АНАЛИЗ КОНФОРМНОСТИ НА ЭТАПЕ ПРОЕКТИРОВАНИЯ СЕТЕВЫХ ПРОТОКОЛОВ

УДК 623.618

ИЗОТОВ Антон Сергеевич
аспирант ХНУРЭ.
Научные интересы: анализ сетевых технологий, диагностирование сетевых протоколов.

НЕМЧЕНКО Владимир Петрович
канд. техн. наук, профессор ХНУРЭ.
Научные интересы: сетевые технологии нового поколения,
защита сетевой информации, диагностирование сетевых протоколов.

ВВЕДЕНИЕ

Интенсивная разработка сетевых протоколов нового поколения в последнее время приводит к острой необходимости тестирования новых (разрабатываемых) протоколов как на соответствие техническим спецификациям, так и на возможность их корректного взаимодействия между собой и с ранее разработанными и использующимися в настоящее время протоколами. Решение данных вопросов приобретает особую актуальность в связи с переходом на новое поколение сетевых протоколов стека TCP/IP версии 6 [1,2].

Как отмечается в литературе [http://testingworld.ru/category/], «ПО считается пригодным к выпуску, если в нём устранены все критические ошибки и устранено 85% некритических ошибок». При этом аналитики отмечают, что распределение затрат по стадиям жизненного цикла ПО примерно таково: на анализ требований, создание спецификации, проектирование и кодирование приходится около 18% затрат. Остальные 82% затрат приходятся на тестирование, промышленное производство и сопровождение.

ПОСТАНОВКА ЗАДАЧИ

Актуальной задачей на этапе создания программного продукта, в том числе и сетевых протоколов, является задача тестирования протоколов перед внедрением их в «производство» поскольку тестирование на этапе производства и сопровождения ПО требует значительно больших затрат по сравнению с затратами на тестирование на этапе проектирования и разработки протоколов.

ЖИЗНЕННЫЙ ЦИКЛ СЕТЕВОГО ПРОТОКОЛА

Как известно, «жизненный цикл» [3] сетевого протокола содержит следующие этапы.

1) Этап *эскизного проектирования*, на котором предлагается структура будущего протокола (или его модификации) и который является одним из начальных этапов создания протокола.

2) *Спецификация протокола*, которая является по сути его стандартом. В настоящее время в качестве спецификаций используется информация, представленная в документах RFC (англ. - Request For Comments). Например, на странице http://www.protocols.ru/files/RFC/rfc793.pdf мы находим спецификацию протокола TCP – Transmission Control Protocol.

3) *Программная реализация* протокола.

4) Этап *тестовой эксплуатации* протокола позволяет пользователям ознакомиться с новым протоколом на практике и сделать заключение о его работоспособности и взаимодействии с другими протоколами стека протоколов.

5) Этап *внедрения в эксплуатацию* разработанного протокола предполагает начало его широкого

использования на практике путем включения как составного компонента в распространенные сетевые операционные системы.

На рисунке 1 приведена упрощенная структура жизненного цикла протокола. В ней для краткости опущены такие этапы, как предложение черновых вариантов протокола и выбор лучшего из них, а также этап старения протокола и выведения его из эксплуатации.

Рисунок 1 – Упрощенная структура жизненного цикла сетевого протокола

С целью повышения эффективности проектирования сетевых протоколов, а также надежности и бесперебойности их работы предлагается дополнить рассмотренную структуру жизненного цикла протокола дополнительными этапами, которые позволяют значительно упростить процесс создания сетевых протоколов и повысить их эффективность и надежность функционирования путем анализа конформности сетевых протоколов задекларированной спецификации. Для этого сделаем некоторые изменения в структуре рассмотренного жизненного цикла. Модифицированная структура жизненного цикла сетевого протокола показана на рисунке 2.

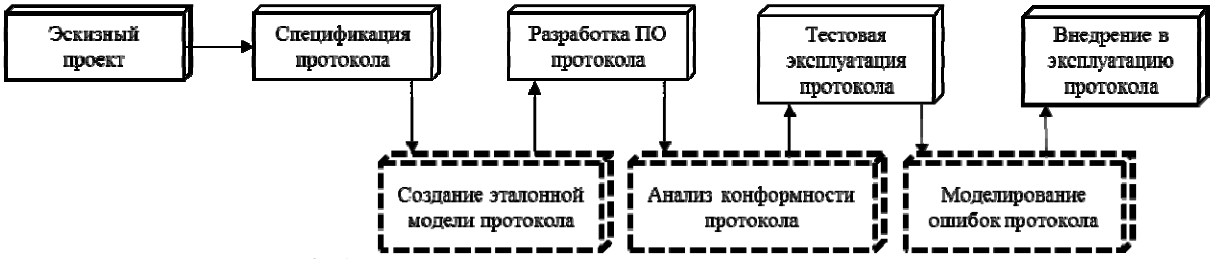


Рисунок 2 – Модифицированная структура жизненного цикла сетевого протокола

Как видим, первые этапы цикла («Эскизный проект» и «Спецификация протокола») остаются без изменения. Следующим этапом после создания спецификации протокола предлагается осуществлять моделирование протокола на основании его спецификации. В работе были проанализированы различные возможности моделирования. В том числе метод сетей Петри, а также метод автоматного моделирования на основе теории конечного автомата Мили. Последний метод был взят за основу дальнейшего исследования как наиболее адекватный и перспективный. В результате была получена на практике автоматная модель одного из наиболее используемых сетевых протоколов транспортного уровня - TCP (*англ.*- Transmission Control Protocol). В качестве апробации полученной модели она была реализована на языке VHDL, который является базовым языком при разработке аппаратуры современных вычислительных систем. Полученные результаты, в том числе и временные диаграммы работы протокола, свидетельствуют об адекватности полученной модели и о целесообразности ее применения при разработке и создании се5тевых протоколов.

Этап «Анализ конформности прокола» позволяет сделать вывод о соответствии некоторой имплементации протокола его задекларированной спецификации. Выводы о конформности протокола можно сделать сравнивая реакции заданной реализации протокола с реакцией эталонной модели на одинаковые входные воздействия (команды).

Известно следующее определение конформности: «Conformance (конформность) – это свойство документа, означающее полное соответствие его разметки как синтаксическим, так и семантическим требованиям заявленного стандарта» [4]. В контексте диагностирования сетевых протоколов нужно уточнить, что конформность протокола определяет приложение, которое объявляется конформным одному или большему числу профилей документа OSI, должно использовать только

средства, описанные в данных профилях, а также в базовых стандартах, на которые имеются ссылки в этих профилях. Таким образом, для проверки конформности разрабатываемого протокола задекларированной спецификации достаточно сравнить выходные реакции протокола в ответ на входные воздействия с эталонными реакциями модели спецификации на те же входные воздействия.

МОДЕЛИРОВАНИЕ ОШИБОК ПРОТОКОЛА

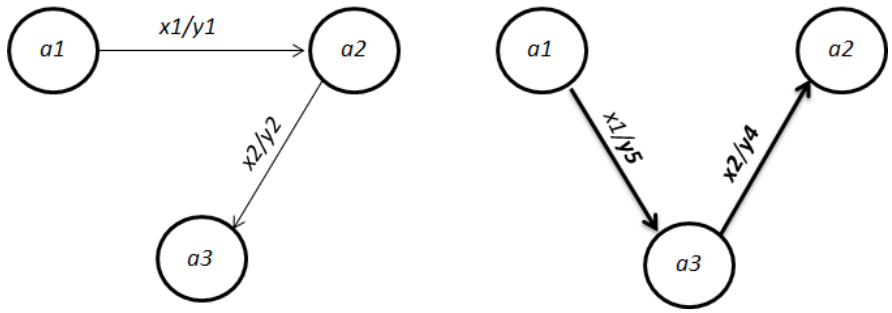
Этап «Моделирование ошибок протокола» предшествует завершающему этапу жизненного цикла, связанному с внедрением в эксплуатацию разработанного протокола. Отметим, что протокол может быть представлен либо в виде графа потока управляющих команд контроллера или в виде графа потока данных между входными параметрами и выходными переменными контекста. Как отмечалось выше, предлагается использовать автоматную модель Мили [5]. При таком задании алгоритма функционирования протокола можно выделить следующие классы ошибок.

1) *Ошибки выхода* автомата имеет место в случае, когда полученный выходной сигнал не совпадает с ожидаемым. При этом переход автомата из одного состояния в другое является правильным.

2) *Ошибки перехода* имеют место в случае, когда автомат под воздействием правильного (заданного) входного слова переходит в некоторое не предусмотренное алгоритмом состояние.

3) Под *ошибкой состояния* будем понимать случай, когда множество состояний автомата A={a0, a1,…, an} не совпадает с множеством устойчивых состояний, заданным спецификацией протокола.

Характерно, что на практике в большинстве случаев в автоматах могут иметь место различные сочетания ошибок, как это показано на рисунке 3. В данном примере в результате непредвиденных сбоев в функционировании автомата произошло следующее: вместо запланированных переходов появились два новых. Из состояния a1 под воздействием входного слова x1 автомат перешел в состояние a3 вместо состояния a2. При этом было выработано выходное слово y5, а затем из состояния a3 под воздействием входного слова x2 автомат перешел в состояние a2 и выработал при этом незапланированное выходное слово y4. Поскольку состояние a2 является одним из устойчивых состояний автомата, то дальнейшее функционирование автомата может пойти по ложному пути.



Без ошибки С комбинированной ошибкой

Рисунок 3 – Пример автомата с комбинированной ошибкой

Другими словами, на разработанной модели сетевого протокола проверяется его реакция на некоторое заданное множество ошибок в работе протокола. Таким образом использование эталонной модели позволяет либо заранее составить таблицу реакций протокола на наперед заданное или сгенерированное некоторым (возможно случайным) образом множество ошибок с тем, чтобы проверить реализацию протокола на наличие или отсутствие этих ошибок.

ТЕСТИРОВАНИЕ СЕТЕВЫХ ПРОТОКОЛОВ

Предлагается приведенная на рисунке 4 структура системы. В состав данной системы входят следующие блоки:

- TG (Test Generator) – генератор тестовых наборов;

- TC (Test Converter) – преобразователь тестовых наборов, приводящий их к единому стандартизированному виду;
- CORE (Main functional block) – основной функциональный блок, ядро системы;
- RA (Results Analyzer) – анализатор результатов тестирования;
- RVR (Results View Rendering) – блок визуализации результатов.

В качестве входных интерфейсов можно назвать:

- tech-docs, dev-notes – техническая документация, заметки разработчиков;
- protocol's model – логическая модель протокола;
- protocol's specification – спецификация протокола.

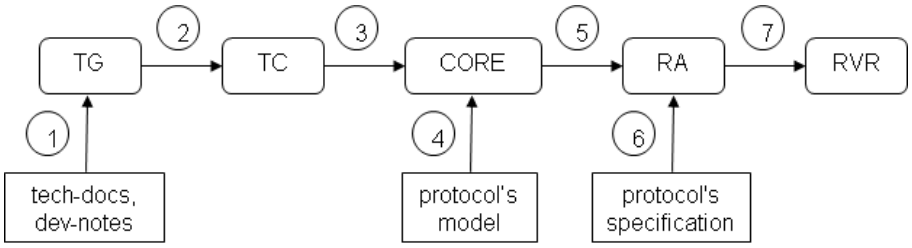


Рисунок 4 - Общая структура системы тестирования

Кроме перечисленных блоков система содержит следующие типы потоков данных.

1) Обработчик черновых материалов на базе UniTESK;
2) Таблица тестовых входных воздействий;
3) Нормализированные наборы тестов;
4) Графическая модель протокола;
5) Результаты тестирования в виде таблицы значений;
6) Заявленные нагрузки и характеристики в виде таблицы значений;
7) Единая таблица сравнений результатов с ожидаемыми значениями.

Отметим, что данная структура представляет универсальный механизм, позволяющий провести проверку правильности функционирования разрабатываемого теста на конформность задекларированной спецификации как на этапе разработки, так и после ее окончания основываясь на утвержденной спецификации.

Первым звеном этого механизма является блок генерации тестов (TG), входные значения которого получены основываясь на заметках разработчиков на этапе производства и на утвержденной спецификации после выпуска последней стабильной версии протокола. Выходные данные этого блока необходимо представить в виде структурированной таблицы критических значений с параметром нагрузки.

Вторым блоком модели есть анализатор выходных значений генератора (TC), который должен интерпретировать их в единый формат для покрытия всех логических элементов протокола.

Исполнительным звеном системы тестирования (ядром системы) является третий блок (CORE). Это блок с двумя потоками входных данных, приведенных к единому формату, то есть тестовые наборы и модель протокола. Данные, которые получены после анализа спецификации покрывают модель протокола и результате мы получаем набор выходных значений тестирования классифицированных по заявленным нагрузкам.

Анализ полученных результатов тестирования осуществляет конечный блок (RA). Результаты, полученные при испытании, сравниваются с заявленными техническими характеристиками и определяются возможные дефекты, которые могут возникнуть при реальных нагрузках после введения протокола в массовую эксплуатацию.

Так же планируется использовать дополнительный блок для отображения результатов тестирования в виде графиков и диаграмм (RVR), которые необходимо наложить на заявленные в спецификации характеристики.

На этапе концептуального проектирования системы тестирования сетевых протоколов необходимо решить ряд проблемных вопросов, первым из которых является — в каком виде представлять входные данные первого блока (заметки разработчиков и черновые варианты документаций)? Как уже отмечалось выше, стандартным форматом нормативной документации сетевых протоколов являются документы RFC. Требования в этих документах изложены на английском языке и представляют собой неформальный текст, описывающий задекларированное поведение системы. В рамках технологии UniTESK [6] для формальной записи требований используются спецификационные расширения языков программирования – Java или C.

ВЫВОДЫ

Предложенный в работе подход к анализу конформности сетевых протоколов на этапе их проектирования был апробирован и показал свою работоспособность и эффективность на практике. Для примера был взят сетевой протокол транспортного уровня TCP версии 4. Для заданной спецификации построен граф переходов протокола. При создании модели на базовом языке разработки вычислительных систем VHDL авторы столкнулись с необходимостью разработки нового подхода, заключающегося в метамоделировании процесса описания сетевого протокола [7]. Использование разработанной метамодели позволило более адекватно представить моделируемый протокол на языке входных терминов VHDL. Проведенное моделирование некоторого множества ошибок функционирования протокола позволило сформировать первичный «словарь ошибок» сетевого протокола TCP, который будет использован при тестировании реализаций этого протокола.

Анализ конформности на этапе проектирования сетевых протоколов приобретает особую актуальность в связи с происходящим в настоящее время переходом интернет-сообщества к использованию стека сетевых протоколов TCP/IP нового поколения версии 6 поскольку целый ряд протоколов этого стека находятся сейчас в стадии разработки и апробирования.

ЛИТЕРАТУРА:

1. Schaff A., Nemchenko V. Test of the new generation internet protocols IPv6. / Radioélectronika i informatika. - Kharkiv, Ukraine. 2001. No.1, P. 87-89.
2. Fejt Sidni M. TCP/IP. Arhitektura, protokoly, realizacija (vkljuchaja IP versii 6 i IP Security) – McGraw-Hill. – 2000, ISBN: 5-85582-072-6.
3. Dzharratano D., Rajli G. Jekspertnye sistemy: principy razrabotki i programmirovanie»: Per. s angl. — M.: Izdatel'skij dom «Vil'jams», 2006. — 1152 s.
4. Open Systems Interconnection – Conformance testing. SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS. ITU-T Recommendation X.292. - Geneva, 2003. - 236 p.
5. Kagan B.M. Jelektronnye vychislitel'nye mashiny i sistemy. M.: Jenergoatomizdat. 1991. – 526 s.
6. Ivannikov P., Kamkin A.S., Kuljamin V.V., Petrenko A.K. Primenenie tehnologii UniTesK dlja funkcional'nogo testirovanija modelej apparatnogo obespechenija. / Preprint Instituta Sistemnogo Programmirovanija RAN, 2005.
7. International Organization for Standardization / International Electrotechnical Commission. ISO/IEC 24744. Software Engineering - Metamodel for Development Methodologies - 2007.