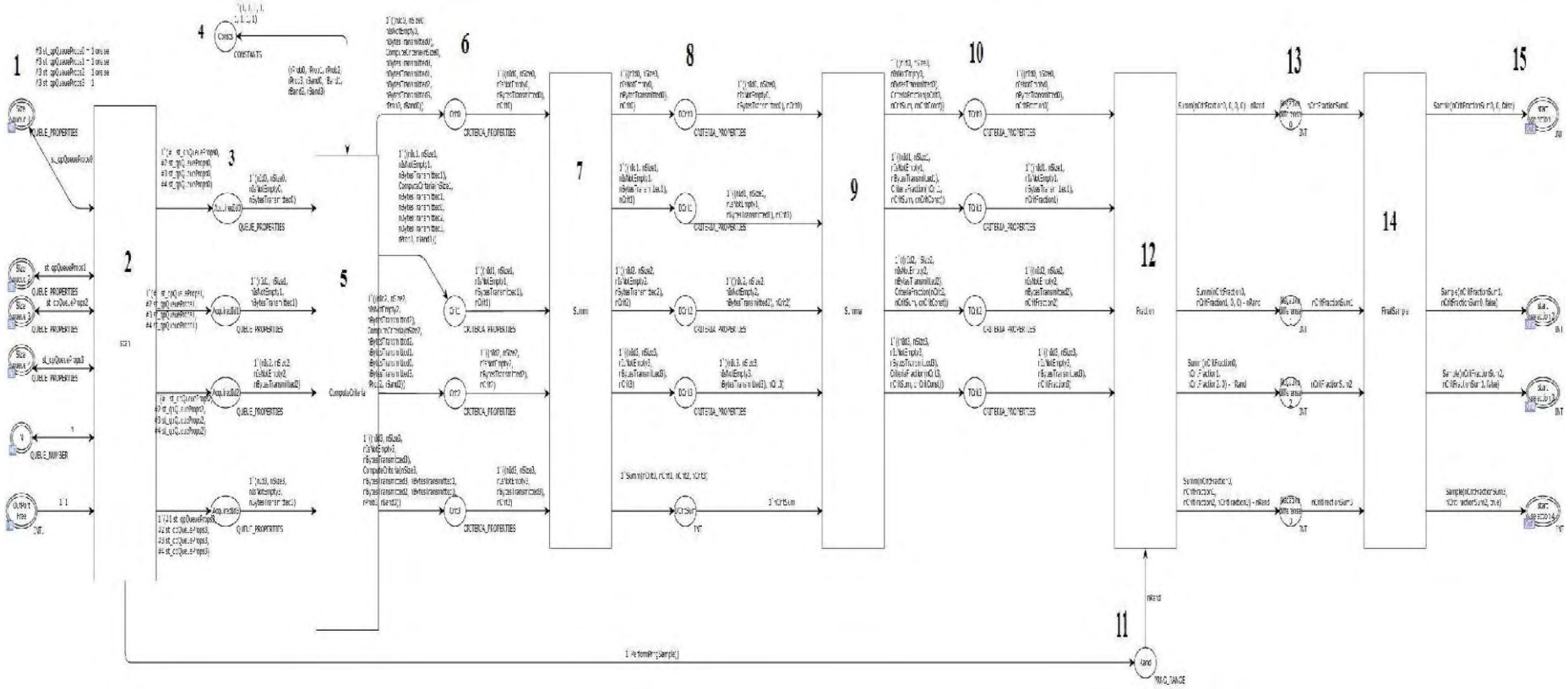


Рис. 1. Подсеть Петри, реализующая стохастической выбор очереди

Устройство стохастической диспетчеризации в составе коммутатора или маршрутизатора предназначено для непосредственного управления и надзора за процессом распределения канального времени между разными очередями. Узел надзора осуществляет контроль над выполнением процесса обработки и продвижения кадров кадровым обработчиком. Если обслуживание одного из классов трафика не укладывается по времени в нормативы, определяемые как количество квантов времени, выделяемых узлом надзора на обслуживание и передачу в канал из передаточной очереди кадра заданного размера, то такое обслуживание заканчивается узлом надзора принудительно. Таким образом, реализуется одно из базовых правил массового обслуживания.

Процесс расчета номера обслуживаемой очереди является циклическим, запускается блоком выборки кадра (1, 2). Передача аргументов для подсчета вероятностей осуществляется из передаточных очередей заголовков и указателей кадрового обработчика к подсети стохастической диспетчеризации. Узел надзора проверяет аргументы на корректность и разрешает или запрещает расчет. Подсчет вероятностей для каждой из очередей осуществляется параллельно и независимо от подсчета вероятностей выбора других очередей. Однако, в соответствии с предлагаемым способом, происходит подсчет сумм вероятностей (кумулятивных вероятностей). Этот подсчет также ведется в узле подсчета вероятностей выбора очередей на обслуживание.

Узел хранения постоянных величин управляющего блока содержит некоторые числовые константы, предназначенные для расчетов вероятностей, такие как: выделяемые доли обслуживания под каждый тип трафика, приоритеты по умолчанию (начальные значения вероятностей), семя генератора псевдослучайных чисел.

Генератор псевдослучайных чисел в процессе инициализации получает семя из узла хранения постоянных величин, порождает случайное число через фиксированное число тактов часов так, чтобы на каждый цикл диспетчеризации приходилось не менее одного вычисление псевдослучайного числа. Порожденная последовательность псевдослучайных чисел подчиняется равномерному распределению. Любое число из этой последовательности обладает неотрицательным значением, меньшим единицы.

Узел сравнения величин сравнивает значения, подсчитанные узлом подсчета вероятностей выбора очереди на обслуживание, т.е. кумулятивные вероятности, и генератором случайной величины. Для каждого типа трафика ставится либо значение истины или значение лжи в зависимости от разности значений.

Узел подсчета номера обслуживаемой очереди на основе данных сравнения, полученных от узла сравнения величин, производит выдачу обрабатывающим узлам, схеме выборки кадра обрабатывающего блока, а также узлу надзора сигналов разрешения или запрета обслуживания по каждому из классов обслуживания в соответствии с алгоритмом, представленным на рис. 1.

Приведенный вариант реализации стохастического алгоритма может быть применен не только для диспетчеризации очередей в коммутаторах, но и в других телекоммуникационных устройствах второго и третьего уровней сетевой модели.

Библиографический список

1. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы / В. Г. Олифер, Н. А. Олифер. – 4-е изд. – СПб. : Питер, 2010. – С. 943.

2. Семенов, А. О. Формальное представление стохастического алгоритма диспетчеризации очередей в телекоммуникационном оборудовании / А. О. Семенов, Н. Н. Коннов // Новые информационные технологии и системы (НИТиС-2014) : сб. науч. ст. XI Междунар. науч.-практ. конф. (г. Пенза, 25–27 ноября 2014 г.). – Пенза : Изд-во ПГУ, 2014. – С. 301–306.

3. Вегешна, Ш. Качество обслуживания в сетях IP / Ш. Вегешна. – М. : Вильямс, 2003. – С. 368.

4. Кучерявый, Е. А. Управление трафиком и качество обслуживания в сети Интернет / Е. А. Кучерявый. – СПб. : Наука и Техника, 2004. – С. 336.

А. О. Семенов, Н. Н. Коннов

Пензенский государственный университет, Пенза, Россия

МОДЕЛИРОВАНИЕ АЛГОРИТМА СТОХАСТИЧЕСКОЙ ДИСПЕТЧЕРИЗАЦИИ ОЧЕРЕДЕЙ В ПАКЕТЕ

Приводится описание особенностей некоторых временных характеристик модели алгоритма стохастической диспетчеризации очередей в составе сетевых коммутаторов с использованием сетей Петри.

В настоящей работе предлагаются временные характеристики сетей Петри, реализующих различные модификации стохастического алгоритма диспетчеризации, основанного на стохастическом подходе к выбору очередей на обслуживание с адаптацией к длинам очередей и текущему трафику в отдельных классах QoS [1– 4].

В данной работе рассматриваются различные временные характеристики для трѐх алгоритмов: дефицитный циклический (DRR), стохастический с возвратом (St1), стохастический без возврата (St2) [5].

Эффективность алгоритмов была исследована средствами инструментария цветных сетей Петри в программном пакете CPN Tools. В качестве входных данных на входы модели коммутаторов, содержащих три различных исполнения устройств диспетчеризации, подавались два различных аргумента: трафик, согласованный с выделенными полосами, выделенными под каждую совокупность классов обслуживания; и трафик, несогласованный с выделенными полосами, выделенными под каждую совокупность классов обслуживания. В рамках данной работы исследовалась только зависимость дисциплины выбора следующей очереди на обслуживание. При этом, количество выбираемых за одну выборку дисциплиной обслуживания бит не рассматривалось и имело квотный характер, как в алгоритме DRR. Отношения квот соответствуют k_1 / k_2/ k_3/ k_4=4/3/2/1. Выборка каждой очереди, при этом, равновероятна (p_1/p_2/p_3/p_4 = 1/1/1/1).

В качестве исследуемых величин использовались средняя задержка за всѐ время моделирования по всем классам обслуживания (см. строку «Ср задержка» в табл. 1); средняя задержка по каждой совокупности классов обслуживания (0-1; 2 3; 4 5; 6 7); размах средних задержек по каждому классу обслуживания (Размах); максимальная задержка (Макс задержка); максимальная задержка по каждому классу обслуживания (0 1; 2 3; 4 5; 6 7); размах максимальной задержки по классам обслуживания (Размах макс задержки); стандартное отклонение задержки по всем классам обслуживания (Стд откл); стандартное отклонение по каждому классу обслуживания (0 1; 2 3; 4 5; 6 7); размах стандартного отклонения по классам обслуживания (Размах стд откл); а также не количественные, а относительные показатели разброса величины задержки кадров коммутатором, а именно: отношение значения стандартного отклонения к среднему значению задержки кадров по всем классам обслуживания (Отношение откл к ср зад); отношение значения стандартного отклонения к среднему значению задержки кадров по каждой совокупности классов обслуживания (0 1; 2 3; 4 5; 6 7); размах отношений значений стандартного отклонения к среднему значению задержки кадров по всем классам обслуживания (Размах отн).

В результате исследования выяснилось, что стохастические алгоритмы показывают лучшие в сравнении с циклическим статистические и временные характеристики при моделировании, использующем в качестве аргументов данные несогласованного с выделенными полосами трафика (столбец «Несогласованный трафик»). Это касается,

в частности, таких величин, как размах средней задержки, максимальная задержка, размах максимальной задержки (преимущество почти в 5 раз), размах стандартного отклонения и др. (табл. 1).

Таблица 1

Временные характеристики различных алгоритмов

	Согласованный трафик			Несогласованный трафик		
CoS	DRR	ST1	ST2	DRR	ST1	ST2
Ср задержка	24744	24744	24744	25020	25020	25020
0-1	25256	31181	30700	21255	24325	25026
2 3	23987	24299	25500	21162	24662	27748
4 5	26077	24069	20986	22498	27352	27132
6 7	22353	15766	11557	34905	24987	25104
Размах	3723	15415	19142	13742	3026	2723
Макс задержка	225852	256705	217916	418825	241215	216241
0 1	122521	173082	132154	141439	24326	158218
2 3	176252	256705	197346	96584	24662	176172
4 5	170288	212125	217916	159888	27352	206364
6 7	225852	242543	188333	415825	24987	216241
Размах макс задержки	103331	83623	85762	322241	82645	58023
Стд откл	28104	24128	31475	31500	32617	32398
0 1	25754	39905	34444	20783	30154	29598
2 3	25332	28835	31818	20965	29203	32786
4 5	35473	33508	28508	25991	38006	34560
6 7	28682	18465	12974	47452	32087	32266
Размах стд откл	10141	21440	21470	26669	8803	4962
Отношение откл к ср зад	1,13579	0,975105	1,272026	1,258993	1,303637	1,294884
0 1	1,019718	1,279786	1,121954	0,977793	1,23963	1,18269
2 3	1,056072	1,186674	1,247765	0,990691	1,184129	1,181563
4 5	1,360318	1,392164	1,358429	1,155258	1,389514	1,273773
6 7	1,283139	1,171191	1,12261	1,359461	1,284148	1,285293
Размах отн	0,340599	0,220973	0,236475	0,381668	0,205385	0,103731

Что касается моделирования с использованием согласованного с выделяемыми полосами трафика, то по величине максимальной задержки, от которой косвенно зависит доля потерянных данных, преимущество зафиксировано только у алгоритма St2, в то время как по значению размаха этой величины преимущество имеют оба алгоритма стохастического толка. Однако по значению размаха средней задержки преимущество, всѐ же, отмечено у циклического алгоритма.

Таким образом, приведены результаты моделирования стохастического алгоритма. Приведены временные характеристики различных алгоритмов при моделировании на различных входных данных, но одинаковых порциях выборки.

Библиографический список

1. Олифер, В. Г. Компьютерные сети. Принципы, технологии, протоколы / В. Г. Олифер, Н. А. Олифер. – 4-е изд. – СПб. : Питер, 2010. – С. 943.

2. Семенов, А. О. Формальное представление стохастического алгоритма диспетчеризации очередей в телекоммуникационном оборудовании / А. О. Семенов, Н. Н. Коннов // Новые информационные технологии и системы (НИТиС-2014) : сб. науч. ст. XI Междунар. науч.-практ. конф. (г. Пенза, 25–27 ноября 2014 г.) . – Пенза : Изд-во ПГУ, 2014. – С. 301–306.

3. Вегешна, Ш. Качество обслуживания в сетях IP / Ш. Вегешна. – М. : Вильямс, 2003. – С. 368.
4. Кучерявый, Е. А. Управление трафиком и качество обслуживания в сети Интернет / Е. А. Кучерявый. – СПб. : Наука и Техника, 2004. – С. 336.
5. Феллер, Д. Теория вероятностей и еѐ приложения / Д. Феллер. – М. : Финансы и статистика, 1981. – С. 458.

Ю. Ю. Аладьев, К. И. Никишин

Пензенский государственный университет, Пенза, Россия

МОДЕЛИРОВАНИЕ СЕТЕВОГО ТРАФИКА В ПАКЕТЕ CPN TOOLs

Описана модель на основе сетей Петри генератора сетевого трафика для моделей сетей с поддержкой качества обслуживания. Подробно описан формат кадра, основные функции работы модели данного генератора трафика. Для построения модели использовался пакет CPN Tools. Данные, полученные в результате моделирования разработанного генератора, показали, что он может быть использован при исследовании и разработки новых механизмов поддержки качества обслуживания.

Генерация сетевого трафика с заданными характеристиками является важной задачей моделирования компьютерных сетей определенного назначения в частности, с поддержкой качества обслуживания (QoS). Моделирование обработки сетевого трафика коммутатором Ethernet позволяет оценить эффективность обработки трафика путем оценки вероятностных характеристик, в частности джиттера. Перспективным направлением моделирования телекоммуникационных сетей является применение математического аппарата иерархических временных цветных сетей Петри, поддерживаемого свободно распространяемым пакетом CPN Tools, который позволяет подробно исследовать различные вопросы поведения сложных компьютерных сетей. В CPN Tools язык описания моделей представляет собой сочетание графа сети Петри и языка программирования CPN ML [1, 2].

Обычно генерация трафика для моделей реализуется специальной сетью Петри. В частности известны работы, позволяющие генерировать трафик с заданными законами распределения длительности интервалов между поступлениями кадров, длительностью кадров [3, 4].

В настоящей работе представляется генератор сетевого трафика для моделей сетей с поддержкой QoS, в которых обеспечивается возможность дополнительно задавать законы распределения вероятности появления кадров различных классов QoS.

Модель генератора сетевого трафика на основе сетей Петри в пакете CPN Tools с выделением подсетей представлена на рис. 1 и состоит из следующих блоков:

- генератор стохастического трафика;
- генератор регулярного трафика;
- смеситель кадров трафика;
- блок записи полученной последовательности кадров в файл.

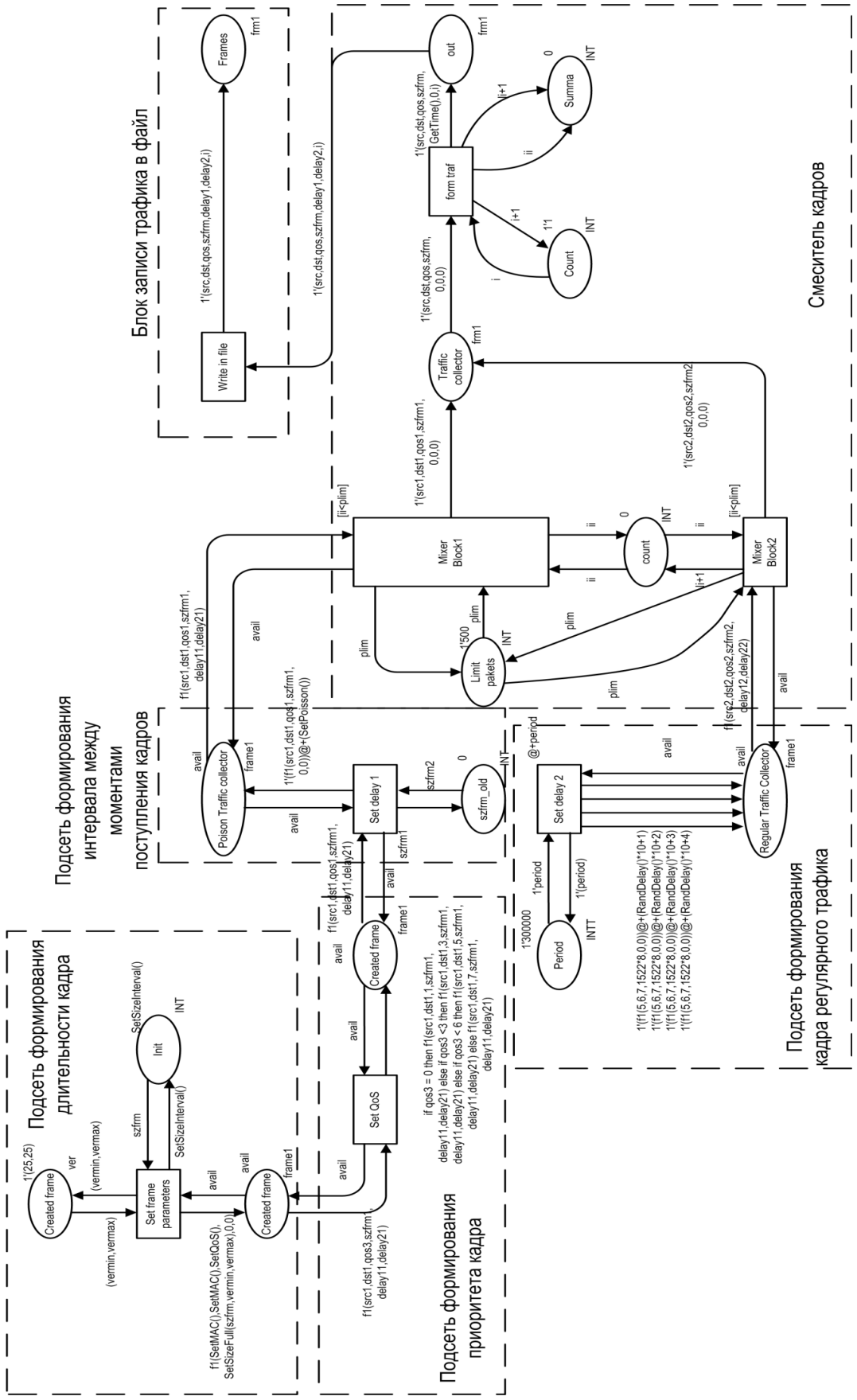


Рис. 1. Модель сети Петри генерирующий сетевой трафик

Генераторы регулярного и стохастического трафиков вырабатывают последовательности маркеров, соответствующие кадрам трафика. Смеситель кадров обеспечивает смешивание отдельных потоков кадров различных видов трафика (стохастического и регулярного) и контролирует длину их генерируемого потока. С помощью блока записи полученная на выходе последовательность кадров записывается в файл.

В модели генератора сетевого трафика представлены следующие цвета:

- цвет frm, который описывает структуру кадра и состоит из MAC-адресов источника и приемника, приоритета кадра, размера передаваемых данных и двух полей задержек;
- цвет frame, который может быть либо занят передачей кадра (frm), либо свободен (avail);
- цвет QoS, который описывает приоритет кадра;
- цвет MAC, который описывает адрес источника или приемника кадра;
- цвет INTT, который описывает временные задержки;
- цвет FRM_SIZE, который описывает размер кадра.

Все параметры кадра являются настраиваемыми, за счет этого обеспечивается гибкость систем.

В модели MAC-адрес источника или приемника кадра определяется функцией *fun SetMAC() = MAC.ran()* в диапазоне от 1 до 10.

Другим важным параметром кадра является приоритет кадра. В данной модели использовался частный случай с четырьмя видами приоритетов: от высокого приоритета к самому низкому с вероятностью распределения 40:30:20:10 процентов. Высокий приоритет имеет всегда значение 7, низкий имеет значение 1. Приоритет регулярного трафика всегда высокий. Приоритет стохастического трафика генерируется в два этапа. Сначала формируются значения по равномерному распределению в диапазоне от 0 до 9 *(fun SetQoS() = = discrete(0,9))* и в зависимости от интервала попадания формируется итоговый приоритет кадра. В результате получается, что приоритеты стохастического трафика распределены с вероятностью 40:30:20:10 от наиболее высокого приоритета к самому низкому. Ниже приведен фрагмент определения приоритета кадра.

```
qos3 = discrete(0,9).
if qos3 = 0 then f1(src1,dst1,1,szfrm1,delay11,delay21)
else if qos3 <3 then f1(src1,dst1,3,szfrm1,delay11,delay21)
else if qos3 < 6 then f1(src1,dst1,5,szfrm1,delay11,delay21)
else f1(src1,dst1,7,szfrm1,delay11,delay21)
```

Следующий параметр это длина кадра. Распределение длины поля данных имеет ярко выраженный бимодальный характер: до 50 % всех кадров имеют минимальную или максимальную длину (64 и 1500 байт, соответственно), длину остальных можно считать равномерно распределенной в оставшемся диапазоне[5].

Формирование кадров стохастического трафика происходило через временные интервалы, распределенные по экспоненциальному закону. Через экспоненциальное распределение в функции *fun SetPoisson() = floor(exponential(1.0/10000.0)+0.5)* можно регулировать загрузку канала.

Формирование кадров регулярного трафика происходило с определенным постоянным периодом. Одновременно формировались 4 кадра. Для устранения одновременного прихода кадров добавлялась задержка, которая вычислялась по формуле:

$$\text{Задержка} = discrete(1,4) * 10 + \text{Номер\_кадра}$$

Запись полученного трафика в файл осуществлялась стандартными средствами пакета CPN Tools с помощью монитора. Фиксация сгенерированного трафика в файле обеспечивает повторяемость экспериментов при смене или коррекции исследуемой модели компьютерной сети, тем самым позволяет анализировать и выбирать наилучшую модель в конкретной ситуации.

На рис. 2 представлены диаграммы вероятностных характеристик трафика, полученные в результате моделирования разработанного генератора сетевого трафика.

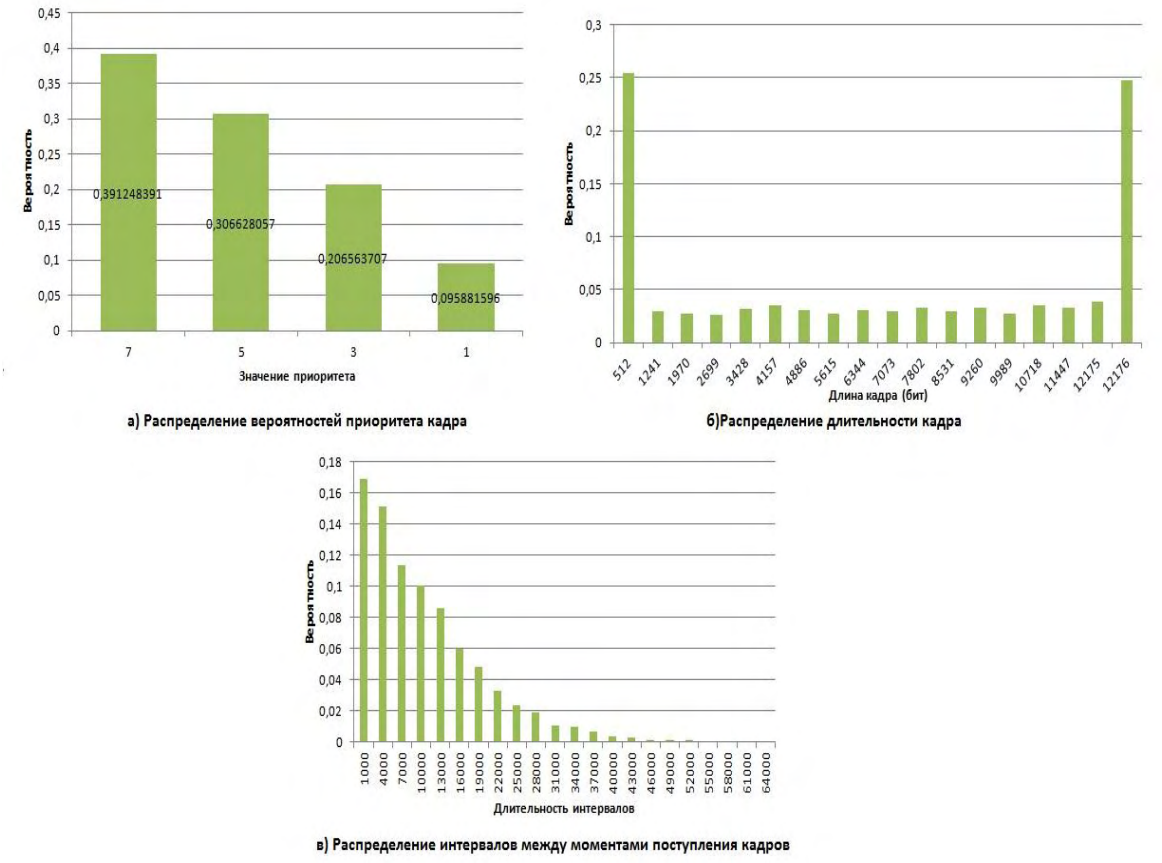


Рис. 2. Результаты моделирования

Данные, полученные в результате экспериментального исследования работы предлагаемого генератора показали, что он может быть использован при исследовании методами имитационного моделирования компьютерных сетей с поддержкой QoS с целью параметрической оптимизации работы корпоративных сетей или разработки новых механизмов поддержки качества обслуживания.

Библиографический список

1. Jensen, K. Coloured Petri Nets: modelling and validation of concurrent systems / Kurt Jensen, Lars M. Kristensen. – Springer, 2009. – 384 p.

2. Зайцев, Д. А. Моделирование телекоммуникационных систем в CPNTools / Д. А. Зайцев, Т. Р. Шмелева. – Одесса : Одесская национальная академия связи им. А. С. Попова, 2008. – 68 с.

3. Никишин, К. И. Моделирование трафика сети Ethernet цветными сетями Петри / К. И. Никишин, Н. Н. Коннов, А. Л. Домнин // Современные проблемы компьютерных наук (СПКН-2013) : сб. материалов I Междунар. науч.-практ. конф., посвящ. 70-летию образования Пензенского государственного университета (г. Пенза, 29–30 октября 2013 г.) / под ред. Л. Р. Фионовой. – Пенза : Изд-во ПГУ, 2013 – С. 120–124.

4. Никишин, К. И. Генератор трафика ethernet на основе цветных сетей петри / К. И. Никишин, Н. Н. Коннов // Модели, системы, сети в экономике, технике, природе и обществе. – 2016. – № 1(17). – С. 299–307.

5. Петров, В. В. Структура телетрафика и алгоритм обеспечения качества обслуживания при влиянии эффекта самоподобия : автореф. дис. … канд. техн. наук : 05.12.13 / Петров В. В. – М., 2004. – 198 с.

М. Ю. Новиков

Пензенский государственный университет, Пенза, Россия

РАДИОСЕТЬ ПЕРЕДАЧИ ДАННЫХ SKYMAN/CA

Педставлен обзор продукта компании CompTEK AquA, коммуникационной структуры Sky Metropolitan Area, которая является разработкой группы российских инженеров.

Развитие информационных связей в крупных старых городах России в связи с резким возрастанием требований по производительности связной инфраструктуры из-за вовлечения в информационный обмен цифровых компьютерных потоков, зашло в тупик. Единственной связной инфраструктурой общего пользования является телефонная сеть. В США и других развитых странах телефонные сети строятся на современных цифровых принципах, что позволяет использовать их для обмена цифровой информацией с достаточной скоростью. Доставшаяся нам в наследство телефонная сеть даже при применении самых современных систем и методов сжатия информации в лучшем случае обеспечивает скорость около 28Кбит/с. Учитывая, что такая скорость достигается только в определенных непродолжительных благоприятных ситуациях, в большинстве случаев такой производительности уже явно недостаточно.

Все существующие и нарождающиеся компьютерные российские глобальные сети строятся на телефонных сетях или на соответствующих им по скорости, но несколько превосходящим по надежности ведомственных линиях связи. Это создает серьезные трудности в создании полноценных информационных связей между разрозненными локальными быстродействующими кабельными сетями предприятий и учреждений, ограниченных их территориями, узлами спутниковой и международных связей и отдельными потребителями информации. Идея создания новой всепокрывающей связной информационной инфраструктуры с использованием перспективных оптоволоконных сетей для соединения уже сегодня большого числа абонентов на обширных территориях трудно реализуема из-за колоссальных временных и финансовых затрат, что диктует необходимость параллельного создания альтернативных способов передачи информации.

Использование лазерных систем связи в открытом пространстве над городами средней полосы России не представляется возможным из-за погодных условий. Целью реализации данного проекта является создание беспроводной компьютерной сети общего пользования. Использование сверхвысокочастотного диапазона радиоволн в качестве среды передачи информации решает вопрос создания высокоскоростных компьютерных коммуникаций между абонентами практически независимо от их расположения в пределах выделенных территорий, быстро и без значительных капитальных затрат для абонента.

В основу архитектуры системы заложены следующие принципы:

- строгое соблюдение международных стандартов в области систем связи и информационно-вычислительных сетей;
- использование сертифицированной Министерством Связи Российской Федерации аппаратуры;
- максимальная прозрачность в использовании;
- совместимость с большей частью сетевого программного обеспечения;
- использование новейших достижений в информатике и связи с целью реализации всех необходимых прикладных функций и потребностей пользователей системы;
- ремонтопригодность системы, обеспечиваемая возможностью оперативной замены типовых блоков;
- живучесть сети, определяемой наличием резервных путей и гибкой, динамической маршрутизацией, а также целым комплексом других мер, обеспечивающих сохранение работоспособности системы в целом при неисправности отдельных ее элементов;

• управляемость системы, достигаемая с помощью полного дистанционного управления маршрутизаторами.

Ключевой концепцией данного проекта является использование новой технологии построения беспроводных компьютерных сетей, основанной на передаче шумоподобных сигналов. Данная технология позволяет без использования проводных каналов построить многосотовую структуру для крупных городов области и односотовые кусты в мелких населенных пунктах, не имеющих своей проводной инфраструктуры, необходимой для решения нарождающихся информационных задач. Эта технология дает возможность развернуть предлагаемую систему в течение считанных месяцев, обеспечив при этом беспрецедентную простоту и оперативность подключения конечных пользователей.

SkyMAN/CA базируется на технологии передачи на размытом спектре прямой последовательности (DSSS), что обеспечивает высокую помехозащищенность в полосе приема. Другим положительным моментом систем с DSSS является относительно несложная процедура получения разрешения на эксплуатацию системы. В SkyMAN/CA используются радиоустройства, совместимые в рамках стандартовIEEE 802.11/802.11b.

В беспроводных технологиях, ориентированных на множественный доступ особенно болезненным является вопрос об эффективном разделении эфира между станциями. Наиболее часто используемым в этих случаях является протокол CSMA/CA, который построен на базе широко распространенного протокола CSMA/CD, входящего в состав Ethernet. Однако, в отличие от Ethernet, он не обеспечивает прослушивание среды во время передачи, что приводит к чрезвычайно большим накладным расходам при множественном доступе. При количестве станций в сегменте около 25 и интенсивном трафике, реальная пропускная способность составляет около 18 % от максимальной. Технология SkyMAN/CA включает в себя поллинговые протоколы обслуживания среды, оптимизированные для обеспечения множественного доступа в соте(WMA, WMA-CD).

За взаимодействие элементов SkyMAN/CA отвечает протокол множественного доступа, основанный на маршрутизации. RMA (Routed Multiple Access protocol), то есть протокол множественного доступа, основанный на маршрутизации, является одним из главных элементов архитектуры SkyMAN/CA. RMA позволяет в полной мере использовать преимущества сотовой архитектуры с применением многосекторных базовых станций. Протокол RMA построен на 3 основных механизмах:

• протокол тестирования каналов;

• регистрация абонентов на базовой станции с постоянным контролем качества канала;

• механизм роуминга (перемещение абонента от станции к станции) основанный на динамической IP маршрутизации.

На каждой базовой станции в таблицах хранится информация о правах и параметрах всех абонентов, которые могут быть к ней подключены. При разрастании сети объем информации возрастает, что приводит к соответствующим затруднениям в работе оператора сети.

Служба разрешения удаленного доступа RAPSэффективно решает эту проблему путем централизации всей информации об абонентских блоках в одной базе данных и автоматизации процесса конфигурации. Практически отпадает всякая необходимость непосредственного управления базовыми станциями, которые выполняют роль посредника между абонентским блоком и сервером RAPS.

SkyMAN/CA использует метод передачи информации с коммутацией пакетов. Базовым сетевым протоколом в системе является протоколIP (Internetwork Protocol), применяемый в сети Internet. Все остальные сетевые протоколы (такие, как IPX, Netbios, HDLC, SDLC, CLNS, X.25) должны инкапсулироваться в протокол IP устройствами пользователя. Этот подход позволяет использовать главные преимущества протокола IP, такие, как эффективная многопутевая маршрутизация, маршрутизация с объявлением состояния о канале (link-state), протоколRMA(Routed Multiple Access Protocol) и многие др.

Главной особенностью услуг, предоставляемых сетью SkyMAN, является то, что в отличие от других сетей передачи данных, использующих коммутируемые телефонные или выделенные линии связи на арендной основе, сеть SkyMAN имеет свою собственную транспортную структуру, специализированную на передаче цифровой информации, обеспечивающую для конечного пользователя скоростные и надежностные характеристики, недостижимые в других сетях при соизмеримых затратах.

Сеть реализуется в виде сотовой структуры, образуемой базовыми станциями (БС). К сети могут подключаться абоненты, расположенные в прямой видимости одной из БС. БС соединяются между собой и основными информационными узлами города беспроводной магистралью. Базовые станции размешаются на главенствующих высотах города, что дает возможность реализовать максимальное количество запросов на подключение. С точки зрения структуры, сеть SkyMAN представляет собой множество сетей, объединенных специализированными радиомаршрутизаторами Callisto, поддерживающими протоколы RTIP, TCP/IP, IPX/SPX, X.25. Использование множества взаимодействующих маршрутизаторов с централизованным администрированием позволяет обеспечить полный контроль за трафиком в сети. Многоуровневая система защиты от несанкционированного доступа предотвращает бесконтрольный доступ к сетям абонентов извне.

Сеть пользователя стыкуется с Сетью при помощи блоков доступа к сети, являющимися техническими границами Сети. Блок доступа к SkyMAN является радиомаршрутизатором. Программное обеспечение блоков доступа обеспечивает изоляцию трафика Сети от трафика сети пользователя и их взаимодействие в соответствии с параметрами, устанавливаемыми дистанционно с центра управления сетью. Протоколы взаимодействия с сетями пользователей: TCP/IP, IPX, X.25, RTIP. Антенно-кабельная система (от комнатной антенны, при наличии прямой видимости БС из окна здания абонента, до внешней параболической антенны) обеспечивает соединение блока доступа с БС.

Скоростные характеристики блока доступа.

Максимальная программная скорость передачи данных при связи с другим абонентом внутри одной соты – БС – 400 Кбит/с;

Максимальная программная скорость передачи данных при связи с другим абонентом другой соты – БС – 800 Кбит/с;

Блок доступа можно подключать к локальной сети абонента по оптоволокну, обеспечивая этим гальваническую развязку с локальной сетью и снимая ограничение на расстояние между локальной сетью и местом установки блока доступа. Дальность от центра соты до абонента зависит от комплектации базовой станции и абонента. Для обеспечения максимальной производительности системы необходимо не допускать работу линии без запаса по энергетике (работу на максимальной дальности). Для работы в любую погоду необходим запас на снижение эффективности антенн из-за осадков в виде мокрого снега. Затуханием сигнала на трассе из-за атмосферных осадков можно пренебречь из-за незначительного поглощения сигналов частоты 2400 МГц.

Питание БС осуществляется через непрерываемые источники питания типа APC Smart UPS 700XL, обеспечивающие мониторинг наличия напряжения сети, передачу информации в центр управления сетью, сохранение энергопитания БС при отключении внешнего энергопитания на время более 10 ч.

Управление сетью обеспечивается с Центра управления сетью (ЦУС). Мащрутизатор каждой из БС обеспечивает мониторинг соединенных с ним сетей и предоставляет необходимую информацию ЦУС.

Сопряжение сети с сетями Internet, INFOTEL и другими глобальными сетями обеспечивается центром коммутации пакетов. Данное решение обеспечивать для своих пользователей техническую возможность обмена данными с пользователями любой другой СДОП, либо с использованием непосредственного межсетевого сопряжения, либо с помощью транзита через третьи сети. Реализация такой возможности осуществляется по соглашению между операторами сетей.

Библиографический список

1. Кучерявый, Е. А. Управление трафиком и качество обслуживания в сети Интернет / Е. А. Кучерявый. – СПб. : Наука и Техника, 2004. – С. 336.
2. Мешковский, К. А. Цифровые системы передачи: принципы работы / К. А. Мешковский. – СПб. : Энергомашиностроение, 2007.

Д. А. Замятин, В. С. Богданова, С. В. Степанова

Сибирский государственный университет науки и технологий
имени академика М. Ф. Решетнева (СибГУ), Красноярск, Россия

КОНЦЕПЦИЯ НЕЗАВИСИМОЙ СПУТНИКОВОЙ КОМПЬЮТЕРНОЙ СЕТИ

Описывается разработка концепции крупномасштабной компьютерной сети с размещением центров обмена данными на орбите Земли и независимой от сети Интернет.

На сегодняшний день одной из важных задач развития информационных технологий является повышение уровня сетевой безопасности. В частности, актуальна проблема нахождения компромисса между обеспечением с одной стороны эффективного противодействия экстремизму, а с другой стороны – возможности рядовых пользователей при пользовании сетью чувствовать себя достаточно комфортно, не опасаясь утечки персональных данных.

Принципы работы сети Интернет практически исключают такой компромисс, что делает необходимым создание независимой от Интернета компьютерной сети с совершенно иными принципами, частично имеющей сходство с компьютерными сетями военного назначения, что не позволит сети стать источником вреда как для ее пользователей, так и для общественной безопасности.

Основные принципы проекта сети:

1) деловая и научная направленность;
2) полная независимость от сети Интернет, на аппаратном уровне;
3) ограничение возможности внесения контента с клиентских устройств;
4) обеспечение максимально возможной конфиденциальности обмена информацией между пользователями.

В числе прочих особенностей проекта сети важное место занимает отсутствие проводной связи крупных серверных комплексов как с клиентскими устройствами, так и между собой. Вместо этого вся связь осуществляется через спутники связи и орбитальные серверные комплексы. На ранних этапах проекта рассматривался вариант размещения всех без исключения физических носителей информации такой сети на орбите Земли, что на сегодняшний день является весьма сложной и дорогостоящей для реализации идеей. В связи с этим было принято решение несколько скорректировать вектор развития проекта [1].

Таким образом, на сегодняшний день цель проекта – найти оптимальный вариант концепции крупной компьютерной сети, с полной независимостью от Интернета и повышенным уровнем защищенности.

Задачи:

1) оценка возможностей вывода на орбиту крупных серверных комплексов и формирования орбитальных центров обмена данными;

2) выбор целевой орбиты и подходящего носителя для заданной цели;

3) разработка концепции космического аппарата для решения поставленных задач;

4) решение вопроса обеспечения надежной связи между космическими аппаратами и с наземными устройствами.

Размещение аппаратного обеспечения крупной компьютерной сети на орбите Земли делает необходимым разработку крупномасштабной космической программы. В данной работе рассмотрена оптимизация изначальной концепции сети с учетом выявленных на предыдущих этапах преимуществ и недостатков.

Уязвимость к космическому мусору, сложность доставки на орбиту и сильно ограниченные возможности обслуживания на данный момент делают полное размещение всех вычислительных мощностей на космических аппаратах не вполне оптимальным вариантом. К первоначальной концепции полного размещения всех центров обмена данными в космосе планируется вернуться на более поздних этапах развития, после появления и широкого распространения новых поколений информационных носителей с более высокой плотностью данных, что сделает возможным создание серверов меньших размеров при той же вычислительной мощности.

Размещение центров обмена данными на орбите, однако, обеспечивает большую зону покрытия сети и высокую скорость передачи информации, причем даже в местности, где другие виды связи недоступны, а недосягаемость для человека может являться не минусом, а плюсом, ввиду высокой защищенности от злоумышленников. Поэтому на текущем этапе был предложен компромиссный вариант. Основные узлы сети – орбитальные модули связи, служащие для обмена информацией с наземными приемно-передающими устройствами и другими такими же КА. К модулю связи пристыковываются блоки хранения и обработки данных [1].

В отличие от первоначального варианта, как таковой серверный комплекс не размещается полностью на космическом аппарате по причине слишком большой массы, осложняющей задачу доставки КА на целевую орбиту и ограничивающей возможности обслуживания серверного комплекса. Часть комплекса находится на Земле, в космосе же предлагается оставить так называемые «кэшевые» серверы, так как на них хранятся лишь копии данных, находящихся непосредственно в работе, т.е. в течение сравнительно непродолжительного времени.

Следовательно, орбитальный центр обмена данными имеет сравнительно небольшую массу, что также обусловлено отсутствием крупных систем охлаждения серверов ввиду низкой температуры окружающей среды. Таким образом, размещение части аппаратного обеспечения сети на Земле решает проблемы как охлаждения серверных комплексов, так и их доставки на орбиту.

Общий принцип передачи данных при этом выглядит следующим образом: крупный наземный центр хранения информации обменивается данными с геостационарным узлом связи, расположенным на соответствующей долготе, а узел связи передает пакеты данных на низкоорбитальные кэшевые серверы, ближайшие к тому или иному клиентскому устройству, запрашивающему информацию.

Одним из определяющих параметров при выборе орбиты является масса космических аппаратов. Другим – скорость обмена данными между орбитальным дата-центром и наземными устройствами. При оценке этих параметров наиболее уместным решением выглядит размещение кэшевых орбитальных серверов на низкой околоземной орбите (НОО) и узлов связи на геостационарной орбите (ГСО) [2].

В качестве предполагаемого средства выведения серверных комплексов на орбиту рассматриваются российские ракеты-носители семейства «Ангара», что открывает достаточно широкие возможности при выборе оптимальной орбиты и проектировании космического аппарата для представленных целей [3].

Следующий этап работы включает в себя расчет массогабаритных характеристик космического аппарата для заданных целей и выбор наиболее подходящей для него орбиты, а также уточнение выбора носителя.

Таблица 1

Характеристики ракет семейства «Ангара»

	Ангара 1.2	Ангара А3	Ангара А5
Стартовая масса, т	171	481	773
Масса полезной нагрузки, т:			
– для НОО (Н = 200 км, i = 63°)	3,5	14,0	24,0
– на ГПО (Н = 5500 км, i = 25°)	–	2,4 («Бриз-М»)	5,4 («Бриз-М»)
			7,5 («КВТК»)
– на ГСО	–	1,0 («Бриз-М»)	2,8 («Бриз-М»)
			4,5 («КВТК»)

Исследование выполнено при поддержке краевого государственного автономного учреждения «Красноярский краевой фонд поддержки научной и научно-технической деятельности» в рамках участия в мероприятии XIV Международная научно-техническая конференция «Новые информационные технологии и системы (НИТиС-2017)».

Библиографический список

1. Замятин, Д. А. Разработка концепции орбитальной компьютерной сети / Д. А. Замятин, Е. П. Шашило, В. В. Кольга // Решетневские чтения : материалы XX Междунар. науч. конф. (г. Красноярск, 9–12 ноября 2016 г.) : в 2 ч. / под общ. ред. Ю. Ю. Логинова ; Сиб. гос. аэрокосм. ун-т. – Красноярск, 2016. – С. 27–28.
2. Наблюдение ИСЗ. – URL: http://www.sat.belastro.net/ (дата обращения: 14.09.2016).
3. ГКНПЦ им. М. В. Хруничева. Семейство ракет-носителей «Ангара». – URL: http://www.khrunichev.ru/main.php?id=44 (дата обращения: 20.05.2017).

М. Н. Коннов

Пензенский государственный университет, Пенза, Россия

ПОВЫШЕНИЕ ТОЧНОСТИ УСТАНОВКИ СЕТЕВОГО ВРЕМЕНИ ПРОТОКОЛОМ IEEE-1588

Рассматривается возможность повышения синхронизации по протоколу IEEE-1588 устройств в сети Ethernet за счет организации в коммутаторах виртуального тайслота.

Технология Ethernet в настоящее время начала широко применяться в системах промышленной автоматизации, в контрольно-измерительных и управляющих системах. Обязательным условием функционирования таких систем является синхронизация операций, выполняемых в объединенных сетью ЭВМ или контроллерах устройств. Для синхронизации часов реального времени в сетевых устройств используется протокол «точного времени» (PTP – Precision Time Protocol), который действует на втором уровне модели взаимодействия открытых систем (OSI). Протокол PTP описан в стандарте IEEE 1588 [1, 2].

Стандарт IEEE 1588 метод синхронизации устройств в сети с точностью выше 1 мкс (до 10 нс). Данный протокол обеспечивает синхронизацию ведомых устройств от ведущего, удостоверяясь, что события и временные метки на всех устройствах используют одну и ту же временную базу. В протоколе предусмотрены две ступени для синхронизации устройств: определение ведущего устройства и коррекция разбега во времени, вызванного смещением отсчета часов в каждом устройстве и задержками в передаче данных по сети.

При инициализации системы протокол PTP использует алгоритм «наилучших ведущих часов» для определения самого точного источника синхронизации в сети, который ведущим, а все остальные устройства в сети – ведомые и корректируют свои часы по ведущему устройству. Разница во времени между ведущим и ведомым устройствами является

комбинацией смещения отсчета часов и задержки передачи синхронизирующего сообщения. Поэтому коррекция временного сдвига должна выполняться в два этапа: вычисление задержек передачи и сдвига, а затем их коррекция.

Порядок синхронизации часов двух устройств приведен на рис. 1.

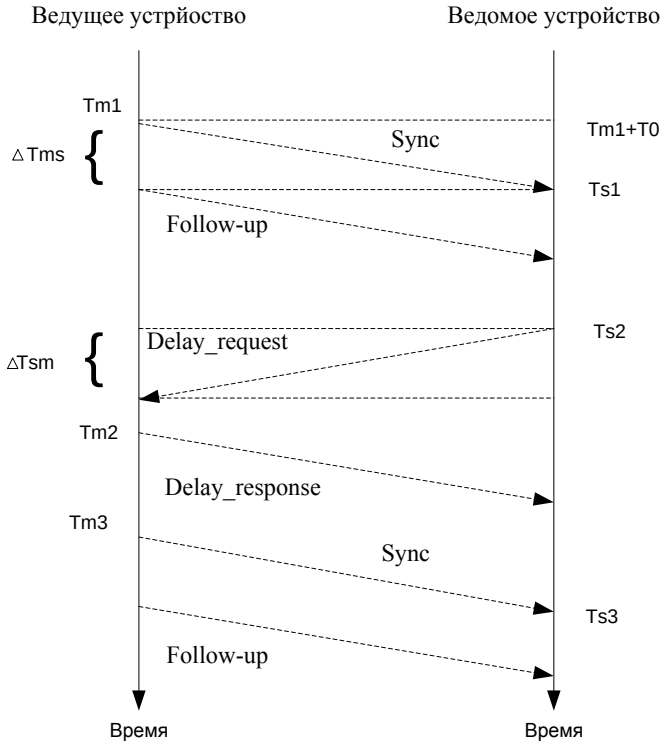


Рис. 1. Обмен сообщениями при синхронизации двух часов

В первой фазе корректируется расхождение между ведущими и ведомыми часами за счет циклической передачи ведущими часами сообщений, которые содержат расчетное значение точного времени его передачи (сообщения Sync и Follow-up). Подчиненные часы вычисляют коррекцию, учитывая при этом время передачи кадра, которое определяется во второй фаза процесса синхронизации.

С этой целью ведомые часы направляют ведущим пакет запроса Delay_request, а ведущие часы фиксируют время приема пакета и посылают его значение подчиненным часам в специальном пакете Delay_response. Затем подчиненные часы вычисляют время задержки, при этом особо важным является равнозначность задержки в каждом из направлений поскольку при вычислении задержки передачи сообщения принимается, что средняя задержка передачи данных в канале равна среднему арифметическому задержек распространения в разные стороны канала.

Окончательная синхронизация часов выполняется после отправки ведущим устройством второго набора сообщений Sync и Follow-up После этого ведомое устройство подстраивает свои часы в соответствии с вычисленными значениями.

Поскольку опорные источники синхронизации в каждом устройстве нестабильны, а задержки в канале могут меняться со временем, необходимо периодически повторять коррекцию часов ведомого устройства.

Точность синхронизации зависит от точности измерения задержек передачи кадра в каждом звене сети Ethernet, которая складывается из следующих составляющих:

- задержка распространения, вызванная конечной скоростью распространения сигнала в среде передачи (меди, оптоволокне);
- задержка сериализации, связанная с последовательной (побитной) передачей кадров в выходных портах;

• задержка коммутации, определяемая временем для продвижения кадра из входного в выходной порт;

• задержка в очереди, возникающая при задержке кадров из-за перегрузки сети.

Две последние составляющие носят случайный характер и зависят от нагрузки сети, причем задержка кадров в очередях может достигать сотен микросекунд.

В стандарте IEEE 1588 v.1 2002 г. подразумевается, что задержки в канале связи симметричны; то есть, задержка передачи в прямом канале равна задержке передачи в обратном канале. Кроме того, задержка не должна меняться в процессе ее измерения. Изменение задержки в процессе ее измерения приведет к асимметрии и джиттеру, которые неизбежно повлияют на точность синхронизации, поэтому протокол первой версии чрезвычайно чувствителен к разбросу задержек при вычислении коррекции.

В стандарте IEEE 1588 v.2 2008 г. При программно-аппаратной реализации наиболее чувствительные функции протокола, такие как запись временной метки PTP-пакета, реализуются на физическом уровне Ethernet, при прохождении пакета через коммутатор, позволяя добиться точности протокола PTP порядка 10–20 нс [3]. Естественно, вторая версия требует применение специальных сетевых плат и коммутаторов.

В работе [4] автор показал, что погрешности оценки задержек при использовании протокола PTP версии 2002 г. могут быть достаточно просто устранены за счет организации в коммутаторе передачи кадров через так называемый «виртуальный тайм-слот» (ВТС) с пост-блокировкой [5, 6].

Необходимыми условиями реализации ВТС являются:

• поддержка протоколов IEEE 802.1 q /p [7];

• кадрам PTP назначен наивысший приоритет;

• полоса, выделяемая в PTP-трафика, исключает накопление PTP кадров в буферах коммутаторов.

Передняя граница виртуального тайм-слота в каждом коммутаторе определяется моментом приема и распознания очередного PTP-кадра, после чего он задерживается в буфере высшего приоритета на фиксированное время Тбл > Тмах, где Тмах – время передачи кадра IEEE 802.1 q /p наибольшей длины. На временя блокировки запрещается начало передачи через выходной порт любых кадров. После снятия блокировки через освобожденный выходной порт начнется передача PTP-кадра и кадров данных.

Рассмотренный вариант управления трафиком позволяет исключить джиттер, но приводит к дополнительной фиксированной задержке PTP-кадра в каждом узле.

Однако дальнейшему увеличению точности препятствует собственный разброс задержек транспортировки кадров внутри коммутатора. Эта погрешность не может быть устранена организацией ВТС, описанной выше, так как момент блокировки связан с моментом классификации, выполняемым в порту назначения.

Автором предлагается способ устранения этой погрешности за счет привязки момента начала блокировки к моменту поступления или окончания приема пакета во входном порту. Цель – повышения точности протокола временной синхронизации в сети за счет фиксации задержек коммутатора по методу виртуального таймслота.

Естественно, что эти действия должны выполняться на физическом уровне, как показано на рис. 2.

Пусть в момент Т1 начнется прием очередного кадра в входной порт.

В момент Т2 прием закончится и по сигналу аппаратуры входного порта зафиксируется показание внетренних часов коммутатора (например, с точностью 10 нс). Код добавляется к принятому кадру и направляется в порт назначения.

В момент Т3 завершиться коммутация.

В момент Т4 произойдет идентификация типа кадра. Если кадр не относится к протоколу PTP. Он просто направляется в соответствующий буфер, а его метка времени отбрасывается. В противном случае метка будет в буфер атрибутов соответствующей высокоприоритетной очереди.

Пусть в момент Т5 выходной порт освободится, и схема обзора считает атрибут высокоприоритетной очереди. Наличие метки в этом буфере вызывает запуск блокировки ВТС.

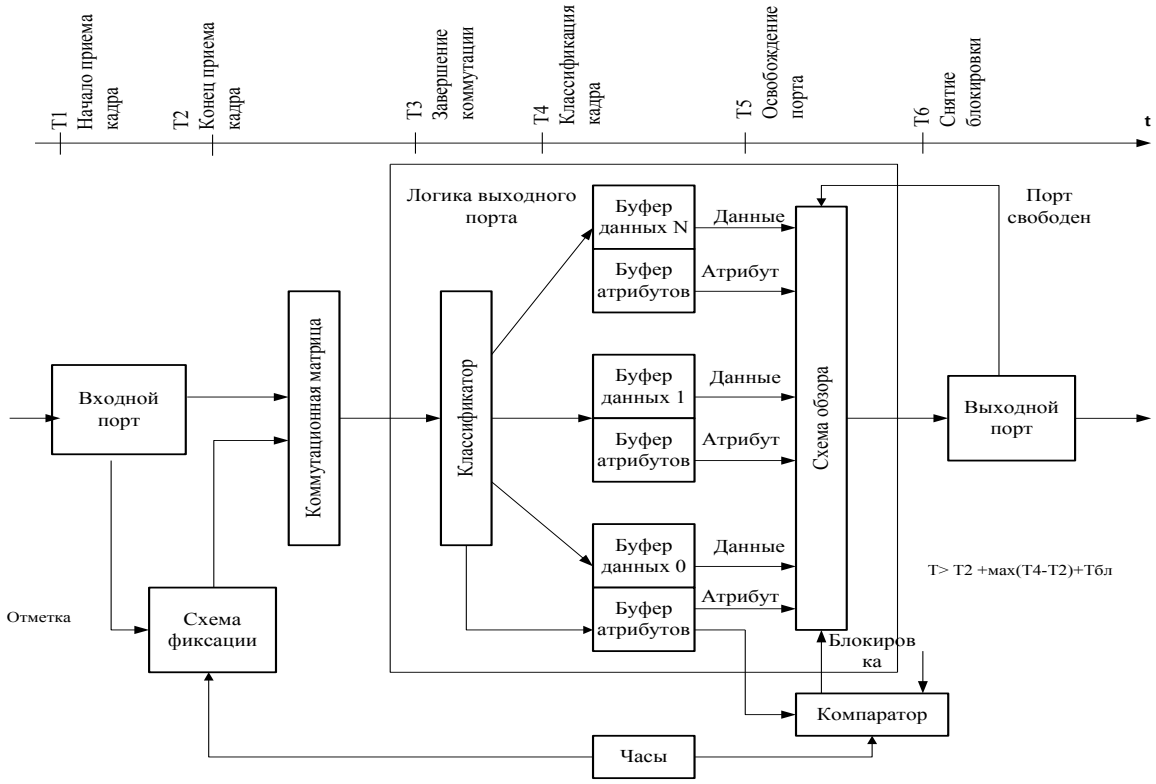


Рис. 2. Схема счет фиксации задержек коммутатора по методу виртуального таймслота

Момент снятия блокировки определится так $T > T2 + мах(T4–T2) + Тбл$, где мах(T4–T2) – заранее заданный наибольший разброс задержки коммутации.

Пусть в момент T5 выходной порт освободится, и схема обзора считает из буфера атрибут высокоприоритетной очереди. Наличие метки PTP в этом буфере вызывает запуск блокировки BTC.

В момент T6 начинается передачи кадра через выходной порт. Временной интервал T6–T2 должен быть постоянен (с точностью < 10 нс). Метки времени в передаваемый кадр не помещаются.

Таким образом, повышения точности протокола временной синхронизации в сети будет достигнуто за счет фиксации задержек передачи PTP пакета через коммутатор по методу виртуального таймслота. Применение предложенного способа позволит упростить схему коммутатора, исключив трудоемкие операции по корректировки пакета PTP для записи в него измеренной задержки. Упрощается так же обработка кадра в ведомых часах и появляется возможность применять в овонечных устройствах обычные микроконтроллеры.

Библиографический список

1. Eidson, J. C. Measurement, Control, and Communication Using IEEE 1588 / J. C. Eidson. – Springer-Verlag London, 2006. – P. 284.

2. Телегин, С. Протокол PTP для синхронизации сетей NGN. Вопросы применения / С. Телегин // Первая миля. – 2009. – № 5-6. – С. 20–23.

3. Вербенин, Б. А. Особенности синхронизации протокола IEEE 1588 / Б. А. Вербенин // Электронные компоненты. – 2011. – № 11. – С. 52–55.

4. Коннов, М. Н. О повышении точности синхронизации сети Ethernet по протоколу IEEE 1588 / М. Н. Коннов // Информатика: проблемы, методология, технологии : сб. материалов XIV Междунар. науч.-метод. конф. : в 3 т. – Воронеж : Вэлборн, 2014. – Т. 3. – С. 210–214.

5. Коннов, М. Н. Об одном механизме передачи трафика реального времени в сетях с пакетной коммутацией / М. Н. Коннов, В. Б. Механов // Современные информационные технологии. – 2010. – № 11. – С. 139–141.

6. Механов, В. Б. Механизмы компенсации джиттера в коммутаторах Ethernet / В. Б. Механов, М. Н. Коннов // Телематика 2013 : тр. XX Всерос. науч.-метод. конф. – СПб. : СПбГУ ИТМО, 2013. – Т. 1. – С. 235–237.

7. Филимонов, А. Построение мультисервисных сетей Ethernet / А. Филимонов. – СПб. : БХВ-Петербург, 2007. – С. 592.

Е. А. Кизилов, Н. Н. Коннов, Д. В. Патунин, И. Ю. Фролов

Пензенский государственный университет, Пенза, Россия

ОЦЕНКА РАЗМЕРОВ ОЧЕРЕДЕЙ В КОММУТАТОРЕ С ПОДДЕРЖКОЙ QOS

Проводится сравнительный анализ статистических характеристик очередей в коммутаторах Ethernet с поддержкой качества обслуживания, использующих дефицитный циклический и динамический приоритетный алгоритмы диспетчеризации очередей.

В современных коммутаторах Ethernet функционируют службы поддержки качества обслуживания (QoS- quality of service), которые обеспечивают для трафика определенного класса заданные требования по полосе пропускания, задержке кадров, ее вариации (джиттера) и доле потерянных кадров. Важным механизмом службы QoS является диспетчеризация очередей в коммутаторах, от алгоритмов которой зависят значения задержки и джиттера передаваемых кадров, определяющие возможность передачи в сети Ethernet трафика реального времени [1].

Для диспетчеризации очередей широко используется циклические алгоритмы, в частности алгоритм дефицитного циклического обслуживания (DRR – Deficit Round Robin). В этом алгоритме для каждого класса QoS-трафика выделяется отдельная очередь. Очереди обслуживаются в порядке возрастания их номеров, причем при отсутствии кадров в некоторой очереди алгоритм переключается на другую очередь с большим номером. Количество данных, передаваемых из каждой очереди за один сеанс обслуживании, задается закрепленным за ней весом, пропорциональным полосе пропускания выходного порта, выделяемой для трафика соответствующего класса, и максимальному количеству (бит) байтов данных, передаваемых из очереди за один сеанс [2].

Достоинства DRR алгоритма – исключение монополизации полосы пропускания выходного порта, высокая точность задания выделяемой полосы пропускания для определенного класса трафика независимо от разброса размеров продвигаемых коммутатором кадров, а так же простота реализации. Недостатки DRR алгоритма - неконтролируемая величина джиттера коммутируемых кадров из-за разброса время ожидания в различных очередях, вызванного детерминистским порядком их обслуживания. Кроме того, задержка кадров определенного класса QoS, а так же размеры соответствующих очередей, зависят от выделенной для конкретного класса полосы пропускания, в результате алгоритмы не позволяют эффективно перераспределять неиспользуемую полосу пропускания недогруженных каналов для расширения полосы пропускания перегруженных. Это приводит при пульсирующем характере трафика или в случае несогласованности фактической нагрузки и выделенной полосы пропускания для трафика класса QoS к увеличении задержки кадров и перегрузке очередей в каналах с малой назначенной полосой пропускания.

Для решения этой проблемы был предложен динамический алгоритм диспетчеризации DRR-TSS (TSS –Time Selected Service), совмещающий принципы циклической диспетчеризации с учетом времени ожидания обслуживания кадров (временной селекцией) при ограничении длительности непрерывного обслуживания каждой очереди. Проведенные исследования джиттера кадров различного вида трафика показали, что DRR-TSS алгоритм обеспечивает более справедливость распределения выделяемой для каждого класса трафика полосы пропускания [3, 4].

В настоящей работе приводятся результаты исследований другой важной характеристики, показывающей эффективность алгоритма диспетчеризации очередей, – размеров очередей. Оценка эффективности алгоритмов динамической диспетчеризации по сравнению с алгоритмами DRR выполнялась их имитационным моделированием в пакете CPN Tools, использующем математический аппарат цветных иерархических временных сетей Петри [5]. Структура сетей Петри (рис. 1), моделирующих буфер выходного порта, в котором реализуются алгоритмы диспетчеризации очередей реализуются, были построены по принципам, изложенным в [3, 6, 7], при этом для контроля размеров очередей были в подсети Петри, моделирующие буферную память коммутатора, были добавлены специальные измерительные элементы.

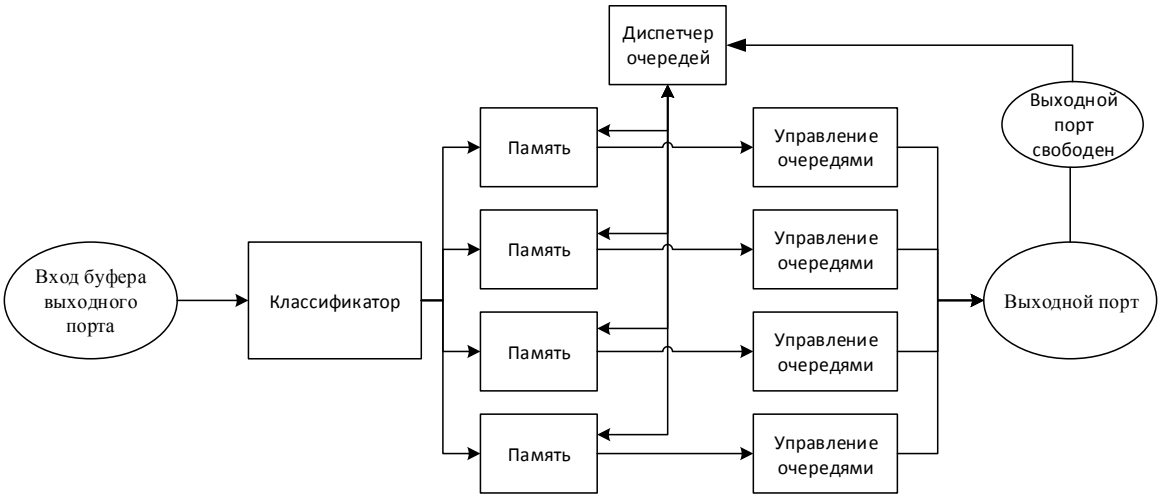


Рис. 1. Структура модели буфера выходного порта коммутатора

Пример такой подсети ***Queue*** – очереди типа FIFO, буферизирующей кадры одного класса, случая алгоритма DRR-TSS приведен на рис. 2. В этой же подсети также реализованы подсчет размера текущей очереди и отбрасывание поступивших кадров при переполнении буфера.

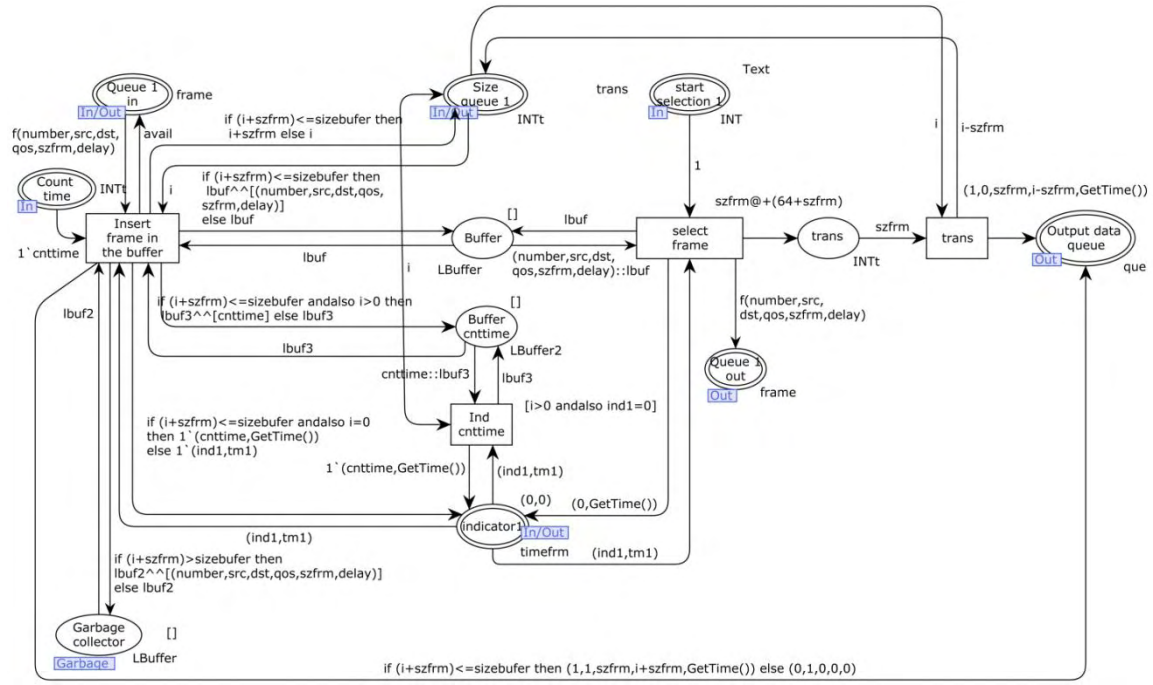


Рис. 2. Сеть Петри, моделирующая алгоритм работы памяти

При поступлении кадра в позицию ***Queue 1 in*** проверяется наличие свободного места в очереди. Если свободное место есть, то в позиции ***Size queue 1*** изменяется количество хранимых кадров на значение поля ***szfrm*** кадра (размер кадра в битах), в позицию ***Buffer*** в хвост очереди добавляется текущий кадр, Если в очереди нет свободного места, то кадр отбрасывается, помещаясь в позицию ***Garbadge collector.***

Когда в очередь поступает сигнал считывания кадра (позиция ***start selection 1***) из позиции ***Buffer*** считывается кадр, стоящий в голове очереди, и помещается в позицию ***Queue 1 out***.

Позиции ***Buffer cnttime*** и ***indicator1*** служат для формирования, хранения и выдачи специального атрибута каждого кадра, отмечающего его время поступления в очередь по временной метки позиции ***Count time***. Этот атрибут используется для временной селекцией кадров.

Позиции **Output data queue** и **Trans** совместно с одноименным переходом используются для сбора статистических данных о состоянии очереди, выполняемого следующим образом:

- при срабатывании перехода ***Insert frame in the buffer*** одновременно с помещением очередного кадра в буфер записывает в позицию ***Output data queue*** маркер с цветом ***1, 1, szfrm, i+ szfrm, getTime(),*** которая отмечает событие увеличения очереди 1 на ***szfrm*** бит в момент времени поступления кадра в очередь***;***
- при срабатывании перехода ***Select frame*** после чтения очередного кадра из буфера с задержкой ***64+ szfrm*** (время передачи кадра с учетом преамбулы [1]) в позицию **Trans** замисывается маркер, указывающий на размер переданного кадра;
- в позицию ***Output data queue*** перезод ***Trans*** записывает маркер с цветом ***1, 0, szfrm, i-szfrm, getTime(),*** которая отмечает событие уменьшения очереди 1 на величину ***szfrm*** бит в момент времени окончания передачи удаленного из очереди 1 кадра.

В результате работы модели в позиции ***Output data queue*** накапливаются маркеры, отмечающие динамику изменения очередей, последовательность значений цветов которых выводится в текстовый файл с помощью функции мониторинга, написанной на языке CPN ML. Дальнейшая обработка статистических данных очередей может выполняться любым математическим пакетом, например, Excel или MathCad.

Входной трафик моделировался с помощью специальной сети Петри [8] и представляет собой комбинацию случайного потока кадров с экспоненциальным распределением времени между кадрами и полимодальной длительностью кадров, и квазирегулярного потока одновременное поступающих кадров всех классов максимальной длинны имитирующих «пачечность». При этом варьировались такие параметры трафика, как общая интенсивность трафика, доли трафика различных классов QoS, имитируя потоки данных с параметрами трафика, которые согласованны (поток 1) и несогласованны (поток 2) с выделяемой полосой пропускания для различных классов QoS. представляет комбинацию потока кадров, принадлежащих с равной вероятностью 4 классам QoS (табл. 1).

Таблица 1

Распределение полосы пропускания между классами трафика для различных потоков

№ потока	Параметры	Классы *QoS:*			
		0–1	2–3	4–5	6–7
1	Выделяемая полоса пропускания, %	10	20	30	40
	Доля трафика	0,1	0,2	0,3	0,4
2	Выделяемая полоса пропускания, %	10	20	30	40
	Доля трафика	0,25	0,25	0,25	0,25

На рис. 2 представлен фрагмент Временной профиля изменения очереди 4 для различных алгоритмов.

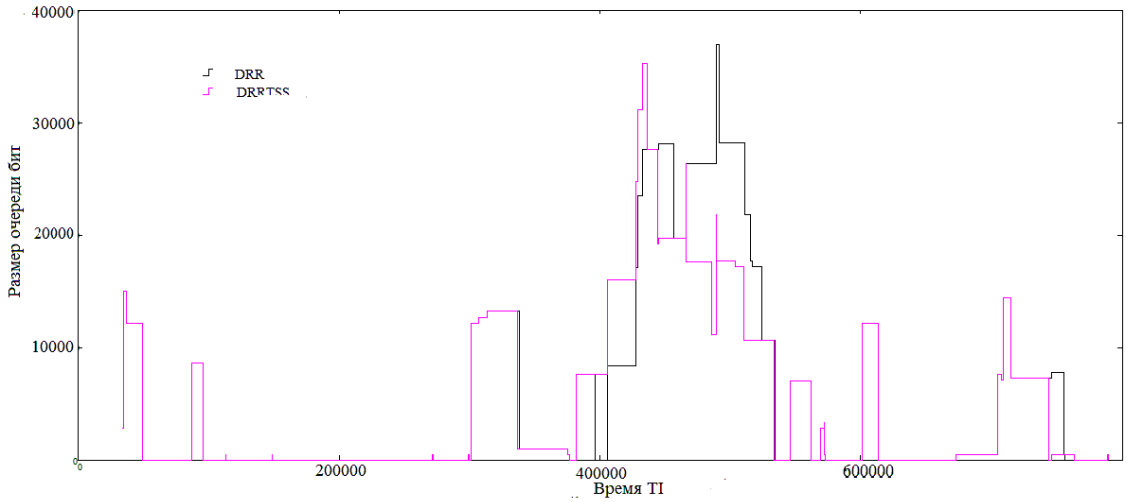


Рис. 2. Профиль изменения очередей

Результаты обработки статистической обработка данных об размерах очередей трафика различных классов QoS, полученных в результате моделирования DRR и DRR-TSS алгоритмов диспетчеризации очередей, приведены в табл. 2.

Таблица 2

Средние максимальные значения и очередей

Поток	Класс QoS	DRR		DRR–TSS	
		Максимальное значение размера очереди	Среднее значение размера очереди	Максимальное значение размера очереди	Среднее значение размера очереди
1	0–1	34136	3037	24352	2733
	2–3	70360	6865	58184	5828
	4–5	51336	8027	89576	8734
	6–7	71864	10425	71864	11047
	Общая максимальная очередь	71864		89576	
	Вариация средних очередей		7388		8314
	Средняя общая очередь		28354		28342
2	0–1	97848	10172	62264	7559
	2–3	71032	6998	71032	7486
	4–5	48192	5639	60648	6959
	6–7	59688	5548	68280	6366
	Общая максимальная очередь	97848		71032	
	Вариация средних очередей		10384		1193
	Средняя общая очередь		28357		28370

Анализ показывает, что при нагрузке, согласованной с выделенной полосой пропускания для трафика различных классов QoS алгоритм DRR-TSS приводит возрастанию вариации средних значений джиттера трафика различных классов на 12,5 % и увеличению

максимальной задержки на 25 %, но в случае рассогласования нагрузки алгоритм DRR-TSS уменьшает вариацию средних значений джиттера трафика различных классов на в 8,7 раз и снижает максимальную задержку на 28 %, при этом средний размер общей очереди практически не зависит от алгоритма диспетчеризации и вида трафика.

Таким образом, можно утверждать, что алгоритмы динамической диспетчеризации обеспечивают более комфортные условия передачи трафика в случае разбалансирования выделяемой для них полосы пропускания, чем классические циклические алгоритмы, что достигается за счет ограничения время непрерывной передачи данных при обслуживании каждой очереди и «перемешивание» кадров различных классов QoS трафика.

Библиографический список

1. Олифер, В. Компьютерные сети. Принципы, технологии, протоколы / В. Олифер, Н. Олифер. – СПб. : Питер, 2010. – 992 с.

2. Кучерявый, Е. Управление трафиком и качество обслуживания в сети Интернет / Е. А. Кучерявый. – СПб. : Наука и Техника, 2004. – 336 с.

3. Кизилов, Е. А. Моделирование адаптивной диспетчеризации очередей в коммутаторе с поддержкой QoS / Е. А. Кизилов, Н. Н. Коннов, Д. В. Патунин // Модели, системы, сети в экономике, технике, природе и обществе. – 2017. – № 2 (22). – С. 170–183.

4. Кизилов, Е. А. Эффективность «дефицитных» алгоритмов диспетчеризации очередей в коммутаторах с поддержкой QoS / Е. А. Кизилов, Н. Н. Коннов, Д. В. Патунин // Информатика: проблемы, методология, технологии : сб. материалов XVII Междунар. науч.-метод. конф. : в 5 т. – Воронеж : Вэлборн, 2017. – Т. 3. – С. 95–100.

5. Jensen, K. Coloured Petri Nets: modelling and validation of concurrent systems / K. Jensen, L. Kristensen. – Springer Science & Business Media, 2009. – P. 125.

6. Кизилов, Е. А. Моделирование коммутаторов Ethernet в промышленных сетях / Е. А. Кизилов, Н. Н. Коннов, В. Б. Механов // Новые информационные технологии и системы : тр. XI Междунар. науч.-техн. конф. – Пенза : Изд-во ПГУ, 2014. – С. 326–329.

7. Kizilov, E. Modeling of QoS in the industrial Ethernet switches / N. Konnov, D. Pashchenko, D. Trokoz // 2015 The 5th International Workshop on Computer Science and Engineering-Information Processing and Control Engineering (WCSE 2015-IPCE). – Moscow, Russia, 2015, April 15–17. – P. 185–190.

8. Никишин, К. Генератор трафика Ethernet на основе цветных сетей Петри / К. И. Никишин, Н. Н. Коннов // Модели, системы, сети в экономике, технике, природе и обществе. – 2016. – № 1 (17). – С. 299–307.

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

В. Ю. Егоров

ООО НТП «Криптософт», Пенза, Россия

ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ ОПЕРАЦИОННОЙ СИСТЕМЫ QP ОС В РАБОЧИХ СТАНЦИЯХ

Представлено описание операционной системы QP ОС для применения в качестве системы для рабочих станций. Дается общее представление о возможностях системы.

Научно-техническое предприятие «Криптософт» в течение более 20 лет разрабатывает и внедряет операционную систему QP ОС, а также вычислительные комплексы на ее основе.

В течение 2017 г. в составе QP ОС произошли существенные изменения, касающиеся, в первую очередь, внешнего вида системы. В системе появился полноценный рабочий стол, разработаны графические приложения. Теперь на рабочем столе QP ОС отображается список запущенных задач, можно размещать иконки для запуска приложений и формировать перечень необходимых программ. Также рабочий стол системы поддерживает экранные виджеты. Внешний вид системы представлен на рис. 1.

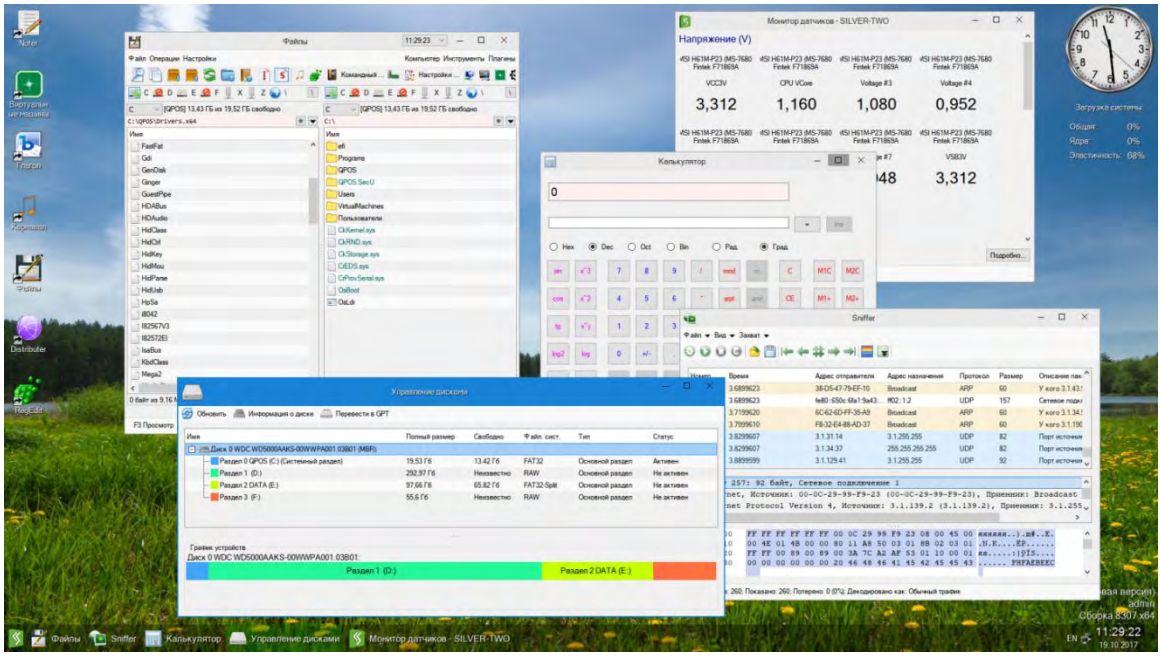


Рис. 1. Внешний вид операционной системы QP ОС

Как уже не раз отмечалось, QP ОС представляет собой универсальную операционную систему общего назначения, полностью разработанную на предприятии «Криптософт». QP ОС не является клоном какой бы то ни было другой операционной системы.

Все драйверы в составе QP ОС, как низкоуровневые, так и высокоуровневые, были написаны заново силами специалистов НТП «Криптософт». В состав драйверов также входят стеки сетевых протоколов TCP/IP 4 и 6 версий, а также протоколы SMB версий 1 и 2.

QP ОС сертифицирована ФСБ России для работы со сведениями, составляющими государственную тайну.

QP ОС обладает рядом преимуществ, позволяющих решать ответственные задачи по построению информационно-вычислительных систем. Среди них такие преимущества как:

- защищенная платформа;
- доверенная среда исполнения программ и операционных систем;
- быстродействие на уровне современных аналогов;
- комплексный подход к обеспечению безопасности.

QP ОС обладает развитой подсистемой безопасности. К компонентам безопасности в составе QP ОС относятся:

- дискреционный контроль доступа;
- мандатный контроль доступа;
- мандатное управление доверием;
- контроль целостности среды исполнения.

В настоящее время для операционной системы QP ОС разрабатывается набор офисных приложений. Одно из таких приложений – редактор документов Глагол. Программа Глагол, равно как и другие приложения в системе, поддерживает вывод документов на печать. Глагол будет активно развиваться в течение следующего года для обеспечения полноценной возможности создания документов. Внешний вид программы Глагол, а также поддержка вывода документов на печать показаны на рис. 2.

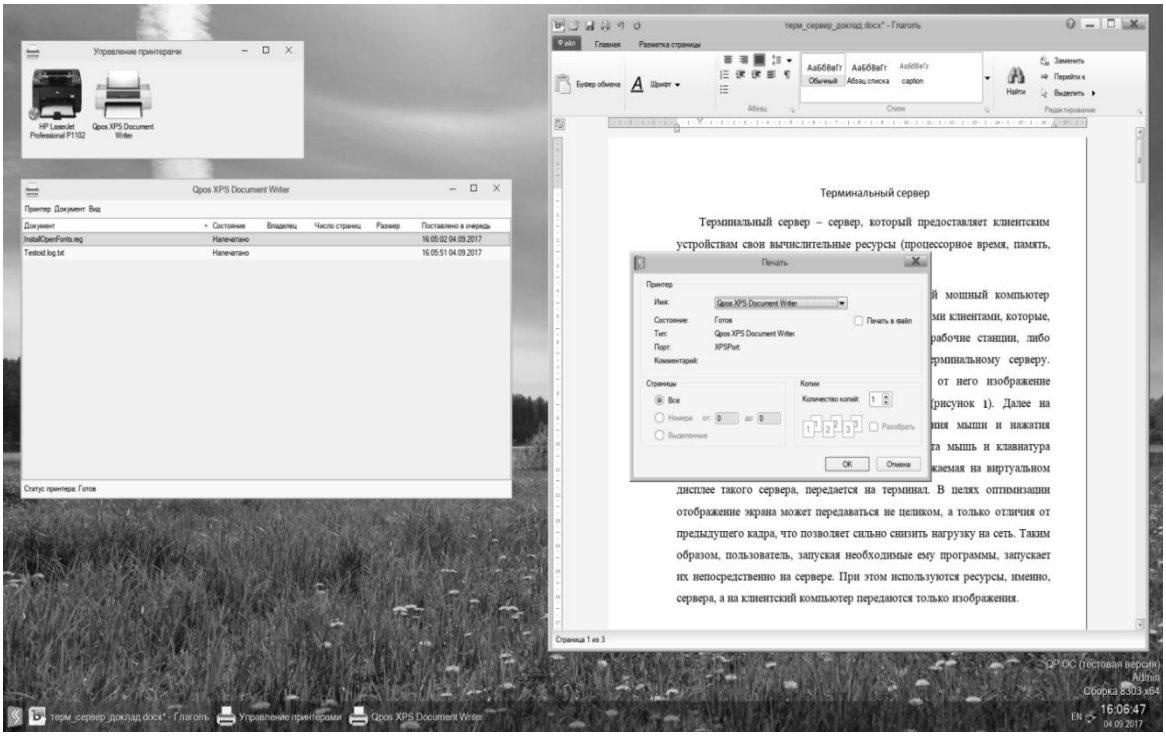


Рис. 2. Вывод документов на печать из программы Глагол

Поддержка мультимедиа в системе QP ОС реализована посредством программы Карнавал. Это медиапроигрыватель, который сейчас позволяет воспроизводить звуковые файлы различных форматов. Программа Карнавал представлена на рис. 3.

В составе операционной системы QP ОС появился браузер под названием Интернет. В настоящее время он обеспечивает базовый функционал по просмотру веб-страниц.

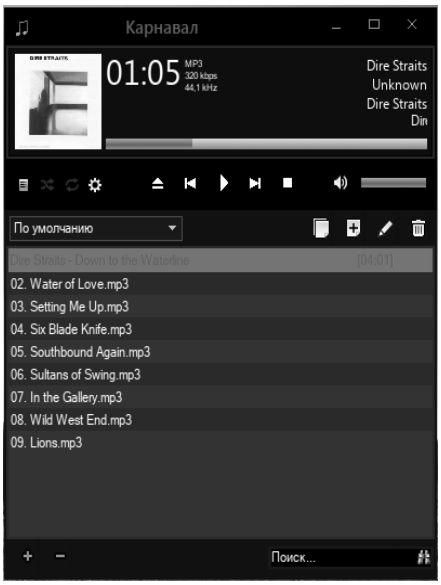


Рис. 3. Медиапроигрыватель Карнавал

Операционная система QP ОС постоянно и динамично развивается. Развитие происходит сразу по нескольким направлениям: это и дальнейшее усовершенствование ядра системы, и поддержка других аппаратных платформ, и появление новых прикладных программ. В ближайшее время по направлению применения QP ОС в качестве системы для прикладных программ планируется внедрение поддержки сканеров, просмотра видео, а также приложений для работы с презентациями и электронными таблицами.

Библиографический список

1. Официальная страница операционной системы QP ОС. – URL: http://cryptosoft.ru/qpos.html
2. Операционная система QP ОС на конференции OS DAY 2017. – URL: http://osday.ru/egorov.html#speaker#speaker

А. И. Иванов[1], А. В. Безяев[2], А. Г. Банных[3]

[1]Пензенский научно-исследовательский электротехнический институт, [2]Пензенский филиал ФГУП унитарного предприятия «Научно-технический центр «Атлас»,
[3]Пензенский государственный университет, Пенза, Россия

ХИ-КВАДРАТ ПРИБЛИЖЕНИЕ РАСПРЕДЕЛЕНИЯ ОШИБОК ВЫЧИСЛЕНИЯ ЭНТРОПИИ БИОМЕТРИЧЕСКОГО ОБРАЗА «ЧУЖОЙ» В ПРОСТРАНСТВЕ СВЕРТОК ХЭММИНГА

Целью работы является статистическое описание ошибок вычисления энтропии длинных кодов с зависимыми разрядами на малой тестовой выборке. Показано, что при переходе в пространство расстояний Хэмминга наблюдается логарифмическое сжатие размеров исходного алфавита спектра выходных состояний нейросетевой молекулы преобразователя биометрия-код. Это в конечном итоге позволяет выполнять быстрое тестирование преобразователей через вычисление их энтропии.

Обычно энтропию объекта исследования оценивают через наблюдение вероятности появления возможных состояний заранее заданного алфавита по Шеннону [1]. Когда число

состояний мало, проблем с оценкой энтропии не возникает. Этот классический подход вполне применим к оценке энтропии состояний преобразователей биометрия-код. Например, такие преобразователи за рубежом строят с использованием так называемых «нечетких экстракторов» [2–5]. Убедиться в этом можно, воспользовавшись достоверными биометрическими данными, самостоятельно получаемыми в среде моделирования «БиоНейроАвтограф» [6]. Этот продукт позволяет получать 416-ти мерные вектора биометрических параметров и соответствующие им 256 битные коды.

Проблема оценки энтропии «нечетких экстракторов» технически разрешима из-за того, что «нечеткие экстракторы» имеют короткий выходной код. Так, если в «нечетком экстракторе» при обработке 416 параметров применен код с 35 кратной избыточностью, то длина его информационной части составит не более 12 бит. Для 12 битных кодов проблем с оценкой их энтропии по Шеннону не возникает.

Однако, если воспользоваться российско-казахстанским опытом и применить нейросетевые преобразователи биометрия-код [7, 8], то длина выходного кода составит 256 бит и применить алгоритм Шеннона для вычисления энтропии станет невозможным.

Выход из этого положения состоит в переходе из пространства вероятности появления обычных кодов в пространство появления расстояний Хэмминга [9, 10]. На рис. 1 дана схема вычислений, рекомендуемая ГОСТ Р 52633.3.

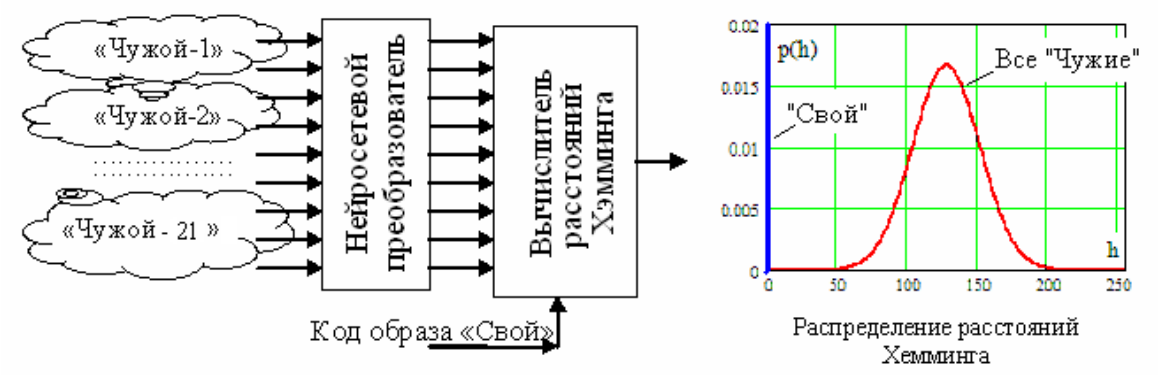


Рис. 1. Вариант схемы и результаты тестирования преобразователя биометрия-код при переходе в пространство расстояний Хэмминга

Нейронная сеть среды моделирования «БиоНейроАвтограф» [6] преобразует 416 биометрических параметров в код длиной 256 бит. То есть его выходные коды могут давать 2^{256} состояний, что не позволяет оценить вероятности их появления. Однако, если перейти в пространство расстояний Хэмминга между кодами образов «Чужой-1», «Чужой-2», …., «Чужой-21» и кодом образа «Свой», то мы получим спектр, имеющий всего 256 возможных состояний. Наблюдается логарифмическое свертывание числа возможных спектральных состояний $\log_2(2^{256}) = 256$ и, следовательно, экспоненциальное упрощение задачи оценки энтропии.

Все криптографические операции создаются таким образом, чтобы давать данные близкие к белому шуму. То есть разряды кодов независимые, а энтропия кодов точно совпадает с длиной кода. Для биометрических данных все иначе, их разряды зависимы (коррелированны). Теоретически 256 зависимых бит можно заменить на меньшее число независимых бит, например путем применения процедуры декорреляции.

Стандартная процедура вычислений по ГОСТ Р 52633.3 построена на том, что подав на нейросеть 21 образ «Чужой», получается 21 код «$\bar{x}$». Образ «Свой» дает код «$\bar{c}$», соответственно можно вычислить расстояние Хэмминга сложением по модулю два их разрядов:

$$h = 256 - \sum_{i=0}^{255} ("c_i") \oplus ("x_i") \qquad (1)$$

Далее по выборке в 21 опыт мы можем вычислить математическое ожидание – $E(h)$ и стандартное отклонение – $\sigma(h)$. Знание о двух статистических моментах позволяет оценить вероятность угадывания кода «Свой» – P_2, подставляя случайные образы «Чужой»:

$$P_2 \approx \frac{1}{\sigma(h)\sqrt{2\pi}} \int_0^1 \exp\left\{\frac{-\left(E(h)-u\right)^2}{2\left(\sigma(h)\right)^2}\right\} \cdot du. \quad (2)$$

В этом случае энтропия нейросетевого преобразователя оценивается следующим образом:

$$H("x_1, x_2, ..., x_{256}") \approx -\log_2(P_2). \quad (3)$$

Каждая достаточно сложная вычислительная процедура, как правило, накапливает ошибки исходных данных. Для оценки показателя обусловленности последовательности процедур (2), (3) необходимо использовать большую базу образов в 300 выборок по 21 образу «Чужой». В результате тестирования на таком объеме данных математическое ожидание составляет – E(h) = 107,26 бит, а его стандартное отклонение – σ{E(h)} = 6,77 бит. Это означает, что на выборке в 21 опыт относительная ошибка вычисления математического ожидания может составлять до 18,9 % при нормальном законе распределения значений.

Математическое ожидание стандартного отклонения составляет E{σ(h)} = 27,84 бит, а его стандартное отклонение σ{σ(h)} = 4,04 бит. Это означает, что стандартное отклонение на выборке в 21 опыт может оцениваться с относительной ошибкой до 43,5 % при нормальном законе распределения значений. Все это приводит к итоговым относительным ошибкам вычисления энтропии до 271 %. Мы имеем дело с плохо обусловленной задачей.

На рис. 2 приведен пример гистограммы распределения значений энтропии, вычисленной по 300 выборкам, каждая из которых состоит из 21 опыта.

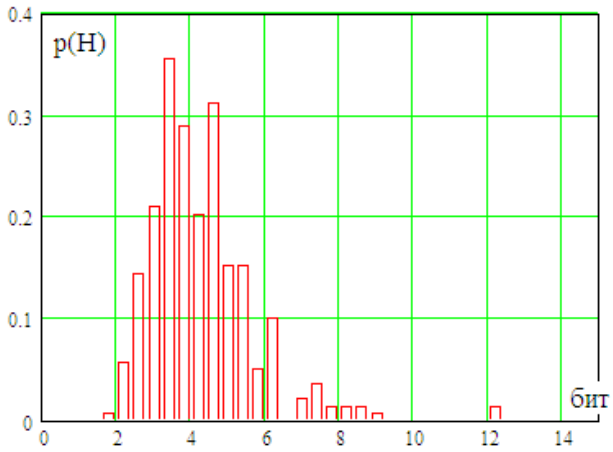


Рис. 2. Гистограмма распределения значений энтропии, вычисленной на выборке в 21 опыт

Из рис. 2 видно, что при математическом ожидании энтропии – E(H)=4.52 бита (данные численного эксперимента) минимальное значение энтропии составляет 1,8 бита, а максимальное значение энтропии составляет 12,2 бита. Максимальная оценка оказывается выше математического ожидания на 271 %, что еще раз подтверждает плохую обусловленность решаемой задачи. Относительная ошибка вычислений нелинейно усиливается и оказывается много больше, чем сумма средних значений максимумов относительных ошибок исходных данных:

$$cond(H) \approx \frac{2\max(\left|E(H)-H_i\right|)}{(\max(\left|E(h)-h_i\right|)+\max(\left|E(\sigma(h))-\sigma(h_i)\right|)} \approx 8{,}96. \quad (4)$$

Наблюдается почти 9-ти кратное усиление ошибок исходных данных при стандартных вычислениях по ГОСТ Р 52633.3 [10] на малой выборке в 21 опыт. Для снижения столь

существенного эффекта накопления ошибок на порядок потребуется увеличить объем тестовой выборки на 2 порядка до 2100 опытов.

В силу того, что число обусловленности вычисления энтропии в пространстве сверток Хэмминга оказывается высоко (4) необходимо проводить регуляризацию подобных вычислений [11–13]. Для этой цели необходима априорная информация о законе распределения ошибок вычисления энтропии по формуле (3).

Проведенный численный эксперимент для 1000 выборках по 8 опытов, показал, что закон распределения плотности вероятности значений близок к хи-квадрат распределению. Соответствующие графики распределений приведены на рис. 3.

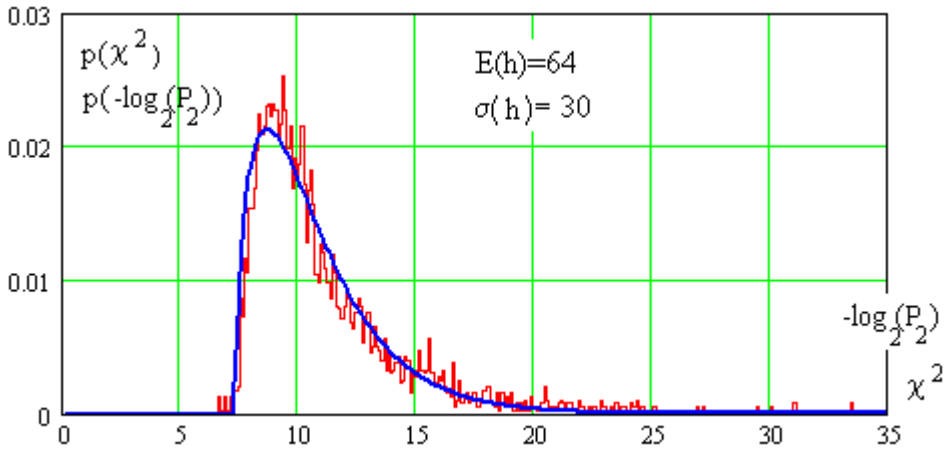


Рис. 3. Распределение вероятности ошибок вычисления энтропии на малой выборке из 8 опытов для нормального закона распределения значений расстояний Хэмминга с математическим ожиданием $E(h) = 64$ и стандартным отклонением $\sigma(h) = 30$

Из рис. 3 видно, что реальное распределение ошибок (кривая с шумом) хорошо описывается смещенным на величину 7,278 бит хи-квадрат распределением с числом степеней свободы $m = 3{,}6$ (сплошная линия).

Таким образом, можно сделать вывод, что плотность распределения ошибок вычисления энтропии на малых выборках хорошо описывается хи-квадрат распределениями с дробным (фрактальным) числом степеней свободы. Пользуясь этим, можно дать достаточно точное аналитическое описание распределения ошибок вычислений, которое позднее может быть использовать на следующих этапах регуляризации.

Библиографический список

1. Яглом, А. М. Вероятность и информация / А. М. Яглом, И. М. Яглом. – М. : ДомКниги, 2007. − 512 с.

2. Juels, A. Fuzzy Commitment Scheme / A. Juels, M. A. Wattenberg // Proc. ACM Conf. Computer and Communications Security. – 1999. − P. 28–36.

3. Monrose, F. Cryptographic key generation from voice / F. Monrose, M. Reiter, Q. Li, S. Wetzel // In Proc. IEEE Symp. on Security and Privacy. – 2001.

4. Dodis, Y. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy / Y. Dodis, L. Reyzin, A. Smith // In EUROCRYPT. – 2004. − P. 523–540.

5. Hao, F. Crypto with Biometrics Effectively / Feng Hao, Ross Anderson, John Daugman // IEEE TRANSACTIONS ON COMPUTERS. – 2006. – Vol. 55, № 9, September

6. Иванов, А. И. Среда моделирования «БиоНейроАвтограф» / А. И. Иванов, О. С. Захаров. – 2009. – URL: http://пниэи.рф/activity/science/noc.htm

7. Волчихин, В. И. Быстрые алгоритмы обучения нейросетевых механизмов биометрико-криптографической защиты информации : моногр. / В. И. Волчихин, А. И. Иванов, В. А. Фунтиков. – Пенза : Изд-во ПГУ, 2005. − 273 с.

8. Технология использования больших нейронных сетей для преобразования нечетких биометрических данных в код ключа доступа : моногр. / Б. С. Ахметов, А. И. Иванов, В. А. Фунтиков, А. В. Безяев, Е. А. Малыгина. – Казахстан, Алматы : ТОО «Изд-во LEM», 2014. − 144 с. – URL: http://portal.kazntu.kz/files/publicate/2014-06-27-11940.pdf

9. Малыгин, А. Ю. Быстрые алгоритмы тестирования нейросетевых механизмов биометрико-криптографической защиты информации : моногр. / А. Ю. Малыгин, В. И. Волчихин, А. И. Иванов, В. А. Фунтиков. – Пенза : ПГУ, 2006. − 161 с.

10. ГОСТ Р 52633.3–2011 Защита информации. Техника защиты информации. Тестирование стойкости средств высоконадежной биометрической защиты к атакам подбора.

11. Тихонов, А. Н. Методы решения некорректных задач / А. Н. Тихонов, В. Я. Арсенин. – М. : Наук, 1984. − 256 с.

12. Бабушинский, А. Б. Некорректные задачи. Численные методы и приложения / А. Б. Бабушинский, А. В. Гончарский. – М. : Изд-во Моск. ун-та, 1989. − 199 с.

13. Васин, В. В. Некорректные задачи с априорной информацией / В. В. Васин, А. Л. Агеев. – Екатеринбург : УИФ «Наука», 1993.− 264 с.

В. И. Волчихин[1], А. И. Иванов[2], С. Е. Вятчанин[1]

[1]Пензенский государственный университет,
[2]Пензенский научно-исследовательский электротехнический институт, Пенза, Россия

ОСОБЕННОСТИ ОБУЧЕНИЯ СЕТЕЙ ВЕРОЯТНОСТНЫХ НЕЙРОНОВ «КРАМЕРА – ФОН МИЗЕСА» НА МАЛЫХ БИОМЕТРИЧЕСКИХ ВЫБОРКАХ

Синтезированный искусственный нейрон «Крамера – фон Мизеса» унаследовал от квадратичных форм устойчивость алгоритма обучения. Сети нейронов «Крамера – фон Мизеса» дают выходной код для образов «Чужой» с низкой энтропией из-за отсутствия баланса состояний «0» и «1» в его разрядах. Этот недостаток предложено устранить путем применения выходного квантователя на выходе сумматоров нейронов «Крамера-фон Мизеса» с тремя выходными состояниями. Это позволяет рассматривать данные сети нейронов как перспективу для следующего поколения нейросетевых преобразователей биометрия-код.

Существует множество статистических критериев для поверки гипотезы нормального распределения значений биометрических данных. Наиболее часто на практике используется хи-квадрат критерий [1] в силу того, что для этого критерия Пирсон в 1900 г. построил аналитическое описание хи-квадрат распределения. К сожалению, хи-квадрат критерий оказывается работоспособен только на выборках в 200 опытов и больше. При обработке биометрических данных приходится иметь выборки в 20 примеров, что мешает применять хи-квадрат критерий.

Такая же ситуация возникает и случаи использования других не параметрических статистических критериев [2], все они так же дают надежные оценки при выборках существенно более 20 примеров.

Для биометрии данные одного контролируемого параметра примеров образа «Свой» имеют распределение близкое к нормальному. Для данных случайно выбранных образов «Чужие» распределение оказывается близко к равномерному. В связи с этим мы можем исследуемые критерии оценивать по отношению к друг другу сравнивая между собой, даваемые ими равные ошибки первого и второго рода $P_1 = P_2 = P_{EE}$ при использовании малых выборок нормального и равномерного законов распределения. На рис. 1 приведены функции снижения вероятностей ошибок по мере роста размеров тестовой выборки. Из рис. 1 видно, что мощность критерия KfM при малых выборках оказывается выше, чем у хи-квадрат критерия. Это делает критерий KfM более перспективным для биометрии в сравнении с хи-квадрат критерием, критерием Фроцини и критерием Джини.

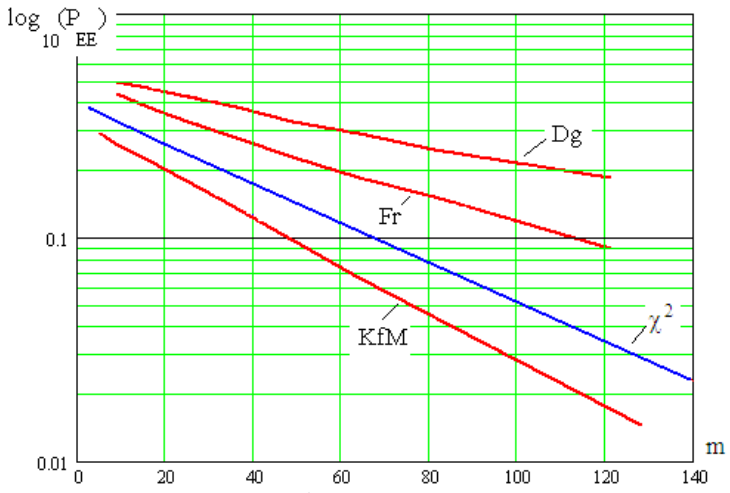


Рис. 1. Сопоставление мощностей интегральных статистических критериев Джини (Dg), Фроцини (Fr), хи-квадрат (χ2) и Крамера – фон Мизеса (KfM)

Видимо основной причиной высокой мощности «Крамера – фон Мизеса» (KfM) критерия является то, что у него оказываются минимальные шумы квантования континуальных данных.

Исходный критерий «Крамера-фон Мизеса» вычисляется через интеграл квадрата разницы между гипотетической функцией вероятности – $P(\nu)$ эмпирической функции вероятности $\hat{P}(\nu)$:

$$KfM = \int_{-\infty}^{+\infty} (P(v) - \hat{P}(v))^2 dv. \tag{1}$$

Формальный переход от интеграла (1) к операции суммирования нейроном с n входами не сложен:

$$\begin{cases} KfM(\nu) = \sum_{i=1}^{n} (P(v_i) - \hat{P}(v_i))^2 \\ KfM(\xi) = \sum_{i=1}^{n} (P(v_i) - \hat{P}(\xi_i))^2 \end{cases}, \tag{2}$$

где ν_i пример образа «Свой» по n-контролируемым нейроном биометрическим параметрам; ξ_i пример образа «Чужой» по n-контролируемым нейроном биометрическим параметрам.

Эталонная функция вероятности – $P(\nu)$ оказывается легко вычислима только в том случае, когда выполнено центрирование и нормирование, контролируемых биометрических параметров образа «Свой»:

$$\tilde{\nu}_i = \frac{\nu_i - E(\nu_i)}{\sigma(\nu_i)}. \tag{3}$$

В новой центрированной и нормированной системе биометрических координат эталонная функция вероятности имеет нулевое математическое ожидание и единичное стандартное отклонение:

$$P_\nu(\tilde{v}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\tilde{v}} \exp\left\{\frac{-u^2}{2}\right\} du. \tag{4}$$

На рис. 3 дан пример эталонной функции вероятности для нормального закона.

Из рис. 3 видно, что данные примеров образа «Свой» после их преобразования в вероятность должны оказываться в интервале от 0.0 до 1.0. Причем появление предельных значений 0.0 и 1.0 маловероятно.

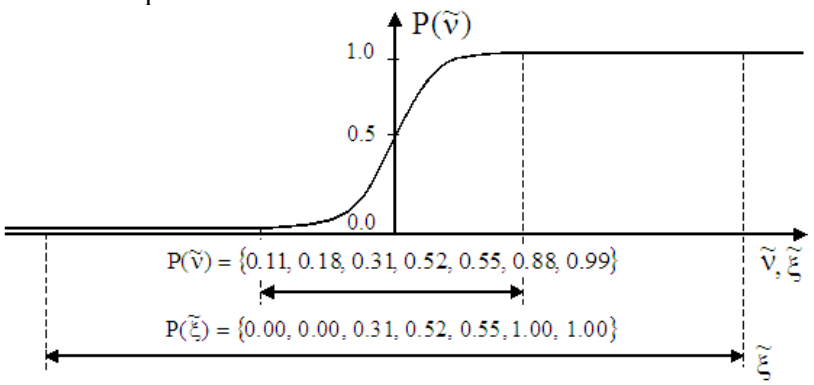


Рис. 3. Работа функции преобразования вектора значений переменных в вектор значений их вероятности

Совершенно иная ситуация возникает, если вероятностному нейрону «Крамера – фон Мизеса» будут предъявлены данные примера образа «Чужой». Как видно из рис. 3 динамический диапазон данных – $\tilde{\xi}_i$ примерно в три раза больше, чем данных образа «Свой». Это означает, что на входы нейрона KfM будут поступать несколько предельных минимальных значений вероятности {0.0, 0.0,….} в начале упорядоченного списка и несколько предельных значений в конце упорядоченного списка {….., 1.0, 1,0}.

В целом вероятностный нейрон KfM работает за счет того, что выявляет неоправданно большое число предельных состояний вероятности в наблюдаемом биометрическом образе. Нейрон KfM и структура нейронной сети, сформированной из нейронов KfM представлены на рис. 4.

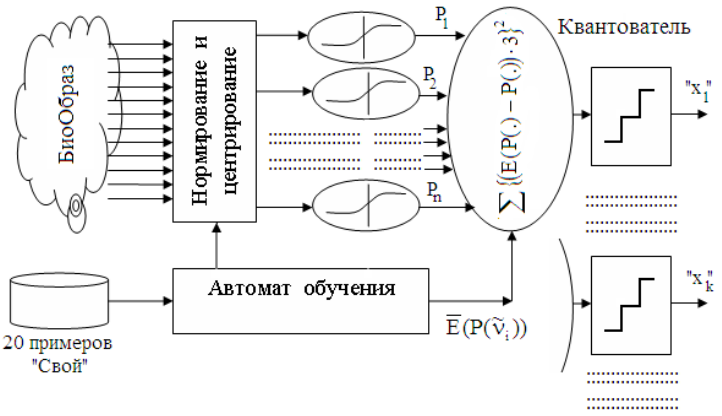


Рис. 4. Обобщенная структура нейросетевого преобразователя биометрических данных в код, построенная на использовании вероятностных нейронов «Крамера – фон Мизеса»

При использовании трехуровнего квантования [3, 4] (см. рис. 4) вероятности состояний «0» и «1» на выходе разрядов сети нейронов KfM выравниваются. Наблюдается рост энтропии выходных кодов «Чужой» до величины порядка 45 бит, что в полтора раза больше, чем тот же показатель для линейных нейронов, обученных по ГОСТ Р 52633.5 [5].

Библиографический список

1. Р 50.1.037–2002 Рекомендации по стандартизации. Прикладная статистика. Правила проверки согласия опытного распределения с теоретическим. Ч. I. Критерии типа χ2. – М., 2001. − 140 с.

2. Р 50.1.037–2002 Прикладная статистика. Правила проверки согласия опытного распределения с теоретическим. Ч. II. Непараметрические критерии. – М., 2002. − 123 с.

3. ГОСТ Р 52633.5–2011 Защита информации. Техника защиты информации. Автоматическое обучение нейросетевых преобразователей биометрия-код доступа. – М., 2011.

А. И. Иванов[1], Е. А. Малыгина[2]

[1]Пензенский научно-исследовательский электротехнический институт,
[2]Пензенский государственный университет, Пенза, Россия

ОСОБЕННОСТИ МНОГОМЕРНОЙ СТАТИСТИЧЕСКОЙ ОБРАБОТКИ БИОМЕТРИЧЕСКИХ ДАННЫХ

Использование многомерной статистической обработки малых выборок биометрических данных позволяет получать решения с достаточно высокой достоверностью.

Популярность использования хи-квадрат критерия Пирсона [1] во многом обусловлена тем, что при $n\to\infty$ его распределение описывается через гамма функцию с $m = k–1$ числом степеней свободы:

$$p_{\chi^2}(n=\infty, m=k-1, x)=\frac{1}{2^{\frac{m}{2}}\cdot\Gamma\left(\frac{m}{2}\right)}\cdot x^{\frac{m}{2}-1}\cdot e^{\frac{-x}{2}}.$$

Применение хи-квадрат критерия для многомерных и сильно коррелированных биометрических данных дает неудовлетворительные результаты [2].

Так для принятия решений с уровнем доверия 0,99 придется использовать выборку, состоящую из 400 результатов испытаний. Применение столь больших тестовых выборок недопустимо для биометрии, поэтому необходимо добиться их снижения примерно на порядок.

Известно, что биометрические данные многомерны [3]. Например, нейросетевой преобразователь биометрия-код [3], преобразует 416 рукописных биометрических параметров в код личного ключа длиной 256 бит. То есть имеется возможность анализировать не один, а 416 биометрических параметров написания слова-пароля. При этом можно увеличить объем обрабатываемых данных и повысить достоверность принимаемых решений.

Для многомерной обработки воспользуемся сложением частных критериев Пирсона:

$$\chi^2(\nu_1,\nu_2,...,\nu_q)=\frac{\chi^2(\nu_1)+\chi^2(\nu_2)+...+\chi^2(\nu_q))}{q}. \quad (1)$$

Преобразование (1) эквивалентно использованию сети частных критериев Пирсона. Сеть частных хи-квадрат критериев Пирсона имеет входные и выходные нелинейные преобразования при линейном суммировании данных между ними.

При переходе к моделированию сети частных критериев хи-квадрат Пирсона достаточно вместо двух программных генераторов псевдослучайных чисел использовать q пар генераторов. При малой размерности входных данных $q \le 16$ особых сложностей в программировании численного эксперимента не возникает.

При многомерной статистической обработке биометрических данных отпадает необходимость вычисления многомерной корреляционной матрицы.

С ростом размерности решаемой задачи все более актуальным становится процедура ее симметризации. В нейросетевой биометрии, принято [4] осуществлять симметриза-

цию влияния корреляционных связей через вычисление математического ожидания модулей парных коэффициентов корреляции. Для этого берется несколько сотен пар случайно выбранных биометрических параметров. При любой размерности задачи, удается заменить реальные данные их некоторым эквивалентом, имеющим одинаковые корреляционные связи между всеми заданными параметрами. Такое упрощение задачи позволяет легко оценить эквивалентный показатель размерности:

$$\tilde{q} \approx (q-1)\cdot(1-R^2)+1. \qquad (2)$$

Чем выше корреляция данных, тем меньше оказывается эквивалентная размерность, обработки статистических данных.

Рост размерности обработки и рост коррелированности данных работают в противоположных направлениях. Всегда можно понизить размерность задачи, компенсируя тем самым влияние коррелированности данных. То есть, пользуясь преобразованиями вида (2) вполне возможно свести задачу применения многомерных сетей Пирсона к зависимым данным к более простой задаче обработки независимых данных.

Соответствующие таблицы пересчета уже созданы для искусственных нейронных сетей и опубликованы в работе [4]. Как следствие, аналогичные таблицы преобразований могут быть построены и для сетей частных критериев Пирсона.

Библиографический список

1. Р 50.1.037–2002 Рекомендации по стандартизации. Прикладная статистика. Правила проверки согласия опытного распределения с теоретическим. Ч. I. Критерии типа χ2. – М., 2001. − 140 с.

2. Ахметов, Б. С. Оценка рисков высоконадежной биометрии : моногр. / Б. С. Ахметов, Д. Н. Надеев, В. А. Фунтиков, А. И. Иванов, А. Ю. Малыгин. – Алматы : КазНТУ, 2014. − 108 с.

3. Иванов, А. И. Многомерный статистический анализ качества биометрических данных на предельно малых выборках с использованием критериев среднего геометрического, вычисленного для анализируемых функций вероятности / А. И. Иванов, К. А. Перфилов, А. Ю. Малыгин // Измерения. Мониторинг. Управление. Контроль. – 2016. – № 1. – С. 64–72.

4. «БиоНейроАвтограф» – среда моделирования больших искусственных нейронных сетей, преобразующих данные рукописных образов в код пароля. – URL: http://пниэи.рф/activity/science/noc.htm

5. Ахметов, Б. С. Алгоритмы тестирования биометрико-нейросетевых механизмов защиты информации / Б. С. Ахметов, В. И. Волчихин, А. И. Иванов, А. Ю. Малыгин. – Казахстан, Алматы : КазНТУ им. Сатпаева, 2013. − 152 с. – URL: http://portal.kazntu.kz/files/publicate/2014-01-04-11940.pdf

К. М. Фролов[1], А. В. Дубравин[2]

[1]Пензенский государственный университет, Пенза, Россия
[2]ООО НТП «Криптософт», Пенза, Россия

ГОРИЗОНТАЛЬНОЕ МАСШТАБИРОВАНИЕ ДАННЫХ В СОВРЕМЕННЫХ СУБД

Приведены возможности распределенных систем, проблемы, возникающие при их внедрении, методы их разрешения. Также приведены данные о возможностях реализации горизонтального масштабирования на базе разрабатываемой в ООО НТП «Криптософт» системы управления базами данных.

Введение

С ростом любой компании, эксплуатирующей сервера баз данных, объем хранимой информации и нагрузка на систему увеличивается. При достижении определенной нагрузки, сервер базы данных перестает справляться с поставленным задачами. Одним из путей

решения является оптимизация архитектуры самой базы, внедрение дополнительных индексов или увеличение мощности ЭВМ, эксплуатируемой в качестве сервера. Однако все эти методы не помогут, если масштабы роста нагрузки на систему будут чересчур высокими. В этом случае на помощь может прийти горизонтальное масштабирование базы данных, то есть увеличение числа ЭВМ, решающих поставленную задачу. Выгода, достигаемая за счет использования такого рода архитектуры, очевидна – возможность повышения производительности системы путем добавления новой узловой ЭВМ и синхронизации данных с другими узлами. Пример распределенной системы представлен на рис. 1.

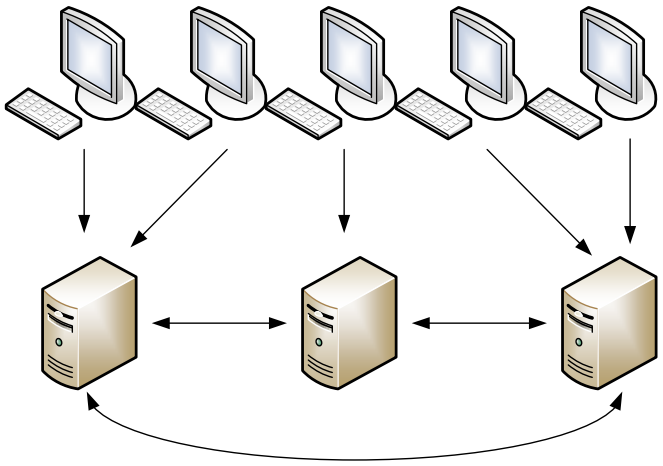


Рис. 1. Пример распределенной системы

Однако и недостатков такая система не лишена. Один из этих недостатков заключается в невозможности обойти CAP теорему, которая оперирует тремя свойствами распределенной системы:

- consistency (согласованность) предполагает, что, если данные успешно записаны в распределенное хранилище, то любой клиент при запросе получит именно актуальные данные;
- availability (доступность) указывает, что клиент в любой момент может получить данные из распределенного хранилища или сообщение об отсутствии данных, если они не были сохранены;
- partition tolerance (устойчивость к разделению системы) предполагает, что при потери сообщений между компонентами или выходе компоненты системы из строя сама система не теряет работоспособность.

В CAP теореме говорится, что при построении распределенных систем, можно удовлетворить только 2 из 3 вышеперечисленных свойств системы.

В реальных системах даже достижение 2 критериев из 3 весьма затруднительно, поскольку при разработке системы сказывается также и человеческий фактор. Сама CAP теорема лишь частично применима к современным распределенным системам [1]. В реальных системах ни один разработчик не станет жертвовать свойством устойчивости к разделению системы, поскольку в противном случае смысл распределенной архитектуры теряется. Таким образом при разработке распределенных систем пожертвовать можно только доступностью или согласованностью и реализовать оставшуюся часть.

Примером распределенных систем, в которых сочетаются свойства доступности к данным и устойчивости системы (AP-системы), является множество NoSQL решений. Считается, что в таких базах данных целостность данных не гарантирована. В случае таких систем обеспечивается так называемый целесообразный уровень целостности данных. Распределенные базы данных, построенные с помощью NoSQL решений, основываются на подходе, при котором требования к доступности и целостности данных выполняются не в полной мере. Такой подход получил акроним BASE (Basically Available, Soft-state, Eventually consistent) – базовая доступность, неустойчивое состояние, согласованность в конечном

счете. Этот подход является прямой противоположностью принципу ACID, который используется практически во всех современных реляционных базах данных.

Возможности реализации распределенной архитектуры

Для понимания возможностей реализации распределенной системы на базе разрабатываемой в ООО НТП «Криптософт» СУБД на рис. 2 приведена архитектура этой СУБД.

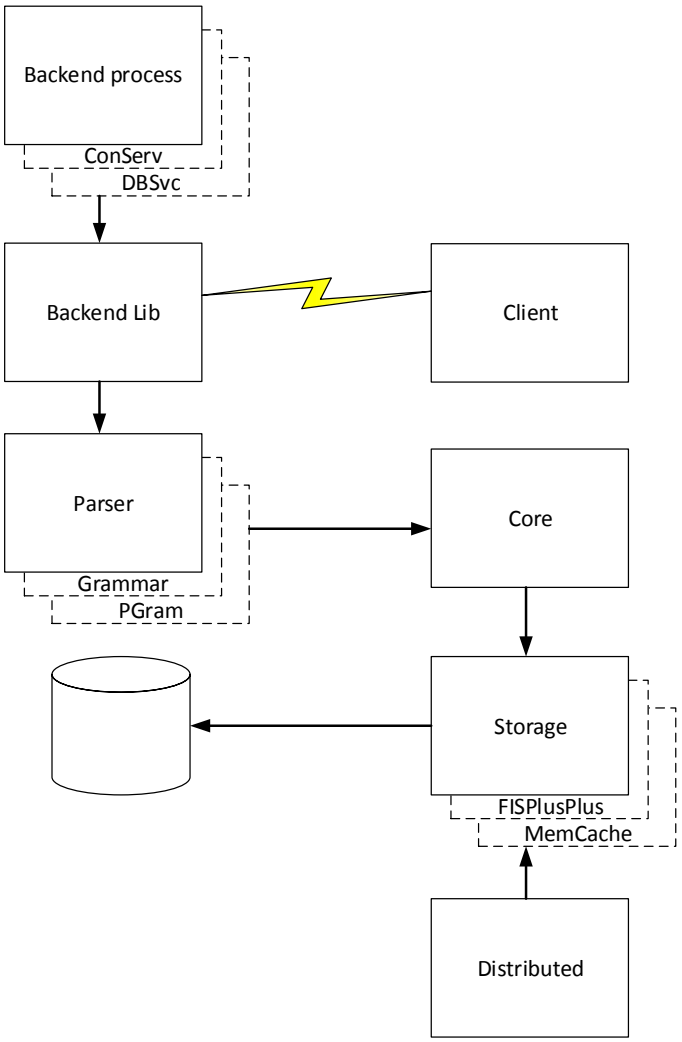


Рис. 2. Архитектура разрабатываемой СУБД

Так как в данной СУБД поддерживается работа с различными хранилищами, то добавление новых компонентов на этом уровне в архитектуру происходит безболезненно за счет использования интерфейсов. Для реализации возможности создания БД с распределенной архитектуры планируется провести декомпозицию задачи поддержания согласованного состояния данных в БД и решать отдельные подзадачи с помощью выделенных модулей.

Для решения проблемы отказоустойчивости и поддержания согласованности данных будет необходимо реализовать модуль поиска консенсуса, который должен будет поддерживать согласованное состояния данных во всей системе. Для поддержания согласованности данных на физическом уровне в разрабатываемой СУБД также планируется ввести новый тип хранилища, которое будет являться распределенным и защищенным. Кроме того, на каждом сервере БД в системе должны быть запущены собственные бэкенд-процессы (ConServ или DBSvc), поэтому потребуется внести в соответствующие модули

коррективы, учитывающие возможность использования соответствующих приложений в распределенной архитектуре.

Задача достижения консенсуса

Для понимания устройства распределенной базы данных на рис. 3 приведена схема типовой распределенной БД, являющейся AP-системой.

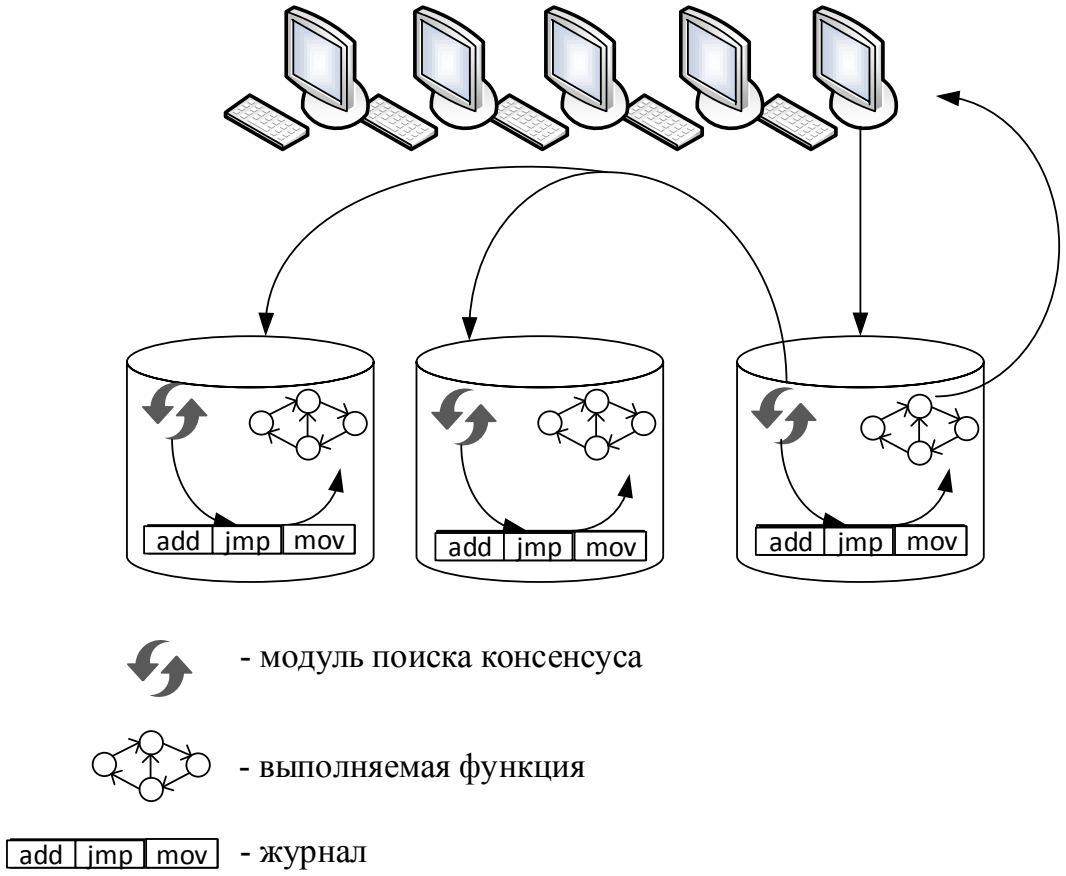


Рис. 3. Пример распределенной базы данных

Чтобы достигнуть свойства partition tolerance, при организации распределенной системы нужно принимать во внимание фактор отказов оборудования. Представим себе распределенную систему, состоящую из нескольких ×86 серверов. Для одного сервера вероятность отказа довольно мала, но в случае распределенной системы мы будем иметь дело с множеством серверов, а, следовательно, вероятность отказа растет в несколько раз [2].

Помимо вероятности отказа самих серверов также нужно закладывать вероятность отказа отдельных компонент, которые могут привести к несогласованности данных. На основе статистики, собранной в Google, утверждалось, что для кластера из 1800 машин имеет место быть порядка 1000 отказов в первый год эксплуатации, т.е. около 3 отказов в день. Таким образом, если отказоустойчивость не закладывается в ПО распределенной системы, то после разработки будет иметься система, в которой каждая из описанных причин может привести к отказу всей системы.

Задача достижения консенсуса заключается в получении согласованного значения группой участников распределенной системы, когда возможны отказы отдельных ее участков, предоставление этими участками некорректной информации или искажение информации самой средой передачи данных. Решается задача достижения консенсуса с помощью алгоритмов поиска консенсуса, наиболее популярными из которых являются PAXOS и RAFT.

Заключение

Горизонтального масштабирования данных в современных реалиях является одним из ключевых компонентов любой конкурентоспособной СУБД. Таким образом, реализация алгоритмов и компонентов, обеспечивающих горизонтальное масштабирование в СУБД, является актуальной задачей.

Библиографический список

1. Ефимов, С. Оптимизация структуры распределенных вычислительных систем / С. Ефимов. – М. : Lambert Academic Publishing, 2012. – 108 с.
2. Таненбаум, Э. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. Ван Стеен. – СПб. : Питер, 2003. – 880 с.

В. В. Кормишина

Пензенский государственный университет, Пенза, Россия

ПРИМЕНЕНИЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ В КРИПТОГРАФИИ

Объектом исследования является возможность применения теории эллиптических кривых при реализации криптографических протоколов (сеансов) с использованием слабозащищенных, потенциально доступных для взлома каналов связи. В этом случае возникает необходимость дополнительной защиты передаваемой информации, особенно, если эта информация является изображением, и именно эллиптическая криптография создает возможность ее обеспечения. Были решены модельные задачи для различных изображений различной природы и цветовой насыщенности

В конце прошлого века на стыке теории алгоритмов, чисел и компьютерной алгебры зародилось и в настоящее время бурно развивается направление, известное нам как криптография с открытым ключом. Данный раздел криптографии позволяет успешно решать различные задачи, связанные с защитой информации, в компьютерных сетях.

В последние годы активное развитие получила так называемая криптография эллиптических кривых, которая была открыта независимо Нилом Коблицем и Виктором Миллером в 1985 году (также используется термин «эллиптическая криптография»). Роль основной криптографической функции (операции) здесь выполняет скалярное умножение точки эллиптической кривой на константу, которая, в свою очередь, определяется через операции сложения и удвоения точек эллиптической кривой. Последние выполняются на основе операций умножения, сложения и инвертирования в конечном поле, над которыми рассматривается кривая.

Криптография в современном мире имеет огромное значение, в частности от того, что ресурсы становятся доступными и многие из них требуют более сильной защиты. Постоянно меняются методы криптографии, однако и виды криптографических атак становятся разнообразными и позволяют за короткое время взломать практически любые данные. Мы должны разрабатывать более устойчивые методы для защиты данных, чтобы сохранить информацию и безопасно передать ее.

Эллиптическая криптография – принципиально новое направление, и ее применение в беспроводных коммуникациях имеет преимущества, которыми обусловлен особый интерес к этому направлению: небольшая длина ключа и высокое быстродействие.

Эллиптической кривой над полем F будем называть гладкую кривую, задаваемую уравнением вида

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6, a_i \in F. \quad (1)$$

Далее для произвольного поля будем использовать следующее определение: гладкая кривая, задаваемая уравнением вида (1), называется эллиптической кривой $E\varepsilon$ над полем F.

На множестве $E(F)$, состоящем из точек эллиптической кривой (1) и еще одного элемента – бесконечно удаленной точки, можно определить операцию, которая обладает свойствами операций группы, называемой абелевой [1, с. 90]. Уточним, что формально эта точка пока не являются точкой кривой. Принято рассматривать как аддитивную получающуюся при этом группу, а операцию называть операцией сложения и привычно обозначать знаком плюс. Упомянутая дополнительная точка выступает в качестве нейтрального элемента (при аддитивной записи – это ноль) этой группы и обозначается O.

Если K – расширение поля F, то $E(K)$ обозначает множество точек $(x, y) \in K^2$, удовлетворяющих (1), вместе с точкой O. Будем использовать обозначение $E(K)$ только в тех случаях, когда в рассмотрении участвует не только поле F, но и его расширение K.

По определению полагаем для любой точки $(x, y) \in E(F)$

Общая схема алгоритма сложения точек $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ в соответствии с определением операции сложения в группе точек эллиптической кривой представляется в следующем виде.

Эта страница представляет собой шаблон оформления текста доклада. Текст оформляется следующими стилями:

Алгоритм 1. Общая схема сложения и удвоения точек
ВХОД: Коэффициенты эллиптической кривой,
точки $P_1 = (x_1, y_1)$ (или $P_1 = O$) и $P_2 = (x_2, y_2)$ (или $P_2 = O$).
ВЫХОД: $P = P_1 + P_2$,
Вычислить: если $P_1 = O$, то $P = P_2$,
если $P_2 = O$, то $P = P_1$,
если $P_2 = -P_1$, то $P = O$,
если $x_1 \neq x_2$, то $P_2 = -(x_3, y_3)$,
иначе $P = 2P_1 = -(x_3, y_3)$.
Вернуть P.

В данный алгоритм включен случай удвоения точки (последняя строка), когда слагаемые не взаимно противоположны, но одинаковы. У некоторых кривых есть «самопротивоположные» точки (такие, что $P = -P$), правило удвоения к таким точкам не применимо, и в этом случае алгоритм, не доходя до последней строки, заканчивает работу.

Для группы точек эллиптической кривой проблема дискретного логарифма заключается в том, чтобы по известным точке P данной эллиптической кривой и точке $Q = kP$, определить числа k. Сложность этой проблемы не менее сложности DLP для поля, над которым определена кривая. Для эллиптических кривых неизвестны алгоритмы решения DLP, полиномиальные по сложности, но для суперсингулярных известен алгоритм сведения этой проблемы к DLP некоторого расширения невысокой степени поля, над которым определена кривая.

Классическая проблема Диффи – Хеллмана формулируется применительно к мультипликативной группе конечного поля. Она заключается в вычислении элемента α^{xy} по элементам α, α^x и α^y. Неизвестно, возможно ли ее решить не производя предварительные вычисления индексов x и y, т.е. минуя проблему дискретного логарифма. Говоря о циклической подгруппе группы точек эллиптической кривой, то применительно к ней эта проблема заключается в том, чтобы при известных точке $P \in E(F)$ и двух ее кратных k_AP и k_BP найти точку k_Ak_BP. Также неизвестно, можно ли это сделать без предварительного вычисления констант k_A и k_B, т.е. не решая DLP для эллиптических кривых.

Рассмотрим распределение ключей для классической криптосистемы (протокол Диффи – Хеллмана) [4, с. 102]. Стоит отметить, что в качестве ключа классической криптосистемы можно использовать неизвестную посторонним (секретную) случайную точку $(x, y) \neq O$ группы точек эллиптической кривой $E(F)$, если условиться, каким образом конвертировать ее в натуральное число, например, одну из координат, скажем, x считать двоичной записью натурального числа.

Модификацию протокола Диффи – Хеллмана можно использовать для получения такой секретной точки на двух терминалах открытого канала связи.

Допустим, что E – эллиптическая кривая и P – предварительно согласованная и опубликованная точка этой кривой. Абонент A выбирает, сохраняя в секрете случайное число k_A (секретный ключ A), вычисляет координаты точки k_AP (свою «половинку» ключа) и пересылает их абоненту В. Аналогично В выбирает секретный ключ k_B, вычисляет и пересылает абоненту A «половинку» k_BP ключа. Общим ключом является точка $P = k_Ak_BP$. A вычисляет ее, умножая на свой секретный ключ k_A на «половинку» ключа, вычисленную B, а B вычисляет эту же точку, умножая сообщение, поступившее от A, на свой секретный ключ k_B. Ввиду того, что группа точек эллиптической кривой абелева, результат не зависит от порядка вычисления и, следовательно, A и B имеют координаты секретной точки $k_A(k_BP) = k_B(k_AP) = k_Ak_BP = (x, y)$ и могут использовать x в качестве ключа симметричной криптосистемы (при условии достаточности длины этой двоичной записи, что зависит от порядка поля, над которым построена эллиптическая кривая, и также, если секретные ключи k_A и k_B были выбраны как случайные или как криптографически стойкие псевдослучайные числа). Теперь оба абонента A и B имеют одинаковые копии искомой секретной точки эллиптической кривой.

Проблема, с которой столкнется посторонний наблюдатель, имеющий намерение узнать секретный ключ, заключается в вычислении k_Ak_BP по известным P, k_AP, k_BP, но при неизвестных k_A, k_B, в этом и состоит проблема Диффи – Хеллмана для эллиптических кривых.

Кроме того, существует множество других протоколов для классической криптосистемы. Так, например, при помощи протокола Месси – Омуры можно произвести передачу сообщения от абонента A абоненту B по открытому каналу связи без предварительной передачи какой бы то ни было ключевой информации.

Первоначально протокол Месси – Омуры был описан применительно к мультипликативной группе Z^*p, где p – простое число. Он представляет собой аналог передачи секрета с помощью ящиков, запираемых на один или два замка. Так, абонент A запирает своим ключом ящик с письмом и пересылает ящик абоненту B, который запирает ящик своим ключом и отправляет его к A. Последний должен снять свой замок и вернуть ящик B, который снимает свой замок. Вместо механических замков абоненты (их может быть два или более) могут использовать электронные, т.е. хранящиеся в компьютерной памяти ключи. Описанный вариант системы возможно реализовать, не используя процедуры возведения в степень в конечных полях, однако задача дискретного логарифмирования достаточно сложна для В, чтобы тот по первому ключу от A не смог определить ключ и впредь читать не предназначавшиеся ему сообщения. Устойчивая система подписи сообщений является обязательным условием надежности. В случае, когда подписи не используются, любое постороннее лицо C может представиться абонентом B и вернуть A со своим ключом. А продолжит процедуру и, «сняв свой замок», откроет самозванцу C путь к расшифрованию сообщения. В связи с этим, шифрование вторым ключом должно сопровождаться аутентификацией.

Данный протокол в эллиптическом варианте предоставляет возможность передавать сообщение от A к B по открытому каналу без предварительной передачи какой-либо ключевой информации.

На основе рассмотренных выше протоколов можно сделать вывод, что основным преимуществом эллиптической криптографии является малый размер ключа относительно других схем асимметричного шифрования. Это свойство особенно важно при реализации криптографических протоколов в условиях ограниченности ресурсов памяти и производительности, например, при программировании смарт-карт.

С помощью эллиптических кривых можно зашифровать не только текст, но и фотографии. Задача заключается в том, чтобы изображение было наиболее неузнаваемым. Для этого будет использоваться стохастическое шифрование и «гаммирование».

«Гаммирование» заключается в сложении точек эллиптической кривой, соответствующих гамме, с соответствующими символам алфавита точками эллиптической кривой. «Гамма» – это псевдослучайная последовательность точек эллиптической кривой.

Первоначальные фотографии, которые следовало зашифровать с помощью описанного выше метода, представлены на рис. 1 в столбце «Оригинал изображения». При помощи различных коэффициентов получаем более или менее зашифрованные картинки. Во всех случаях первый режим не дает желаемого результата: изображения достаточно узнаваемы, и их передача по открытому каналу связи небезопасна, так как злоумышленник может увидеть скрытые на них данные. В случае использования второго режима удалось добиться стойкости алгоритма, и передача данных по открытому каналу связи полностью безопасна: зашифрованное изображение скрыто от посторонних.

Рис. 1. Пример алгоритма шифрования

На основании изучения данной работы можно выделить основные достоинства и недостатки эллиптической криптографии.

Гораздо меньшая длина ключа по сравнению с «классической» асимметричной криптографией, что повышает эффективность вычислений. Кроме того, скорость работы эллиптических алгоритмов гораздо выше, чем у классических. Это объясняется как размерами поля, так и применением более близкой для компьютеров структуры бинарного конечного поля.

Из-за маленькой длины ключа и высокой скорости работы, алгоритмы асимметричной криптографии на эллиптических кривых могут использоваться в смарт-картах и других устройствах с ограниченными вычислительными ресурсами.

Все плюсы эллиптической криптографии вытекают из одного конкретного факта: для задачи дискретного логарифмирования на эллиптических кривых не существует субэкспоненциальных алгоритмов решения, что позволяет уменьшить длину ключа и увеличить производительность. Однако если такие алгоритмы появятся, то это будет означать крах эллиптической криптографии.

Библиографический список

1. Болотов, А. А. Элементарное введение в эллиптическую криптографию. Алгебраические и алгоритмические основы / А. А. Болотов, С. Б. Гашков, А. Б. Фролов. – М. : КомКнига, 2006. – С. 89–98.

2. Соловьев, Ю. П. Эллиптические кривые и современные алгоритмы теории чисел / Ю. П. Соловьев, В. А. Садовничий, Е.Т. Шавгулидзе. – М. : Ижевск, 2003. – С. 60–92.

3. Хоффман, Л. Дж. Современные методы защиты информации / Л. Дж. Хоффман. – М. : Советское радио, 1980. – С. 87–164 с.

К. И. Никишин

Пензенский государственный университет, Пенза, Россия

ШИФРОВАНИЕ ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ ПЕРЕСТАНОВОЧНОГО ШИФРА

Рассматривается программа шифрования/дешифрования текста с использованием перестановочного шифра. Перестановочный шифр разбивает информацию на блоки по некоторому количеству символов, для каждого такого блока осуществляется некоторая перестановка.

Информация является одним из признаков развития информационного общества [1]. В жизни каждого человека информация занимает особое место: личная информация, корпоративная информация и т.д. Сейчас трудно представить человека, который не пользуется сетью Интернет, не общается с другими пользователями в социальных сетях, по электронной почте.

Но любую информацию можно взломать, для обеспечения пользователям конфиденциальности предусмотрены различные механизмы шифрования. Само понятие шифрования включает в себя обратимое преобразование информации в целях скрытия от неавторизованных лиц, с предоставлением, в это же время, авторизованным пользователям доступа к ней.

В разных странах существуют различные стандарты шифрования данных, которые определены на законодательном уровне и регулируются нормативно-правовой базой страны. Самым распространенным алгоритмом является DES (Data Encryption Standart) [2]. Алгоритм Data Encryption Standart использует множество подстановок и перестановок, производит шифрование 64-битовых блоков с помощью ключа размером 64 бит. Российский стандарт шифрования данных представлен в ГОСТ 28147–89 «Система обработки информации. Защита криптографическая. Алгоритм криптографического преобразования» [3].

В статье рассматривается шифрование с перестановочным шифром. Данное шифрование предназначается для простых пользователей на уровне передачи информации через электронную почту, переписку в социальных сетях. Для крупных компаний данная программа не предназначается, поскольку необходимо зашифровывать информацию со стойкими алгоритмами шифрования.

Но исходя из результатов ежегодного исследования Norton Cybercrime Report за 2016 г. [4] совершения киберпреступлений следует, что средний ущерб от кибератаки на одного среднестатистического пользователя составляет $197, каждый пятый (21 %) онлайн-пользователь старше 18 лет становился жертвой кибератаки либо в социальных сетях, либо через мобильные устройства, либо через электронную почту, 39 % пользователей социальных сетей становились жертвами киберпреступлений.

Программа шифрования/дешифрования информации с использованием перестановочного шифра состоит из двух важных компонентов: шифратора и дешифратора программы. Интерфейс программы представлен на рис. 1.

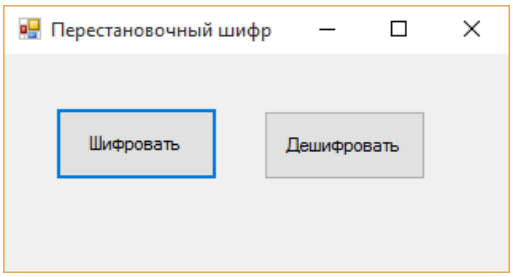


Рис. 1. Интерфейс программы шифрования

Перестановочный шифр с фиксированным периодом подразумевает разбиение исходной информации на блоки на некоторое количество символов и для каждого такого блока используется некоторая перестановка [5]. Ключом такого шифра является используемая при шифровании перестановочная матрица *P* или вектор *t*, указывающий правило перестановки. Таким образом, общее число возможных ключей определяется длиной блока *n* и равно *n!*. При дешифрации используется матрица обратной перестановки *D*, являющаяся обратной к матрице *P* по умножению, то есть $D*P = I$, где *I* – единичная матрица. Более подробно криптоанализ обсуждается в [6, 7].

Таким образом, при выполнении шифрования образуется таблица с заданной определенной длиной и горизонтально заполняется информацию, которую необходимо зашифровать. После этого информация считывается по столбцам таблицы (вертикально) и получается шифротекст (зашифрованная информация). Дешифрация информации уже осуществляется по полученному шифротексту и происходит восстановление исходной информации из шифротекста.

Программа шифратора информации работает следующим образом: пользователей указывается текстовый файл, который необходимо зашифровать. После этого программа считывает информацию из файла и заполняет новый массив информации с определенным размером блока информации. Обрабатывается массив согласно алгоритму перестановочного шифра, и шифротекст записывается в новый файл. Результат работы шифратора программы представлен на рис. 2.

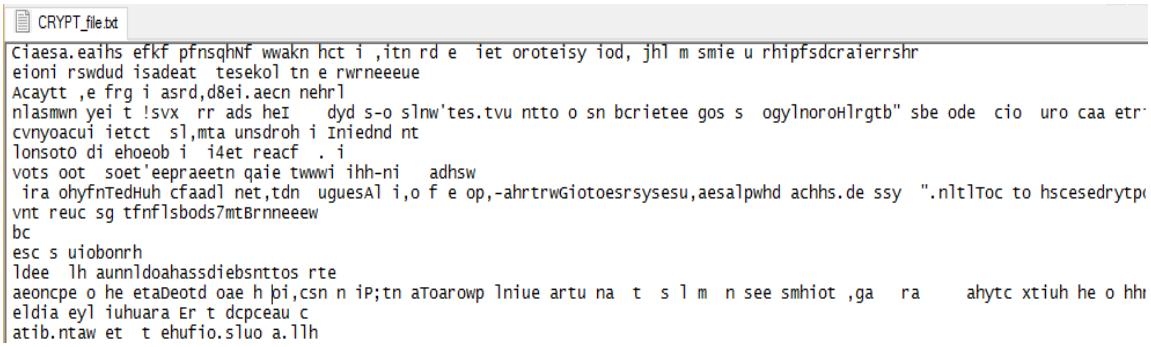


Рис. 2. Файл с зашифрованной информацией (шифротекст)

Программа дешифратора информации осуществляет обратные действия шифратора. Считывается информация уже из файла с шифротекстом, заполняется массив с определенным размером блока информации и восстанавливается исходная информация из шифротекста. Полученная информация сохраняется в файл.

Было проведено сравнение по содержимому исходного шифруемого файла с дешифрованным файлом, результат оказался положительным, поскольку должны были совпасть данные.

В статье рассмотрена реализация программы шифрования/дешифрования информации с использованием перестановочного шифра. Описана работа шифратора и дешифратора программы с учетом специфики перестановочного шифра.

Библиографический список

1. Понятие информационного общества. – URL: https://ru.wikipedia.org/wiki/Информационное_общество.
2. Стандарты шифрования данных. – URL: https://ru.wikipedia.org/wiki/Стандарты_шифрования_данных.
3. ГОСТ 28147–89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.
4. Исследования Norton Cybercrime Report. – 2016. – URL: https://us.norton.com/cybersecurity-insights
5. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М. : ТРИУМФ, 2002.
6. Sinkov, A. Elementary Cryptanalysis / A. Sinkov. – Mathematical Association of America, 1966.
7. Games, H. F. Cryptanalysis / H. F. Games. American Photographic Press. – 1937. (Reprinted by Dover Publications, 1956.1)

Д. В. Пащенко, Е. А. Бальзанникова

Пензенский государственный университет, Пенза, Россия

ПОВЫШЕНИЕ ТОЧНОСТИ ИДЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЯ ПО БИОМЕТРИЧЕСКИМ ДАННЫМ КЛАВИАТУРНОГО ПОЧЕРКА

Рассмотрен вопрос защиты информационных систем от несанкционированного доступа с использованием биометрической информации. Рассматривается метод биометрической идентификации по клавиатурному почерку. Предложены и апробированы методы повышения точности идентификации.

На сегодняшний день защита конфиденциальной информации от несанкционированного доступа является крайне актуальной задачей. Для достижения этой цели используется комплекс средств, основанных на одном или нескольких компонентах:

- атрибуты – к ним относятся пропуск, ключ, proxy-карта, документы;
- знания, такие как пароли, коды или ответ на вопрос;
- биометрическая информация – совокупность индивидуальных и уникальных физиологических параметров пользователя.

Однако системы, обеспечивающие безопасность данных на основании собственности или данных, слишком уязвимы и могут легко пострадать от подделки, воровства и других факторов. Поэтому все больший интерес вызывают методы биометрической идентификации, позволяющие определить личность человека по его физиологическим характеристикам путем распознания по заранее сохраненным образцам. Подобные способы аутентификации удобны для пользователей тем, что носители информации находятся всегда при них, не могут быть утеряны либо украдены. Биометрический контроль доступа считается более надежным, так как идентификаторы не могут быть переданы третьим лицам, скопированы.

Основными методами, использующими статические биометрические характеристики человека, являются идентификация по папиллярному рисунку на пальцах, радужной оболочке, геометрии лица, сетчатке глаза, рисунку вен руки, геометрии рук. Также существует семейство методов, использующих динамические характеристики: идентификация по голосу, динамике рукописного подчерка, сердечному ритму, походке. Таким образом, организация защиты системы от несанкционированного доступа (НСД) с использованием средств биометрической идентификации, требует наличия дополнительного оборудования.

Исходя из этого, как правило, системы аутентификации по биометрическим данным размещаются на объектах с повышенными требованиями по безопасности и защиты от НСД, тогда как дороговизна такого оборудования или неудобство ежедневного использования являются существенным ограничением широкого применения подобных систем.

Однако, на сегодняшний день существуют методы и средства идентификации и аутентификации пользователей при помощи стандартных устройств ввода персональной ЭВМ: клавиатуры и мыши. Механика набора текста на клавиатуре, по сравнению с особенностями движения мыши, обладает более простыми и легко интерпретируемыми параметрами, такие как время удержания клавиши или интервал между нажатиями клавиш, исходя из чего, в качестве биометрических данных в данной работе предлагается использовать именно клавиатурный почерк. Подобно рукописному почерку, клавиатурный почерк является уникальными для каждого человека, что позволяют с высокой долей вероятности идентифицировать пользователя, известного системе или классифицировать пользователя как «чужого», если он не соответствует ни одному образу, имеющемуся в базе. Зачастую, ввиду особенностей такой биометрической информации, системы анализа клавиатурного почерка обладают достаточно высокой точностью, по сравнению с системами, основанных на физиологических параметрах.

Определим клавиатурный почерк как совокупность параметров нажатий клавиш, отражающих процесс набора контрольной фразы, которая является результатом сознательных и бессознательных процессов мышления, а также конструктивных особенностей клавиатуры.

Существует множество эффективных и надежных алгоритмов идентификации пользователя по образу клавиатурному почерка. Однако имеется общая проблема данного вида идентификации: ухудшение качества идентификации ввиду нестабильности клавиатурного почерка пользователя. Даже у опытных пользователей персонального компьютера с формировавшейся манерой и механикой набора текста на клавиатуре могут наблюдаться существенные отклонения в динамике набора одного и того же текста. Эти отклонения могут быть обусловлены различными факторами: психофизическое состояние, вариации последовательности используемых пальцев при наборе на клавиатуре в рамках устоявшегося почерка или случайные или независящие от пользователя отклонения. Такого рода отклонения имеют особую важность, так как они повлияют на формирование эталонного образа клавиатурного почерка пользователя и могут в дальнейшем негативно сказаться в дальнейшем на точности идентификации.

Традиционно для повышения точности результатов анализа данных применяется рад мер по предварительной обработке данных, которые включают в себя: фильтрацию выбросов и нормализацию данных. Также возможно извлечение новых, линейно независимых признаков из входных данных.

Для проведения оценки точности алгоритмов были собраны данные о вводе парольной фразы несколькими пользователями, для которых методом кросс-валидации была произведена вероятности ложного срабатывания (FAR) и ложного отказа (FRR) для двух пользователей, которые условно были определены как образцы стабильного и нестабильного почерка.

Результаты данного анализа для различных порогов доступа показаны на рис. 1. На графиках слева показаны результаты для данных без предварительной обработки, справа – для данных после проведения фильтрации и нормализации данных. Графики сверху отражают результаты для образцов «нестабильного».

Как видно из приведенных результатов, предварительная обработка позволяет добиться повышения качества идентификации только для «стабильного» почерка, для «нестабильного» улучшения несущественны.

Таким образом, основной проблемой применения алгоритмов идентификации пользователя по клавиатурному почерку является нестабильность основных характеристик, обусловленная вариациями способов набора одной и той же парольной фразы в рамках клавиатурного почерка одного пользователя.

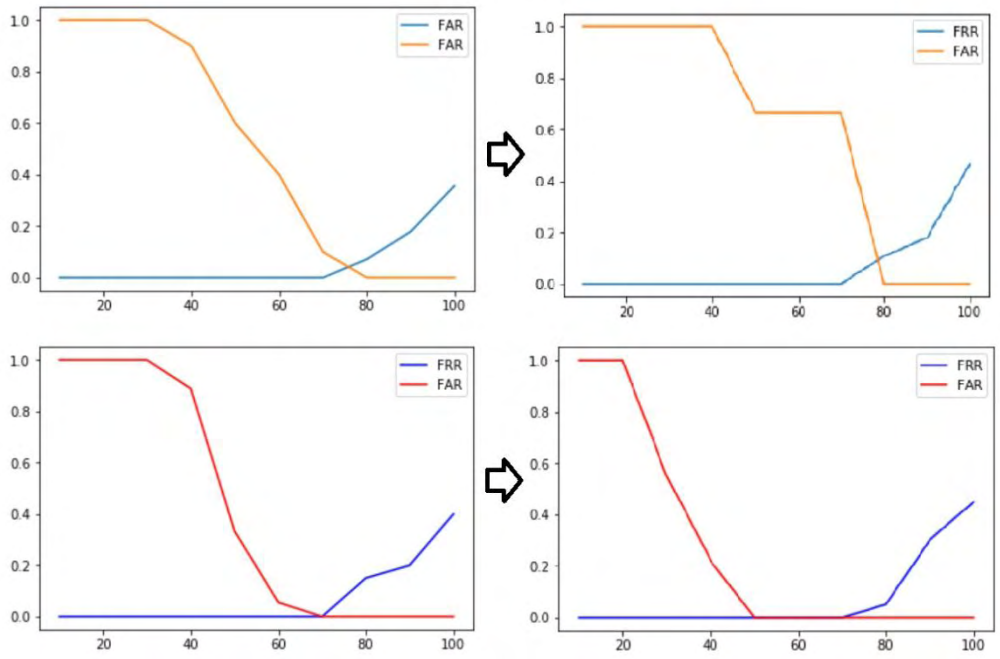


Рис. 1. Вероятность ошибок FAR и FRR для различных порогов допуска
с применением предварительной обработки

В большинстве случаев набор значений одного признака рассматривается как единая выборка, подчиняющаяся некоторому закону распределения. В рамках данной работы предлагается рассматривать значения признаков в виде отдельных групп, соответствующих альтернативными вариантами способа набора парольной фразы или текста. Предполагаем, каждая из групп также подчиняется нормальному закону распределения. Таким образом, данный способ обработки предполагает предварительную кластеризацию каждого набора признаков и последующим применением алгоритма идентификации применительно для каждого кластера.

Результаты оценки качества идентификации с применением кластеризации и предварительной обработки представлены на рис. 2.

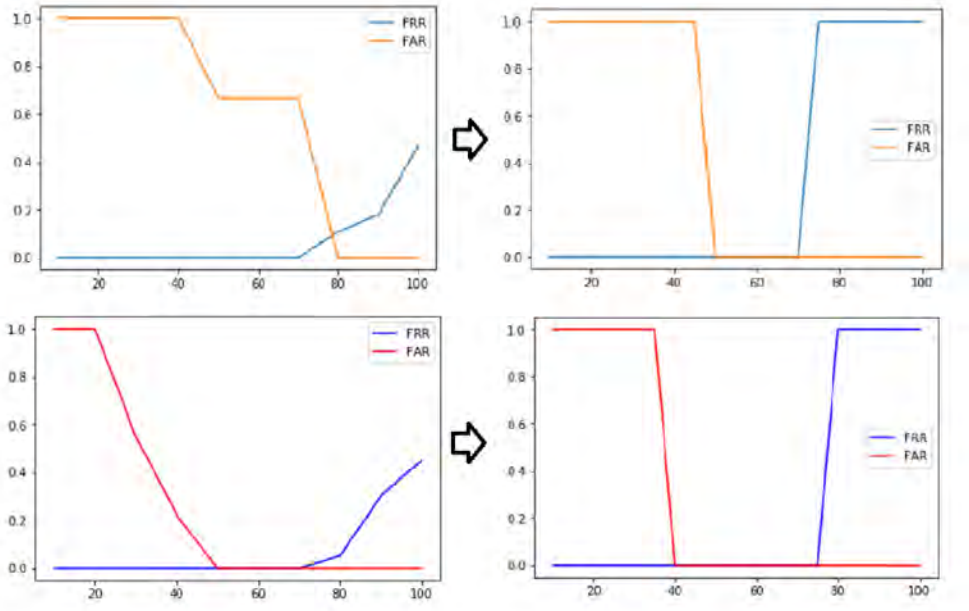


Рис. 2. Вероятность ошибок FAR и FRR для различных порогов допуска
с применением кластеризации

Таким образом, применение предложенного метода повышения качества идентификации по клавиатурному почерку с использованием кластеризации позволяет компенсировать основной недостаток всех существующих методов – снижение точности распознавания для пользователей с нестабильной механикой набора парольной фразы. При этом за счет значительного расширения возможностей существующих алгоритмов, возможно более широкое распространение методов идентификации пользователей по биометрическим образам клавиатурного почерка.

Библиографический список

1. Иванов, А. И. Биометрическая идентификация личности по динамике подсознательных движений / А. И. Иванов. – Пенза : Изд-во ПГУ, 2000. – 188 с.

2. Брюхомицкий, Ю. А. Статистические методы распознавания клавиатурного почерка / Ю. А. Брюхомицкий // Известия Южного федерального университета. Технические науки. Тематический выпуск. – 2010. – С 139–147.

3. Ходашинский, И. А. Технология усиленной аутентификации пользователей информационных процессов – Управление, вычислительная техника и информатика / И. А. Ходашинский, М. В. Савчук, И. В. Горбунов, Р. В. Мещеряков // Доклады ТУСУРа. – 2011. – № 2 (24), ч. 3. – С. 236–248.

4. Брюхомицкий, Ю. А. Гистограммный метод распознавания клавиатурного почерка / Ю. А. Брюхомицкий // Известия Южного федерального университета. Технические науки. – 2010. – № 11, т. 112. – 8 с.

5. Воронцов, К. В. Машинное обучение : курс лекций / К. В. Воронцов. – 2017. – URL: www.machinelearning.ru

Н. П. Вашкевич, П. Ю. Мясин, С. Ю. Поздняков

Пензенский государственный университет, Пенза, Россия

ПРИНЦИПЫ ПОСТРОЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В УСЛОВИЯХ СОВРЕМЕННОГО ИНФОРМАЦИОННОГО ПРОТИВОБОРСТВА

Рассмотрены некоторые информационные угрозы автоматизированных систем управления военного назначения, их жизненный цикл. Рассматриваются элементы и комплексы средств защиты информации, а также альтернативный способ представления уязвимостей программного обеспечения.

Эффективная деятельность любых структур военного назначения невозможна без использования современных информационных технологий, что предъявляет ряд требований к обеспечению информационной безопасности.

При использовании современных информационных технологий (ИТ) в автоматизированных системах управления (АСУ) военного назначения, таких, например, как компьютерные сети и базы данных, ценность информации (с точки зрения решения задач ведения боевых действий) увеличивается, поскольку она дает возможность повысить степень осведомленности, улучшить взаимодействие между командованиями различного уровня, органами военного управления и разведки и, тем самым, реализовать свое информационное превосходство [1]. Сравнивая требования информационной безопасности, применяемые в структурах военного назначения стран НАТО, одним из ключевых элементов информационного противоборства является киберкомандования родов вооруженных сил.

Например, комитет начальников штабов вооруженных сил США утвердил документ «Информационные операции», в котором излагались взгляды американского военно-

го руководства на их подготовку и проведение, уточнены цели, задачи основные принципы информационного противоборства, а так же обязанности должностных лиц по подготовке и проведению таких операций в мирное и военное время[2].

Как следовало из документа, информационные операции представляют собой комплекс мероприятий по воздействию на людские и материальные ресурсы противника для того, чтобы затруднить или сделать невозможным принятие верного решения с одновременной защитой своих информационно – коммуникационных сетей и компьютерных систем. Такие операции включали в себя пять основных составляющих:

- радиоэлектронную борьбу (electronic warfare);
- психологические операции (psychological operations);
- операции в информационно – коммуникационных сетях (computer network operations);
- военную дезинформацию (military deception).

Рассмотрим одну из информационных угроз – уязвимость программного обеспечения (ПО). Уязвимость (информационных систем); брешь – свойство информационной системы (ИС), обуславливающее возможность реализации угроз безопасности обрабатываемой в ней информации [3].

Наличие в ИС уязвимостей позволяет злоумышленникам проводить деструктивные воздействия на нее, которые реализуются с помощью эксплойтов определенной последовательности команд целевого процессора, которые используют уязвимость ИС. Процесс применения разработанного эксплойта называют атакой. Типовой жизненный цикл уязвимости ПО [5] приведен на рис. 1.

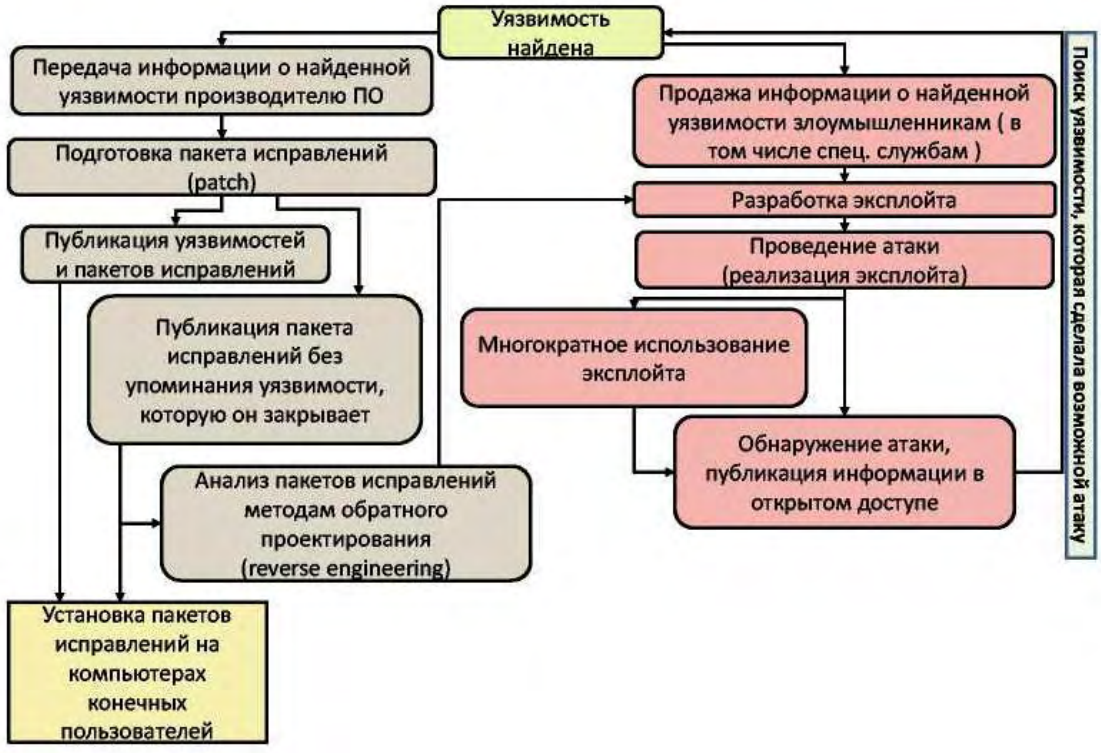


Рис. 1. Типовой жизненный цикл уязвимости ПО

Одним из выходов предотвращения информационных угроз является создания централизованной системы управления военного назначения.

Централизованная система управления военного назначения должна охватывать как унаследованные, хорошо себя зарекомендовавшие в условиях реальной эксплуатации, тех-

нические решения, так и перспективные, разрабатываемые в настоящее время элементы и комплексы средств защиты информации.

В качестве некоторых элементов и комплексов средств защиты информации рассмотрим:

- унифицированные аппаратно-программные модули доверенной загрузки с функциями управления по сети информационно-управляющей системы военного назначения (ИУС ВН), обеспечения доверенной виртуализации и контроля отчуждаемых носителей информации;
- программно-аппаратные комплексы за щиты информации от информационно технического воздействия;
- комплексы защиты информации технологии «тонкий клиент» и средств виртуализации;
- высокопроизводительные криптографические средства защиты информации, включая криптографические пулы;
- программные комплексы централизованного управления средствами защиты информации взаимодействующих ИУС ВН в рамках одной мультиплатформенной системы управления информационной безопасностью (СУИБ);
- средства защиты от НСД к вычислительным ресурсам центров обработки данных (ЦОД) и средства оперативной миграции виртуальных машин и серверов;
- автоматизированные средства построения ложных объектов;
- средства взаимодействия с несколькими удостоверяющими центрами;
- средства централизованной биометрической идентификации и аутентификации.

Современные тенденции развития ИУС ВН требуют при создании АСУ ВН учитывать следующие перспективные направления развития систем комплексной защиты информации;

- разработку средств защиты информации распределенных вычислений и услуг ЦОД («облачные вычисления»);
- реализацию криптографических сервисов для распределенных вычислений;
- обеспечение централизованного управления инфраструктурой электронной подписи с учетом построения доверенного пространства многоуровневой системы удостоверяющих центров;
- реализацию технологии интеграции АС, основанной на интеграции сервисов с предоставлением возможностей виртуализации выносных рабочих мест, серверных компонентов и электронного документооборота, включающего обработку потоков документов, заданий, индексацию и поиск информации (Интеграционный комплекс информационно-сетевых сервисов (ИКИСС);
- реализацию автоматизированных средств (сервисов) оценки эффективности защиты взаимодействующих ИУС ВН;
- защиту информации, обрабатываемой средствами виртуализации, в том числе с использованием виртуальных средств организации взаимодействия между сегментами АСУ (виртуальная сетевая инфраструктура, рабочие места и вычислительные комплексы);
- централизованное управление персональными биометрическими данными для доступа к ресурсам АС, а так же на объекты охраны;
- реализацию возможностей централизованного управления гетерогенными СЗИ взаимодействующих ИУС ВН в рамках единой СУИБ;
- построение системы компетенций СЗИ и системы взаимного доверия сервисов взаимодействующих ИУС ВН;
- криптографическую защиту информации при передаче по беспроводным каналам связи, включая мобильные устройства и оконечные исполнительные устройства;
- внедрение перспективных средств идентификации абонентов и носителей ключевой информации и др.

Таким образом, только на основе реализации комплексного подхода в обеспечении отражения информационных угроз на основе перечисленных технологий и механизмов защиты может быть достигнут требуемый высокий уровень информационной безопасности,

надежности и функциональной устойчивости АСУ ВН в условиях современного информационного противоборства.

Библиографический список

1. Зубарев, И. В. Кибер безопасность автоматизированных системы управления военного назначения / И. В. Зубарев, И. В. Жидков, И. В. Кадушкин // Вопросы кибербезопасности. – 2013. – № 1.

2. Information Operations Primer: Fundamentals of Information Operations. – U.S. Army War College, 2011. – 204 p.

3. ГОСТР 50922–2006 Защита информации. Основные термины и определения.

4. Engle, S. Tree Approach to Vulnerability Classification / S. Engle, S. Whalen, D. Howard, M. Bishop // Technical Report CSE-2006-10, Dept. of Computer Science, University of California at Davis, Davis, CA 95616-8562.

5. Марков, А. С., Систематика уязвимостей и дефектов безопасности программных ресурсов / А. С. Марков, А. А. Фадич // Защита информации: INSIDE. – 2013. – № 3. – С. 18–21.

6. Вашкевич, Н. П. Недетерминированные автоматы в проектировании систем параллельной обработки : учеб. пособие / Н. П. Вашкевич. – Пенза : Изд-во ПГУ, 2004. – 280 с.

7. Бородакий, Ю. В. Предложения в межведомственную концепцию создания доверенной аппаратно-программной среды для автоматизированных систем органов управления / Ю. В. Бородакий, А. Ю. Добродеев, И. В. Бутусов // Вопросы радиоэлектроники. – 2013. – Т. 3, № 2. – С. 15–22.

8. Бородакий, Ю. В. Основные задачи и проблемы создания криптографической подсистемы защиты распределенных автоматизированных систем управления и связи специального назначения / Ю. В. Бородакий, А. Ю. Добродеев, Ю. П. Свиридюк, П. А. Нащекин // Информационное противодействие угрозам терроризма. – 2005. – № 4. – С. 176–178.

А. А. Яужев

Пензенский государственный университет, Пенза, Россия

КОМБИНИРОВАННОЕ СРЕДСТВО АУТЕНТИФИКАЦИИ

Использование аппаратных средств аутентификации в многопользовательских информационных системах, не реализующих поддержку специализированных токенов, для ускорения и упрощения процедуры аутентификации по паролю или пин-коду.

Информационные системы выполняют разграничение доступа к своим подсистемам на основе данных, предоставляемых пользователями в процессе аутентификации. Эти данные вне зависимости от формы их представления представляют собой факторы аутентификации [1], присущие конкретному пользователю. Факторы аутентификации позволяют однозначно идентифицировать пользователя и предоставить ему некоторый набор прав.

К факторам аутентификации относят:

- токены – специальные устройства и их программные эмуляторы, формирующие аутентификационные последовательности на основе некоторого секрета с использованием определенного алгоритма (например, TOTP);
- биометрические данные – физические особенности пользователя, преобразуемые к цифровому виду (отпечаток пальца, рис. радужной оболочки);
- некоторые сведения, известные только пользователю, представленные в текстовом виде – пароль или пин-код.

Последовательность символов в сочетании с именем пользователя (логином) является наиболее простым в реализации способом аутентификации пользователя в информационной системе. Разработчику, а также пользователю, не требуется использовать специализированное программное или аппаратное обеспечение для разработки и работы

в системе соответственно. Однако для обеспечения защиты системы от несанкционированного доступа с использованием существующих механизмов аутентификации администраторы системы могут устанавливать особые требования к длине, алфавиту, а также сроку действия паролей – эти меры могут создавать существенные сложности для пользователей, а в некоторых случаях – снизить уровень безопасности.

Для решения проблемы хранения множества паролей к множеству информационных систем были разработаны менеджеры паролей – специализированные программы, хранящие аутентификационные данные в зашифрованных базах данных. Такой подход имеет несколько существенных недостатков: пользователь должен быть авторизован в операционной системе, а также иметь соответствующие права для запуска программы.

На основе микроконтроллера ATMega32U4, а также с использованием средств языка программирования C++ разработано устройство, предоставляющее аутентификационные данные (логины, пароли и пин-коды) для заданного множества информационных систем. Устройство представляет собой USB-совместимый токен, реализующий базовые функции менеджера паролей.

Устройство представляет собой изолированное хранилище данных, используемых для формирования аутентификационных данных той или иной информационной системы. Формирование паролей осуществляется с использованием шума, сформированного функцией губки Keccak [2] на основе данных контекста информационной системы.

В состав устройства входят:

- микроконтроллер Microchip ATMega32U4;
- подсистема ввода-вывода: клавиши управления и экран на основе контроллера SSD1306, сопряженный с микроконтроллером по шине I2C;
- подсистема хранения данных в виде выделенной микросхемы памяти серии Atmel AT24C;
- регулятор напряжения для обеспечения электрической совместимости компонентов.

Контекст информационной системы – это данные, на основе которых из фиксированного символьного алфавита формируются аутентификационные данные (секрет). В контекст входят: наименование системы, наименование формируемых данных, а также один из счетчиков. Процесс формирования секрета представлен на рис. 1.

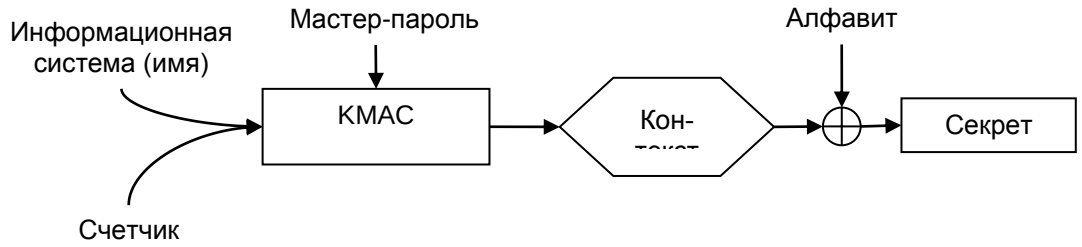


Рис. 1. Формирование секрета

Все хранимые данные подвержены маскированию с использованием мастер-ключа пользователя; при использовании демаскированных данных осуществляется их контроль целостности, а также применяются меры по маскированию используемых буферов в памяти микроконтроллера.

Устройство использует эмуляцию клавиатуры для вывода данных в компьютер, к которому оно подключено. Это достигается с использованием библиотеки LUFA, реализующей упрощенный стек протокола USB на микроконтроллерах AVR [3]. Взаимодействие осуществляется в одностороннем порядке (только вывод), за исключением этапа начальной инициализации, в ходе которого пользователь вводит список информационных систем и мастер-ключ.

Криптографические операции, выполняемые устройством, основаны на функции Keccak, используемой в различных режимах: формирование псевдослучайной последовательности (маскирование), хеширование и MAC (обеспечение контроля целостности).

Устройство может быть использовано в существующих информационных системах, использующих пару логин-пароль в качестве фактора аутентификации. Для его работы в среде операционных систем Windows, Linux и Android не требуется использование специализированных драйверов: используется стандартный драйвер клавиатуры. Перед промышленным внедрением необходимо провести опытную эксплуатацию и убедиться в полной совместимости с конкретной информационной системой.

Библиографический список

1. Смит, Р. Аутентификация: от паролей до открытых ключей / Р. Смит. – М. : Вильямс, 2002. – С. 432.

2. Bertoni, G. Sponge Functions / G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. – URL: http://sponge.noekeon.org/SpongeFunctions.pdf (дата обращения: 19.07.2017)

3. Баранов, В. И. Применение микроконтролеров AVR. Схемы. Алгоритмы. Программы / В. И. Баранов. – М. : Додэка XXI, 2006. – С. 288.