ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ЗАЩИТА ИНФОРМАЦИИ

УДК 681.3

С. А. Андреева, ***В. А. Степанов***– студенты кафедры информационных систем
Л. В. Коблякова – научный руководитель

ТЕСТИРОВАНИЕ КОММУТАЦИОННЫХ УСТРОЙСТВ SPACEWIRE

Стандарт SpaceWire [1] разрабатывается для высокоскоростной надежной связи по сети различных устройств. Для реализации связей между устройствами могут быть использованы различные коммутационные устройства. В настоящее время ведется разработка множества коммутационных устройств для сетей SpaceWire. Разрабатываемые отечественные СБИС (коммутаторы МСК-01, МСК-02...) имеют встроенное RISC ядро позволяющее писать для данных устройств программное обеспечение. В частности, коммутатор МСК-01 поставляется пользователям со штатным встроенным ПО. При разработке подобного оборудования важным этапом является тестирование готовых устройств для определения их работоспособности и рабочих характеристик в различных условиях. Актуальной задачей является создание набора универсальных гибких производственных тестов, позволяющих осуществлять тестирование различных коммутационных устройств без значительного изменения самих тестов. Благодаря наличию в коммутационных устройствах встроенных RISC процессоров возможно создание наборов тестов обладающих широким набором изменяемых пользователем параметров и позволяющих производить исследование всех параметров устройства с накоплением обширных статистических данных для последующего детального анализа.

Тестирование является важным этапом в разработке и функционировании любого устройства, оно призвано определить работоспособность устройства и его соответствие заданным характеристикам. [2] Обычно тестирование подразделяется на различные этапы. В случае коммутационного устройства SpW тестирование можно условно разделить на три вида:

1. Самотестирование: тесты, которые всегда выполняются устройством при включении. Данные тесты являются частью встроенного ПО и поставляются в готовом продукте. Позволяют определить работоспособность памяти и других внутренних систем устройства. Не требуют участия пользователя.
2. Тестирование в рамках одного устройства.
3. Тестирование устройства в составе сети.

В данной статье мы будем подробно рассматривать последние два вида тестирования.

Основная задача любого тестирования – однозначно определить исправность устройства, его пригодность к использованию. Следовательно, необходимо протестировать устройство, как в его обычных условиях эксплуатации, так и в критических. Для решения данной задачи разрабатывается набор производственных тестов.

Разрабатываемый набор производственных тестов позволяет производить тестирование всех параметров устройства, как в условиях несильной загруженности, так

и моделировать различные критические ситуации. Также этот набор тестов позволяет производить тестирования коммутационных устройств в двух режимах: тестирование одного устройства и тестирование в составе сети.

Разрабатываемый набор тестов позволяет тестировать различные коммутационные устройства SpW (МСК-01, МСК-02...) без существенных изменений в самой тестирующей программе. Для смены исследуемого устройства достаточно заменить в проекте файл, описывающий адресное пространство, на соответствующий используемому устройству и скомпилировать.

Набор тестов для тестирования в рамках одного устройства направлен на определение работоспособности и исследование основных характеристик коммутационных устройств, таких как скоростей соединения, интенсивности передачи данных, размера пакетов передаваемых данных, при которых устройство работает исправно.

При данном виде тестирования тесты запускаются на устройстве, не подключенном к сети или другому устройству. Порты SpaceWire соединяются между собой, образуя петли. На устройстве работает только тестирующая программа.

Набор тестов в рамках одного устройства охватывает все основные характеристики коммутационного устройства и включает в себя следующие тесты:

– тесты установки соединения на базовой скорости;

– тесты скоростей доступных в каналах;

– тесты передачи пакетов на различных скоростях с различной интенсивностью (с проверкой правильности принятого пакета).

Например, данные для передачи могут формироваться согласно маске, задаваемой пользователем теста, путем повторения необходимое число раз для заполнения пакета заданного объема. При проверке правильности принятого пакета производиться сравнение отправленных данных и принятых (рис. 1).

После прохождения устройством данного набора тестов мы можем получить следующие характеристики:

– среднее время установки соединения на различных скоростях;

– дисперсию времени соединения на различных скоростях;

– минимальную и максимальную скорость доступную в каналах связи;

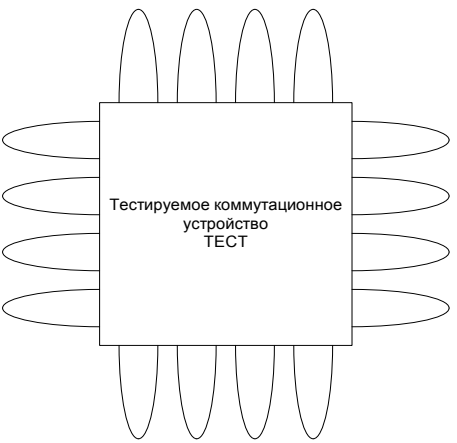


Рис. 1. Тестирование в рамках одного устройства

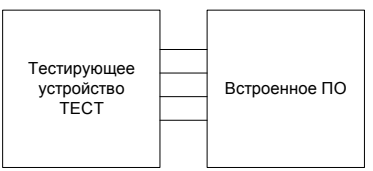


Рис. 2. Тестирование коммутационных устройств через сеть

– средние размеры пакетов, передающихся без ошибок, на различных скоростях;

– максимальную интенсивность безошибочной передачи различных пакетов

и т. д.

При тестирования коммутационного устройства через сеть производится исследование устройства в той комплектации, в которой оно поставляется пользователю (плата + Встроенное ПО). Тест запускается на тестирующем устройстве, которое подключено к тестируемому, с запущенным на нем встроенном ПО. В качестве тестирующего устройства может выступать также коммутационное устройство SpW (рис. 2).

При тестировании коммутационных устройств в составе сети решаются следующие задачи:

– тестирование установки соединений на различных скоростях;

– тестирование передачи пакетов данных на различных скоростях с различной интенсивностью тестирования адаптивной групповой маршрутизации.

Для обеспечения полноты тестирования предусмотрено комбинирование различных значений параметров тестирования – различная интенсивность подачи пакетов разных типов при разных скоростях. Возможность комбинирования значений параметров делает разрабатываемый набор тестов гибким и пригодным для исследования широкого ряда устройств.

Например, параметры могут изменяться следующим образом:

– изменение скорости:

- подъем скорости до максимальной скачком;
- плавный подъем скорости до максимальной (различные длины шагов);
- изменение скорости в различных направлениях различными шагами;

и т. д.

– изменение состава пакетов:

- короткий заголовок + короткие данные;
- длинный заголовок + короткие данные;
- различные сочетания длин заголовков и данных;

и т. д.

– изменение интенсивности подачи пакетов:

- легкая загруженность;
- максимальная загрузка.

Для создания максимальной загруженности сети к тестируемому устройству подключаются платы с запущенными на них тестами, генерирующими потоки данных (рис. 3).

При тестировании передачи пакетов данных необходимо проверять правильность переданного пакета. Для этого можно использовать RMAP пакеты, т.к. у них есть подтверждение правильности принятого пакета, также это может помочь сформировать максимальный поток данных на тестируемом устройстве. Для этого можно, например, создать такую ситуацию как на рис. 4.

В данном случае в коммутационному устройству с запущенным на нем Встроенным ПО, подключаются устройства передатчики RMAP-пакетов и устройства приемники RMAP-пакетов (любые устройства способные отослать ответ о правильности принятого пакета).

Коммутационные устройства SpW обладают адаптивной группо-

Рис. 3. Создание максимальной загруженности сети

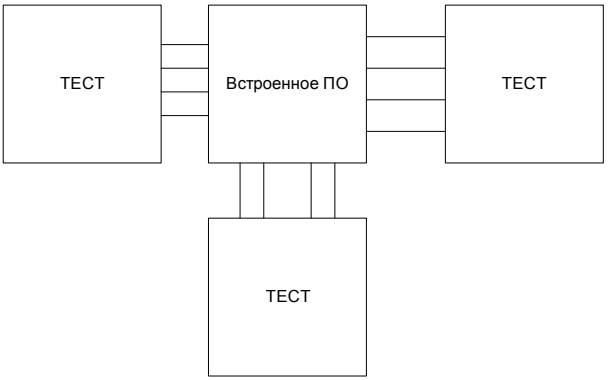


Рис. 4. Тестирование передачи пакетов данных

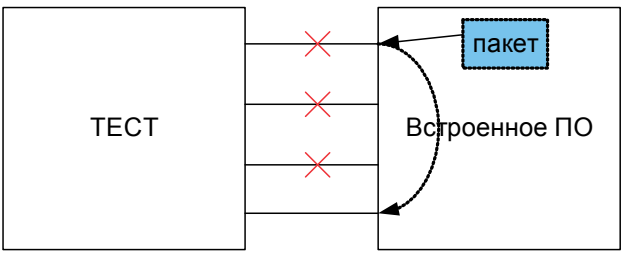


Рис. 5. Передача пакетов при неработающих портах

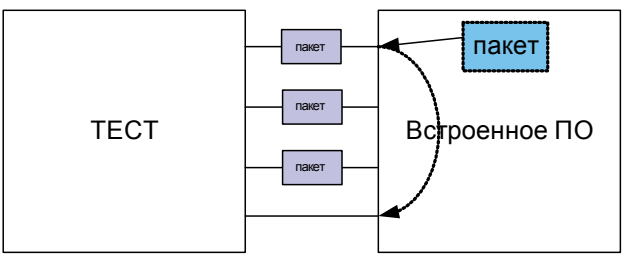


Рис. 6. Передача пакетов при занятых портах

вой маршрутизацией. Групповая адаптивная маршрутизация позволяет при наличии дублирующих соединений (группы) перенаправлять потоки данных с одного устройства на другое в случае выхода из строя или сильной загрузки одного или нескольких соединений из группы.

При тестировании адаптивной групповой маршрутизации возможны два варианта.

1. Какие-то порты из группы не работают, а на них пришел пакет, тогда он перенаправляется на работающие порты группы.

На рис. 5 и 6 показана передача пакета между двумя коммутаторами МСК-01 связанными несколькими каналами.

2. Какие-то порты группы заняты, и чтобы не было задержки при пересылке пакета, он перенаправляется на свободный порт.

Для формирования данной ситуации формируется сильная загруженность одного или нескольких портов из группы.

Представленный в статье набор тестов позволяет определять основные характеристики тестируемого устройства, оценивать предельные значения внешних параметров. Комбинирование значений различных параметров позволяет достичь максимальной полноты тестирования и оценить работу устройства, в том числе и в нештатных ситуациях. При успешном прохождении устройством данного теста, можно будет однозначно утверждать о его работоспособности и соответствии его сетевых параметров заявленным характеристикам. Благодаря универсальности разрабатываемых тестов с их помощью могут быть протестированы различные коммутационные устройства SpW без внесения существенных изменений в тестирующую программу.

Библиографический список

1. ECSS-E-50-12C. SpaceWire – Links, nodes, routers and networks. – European Cooperation for Space Standardization (ECSS), 31 July 2008.
2. Nigel Barges. Network testing, http://www.ccc.ru/magazine/depot/05_07/

УДК 004.413

А. В. Грибовская – магистрант кафедры информационно-сетевых технологий
Г. С. Бритов (канд. техн. наук, доц.) – научный руководитель

ПРОЦЕССНОЕ МОДЕЛИРОВАНИЕ ДЕЯТЕЛЬНОСТИ КОММЕРЧЕСКИХ БАНКОВ

На сегодняшний день эффективного управления ресурсами и операционной деятельностью банка недостаточно для его стабильного развития и конкурентного преимущества. Необходим переход на новый уровень – уровень развития и постоянного совершенствования. Четкая стратегия, регламентированность и адаптивность бизнес-процессов, продуманная оргструктура и эффективная взаимосвязанная система менеджмента банка в целом – таковы главные факторы, которые позволят банку удержать лидирующие позиции на рынке в ближайшие годы.

Если более внимательно посмотреть на управленческую деятельность практически любого банка, то станет ясно, что все ее элементы тесно взаимосвязаны. Например, при описании и оптимизации бизнес-процессов параллельно приходится работать с организационной структурой и документооборотом банка. Для успешной реализации стратегии банка ее элементы (например, цели и проекты) необходимо декомпозировать до уровня сотрудников, функций (элементарных действий), спроецировать на бизнес-процессы. [1]

Бизнес-процессы являются объектом меоделирования при изучении технологий обработки информации в банковской деятельности. Они представляют собой цепочку операций выполняемых по отношению к бизнес-объектам, в этом процессе участвующим. Каждый бизнес-процесс описывает некоторый банковский процесс, например, выдачу кредита, международный платеж, и т. п. Как правило, бизнес-процесс описывает жизненный цикл одного из участвующих в нем объектов – этот объект называют главным участником процесса. Главный участник, по сути, связан с процессом в тече-

Рис. 1. Следствия неформализованности бизнес-процессов банка

ние всего срока его жизни, прочие же объекты могут включаться в список участников и исключаться из этого списка.

Описание бизнес-процессов прямым образом влияет на операционную и стратегическую эффективность коммерческого банка. Это в итоге влияет на показатели прибыльности. Также подобное описание бизнес-процессов в виде модели может оказать значительную помощь как в построении с нуля различных систем управления банком, так и в организации и совершенствовании работы подразделений и бизнес-процессов банка по соответствующим системам управления. [2]

Модель ведет к формализации бизнес-процессов банка. Покажем основные следствия, вызванные их неформализованностью (рис. 1).

Причины слабой формализованности и регламентированности бизнес-процессов следующие

– во-первых, не распределенная четко ответственность между сотрудниками, отделами;

– во-вторых, несовершенная бизнес-логика процессов и несоответствие реальным требованиям. Довольно часто бывает, что бизнес-процесс выполняется по устаревшим правилам и схемам, которые давно не соответствуют требованиям современного бизнеса и клиентов. Иногда бывает, что для бизнес-процесса можно сделать альтернативную бизнес-логику, которая будет более эффективной. Несовершенная бизнес-логика также может выражаться в дублировании действий (процедур), отсутствии важных (ключевых) действий, наличии ненужных (лишних) действий;

– в-третьих, слабая автоматизация бизнес-процессов и несоответствие инфраструктуре. Не секрет, для того чтобы автоматизировать бизнес-процесс, следует сначала его описать и на основе формализованного описания и регламента разработать техническое задание на автоматизацию;

– также можно указать как причину неосведомленность персонала о правилах выполнения отдельных действий и взаимодействия с другими подразделениями. Поскольку нет формализованных схем и регламентов, основная информация по бизнес-процессам хранится в памяти сотрудников.

Как следствия из вышеуказанного можно отметить следующие положения:

– элементы управления абстрактны (существуют лишь на бумаге) и почти не соответствуют реальной ситуации в управлении банком. Например, многие нормативные документы хранятся «для виду» и для проверок Банка России, некоторые из них устарели и «не работают»;

– в элементах управления наблюдается большая несогласованность, а иногда даже противоречивость. Например: проблемное взаимодействие подразделений; долгие согласования нормативных документов; размытая ответственность за процессы, задачи и проекты;

– слабая адаптация управленческой деятельности к изменениям внешней среды. Например, отсутствие механизмов оперативного принятия и реализации правильных управленческих решений;

– слабая коммуникативность элементов управления. Например, слабая осведомленность персонала банка о его стратегии, бизнес-процессах и корпоративной культуре, непонимание (а соответственно и неиспользование в работе) персоналом нормативных документов;

– отсутствие формализованной структуры управленческой деятельности (элементы управления закреплены на словах и в мысленных представлениях руководителей);

– на основе данных проблем можно сделать вывод о необходимости формализации управления в банке и, как следствие, получении больших выгод и преимуществ в результате проведения данной работы;

– проблемы из-за увольнения ключевых сотрудников. Любой ключевой сотрудник обладает бесценным опытом по своему участку работы, своими наработками, мастерством. И уход такого сотрудника является для банка большим риском и высокими издержками. Чтобы минимизировать риски от ухода сотрудников и снизить издержки на обучение новых сотрудников, следует формализовать все знания и опыт работы сотрудника в виде технологических карт и регламентов бизнес-процессов, которые он выполняет;

– ошибки в работе сотрудников и некачественное оказание услуг. Любого рода ошибки влекут за собой дополнительное время и издержки на их устранение;

– снижение удовлетворенности клиентов.

Если говорить о выгодах моделирования бизнес-процессов для банка, то список преимуществ, которые может получить банк, и список задач, которые можно решить, благодаря описанию бизнес-процессов выглядит следующим образом.

1. Повышение прозрачности, управляемости и контролируемости деятельности банка на всех уровнях.

2. Снижение времени и издержек, повышение качества и эффективности бизнес-процессов.

3. Возможность тиражировать бизнес банка (создавать дополнительные отделения и офисы).

4. Шаг к комплексному развитию банка. Описание бизнес-процессов является этапом комплексного проекта по развитию деятельности банка. На основе описанных бизнес-процессов можно:

- проводить их дальнейшую оптимизацию;
- проектировать новые бизнес-процессы;
- оптимизировать организационную структуру;
- совершенствовать системы управления банка (информационную систему, систему управления финансами, систему стратегического управления и т. п.).

5. Уменьшение зависимости от персонала, правильный подбор персонала, повышение эффективности работы персонала и руководителей.

6. Повышение лояльности и удовлетворенности клиентов, как следствие репутации банка.

Рассматривая подходы к моделированию бизнес-процессов, следует отличать два типа методик.

1. Методика организации проекта по описанию бизнес-процессов. Они задают последовательность этапов проекта, состав этапов, правила взаимодействия участников проекта.

2. Методика графического описания бизнес-процессов. Они содержат набор графических объектов и правил их использования при разработке диаграмм бизнес-процессов.

Отметим две методики организации проекта по моделированию бизнес-процессов.

1. С помощью дерева бизнес-процессов. Сначала разрабатывается иерархическая структура (дерево) бизнес-процессов банка. Затем из этого дерева берутся бизнес-процессы 1-го уровня и детально описываются. Описывается деятельность владельца бизнес-процесса и в дополнение описывается деятельность всех участников бизнес-процесса.

2. Моделирование по отделам. Сначала описывается организационная структура банка. Затем из организационной структуры выбираются отделы и описывается их деятельность в рамках разных бизнес-процессов. В заключении все схемы одного бизнес-процесса от разных отделов сводятся в единую схему модели. [2]

В дальнейшем бизнес-процессы банка будут рассмотрены не со стороны его организационной структуры (по отделам и подразделениям), а со стороны процессинговых решений (дерево бизнес-процессов), а также взаимодействия с различными смежными организациями (например, Центральный Банк, Международные платежные системы и т. д.).

Перечислим самые распространенные технологии, с помощью которых можно описывать бизнес-процессы: IDEF0, IDEF3, DFD, ARIS, UML.

Как правило, моделирование средствами IDEF0 – первый шаг на этапе системного анализа жизненного цикла информационной системы. Методы и средства этого шага определяют IDEF0-технологию разработки информационной системы. [3]

При проведении сложных проектов обследования предприятий, разработка моделей в стандарте IDEF0 позволяет наглядно и эффективно отобразить весь механизм деятельности предприятия в нужном разрезе. Однако самое главное – это возможность коллективной работы, которую предоставляет IDEF0.

Основная идея моделирования IDEF0 может быть сформулирована следующим образом: бизнес-процессы представляются как некоторые преобразования входного (и, возможно, управляющего) потока в выходной под контролем (управлением) управляющего потока с использованием для преобразования механизма. Бизнес-процессы должны быть представлены на более высоком уровне диаграммы достаточно общим образом, позволяющим уяснить их суть, однако, без излишней детализации, усложняющей понимание и чтение диаграмм.

Для более детального рассмотрения бизнес-процессы должны быть декомпозированы в соответствии с принципами структурного подхода к разработке информационных систем.

Библиографический список

1. Исаев Р. А.. Бизнес-архитектура и системы управления Банка // Управление в кредитной организации. 2009. № 4 (эл.верс.).
2. Исаев Р. А.. Методика описания (структуризации) бизнес-процессов коммерческого Банка и ее практическое применение // Управление в кредитной организации. 2008. № 4 (эл.верс.).
3. Верников Г.Основные методологии обследования организаций. Стандарт IDEF0 (http://md-management.ru/).

УДК 007.51

К. В. Зац – студент кафедры вычислительных систем и программирования
В. П. Калюжный (канд. техн. наук, доц.) – научный руководитель

УСТРОЙСТВО ДИСТАНЦИОННОГО УПРАВЛЕНИЯ АВТОНОМНЫМИ МОБИЛЬНЫМИ ОБЪЕКТАМИ С СЕТЕВЫМ ПРОТОКОЛОМ ОБМЕНА

Автономные мобильные объекты, или роботы, довольно широко применяются последнее время. В их число входят планетоходы, охранные, спортивные, исследовательские роботы и другие.

Автономное функционирование этих роботов не обязательно отменяет необходимость ведения информационного обмена с ними. Многие роботы передают телеметрическую информацию в некоторый центр управления, откуда в свою очередь передаются команды управления. Таким образом осуществляется функциональный контроль, настройка и передача экстренных команд при возникновении нештатных ситуаций.

Иногда целесообразным является применение команды автономных мобильных роботов. В таком случае для осуществления информационного обмена между роботами и пунктом управления целесообразно применять сеть. Примером может служить экспериментальный роботизированный охранный комплекс MDARS (Mobile Detection Assessment and Response System) (США). Для организации информационного обмена между элементами комплекса используется беспроводная локальная IP-сеть на основе стандарта IEEE 802.11, более известного как Wi-Fi.

В рамках студенческого исследовательского проекта «Феникс-3» [1–4], проводимого СКБ ГУАП, был создан иммобилизатор – устройство, дистанционно обездвиживающее робота, в случае возникновения нештатных ситуаций. Необходимость в его создании обуславливается спецификой робота Феникс3.

В рамках проекта проводится экспериментальное изучение возможностей встроенных систем управления на основе нейронных сетей с использованием автономного робота Феникс– 3. Автономный робот Феникс-3 предназначен для автономного патрулирования в заданном районе с целью обнаружения очагов возгорания. В случае обнаружения источника робот должен приблизиться к очагу возгорания и используя бортовой огнетушитель погасить огонь. Предполагается, что роботы будут работать в команде.

Постановка задачи экспериментального изучения подразумевает автономное функционирование робота в реальной обстановке, что создает потенциальную возможность нанесения ущерба как имуществу, так и людям. Для его предотвращения была поставлена задача создать устройство, дистанционно обездвиживающее (иммобилизирующее) робота.

То обстоятельство, что роботы будут работать в команде, делает целесообразным применение сетевых технологий при управлении ими.

Исходя из требований, предъявляемых к иммобилизатору, необходимо разработать устройство, имеющее следующие технические параметры (приведены в табл. 1).

Обездвиживание мобильного объекта можно осуществить путем коммутации цепей питания или цепей управления двигателями.

Первый способ более надежен и универсален, т. к. в случае коммутации питания конструкция иммобилизатора не будет привязана к конкретному типу мобильного объекта. Однако в этом случае приходится коммутировать больший ток (в случае робота Феникс-3 до 100 А), что потребует дополнительного охлаждения коммутирующих элементов (реле или силовых транзисторов). Для этого могут использоваться радиаторы, которые могут дополняться кулерами для уменьшения их размеров и более эффективного охлаждения, возможно использование жидкостной системы охлаждения. Зачастую габариты мобильных устройств, в том числе робот Феникс-3, не позволяют разместить в них требующуюся систему охлаждения. Поэтому был выбран вариант коммутации цепей управления.

Таблица 1

Параметры иммобилизатора

Напряжения питания мобильного объекта	до 15 В
Количество коммутируемых каналов управления	2
Предельное расстояние дистанционного отключения при мощности передатчика 10 мВт не менее	10 м
Время срабатывания	не более 1 сек
Время автономной работы	не менее 2 часов
Габариты не должны превышать	20*100*100 мм
Вес не более	0.1 кг
Тип разъемов для подключения коммутируемых каналов управления	WF-5/HU-5

С учетом вышесказанного была предложена структурная схема иммобилизатора, представленная на рис. 1.

Было принято решение сделать каналообразующую аппаратуру на основе БПУ-07М – мастер-устройства системы цифрового дистанционного управления ЦСДУ-2, внешний вид которого представлен на рис. 2. Он представляет собой малогабаритное

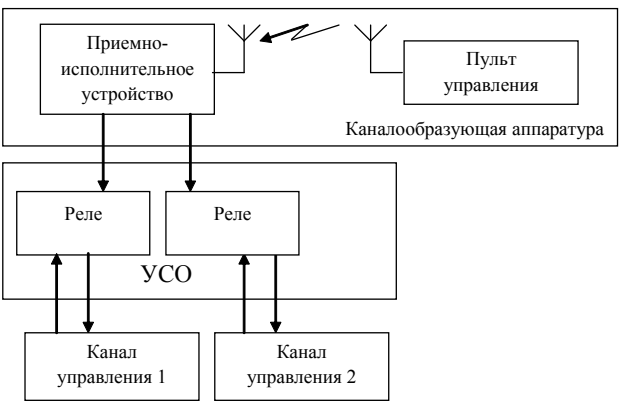


Рис. 1. Структурная схема иммобилизаторагде: УСО – устройство сопряжения с объектом

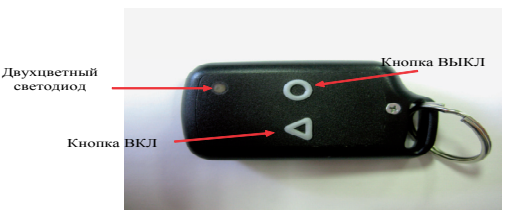


Рис. 2. Внешний вид мастер-устройства системы цифрового дистанционного управления ЦСДУ-2

устройство с автономным питанием, содержащее радиомодем, реализованный на микросхеме CC1100 [5] и контроллер интерфейса, реализованный на микроконтроллере PIC18F2320 [6]. БПУ-07М снабжено двумя кнопками, сдвоенным светодиодом и микромотором. Вместо слэйв-устройства ЦСДУ-2 БПУ-07S в качестве приемно-исполнительного устройства было взять второе мастер-устройство ЦСДУ-2, т. к. только оно имеет собственный корпус.

На сетевом уровне реализуемый протокол не отличается от протокола ЦСДУ-2. Топология сети – «точка-точка». Обмен информацией между пультом управления и приемно-исполнительным устройством производится в полудуплексном режиме. Каждый пульт управления работает со своим приемно-исполнительным устройством. Максимальное количество узлов в сети 255. Адреса 0 и 255 используются для передачи широковещательных (сервисных) команд.

Передаваемый кадр данных (посылка) состоит из 48 байт. Один байт используется для кодирования команды остановки/запуска робота или подтверждения остановки/запуска. Остальные байты в настоящее время не используются, они зарезервированы для дальнейшего использования. В частности с их помощью можно реализовать защищенный протокол передачи данных, осуществлять фоновую проверку уровня сигнала и добавить в устройство новые функции, например, контроль функционирования мобильного устройства.

Во время работы в фоновом режиме пульт управления периодически посылает кадр запроса уровня сигнала, приемно-исполнительное устройство посылает в ответ кадр данных (рис. 3). Если уровень сигнала слабый, пульт управления показывает это оператору своей индикацией. Такой обмен происходит через определенные промежутки времени для экономии питания. Однако эта функция в настоящий момент не реализована.

Команда остановки или запуска передается пультом управления, в ответ приходит подтверждение от приемно-исполнительного устройства (рис. 4). В случае отсутствия подтверждения команда будет передаваться повторно, до 30 раз, пока подтверждение не будет получено (рис. 5). При получении приемно-исполнительным устройством кадра с нарушениями целостности посылки (неверный CRC) подтверждение не высылается, и оно остается в RX состоянии.

Обозначения, принятые на рис. 3–5: MASTER – мастер-устройство системы ЦСДУ-2, пульт управления иммобилизатора, SLAVE – слэйв-устройство системы ЦСДУ-2, приемно-исполнительное устройство иммобилизатора, TX – фаза передачи, RX – фаза приема.

К настоящему времени выполнено натурное макетирование устройства.

Эксперимент показал превышение заявленных характеристик системы. Предельное расстояние дистанционного отключения оказалось больше на 5 метров, чем требовалось и составило 15 метров. Данный результат является хорошим, потому что это расстояние не имеет ограничений в требованиях, чем оно больше, тем лучше.

Дальнейшим развитием иммобилизатора является добавление функции контроля качества связи. Для этого потребуется изменить режим работы пульта управления – он должен будет находиться включенным все время работы, а не только на время посылки команды и ожидания подтверждения. Также потребуется изменить протокол обмена.

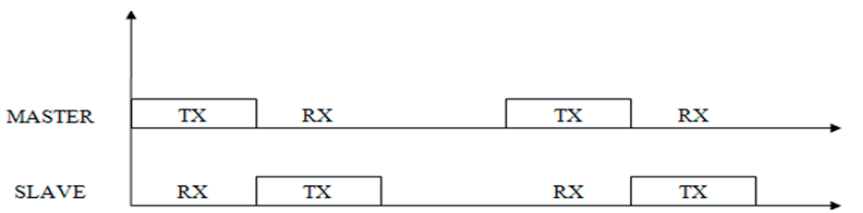


Рис. 3. Временная структура фоновой проверки уровня сигнала

Рис. 4. Временная структура «оптимального» обмена при посылке команды

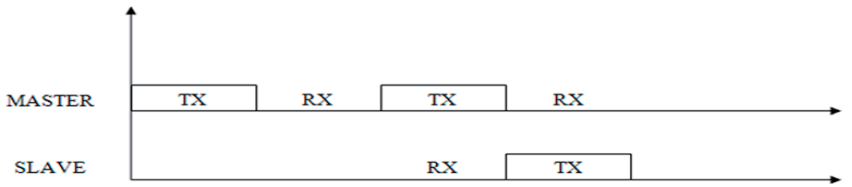


Рис. 5. Временная структура обмена при посылке команды в случае отсутствия подтверждения от приемно-исполнительного устройства

Данную функцию можно реализовать, используя модем CC1100 пульта управления [5]. Сконфигурировав его нужным образом, значение уровня радиосигнала будет добавляться в конец получаемого пакета. В состоянии RX при приеме пакета это значение можно прочитать из статус регистра RSSI. После корректировки можно получить значение мощности принимаемого сигнала в децибелах и в случае достижения критической отметки подать сигнал оператору индикацией.

Разработанный и созданный образец иммобилизатора позволяет останавливать робота, удаленного от оператора на расстояние до 15 метров, осуществлять контроль проведенной иммобилизации посредством индикации на пульте управления и приемно-исполнительном устройстве, а также проверять работоспособность системы посредством тестовой иммобилизации непосредственно с приемно-исполнительного устройства.

Выполненное макетирование и проведенные натурные эксперименты позволили сформулировать ряд выводов:

– проведённые испытания подтвердили работоспособность предлагаемых схемотехнических и программных решений в целом.

– имеет смысл дальнейшее развитие системы, в частности полная реализация разработанного протокола с поддержкой функции контроля качества связи.

– представляется перспективным интегрирование системы иммобилизатора с системой телеметрического контроля автономных мобильных объектов.

В заключение автор выражает благодарность коллективу СКБ ГУАП и команде проекта Феникс-3 да всестороннюю поддержку в реализации данного проекта.

Библиографический список

1. Астапкович Д. А, Гончаров А. А, Дмитриев А., Михеев А. ПРОЕКТ РОБОТ ФЕНИКС-1 URL: http://guap.ru/guap/skb/docs/fen1.doc
2. Astapkovitch A. M. ONE STEP LEARNING PROCEDURE FOR NEURAL NET CONTROL SYSTEM URL: http://guap.ru/guap/skb/docs/tbs.doc
3. Бурдуков А. В. Автономный робот «Феникс-3» http://guap.ru/guap/skb/burdukov_fen3_ru.doc
4. Бурдуков А. Система дистанционного управления роботом «Феникс 3» URL: http://guap.ru/guap/skb/burdukov.doc

5. CC1100 Preliminary Data Sheet, Chipcon AS, 2005. URL: http://www.datasheetarchive.com/pdf-datasheets/Datasheets-37/DSA-733744.html
6. PIC18F2220/2320/4220/4320 Data Sheet, Microchip Technology Inc., 2003. URL: http://ww1.microchip.com/downloads/en/devicedoc/39599c.pdf

---

УДК 004.056.55

Д. О. Иванов, ***Д. А. Рыжо***в – студенты кафедры комплексной защиты информации
А. В. Афанасьева – научный руководитель

ПОСТРОЕНИЕ ЛИЧНОСТНОЙ КРИПТОСИСТЕМЫ НА БАЗЕ ЦИФРОВОЙ ПОДПИСИ КФС

В работе [1] впервые была предложена идея личностной криптосистемы. Каждый пользователь такой системы имеет понятный и легко запоминаемый идентификатор Id в виде строки (например, адрес электронной почты: $bob@vu.spb.ru$), при помощи известной всем хеш-функции любой участник протокола может вычислить публичный ключ пользователя с данным Id : $PK_{Id} = hash(Id) = hash("bob@vu.spb.ru")$. В системе также присутствует некоторая доверенная сущность, именуемая *ГСК* (генератор секретных ключей), которая владеет мастер-ключом, при помощи которого генерирует публичные параметры системы и может вычислить секретный ключ SK_{Id} любого пользователя по его идентификатору.

Такая система состоит из следующих функций:

1. Всамом начале жизни системы *ГСК* в зависимости от параметра безопасности γ выбирает мастер-ключ системы *МКС* и генерирует публичный ключ системы *ПКС*.

$$Setup\ (\gamma) \rightarrow \langle ПКС, МКС \rangle$$

2. По запросу пользователя *ГСК* выдает ему его секретный ключ.

$$KeyExtraction(Id, ПКС, МКС) \rightarrow SK_{id}$$

3. Любой желающий отправить сообщение пользователю с идентификатором *Id*, шифрует это сообщение, используя *Id* и публичный ключ системы *ПКС*.

$$Encrypt(message, Id, ПКС) \rightarrow ciphertext$$

4. Чтобы прочитать полученное сообщение, пользователь дешифрует его, используя свой секретный ключ SK_{id} и публичный ключ системы *ПКС*.

$$Decrypt(ciphertext, Sk_{id}, ПКС) \rightarrow message$$

Существует множество работ, посвященных построению и анализу личностных криптосистем [2–4], основанных на эллиптических кривых, квадратичных вычетах и решетках. По результатам сравнительного анализа этих работ, следует отметить следующие особенности, описанных в них криптосистем. Системы, использующие аппарат квадратичных вычетов, имеют малую скорость шифрования и дешифрования, при этом имеют большую избыточность передачи данных. Системы, использующие аппарат эллиптических кривых, имеют малый размер ключей и шифротекста, но очень высокую вычислительную сложность. Еще важно отметить, что в основе этих схем лежат задачи, которые не являются NP-полными. Поэтому в данной статье предлагается реализация личностной криптосистемы, использующая кодовый подход, где в качестве трудной задачи будем использовать декодирование случайного линейного кода (DACL).

Опишем предложенную схему.

Пусть H_{pk} проверочная матрица случайного линейного кода исправляющего до t ошибок, u – синдром вектора ошибки. Вектор e, имеющий вес до t, – произвольный вектор ошибки. Тогда задача поиска вектора e веса до t из уравнения $e \cdot H_{pk}^T = u$ является NP-полной.

Начальная инициализация заключается в вычислении матрицы $H_{pk} = B \cdot H \cdot P$, где B – невырожденная $(n-k)\times(n-k)$ матрица, P – перестановочная $n \times n$ матрица, H – проверочная матрица (n,k) – линейного кода, исправляющего до t ошибок и имеющего полиномиальный алгоритм декодирования. Публичными параметрами являются H_{pk} и t.

Процедура извлечения секретного ключа основана на подписи КФС[5]. Опишем алгоритм:

Шаг 1. $counter = 1$

Шаг 2. $s = hash(Id, counter)$

Шаг 3. Попытка декодировать s в коде H_{pk}

Шаг 4. Если не удалось декодировать, увеличить $counter$ на 1 и вернуться на шаг 2

Шаг 5. Если удалось декодировать, то полученный вектор ошибки e будет частью секретного ключа

Секретный ключ пользователя состоит из пары: e и $counter$. Пользователь может проверить правильность секретного ключа следующим образом: $e \cdot H_{pk}^T = hash(Id, counter)$.

Пусть требуется зашифровать сообщение для пользователя с идентификатором Id. Обозначим через $u_i = hash(Id, i)$, где $i = 1 \ldots z$. Так как на передающей стороне не известен $counter$, то Id соответствует множество $u_1 \ldots u_z$, то есть для шифрования сообщения пользователю нужно вычислить z значений хеш-функции. Будем рассматривать далее значения хеш-функции как некоторые двоичные вектора длины $n-k$: $u_i = (u_{i,1} \ldots u_{i,n-k})$. Рассмотрим основные этапы шифрования 1 бита для значения хеш-функции u_f, $f = 1 \ldots z$:

Шаг 1. Сгенерировать l случайных векторов $a_1 \ldots a_l$ длины $n-k$ таким образом, чтобы $u_f \cdot a_i^T = b$, где b – передаваемый бит

Шаг 2. Составить $l \times (n-k)$ матрицу A_f из векторов a_i

Шаг 3. Сгенерировать случайную двоичную $n \times l$ матрицу X_f, вероятность появления 1 в которой равна p

Шаг 4. Вычислить матрицу Y_f: $Y_f = H_{pk}^T \cdot A_f^T + X_f$

Покажем, как можно дешифровать Y_f. Так как владельцу секретного ключа известно значение $counter$, $1 \le counter \le z$, то для дешифрации бита b, он выбирает из множества матриц Y_f, $f = 1 \ldots z$, матрицу $Y_{counter}$. Используя эту матрицу и свой секретный ключ e, пользователь вычисляет $\hat{b}$ следующим образом:

$$\hat{b} = e \cdot Y = e \cdot H_{pk}^T \cdot A^T + e \cdot X = u \cdot A^T + e \cdot X = (b_1 \ldots b_l) + (\hat{e}_1 \ldots \hat{e}_l).$$

Таким образом, пользователь получит сумму двух векторов: вектора длины l, все биты которого равны b, и вектора ошибки. Декодируя $\hat{b}$ с исправлением до $\frac{l-1}{2}$ ошибок, пользователь получает значение бита b.

Опишем, как можно передавать несколько бит вместо одного. Для этого в публичные параметры системы добавляется G – порождающая матрица $(l,\hat{k})$ -линейного

кода, имеющего быстрый алгоритм декодирования. Так как мы хотим передавать не бит, а сообщение m, то вычислим кодовое слово $c = m \cdot G$. Для передачи кодового слова c будем генерировать a_i так, чтобы $u \cdot a_i = c_i$. Можно заметить, что при передаче одного бита мы также использовали кодирование, но с использованием тривиального кода с повторением, который состоял из двух кодовых слов. Для передачи кодового слова требуется передавать z шифрограмм. Постараемся уменьшить это число. Общее количество матриц A для конкретного ключа превышает значение $\frac{2^{r-1}}{l}$. Будем вычислять случайную матрицу A так, чтобы: $u_1 \cdot A = u_2 \cdot A = \ldots = u_f \cdot A$.

Тогда количество матриц A уменьшится до $\frac{2^{r-f}}{l}$ (это справедливо в случае исключения линейно-независимых синдромов). Таким образом, количество шифрограмм стало $\frac{z}{f}$.

Построим пример для данной схемы и оценим получившиеся параметры. В качестве кода H будем использовать код Гоппы с параметрами $n = 2^{16}$, $t = 4$. Число z определим как $t! = 24$. Будем объединять по $f = 4$ шифрограммы. Код G возьмем с параметрами $l = 256$, $\hat{k} = 64$, $d = 65$, то есть исправляющий 32 ошибки. Пусть $p = 2^{-5}$, тогда средний вес вектора ошибки будет 29. При таких параметрах количество переданной информации на бит: $\frac{l \cdot n \cdot z}{\hat{k} \cdot f} \approx 192$ килобайта. Оценим также вероятность отказа от дешифрования. Общее количество синдромов $2^{n-k} = n^t$. Число декодируемых синдромов $\sum_{i=1}^{t} C_n^t \approx \frac{n^t}{t!}$. Таким образом вероятность декодировать случайный синдром: $\frac{1}{t!} = \frac{1}{24}$. Следовательно, вероятность отказа от декодирования: $(1 - \frac{1}{t!})^z \approx 0.36$.

В сравнении со схемой, использующей аппарат эллиптических кривых [2], данная реализация выигрывает в скоростях шифрования и дешифрования.

В отличие от схемы, основанной на квадратичных вычетах [3], данная реализация основана на NP-полной задаче.

Библиографический список

1. Shamir A. Identity-based Cryptosystems and Signature Schemes// Proc. of Crypto 84, LNCS 196, 47–53.
2. Boneh D., Franklin M. Identity Based Encryption from the Weil Pairing// Proc. Crypto 01, LNCS 2139, 213–229.
3. Cocks C. An Identity Based Encryption Scheme based on Quadratic Residues// LNCS 2260, 360–363.
4. Gentry C., Peirert C., Vaikuntanathan V. Trapdoors for Hard Lattices and New Cryptographic Constructions// Proc. STOC 08.
5. Courtois N., Finiasz M., Sendrier N. How to achieve a McEliece based digital signature scheme// Springer-Verlag 2001.

УДК 621.391.1

Д. О. Иванов, ***Д. А. Рыжов***, ***А. В. Соловьева*** – студенты кафедры комплексной защиты информации
А. М. Тюрликов (канд. техн. наук, доц.) – научный руководитель

СРАВНЕНИЕ АЛГОРИТМОВ РАЗРЕШЕНИЯ КОНФЛИКТОВ, ИСПОЛЬЗУЕМЫХ ПРИ РАДИОЧАСТОТНОЙ ИДЕНТИФИКАЦИИ

В стандарте [1] описан способ идентификации радиочастотных меток. В системе идентификации RFID-меток имеется опрашивающее устройство и некоторое количество меток, каждая из которых имеет уникальный идентификатор (ID). Идентификация меток осуществляется согласно следующему алгоритму:

Шаг 1. Опрашивающее устройство посылает всем находящимся в зоне видимости меткам сигнал о том, что оно находится в режиме идентификации и способно принимать ответные сигналы.

Шаг 2. Метки, получившие сигнал, посылают в ответ свой ID.

Шаг 3. Опрашивающее устройство идентифицирует метки по выбранному алгоритму разрешения конфликта.

В работах [2–6] приводятся различные модели использования радиочастотной идентификации(оценка эффективности персонала, организация сенсорной сети, «конвейерная» лента, накопители промежуточной информации). Стандарт предлагает набор возможностей, но не указывает, в каких условиях следует использовать те или иные возможности. Целью данной работы является исследование набора возможностей, предлагаемых стандартом, и выдача рекомендаций по его использованию. Для достижения цели предлагается упрощенная модель функционирования системы, основанная на данных стандарта.

Все время работы системы разбито на окна. Передача информации меткой может осуществляться только в начале окна. В конце каждого окна опрашивающее устройство сообщает о событии в окне. Событиями являются успех, пусто или конфликт.

Для идентификации меток используются два алгоритма разрешения конфликта:

1. Динамическая АЛОХА.

2. Древовидный алгоритм.

Рассмотрим древовидный алгоритм. Он является аналогом блокированного неулучшенного стек-алгоритма [7]. Группа меток, которая будет участвовать в разрешении конфликта, выбирается опрашивающим устройством посылкой команд GROUP_SELECT и GROUP_UNSELECT. Эти команды с помощью маски выбирают метки с определенным ID. Выбранные метки устанавливают счетчик в 0. Алгоритм:

Шаг 1. Все метки, у которых счетчик равен 0, передают свой ID.

Шаг 2. Если передавали 2 или более метки, то произошел конфликт. Опрашивающее устройство отправляет сообщение о конфликте (FAIL). При получении команды FAIL метка может находиться в одном из двух состояний:

1. Счетчик не равен 0, тогда увеличить счетчик на 1;

2. Счетчик равен 0, тогда счетчик равен случайному числу из множества {0,1}.

Шаг 3. Если ни одна метка не передавала или передавала только одна, опрашивающее устройство посылает сообщение об успехе (SUCCESS). При получении этой команды метки уменьшают свой счетчик на 1.

Рассмотрим алгоритм динамическая АЛОХА:

Шаг 1. Опрашивающее устройство посылает команду Init_round.

Шаг 2. Каждая метка выбирает случайное окно, в котором она будет передавать(наибольший возможный номер окна содержится в команде Init_round). Все метки, а также опрашивающее устройство, устанавливают счетчик номера окна в 0.

Шаг 3. Метки, выбравшие окно, номер которого совпадает со значением счетчика номера окна, передают опрашивающему устройству свой идентификатор и контрольную сумму.

Шаг 4. Опрашивающее устройство вычисляет контрольную сумму для полученного идентификатора и сравнивает её с полученной контрольной суммой. Если суммы совпали, опрашивающее устройство посылает команду Next_slot, содержащую контрольную сумму успешно передавшей метки. В противном случае, если окно выбрало несколько меток или не выбрало ни одной, опрашивающее устройство посылает команду Close_slot, означающую, что ни одна метка не передала успешно в данном окне.

Шаг 5. Если передавшая в данном окне метка получила команду Next_slot, содержащую верную контрольную сумму, это означает, что метка успешно передала свой идентификатор. Эта метка переходит в неактивное состояние. Все остальные метки(все метки в случае получения команды Close_slot) и опрашивающее устройство увеличивают счетчик номера окна на 1, и алгоритм возвращается к шагу 3.

Шаг 6. Если после увеличения счетчика номера окна, его значение совпало с количеством окон, опрашивающее устройство посылает команду Init_round, содержащее новое количество окон, и алгоритм возвращается к шагу 2.

Для анализа влияния шумов изменим модель следующим образом (рис. 1). Опрашивающее устройство может различать 4 события: пусто, успех, конфликт из-за шума (событие, когда передавала одна метка, но из-за шума оно было воспринято как конфликт) и конфликт (событие, когда одновременно передавало 2 и более меток).

Стандарт предлагает модификацию древовидного алгоритма для такого канала. Если опрашивающее устройство понимает, что произошла ошибка и система находится в состоянии «Конфликт из-за шума», то оно посылает команду RESEND. При получении данной команды метка, осуществлявшая передачу, повторяет ее, а остальные бездействуют (т.е. дублируется предыдущее окно).

Сравним оба алгоритма. Результаты работы алгоритмов разрешения конфликтов можно оценить по зависимости средней задержки в канале от интенсивности входного потока. Графики зависимостей средней задержки для алгоритмов разрешения конфликта при различных условиях работы алгоритмов и вероятностях шума в канале представлены на рис. 2. Сравнение производятся при вероятностях шума в канале 0.1 и 0.3. Моделирование производилось по окнам, но, используя данные стандарта, длительности окон были переведены в секунды, а интенсивность входного потока в коли-

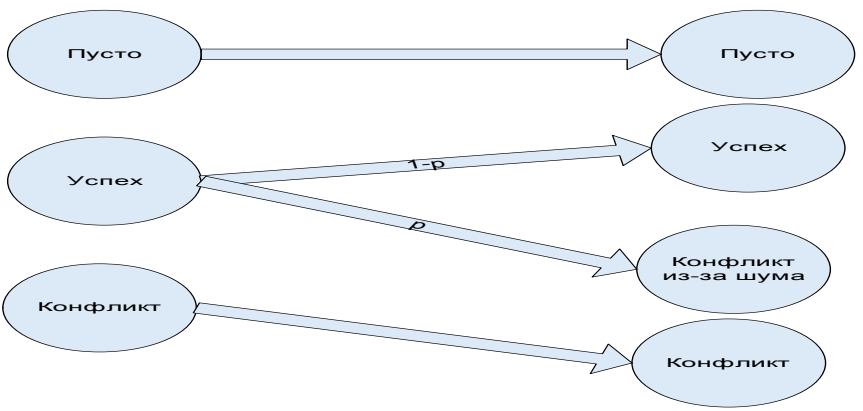


Рис. 1. Модель канала с ложными конфликтами

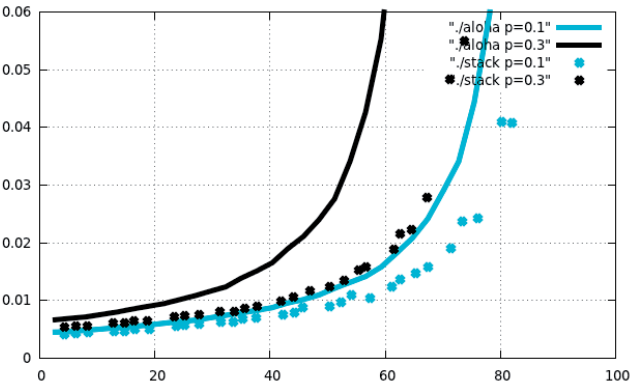


Рис. 2. Сравнение алгоритмов в реальных единицах времени

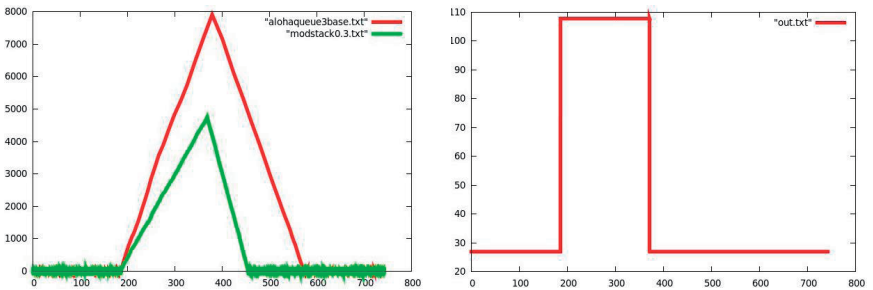


Рис. 3. Поведение алгоритмов на потоке со всплесками

чество меток в секунду. По оси абсцисс отложен входной поток в количестве меток, проходящих через зону считывающего устройства в секунду, по оси ординат – средняя задержка в секундах.

Рассмотрим поведение алгоритмов на потоке со всплесками. В определенный момент происходит всплеск и резко возрастает интенсивность входного потока на непродолжительный промежуток времени. Вследствие чего повышается средняя задержка. На верхней части рис. 3 представлен график зависимости количества необработанных меток от времени, а в нижней – интенсивность входного потока от времени. Результаты моделирования показывают, что стек-алгоритм быстрее возвращается к нормальной работе.

В данной работе были смоделированы алгоритмы разрешения конфликтов, описанные в [1]. Было наглядно продемонстрировано, что лучше использовать стек-алгоритм, но только его модифицированную версию с командой RESEND, что позволяет повысить устойчивость к шумам.

Библиографический список

1. ISO/IEC 18000-6: 2004(E)
2. Yi-Chin Lin, RFID-Enabled Analysis of Care Coordination and Patient Flow in Ambulatory Care.
3. Qingzhen Xu, Eunmi Choi, Analytical Analysis of RFID Sensor Network Model.

4. J. Vales-Alonso и др., Markovian Model for Computation of Tag Loss Ratio in Dynamic RFID Systems.
5. Sapna Tyagi и др., RFID Data Management.
6. Shoewu O., Badejo O., Radio Frequency Identification Technology: Development, Application, and Security Issues.
7. Бертсекас Д., Галлагер Р., Сети передачи данных. М.: Мир. 1989.

УДК 004.412

В. А. Иванова – студентка кафедры информационных систем
А. Ю. Сыщиков – научный руководитель

ПРИМЕНЕНИЕ АЛГОРИТМА ПОИСКА ПОДГРАФОВ ДЛЯ ПРОВЕРКИ РЕЗУЛЬТАТОВ РАБОТЫ АЛГОРИТМА СЛИЯНИЯ ГРАФОВ

Объединение (слияние) графов – операция, при которой из нескольких графов получается новый, множество вершин которого включает все вершины исходных графов, и множество дуг которого включает все дуги исходных графов.

Существуют алгоритмы объединения графов, но в некоторых случаях требуется проверка корректности их работы. Эта проверка необходима для а) выявления ошибок в алгоритме объединения б) для объединения последовательно большого количества графов, когда есть вероятность, что на каком-то шаге в результирующем графе уже будут включены некоторые последующие графы, следовательно, для них не нужно проводить операцию объединения. Задача проверки алгоритма слияния сводится к задаче поиска подграфа в графе.

Рассмотрим алгоритм поиска подграфа на примере data flow graph – это ориентированный граф с типизированными вершинами (рис. 1).

Алгоритм поиска подграфа

Каждый узел графа имеет тип (например, A, B или C) и некоторый уникальный номер. На уровне DFG тип узла определяется обычно операцией, которую он выполняет, но может иметь и большее количество параметров.

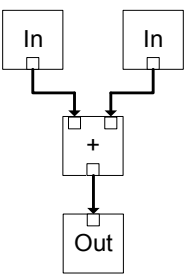


Рис 1. Пример DFG для функции, выполняющей сложение двух элементов

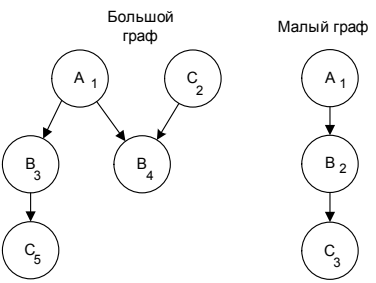


Рис. 2. Исходные графы

Условия

Количество узлов искомого графа должно быть меньше или равно количеству узлов графа, в котором мы будем его искать. Если искомый граф будет больше чем сам граф, то он не может являться подграфом.

В обоих графах не должно быть циклов.

В дальнейшем, искомый граф будем называть «малым графом», а граф, в котором его будут искать – «большим графом» (рис. 2).

Введем понятие таблицы соответствий. Эта таблица состоит из двух столбцов – в первый помещаются вершины большого графа, а во второй – соответствующие им вершины малого графа. Те узлы малого графа, которые попали в таблицу соответствий, будут помечаться, как просмотренные.

1 ЭТАП. Составление первоначальных таблиц соответствий (рис. 3)

Шаг 1. Выбирается некий произвольный узел из малого графа – S_i. Пусть его тип – A. Узел отмечается как просмотренный.

Шаг 2. Из большого графа выбираются все узлы, которые имеют тот же тип A. Если таких узлов не нашлось – малый граф однозначно не входит в большой. Если найден хотя бы один такой узел – переходим к шагу 3.

Шаг 3. С каждым из найденных узлов составляется пара – узел из малого графа и узел из большого графа. Для узла S_i составляется n пар вида: (B_j, S_i).

Каждая пара является первым элементом очередной таблицы соответствий. Таким образом после первого прохода шага 3 у нас имеется столько таблиц соответствий, сколько аналогов было найдено для узла S_i.

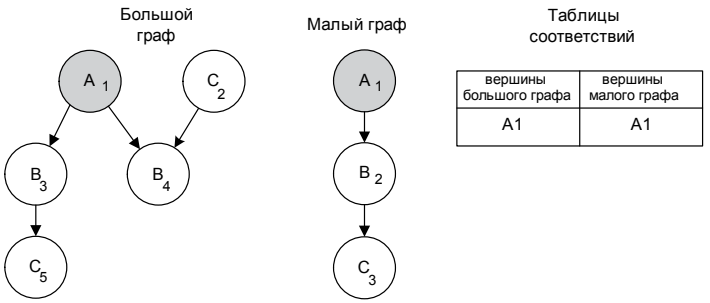


Рис. 3. Первый этап, составление первоначальных таблиц соответствия

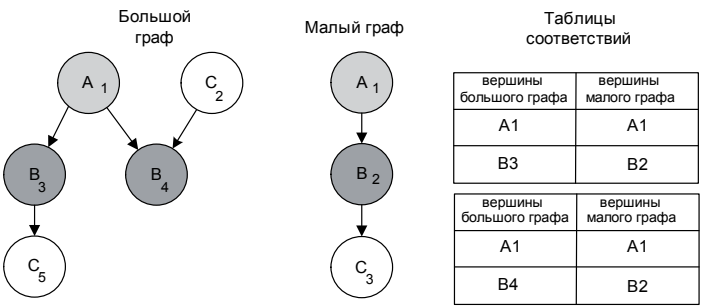


Рис. 4. Второй этап для узла B2 малого графа

2 ЭТАП. Добавление новых не просмотренных узлов в таблицы (рис. 4 и 5)

Шаг 1. Выбираем следующий не просмотренный узел. Допустим, был найден некий узел S_k типа B. Отмечаем узел S_k как просмотренный.

Шаг 2. Берем следующую таблицу соответствий.

Повторяем шаг 2 этапа 1 для узла S_k.Из полученных вершин-аналогов отбрасываем те, которые уже присутствуют в данной таблице соответствий.

Шаг 3. Берем следующую строку из таблицы соответствий.

Эта строка состоит из узла малого графа и аналогичного ему узла из большого графа.

Если между узлом малого графа и рассматриваемым узлом есть дуга, то точно такая же по направлению дуга должна быть между аналогом узла малого графа и любым из аналогов рассматриваемого узла. Например, очередная строка таблицы соответствий содержит пару из узла большого графа B_i и узла малого графа S_i. Из рассматриваемого узла S_k исходит дуга в узел S_i. Для узла S_k существуют аналоги в большом графе – узлы B_m и B_n. Из них выбираются только те узлы, в которые исходит дуга из узла B_i.

Из аналогов рассматриваемого узла удаляются те, которые не имеют подобных дуг. Если же между узлом малого графа из пары и рассматриваемым узлом дуги нет, дополнительных проверок не проводится.

Если в таблице соответствий еще есть нерассмотренные строки, снова переходим на шаг 3. Иначе – на шаг 4.

Шаг 4. На основе списка аналогов узла S_k нужно добавить в таблицы соответствий новые строки.

Берем рассматриваемую таблицу соответствий, размножаем ее в количестве, равном количеству аналогов узла S_k. В каждую из полученных таблиц заносим строку, состоящую из узла S_k и его очередного аналога. Однако эти новые таблицы пока не будут включены в рассмотрение – их будут рассматривать на следующей итерации этапа 2. Если после шага 3 аналогов узла S_k не осталось, исходная таблица просто удаляется.

Если есть еще не рассмотренные таблицы данной итерации – переходим к шагу 2. Если нет – переходим к шагу 5.

Шаг 5. Если все таблицы этой итерации были просмотрены, а новых составлено не было – значит для узла S_k не существует корректного аналога в большом графе, и малый граф не входит в большой. Если имеются новые таблицы и не просмотренные узлы в малом графе – переходим на 1 шаг 2 этапа.

Если не просмотренных узлов нет – все итерации выполнены, переходим на этап 3.

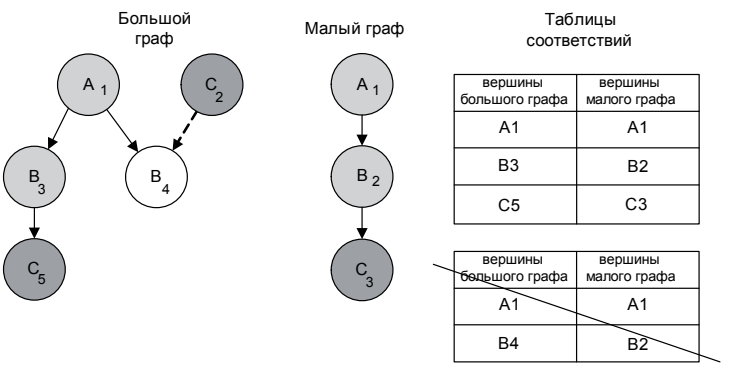
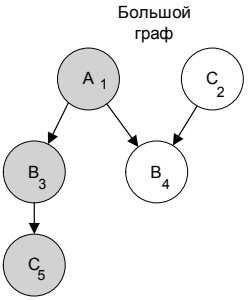
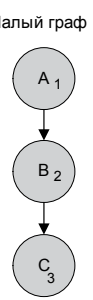


Рис. 5. Второй этап для узла малого графа C3





Таблицы соответствий

вершины большого графа	вершины малого графа
A1	A1
B3	B2
C5	C3

Рис. 6. Результаты работы алгоритма

3 ЭТАП. После выполнения всех итераций таблиц соответствий столько же, сколько вариантов вхождения подграфа в граф. Если нет ни одной таблицы – значит, ни одного вхождения найдено не было (рис. 6).

Представленный алгоритм можно реализовать не только для DFG. Так же он пригоден для неориентированных графов, смешанных графов, графов с взвешенными дугами и т.д. Он во многом облегчает отладку алгоритма объединения графов. Можно оптимизировать его работу, если на 2 этапе, на 1 шаге выбирать не просмотренный узел так, чтобы он имел дугу с любым из уже просмотренных узлов малого графа. Если таких узлов нет, выбирается любой не просмотренный узел. Это делается для того, чтобы сократить количество вариантов, перебираемых на 2 этапе.

УДК 004.056

В. С. Коломойцев – студент кафедры безопасности информационных систем
А. В. Сергеев – научный руководитель

К ВОПРОСУ О МАСШТАБИРУЕМОМ (SCALABLE) СЖАТИИ ИЗОБРАЖЕНИЙ БЕЗ ПОТЕРЬ С ИСПОЛЬЗОВАНИЕМ JPEG-LS

В работе рассматривается возможность построения эффективного масштабируемого по качеству(scalable) низкосложностного кодека на базе алгоритма JPEG-LS, оцениваются некоторые предлагаемые в литературе решения. Алгоритм сжатия изображений JPEG-LS является международным стандартом в области компрессии изображений без потерь, обладающий малым уровнем сложности. Важным преимуществом JPEG-LS по сравнению с другими современными алгоритмами сжатия изображений и видео являются:

– возможность сжатия изображений с заданным уровнем потерь по качеству;
– низкий уровень вычислительной сложности алгоритма;
– минимальные требования к необходимому объему оперативной памяти;
– возможность контролировать уровень потерь.

Алгоритм JPEG-LS разрабатывался, прежде всего, для тех случаев, когда требуется сжимать изображения без(или с контролируемыми) потерями качества: медицина, космос, картография, профессиональная фотография и т. д.

Новые области применения выдвигают новые требования к алгоритмам сжатия данных. Например, для задачи передачи видео по беспроводным каналам(мобильное ТВ, видео-блоггинг, охранные системы, удаленный мониторинг объектов и т. д.). важно иметь возможность получить изображение на приемной стороне(пусть и с меньшим уровнем качества) даже в случае потери части потока данных. Такое свойство алгоритма сжатия принято называть «масштабируемость по качеству». В работе исследуется и проверяется один из известных подходов по добавлению свойства масштабируемости потока к алгоритму JPEG-LS и предлагаются усовершенствования.

В 2002 г. вышла работа [3] двух сотрудников компании Philips – Ren´e J. van der Vleuten и Sebastian Egner, в которой был описана методика применения масштабируемого сжатия в формате JPEG-LS. Помимо применения масштабируемости, в статье было заявлено, что данный метод также приводит к снижению веса изображения. В данной работе проверяются результаты статьи, оценивается эффективность предложенных решений, предлагаются собственные модификации.

Суть описанного в статье алгоритма масштабируемого сжатия в следующем. Исходный бит изображения делится на две части – старшую (msb) и младшую (lsb). Старшая часть сжимается без потерь алгоритмом JPEG-LS, а младшая разбивается на слои (масштабируется), а затем сжимается с требуемым качеством. После выполнения данной операции, из исходного изображения получается несколько изображений, каждый из которых является каким-либо слоем исходного.

Идея алгоритма изложена в оригинальной статье следующим образом: на вход подается сигнал, он анализируется и делится на две части – старшую(msb) и младшую(lsb); старшая часть сжимается без потерь алгоритмом JPEG-LS, а младшая сжимается с любой степенью качества, либо не сжимается вовсе (рис. 1).

Для проверки описанных в статье результатов данный алгоритм был реализован с использованием реализации JPEG-LS от компании HP [2]

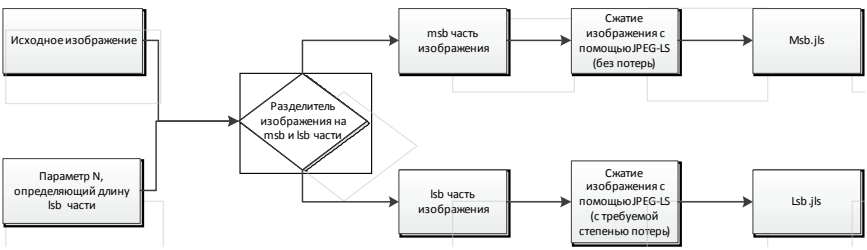


Рис. 1

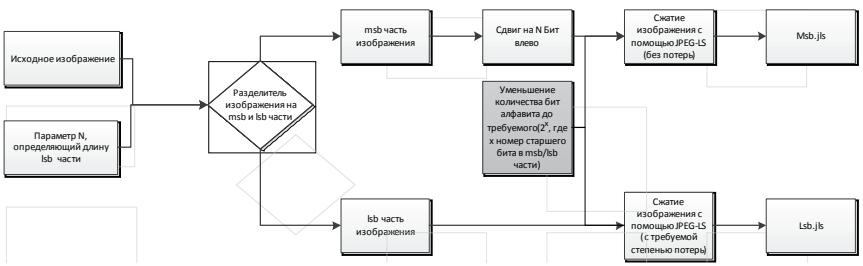


Рис. 2

Опираясь на описанный в статье алгоритм, была предложена своя модификация изложенного в статье алгоритма (рис. 2).

Основная модификация алгоритма заключается в том, что помимо разделения исходного изображения на две части, также был выполнен побитовый сдвиг вправо старшей (msb) части на длину младшей части и уменьшение размера используемого алфавита, для обеих частей изображения. За счет использования данных модификаций размер полученных изображений в среднем уменьшился на 4,5%.

Используя собственную реализацию обоих вариантов алгоритма масштабируемого сжатия(описанную в статье и модифицированную) была оценена их эффективность при сжатии изображений тестового набора Kodak PhotoCD PCD0992[4]. Этот же тестовый набор используется и в оригинальной статье. В табл. 1 представлены значения по средней степени сжатия для старших(msb) битовых частей, младших (lsb) битовых частей и изображений в целом (для исходного алгоритма). Аналогичные значения для модифицированного алгоритма приведены в табл. 2. Усреднение выполнено по всему тестовому набору изображений.

Для сравнения приведены значения при сжатии без потерь старшей части (без сжатия младшей).

Отличие между результатами, полученными в оригинальной статье и в данной работе объясняются, прежде всего, использованием различных реализаций JPEG-LS: University of British Columbia [1] и HP [2].

Таблица 1

K	lsb bit rate	msb bit rate (в статье)	total bit rate (в статье)	relative bit rate (в статье)	msb bit rate (на практике)	total bit rate (на практике)	relative bit rate (на практике)	Процент выигрыша (%)
0	0	4,345	4,345	100	4,198026	4,198026	100	4
1	1	3,393	4,393	101	3,324	4,324	103	2
2	2	2,534	4,534	104	2,4473	4,4473	106	2
3	3	1,811	4,811	111	1,768	4,768	114	1
4	4	1,25	5,25	121	1,213	5,213	124	0,7

Таблица 2

K	Lsb bit rate	msb bit rate (базовый алгоритм)	total bit rate (базовый алгоритм)	msb bit rate (модифицированный алгоритм)	total bit rate (модифицированный алгоритм)	Процент выигрыша (%)
0	0	4,198026	4,198026	4,028026	4,028026	4,5
1	1	3,324	4,324	3,208932	4,208932	3
2	2	2,4473	4,4473	2,384633	4,384633	1,5
3	3	1,768	4,768	1,702449	4,702449	1,4
4	4	1,213	5,213	1,175756	5,175756	0,75

Модифицированный алгоритм для всех тестируемых изображений, показал лучшие результаты, чем алгоритм из оригинальной статьи. Это объясняется уменьшением размера алфавита.

Наилучшие показатели по выигрышу (7,1%) были получены для изображения, содержащего большое количество текстур (рис. 3).

Рис. 3

Рис. 4

Рис. 5

Рис. 6

Наихудшие показатели по выигрышу (0,83%) были получены для изображений, содержащих большое количество неоднородностей(фотографии домов, пейзажи) (рис. 4–6).

В работе были исследованы, реализованы и проверены на практике предложенные в статье методы масштабируемого сжатия для алгоритма JPEG-LS. На практике результаты базового алгоритма по степени сжатия, полученные для нашей реализации, по результатам проведенных экспериментов, оказались лучше описанных в исходной статье в среднем на 2%. Это объясняется, на наш взгляд. использованием другой реализации кодека JPEG-LS(HP вместо University of British Columbia). Величина выигрыша, варьируется в основном из-за объема однородности в тестируемых изображениях. В изображениях с большим количеством однотипных частей(небо, море, одноцветные фоны и т. д.), величина полученного выигрыша больше. В изображениях с малой долей однотипных частей (дома, элементы архитектуры, «сложные» пейзажи), выигрыш является минимальным.

Кроме того, была предложена собственная модификация базового алгоритма. В результате, модифицированная версия алгоритма сжимает от 4,5% до 0.7% лучше описанной в статье, при различных режимах работы.

Библиографический список

1. University of British Columbia, Signal Processing & Multimedia Group, Vancouver, Canada, «JPEG-LS Codec version 2.2», 1999. URL: http://spmg.ece.ubc.ca/
2. Hewlett-Packard laboratory, Palo Alto, California, USA. URL: www.compression-links.info/JPEGLS
3. Sebastian Egner and Ren´e J. van der Vleuten, «Lossless and Fine-Granularity Scalable Near-Lossless Color Image Compression», in Data Compression Conference (DCC 2002), (Snowbird, UT), 2002.
4. «Kodak PhotoCD PCD0992 images», 1999. URL: http://sqez.home.att.net/thumbs/Thumbnails.html

УДК 004.051

Н. Д. Лямина – магистрант кафедры информационно-сетевых технологий
К. В. Недоводеев – научный сотрудник

ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ПОДПРОГРАММ ДЛЯ DSP-ЯДЕР

Написание программ на машинно-ориентированных языках является трудоемким процессом. Программу, написанную на ассемблере для DSP-ядра сложно анализировать из-за использования параллельного ассемблера, который включает несколько операций в одной строке. Простое добавление комментариев не облегчает восприятие программы, поскольку каждый комментарий становится громоздким, из-за того, что каждая из операций требует отдельного пояснения.

В условиях, когда неизвестно насколько можно ускорить программу процесс ручной переработки (оптимизации) становится трудоемким и не гарантирует получение положительного результата. В случае, когда производительность программы далека от пиковой, возникает вопрос о том, насколько эффективна написанная программа. Следует отметить, что пиковая производительность обработки данных отражает лишь возможности системы, не учитывая ограничений, присущих конкретной задаче.

Так, например, перед автором стояла задача оценить производительность подпрограммы, реализующей умножение матриц [3, 5] для DSP-ядер семейства Мультикор [4] работающих в SIMD-режиме. В процессе исследования было обнаружено, что производительность SIMD-программы достигла лишь 65% от пиковой, хотя производительность SISD-программы достигала 90%.

В качестве объекта для сравнения с существующей реализацией алгоритма в данной работе предлагается использовать эталонную программу. Эталонная программа – это программа с максимально возможной производительностью, учитывающая особенности организации вычислительного ядра [5]. Задача оценки эффективности реализации алгоритма сводится к выбору порогового значения и сравнения разности между производительностью эталонной и существующей программы с пороговым значением. Если указанная разность не превышает порогового значения, будем считать, что программу не имеет смысла изменять (оптимизировать).

Рассмотрим факторы, отрицательно влияющие на разработку высокопроизводительных программ для DSP-ядер: машинная ориентированность языка, размещение нескольких операций в одной строке, код и данные программы должны находиться в локальной памяти имеющий малый объем. Из-за наличия первых двух факторов по исходному тексту сложно восстановить состав алгоритма и связи между его фрагментами. Последний фактор ограничивает рост размерности задачи и, как следствие, экстенсивный рост производительности программы, и сужает рамки применения методов оптимизации программы, увеличивающих размер ее кода.

Целесообразно оперировать графическим представлением алгоритма, поскольку в таком случае состав алгоритма и связи между его фрагментами представлены в явной форме. В качестве такого представления в работе предлагается использовать диаграмму потоков в данных. Диаграмма потоков данных включает вершины (операции), например, операции адресной арифметики, выгрузки и загрузки элементов, арифметические операции, а также дуги между ними. Далее приведено описание методики оценки времени работы программы сверху, использующей диаграммы потоков данных.

Методика построения верхней оценки времени выполнения подпрограммы

При построении верхней оценки за основу берется обобщенный последовательный алгоритм. Далее применяется методика, обобщенное описание которой представлено ниже.

1. Листинг программы разбивается на этапы.
2. Каждый этап отдельно иллюстрируется диаграммой потоков данных.

3. Для каждой диаграммы потоков данных:
3.1. Диаграмма потоков данных преобразуется в развернутую диаграмму.
3.2. Производится кластеризация, т.е. объединение вершин в кластеры.
3.3. Выделяется подграф, представляющий собой тело нового цикла.
3.4. Оценивается количество тактов на выполнение нового тела цикла.
3.5. Вычисляется оценка времени для этапа.
4. Оценки для каждого из этапов объединяются.

Пункт 1 и пункт 2 методики предназначены для повышения удобства оперирования данными, поскольку иллюстрация алгоритма в целом усложняет процесс выявления связей по данным (фрагмент программы приведен в листинге 1, пример диаграммы – на рис. 1). Рассмотрение каждого этапа алгоритма по-отдельности упрощает задачу анализа. Выполняемая в пункте 3.1 развертка диаграммы (рис. 2) соответствует развертке цикла в программе [2]. Кластеризация диаграммы в пункте 3.2 соответствует программной конвейеризации цикла. В тело нового цикла включаются независимые операции, принадлежащие различным итерациям старого цикла [1]. В кластеры включаются те вершины, которые соответствуют операциям, упаковывающимся в минимальное количество инструкций. Таким образом, телу нового цикла соответствует минимальное количество кластеров. Чтобы получить оценку времени выполнения этапа алгоритма

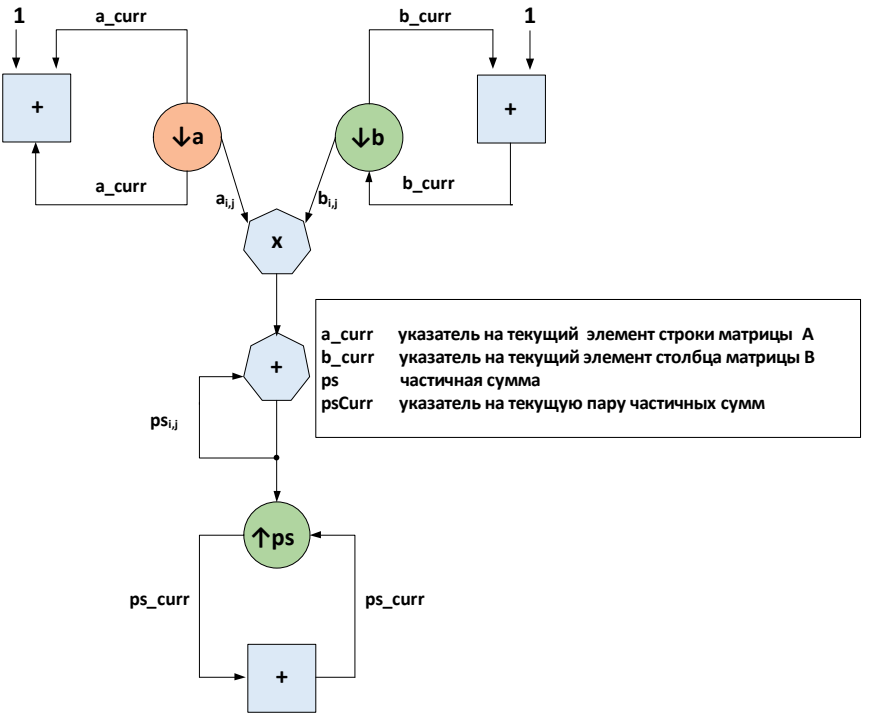


Рис.1. Диаграмма потоков данных для умножения матрицы A на столбец матрицы B

(пункт 3.4) необходимо учесть время входа в цикл, выполнения преамбулы[1] и эпилога[2] (1), где программный конвейер заполняется и сбрасывается соответственно (см. рис. 2, справа–сверху). При выполнении пункта 3.5 время выполнения цикла вычисляется как произведение времени выполнения его тела на количество итераций + время входа в цикл, при этом время выполнения линейных участков программы суммируется (пример расчета приведен в (2)). Полученные частные оценки для этапов объединяются (пункт 4) для получения общей оценки.

$$t - t_{\text{преамбула}} + Nt_{\text{тело цикла}} + L_{\text{эпилог}}, \quad (1)$$

где N – количество итераций конвейеризованного цикла.

$$t = 3 + (M - 3)1 = 4 = (M - 3) + 7 = V + 4 \text{ [такта]}, \quad (2)$$

где M – размерность задачи по измерению j, а количество тактов равно количеству кластеров.

Листинг 1

Умножение матрицы A на вектор– столбец матрицы B

```
1.                                          MOVE A6, R0
2.                                          MOVE R0, AT

3.                                          MOVE 1, IT
4.                                          MOVE A0, R12
5.                      CLRL R16            M A5, R4
6.      DO R28, CR_SGEMM_002
7.                      CLRL R14            M (A0)+, R2         (AT)+IT, R0
8.      DO R26, CR_SGEMM_001
9.      CR_SGEMM_001:
10.        FMPYR2, R0, R14  FADD R16, R14, R16 M (A0)+, R2      (AT)+IT, R0
11.                     FADD R16, R14, R16 M A6, R0
12.                                         MOVE R0, AT
13.                                         MOVE A0, R22
14.                     DEC R22, R22        M R16, (A5)+
15.     CR_SGEMM_002:
16.                     CLRL R16            M R22, A0
17.                                         MOVE R4, A5
18.                                         MOVE R12, A0
```

Для обоснования необходимости оптимизации программы предлагается использовать описанную ранее методику. С помощью предложенной методики можно оценить эффективность существующей реализации алгоритма. Использование методики позволяет определить, когда имеет смысл переписывать программу (в том случае, когда текущая реализация обладает существенно худшей производительностью, чем эталонная программа).

С использованием предложенной методики автору удалось получить верхнюю оценку для SIMD-подпрограммы умножения матриц [3], для которой отклонение составило не более 5%.

---

[1] Преамбулу составляют все кластеры, которые идут до первого кластера, соответствующего инструкции, входящей в тело нового цикла.

[2] Эпилог, в свою очередь, составляют все кластеры, находящиеся ниже кластера, соответствующего последней инструкции, принадлежащей циклу.

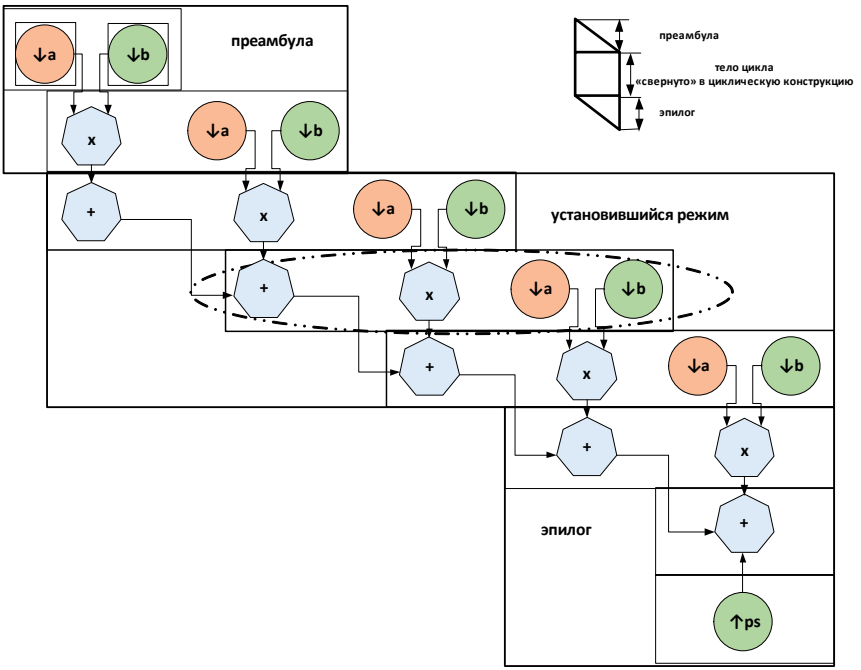


Рис. 2. Диаграмма кластеризации для умножения матрицы A на столбец матрицы B

В дальнейшем планируется расширить класс программ, для которых применима предложенная методика, а именно, обобщить предложенную методику на случай нескольких независимых измерений в задаче.

Библиографический список

1. Bacon D. F., Graham S. L., Sharp O. J. Compiler Transformations for High-Performance Computing // ACIM Computing Surveys. Vol. 26, 1994, pp. 345–420.
2. Лямина Н.Д. Использование методов оптимизации в программах для DSP-ядер многоядерных процессоров семейства «МУЛЬТИКОР» // 62-я международная студенческая научная конференция ГУАП: Сб. докл. / СПБГУАП. СПб., 2009.
3. Лямина Н. Д. Распараллеливание программ для вычислительных ядер в условиях расслоения их локальной памяти // 63-я международная студенческая научная конференция ГУАП: Сб. докл. / СПБГУАП. СПб., 2010.
4. МУЛЬТИКОР – технология проектирования СнК (http://multicore.ru/index.php?id=21).
5. Солохина Т. В., Петричкович Я. Я. Особенности программирования микросхемы 1892ВМ2Т(МС-24) в режиме 2SIMD // Chip News.2005 / № 3. С. 20–26.
6. Солохина Т. В. Иерархический параллелизм архитектуры многоядерных СБИС серии «МУЛЬТИКОР» и многопроцессорных систем на их основе // 2008. С. 5–20.

УДК 004.72

Н. А. Матвеева– студентка кафедры информационных систем
Е. А. Суворова (канд. техн. наук, доц.) – научный руководитель

ИССЛЕДОВАНИЕ АРХИТЕКТУРЫ МАРШРУТИЗИРУЮЩИХ КОММУТАТОРОВ, ПОДДЕРЖИВАЮЩИХ ПОТОКИ ПАКЕТОВ РАЗЛИЧНЫХ УРОВНЕЙ ПРИОРИТЕТОВ

В сетях-на-кристалле широко используется понятие маршрутизирующего коммутатора. Маршрутизирующий коммутатор состоит из множества входных буферов, коммутационной матрицы, множества выходных буферов и некоторой схемы управления. В данной статье рассматривается использование буферов на входах и выходах для локального хранения данных, которые не могут быть немедленно переданы из-за занятости выходного порта или выходного канала коммутатора.

Для организации передачи пакетов с различными приоритетами может использоваться механизм виртуальных каналов. При использовании этого механизма с логической точки зрения каждому виртуальному каналу ставится в соответствие отдельное буферное пространство. Однако на аппаратном уровне при этом могут использоваться различные схемы. Например, для реализации каждого виртуального канала может использоваться физически отдельный блок двухпортовой памяти, либо все виртуальные каналы могут быть реализованы в одном блоке памяти.

В рамках данной статьи будут рассмотрены различные варианты реализации виртуальных каналов, будут рассмотрены коммутаторы, в которых буферная память располагается во входных и выходных портах. Для этих вариантов реализации будут выполнены оценки среднего времени передачи пакетов различных уровней приоритетов при различных параметрах входных потоков данных.

Для проведения исследования была реализована сеть-на-кристалле в модели DNCSimulator. В данной системе во время исследования использовались следующие параметры: длина поля пакета данных – 63 байта; количество точек доступа – от 1 до 4; количество приоритетов пакетов – от 2 до 4; количество буферов приходящихся на один приоритет – от 1 до 20; частота работы системы – 100 МГц. Самым высоким приоритетом обладает пакет с 0 приоритетом. Самым низким оказывается самый старший приоритет из диапазона. Применяется равномерное и экспоненциальное распределение времени между пакетами, при загрузке входных портов 70% и 90%. В маршрутизирующем коммутаторе используется циклическая смена приоритетов обслуживания портов. В данном рассмотрении предполагается, что при передаче через коммутационную матрицу пакеты передаются целиком, их передача не может быть прервана (например, в случае появления пакета с более высоким приоритетом). Предполагается, что все пакеты имеют фиксированную длину. Один пакет размещается в одном буфере (размер буфера соответствует размеру пакета). За каждым из виртуальных каналов закреплен свой уровень приоритета. Все каналы имеют разные уровни приоритета.

В процессе исследования маршрутизирующего коммутатора с 4-мя виртуальными каналами и 16 буферами были получены зависимости среднего времени доставки пакетов от количества буферов закрепленных за каждым виртуальным каналом и соотношения интенсивностей потоков пакетов, изображенные на рис. 1 и 2. На 1-ом графике представлена зависимость при одной точке доступа, а на 2-ом при использовании двух точек доступа.

На этих графиках приведены результаты при равномерном и экспоненциальном распределении времени генерации между пакетами, при загрузке входных портов 70% и 90%. В названии кривых графика первая цифра указывает номер виртуального канала, затем загрузка входного порта, тип распределения времени между генерацией пакетов (равномерный – uni, экспоненциальный – exp) и количество точек доступа.

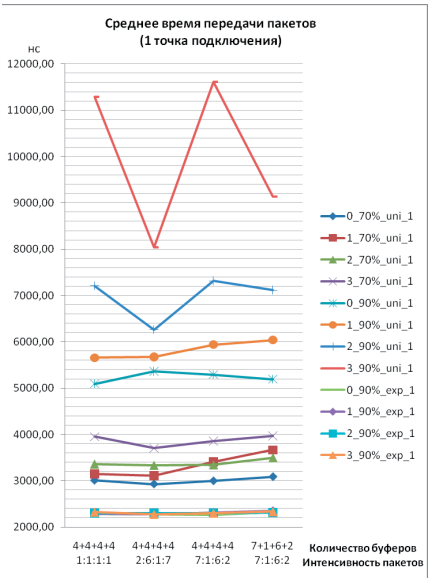


Рис. 1. Среднее время передачи пакетов как функция от количества буферов и интенсивности пакетов (1 точка подключения)

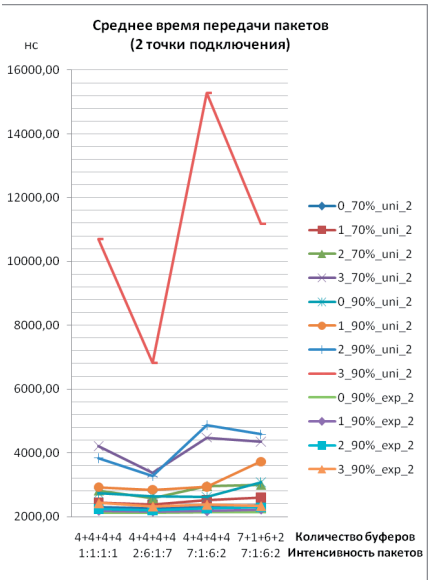


Рис. 2 Среднее время передачи пакетов как функция от количества буферов и интенсивности пакетов (2 точки подключения)

Если интервалы времени между пакетами распределены по экспоненциальному закону, то время доставки пакетов для всех виртуальных каналов практически совпадает и мало зависит от того, насколько интенсивности потоков пакетов, принадлежащих разным виртуальным каналам, соответствуют количеству портов, выделенных для данного виртуального канала.

Если же используется равномерное распределение, то время доставки пакетов более ощутимо зависит от соответствия интенсивностей потоков пакетов количеству выделенных буферов. Однако разброс времени доставки не превосходит 1,5 раза при загрузке 90%, 1,1 раза при загрузке 70% как при одной, так и при двух точках доступа.

Так же было проведено исследование увеличения числа точек доступа при равномерном и экспоненциальном распределении времени генерации между пакетами, при загрузке входных портов на 90%. Полученная зависимость, среднего времени передачи пакетов различных уровней приоритетов от количества точек доступа, представлена на рис. 3.

Из рис. 3 хорошо видно, что особенно ощутимо среднее время доставки сокращается, если время между генерацией пакетов распределено по равномерному закону при переходе от одной точки доступа к двум. При дальнейшем увеличении количества точек доступа время доставки практически не меняется.

Кроме этого исследовалось среднее время доставки пакетов отдельно каждого приоритета. Рисунки 4–7 отражают зависимость среднего времени доставки пакетов разных приоритетов при различном числе точек доступа и различном числе буферов, приходящихся на один приоритет, при равномерном распределении времени генерации пакетов и загрузке каналов на 90%. В названии кривых графика использовались

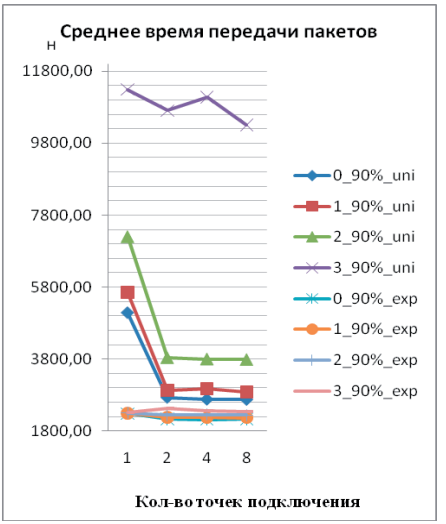


Рис. 3. Соотношение между средним временем передачи пакетов и количеством точек подключения

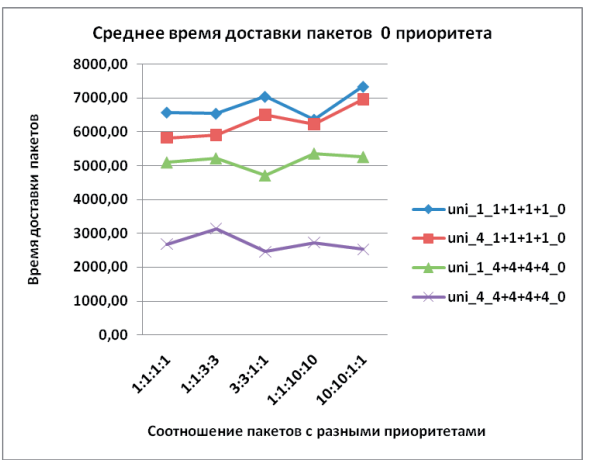


Рис. 4. Среднее время передачи пакетов 0 приоритета

следующие сокращения: uni означает равномерное распределение времени генерации пакетов, следующий символ после него означает количество точек доступа, следующая комбинация означает, сколько буферов приходится на пакет с каждым приоритетом (приоритеты идут от 0 до 3 включительно), последний символ в названии означает номер приоритета пакета.

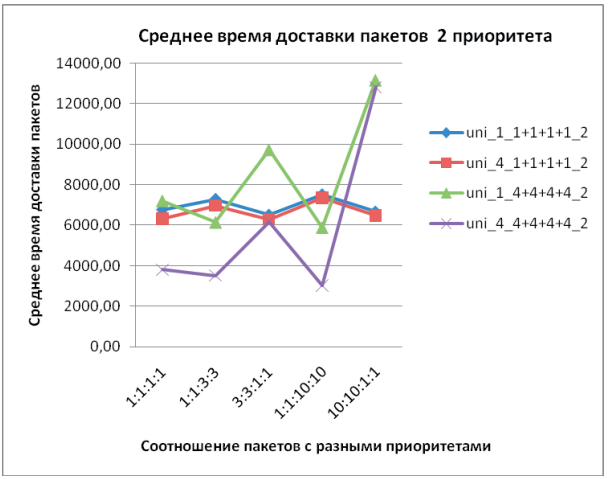


Рис. 5. Среднее время передачи пакетов 2 приоритета

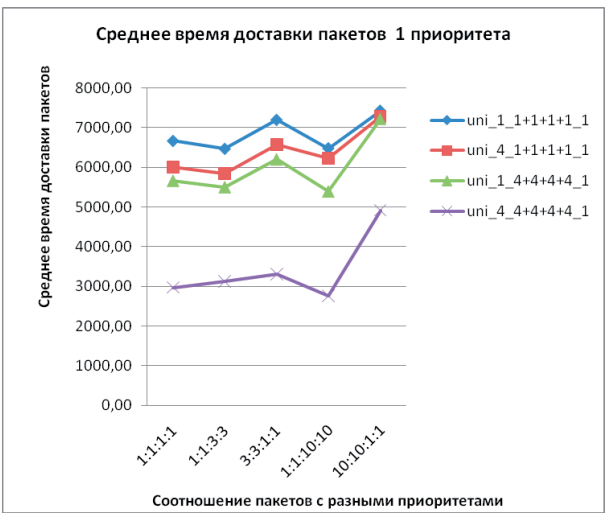


Рис. 6. Среднее время передачи пакетов 1 приоритета

Проводя сравнительную характеристику графиков на рис. 4–7 хорошо видно, что для пакетов с высокими приоритетами (0, 1) увеличение пакетов низкоприоритетных (2, 3) в 3, 10 раз по сравнению с их количеством не приводят к изменению среднего времени доставки высокоприоритетных пакетов. А влияет на это время только количество точек доступа и количество буферов приходящихся на один приоритет.

Рис. 7. Среднее время передачи пакетов 3 приоритета

С низкоприоритетными пакетами ситуация кардинально меняется. При увеличении высокоприоритетных пакетов в 3,10 раз, когда на каждый приоритет приходится по 4 буфера, сильно возрастает время доставки этих пакетов. При увеличении количества высокоприоритетных пакетов в 3 раза при одной точке доступа, время доставки низкоприоритетных пакетов возрастает в среднем в 1,4 раза, а при увеличении в 10 раз время доставки увеличивается в 1,9 раза (для пакета с приоритетом 2).

В ситуации, когда на каждый пакет приходится по 1 буферу для низкоприоритетных пакетов время, их доставки практически не меняется.

Из проведенных исследований можно определить для сетей-на-кристалле взаимосвязь корректного диапазона количества виртуальных каналов для конфигурации маршрутизирующих коммутаторов и параметров, которые могут быть описаны в соответствии с полученными результатами:

– с точки зрения среднего времени доставки пакетов наиболее оптимально, чтобы на одну точку доступа приходилось два или 4 буфера;

– на один порт достаточно двух точек доступа, последующее увеличение их числа не приводит к уменьшению среднего времени передачи пакетов;

– если для системы характерно поступление потоков пакетов по закону, близкому к экспоненциальному без ощутимого ухудшения времени доставки пакетов может использоваться не более двух буферов на один виртуальный канал (дальнейшее увеличение количества буферов практически не приводит к уменьшению времени доставки пакетов);

– если для системы характерно интенсивное поступление пакетов с высокими приоритетами и важно, чтобы пакеты с низкими приоритетами тоже проходили по сети

с меньшим временем, то нужно озаботиться разработкой алгоритма арбитража, который будет обеспечивать продвижение низкоприоритетных пакетов и позволит избежать их сильного торможения в маршрутизирующем коммутаторе.

УДК 681.3

М. А. Пархоменко – студент кафедры информационных систем
Л. В. Коблякова – научный руководитель

ТЕСТИРОВАНИЕ ПАМЯТИ КОММУТАТОРА СЕТИ SPACEWIRE МСК01/02

Любая физическая память может давать сбои во время своей работы, поэтому тестирование памяти – это необходимый инструмент для разработчиков программного обеспечения (ПО), работающего с устройством. Ошибки работы физической памяти могут давать сбои в работе ПО, что может создавать дополнительные проблемы и трудности разработчикам ПО. Например, когда сбой в работе памяти программ (ПП) приводит к изменению хотя бы 1 бита кода программы, устройство может повести себя непредсказуемо: скорее всего, программный счетчик (program counter – PC) окажется за областью кода программы. Еще одна ситуация, при которой ошибка в работе физической памяти оказывает негативное влияние на работу кода внутри устройства – это случай, когда ошибка искажает данные, которыми оперирует программа; данная ошибка вряд ли собьет, но повлияет на работу кода: допустим, мы храним маркеры каких-то процессов в виде маски, которая исказилась в результате ошибки в работе памяти устройства, это приведет к ошибочной обработки данного состояния, так как маска не соответствует действительности.

Так-так ошибки в работе памяти приводят к негативным последствиям их надо выявлять, используя специальные программы-тесты. Тест должен иметь настройки, задаваемые пользователем, и учитывать ошибки различного рода. Минимальный набор параметров теста должен включать в себя адрес ячейки памяти, с которой следует начинать проверку, размер тестируемой области памяти и, желательно, иметь возможность задавать набор выбранных пользователем тестов из всего множества. Результатом работы теста являются данные о наличии ошибок и областях памяти, где они произошли. Так же данные могут включать в себя информацию о роде ошибок и их количестве.

Тестовую программу следует загружать в область памяти, не подвергаемую тестированию, иначе код затрет сам себя. Тестовую программу рекомендуется наделять как скоростными поверхностными тестами, так и сложными, учитывающими более редкие ситуации.

При работе с коммутатором сети SPACEWIRE МСК01/02 были выявлены два основных случая, при которых происходили ошибки: искажение записанной информации по адресу записи и искажение информации в ячейке памяти при попытке записи в совершенно другую ячейку памяти. Вторая ситуация случается намного реже первой. В связи с этим была создана тестовая программа, выявляющая эти ошибки. Ошибки возникали во внешней памяти устройства, поэтому для их регистрирования тестовая программа подгружалась во внутреннюю память кристалла, что не давало возможности коду перезаписать самого себя. Параметры теста включают в себя адрес тестируемой области памяти, ее размер, а так же шкалу, задающую селекцию тестов из множества, включающего в себя двадцать один тест. Из них один тест является поверхностным и не требует большого количества времени для проверки памяти, а двадцать других являются сложными тестами, выявляющими все возможны ошибки.

Простой тест проходит все ячейки тестируемой области памяти, записывает в них последовательный ряд чисел, после чего читает данную область памяти и проверяет считанные значения на предмет искажений. Сложные тесты записывают в тестируемую область памяти заполнители, после чего происходит проверка всей области памяти на наличие искажений при записи заполнителей. Затем тест начинает проходить все ячейки памяти, записывая в них маски и проверяя их на наличие искажений при записи, после чего тест проверяет всю тестируемую область памяти на наличие искажений, происходящих вследствие записи в данную ячейку. То – есть, записав данные в одну ячейку памяти, тест проверяет другие ячейки памяти на наличие искажений в них, что и дает нам возможность зафиксировать сложные ошибки. Сложные тесты отличаются друг от друга заполнителями и маской. Маска не должна соответствовать заполнителю, а их множество должно обеспечивать имитацию всевозможных ситуаций, происходящих в памяти при работе данного устройства. Для этого можно использовать информационные последовательности различного вида. Основные общепринятые варианты – это последовательности всех нулей, всех единиц, чередование единица нуль, чередование нуль единица, а так же для имитации записи более ли менее случайного числа с возможностью его дальнейшей корреляции можно записывать в ячейки их собственные адреса (табл. 1).

Таблица 1

Таблица масок

Шестнадцатеричный вид маски	Двоичный вид маски
0x00000000	0000 0000 0000 0000 0000 0000 0000 0000
0xFFFFFFFF	1111 1111 1111 1111 1111 1111 1111 1111
0xAAAAAAAA	1010 1010 1010 1010 1010 1010 1010 1010
0x55555555	0101 0101 0101 0101 0101 0101 0101 0101
0x982FC000 (любой адрес)	1001 1000 0010 1111 1100 0000 0000 0000

Результаты проверки памяти программа хранит в виде шкал. Одна шкала хранит информацию о номерах тестов, при выполнении которых возникли ошибки, и род этих ошибок, другая шкала хранит в себе данные о количестве ошибок и адресе ячейки, в которой возникла первая ошибка. Ошибки были разделены на 3 типа (ошибка при записи заполнителей, ошибка при записи в ячейку, искажение ячейки при записи в другую ячейку памяти). Это позволяет пользователю понять, в какой ситуации возникает данная ошибка. Данные хранятся персонально для каждого теста из выбранного пользователем множества.

Таким образом, разработчик ПО может исследовать быстрым тестом всю внешнюю память устройства, тем самым выяснив, какие области памяти дают сбои во время работы. После чего он сможет загрузить свой код в исправную область памяти, а при возникновении непредвиденных ошибок во время работы кода, проверить подозрительную подобласть более сложным тестом.

Тестирование памяти является необходимым процессом, так-так никто не застрахован от проблем, возникающих при работе с ней. Так же стоит отметить, что, как правило, быстрее сразу проверить память, чем тратить большое количество времени на поиск непонятных ошибок, возникающих во время работы кода.

УДК 629.735.33

А. А. Сетков – магистрант кафедры информационных систем
А. В. Белоголовый (канд. техн. наук, ст. науч. сотр.) – научный руководитель

ИССЛЕДОВАНИЕ КАЧЕСТВА И ПРОИЗВОДИТЕЛЬНОСТИ АЛГОРИТМОВ ОЦЕНКИ ДВИЖЕНИЯ

Каждый год в мире увеличиваются пропускные способности скоростей интернет-каналов, появляются высокопроизводительные вычислительные комплексы, повышается степень параллельности программ. В этой связи, вопросы качества и скорости обработки информации остаются по-прежнему актуальными. Передача и обработка видеосигналов является частным случаем передачи информации. На данный момент существует большое количество видеокодеков, осуществляющих кодирование и сжатие видео с разной скоростью и с разным качеством выходной видеопоследовательности. Для передачи потокового видео через интернет особую актуальность приобретают видеокодеки, способные менять скорость обработки, качество и размер видео в зависимости от условий передачи (различная скорость интернет-канала на разных сегментах сети, интенсивность и загрузка элементов сети, ошибки при передаче, производительные возможности конечных устройств-приемников). Примером такого рода кодеков является H.264 SVC (Scalable Video Codec).

В последнее время в мире ведется работа над увеличением скорости и уменьшением затрат на обработку без потери в качестве получаемого видеосигнала. Это достигается двумя способами: частичным или полным переносом алгоритмов видеокодирования на многоядреные высокопроизводительные платформы и выбором быстрых алгоритмов обработки, не всегда, однако, гарантирующих соответствующее обычным алгоритмам качество.

Существует большое количество алгоритмов блокового поиска для оценки смещения блоков текущего кадра и одного или нескольких базовых (опорных) кадров. Их можно разделить на два класса: осуществляющие поиск полным перебором и быстрые алгоритмы. В данной статье будет изучен алгоритм оценки на основе быстрого преобразования фурье (БПФ). Также будет показано, что применение данного метода для задач кодирования видео может существенно упростить сжатие без потери качества.

Оценка движения (motion estimation) в последовательности видеокадров является важной частью во многих областях обработки изображений, в том числе, и в видео кодировании. Схемы кодирования зачастую используют высокую временную избыточность между кадрами в последовательности, предсказывая текущий кадр по предыдущему с использованием поля векторов движения (Motion Vector Field). Ошибка предсказания, которая является разностью между предыдущим кадром после оценки движения и текущим, кодируется и передается. Если декодеру не известен алгоритм оценки движения, то также передается поле векторов движения. Обычно, ошибка предсказания содержит намного меньше информации, чем оригинальный кадр, если была произведена точная оценка, и тем самым достигается большой коэффициент сжатия. [1]

Существует несколько метрик, позволяющих оценить качество поиска. Сумма абсолютных разностей (SAD – Sum Of Absolute Differences) широко распространенный, простой алгоритм для оценки похожести двух блоков в последовательности кадров. Вычисляется путем суммирования абсолютных разностей яркостей пикселей блоков базового и текущего кадров:

$$SAD(x,y) = \sum_{i=0}^{n-1} \sum_{k=0}^{m-1} | f(x+k, y+l) - g(k,l) |$$

где f и g – кадр, в котором осуществляется поиск, и искомый домен соответственно.

Метрика SSD (Sum of Square Differences) – сумма квадратов разностей яркостей пикселей, является более точной, хотя и применяется реже, чем SAD, из-за операции возведения в квадрат.[3]

$$SSD(x,y) = \sum_{i=0}^{n-1} \sum_{k=0}^{m-1} (f(x+k, y+l) - g(k,l))^2$$

Все алгоритмы оценки движения делятся на блоковые алгоритмы поиска объектов квадратной или прямоугольной формы (Block Matching), поиск объектов сложной и произвольной формы (Object Matching) и глобальный поиск, где глобальный вектор движения получается из локальных на основе вероятности (Global Matching). Алгоритмы, принадлежащие к первой группе, – наиболее часто используемы (примерно 90%) из-за простоты реализации и меньшего времени вычисления, что существенно для обработки видеопоследовательностей в режиме реального времени. Вторая группа алгоритмов, более сложная, также присутствует в стандартах некоторых видеокодеков. Третья группа используется для устранения временной разности, связанной с сотрясениями камеры.

В типичных алгоритмах блокового поиска текущий кадр разбивается на блоки, и цель оценки заключается в поиске соответствующих векторов движения для каждого блока, в соответствии с их реальным смещением в предыдущем кадре. Существует большое количество разных вариантов таких алгоритмов. Все они приблизительно подразделяются на две категории: алгоритмы, гарантирующие и не гарантирующие нахождение лучшего совпадения блоков. Самые распространенные алгоритмы второго типа достигают ускорения благодаря раннему исключению некоторых векторов-кандидатов из поиска. Они вычисляют нижнюю границу для текущего вектора, сравнивают лучший найденный SAD до достижения границы, а потом удаляют из рассмотрения вектор-кандидат, чья нижняя граница больше (хуже). Однако, проблема с алгоритмами полного поиска, использующими раннее исключение кандидатов, заключается в непредсказуемом времени вычисления. Если видеопоследовательность зашумлена, или в ней много движения, эти алгоритмы будут исключать только малую часть векторов движения – кандидатов и требуют много вычислений. Когда необходимо для каждого положения вычислить сумму абсолютных разностей, прямые вычисления будут требовать намного больше времени и вычислительных ресурсов.

Некоторые из наиболее популярных методов второго типа: Трехшаговый поиск (TSS), Новый трехшаговый поиск (NTSS), простой и эффективных TSS (SES), четырехшаговый поиск (4SS), поиск ромбом (DS), адаптивный поиск по образцу крестом (ARPS) и т. д. Эти типы алгоритмов, однако, склонны к попаданию в локальный минимум.

Все предыдущие алгоритмы производят вычисление в пространственной области. В качестве альтернативы, подобные вычисления можно производить в частотной области. В данном случае будут оцениваться не соответствие блоков по яркости, а фазовая корреляция двух областей напрямую по их фазам. Использование преобразования Фурье для оценки движения позволяет существенно ускорить вычисления, благодаря быстрым БПФ алгоритмам, которые имеют вычислительную сложность O(Nlog(N)). Новый подход для вычисления любой кернфункции был недавно предложен Фитчем.[3] Их подход работает посредством представления входной сопоставляемой поверхности последовательностью членов, содержащих косинус. Чем больше число этих членов, тем медленнее алгоритм, однако, тем больше точность вычислений. Теорема Фурье формулирует, что любая непрерывная функция может быть представлена последовательностью синусоид. Главная сложность заключается в том, что мы должны иметь дело с бесконечной последовательностью, а не с конечными суммами. Некоторые кернфункции могут быть легко представлены в частотной области, используя конечное число членов разложения, содержащие косинус. [2]

В данной статье будет показано, что уже на первом разложении быстрый частотный алгоритм даст результат, сравнимый с полным перебором, за гораздо меньшее время.

В данном методе функция нахождения SAD будет аппроксимироваться членами разложения Фурье. Из-за краткости мы опустим из рассмотрения вывод формулы формулы (1).

$$SCD_L(x,y) = \Re IFFT\left\{\left(\sum_{p=1}^{i} \frac{1}{(2p-1)^2} F_p(u,v) G_p(u,v)\right)\right\} \quad (1)$$

где G_p – БПФ от искомого домена и F_p – БПФ от области поиска базового кадра, $\Re$ – действительная часть комплексного числа и * – комплексно сопряженное число [2]

Оценка алгоритмов производилась по нескольким критериям: теоретическая сложность алгоритмов, вероятность нахождения правильных векторов смещения в сравнении с найденными по метрике SAD и SSD, а также вычисление энергии разности остаточного кадра между восстановленным и исходным кадрами по метрике PSNR.

Вначале была оценена сложность. Так как эти алгоритмы детерминированы, под сложностью будем понимать точное количество операций, а не их аппроксимацию. При подсчете суммировались операции всех типов: присваивания, сложения, умножения, вычисление синуса/косинуса и операции БПФ/ОБПФ. Результаты расчетов для различных размеров области поиска и домена приведены в табл. 1. В БПФ-алгоритме уровень разложения L=1.

Общее количество операций в алгоритме нахождения SAD полным перебором:

$$O = (M - m + 1)^2 \cdot m^2,$$

где M – это область поиска, m – размер домена.

Общее количество операций в алгоритме нахождения SAD в частотной области:

$$M^2 \cdot (2 + 36 \cdot L + \log M \cdot (4 \cdot L + 2)),$$

где L – уровень разложения.

Таблица 1

Общее количество операций

Область поиска/ размер домена	Полный перебор SAD	БПФ метод	Ускорение БПФ-метода, %
64/16	1844017	327680	82.23
64/8	623865	327680	47.48
64/4	178669	327680	-83.4
32/16	221969	75776	65.86
32/8	120025	75776	36.87
32/4	40397	75776	-87.58

Как видно из табл. 1, метод полного перебора работает быстрее для размеров домена 4x4, в остальных случаях БПФ-метод оказывается эффективнее. Это объясняется не зависимостью этого метода от размера домена.

Из этого можно сделать вывод, что маленькие размеры домена выгоднее искать полным перебором, а большие – частотным методом.

С увеличением размера области поиска ускорение БПФ метода становится более наглядным. Для размера области поиска 64x64, ускорение достигает 82%. Это хороший результат для размеров, которые часто используются при видео кодировании.

Качество поиска измерялось несколькими способами: сравнивалось количество ошибочно найденных блоков в сравнении с полным перебором по метрикам SAD и SSD. Также вычислялась энергия разностного кадра по PSNR метрике.

Эксперименты над качеством поиска производились на видеопоследовательности carphone, на первых 150 кадрах с расширением 352x288. Радиус поиска был равен 32x32, размеры домена: 4x4, 8x8, 16x16. Уровни разложения L = 1, 5, 10.

В табл. 2 представлены результаты экспериментов. Первым критерием, который был использован для оценки качества, была вероятность ошибки нахождения домена по метрике SAD. Ранее было показано, что мы аппроксимировали функцию вычисления SAD последовательностью членов разложения Фурье. С увеличением количества разложений вероятность ошибки падает. Однако сама ошибка остается существенной.

В сравнении с SSD метрикой получились более низкие вероятности ошибки. Для достижения наилучшего результата достаточно только одного разложения чтобы получить точность, сравнимую с полным перебором по SSD метрике.

Таблица 2

Результаты экспериментов блокового поиска БПФ-алгоритма с поиском полным перебором SAD и SSD

Размер области поиска / домена		L=1	L=5	L=10	SAD	SSD
32/16	Средняя вероятность ошибки поиска по SAD метрике, %	9,24	8,34	7,05	0	9,23
	Средняя вероятность ошибки поиска по SSD метрике, %	0,15	1,78	3,36	9,23	0
	среднее значение PSNR	35,89	35,88	35,86	35,83	35,9
32/8	Средняя вероятность ошибки поиска по SAD метрике, %	13,95	14,17	13,27	0	11,19
	Средняя вероятность ошибки поиска по SSD метрике, %	2,12	3,31	4,44	11,19	0
	среднее значение PSNR	39,69	39,68	39,64	39,57	39,69
32/4	Средняя вероятность ошибки поиска по SAD метрике, %	21,46	22,30	21,88	0	13,60
	Средняя вероятность ошибки поиска по SSD метрике, %	8,16	9,39	10,05	13,60	0
	среднее значение PSNR	42,16	42,15	42,12	42	42,16
48/16	Средняя вероятность ошибки поиска по SAD метрике, %	11,13	10,32	8,97	0	10,88
	Средняя вероятность ошибки поиска по SSD метрике, %	0,37	1,52	3,36	10,88	0
	среднее значение PSNR	38,23	38,21	38,17	38,1	38,23
48/8	Средняя вероятность ошибки поиска по SAD метрике, %	15,04	14,59	13,79	0	12,20
	Средняя вероятность ошибки поиска по SSD метрике, %	2,94	3,44	4,67	12,20	0
	среднее значение PSNR	39,84	39,82	39,78	39,71	39,84

Таблица 2 (окончание)

Размер области поиска / домена		L=1	L=5	L=10	SAD	SSD
48/4	Средняя вероятность ошибки поиска по SAD метрике, %	23,43	22,99	22,56	0	14,33
	Средняя вероятность ошибки поиска по SSD метрике, %	9,44	9,33	9,99	14,33	0
	среднее значение PSNR	42,46	42,46	42,43	42,3	42,46
64/16	Средняя вероятность ошибки поиска по SAD метрике, %	11,24	10,50	9,17	0	11,04
	Средняя вероятность ошибки поиска по SSD метрике, %	0,30	1,56	3,43	11,04	0
	среднее значение PSNR	38,3	38,28	38,23	38,17	38,3
64/8	Средняя вероятность ошибки поиска по SAD метрике, %	15,16	15,00	14,20	0	12,40
	Средняя вероятность ошибки поиска по SSD метрике, %	2,86	3,65	4,81	12,40	0
	среднее значение PSNR	39,94	39,93	39,89	39,82	39,94
64/4	Средняя вероятность ошибки поиска по SAD метрике, %	23,74	24,18	23,72	0	14,82
	Средняя вероятность ошибки поиска по SSD метрике, %	9,26	10,07	10,64	14,82	0
	среднее значение PSNR	42,75	42,74	42,71	42,57	42,75

В табл. 2 также представлены результаты PSNR. Один уровень разложения дает наилучший результат.

Исходя из результатов теоретических расчетов и экспериментов, можно сделать заключение о том, что в БПФ-алгоритме уже на первом уровне разложения достигается точность поиска, сравнимая с метрикой SSD, и потому можно остановиться и не продолжать дальше разложение.

Домены небольшого размера (4x4) лучше искать методом полного перебора. В остальных случаях существенный выигрыш по количеству операций и по времени выполнения дает БПФ метод.

Библиографический список

1. Young Robert W., Kingsbury Nick G.: Frequency-domain Motion Estimation Using a Complex Lapped Transform. In: IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 2, NO 1 (1993).
2. Essannouni F., Oulad Haj Thami R., Aboutajdine D., Salam A.: Adjustable SAD matching algorithm using frequency domain. In: Springer-Verlag pp 257–265 (2007).
3. E. G. Richardson, Iain., «H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia». Chichester: John Wiley & Sons Ltd., 2003.

УДК 681.3

К. Н. Храменкова – студентка кафедры информационных систем
Л. В. Коблякова – научный руководитель

АДМИНИСТРИРОВАНИЕ И КОНФИГУРИРОВАНИЕ РАСПРЕДЕЛЕННЫХ СИСТЕМ ПО СТАНДАРТУ SPACEWIRE

Конфигурирование и администрирование распределенных систем является одной из важных задач при построении и эксплуатации распределенных бортовых систем по стандарту SpaceWire.

Стандарт SpaceWire определяет средства физического взаимодействия и протоколы надёжной передачи данных по высокоскоростным каналам 2 – 400 Мб/с. Канал SpaceWire – это последовательный полнодуплексный канал. Стандарт описывает разъёмы, кабели, электрические параметры и логические протоколы, составляющие канал передачи данных SpaceWire. SpaceWire определяет методы передачи данных от источника к приёмнику. Содержимое пакетов стандартом SpaceWire не регламентируется.

После подключения всех узлов и коммутаторов распределенной системы, необходимо ее сконфигурировать, то есть сопоставить всем узлам логические адреса, прописать таблицы маршрутизации во всех коммутаторах, задать адаптивно-групповую маршрутизацию и другие параметры, специфические для конкретной системы.

Так же необходимо иметь доступ к статистической информации об ошибках и сбоях, происходящих в системе. От выбора метода конфигурирования и администрирования зависит эффективность и удобство работы с распределенной системой.

Следует разделять конфигурирование коммутаторов и узлов.

Конфигурирование коммутаторов – это, по сути, конфигурирование коммуникационной сети, которая обеспечивает передачу пакетов между узлами в системе.

Конфигурирование узлов – это конфигурирование оконечных устройств, которые могут быть, например, датчиками, вычислительными узлами, выполняющими обработку информации, или на которых работают программы пользователей. Поэтому, для узлов и коммутаторов могут использоваться различные методы.

К основным методам конфигурирования и администрирования коммутаторов можно отнести:

– локальное;

– удаленное;

– автоматизированный процесс конфигурирования и администрирования сети (Plug'n'Play). [1]

Рассмотрим локальное администрирование.

В большинстве случаев конфигурирование коммутаторов производится с использованием команд чтения и записи, соответствующих программно-доступных компонент коммутатора. К каждому коммутатору в системе через COM-порт (PCI или USB) подключается персональный компьютер, на котором работает программа администрирования – рабочее место администратора коммутатора. Для того чтобы настроить какой-либо параметр коммутатора, необходимо с помощью программы администрирования послать соответствующую команду через COM-порт. Встроенное программное обеспечение коммутатора обработает эту команду и пришлет ответ с результатом. Для того чтобы встроенное программное обеспечение могло обработать команду записи (или чтения), она должна содержать информацию об адресе программно-доступного компонента в адресном пространстве коммутатора. [1]

Рассмотрим пример изображенный на рис. 1. К персональному компьютеру через COM-порт подключен коммутатор SpaceWire, к которому в свою очередь подключены другие устройства по портам 3, 6 и 14. На персональном компьютере (ПК) запущено программное обеспечение администрирования коммутаторов. С помощью этого при-

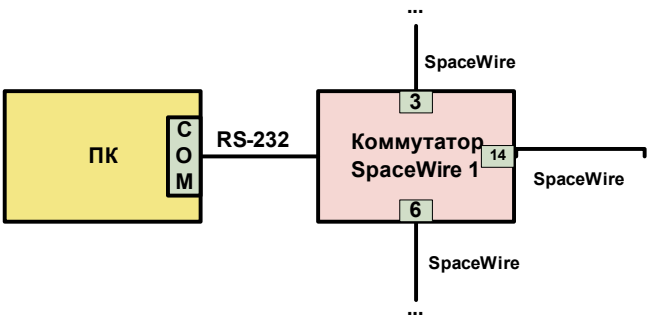


Рис. 1. Часть сети SpaceWire для локального администрирования

ложения можно прочитать регистр «Активные соединения» коммутатора и увидеть, что порты 3, 6 и 14 активны.

Рассмотрим удаленное администрирование.

При удаленном конфигурировании коммутатора используется протокол удаленного доступа в память (RMAP). RMAP может быть использован для конфигурирования коммутаторов SpaceWire, установки их рабочих параметров и записи информации в таблицу маршрутизации. Также он может использоваться для контроля состояния этих коммутаторов. RMAP может применяться для конфигурирования и считывания состояния узлов сети SpaceWire. С помощью этого стандарта появляется возможность читать и изменять программно-доступные компоненты любого сколь удаленного устройства.[2]

Каждый программно-доступный компонент любого коммутатора в системе может быть адресован как обычная ячейка памяти, в которую можно записывать значения (если регистр доступен на запись), и считывать значения. Как видно из формата пакета, под адрес ячейки памяти отводится 32 бита, а также 8 бит для поля расширения адреса, причем RMAP не описывает, как именно должны интерпретироваться адресные поля. Проблемы с адресацией в данном методе точно такие же, как описано в первом методе. Поэтому удобно интерпретировать адресные поля следующим образом. Если поле расширения адреса нулевое, то рассматриваем поле адреса как непосредственный адрес ячейки памяти, если поле расширения адреса имеет отличное от нуля значение, то это значение определяет тип используемого селектора. При необходимости RMAP позволяет перед записью выполнять проверку данных с использованием CRC, а также отсылать или не отсылать подтверждения.

Для удаленного конфигурирования программа администрирования формирует RMAP-пакет, отсылает его на подключенный к персональному компьютеру через COM-порт (USB или PCI) коммутатор, который пересылает этот пакет в сеть в соответствии с указанным в пакете путевым адресом. RMAP-пакет дойдет до требуемого коммутатора, который выполнит команду, при необходимости сформирует и отошлет ответный RMAP-пакет. Этот пакет дойдет до коммутатора-источника, который перешлет его через COM-порт на персональный компьютер. [1]

Рассмотрим пример на рис. 2. С помощью программы администрирования, работающей в удаленном режиме на персональном компьютере (ПК), можно конфигурировать коммутаторы 1 и 2, а так же другие, если такие имеются в сети SpaceWire. В данном случае, как можно заметить, не требуется двух персональных компьютеров, с работающим программным обеспечением администрирования. Достаточно, одного ПК, подключенного через COM – порт к коммутатору, чтобы конфигурировать второй коммутатор (при наличии соединения) и другие.

Под автоматизированным процессом конфигурирования и администрирования сети понимается, что при включении всех устройств системы, сами устройства по специальному алгоритму исследуют и определяют топологию распределенной сети, выдают логические адреса узлам, настраивают таблицы маршрутизации и другие необходимые параметры системы, а также снабжают данной информацией все другие устройства сети, которые этого требуют. Во время штатной работы сети подключение и отключение новых устройств также определяется автоматически.

Все алгоритмы Plug'n'Play в общем случае можно разделить на два вида.

1. Централизованные алгоритмы. В системе есть одно устройство, которое инициирует и полностью управляет процессом Plug'n'Play.

2. Децентрализованные алгоритмы. В системе есть несколько центров, которые отвечают за процесс Plug'n'Play. В этом случае при выходе из строя одного или нескольких таких устройств система будет работоспособной. Но с другой стороны, это накладывает дополнительные требования к сети и центрам, которые осуществляют исследование сети. [1] Необходимо разрешить арбитраж между этими центрами в такой ситуации, как например, если на узел претендуют два центра; если коммутатор уже настроен первым центром, то, как об этом сообщить второму; и кому настраивать отдельные коммутаторы в процессе исследования сети.

На рис. 3 проиллюстрированы оба случая. При централизованном алгоритме в сети существует только один сетевой менеджер (Network Manager 1), который исследует сеть. При децентрализованном алгоритме таких устройств от двух и выше (Network Manager 1 и Network Manager 2).

Также следует отметить, что алгоритмы Plug'n'Play могут быть разного уровня сложности. Некоторые могут применяться для простых регулярных структур, некото-

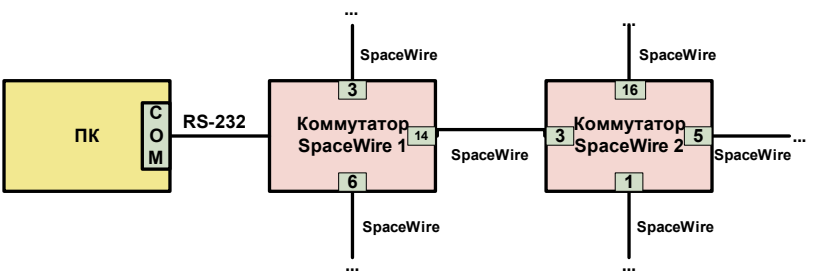


Рис. 2. Часть сети SpaceWire для удаленного администрирования

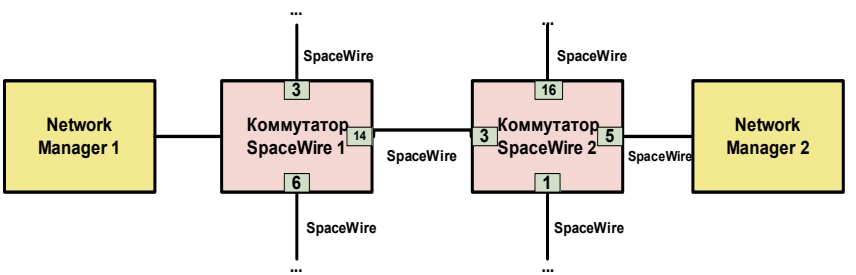


Рис. 3. Часть сети SpaceWire при автоматизированном процессе администрирования

рые – для сети произвольной топологии. Какие – то алгоритмы могут быть более сложными и учитывать множество особенностей сети, учитывать собираемую во время штатной работы статистику и т. д. Поэтому некоторые алгоритмы могут обладать рядом ограничений, наиболее распространенные из которых следующие: алгоритм может быть применен не для всех топологий сети; решение, которое дает алгоритм Plug'n'Play, может быть не оптимальным. Так же следует учесть аппаратуру, которая может отличаться друг от друга набором параметров и функций, то есть добавляется ограничение или дополнительное условие к применению алгоритма Plug'n'Play: в алгоритме будет оговариваться набор параметров и функций, которые необходимы для полного и корректного функционирования алгоритма. Создание же универсального алгоритма является крайне сложной задачей.

Одной из возможных реализаций алгоритма Plug'n'Play является использование протокола удаленного доступа к памяти RMAP, но возможны и другие варианты, например, создание нового протокола передачи данных, что принесет некоторые изменения в программном обеспечении как коммутаторов, так и узлов.

Существуют и требования к технологии Plug'n'Play, среди них:

1. Алгоритм не должен приводить к неопределенным, непредсказуемым результатам или к тупикам. Plug'n'Play должен позволять поддерживать сеть в состоянии, пригодном для решения пользовательских задач как можно дольше.

2. Используется всего два типа устройств (маршрутизирующий коммутатор и узел) для построения любой сетевой структуры

3. Технология должна предоставлять возможность в будущем добавлять новые аппаратные устройств. При этом не должно быть большой нагрузки на сетевую структуру, конфигурация должна выполняться быстро, и при этом не должна существенно сказываться на вычислительной способности сети

4. Данная технология должна реализовываться с минимальной стоимостью, оборудование устройств должно занимать минимум места на плате и потреблять минимум энергии.[3]

Существуют различные варианты и модификации алгоритма P'n'P. Среди них: алгоритмы, предложенные в университете Dundee и космическим агентством NASA.

Технология Plug'n'Play на данном этапе развития стандарта SpaceWire пока не стандартизована. Специально для его разработки создана рабочая группа SpaceWire-PnP – совместный проект NASA и других партнеров.

Библиографический список

1. Онищенко Л. В. Методы администрирования распределенных систем SpaceWire Научная сессия ГУАП, 2008.
2. «SpaceWire RMAP Protocol», ECSS-E-50-11 Draft E, 56 с.
3. Koblyakova L., Setkov A., Khramenkova K. Plug'n'Play Technology in SpaceWire Network. 7th Conference of Open Innovations Framework Program FRUCT, Spb, Russia, 2010. ISBN 978-5-8088-0529-3, p.59–63.