ИЗВЕСТИЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ
№ 11/2 ФИЗИКА 2015

УДК 519.713

Н.М. ЩИПАЧЕВ, Н.Г. КУШИК**

ТЕСТИРОВАНИЕ В КОНТЕКСТЕ ДЛЯ РЕАЛИЗАЦИИ ПРОТОКОЛА TCP НА ОСНОВЕ АВТОМАТНЫХ МОДЕЛЕЙ

В работе рассматривается задача тестирования в контексте для реализации протокола TCP. Тестирование проводится на основе формальных моделей, а именно, в качестве формальной модели исследуется параллельная композиция конечных автоматов, на основе которой осуществляется тестирование по целям. Эксперименты с реализацией протокола TCP, встроенной в программное обеспечение Psi, подтверждают эффективность рассматриваемого подхода.

Ключевые слова: *протокол TCP, тестирование в контексте, конечный автомат.*

В современном мире программное обеспечение (ПО) имеет все большее значение для человека и его жизнедеятельности, однако с ростом развития ПО возрастает и его сложность и, как следствие, количество ошибок в нем. Существует множество сфер деятельности с критическими требованиями к надежности программного обеспечения, к последним, например, относятся медицина и воздушный флот. Время на реализацию всегда ограничено, поэтому с учетом роста количества ошибок возрастает необходимость в эффективном и качественном тестировании программных компонентов технических систем. Соответственно, возрастает необходимость и в инструментах, реализующих новые методы синтеза тестов для программного обеспечения.

При тестировании реальных протокольных реализаций неминуемым образом приходится сталкиваться с тем, что большинство таких реализаций являются внутренней частью более сложного ПО. Этот факт зачастую исключает возможность прямой подачи входных последовательностей на тестируемую реализацию, и возникает необходимость говорить о тестировании в контексте, предполагая, что тестируемая реализация встроена в некоторую внешнюю среду (environment).

Классическим примером «недоступности» проверяемой реализации выступает протокол TCP (Transmission Control Protocol), который используется многочисленными программами без прямого доступа к нему со стороны пользователя. Причина такой частичной управляемости реализаций протокола TCP, в первую очередь, обуславливается назначением протокола, а именно, протокол TCP используется для обеспечения надежности устанавливаемого соединения. Отметим, что этот факт также обосновывает необходимость качественного тестирования реализаций данного протокола.

Как известно, гарантированная полнота тестирования достигается при использовании формальных моделей [1, 2], частным случаем которых являются автоматные модели. Большое количество работ посвящено методам синтеза тестов на основе конечных автоматов и их различных модификаций, в том числе, расширенных автоматов, временных автоматов, их композиций и др. Последние рассматриваются в данной статье применительно к тестированию реализации протокола TCP. Следует отметить, что мы рассматриваем тестирование клиентской части реализации TCP и, в частности, рассматриваем так называемое тестирование по целям (test purposes). Для этого мы синтезируем тест для клиентской части протокола TCP, спецификация которой задана в виде автоматной модели [3]. Далее этот тест транслируется через среду, которая ограничивает доступ к соответствующей реализации [4].

Для моделирования поведения тестируемой системы и ограничения доступа к ней мы рассматриваем параллельную композицию взаимодействующих компонент. В данной работе компоненты этой композиции представляются конечными автоматами, и принимается во внимание условие «медленной» среды, в каждый момент времени активна только одна компонента, а при поступлении входного символа, компоненты работают в режиме диалога. Как известно, такая топология приемлема для описания случаев, когда работа одних компонент моделируемого ПО зависит от реакции других, и во время работы вторых первые неактивны. Соответствующая топология представлена на рисунке 1.

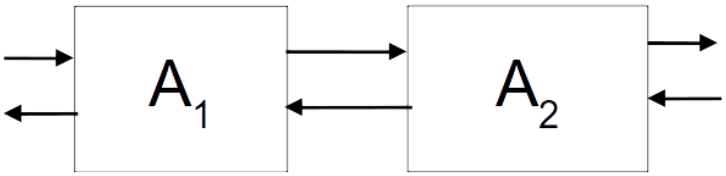

Рисунок 1. Параллельная композиция конечных автоматов A_1 и A_2

Для тестирования реализации протокола TCP с веб сайта Сент-Эндрюсского университета была взята автоматная модель, описывающая данный протокол [5]. Для синтеза тестов для реализации протокола TCP из приведенной автоматной модели был выделен конечный автомат (рисунок 2). Отметим, что в данной модели не учтены переходы по таймауту, что означает мгновенное закрытие соединения и бесконечное ожидание в состояниях *SYN_SENT* и *SYN_RCVD*, однако данный факт не влияет на результаты экспериментов, поскольку рассматриваемые в данной работе цели тестирования не включают проверку переходов по таймауту.

В качестве тестируемой реализации данного протокола была выбрана реализация, включенная в ПО Psi, являющееся реализацией протокола XMPP (Extensible Messaging and Presence Protocol) [6], который, в свою очередь, работает «поверх» проверяемого TCP.

Таким образом, при проведении экспериментов рассматривалась архитектура систем, изображенная на рисунке 3. Здесь и далее мы используем следующие обозначения: *I* — входной алфавит среды, *O* — ее выходной алфавит, *U* — алфавит внутренних входных символов тестируемой реализации, *V* — алфавит внутренних выходных символов тестируемой реализации, *X* и *Y* — входной и выходной алфавиты копии тестируемой реализации. Множества *X, Y, U, V* получены на основе анализа автомата-спецификации. Так множествам *X* и *Y* присвоены символы, помеченные на рисунке 2 флагами *Recv, Send*, а множеству *U* присвоены символы, помеченные флагами *Appl*. Алфавиты *I* и *O* сформированы в результате анализа взаимодействия контекста и тестируемой реализацией TCP. Симметричность протокола TCP, как отмечалось выше, позволяет рассматривать серверную реализацию как копию реализации клиентской части. Таким образом, множества алфавитов определяется следующим образом: *I* = {*Инициализация, завершение*}, *O* = {*Успех_инициализации, ошибка, успех_завершения*}, *X* = {*SYN, FIN, RST, ACK, SYN_ACK, FIN_ACK*}, *Y* = {*SYN, FIN, RST, ACK, SYN_ACK, FIN_ACK*}, *U* = {*passive_open, active_open, send_data, close*}, *V* = {<*nothing*>}.

Для проверки корректности функционирования реализации протокола TCP в контексте ПО Psi мы формируем цели тестирования, на основе которых синтезируем тестовые последовательности. При проведении компьютерных экспериментов в качестве цели тестирования мы рассматривали проверку достижимости каждого из состояний системы. Был синтезирован следующий тест: *TS* = {*passive_open.SYN.close.ACK, passive_open.send_data.SYN, active_open.SYN_ACK.close.FIN, active_open.SYN_ACK.FIN.close*}. Рассмотрим в качестве примера последовательность *active_open.SYN_ACK.close.FIN*. Она состоит из 4 символов и отвечает за проверку достижимости состояния *CLOSING*.

Построенный тест был подан на реализацию протокола TCP через контекст Psi. Отметим, что при проведении экспериментов использовалось ПО Psi версии 0.15-2; пакеты перехватывались с помощью ПО Wireshark версии 1.12.1; эксперимент проводился в системе Debian (Linux 3.16.0 x64).

В результате экспериментов было обнаружено несоответствие тестируемой реализации TCP ее автоматной спецификации. В частности, при проверке достижимости состояния *FIN_WAIT_1*, при условии, что TCP сервер (копия тестируемой реализации) в начальный момент времени находится в состоянии *LISTEN*, а тестируемая реализация TCP клиента – в состоянии *CLOSED*. Тестовая последовательность, направленная на проверку данной цели, имеет вид: *active open. SYN_ACK. close* (канал *U*). Для того чтобы данная последовательность была произведена средой Psi, на вход Psi подается последовательность *инициализация.завершение* (канал *I*). Отметим, что ожидаемая выходная реакция на данную входную последовательность имеет вид *Успех_инициализации.успех_завершения* (канал *O*).

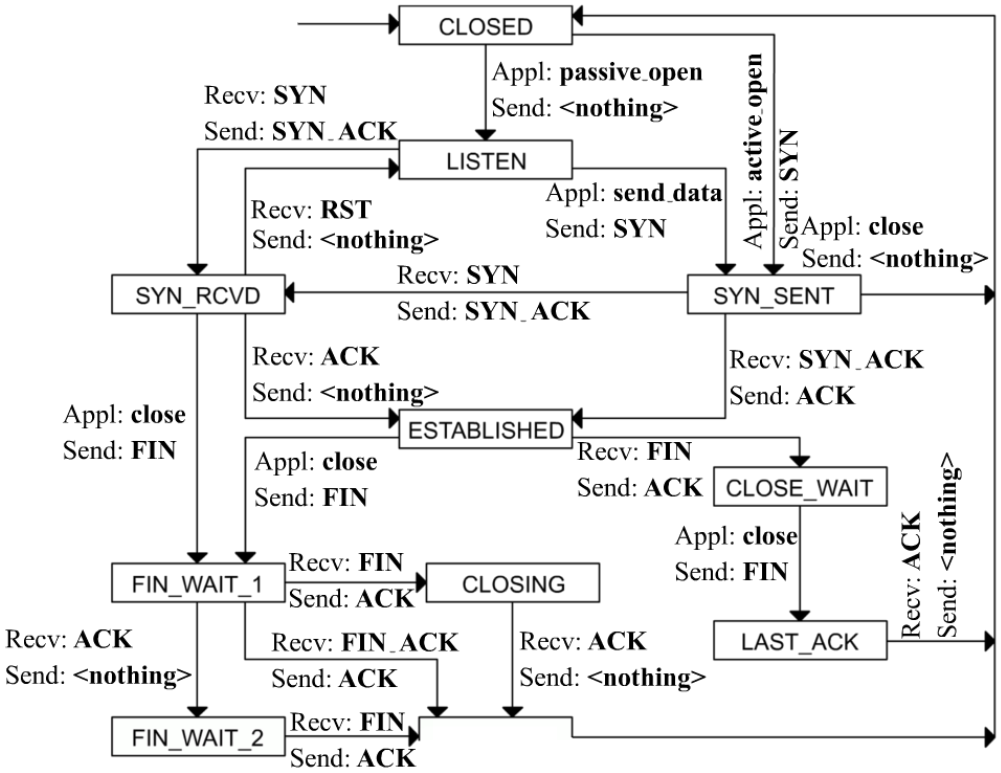

Рисунок 2. Конечный автомат, описывающий TCP

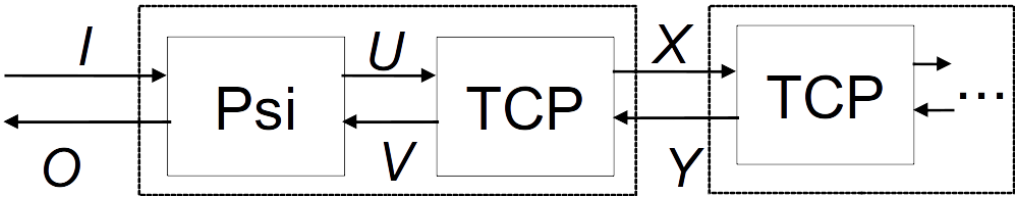

Рисунок 3. Общая схема тестирования реализации TCP в контексте ПО Psi

После подачи на ПО Psi тестовой последовательности *инициализация.завершение* от среды была получена ожидаемая реакция, однако при анализе реакций реализации на выходе *X* был обнаружен «неожиданный» выходной символ. Анализ символов, передаваемых по каналам, помеченным *X* и *Y*, во время эксперимента и наблюдение данного «неожиданного» выходного символа отмечены на рисунке 4. В соответствии со спецификацией тестируемая реализация после подачи на нее последовательности *active_open.SYN_ACK.close* должна перейти из состояния *CLOSED* в состояние *SYN_SENT*, затем в *ESTABLISHED* и далее в состояние *FIN_WAIT_1*. Анализ передаваемых по сети пакетов (рисунок 4) позволяет заключить о переходе реализации из состояния *CLOSED* в *SYN_SENT* и далее в состояние *ESTABLISHED*. Однако при выполнении последнего перехода тестируемой реализацией был передан выходной символ *FIN_ACK* (канал *X*). Несмотря на то, что такой выходной символ присутствует в алфавите, в данном состоянии не определено перехода с выдачей данного выходного символа (рисунок 2). Этот факт позволяет заключить, что реализация TCP, входящая в ПО Psi версии 0.15-2, не соответствует автоматной модели, предложенной на сайте Сент-Эндрюсского университета [5].

В дальнейшем мы планируем рассмотреть другие цели тестирования для реализаций протокола TCP, а также другие методы синтеза и трансляции тестов.

X,*Y*:

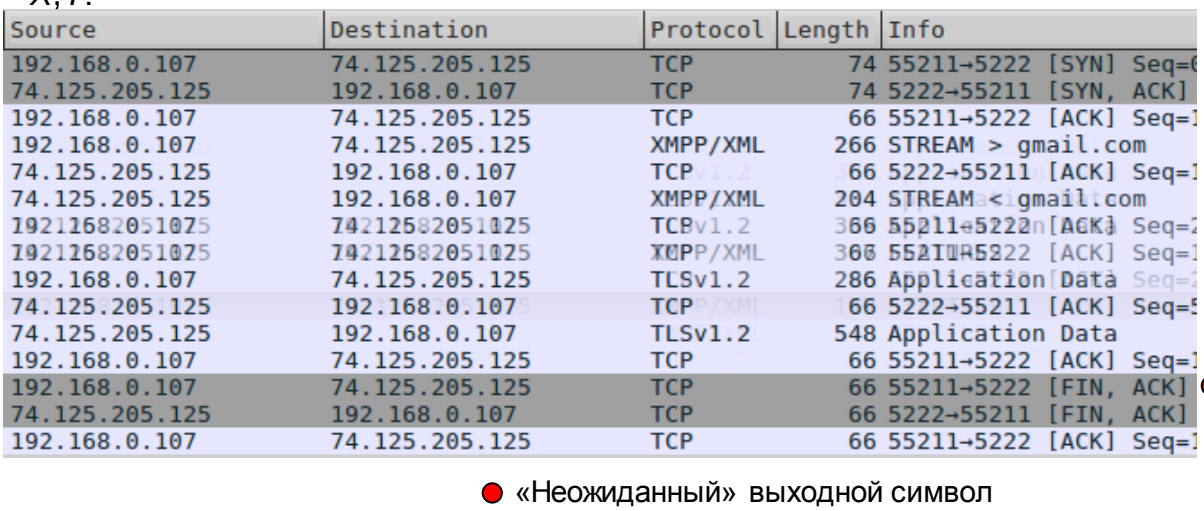

Рисунок 4. Снимок окна ПО Wireshark при проведении экспериментов

СПИСОК ЛИТЕРАТУРЫ

Kushik N., Yevtushenko N. Adaptive Homing is in P // Proceedings Tenth Workshop on Model Based Testing (MBT 2015) // Electronic Proceedings in Theoretical Computer Science 180. – London, 2015 – С. 73-78

Budkowski S., Cavalli A., Najm E. Formal Description Techniques and Protocol Specification, Testing and Verification, FORTE XI / PSTV XVIII'98, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII) Boston, MA.: Springer US, 1998. С. 470

TCP Finite State Machine from School of Computer Science University of St Andrews [электронный ресурс] URL: http://tcp.cs.st-andrews.ac.uk/index.shtml?page=tcp_fsm

El-Fakih K., Petrenko A., Yevtushenko N. FSM Test Translation Through Context // 18th IFIP International Conference on Testing of Communicating Systems (TestCom 2006): LNCS 3964 – New York City: IFIP, 2006. – С. 254-258.

RFC 793 – Transmission Control Protocol. DARPA Internet Program Protocol Specification [электронный ресурс] URL: https://tools.ietf.org/html/rfc793

RFC 6120 – Extensible Messaging and Presence Protocol (XMPP): Core. Internet Engineering Task Force (IETF) [электронный ресурс] URL: https://tools.ietf.org/html/rfc6120

*Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: shchipachevnm@gmail.com, ngkushik@gmail.com

Щипачев Никита Михайлович, студент ТГУ
Кушик Наталья Геннадьевна, доцент ТГУ

N.M. SHCHIPACHEV, N.G. KUSHIK

FSM BASED TESTING IN CONTEXT FOR A TCP IMPLEMENTATION

In this paper, we address to a problem of testing in context for protocol implementations. Tests are derived based on state models and then they are translated via a corresponding environment. A parallel composition of two finite state machines is derived for testing a TCP implementation in the context of a software Psi. Preliminary experimental results show the efficiency of a proposed test derivation technique.

Keywords: *communication protocol, TCP, testing in the context, FSM.*