Особенности применения технологии UniTESK для тестирования функций мобильности в протоколе IPv6

Д.В. Зацепин, В.З. Шнитман

Аннотация. Статья посвящена разработке тестового набора для проверки соответствий реализаций мобильного узла спецификациям протокола Mobile IPv6 [1]. Для построения тестового набора использовалась передовая технология автоматического тестирования UniTESK [2] и инструмент CTesK [3], разработанный на основе этой технологии. В ходе выполнения работы было выявлено несколько особенностей поведения одного из объектов протокола – мобильного узла, которые затрудняют его тестирование в рамках указанной технологии. В статье подробно описаны эти особенности и способы преодоления трудных моментов в условиях ограничений технологии UniTESK и поддерживающего эту технологию инструмента CTesK.
Работа выполнялась в Институте системного программирования РАН в рамках проекта «Верификация функций безопасности и мобильности протоколов IP» при поддержке гранта РФФИ № 04-07-90308.

1. Введение

В последнее время мобильные устройства получают все большее распространение, и потребность постоянного доступа к сети Интернет таких устройств является естественной. Предполагается, что в недалеком будущем Интернет примет на вооружение протокол нового поколения IPv6 [4]. Для поддержки функций мобильности техническим комитетом по проектированию Интернет (Internet Engineering Task Force, IETF) разработано расширение этого нового протокола, получившее название Mobile IPv6, и выпущен документ RFC 3775 "Mobility Support in IPv6" [1], который можно рассматривать в качестве будущего стандарта (процесс стандартизации этого расширения, как и самого протокола IPv6, на сегодняшний день еще не завершен).

Функциональность Mobile IPv6 нацелена на поддержку целого ряда устройств, в частности, персональных компьютеров, ноутбуков, карманных компьютеров, мобильных телефонов, маршрутизаторов и точек доступа для беспроводных сетей. Очевидно, в перспективе ожидается появление большого количества реализаций Mobile IPv6 от различных производителей.

Тестирование соответствия стандартам является одним из основных средств обеспечения совместимости реализаций различных производителей и позволяет решать вопросы повышения надежности и отказоустойчивости глобальной сети. В настоящее время отсутствие средств такого тестирования является одной из причин, препятствующих внедрению перспективных технологий в практику.

В Институте системного программирования РАН (ИСП РАН) в течение ряда лет ведутся работы по исследованию и развитию методов формального моделирования телекоммуникационных протоколов. В частности, одним из направлений работ по проекту «Верификация функций безопасности и мобильности протоколов IP», который выполнялся при поддержке гранта РФФИ № 04-07-90308, была разработка тестового набора, обеспечивающего проверку соответствия реализаций стандарту Mobile IPv6. В качестве основы для построения такого тестового набора использовался опыт создания и внедрения разработанной в ИСП РАН методологии тестирования на основе формальных спецификаций UniTESK [2].

В статье рассматриваются некоторые особенности протокола Mobile IPv6, затрудняющие тестирование реализаций с помощью этой технологии, а также способы преодоления возникших трудностей в рамках ее ограничений. В качестве объекта тестирования принята реализация основного объекта протокола Mobile IPv6 – мобильного узла (узла, который может совершать перемещения между различными сегментами сети Интернет). Такой выбор обусловлен тем, что тестирование мобильного узла представляет наибольшую сложность, в частности, из-за того, что он является инициатором большинства служебных процедур обмена протокольными сообщениями.

Статья содержит восемь разделов, включая введение и заключение. В следующих двух разделах кратко рассматриваются вопросы функционирования протокола Mobile IPv6, а также возможности инструмента CTesK [3], реализующего технологию UniTESK. Четвертый раздел посвящен проблемам тестирования мобильного узла и способам их решения. В пятом разделе описываются структура разработанного тестового набора и этапы исполнения сценарных функций. В шестом разделе рассматриваются способы адаптации инструмента CTesK к предложенной схеме тестового набора. Седьмой раздел посвящен результатам применения разработанного тестового набора для тестирования одной из известных реализаций Mobile IPv6. В заключении приведены основные выводы и результаты работы.

2. Обзор протокола Mobile IPv6

Данный раздел является кратким введением в протокол Mobile IPv6.

Как уже отмечалось, этот протокол является расширением протокола нового поколения IPv6, которое позволяет узлу, совершающему перемещения по сети Интернет, все время оставаться доступным по адресу, который он имел в

первоначальной (домашней) сети. Такая возможность позволяет избежать потери соединений транспортного уровня при перемещениях межсетевого уровня модели TCP/IP, а также дает возможность другим узлам устанавливать соединение с *мобильным узлом* (Mobile Node) в то время, когда он находится за пределами домашней сети. Для простоты мы будем рассматривать случай, когда у мобильного узла есть только один домашний адрес. Поведение мобильного узла в случае нескольких домашних адресов можно считать аналогичным.

Когда мобильный узел находится в домашней сети, никакой дополнительной функциональности от узлов, участвующих во взаимодействии, не требуется, и мобильный узел ведет себя как обычный IPv6-узел (см. Рис. 1).

Когда мобильный узел переходит в сеть, которая обслуживается другими маршрутизаторами и имеет другие сетевые префиксы, без дополнительной поддержки со стороны Mobile IPv6 узел становится недоступным по своему домашнему адресу. Пакеты, отправляемые ему другими узлами, будут по-прежнему приходить в домашнюю сеть, и не будут доходить до мобильного узла. Это приведет к потере установленных соединений.

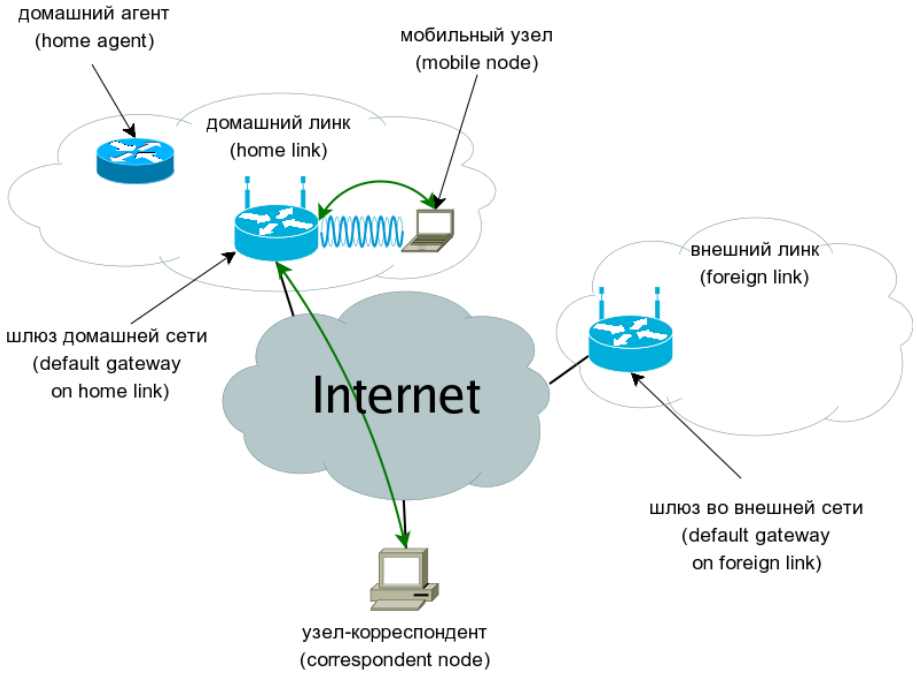

Рис. 1. Мобильный узел в домашней сети

Для того чтобы этого не происходило, в протоколе Mobile IPv6 вводится специальный вид маршрутизаторов, которые называются *домашними агентами* (Home Agents). Домашний агент (см. Рис. 2) всегда находится в домашней сети и осуществляет переадресацию пакетов мобильному узлу в то время, когда последний находится за пределами домашней сети. Это осуществляется при помощи механизма, который называется *двусторонним туннелированием* (Bi-directional Tunneling).

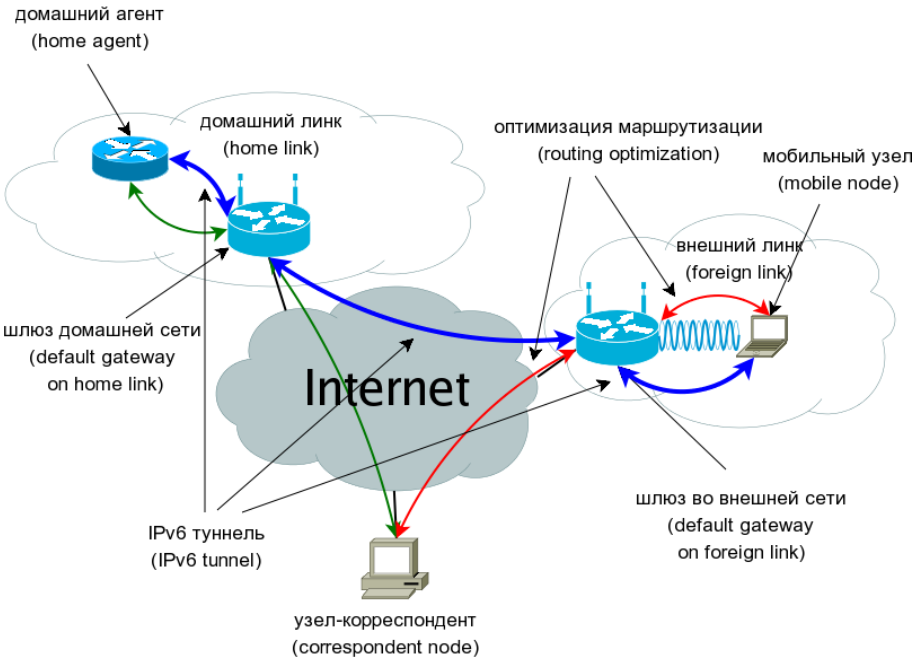

Рис. 2. Мобильный узел вне дома

Структура IPv6-пакета подробно описана в RFC 2460 [4]. Однако здесь следует отметить, что в протоколе IPv6 в пакет могут добавляться заголовки расширения, которые содержат некоторую дополнительную информацию о том, как этот пакет должен обрабатываться конечным получателем или промежуточными узлами сети. Заголовки расширения связаны в односвязный список при помощи поля Next Header, которое содержится во всех заголовках уровня IP. Поле Next Header последнего заголовка расширения IPv6-пакета указывает на протокол транспортного уровня, сегмент которого передается в качестве полезной нагрузки. Каждый из заголовков расширения имеет фиксированную часть, которая всегда должна присутствовать в заголовке данного типа, и может содержать дополнительную часть в виде опций. Опции располагаются в конце заголовка и описываются в представлении TLV (Type-Length-Value).

Когда мобильный узел попадает во внешнюю сеть, он формирует один или несколько *временных адресов* (Care-of Addresses) согласно механизму автоматической конфигурации адресов [5], который используется во внешней сети. После этого мобильный узел выбирает один из своих временных адресов и сообщает его домашнему агенту; такой адрес называется *основным временным адресом* (Primary Care-of Address).

Это осуществляется при помощи процедуры, которая называется *регистрацией основного временного адреса* (Primary Care-of Address Registration). Для этого мобильный узел отправляет домашнему агенту служебное сообщение Binding Update со своего основного временного адреса. В этом сообщении мобильный узел при помощи заголовка расширения Destination Options, содержащего опцию Home Address, указывает свой домашний адрес, для которого выполняется данная процедура. В ответ на сообщение Binding Update домашний агент отправляет служебное сообщение Binding Acknowledgement, которое либо свидетельствует о том, что сообщение Binding Update принято, либо указывает причину, по которой оно было отклонено. Для обеспечения безопасности такого обмена сообщениями используются средства протокола IPsec.

Перед отправлением сообщения Binding Acknowledgement домашний агент запоминает связь между домашним адресом мобильного узла и его основным временным адресом в структуре данных, которая носит название *кэш привязок* (Binding Cache), а сама связь называется *привязкой* (binding).

После установления привязки домашний агент начинает перехватывать пакеты, которые приходят на домашний адрес мобильного узла, и через IPv6-туннель пересылать их мобильному узлу на его основной временный адрес.

Такой перехват пакетов, направленных мобильному узлу, осуществляется при помощи механизма, называемого Proxy Neighbor Discovery. Суть этого механизма заключается в том, что домашний агент отправляет необходимые сообщения Neighbor Advertisement, используемые в механизмах Address Resolution и Neighbor Unreachability Detection, от имени мобильного узла, но со своим адресом канального уровня. Таким образом, мобильный узел, находясь в чужой сети, через IPv6-туннель с домашним агентом, получает сообщения, которые отправляются ему в домашнюю сеть.

Мобильный узел, в свою очередь, поддерживает структуру данных, которая называется Binding Update List. В этой структуре данных содержится информация обо всех привязках и об их состоянии. Перед отправлением пакета в сеть мобильный узел просматривает Binding Update List. Если там есть привязка, для которой зарегистрированный домашний адрес совпадает с *адресом отправителя пакета* (Source Address), то пакет отправляется не напрямую, а через IPv6-туннель с домашним агентом, указанным в привязке. Домашний агент, получив из IPv6-туннеля такой пакет, отправляет его из домашней сети по обычному маршруту. Таким образом, мобильный узел всегда остается доступным по своему домашнему адресу.

Узел, с которым мобильный узел обменивается сообщениями, называется *узлом-корреспондентом* (Correspondent Node). Для обмена информацией с мобильным узлом при помощи двустороннего туннелирования от узла-корреспондента требуется лишь поддержка базовой функциональности IPv6. Дополнительной поддержки Mobile IPv6 от узла-корреспондента в этом случае не требуется. Однако двустороннее туннелирование создает дополнительный обмен сообщениями и дополнительную нагрузку на домашнего агента. Протокол Mobile IPv6 позволяет этого избежать и производить обмен информацией напрямую (не через домашнюю сеть), если узел-корреспондент поддерживает дополнительную функциональность Mobile IPv6, которая называется *оптимизацией маршрута* (Route Optimization).

Узел-корреспондент, поддерживающий оптимизацию маршрута, так же как и домашний агент, имеет кэш привязок, в котором содержится информация о привязках с мобильными узлами и об их состояниях. У мобильного узла, в свою очередь, в структуре Binding Update List содержится информация о привязках с узлами-корреспондентами.

Перед отправлением в сеть пакета, у которого адрес отправителя совпадает с домашним адресом, мобильный узел просматривает Binding Update List, и если находит привязку с узлом-корреспондентом, адрес которого совпадает с адресом получателя пакета, то пакет отправляется напрямую. При этом в качестве адреса отправителя пакета указывается временный адрес, а в сообщение добавляется заголовок расширения Destination Options, который содержит опцию Home Address с домашним адресом мобильного узла. Если же такой привязки нет, то мобильный узел ищет привязку с домашним агентом и отправляет пакет через туннель домашнему агенту, который пересылает его узлу-корреспонденту.

При получении пакета с опцией Home Address узел-корреспондент ищет в кэше привязок запись, у которой временный адрес совпадает с адресом отправителя, указанным в пакете, а домашний адрес совпадает с адресом, указанным в опции Home Address. Если такая привязка находится, то узел-корреспондент меняет местами адрес отправителя пакета и адрес, указанный в опции Home Address; затем пакет передается на дальнейшую обработку.

При отправлении пакета в сеть узел-корреспондент просматривает кэш привязок и ищет там привязку, у которой домашний адрес мобильного узла совпадает с *адресом получателя пакета* (Destination Address). Если такая привязка находится, то узел-корреспондент подставляет временный адрес, указанный в привязке, вместо адреса получателя, и в пакет добавляется заголовок расширения Routing Header Type 2, в котором указывается домашний адрес мобильного узла.

При получении такого пакета мобильный узел проверяет, что адрес, указанный в заголовке Routing Header Type 2, является его домашним адресом и меняет местами адрес получателя пакета с адресом, указанным в данном заголовке. Затем пакет передается на дальнейшую обработку. Такой механизм

обмена сообщениями между мобильным узлом и узлом-корреспондентом (см. Рис. 2) является прозрачным для протоколов верхнего уровня и носит название оптимизации маршрута.

Механизм установления привязки с узлом-корреспондентом аналогичен механизму установления привязки с домашним агентом. Для этого мобильный узел отправляет сообщение Binding Update узлу-корреспонденту, в котором содержатся домашний и временный адрес. В некоторых случаях узел-корреспондент должен отправить в ответ сообщение Binding Acknowledgement, подтверждающее установление привязки, хотя обычно этого не требуется.

Для обеспечения безопасности механизма установления привязки с узлом-корреспондентом в сообщения Binding Update и Binding Acknowledgement добавляется опция Binding Authorization Data, которая содержит криптографическую хэш-сумму этих сообщений. Хэш-сумма вычисляется по алгоритму HMAC_SHA1 с использованием ключа управления привязкой (Kbm), который генерируется перед обменом сообщениями Binding Update и Binding Acknowledgement при помощи *процедуры обратной маршрутизируемости* (Return Routability Procedure).

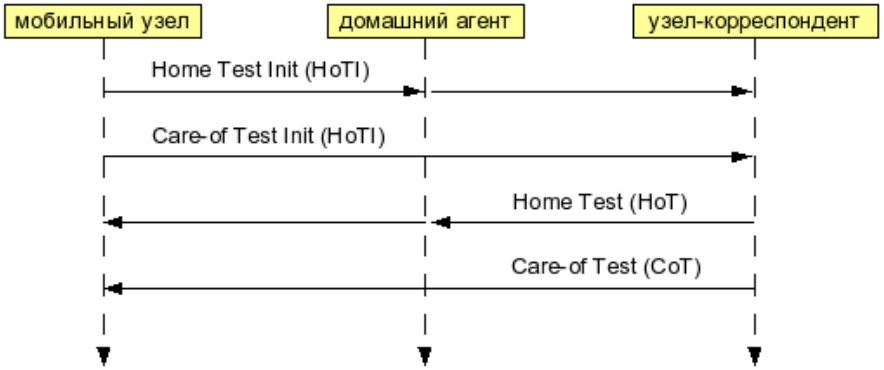

Рис. 3. Процедура обратной маршрутизируемости (return routability procedure)

Процедура обратной маршрутизируемости выполняется с целью аутентификации узла-корреспондента. Для этого обмен сообщениями с узлом-корреспондентом производится по двум маршрутам: напрямую и через домашнюю сеть (см. Рис. 3). А именно, мобильный узел посылает со своего временного адреса сообщение Home Test Init через туннель с домашним агентом, а сообщение Care-of Test Init – напрямую. В ответ на эти сообщения узел-корреспондент посылает сообщения Home Test на домашний адрес и Care-of Test на временный адрес мобильного узла; эти сообщения содержат

маркеры home keygen token и care-of keygen token соответственно. При помощи этих маркеров и генерируется ключ управления привязкой (Kbm).

Протокол Mobile IPv6 включает в себя несколько вспомогательных служебных процедур обмена сообщениями для мобильного узла: Generic Movement Detection, Dynamic Home Agent Address Discovery и Mobile Prefix Discovery. Эти процедуры используют сообщения протокола ICMPv6. Процедура Generic Movement Detection позволяет отследить перемещение мобильного узла при помощи средств протокола Neighbor Discovery [6]. Для двух оставшихся процедур вводятся четыре новых типа сообщений протокола ICMPv6 [7]. Процедура Dynamic Home Agent Address Discovery позволяет узнать адрес домашнего агента в то время, когда мобильный узел находится за пределами домашней сети. Процедура Mobile Prefix Discovery позволяет мобильному узлу узнать об изменении префиксов в домашней сети в то время, когда мобильный узел находится за ее пределами, и, тем самым, принимать участие в смене префиксов домашней сети (Network Renumbering).

3. Инструмент CTesK

Для построения тестового набора в данной работе использовался инструмент CTesK, который позволяет эффективно распараллелить разработку тестового набора и автоматизировать разработку тестовых сценариев. Ниже приведено краткое описание возможностей этого инструмента и обзор технологии UniTESK, которую он реализует.

В инструменте CTesK объектом тестирования является целевая система. У этой системы имеется набор интерфейсов, с помощью которых она взаимодействует с окружением. Набор этих интерфейсов изначально определен и специфицирован. Все требования, которые предъявляются к системе, накладываются на наблюдаемые действия системы, которые можно отследить через интерфейсы ее взаимодействия с окружением. Эти требования не затрагивают внутренней реализации системы, т.е. тестирование производится по принципу «черного ящика».

При использовании инструмента CTesK требования, предъявляемые к целевой системе, должны быть описаны на формальном языке SeC, который является спецификационным расширением языка C. Впоследствии из таких требований автоматически генерируется исполняемый код, который используется в процессе тестирования.

Взаимодействие с целевой системой осуществляется при помощи стимулов и реакций. Стимулами называются средства, при помощи которых можно передавать целевой системе какую-либо информацию. Целевая система может непосредственно возвращать некоторый ответ на стимул, а также изменять свое внутреннее состояние. Такой ответ и изменение состояния, если имеется доступ к состоянию системы, возвращаются в тестовую систему, и производится проверка соответствия поведения системы наложенным на нее

требованиям. Требования описываются на формальном языке в виде постусловий. Для стимулов в виде предусловия описываются ограничения, при которых можно вызывать данный стимул. Дополнительно могут накладываться ограничения на возможные значения данных определенного типа в виде инвариантов.

Некоторые следствия работы системы и результаты воздействия на нее определенных стимулов могут наблюдаться не сразу, а после завершения внутренних процессов. Такое поведение приводит к тому, что реакция на стимул поступает наружу через какое-то время после воздействия на систему одного или нескольких стимулов, или может даже проявляться без воздействия стимулов на систему. Такую информацию не во всех случаях можно охарактеризовать как ответ на какой-либо стимул, и она называется реакцией (отложенной реакцией) системы. На реакции системы обычно также накладывается ряд требований. Аналогично случаю со стимулами, в инструменте CTesK они описываются в виде пред- и постусловий. Предусловия определяют ограничения на состояния системы, при которых может происходить данная реакция. А постусловия описывают ограничения на саму реакцию, то есть на данные, которые передаются окружению при помощи этой реакции, и на изменение состояния целевой системы, которое может происходить при возникновении такой реакции.

Для описания требований к стимулам и реакциям тестовый набор должен обладать информацией о состоянии целевой системы и о его изменениях. Для этого тестовым набором поддерживаются структуры данных, моделирующие это состояние. Далее состояние этих структур данных будет называться модельным состоянием. Если имеется доступ к внутренним структурам данных целевой системы, описывающих это состояние, то возможна синхронизация состояния модели и целевой системы. В таком случае возможна проверка требований, касающихся изменения состояния внутренних структур данных. В противном случае состояние модели и состояние целевой системы изменяются независимо и синхронизируются лишь на начальном этапе тестирования. При этом требования накладываются только на действия системы, наблюдаемые извне. Нарушение требований обработки внутренних структур данных проверяются лишь неявно при последующих некорректных ответах на стимулы и возникновении некорректных реакций.

Тестовый набор, разрабатываемый при помощи инструмента CTesK, может быть ориентирован не на конкретную целевую систему, а на группу целевых систем, поведение которых описывается одними и теми же требованиями. Причем некоторые целевые реализации могут предоставлять доступ к внутренним структурам данных, а другие нет. Для обеспечения универсальности в тестовый набор включается специальная часть, которая называется медиатором. Для каждого отдельного стимула медиатор осуществляет преобразование данных из модельного представления тестового набора в представление целевой системы, а для каждой реакции выполняет

обратное преобразование. Таким образом, медиатор зависит от конкретной целевой системы. Поэтому, для того чтобы произвести тестирование новой целевой системы, необходимо заново разработать для нее медиаторы.

Вся информация о ходе тестирования сохраняется в специальном файле, который называется тестовой трассой. По данным, которые содержатся в этом файле, генерируется отчет об ошибках и покрытии. Трасса хранится в формате XML, что позволяет легко разрабатывать средства для ее анализа.

В инструменте CTesK имеются средства для автоматического измерения тестового покрытия. Для этого в описание стимула добавляется специальный блок coverage, в котором, исходя из модельного состояния и параметров вызова стимула, определяется, какую из ветвей функциональности целевой системы задействует данный стимул. Инструмент CTesK включает в себя утилиту, которая после завершения тестирования по информации, содержащейся в тестовой трассе, генерирует подробную статистику и отчет о тестовом покрытии.

Стимулы и реакции на языке SeC описываются в виде спецификационных функций. Для стимула такая спецификационная функция включает в себя предусловие, постусловие и блок coverage. Для реакции – только предусловие и постусловие. Для каждой спецификационной функции разрабатывается медиатор, который оформляется в виде функции. Медиатор для стимула имеет два блока: блок call, который отвечает за передачу соответствующего стимула реализации и за получение результата, и блок state, который отвечает за обновление модельного состояния. Для реакции медиатор содержит только блок state, который выполняет аналогичную функцию.

В инструменте CTesK термин *тестовый сценарий* используется для обозначения группы тестовых операций, обычно предназначаемых для тестирования логически связанных аспектов функциональности целевой системы. Если в ходе исполнения тестового сценария обнаруживается ошибка, то тестовый сценарий завершается. Такая мера является вынужденной, так как после возникновения ошибки состояния тестового набора и целевой системы оказываются не синхронизированными.

Выполнение тестового сценария заключается в вызовах сценарных функций. Сценарная функция содержит исполняемый код и вызовы стимулов. В ходе выполнения сценарной функции собираются реакции от системы (см. Рис. 4). После завершения сценарной функции запускается механизм, называемый *сериализацией*. В процессе сериализации перебираются возможные порядки вызовов стимулов и наступления реакций (см. Рис. 5). Чтобы поведение целевой системы могло считаться корректным, требуется, чтобы хотя бы один из порядков удовлетворял требованиям в виде пред- и постусловий стимулов и реакций.

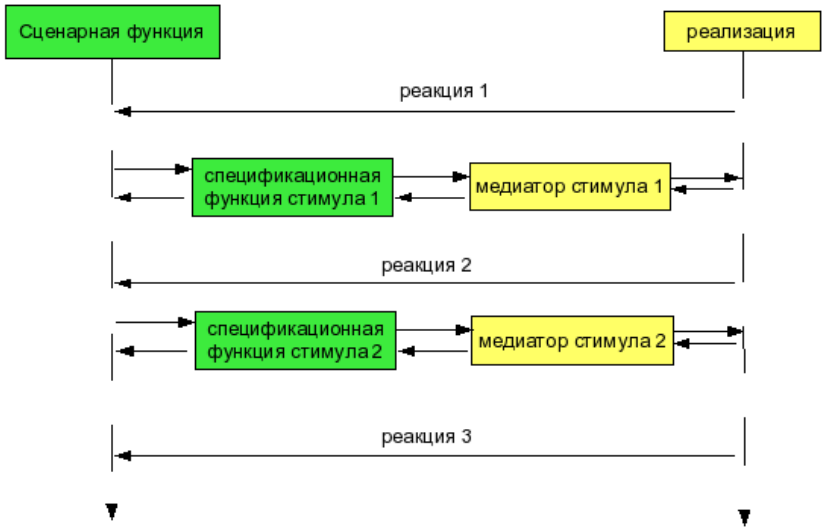

Рис. 4. Исполнение сценарной функции

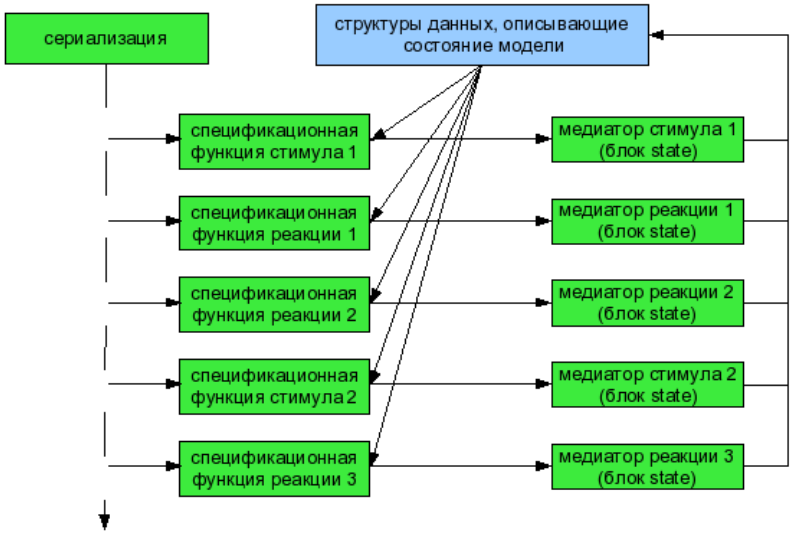

Рис. 5. Один из сценариев сериализации

В процессе проверки одного из порядков сериализации тестовый сценарий может проходить через последовательность состояний модели. Такие

промежуточные состояния называются нестационарными. Вызов сценарной функции должен обязательно закончиться в стационарном состоянии, иначе поведение целевой системы является некорректным. Кроме того, все допустимые последовательности сериализации, которые заканчиваются в стационарном состоянии, должны приводить к одному и тому же состоянию. Иначе выполнение тестового сценария будет прервано с сообщением об ошибке. В функциях, выполняющих сериализацию, используются методы сохранения состояния модели, восстановления состояния модели, а также определения, является ли данное состояние стационарным.

Если имеется информация о частичном порядке вызовов стимулов и поступления реакций, то можно существенно сократить процесс сериализации по сравнению с полным перебором или даже свести ее к рассмотрению одного единственного варианта.

В инструменте CTesK предусмотрены средства для разбиения множества модельных состояний на группы. Такое разбиение называется *факторизацией состояний*, а полученные обобщенные состояния – *состояниями тестового сценария*. На выбор факторизации накладывается следующее ограничение: после вызова определенной сценарной функции из определенного начального состояния тестового сценария тестовый сценарий всегда должен перейти в одно и то же конечное состояние. Пример факторизации модельных состояний схематично представлен на Рис. 6.

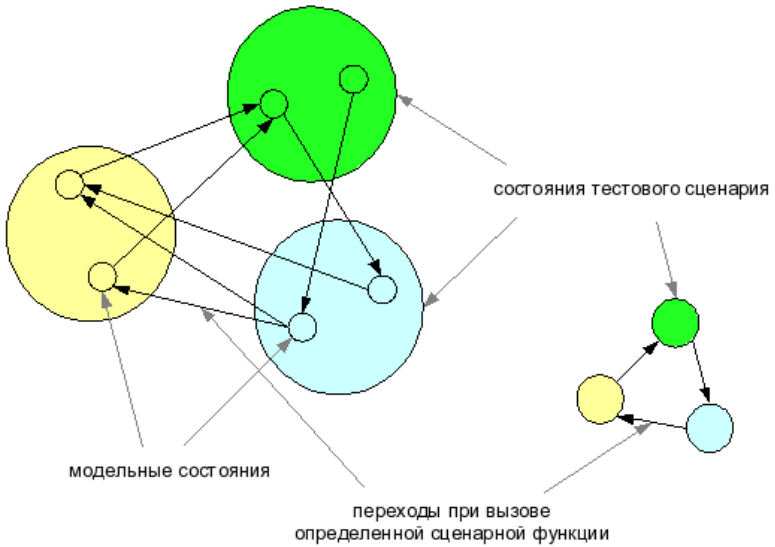

Рис. 6. Переходы в модельных состояниях и состояниях тестового сценария

Для осуществления факторизации разработчик должен предоставить тестовому сценарию функцию, которая по текущему модельному состоянию возвращает состояние тестового сценария.

При разработке тестового сценария разработчик определяет начальное модельное состояние и набор сценарных функций. Тестовый сценарий автоматически выбирает порядок вызовов сценарных функций. Такую функциональность осуществляет компонент под названием *обходчик*. Обходчик перебирает вызовы всех возможных сценарных функций во всех достижимых состояниях тестового сценария. Это осуществляется при помощи обхода графа, вершинам которого соответствуют состояния тестового сценария, а дугам – переходы по вызовам сценарных функций. Такой граф задается для тестового сценария неизбыточным образом путем указания начального состояния тестового сценария, набора сценарных функций и функции, определяющей состояние тестового сценария после выполнения сценарной функции.

Инструмент CTesK включает в себя два обходчика: dfsm [8], который реализует обход в глубину графа состояний тестового сценария, и ndfsm [9], использующий жадный алгоритм для выбора ближайшей вершины, в которой применялись не все возможные воздействия. На практике второй обходчик оказывается более эффективным, поэтому он и использовался в разрабатываемом тестовом наборе.

Для корректной работы обоих обходчиков граф состояний тестового сценария должен удовлетворять требованиям связности и детерминированности. Связность в данном случае может формулироваться следующим образом: из любого состояния тестового сценария, в которое можно попасть из начального состояния при помощи последовательности вызовов сценарных функций, можно вернуться в начальное состояние при помощи последовательности вызовов сценарных функций. Условие детерминированности означает, что при вызове одной и той же сценарной функции из одного и того же состояния тестового сценария мы всегда попадаем в одно и тоже конечное состояние тестового сценария.

Обходчик в своей работе может пользоваться компонентом под названием *итератор*. Итератор используется, если в сценарной функции на разных этапах ее запуска необходимо вызывать стимулы с различными параметрами. Такую функциональность можно осуществить при помощи написания отдельной сценарной функции для каждой группы параметров, но в таком случае может понадобиться очень большое количество сценарных функций.

В языке SeC используется специальная конструкция iterate для перебора параметров стимула. Синтаксис этой конструкции напоминает синтаксис оператора цикла for. Счетчик цикла для этой конструкции называется счетчиком итерации. Таким образом, когда обходчик встречает в коде сценарной функции такую синтаксическую конструкцию, он вызывает итератор. Итератор возвращает значения счетчиков итерации для вызова

данной сценарной функции так, чтобы в каждом состоянии тестового сценария осуществлялся перебор всех возможных наборов счетчиков итерации. С использованием итератора обходчик вызывает все сценарные функции со всеми возможными значениями счетчиков итерации во всех достижимых состояниях тестового сценария. Такая автоматизация существенно упрощает разработку тестового сценария.

4. Проблемы тестирования мобильного узла и способы их решения

Реализация функциональности мобильного узла интегрируется в IPv6-стек, и воздействовать на нее можно двумя способами: при помощи отправления сообщений с транспортного уровня и при помощи сообщений с канального уровня. Кроме того, в конкретных реализациях Mobile IPv6 могут предоставляться средства для чтения или изменения внутренних структур данных, описывающих состояние реализации. При сравнении двух различных реализаций эти интерфейсы могут существенно отличаться, так как стандарт не накладывает на них каких-либо ограничений. В некоторых реализациях Mobile IPv6 таких интерфейсов может вообще не быть, поэтому при разработке тестового набора было принято решение не использовать взаимодействие с целевой реализацией при помощи интерфейсов доступа к внутренним управляющим структурам данных.

Более того, с целью построения универсального тестового набора, а также для упрощения синхронизации и анализа результатов, полученных в ходе исполнения сценарных функций, воздействия на целевую реализацию при помощи сообщений с транспортного уровня также не используются. Единственный оставшийся способ взаимодействия заключается в обмене сообщениями через сетевой интерфейс. Таким образом, в принятом подходе воздействиями на тестируемую реализацию, или стимулами являются сообщения со стороны тестирующего узла, а ответами реализации или реакциями – сообщения со стороны мобильного узла.

Для поддержания постоянной достижимости мобильного узла по его домашнему адресу протокол Mobile IPv6 содержит ряд служебных процедур обмена сообщениями между мобильным узлом и домашним агентом или между мобильным узлом и узлом-корреспондентом, например, процедуру *регистрации на домашнем агенте* (Home Registration), процедуру *регистрации на узле-корреспонденте* (Correspondent Registration), процедуру *обратной маршрутизируемости* (Return Routability Procedure) и так далее.

Заметим, что мобильный узел является инициатором выполнения большинства служебных процедур протокола Mobile IPv6. Это создает некоторые дополнительные сложности при организации тестирования. В соответствии с требованиями стандарта RFC 3775 мобильный узел должен инициировать такие процедуры либо после смены точки подключения, либо

по истечении определенного временного интервала после какого-либо события, хотя некоторые служебные обмены сообщениями могут инициироваться мобильным узлом в любое время. Таким образом, единственное явное воздействие, которое может заставить мобильный узел начать служебные обмены сообщениями, – это смена точки подключения мобильного узла.

Для упрощения схемы тестового стенда и возможности создания для реализации мобильного узла некоторых внештатных ситуаций было принято решение об имитации процесса смены точки его подключения. При этом фактической смены точки подключения мобильного узла не происходит. Тестовый стенд состоит из двух узлов, соединенных сегментом Ethernet. Один из узлов – тестируемый узел. На тестируемом узле статически настраивается и запускается целевая реализация мобильного узла. Второй узел является инструментальным. На этом узле функционирует тестовый набор, который имитирует для тестируемой реализации некоторую виртуальную сеть.

Смена точки подключения мобильного узла имитируется при помощи вспомогательных сообщений Router Advertisement. Такие сообщения периодически отправляются тестовым сценарием от имени маршрутизатора в той сети, в которой мобильный узел находится в данный момент времени. Для имитации смены точки подключения мобильного узла тестовый сценарий прекращает отправление сообщений Router Advertisement от имени маршрутизатора в текущей сети и начинает отправление аналогичных сообщений от имени маршрутизатора в новой сети.

Для реализации описанной выше модели тестирующий узел должен имитировать функциональность всех узлов, которые принимают участие в обмене сообщениями с мобильным узлом в модели виртуальной сети, т.е. отправлять все необходимые сообщения от имени узлов виртуальной сети. При этом некоторые из таких сообщений являются частью определенных процедур и должны отправляться синхронно в ответ на сообщения от мобильного узла.

Не углубляясь в детали, поведение мобильного узла в целом можно охарактеризовать следующим образом. При смене точки подключения мобильный узел должен выполнить несколько служебных обменов сообщениями с домашним агентом и узлами-корреспондентами. В результате этих обменов будут установлены необходимые привязки, и мобильный узел станет доступным по своему домашнему адресу. После этого мобильный узел начинает служебные обмены сообщениями только перед истечением времени жизни привязки, домашнего адреса или временного адреса.

Заметим, что после перехода в новую сеть мобильный узел может выполнять некоторые из процедур обмена сообщениями параллельно, что делает его поведение недетерминированным. Например, процедура регистрации на узле-корреспонденте может выполняться параллельно с процедурой Mobile Prefix Discovery. Кроме того, стандарт допускает недетерминированное поведение

мобильного узла в рамках одной служебной процедуры. Например, перед регистрацией на узле-корреспонденте мобильный узел может выполнить процедуру обратной маршрутизируемости либо полностью, либо частично, а в некоторых случаях может ее не выполнять вообще. Некоторые сообщения тестируемая реализация должна отправлять через определенное время после поступления стимула. Из-за таких особенностей поведения мобильного узла возникают затруднения с определением группы сообщений, которые можно охарактеризовать как воздействие и отклик на него. Такая группа может охватывать несколько служебных процедур, и для нее будет сложно построить спецификации методом Design-by-Contract [10]. Для того чтобы избежать таких сложностей, используется механизм отложенных реакций инструмента CTesK, который позволяет не указывать в явном виде связь между стимулом и реакцией.

Заметим также, что инструмент CTesK, который использовался для разработки тестового набора, позволяет проводить тестирование лишь с использованием стационарных модельных состояний. Однако мобильный узел не имеет стационарных состояний за исключением некоторых внештатных ситуаций. Для преодоления этого ограничения исследовалась возможность применения двух различных подходов.

Первый подход заключался в использовании при работе обходчика состояний, близких к стационарным. Если для привязок и адресов использовать достаточно большие значения времени жизни, то состояние мобильного узла, после смены точки подключения и завершения служебных процедур, можно рассматривать как «приближенно» стационарное, которое пригодно для используемого метода тестирования. Слово «приближенно» используется потому, что такие состояния в действительности не являются стационарными. Для того чтобы мобильный узел находился в таком состоянии длительное время, он должен периодически получать от маршрутизатора сообщения Router Advertisement и выполнять процедуру Neighbor Unreachability Detection.

Для обхода графа таких состояний можно ввести группу стимулов, которые производят смену точки подключения мобильного узла. Такие стимулы будут отличаться от отдельных сообщений, отправляемых тестовым сценарием, так как будут действовать на всю виртуальную сеть, а не на мобильный узел в отдельности. При воздействии таким стимулом состояние тестируемой реализации явно не изменяется, а изменяется лишь состояние виртуальной сети в целом. Но благодаря обмену сообщениями между узлами виртуальной сети впоследствии изменится и состояние тестируемой реализации. После воздействия одним из таких стимулов мобильный узел должен через некоторое время перейти в приближенно стационарное состояние, а значит, такие стимулы уже можно использовать в инструменте CTesK.

Такой подход использовался в ранней версии тестового набора, который проверял только установление привязок с домашним агентом. Однако при расширении этого тестового набора возникли дополнительные трудности, и от

использования таких приближенно стационарных состояний пришлось отказаться. Первая проблема заключается в том, что стандарт RFC 3775 не содержит исчерпывающего описания ограничений на то, какие сообщения мобильный узел может отправлять, находясь в определенном состоянии. Более того, стандарт указывает, что некоторые служебные процедуры, такие как формирование временного адреса и регистрация на узле-корреспонденте, мобильный узел может начать в любой момент времени.

Вторая проблема возникает, когда при тестировании мобильного узла необходимо использовать небольшие значения времени жизни для привязок. Из-за недетерминированного поведения мобильного узла сложно предсказать время, когда привязка будет установлена, а значит и время, когда реализация мобильного узла должна отправить сообщение Binding Update для обновления активно используемой привязки, время жизни которой вскоре истечет. При использовании небольших значений времени жизни привязки такая реакция может возникнуть во время анализа результатов выполнения сценарной функции, что в свою очередь может привести к ошибке в работе обходчика инструмента CTesK.

В результате из-за описанных выше проблем от использования стационарных модельных состояний целевой реализации пришлось отказаться. Основные причины использования стационарного тестирования заключаются в следующем:

1. Использование стационарных состояний позволяет проверить, что тестируемая реализация действительно выполнила все необходимые действия, которые были определены спецификацией.
2. Обходчик может работать лишь со стационарными состояниями. Если же исполнение сценарной функции завершается в нестационарном состоянии, то за время, которое необходимо обходчику для начала выполнения следующей сценарной функции, система может перейти в другое состояние. Это может привести к неполному обходу графа модельных состояний или вынесению вердикта о недетерминированном поведении целевой реализации.

Второй подход заключается в отказе от стационарного тестирования. При этом необходимо искусственным образом имитировать работу обходчика для случая стационарного тестирования и следить за выполнением всех необходимых действий, которые определены спецификацией.

Чтобы проверить, что тестируемая реализация действительно отправила все необходимые сообщения, используется список действий, которые должна произвести система. Действием является отправление целевой реализацией сообщения определенного типа с определенными значениями некоторых полей; значения других его полей могут быть произвольными. Для каждого действия отводится время, за которое система должна его произвести. Действие может быть отменено при получении какого-либо стимула, если оно еще не было выполнено тестируемой реализацией.

В стандарте Mobile IPv6 [1] требования по уровню обязательности выполнения классифицированы в соответствии с RFC 2119 [11]. В предлагаемом подходе для различных групп требований предусмотрены три группы действий:

1. Обязательные действия (*obligated*) – для требований, обязывающих мобильный узел отправить определенное сообщение (требования, выделенные словами MUST или SHOULD согласно RFC2119). Если тестируемая реализация отправляет такое сообщение в отведенный временной интервал, то сообщается о том, что ее поведение в данном случае удовлетворяет требованию спецификации, которое определяет данное действие. В противном случае выносится вердикт о том, что тестируемая реализация не выполнила данное требование. Для этой группы требований можно задавать альтернативные варианты действий. То есть задаются несколько возможных сообщений для одного требования, и если реализация отправляет одно из таких сообщений, условия требования считаются выполненными.
2. Запрещенные действия (*prohibited*) – для требований, запрещающих мобильному узлу отправлять определенное сообщение после наступления какого-либо события (требования, выделенные словами MUST NOT или SHOULD NOT). Если данное сообщение было отправлено в выделенный временной интервал, то выносится вердикт о том, что тестируемая реализация нарушила данное требование. В противном случае сообщается о том, что она выполнила требование, определяющее данное действие.
3. Допустимые действия (*allowed*) – для наблюдения за тем, что мобильный узел реализует некоторую необязательную функциональность протокола (требования, выделенные словом MAY). Если данное сообщение отправлено в отведенный интервал времени, то сообщается о том, что тестируемая реализация поддерживает необязательную функциональность, которая определяется данным требованием. В противном случае ничего не сообщается.

Такой подход позволяет явно описывать требования, касающиеся отправления сообщений, и собирать детальную информацию о проверке таких требований, однако предполагает затрату бо́льших усилий при разработке спецификаций, чем в случае со стационарными модельными состояниями.

Для того чтобы обеспечить возможность использования стандартного обходчика, в качестве модельных состояний выделяются состояния виртуальной сети в целом, а не группы состояний модели целевой реализации, как это традиционно делается при использовании инструмента CTesK. С учетом того, что в работе имитируются только виртуальные сети со стационарными маршрутизаторами, домашними агентами и узлами-корреспондентами, состояние виртуальной сети определяется лишь ее топологией, набором ее узлов и точкой подключения мобильного узла.

Топология виртуальной сети и набор узлов задаются перед запуском тестового сценария. Таким образом, состояние виртуальной сети определяется лишь точкой подключения мобильного узла.

Такие модельные состояния являются стационарными, так как не зависят от поведения реализации и являются объединениями нескольких приближенно стационарных состояний мобильного узла. Для обеспечения такого соответствия необходимо использовать достаточно большой интервал времени, который отводится на исполнение сценарной функции, чтобы тестируемая реализация успела осуществить переход в приближенно стационарное состояние. Кроме того, необходимо описать все действия, определенные стандартом при данном переходе.

Конечное приближенно стационарное состояние мобильного узла после перехода в один и тот же сегмент виртуальной сети зависит от поведения других узлов этой сети. В результате введения описанных выше состояний может существенно ухудшиться качество тестирования, поэтому помимо текущей точки подключения мобильного узла в модельных состояниях должен учитываться также и способ, которым мобильный узел был переведен в данную сеть. Сеть, в которой мобильный узел находился до осуществления смены точки подключения, в модельном состоянии не учитывается, так как для любого способа, которым мобильный узел может быть переведен в данную сеть, при обходе графа модельных состояний автоматически будут произведены переходы из всех сетей, из которых это возможно.

В предлагаемом подходе не учитывается также и предыдущая история переходов. В подавляющем большинстве случаев предыдущие переходы не оказывают влияния на поведения мобильного узла, а учет таких переходов экспоненциально увеличивает количество модельных состояний.

В качестве сценарных функций в тестовом наборе задаются возможные способы перевода мобильного узла из одной сети в другую. Задача обходчика в таком случае заключается в осуществлении всех возможных переходов для мобильного узла всеми возможными способами. Под различными способами здесь понимается различное поведение узлов виртуальной сети после осуществления смены точки подключения мобильного узла в этой сети, которое определяется отправляемыми ими сообщениями.

Для обеспечения полноты тестирования необходимо оказать воздействие всеми возможными стимулами во всех достижимых состояниях, включая начальное состояние. Однако приведение тестируемой реализации мобильного узла в начальное состояние только при помощи обмена сообщениями через сетевой интерфейс не представляется возможным. Это связано с тем, что даже после удаления привязки при возвращении в домашнюю сеть мобильный узел сохраняет некоторую информацию в структурах данных, описывающих его состояние, например, сохраняет адрес домашнего агента. На Рис. 7 изображен пример графа состояний тестового сценария для случая, когда виртуальная сеть состоит только из сегмента

домашней сети и одного сегмента внешней сети, и определены только два способа перевода мобильного узла между этими сетями. В состояниях 0, 1 и 2 мобильный узел находится в домашней сети, в состояниях 3 и 4 – во внешней сети. Чтобы отличить способы перевода мобильного узла на рисунке используется разная толщина дуг графа.

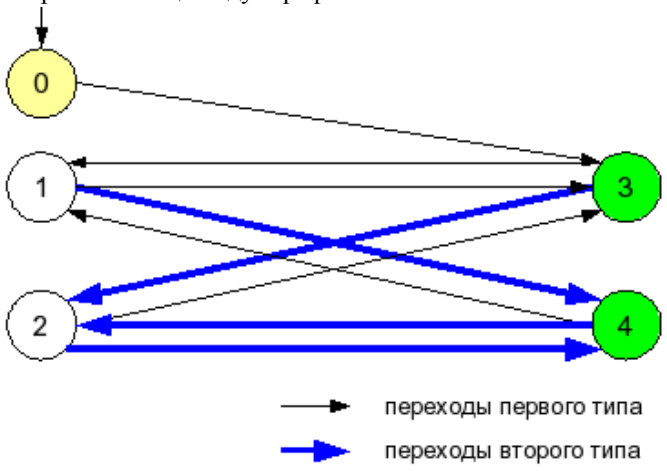

Рис. 7. Пример графа состояний тестового сценария

Этот пример показывает, что граф состояний тестового сценария не является сильно связным, так как нет дуги, которая возвращала бы тестовый сценарий в начальное состояние. Однако такая дуга необходима для осуществления обходчиком всех возможных переводов мобильного узла из всех достижимых состояний.

Это одна из проблем, возникающих при использовании состояний, принятых в нашем подходе. Для гарантированного приведения тестируемой реализации в первоначальное состояние перед выполнением очередного тестового сценария используется перезагрузка целевого узла. Чтобы решить проблему с несвязностью графа состояний, можно было бы ввести стимул, осуществляющий перезагрузку системы, но это приводило бы к большому количеству перезагрузок во время тестирования и заняло бы больше времени на исполнение теста. Кроме того, сложно было бы отследить, чем именно вызвана перезагрузка: нестабильностью тестируемой реализации или воздействием такого стимула.

Вместо введения стимула, осуществляющего перезагрузку целевой реализации, в предлагаемом подходе используются два приема: объединение состояний и обеспечение всех возможных воздействий. А именно, начальное состояние мобильного узла объединяется с состоянием «мобильный узел в домашней сети», в которое он попадает после возвращения в домашнюю сеть,

при условии, что тестовый сценарий отправит все необходимые сообщения от имени узлов виртуальной сети. Для обеспечения же перебора всех возможных воздействий в начальном состоянии первоначальные тестовые сценарии разделяются на несколько подсценариев. Этот процесс можно описать следующим образом.

Сначала, как и обычно, для проверки определенных групп требований стандарта разрабатываются конкретные тестовые сценарии. Затем после определения всех возможных стимулов, необходимых для обеспечения полноты тестирования, для каждого из этих первоначально разработанных тестовых сценариев создается несколько подсценариев, которые отличаются лишь тем, что на целевую реализацию, находящуюся в первоначальном состоянии, осуществляется воздействие одним конкретным стимулом, отличным от первых стимулов в других подсценариях. При этом общее количество создаваемых подсценариев равно количеству возможных стимулов для данного тестового сценария. Таким образом, на реализацию, находящуюся в первоначальном состоянии, осуществляется воздействие всеми возможными стимулами.

На Рис. 8 изображены подсценарии, полученные из предыдущего примера при помощи описанных выше действий. Заметим, что у этого графа дуга из состояния 1 в состояние 3, соответствующая первому способу перевода мобильного узла из домашней сети во внешнюю сеть, обозначена пунктиром. Это означает, что обходчик не обязан осуществлять такое воздействие, так как может считать, что оно уже произведено при выполнении перевода мобильного узла из начального состояния первым способом. По этой же причине во втором подсценарии выделена пунктиром дуга из состояния 1 в состояние 4, соответствующая второму способу перевода мобильного узла из домашней сети во внешнюю сеть. Из-за подобных особенностей полученных подсценариев в случае, когда первоначальный тестовый сценарий включает в себя лишь одно воздействие, и, соответственно, имеется только один подсценарий, тестирование может оказаться неполным. Однако, как это видно из Рис. 8, при наличии двух и большего числа подсценариев необходимые воздействия будут гарантированно осуществляться в других подсценариях.

Таким образом, для обеспечения полноты тестирования необходимо, чтобы в каждом из начальных сценариев использовалось как минимум два стимула. Следует отметить, что обходчики dfsm [8] и ndfsm [9] инструмента CTesK, благодаря своему внутреннему устройству, всегда осуществляют воздействие, отмеченное на Рис. 8 пунктирной линией, поэтому на практике можно использовать тестовые сценарии, включающие в себя лишь один стимул.

В результате в разработанном тестовом наборе используются две группы состояний: состояния виртуальной сети, дополненные указанием способа, которым был произведен последний перевод мобильного узла, и состояния модели тестируемой реализации. Соответственно, имеются две группы стимулов: стимулы, осуществляющие смену точки подключения мобильного

узла в виртуальной сети, которые изменяют только состояние виртуальной сети, и сообщения от узлов виртуальной сети, которые изменяют только состояние модели целевой реализации.

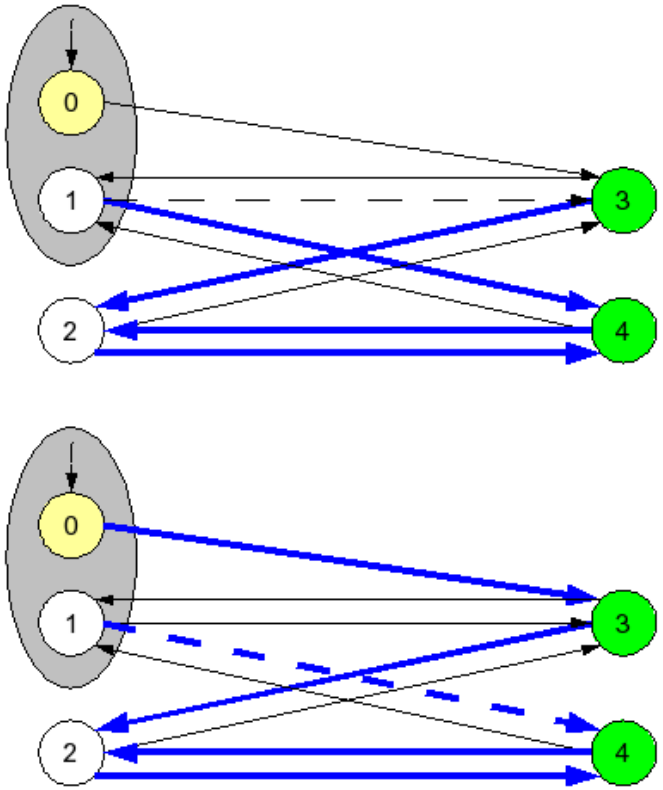

Рис. 8. Полученные тестовые подсценарии

Реакциями являются сообщения от тестируемой реализации, которые также изменяют только состояние модели целевой реализации. Реакции со стороны виртуальной сети в данном тестовом наборе не предусмотрены. Спецификации разрабатываются только для группы стимулов и реакций, которые работают с состоянием модели целевой реализации.

Спецификации состоят из постусловий стимулов и реакций, а также функции, которая следит за истечением временного интервала, отведенного на действия. Последняя функция вызывается в начале работы медиаторов для стимулов и реакций, описывающих сетевые сообщения. В качестве аргумента этой функции передается время, когда пакет был получен на сетевом интерфейсе;

на основании этого времени и происходит синхронизация списка действий, то есть удаление действий, время которых истекло.

5. Структура тестового набора и этапы исполнения сценарной функции

Как было указано в предыдущем разделе, для реализации предложенной модели тестирования, тестирующий узел должен имитировать функциональность всех узлов, которые принимают участие в обмене сообщениями с мобильным узлом в виртуальной сети. Имитация функциональности узлов виртуальной сети необходима на протяжении всего выполнения тестового сценария, в том числе и во время обработки результатов вызова сценарных функций. Такая необходимость вызвана тем, что при тестировании используются нестационарные модельные состояния, и большинство служебных сообщений, отправляемых мобильным узлом, является частью синхронных процедур обмена сообщениями с одним из узлов сети. Поэтому часть тестового набора, имитирующая данную функциональность, должна быть реализована вне тестового сценария.

Схема виртуальной сети и набор ее узлов специфичны для каждого теста. С учетом этого наиболее простое и логичное решение заключается в том, чтобы обособить функциональность, свойственную каждому узлу. Для этого написаны отдельные демоны, каждый из которых полностью имитирует функциональность узла определенного типа: демон маршрутизатора, демон домашнего агента, демон узла-корреспондента.

Демоны запускаются в отдельных процессах в начале тестового сценария и функционируют на протяжении всего его выполнения. Это позволяет мобильному узлу отправлять необходимые вспомогательные сообщения во время обработки стимулов и реакций, собранных тестовым сценарием между вызовами сценарных функций. При переходе исполнения от одной сценарной функции к другой изменяется лишь поведение демонов, которое заключается в отправлении или не отправлении синхронных и асинхронных сообщений мобильному узлу, меняются также некоторые поля в этих сообщениях. В процессе исполнения тестового сценария топология виртуальной сети и положение в ней демонов остаются неизменными.

Все демоны подключаются к одному и тому же сетевому интерфейсу; это упрощает организацию маршрутизации и внештатного функционирования демонов, которое необходимо для проверки некоторых требований спецификации. Взаимодействие между демонами, там, где это необходимо, должно производится не через сетевой интерфейс, а с использованием других средств, так как все такие сообщения будет получать мобильный узел, если его сетевой интерфейс находится в режиме *promiscuous*. В предлагаемом тестовом наборе такое взаимодействие необходимо только между демонами узлов-корреспондентов и демонами домашних агентов. Для этого

используется общий Binding Update List, в который домашние агенты записывают информацию об установленных привязках. Узлы-корреспонденты используют эту информацию для извлечения из туннеля входящих пакетов и туннелирования исходящих пакетов при обмене сообщениями с мобильным узлом в режиме двустороннего туннелирования.

Общая картина выполнения тестового сценария выглядит следующим образом (см. Рис. 9). Перед инициализацией тестового сценария на тестируемом узле должна быть настроена и запущена целевая реализация. Перед началом тестирования запускается процесс под названием *кэтчер*, который получает и запоминает все сообщения, поступающие на сетевой интерфейс. Затем запускаются необходимые демоны с первоначальными значениями структур данных, которые определяют их поведение. После этого тестовый сценарий выжидает некоторый интервал времени, в течение которого при помощи обмена сообщениями с демонами тестируемая реализация выполняет автоматическую конфигурацию.

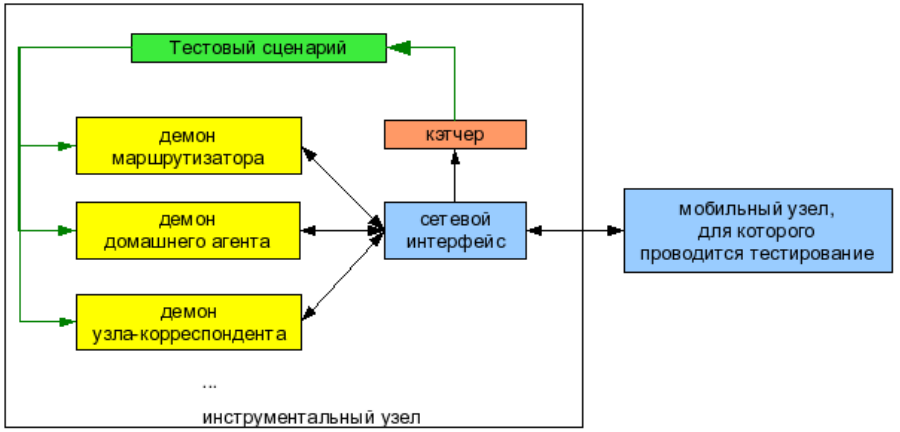

Рис. 9. Схема исполнения тестового сценария

Выполнение тестового сценария заключается в последовательном вызове сценарных функций. При вызове каждой сценарной функции соответствующим образом изменяются внутренние структуры данных, определяющие поведение демонов. Такое изменение состояния и поведения демонов оказывает воздействие на целевую реализацию мобильного узла и заставляет ее выйти из стационарного состояния. Затем тестовый сценарий некоторое время находится в состоянии ожидания, по истечении которого работа сценарной функции завершается, и тестовый сценарий получает от кэтчера пакеты, собранные за время ожидания и обработки результатов вызова предыдущей сценарной функции. Далее производится анализ полученных пакетов и регистрация соответствующих стимулов и реакций.

Таким образом, стимулы, которые должны применяться в данной сценарной функции, задаются неявным образом через поведение демонов. После завершения сценария прекращается работа демонов и кэтчера.

6. Адаптация средств инструмента CTesK к предложенной схеме тестового набора

В используемом тестовом стенде у инструментального и целевого узла имеется по одному сетевому интерфейсу, которые соединены кросс-кабелем, поэтому пакеты в процессе доставки не перемешиваются. Поскольку в большинстве случаев обмены сообщениями между мобильным узлом и демонами являются синхронными, порядок поступления сообщений может изменяться лишь в том случае, если два сообщения отправляются одним и тем же узлом и являются частью разных параллельно выполняющихся процедур обмена сообщениями. Стандарт RFC 3775 не накладывает никаких ограничений на связь между параллельно выполняющимися процедурами, поэтому такое нарушение порядка не может повлиять на выносимый тестовым сценарием вердикт. Это позволяет в разрабатываемом тестовом наборе не использовать сериализацию. Точнее, при сериализации рассматривается лишь один порядок поступления стимулов и реакций.

В выбранной модели тестирования используются две группы стимулов: стимулы, действующие на виртуальную сеть, и стимулы, действующие на тестируемую реализацию. Сбор пакетов не приостанавливается между вызовами сценарных функций, поэтому пакеты, собранные за время обработки результатов вызова сценарной функции, будут рассматриваться после вызова следующей сценарной функции. Это означает, что отключение сериализации приводит к нарушению порядка поступления стимулов. Однако порядок вызова стимулов, воздействующих на тестируемую реализацию, будет сохраняться, а стимулы, воздействующие на сеть, не изменяют структур данных, имитирующих состояние тестируемой реализации. Поэтому такое нарушение порядка поступления стимулов не повлияет на вердикт, выносимый спецификациями.

Стандарт RFC 3775 представляет собой обычный текст, из которого можно выделить несколько сотен четких требований. При построении формальных спецификаций в виде пред- и постусловий для сообщений от тестируемой реализации может оказаться, что некоторые постусловия содержат проверку для десятков требований стандарта. Инструмент CTesK в случае нарушения одного из таких требований выдаст немногословный вердикт *false*. Однако по сообщению, на котором был выдан такой вердикт, не всегда просто понять, какое же именно требование стандарта было нарушено. Чтобы решить эту проблему, введена вспомогательная функция conformance_check, осуществляющая проверку требования стандарта и сбрасывающая в тестовую трассу информацию о результате его проверки, а также о ситуации, в которой

эта проверка производилась. По такой информации впоследствии генерируется детальный отчет о проверенных требованиях стандарта.

Элементарной реакцией от целевой системы в рассматриваемой модели тестирования является пакет, но многие требования стандарта применяются не ко всему пакету, а к отдельным его заголовкам. Проверку таких требований в инструменте CTesK принято осуществлять в инвариантах типов данных, описывающих заголовки пакета. Однако в тестовом наборе используются одни и те же типы данных для заголовков, содержащихся в пакетах от мобильного узла и в пакетах от демонов. Проверка же должна производится только для пакетов от мобильного узла, так как некорректно построенные заголовки в пакетах от демонов допускаются и используются при тестировании внештатных ситуаций. Поэтому проверка таких требований вынесена за пределы спецификационной функции и происходит еще до регистрации реакций на этапе разбора пакета по заголовкам.

Для идентификации пакета в отчете удобно использовать время, когда был получен пакет. Время получения передается стимулам и реакциям вместе с другой информацией о пакете через аргумент спецификационной функции или реакции соответственно. Однако инструмент CTesK не позволяет использовать в предусловии значение аргумента передаваемого реакции, поэтому проверку всех требований пришлось вынести в постусловие реакций.

Следует также отметить, что отчеты, которые может генерировать инструмент CTesK, в данной работе не используются. Вместо этого, отчет о проведенном тестировании генерируется отдельной частью тестового набора по тестовой трассе и нескольким вспомогательным файлам. Такие отчеты ориентированы на конкретную задачу, более наглядны и понятны пользователю, не знакомому с технологией UniTESK.

7. Апробация разработанного тестового набора

При помощи разработанного тестового набора было проведено тестирование открытой реализации Mobile IPv6 KAME для операционной системы FreeBSD 5.4 (CVS Snapshot 2006.11.13). Во время проведения тестирования эта реализация была одной из наиболее распространенных открытых реализаций Mobile IPv6.

При прогоне тестовых сценариев был выявлен ряд нарушений требований стандарта RFC 3775 [1], например:

- мобильный узел выполняет процедуру Duplicate Address Detection не во всех случаях, когда это необходимо;
- мобильный узел при смене точки подключения не всегда производит регистрацию основного временного адреса на домашнем агенте;
- мобильный узел нарушает ограничения темпа отправления сообщений Binding Update;
- мобильный узел не соблюдает некоторые рекомендации процедуры

Generic Movement Detection;
- мобильный узел не всегда использует корректные значения для времени ожидания ответа на такие сообщения, как Binding Update, Home Test Init, Care-of Test Init, Mobile Prefix Solicitation.

Помимо этого, в некоторых случаях поведение тестируемого мобильного узла было заведомо некорректным, например, некоторые сообщения отправлялись на неправильный адрес канального уровня. Кроме того, исполнение некоторых тестовых сценариев приводило к зависанию или перезагрузке мобильного узла, что свидетельствует о нестабильности целевой реализации. Результаты проведенного тестирования с указанием нарушенных требований стандарта доступны по адресу *http://ipv6.ispras.ru/mobility/mn/kame.html*.

8. Заключение

В статье описаны особенности тестового набора, разработанного для проверки соответствия реализаций мобильного узла спецификации протокола Mobile IPv6 [1]. Тестовый набор разрабатывался при помощи инструмента CTesK, который реализует технологию автоматического тестирования UniTESK. В ходе выполнения работы были выявлены некоторые особенности мобильного узла, которые затрудняют его тестирование с помощью технологии UniTESK, а именно:

- Мобильный узел является инициатором большинства служебных обменов сообщениями. Некоторые из таких обменов сообщениями можно вызвать неявным образом, создав определенные условия при помощи вспомогательных сообщений. Другие обмены сообщениями мобильный узел может начинать в произвольные моменты времени. В обоих случаях на сообщения со стороны мобильного узла может потребоваться быстрый ответ со стороны тестирующего узла.
- Спецификация протокола Mobile IPv6 не содержит исчерпывающего описания поведения мобильного узла, поэтому нельзя осуществить перебор всех возможных воздействий во всех состояниях.
- Спецификация протокола Mobile IPv6 допускает недетерминированное поведение мобильного узла.

Однако все эти особенности удалось успешно преодолеть, не выходя за рамки ограничений технологии UniTESK.

При разработке тестового набора было обнаружено несколько сложных моментов, связанных с применением инструмента CTesK, а именно:

- отсутствие у мобильного узла стационарных состояний;
- сложность использования инвариантов типов данных из-за того, что их проверка производится только автоматически и не может контролироваться;
- недостаточная информативность при задании формальных спецификаций в виде пред- и постусловий.

Для преодоления этих трудностей потребовалось несколько отступить от прямого использования традиционного подхода к разработке тестового набора, задаваемого инструментом CTesK. Тем не менее, благодаря гибкости инструмента, удалось решить поставленную задачу его средствами. Единственное, от чего действительно пришлось отказаться, – это использование стандартных для CTesK средств генерации отчетов о тестировании. Широта и сложность предметной области требует для анализа результатов тестирования существенно большего объема информации, чем предоставляет инструмент. Поэтому был реализован сбор дополнительной информации о ходе тестирования средствами самого тестового набора и разработана система генерации отчета, предназначенная для конкретной задачи. В остальном, несмотря на все выявленные особенности, удалось без существенных трудностей использовать инструмент CTesK для тестирования реализаций мобильного узла.

Литература

[1] D. Johnson, C. Perkins, J. Arkko "Mobility Support in IPv6", RFC 3775, June 2004.
[2] http://www.unitesk.com - сайт, посвященный технологии тестирования UniTESK и реализующим ее инструментам.
[3] http://www.unitesk.com/products/ctesk/ – страница инструмента CTesK.
[4] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
[5] S. Thomson, T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.
[6] T. Narten, T. Nordmark, E. and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.
[7] A. Conta, S. Deering "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.
[8] И. Б. Бурдонов, А. С. Косачев, В. В. Кулямин. Неизбыточные алгоритмы обхода ориентированных графов: детерминированный случай. Программирование, т. 29, № 5, стр. 59-69, 2003.
[9] А. В. Хорошилов. Отчет о научно-исследовательской работе. Алгоритмы обхода недетерминированных графов, обладающих детерминированным полным остовным подграфом.
[10] Bertrand Meyer. Object-Oriented Software Construction, Second Edition. Prentice Hall, 1997.
[11] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.