СПИСОК ЛИТЕРАТУРЫ

1. Vermote E.F., Vermeulen A. Atmospheric correction algorithm: spectral reflectances (MOD09) //Algorithm Theoretical Background Document, version 4.0. 1999. URL: http://modis.gsfc.nasa.gov/data/atbd/atbd_mod08.pdf (дата обращения 10.09.2012).
2. Rahman H., Dedieu G. SMAC: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum // Int. J. Remote Sensing. – 1994. – V. 15. – P. 123–143.
3. Richter R., Mueller A., Heiden U. Aspects of operational atmospheric correction of hyperspectral imagery // Int. J. Remote Sensing. – 2002. – V. 23. – P. 145–157.
4. Schlaepfer D., Richter R. Geo-atmospheric processing of airborne imaging spectrometry data. P. 1. Parametric orthorectification // Int. J. Remote Sensing. – 2002. – V. 23. – P. 2609–2630.
5. Irish R.R., Barker J.L., Goward S.N., Arvidson T. Characterization of the Landsat-7 ETM Automated Cloud-Cover Assessment (ACCA) // Algorithm. Photogrammetric Engineering & Remote Sensing. – 2006. – V. 72. – № 10. – P. 1179–1188.
6. Richter R. Atmospheric correction of satellite data with haze removal including a haze/clear transition region // Computers and Geosciences. – 1996. – V. 22. – P. 675–681.
7. Crist E.P., Cicone R.C. A physically-based transform of Thematic Mapper data – the Tasseled Cap // IEEE Trans. Geosci. Remote Sensing. – 1984 – V. GE-22. – P. 256–263.
8. Zhang Y., Guindon B., Cihlar J. An image transform to characterize and compensate for spatial variations in thin cloud contamination of Landsat images // Remote Sensing of Environment. – 2002. – V. 82. – P. 173–187.

Поступила 12.09.2012 г.

УДК 004.54

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ UNITESK ДЛЯ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ СЕТЬЮ ШИРОКОПОЛОСНОГО БЕСПРОВОДНОГО ДОСТУПА СТАНДАРТА IEEE 802.16

И.В. Бойченко, Е.В. Бортников, А.А. Немеров

Томский государственный университет систем управления и радиоэлектроники
E-mail: nemerov@asu.tusur.ru

Описана технология UniTESK и ее применение для функционального тестирования системы управления сетью широкополосного беспроводного доступа (СУ СШБД) стандарта IEEE 802.16 (WiMAX). Показано, что технология UniTESK достаточно проста и эффективна для тестирования событийно-управляемых систем. Рассматривается схема взаимодействия тестирующей и целевой системы.

Ключевые слова:
Коммуникационный протокол, широкополосный беспроводной доступ, система управления сетью, тестирование программного обеспечения, UniTESK, CTESK, WiMAX, тестирование.

Key words:
Communication protocol, broadband wireless access, network management system, software testing, UniTESK, CTESK, WiMAX, testing.

Введение

В настоящее время все большее распространение получают сети широкополосного беспроводного доступа. Это распространение вызвано появлением широкого спектра портативных устройств (ноутбуки, смартфоны), а широкополосные беспроводные сети, как известно, позволяют успешно решить проблему «последней мили». Также, беспроводная сеть может быть развернута в географически труднодоступных районах, где прокладка обычной кабельной сети сопряжена со значительными затратами или вообще невозможна.

В общем виде сети широкополосного беспроводного доступа состоят из следующих основных частей: базовых (БС) и абонентских станций (АС), а также оборудования, связывающего базовые станции между собой, с поставщиком сервисов и глобальной сетью Интернет. Для соединения базовой станции с абонентской, как правило, используется высокочастотный диапазон радиоволн.

Стандарт IEEE 802.16 описывает сеть широкополосного беспроводного доступа масштаба города (MAN – Metropolitan Area Network). Коммерческое название сетей, работающих по протоколу 802.16 – WiMAX.

Сеть WiMAX представляет собой совокупность беспроводного и базового (опорного) сегментов. Беспроводной сегмент определен стандартом IEEE 802.16 [1], базовый – определяется спецификациями WiMAX Forum (WMF). Базовый сегмент – это все, что не относится к радиосети, т. е. связь базовых станций друг с другом, связь с локальными и глобальными сетями (в том числе с Интернет). Базовый сегмент основывается на IP-протоколах и стандартах Ethernet (IEEE 802.3). Однако, собственно архитектура сети, включая механизмы криптозащиты, роуминга, хэндовера и т. п., описывается в документах WMF [2]. Стандарт IEEE 802.16 описывает физический и MAC-уровни сети.

Архитектура WiMAX-сети обеспечивает независимость архитектуры сети доступа, включая радиосеть, от функций и структуры транспортной IP-сети. Масштабируемость и гибкость WiMAX-сети возможна по таким эксплуатационным параметрам, как число абонентов, географическая протяженность зоны покрытия, частотные диапазоны, топология сети, мобильность абонентов [2].

Протоколы сетей широкополосного беспроводного доступа отличаются большей сложностью по сравнению с проводными сетями, такими как Ethernet, или беспроводными сетями с небольшой зоной покрытия, например, WiFi. В сети возможно большое число событий и состояний, предусмотренных протоколом. Это связано, во-первых, с нестабильностью беспроводных каналов, и, следовательно, необходимо выполнять мониторинг и подстройку. Во-вторых, протокол ориентирован на соединения с обеспечением качества обслуживания и основной контроль по управлению сетью должна брать на себя базовая станция. Все возможные события сети WiMAX описаны в стандарте IEEE 802.16 в виде сообщений.

В связи со сложностью протокола 802.16, усложняется также и тестирование реализации системы управления сетью на предмет соответствия протоколу, то есть проверка того, что на данное конкретное сообщение система управления сетью реагирует именно так, как того требует стандарт.

Суммарный объем документации стандарта IEEE вместе с рекомендациями WMF составляет несколько тысяч страниц. Как будет показано далее, технология UniTESK позволяет перевести технические спецификации стандарта на формальный язык автоматических тестов, не зависящих от конкретной программной или схемотехнической реализации.

Реализация системы управления сетью стандарта IEEE 802.16

Все управляющие воздействия, описанные в стандарте IEEE 802.16, передаются между программными модулями, в том числе между различными станциями, посредством управляющих MAC-сообщений.

Разрабатываемая кафедрой Автоматизированных систем управления (АСУ) Томского государственного университета систем управления и радиоэлектроники (ТУСУР) совместно с ЗАО «НПФ МИКРАН» реализация системы управления сетью стандарта IEEE 802.16 основана на событийно-ориентированной архитектуре. Событийно-ориентированная архитектура [3] является шаблоном архитектуры программного обеспечения, позволяющим создание, определение, потребление и реакцию на события. Модульная событийно-ориентированная архитектура хорошо подходит для реализации программного обеспечения базовой и абонентской станции. Такая архитектура соответствует идеологии стандарта IEEE 802.16. В такой архитектуре реализуется абстракция детерминированных автоматов. В данной реализации системы каждое управляющее MAC-сообщение стандарта передается с помощью определенного события.

Программное обеспечение базовой и абонентской станций декомпозируется на отдельные модули, например, модули управления сервисными потоками, модули поддержки мобильности и т. п. Обмен информацией между модулями осуществляется посредством событий: каждый модуль подписывается на те события, которые он должен обработать. Каждый модуль может генерировать в процессе работы события, которые должны быть обработаны другими модулями. Обработка события осуществляется в отдельном потоке операционной системы.

Реализация протокола IEEE 802.16 на основе событийно-ориентированной архитектуры легко поддается тестированию (модульному, функциональному, нагрузочному, стресс-тестированию) даже при отсутствии реальных радиоустройств. Любой модуль может быть заменен другим, выполняющим те же функции, но обладающим своими специфическими особенностями. Например, драйвер радиомодема может быть заменен на драйвер, ориентированный на работу с виртуальными сетевыми устройствами. Все события, которые могут возникнуть в сети, могут генерироваться в случайной или заданной последовательности внешними генераторами, что позволяет провести нагрузочное, функциональное и другие виды тестирования системы в условиях, приближенных к реальным, без применения радиооборудования.

Архитектура тестирующей системы

Тестируемую систему можно условно разделить на две части: ядро системы и подключаемые модули. Ядро системы организует всю работу системы, отвечает за размещение и функционирование модулей, связь между ними, и предоставляет прикладной интерфейс (API) для управления жизненным циклом модулей. Модули, как правило, создают сторонние разработчики, не имеющие доступа к исходному коду ядра системы.

В качестве тестируемой системы представлена реализация протокола IEEE 802.16 (WiMAX) [1]. Исходный код ядра системы и ряда модулей, недоступен, так как разрабатывается отдельной рабочей группой. Каждый модуль собирается в виде динамической библиотеки. Модули взаимодействуют посредством обмена сообщениями.

Для проведения автоматизированного тестирования на основе спецификаций был применен инструмент CTESK [4]. Интеграция тестируемой системы с инструментом CTESK была выполнена согласно схеме (рис. 1).

На рис. 1 представлены отдельные модули тестируемой и тестирующей системы. Связь между тестируемой и тестирующей системой организована через сокеты. К тестируемой системе добавлено два вспомогательных модуля: сервис-приемник и сервис-отправитель. Их цель– обеспечить ин-

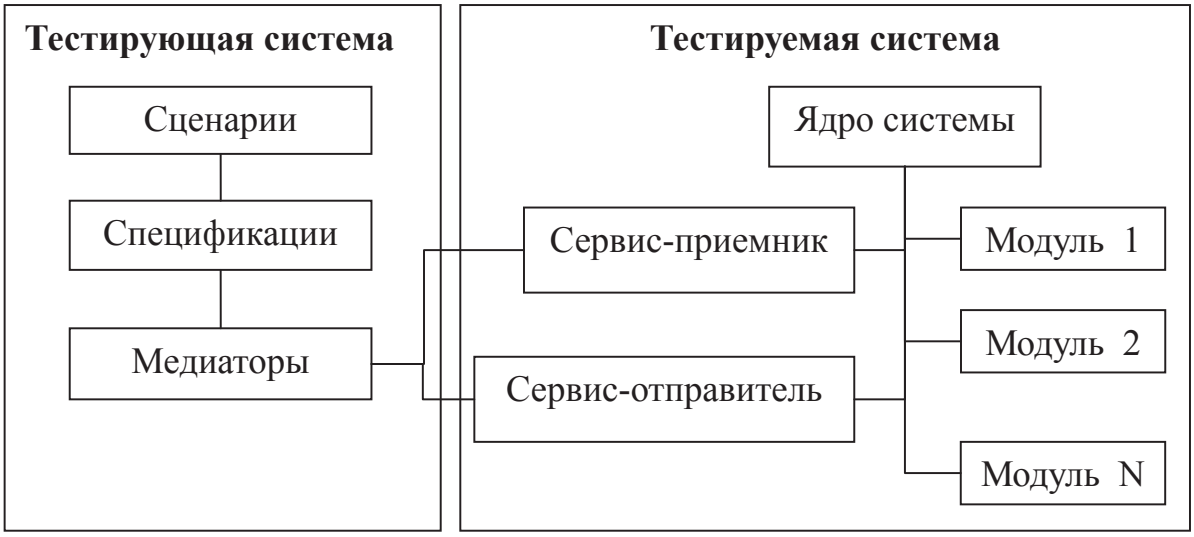


Рис. 1. *Структурная схема тестирования взаимодействия*

терфейс между тестируемой и тестирующей системой. Сервис-приемник прослушивает сетевой порт в ожидании сообщения от тестирующей системы и транслирует это сообщение в термины тестируемой системы. У сервиса-отправителя обратная функция: он переводит сообщение в термины тестирующей системы и записывает его в сокет. Во избежание взаимных блокировок для принятия и отправки сообщений используются разные сокеты. Также, взаимодействие через сокеты может быть полезным на следующих стадиях разработки, когда тестируемая система будет развернута на целевой аппаратной платформе. Сервис-приемник и сервис-отправитель разрабатываются как программные компоненты в течение всего цикла разработки системы.

Предложенный способ коммуникации тестируемой и тестирующей системы может быть применен не только в данной разработке, но и для тестирования других систем, ориентированных на обработку сообщений и имеющих собственный поток управления.

Сложность тестирования представленной системы в том, что она: 1) имеет собственные потоки управления; 2) может отвечать на тестовые воздействия асинхронно. Для тестирования таких систем в CTESK имеется механизм отложенных реакций [5]. Тестирующая система после посылки некото-

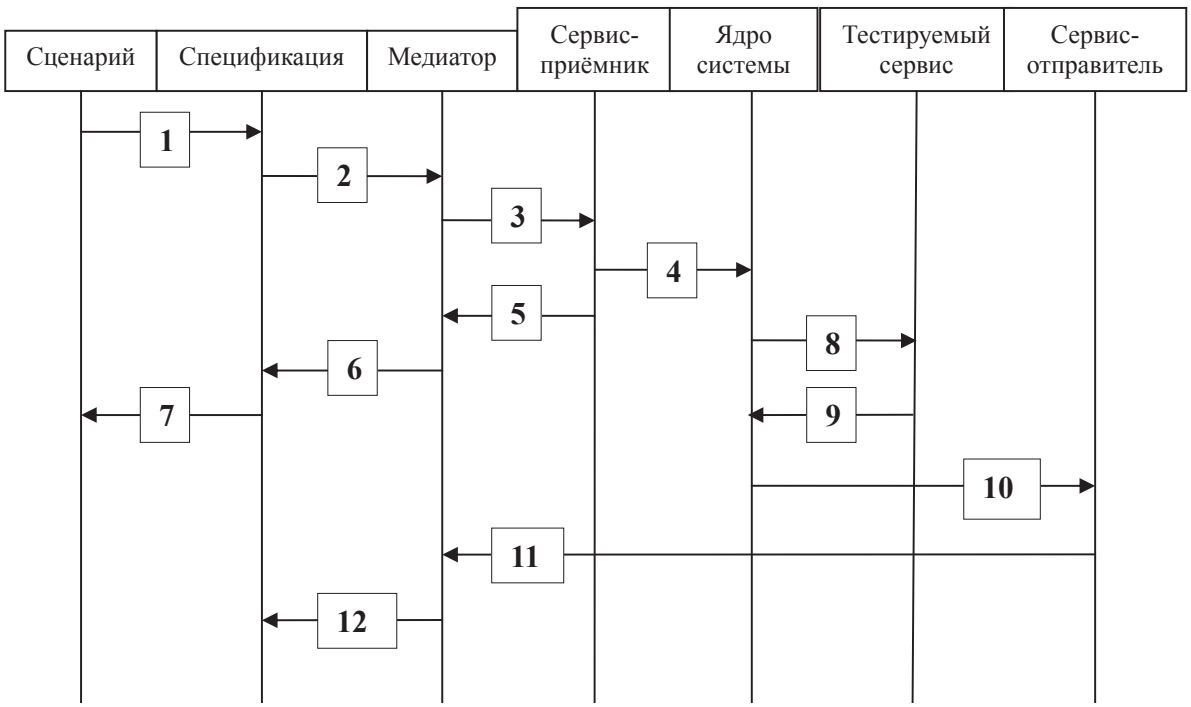


Рис. 2. *Диаграмма последовательности тестирования*

рых тестовых импульсов, ожидает некоторое время возникновения реакций на эти импульсы, затем обрабатывает реакции в соответствующих функциях, определяя, была ли корректна работа тестируемой системы. В этом случае уже не тестирующая, а тестируемая система является инициатором взаимодействия.

Ниже представлена диаграмма последовательности процесса тестирования системы. Опишем более подробно взаимодействия, представленные на диаграмме (рис. 2).

1. Сценарная функция вызывает спецификационную функцию;
2. Спецификационная функция вызывает медиаторную функцию;
3. Медиаторная функция посылает в сокет информацию о тестовом импульсе и необходимые данные;
4. Тестовый сервис-приемник формирует запрос к тестируемому сервису и отправляет его;
5. Тестовый сервис посылает медиатору уведомление, что запрос направлен;
6. Возврат из медиаторной функции;
7. Возврат из спецификационной функции;
8. Посылка сообщения тестируемому сервису;
9. Ответ тестируемого модуля;
10. Возврат результата запроса тестовому сервису отправителю;
11. Тестовый сервис-отправитель посылает результат перехватчику отложенных реакций в медиаторе.
12. Перехватчик реакций вызывает функцию-обработчик отложенных реакций.

Сообщения 1–7 – это посылка импульса в тестируемую систему, инициатором является тестирующая система. Сообщения 10–12 – это реакции тестируемой системы, которые фиксируются и обрабатываются тестирующей системой, инициатор–тестирующая система. Сообщения 8–9 – обработка тестовых импульсов внутри тестируемой системы.

Как видно из описания, спецификации теста являются абстрактными и могут использоваться для тестирования не только данной системы, но и целого класса систем, например различных реализаций сетевого протокола.

Тестирование ранжирования при входе АС в сеть

Рассмотрим тестирование начального ранжирования. В данном случае тестирующая система имитирует работу абонентской станции.

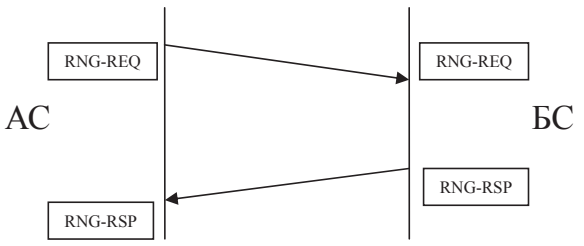


Рис. 3. *Диаграмма взаимодействия станций при ранжировании*

На этапе ранжирования АС посылает базовой станции (БС) сообщение RNG_REQ [1], по специально выделенному для этого каналу. БС, получив запрос, должна сформировать ответ – RNG_RSP. В ответе содержится информация о физических характеристиках (мощность АС, временное смещение в канале), MAC-адрес АС и другие поля, среди которых идентификатор основного соединения и идентификатор первичного соединения [2]. Одна из задач тестируемой системы на данном этапе – создать основной и первичный сервисные потоки и отправить абонентской станции ответное сообщение RNG_RSP.

Тестирование ранжирования АС выполняется в два этапа:

1. Тестирование взаимодействия (соответствие системы протоколу IEEE 802.16), которое проверяет следующие условия:
   1.1. на каждое посланное сообщение RNG-REQ должно приходить ответное сообщение RNG-RSP;
   1.2. идентификаторы основного и первичного соединения должны присутствовать в сообщение RNG-RSP и иметь уникальные значения;
   1.3. значение идентификатора основного соединения должно находиться в интервале 1...N, где N – максимально поддерживаемое число абонентских станций;
   1.4. значение идентификатора первичного соединения должно находиться в интервале N+1...2N, где N – максимально поддерживаемое число абонентских станций;
2. Модульное тестирование с просмотром внутренних состояний тестируемой системы:
   2.1. в хранилище сервисных потоков на базовой станции должны добавиться два новых потока.

Тестируемые требования были оформлены в виде спецификационных функций и отложенных реакций. Тестирующая система, имитируя, абонентскую станцию, посылает тестируемой системе запрос, на который через некоторое время обратно посылается ответ. Ответ перехватывается и обрабатывается в функции обработки отложенной реакции. Корректность этого ответа проверяется тестирующей системой. Ниже представлен код функции обработки отложенной реакции:

```
reaction RNG_RSP* rngRSP_spec (void)
   updates waitingList, activeFlowsCount
{
   post {
      if (! rngRSP_spec) return true;
      bool  testLogic  =  value_Integer
      (rngRSP_spec->basicCID)!= 0 &&
        value_Integer (rngRSP_spec->pri-
        maryCID)!= 0 &&
        value_Integer  (rngRSP_spec->ba-
        sicCID) >= minBasicCID &&
        value_Integer  (rngRSP_spec->ba-
        sicCID) <= maxBasicCID &&
```

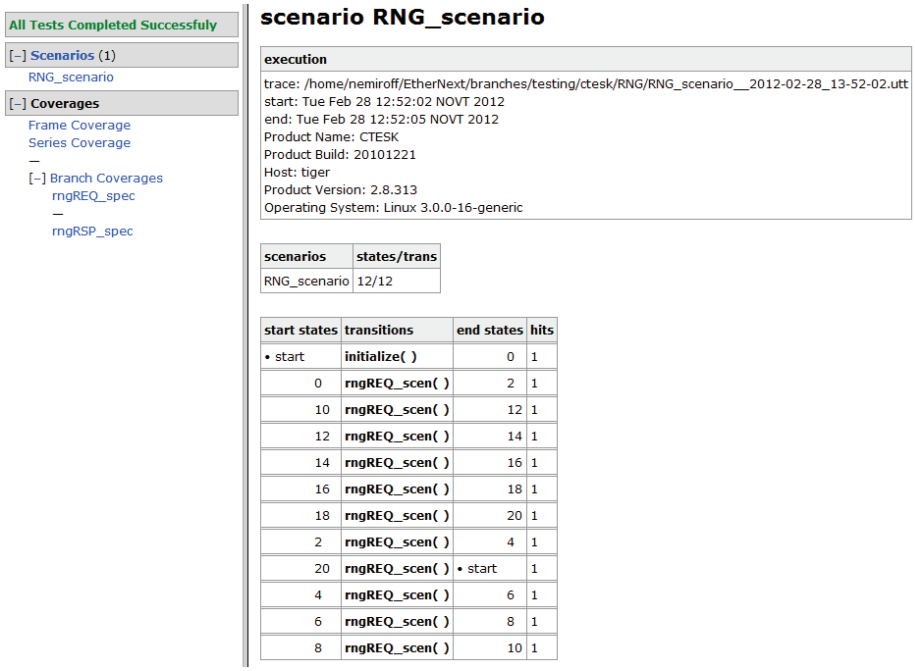


Рис. 4. *Успешное прохождение теста*

```
        value_Integer (rngRSP_spec->pri-
        maryCID) >= minPrimaryCID &&
        value_Integer (rngRSP_spec->pri-
        maryCID) <= maxPrimaryCID &&

        value_Integer  (activeFlowsCount)==
        value_Integer (@activeFlowsCount)+2;
     if (testLogic) {
        printf («Success\n»);
     } else {
        printf («Failure\n»);
     }
        return testLogic;
  }
}
```

В списке *waitingList* содержатся отправленные запросы ранжирования. По мере прихода ответов запросы из списка удаляются. *activeFlowsCount* – количество активных потоков в хранилище. Модификатор updates означает, что эти переменные должны измениться. В обработчике реакции, как и в спецификационной функции, можно получить значение такой переменной как до, так и после тестового воздействия на систему.

При тестировании протокола без доступа к реализации, проверяя только получаемые сообщения, нельзя узнать реальное количество активных потоков. Можно лишь создать модель хранилища на стороне тестирующей системы и изменять ее в зависимости от тестовых взаимодействий. При проходе сценариев прямо или косвенно могут проявиться отклонения, связанные с неправильной работой реального хранилища. Информацией, откуда берется количество активных потоков, владеет слой медиаторов. Реализация на спецификации не влияет.

Результаты выполнения тестов

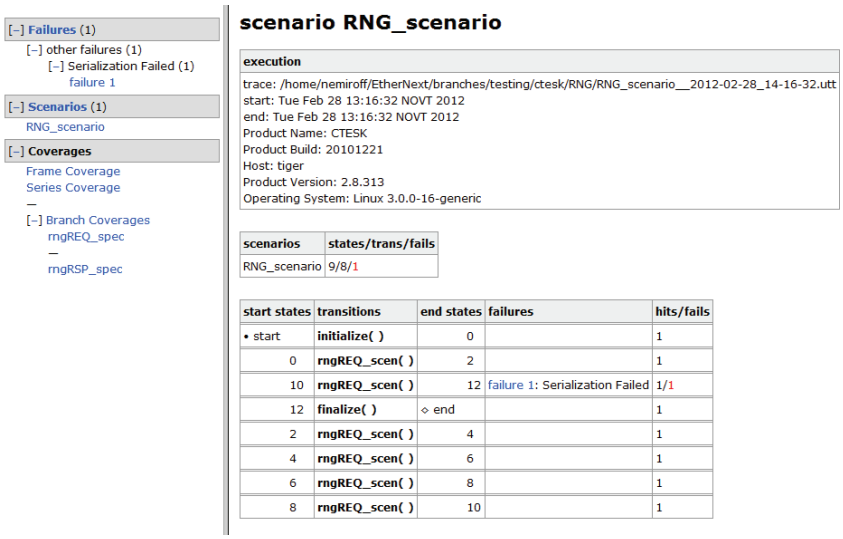
После прохождения тестов тестирующая система формирует трассу их прохождения в формате XML. Трасса содержит все действия тестирующей системы и может быть представлена в формате простого текста или в формате HTML-страницы.

В текстовом формате UniTESK Reports помещает главную статистическую информацию о количестве пройденных тестов в начало файла. В качестве примера ниже представлена основная (статистическая) часть файла результата при выполнении тестов без ошибок.

Для восприятия человеком более удобен HTML-формат, в котором с помощью ссылок можно переходить к просмотру выполнения определенного сценария, спецификационной функции, сообщения об ошибке. Также, HTML-представление отображает информацию в виде удобных для восприятия таблиц, рис. 4.

Чтобы убедиться в работе тестирующей системы в тестируемую были намеренно введены дефекты. Одним из таких дефектов была ошибка генерации значения первичного идентификатора потока. Идентификатор потока мог быть создан за рамками допустимого диапазона. Тестирующая система успешно справилась с задачей и, как только значение вышло за рамки диапазона, она просигнализировала об ошибке, рис. 5.

В случае неуспешного теста разработчика будет интересовать конкретное состояние системы и воздействие, вызвавшее ошибку. Эту информацию можно увидеть в развернутом описании ошибки (рис. 6).



[–] **Failures** (1)
[–] other failures (1)
[–] Serialization Failed (1)
failure 1
[–] **Scenarios** (1)
RNG_scenario
[–] **Coverages**
Frame Coverage
Series Coverage
—
[–] Branch Coverages
rngREQ_spec
—
rngRSP_spec

scenario RNG_scenario

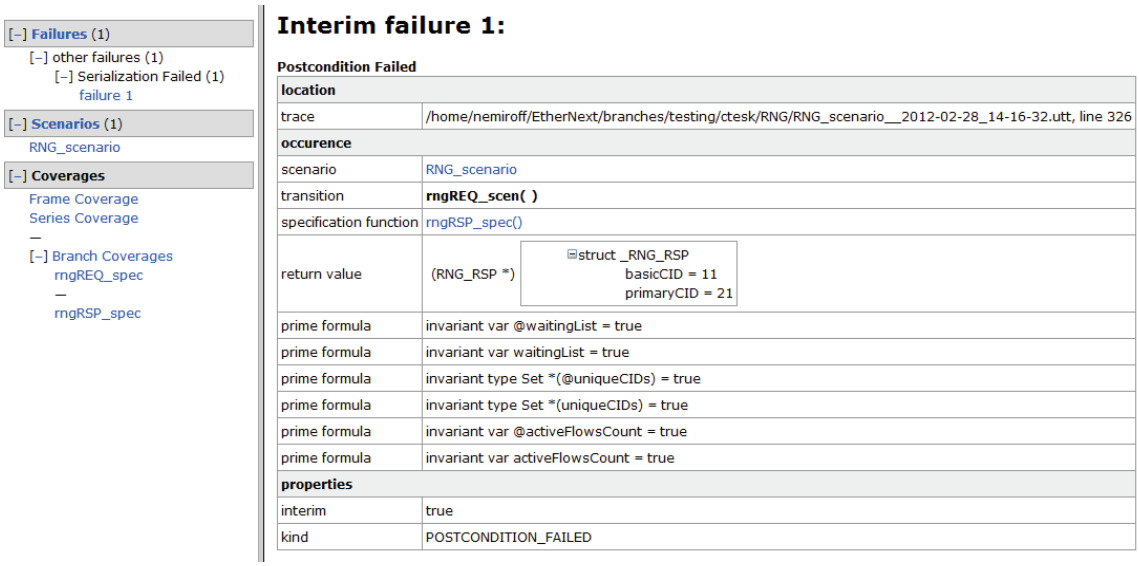
execution

trace: /home/nemiroff/EtherNext/branches/testing/ctesk/RNG/RNG_scenario__2012-02-28_14-16-32.utt
start: Tue Feb 28 13:16:32 NOVT 2012
end: Tue Feb 28 13:16:32 NOVT 2012
Product Name: CTESK
Product Build: 20101221
Host: tiger
Product Version: 2.8.313
Operating System: Linux 3.0.0-16-generic

scenarios	states/trans/fails
RNG_scenario	9/8/1

start states	transitions	end states	failures	hits/fails
• start	**initialize( )**	0		1
0	**rngREQ_scen( )**	2		1
10	**rngREQ_scen( )**	12	failure 1: Serialization Failed	1/1
12	**finalize( )**	◇ end		1
2	**rngREQ_scen( )**	4		1
4	**rngREQ_scen( )**	6		1
6	**rngREQ_scen( )**	8		1
8	**rngREQ_scen( )**	10		1

Рис. 5. *Результаты прохода теста с ошибкой: трасса*



[–] **Failures** (1)
[–] other failures (1)
[–] Serialization Failed (1)
failure 1
[–] **Scenarios** (1)
RNG_scenario
[–] **Coverages**
Frame Coverage
Series Coverage
—
[–] Branch Coverages
rngREQ_spec
—
rngRSP_spec

Interim failure 1:

Postcondition Failed

location	
trace	/home/nemiroff/EtherNext/branches/testing/ctesk/RNG/RNG_scenario__2012-02-28_14-16-32.utt, line 326
occurence	
scenario	RNG_scenario
transition	**rngREQ_scen( )**
specification function	rngRSP_spec()
return value	(RNG_RSP *) ⊟struct _RNG_RSP basicCID = 11 primaryCID = 21
prime formula	invariant var @waitingList = true
prime formula	invariant var waitingList = true
prime formula	invariant type Set *(@uniqueCIDs) = true
prime formula	invariant type Set *(uniqueCIDs) = true
prime formula	invariant var @activeFlowsCount = true
prime formula	invariant var activeFlowsCount = true
properties	
interim	true
kind	POSTCONDITION_FAILED

Рис. 6. *Результаты прохода теста с ошибкой: описание ошибки*

Из файлов результатов можно получить подробности прохождения теста: переходы между состояниями, стек вызовов сценарных функций, значения переменных. Таким образом, можно регулярно проводить регрессионное функциональное тестирование реализации системы управления сетью стандарта IEEE 802.16.

Дальнейшая работа будет связана с наполнением базы тестов, а также с адаптацией технологии UniTESK для тестирования таких механизмов протокола IEEE 802.16 как качество обслуживания – QoS.

Выводы

В работе предложена технология тестирования реализаций системы управления сетью стандарта IEEE 802.16. Оригинальность предложенной технологии заключается в применении технологии UniTESK для тестирования промышленной реализации системы.

В итоге, можно выделить следующие преимущества предлагаемого подхода:

- формальность и абстрактность спецификаций обеспечивает возможность применения набора тестов для тестирования некоторого класса систем, например, различных реализаций одного сетевого протокола;
- автоматическое выполнение тестов позволяет выполнять регрессионное тестирование сложных систем в ходе разработки;
- предложенный способ коммуникаций тестируемой и тестирующей системы позволяет выполнять тестирование как по типу вход-выход, так и с учетом внутренних состояний системы.

Разработка осуществляется в соответствии с госконтрактом № 13.G25.31.0011 от 07 сентября 2010 г.

Данная технология использована в НИР, выполняемой по ФЦП «Научные и научно-педагогические кадры инновационной России» (госконтракт № 14.740.11.0398; шифр заявки 2010-1.1-215-138-022).

СПИСОК ЛИТЕРАТУРЫ

1. IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems // IEEE Std 802.16–2009 (Revision of IEEE Std 802.16–2004). PP. C1–2004, May 29, 2009. doi: 10.1109/IEEESTD.2009.5062485). URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5062485&isnumber=5062484, (дата обращения: 04.09.2012).
2. Вишневский В.М., Портной С.Л., Шахнович И.В. Энциклопедия WiMAX. Путь к 4G. – М.: Техносфера, 2009. – 472 с.
3. Taylor H., Yochem A., Phillips L., Martinez F. Event-Driven Architecture: How SOA Enables the Real-Time Enterprise. – Boston: Addison-Wesley, 2009. – 308 p.
4. UniTESK. URL: http://unitesk.ru/ (дата обращения: 04.09.2012).
5. Грошев С.Г. Применение технологии UniTesK для тестирования систем с различной конфигурацией активных потоков управления. // citforum.ru. 2012. URL: http://citforum.ru/SE/testing/unitest_use/ (дата обращения: 04.09.2012).
6. Nuaymi L. WiMAX: Technology for Broadband Wireless Access. – New York: John Wiley & Sons, 2007. – 310 p.

Поступила 14.09.2012 г.

УДК 004.021

ЭКСПЕРИМЕНТ ПО ФРАКТАЛЬНОМУ СЖАТИЮ RGB-ИЗОБРАЖЕНИЙ НА ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ

И.В. Бойченко, С.С. Кулбаев, А.А. Немеров, В.В. Голенков

Томский государственный университет систем управления и радиоэлектроники
E-mail: kulbaev@asu.tusur.ru

Описан эксперимент по сжатию полноцветных изображений на основе фракталов с применением высокопроизводительной вычислительной системы с распределенной памятью – вычислительном кластере. Межпроцессный обмен осуществляется на основе технологии MPI. Показана линейная зависимость времени вычислений от количества вычислительных процессов. Выявлена неравномерность нагрузки вычислительных процессов, вызванная неоднородностью сжимаемых изображений. Проведено сравнение качества и размера сжимаемых изображений на основе фракталов и на основе алгоритма JPEG.

Ключевые слова:
Параллельные вычисления, интерфейс передачи сообщений, фракталы, сжатие изображений, вычислительные процессы, высокопроизводительные вычислительные системы.

Key words:
Parallel computation, message passing interface, fractals, image compression, computational processes, high-performance computing systems.

Повышенный интерес к алгоритмам сжатия изображений на основе фракталов вызван необходимостью минимизации размеров передаваемых или хранимых данных. Для сжатия статических изображений существует множество методов [1–3]. Наиболее известными из методов сжатия графической информации с потерей качества являются алгоритмы JPEG и JPEG2000. Поэтому, при разработке новых алгоритмов, проводят сравнение именно с JPEG. При рассмотрении алгоритмов сжатия наиболее важными являются три основных свойства: коэффициент компрессии, степень потери качества по сравнению с оригиналом, трудоемкость. Однако в рамках различных задач ценность этих параметров не равнозначна. Для случая дефицита дискового пространства и полосы пропускания каналов, при достаточном количестве вычислительных мощностей, представляет интерес использование фрактального сжатия [2], обладающего потенциально более высоким коэффициентом компрессии, но более трудоемким при сжатии. Обратный процесс – распаковка, требует меньше вычислений, чем у JPEG [3]. Свойство, заключающееся в независимости восстанавливаемого изображения от разрешения, позволяет использовать фрактальный алгоритм для визуализации цифровых изображений на больших экранах.

Как правило, в работах, посвященных исследованию алгоритмов фрактального сжатия, накладывается такое ограничение, что алгоритм должен обеспечивать приемлемое время сжатия на персональном компьютере [4, 5]. Поскольку полный перебор при поиске соответствия доменных и ранговых блоков приводит к большому количеству операций, то усилия разработчиков, в основном, направлены на сокращение количества сравнений блоков за счет предварительной классификации [6]. Сокращение числа рассматриваемых блоков неизбежно приводит к большим потерям качества по сравнению с полным перебором. В данном исследовании ограничение на производительность вычислительной системы было ослаблено исходя из того, что высокопроизводительные системы разных классов становятся все более доступными для конечных пользователей. Так, сервисы типа GRID и «облачные вычисления» позволяют задействовать удаленные вычислительные мощности в режиме on-line. Другим трендом развития совре-