НАУКОВО-ТЕХНІЧНИЙ ЖУРНАЛ *ISSN 1681-4886*

4 (89)' 2011

Виходить 6 разів на рік

Видається з 23 квітня 1996 р.

ІНФОРМАЦІЙНО -КЕРУЮЧІ СИСТЕМИ НА ЗАЛІЗНИЧНОМУ ТРАНСПОРТІ

INFORMACIJNO-KERUÛCI SISTEMI NA ZALIZNICNOMU TRANSPORTI

Посвящается
двадцатилетию
независимости
Украины

В номере опубликованы статьи участников 24-й международной конференции «Перспективные компьютерные, управляющие и телекоммуникационные системы для железнодорожного транспорта Украины»

Зміст – Содержание – Contents

УДК 658.512.011:681.326:519.713

НЕМЧЕНКО В. П. к.т.н., профессор,
ИЗОТОВ А. С. аспирант (ХНУРЭ)

Построения системы генерации тестовых последовательностей для сетевых протоколов

Представил д.т.н., профессор Хаханов В.И.

Анализ предметной области и постановка задачи

Сегодня в связи с переходом на использование протоколов стека TCP/IPv6 актуальной является проблема тестирования сетевых протоколов нового поколения. При этом на первый план выступает задача разработки общей стратегии тестирования с учетом новых функциональных возможностей тестируемых сетевых протоколов.

Процедура тестирования сетевых протоколов состоит из нескольких последовательных этапов, среди которых можно выделить основные:

- Исходная модель - спецификация протокола.
- Машинная модель протокола.
- Язык программирования.
- Генерирование тестовой последовательности.

Естественно, полная структура системы тестирования не ограничивается вышеприведенными составляющими. Она может включать в себя как дополнительные этапы и составляющие, так и требовать выполнение иных этапов вместо указанных.

Задачей настоящей работы является разработка общей структуры системы генерации тестовых последовательностей для тестирования сетевых протоколов с характеристикой отдельных этапов.

1 Виды моделей сетевых протоколов

В дальнейшем под спецификацией протокола мы понимаем некоторое формальное описание структуры и функционирования сетевого протокола. В настоящее время, каждый сетевой протокол изначально представлен в виде опубликованной спецификации как RFC (Request for Comments). Таким образом, целесообразно в качестве первоначальной информации для нашей системы тестирования сетевых протоколов использовать соответствующие RFC, которые можно найти на официальной web-странице RFC.

Поскольку принятая в RFC описательная форма протоколов не является формальной, следует рассмотреть некоторые возможности моделирования протоколов.

Одной из наиболее очевидных форм представления спецификации протоколов является графическая форма представления в виде конечного графа автоматов (Finite State Machines - FSM). Главным преимуществом такой формы является простота представления спецификации в виде графа с последующей формализацией преобразований графа [FUJI].

Это правильно для относительно простых графов. Рост количества вершин и дуг в графе приводит к резкому их усложнению. А это, в свою очередь, влечет за собой снижение наглядности и простоты модели, а также резко усложняет обработку такой модели.

Очевидно, что еще одним видом модели может служить модель, построенная на использовании сетей Петри. Рассмотрению данного подхода будет посвящена отдельная работа.

1.1 Автоматная модель

Для дальнейшего рассмотрения воспользуемся спецификацией, представленной в графической форме, описывающей установление логической связи (Handshake) с помощью протокола TCP (рис. 1). Упрощая графический вид спецификации, мы приходим к графу, приведенному на рис. 2, что дает возможность применить к заданной модели известные методы преобразования графов.

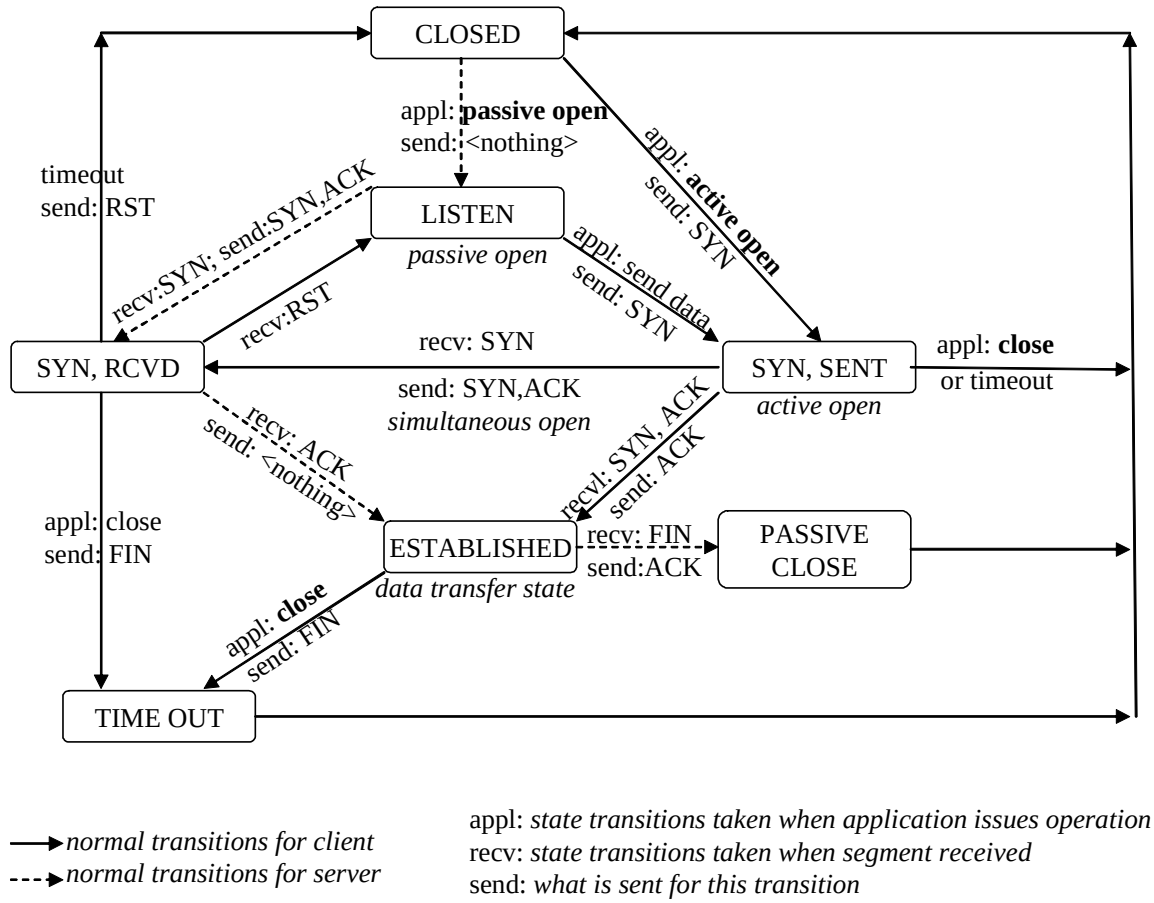


Рисунок 1 - Графическая модель фазы Handshake протокола TCP

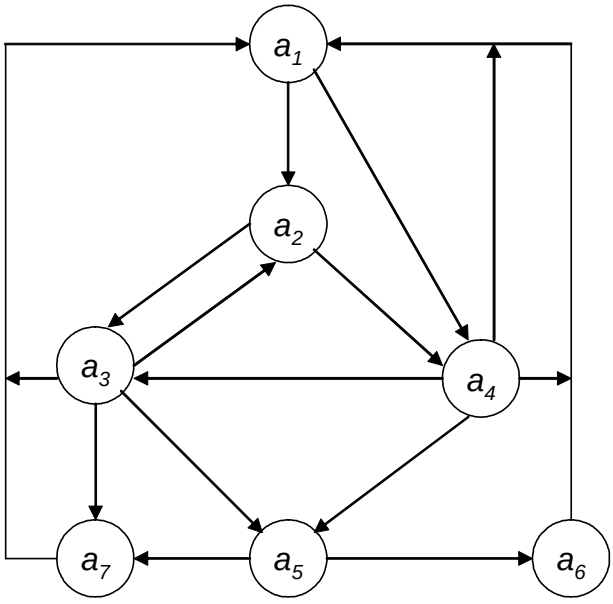


Рисунок 2 - Пример графа фазы Handshake

1.2 Модель Message Sequence Charts

Другой тип модели, который можно использовать для описания сетевых протоколов, основан на использовании MSCs (Message Sequence Charts). В качестве примера использования модели MSCs рассмотрим его использование для описания вышеприведенного фрагмента протокола TCP (рис. 2), служащего для установления логического соединения между узлами A и B.

Пусть узел A является инициализатором (Initiator User A) связи с узлом B. Т.е. узел B в нашем случае является ответчиком (Responder User B). Службы, использованные для осуществления подключения, носят имя Medium (рис. 3).

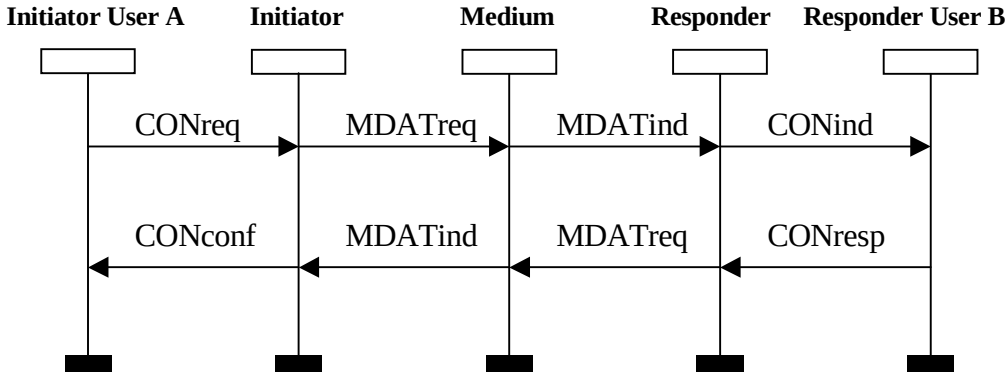


Рисунок 3 – Модель MSCs установления логической связи
(CON - connection; MDAT - Medium Data; req - Request; ind - Indication; conf – Confirmation; resp - Respons)

В общем случае имеем два направления прохождения запросов (*Request)* так называемая Служба примитивов - Service Primitives (SPs). Единица передаваемой информации называется Protocol Data Units (PDUs).

« Initiator User A ↔ Initiator ↔ Medium ↔ Responder User B ↔ Responder ».

Воспользуемся принципами описания процесса, приведенными на рис. 4, для построения MSCs для протокола TCP, устанавливающего логическое соединение, называемое иначе – виртуальный канал. Очевидно, что Medium должен размещаться между Initiator и Responder. Но на нашем примере мы его проигнорируем для упрощения общей структуры MSCs.

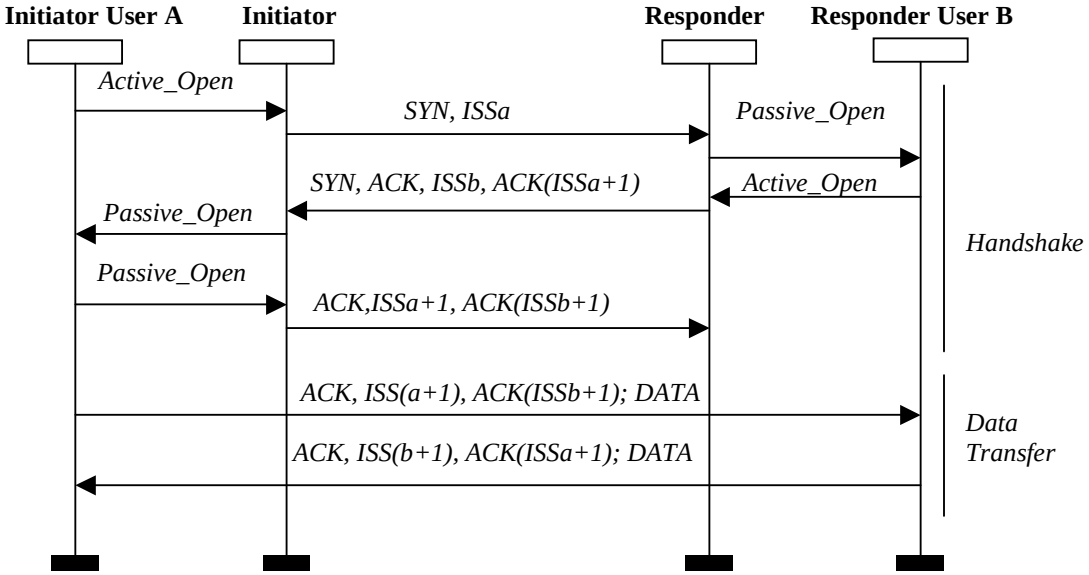


Рисунок 4 - MSCs установления виртуального канала протоколом TCP

Установление виртуального канала состоит в реализации трех последовательных шагов процедуры, называемой " Handshake ". Предположим, узел А инициализирует подключение к узлу В. Рассмотрим последовательность шагов, реализующих такое подключение.

Шаг 1: Узел А посылает своему протоколу TCP запрос на активное открытие порта - *Active_Open*. В ответ протокол TCP посылает узлу В запрос *SYN* (Synchronize Sequence Number) и помещает в поле Sequence Number исходное значение *ISNa* (Initial Sequence Number). В дальнейшем данное число будет служить решению задачи аутентификации узла *B*. Протокол TCP узла В посылает узлу В подтверждение о пассивном открытии порта *Passive_Open*.

Шаг 2: Узел В в свою очередь посылает своему протоколу TCP также запрос на активное открытие порта *Active_Open*. Протокол TCP узла В посылает протоколу TCP узла А сигналы *SYN*, *ACK* (Acknowledgment) и *ISNb* (свой собственный Initial Sequence Number), а также подтверждение о получении *ISNa* т.е. *ACK (ISNa+1)*.

Шаг 3: В завершение "рукопожатия" узел А посылает своему протоколу TCP подтверждение о пассивном открытии порта *Passive_Open*, а также по аналогии с вышесказанным посылает узлу В сигналы *ACK, ISNa+1* и *ACK (ISNb+1)*.

На этом виртуальный канал между узлами А и В установлен, и они могут обмениваться информацией (*Data Transfer*). При этом подчеркнем, что на самом деле установлено два канала связи для передачи информации от *A* к *B* и навстречу – от *B* к *A*.

Использование модели MSCs для описания сложных систем приводит к необходимости упрощения структуры MSC_S, потому что эта модель становится не наглядной из-за ее развитой и чересчур детализированной структуры. Это упрощение может быть осуществлено благодаря использованию модели Hyper Message Sequence Charts (HMSCs).

Главная особенность HMSCs состоит в так называемой гипертекстовой форме представления информации. В этом случае некоторые части MSC представлены в текстовой форме, связанной с соответствующими ссылками. Таким образом, некоторые части MSC детализируются, а другие представлены в сокращенной форме.

Например, описание протокола TCP или, более точно, его части, посвященной созданию виртуального канала протоколом TCP, может быть осуществлено в виде HMSCs, как это представлено на рис. 5.

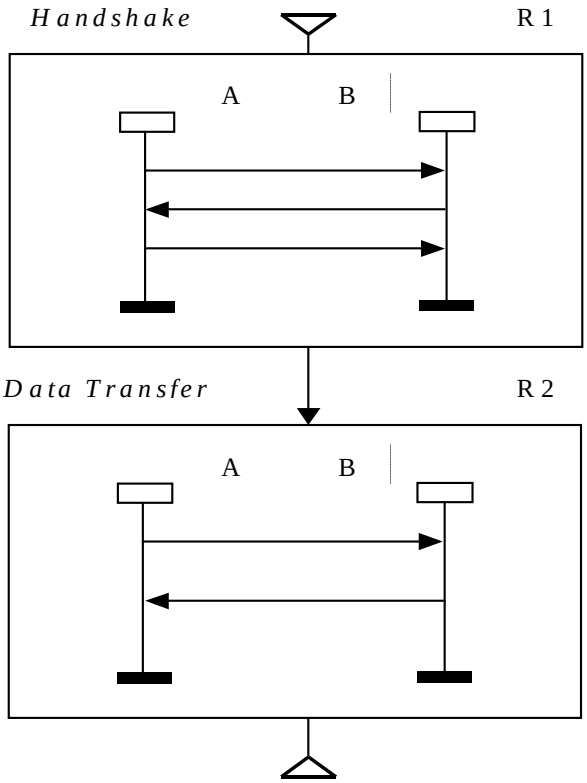


Рисунок 5 – HMSCs создания виртуального канала протоколом TCP

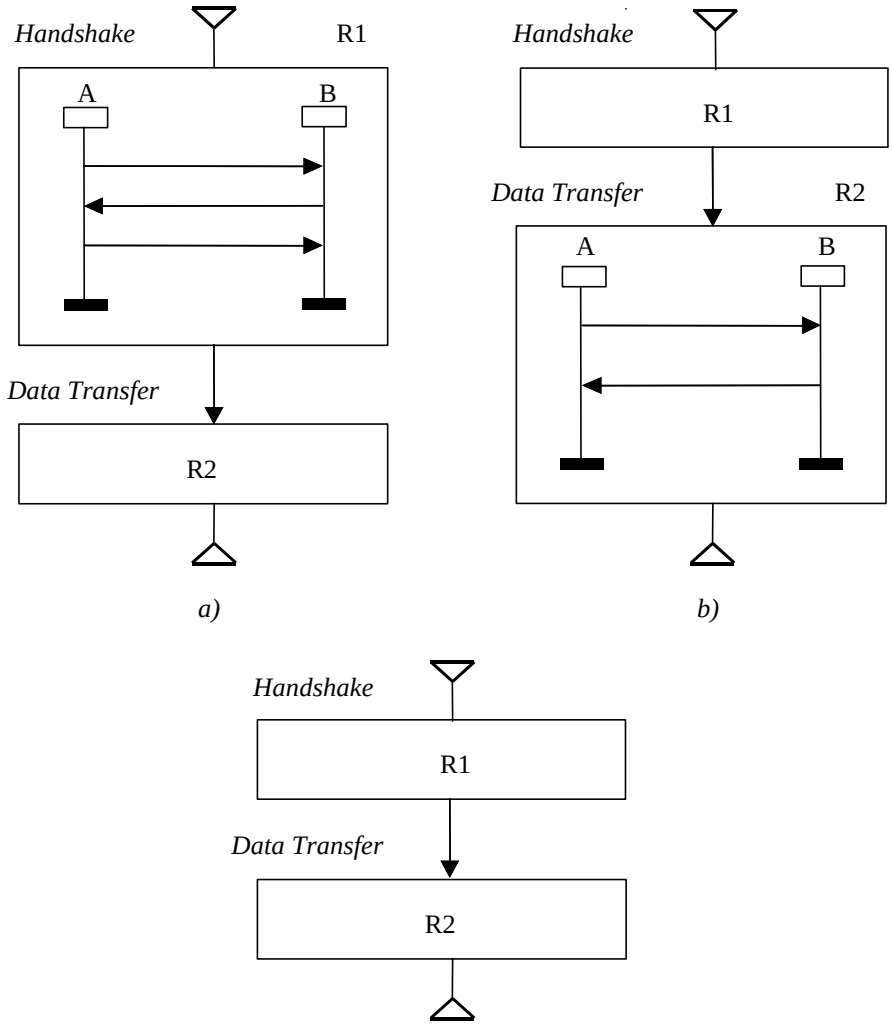
Как видим, HMSCs состоит из двух основных блоков R1: *Handshake* и R2: *Data Transfer*. На рисунке 6 оба блока представлены в подробной форме. Для нашего примера можно привести еще три возможности представления HMSCs (рис. 6, a , b, c).

Естественно, каждый сокращенный блок должен быть сопровожден комментариями, описывающими его структуру. Выбор одной из структур, приведенных на рис. 6 (a, b, c), полностью зависит от сложности моделируемой структуры.

Таким образом, при моделировании сетевого протокола мы последовательно проходим следующие этапы:

«Граф конечных состояний => MSCs => HMSCs».

Следующим этапом будет представление модели на некотором формальном языке.



a) *b)* *c)*

Рисунок 6 – Формы представления HyperMSCs

Естественно, каждый сокращенный блок должен быть сопровожден комментариями, описывающими его структуру. Выбор одной из структур, приведенных на рисунке 6 (a, b, c), полностью зависит от сложности моделируемой структуры.

Таким образом, при моделировании сетевого протокола мы последовательно проходим следующие этапы:

«Граф конечных состояний => MSCs => HMSCs».

Следующим этапом будет представление модели на некотором формальном языке.

2 Использование языков программирования описания модели

Прежде чем перейти к анализу и выбору языка моделирования сетевых протоколов, использующихся в системе генерации тестовых последовательностей, отметим, что в нашем случае мы не видим строгой «границы» между понятием *модели* и понятием *языка моделирования*. Главным образом, это относится к моделям MSCs и HMSCs с одной стороны, и языку моделирования TTCN, о котором речь будет идти ниже, с другой стороны. Оба могут быть трактованы и как *модели* и как *языки* в то же самое время. Это замечание касается только терминологической стороны и не меняет ничего существенно.

2.1 Язык SDL

Среди имеющихся сегодня языков моделирования наиболее приемлемым с нашей точки зрения является специализированный язык SDL (Specification and Description Language) - язык спецификации и описания, разработанный бывшим Консультативным Междуна-

родным Комитетом по Телеграфированию и Телефонии (CCITT). Он широко используется сегодня как формальный язык для описания поведения сетевых протоколов.

Изначально он был предназначен для описания структуры и функционирования систем реального масштаба времени в особенности сетей связи. Отметим, что SDL построен на базе модели конечных автоматов по объектно-ориентированной схеме. При этом вершины графа модели представляют процессы, а дуги между ними представлены сигналами, которыми либо процессы обмениваются между собой, либо процессы обмениваются с внешней по отношению к модели системы средой. Этот язык очень хорошо приспособлен для описания поведения сетевых протоколов, так как последние также используют модель конечных автоматов для их описания.

Таким образом, все это - серьезный аргумент в пользу использования языка SDL в системах генерации тестовых последовательностей для сетевых протоколов нового поколения в качестве базового языка моделирования протоколов.

2.2 Язык TTCN

Рассмотрим еще одну интересную возможность для представления тестовых последовательностей сетевых протоколов - TTCN (Tree and Tabular Combined Notation) [стандарт ISO9646, часть 3]. Это представление дает описание тестовых последовательностей независимое от тестовой архитектуры. Можно сказать, что форма TTCN есть язык TTCN, который определяет всю тестовую последовательность.

Различают следующие части описания TTCN:

- *Test suite overview* представляет собой список тестовой последовательности.
- *Declarations part* описывает тип сообщения и данных.
- *Constraints part* представляет условия параметров запроса.
- *Dynamic part* описывает последовательность запросов обмена для каждой тестовой последовательности.

Существуют две различные синтаксические формы представления TTCN: TTCN / MP (TTCN Machine Processible form) - текстовая форма и TTCN / GR (TTCN Graphical form) - графическая форма. Обе формы эквивалентны и они могут взаимно трансформироваться.

Структура TTCN описана и подробно рассмотрена в [GRAB94]. Здесь мы даем только несколько главных языковых характеристик TTCN. Обычно TTCN предусматривает две части:

- Декларирующая и ограничивающая часть.
- Динамическая часть.

Первая часть описывает условия, которые надо выполнять, чтобы реализовать данный тест. Вторая - представляет последовательность событий, которые должны иметь место на тестирующем оборудовании.

Теоретически, можно ожидать три результата тестирования:

- *Passed* – положительный результат тестирования (имеет место соответствие спецификации).
- *Inconclusive* - безрезультатный тест.
- *Fail* – отрицательный результат тестирования (нарушено соответствие спецификации).

На практике результат *Inconclusive* может быть интерпретирован как *Fail*. Тогда, в этом случае будет только два результата теста: *Passed* и *Fail* .

Развитие языка TTCN повлекло за собой также и появление его модификаций, например, TTCN-3.. Не вникая в детали, сделаем здесь только несколько замечаний относительно этого языка.

Обычно базовый модуль модуль состоит из двух модулей: *module definition part* и *module control part.*

Module definition part задает определения высокого уровня модуля. Эти определения могут быть использованы либо в самом модуле, либо в *module control part*, равно как они могут быть импортированы и из другого модуля.

Module control part описывает порядок исполнения тестовой последовательности.

Здесь мы хотим показать принцип генерирования тестов TTCN на базе MSCs, рассмотренных выше. Необходимо выполнить четыре шага, чтобы сгенерировать тест в форме TTCN исходя из MSCS.

1. Тестируемый протокол должен быть представлен в виде MSCs.

2. Поскольку нас здесь интересует только поведение тестирующей части системы, мы изымем из MSCs ту часть, которая не касается тестера. А именно, в нашем примере (рис. 4) изымем часть MSCS, связанную с блоками *Initiator* и *Responder*. Остальная часть MSCs представляется в виде последовательности действий тестера, как это показано на рис. 8. При этом тестер посылает команду (в TTCN обозначается «!») и получает ответ (отмечается символом «?». Отметим, что в общем случае MSCs может осуществить несколько последовательностей такого рода.

3. Последовательности, полученные на шаге 2, группируются в эквивалентные классы. При генерации теста TTCN берется только один представитель из каждого класса.

4. Последовательности, выбранные на шаге 3, преобразуются в форму обозначения TTCN.

Таким образом, последовательность действий тестера может быть представлена так, как это показано на рисунке 7.

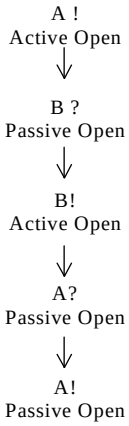


Рисунок 7 - Последовательность действий тестера

Символ "! " после имени тестера (A или B) говорит, что именно этот тестер вырабатывает нижеследующую команду. Символ "? " говорит, что данная команда должна быть подана на тестер.

На рис. 8 смещение команд в TTCN соответствует смещению данных команд во времени. Альтернативные команды должны расположиться на той же вертикальной линии равно как и альтернативные ответы.

Test Step Dynamic Behaviour

Test Step Name : Preamble
Groupe :
Objective :
Defaults : UnexpectedEvents
Comments : Preamble of the test case HANDSHAKE

Nr	Label	Behaviour Description	Constraints Ref	Verdict	Comments
1		A!ActiveOpen			
2		B?PassiveOpen			
3		B!ActiveOpen			
4		A?PassiveOpen			
5		A!PassiveOpen			

Detailed Comments :

Рисунок 8 - Тест в форме TTCN

Проведенный выше анализ показывает, что все рассмотренные элементы могут быть включены в автоматизированную систему генерации тестов.

Выводы

Таким образом, нами была предложена концепция структуры системы генерации тестов для тестирования сетевых протоколов, в том числе и для нового поколения IPV6. Естественно, данный подход может быть использован также и для тестирования сетевых протоколов ныне действующего поколения IPV4, а также других реализаций.

Структуру системы генерации тестов для сетевых протоколов можно представить в следующем виде:

- - Модель FSM (сети Петри).
- - Модель MSCs (HMSCs).
- - Язык SDL.
- - Язык TTCN (TTCN-3).

Здесь приведены лишь общие соображения относительно структуры системы. Дальнейшая проработка отдельных элементов системы будет произведена в последующих работах.

Литература

1. *Fujiwara S., Bochmann G. V., Khendek F., Amalou M., Ghedamsi A.* Test selection based on finite state models // Universite de Montreal. Canada, 2006.
2. *Grabowski J. TTCN-3.* A new Test Specification Language for Black-Box Testing of Distributed Systems. TCS'2000. – Washington D.C., June 2000.
3. *Schaff A., Nemchenko V.* Test of the new generation internet protocols IPv6. // Radioélectronika i informatika. – 2001. – No.1. – Pp. 87-89.
4. *Graubman P,* etc. Component Interface Description Using HyperMSCs and Connectors, 2001

Резюме

Рассмотрена проблема построения системы генерации тестовых последовательностей для проверки сетевых протоколов. В качестве модели для представления спецификации протокола было предложено использовать либо автоматную модель, либо метод сетей Петри. Предложенный подход может быть использован для решения поставленной задачи

Розглянуто проблему побудови системи генерації тестових послідовностей для перевірки мережевих протоколів. В якості моделі для відображення специфікації протоколу запропоновано використання автоматної моделі, або методу мереж Петрі. Запропонований підхід може бути використаний для розв'язання поставленої задачі

It was considered the problem of the test sequences system constructing for the network protocols test. It was shown that cellular automata or Petri nets method can be regarded as a model for the representation of protocol specification. The proposed approach can be used for solve of given problem

Ключевые слова: тест, сетевой протокол, графовая модель

Поступила 01.06.2011 г.