Применение формальных методов для тестирования реализации IPv6

Г.В. Ключников, А. С. Косачев, Н.В. Пакулин, А. К. Петренко, В.З. Шнитман

Аннотация. В статье представлен опыт разработки тестового набора для реализации протокола IPv6. Для разработки тестового набора использовался метод разработки тестовых наборов на основе формальных спецификаций UniTesK, развиваемый в Институте системного программирования РАН. В качестве объекта тестирования была выбрана реализация IPv6 от Microsoft Research. В статье подробно описывается устройство полученного тестового набора и обсуждаются результаты проекта.

1. Введение

В статье представлен опыт разработки тестового набора для реализации протокола IPv6 на Windows 2000. Тестовый набор предназначался для проверки соответствия реализации протокола IPv6 спецификациям IPv6. Проект по разработке тестового набора проходил при поддержке исследовательского гранта Microsoft Research.

Разработка тестового набора проводилась с использованием методологии тестирования UniTesK, которая разработана и развивается в Институте системного программирования РАН. В качестве объекта тестирования была выбрана MSR IPv6 – реализация IPv6, разработанная в Microsoft Research.

Перед тестовым набором стояла задача проверить, насколько реализация протокола соответствует стандарту протокола. Кроме того, перед разработчиками тестового набора были поставлены следующие цели:

- Продемонстрировать применимость формальных методов для специфицирования и тестирования сложного API, такого, как реализация IPv6
- Исследовать применимость UniTesK к специфицированию и тестированию внутренних подсистем Windows NT или приложений для Windows NT
- Исследовать применимость UniTesK к тестированию индустриального программного обеспечения

Статья построена следующим образом. Раздел 2 содержит краткое введение в IPv6 и описание функций IPv6, которые мы тестировали. В разделе 3 указаны общие сведения о MSR IPv6 и обсуждаются особенности MSR IPv6 в контексте

тестирования. В разделе 4 дано общее описание метода разработки тестовых наборов UniTesK. Раздел 5 посвящен применению UniTesK для тестирования систем с отложенными реакциями. В разделе 6 дано описание устройства тестового набора для MSR IPv6, в разделе 7 перечисляются результаты проекта по разработке тестового набора. Раздел 8 – Заключение.

2. Введение в IPv6.

IPv6 – сетевой протокол нового поколения. Он призван заменить существующий протокол сетевого уровня IPv4. IPv6 содержит ряд усовершенствований по сравнению с IPv4 и разрешает проблемы, которые ограничивают дальнейшее развитие сетей на основе IPv4.

По эталонной модели OSI [1] протокол IPv6 относится к протоколам сетевого уровня. На этом уровне происходит маршрутизация пакетов на основе преобразования сетевых адресов в адреса канального уровня. Сетевой уровень обеспечивает прозрачную передачу данных между транспортным уровнем и канальным уровнем.

2.1. Функции IPv6, выбранные для тестирования

Для тестирования мы выбрали минимальное подмножество функций IPv6, которое обеспечивает корректное функционирование оконечного узла (host).

2.1.1. Отправка и получение пакетов.

Протокол обеспечивает обмен данными между протоколами более высокого уровня. Также протокол производит обмен служебными сообщениями сетевого уровня между узлами. Формат пакетов IPv6 и правила обработки пакетов определены в RFC 2460 [2].

При отправке пакета протокола верхнего уровня IPv6 формирует свой собственный пакет, включает в него заданный пакет верхнего уровня и передает полученный пакет IPv6 в сеть. При тестировании мы проверяем, что реализация формирует корректные пакеты IPv6, и данные протокола верхнего уровня передаются в сеть без искажений.

При получении пакета из сети реализация IPv6 разбирает пакет. В том случае, если в пакете содержатся ошибки, то реализация отбрасывает пакет или отправляет источнику пакета сообщение об ошибке. Если же пакет корректен, то реализация определяет протокол верхнего уровня, которому адресованы данные, содержащиеся в пакете, и передает их получателю. При тестировании мы проверяем поведение реализации как на корректных пакетах, так и на некорректных пакетах.

2.1.2. Фрагментация и сборка пакетов.

Большинство протоколов канального уровня накладывают ограничения на размер передаваемых данных. Так, в сети Ethernet максимальный размер

пакета сетевого уровня, который передается в одном кадре Ethernet, равен 1500 байт[3].

Для передачи больш*и*х массивов данных в IPv6 предусмотрена функция фрагментации пакетов. Если размер пакета протокола верхнего уровня превосходит максимальный допустимый размер, то реализация IPv6 пересылает такой пакет по частям.

Реализация нарезает данные на фрагменты и отправляет каждый фрагмент в отдельном пакете. В точке назначения фрагменты накапливаются, и после получения всех фрагментов из них восстанавливается первоначальный пакет. Фрагментация и сборка пакетов происходит прозрачно для протоколов верхнего уровня.

При тестировании мы проверяли, что реализация протокола корректно строит фрагменты, корректно восстанавливает данные из последовательности фрагментов, а также надежно обрабатывает ошибки во фрагментах и сериях фрагментов.

Именно в функции сборки фрагментов мы обнаружили наиболее серьезный дефект в MSR IPv6, который приводит к перезагрузке операционной системы.

2.1.3. ICMPv6

ICMPv6 – это протокол, который используется для передачи служебных сообщений на сетевом уровне. В частности, формат пакетов ICMPv6 используются в протоколах Neighbor Discovery (см. далее). Протокол ICMPv6 специфицирован в RFC 2463 [4].

Кроме того, ICMPv6 используется в некоторых протоколах верхнего уровня (например, UDP) для передачи сообщений об ошибках.

2.1.4. ICMPv6 Echo

ICMPv6 Echo – это простой диагностический протокол, который используется для определения пропускной способности сети и состояния удаленного узла IPv6. В частности, данный протокол используется в утилите ping. Протокол ICMPv6 Echo является неотъемлемой частью ICMPv6 и определен также в RFC 2463 [4].

Протокол использует формат пакетов ICMPv6. Всего в протоколе два вида сообщений – Echo Request и Echo Reply. При получении Echo Request узел IPv6 должен выслать ответ, Echo Reply, тому узлу, от которого пришел запрос. Таким образом, узел – источник запроса – может установить, что адресат запроса достижим. Если же узел-источник не получит ответа, то либо узел-адресат не функционирует, либо сеть перегружена и не в состоянии доставить пакет адресату.

При тестировании мы проверяли, что реализация корректно строит пакеты-запросы Echo Request, и выдает корректные ответы (Echo Replies) на корректные запросы Echo Request.

2.1.5. Neighbor Discovery

Neighbor Discovery (ND) объединяет группу протоколов, которые решают следующие задачи:

- Определение адресов канального уровня для узлов, которые располагаются на одном сегменте локальной сети (*соседей*)
- Определение маршрутизаторов, которые подсоединены к сегменту локальной сети
- Определение того, через какой маршрутизатор следует отправлять пакеты, адресованные узлам за пределами сегмента локальной сети

Для сбора информации о соседних узлах узел IPv6 отправляет и получает служебные пакеты. Всего в ND на данный момент насчитывается пять видов служебных пакетов. Во всех случаях используется формат пакетов ICMPv6. Протокол ND задан в RFC 2461 [5]

Для уменьшения нагрузки на сеть собранная информация сохраняется узлом, поэтому на протоколы ND возлагается задача своевременно обновлять сведения соседних узлах и удалять устаревшие записи.

При тестировании мы проверяли, что реализация правильно обновляет информацию о соседях, правильно выбирает маршрутизаторы для отправки пакетов за пределы локальной сети, правильно сообщает информацию о себе соседям.

3. MSR IPv6

MSR IPv6 – это свободно распространяемая открытая реализация IPv6 для Windows NT/Windows 2000. Как заявляют разработчики, назначение MSR IPv6 состоит в следующем:

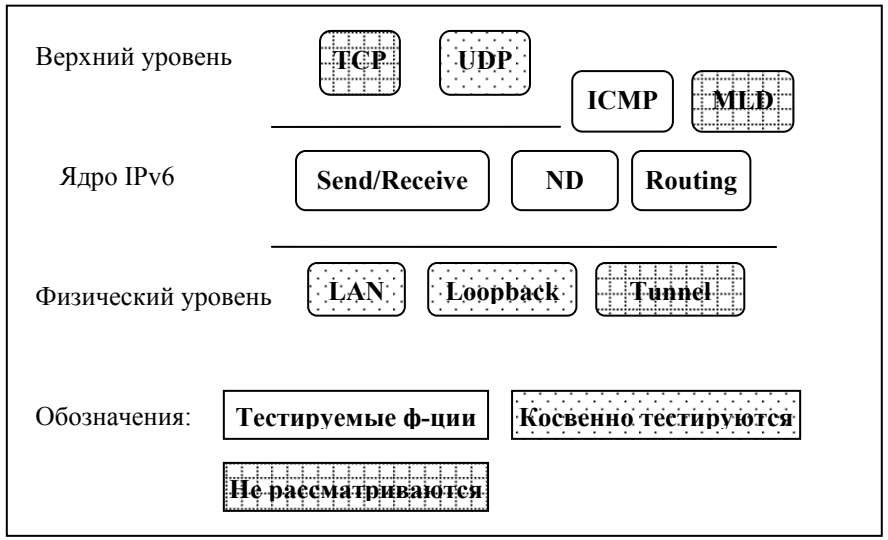
1. Бесплатная общедоступная реализация IPv6
2. Пример реализации протокола сетевого уровня для Windows NT
3. Исследования протоколов IPv6

В MSR IPv6 реализованы основные функции оконечного узла, описанные в предыдущем разделе, а также ряд базовых функций маршрутизаторов IPv6. Кроме того, MSR IPv6 содержит собственную реализацию транспортных протоколов TCP и UDP и интерфейс сокетов для доступа отправки и получения пакетов в сети IPv6. MSR IPv6 может устанавливаться параллельно стеку IPv4 без каких бы то ни было последствий для работоспособности последнего.

Более подробную информацию о MSR IPv6 можно получить на сайте Microsoft Research [6] и в работе [7].

3.1. Разбиение MSR IPv6 на подсистемы

MSR IPv6 отличается хорошей структуризацией исходных текстов. Код, который реализует некоторую функцию IPv6, как правило, сгруппирован в отдельные файлы, и может рассматриваться как отдельная подсистема MSR



IPv6. На Рис. 1 представлено разбиение MSR IPv6 на подсистемы [7].

Рис. 1. *Разбиение MSR IPv6 на подсистемы.*

В средней части рисунка представлены подсистемы, которые реализуют базовые функции IPv6 – отправка и получение пакетов, ND, ICMPv6. Прямоугольник, помеченный как “Routing”, реализует часть функций Neighbor Discovery, которые относятся к работе с маршрутизаторами.

В верхней части рисунка представлены транспортные протоколы, реализации которых имеются в MSR IPv6 – TCP и UDP. В нижней части рисунка расположены подсистемы, которые обеспечивают взаимодействие IPv6 с протоколами нижнего уровня (LAN) и псевдоустройством закольцованной связи (loopback).

3.2. Особенности тестирования MSR IPv6

Рассмотрим особенности, которые отличают тестирование MSR IPv6 от тестирования обычных программных интерфейсов.

Отложенные реакции. Целевая система демонстрирует реакции, которые происходят спустя некоторое время после воздействия извне, причем это реакции нескольких видов.

Непроцедурные стимулы. Целевая система допускает как процедурные, так и непроцедурные воздействия. Процедурные воздействия – вызов процедур, которые входят в программный интерфейс. Непроцедурные воздействия – входящие пакеты IPv6, которые целевая система получает из сети.

Доступ к процедурным стимулам. Реализация MSR IPv6 расположена по большей части в ядре Windows 2000, поэтому программный интерфейс MSR IPv6 напрямую не доступен. Для обращения к процедурам, которые экспортирует модуль, расположенный в ядре, приходится пользоваться специальными средствами операционной системы.

Недетерминизм реакций. Отложенные реакции, вообще говоря, недетерминированы. При одном и том же внешнем воздействии целевая система может демонстрировать различные отложенные реакции. Это связано с тем, что реакции системы определяются составляющими её внутреннего состояния, которые невозможно определить извне, а также с тем, что в ряде алгоритмов IPv6 используются случайные числа.

Рассмотрим теперь, какие средства предлагает UniTesK для решения выявленных проблем.

Для тестирования систем с отложенными реакциями в UniTesK разработаны методы разработки спецификаций и тестовых сценариев, основанные на формализме конечных автоматов с отложенными реакциями.

Механизм медиаторов позволяет отделить модель стимулов от того, как в действительности происходит воздействие на целевую систему. С точки зрения модели не играет никакой роли то, как осуществляется воздействие – путем вызова процедуры, обращения к модулю ядра, отправки пакета в целевой компьютер извне. Конкретное воздействие определяется соответствующим медиатором.

В подходе UniTesK используются имплицитные спецификации. Такие спецификации описывают, какие реакции целевой системы и какие изменения в состоянии целевой системы являются допустимыми. Спецификации не вычисляют реакции и конечное состояние целевой системы, а лишь проверяют условия допустимости. Это позволяет описывать недетерминизм поведения целевой системы.

4. Введение в UniTesK

Данный раздел содержит очень поверхностное введение в UniTesK. Более подробное описание можно найти в [8].

С 1994 года в ИСП РАН активно разрабатываются методы и инструменты тестирования программного обеспечения на основе формальных методов.

В 1994-1999 годах по контрактам с Nortel Networks в ИСП РАН был разработан и активно использовался метод тестирования KVEST [9, 10]. KVEST отличался от обычных методов тестирования индустриального программного обеспечения тем, что в KVEST использовались формальные спецификации в форме пред- и постусловий для построения тестовых оракулов, а также модель конечных автоматов для построения тестовых воздействий.

Применение KVEST показало, что формальные методы можно с большим успехом применять при тестировании индустриального программного обеспечения.

Опыт применения KVEST показал, что использование академических языков формальных спецификаций и специальных языков описания тестов препятствует встраиванию инструментов, основанных на этих языках, в процесс разработки ПО. Чем ближе язык спецификаций к языку, на котором ведется разработка, тем проще разработчикам писать спецификации и тесты [10].

UniTesK разрабатывался на основе опыта, полученного при разработке и применении KVEST. Рассмотрим основные особенности UniTesK:

- Разделение построения тестовых воздействий и проверки правильности поведения целевой системы. Тестовые воздействия строятся в *тестовых сценариях,* а проверка правильности поведения целевой системы в *тестовых оракулах*.
- Автоматизированное построение тестовых воздействий.
- Представление функциональных требований к целевой системе в виде формальных спецификаций.
- Для записи формальных спецификаций используется язык, «близкий» к языку, на котором разработана целевая система.
- Автоматическая генерация тестовых оракулов из спецификаций.
- Оракулы и реализация связаны посредством тонкой прослойки *медиаторов.*
- Язык описания тестовых воздействий «близок» к языку, на котором разработана целевая система.
- Автоматически генерируются критерии качества покрытия требований
- Автоматически производится оценка качества покрытия требований при прогоне тестов.

4.1. Оракулы и спецификации

Оракулы – это процедуры, которые проверяют выполнение ограничений, наложенных на поведение целевой системы. В UniTesK оракулы полностью автоматически генерируются из описаний ограничений.

В UniTesK ограничения описываются преимущественно в форме *имплицитных спецификаций* стимулов и реакций. Кроме того, часть ограничений представляется в виде ограничений на значения типов (*инварианты типов*) и ограничений на значения глобальных переменных (*инварианты глобальных переменных*).

Имплицитные спецификации состоят из пред- и пост- условий. Предусловие содержит требования к тому, в каком состоянии и с какими параметрами можно оказывать воздействие на целевую систему. После воздействия целевая система может продемонстрировать реакции и/или перейти в другое состояние. Постусловие определяет, допустимо ли изменение состояния и продемонстрированное поведение целевой системы.

Состояние целевой системы моделируется набором структур данных, которые получили название *абстрактное состояние*. Пред- и постусловия используют информацию, содержащуюся в абстрактном состоянии, для вынесения вердикта.

4.2. Тестовые сценарии

Тестовый сценарий в UniTesK определяет последовательность воздействий, которые оказываются на целевую систему.

В UniTesK в качестве теоретической основы для построения тестового сценария выбрана модель конечных автоматов.

Тестовый сценарий обладает собственным состоянием, которое, как правило, вычисляется на основе состояния целевой системы. В каждом состоянии задаются воздействия, которые в данном состоянии можно оказать на целевую систему. Тестовый сценарий в процессе работы обходит все состояния и в каждом состоянии оказывает все перечисленные воздействия.

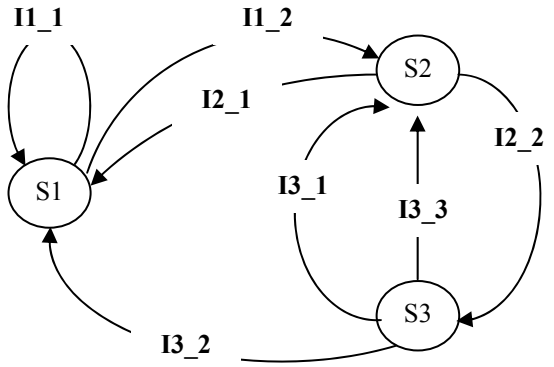


Рис. 2. *Пример автомата с тремя состояниями и семью переходами.*

На Рис. 2 изображен пример тестового сценария, в котором насчитывается три состояния (S1, S2 и S3) и семь воздействий (I1_1, I1_2, I2_1, I2_2, I3_1, I3_2, I3_3). При тестировании тестовый сценарий побывает в каждом состоянии и пройдет по каждой дуге.

К сожалению, для большинства целевых систем невозможно получить описание тестового сценария из спецификаций полностью автоматически, без помощи человека. Можно указать следующие причины:

- Число состояний целевой системы, как правило, очень велико.
- Число воздействий, которые можно оказать на целевую систему в каждом состояний, как правило, очень велико.

При всем при том, число групп *разных* состояний, как правило, вполне обозримо. Но критерий различения состояний автоматически определить невозможно, здесь нужна помощь человека – разработчика тестов.

Число *различных* воздействий на целевую систему также намного меньше общего числа воздействий. Из формальных спецификаций можно автоматически получить критерий различения воздействий, но автоматически построить воздействия можно только в простейших случаях.

В UniTesK реализован компромиссный подход к разработке тестовых сценариев.

Разработчик тестового сценария пишет процедуру различения состояний целевой системы, тем самым определяя классы эквивалентности состояний целевой системы. Каждый класс эквивалентности определяет одно состояние автомата тестового сценария.

Разработчик тестового сценария также задает процедуру, которая строит всевозможные тестовые воздействия, то есть переходы автомата тестового сценария. Эти воздействия фильтруются в соответствии с одним из сгенерированных критериев покрытия. Фильтры отсеивают избыточные воздействия, то есть воздействия, которые не улучшают уже достигнутое покрытие. Фильтрация воздействий существенно упрощает написание процедур перебора воздействий.

По предоставленным описаниям динамически строится граф состояний автомата тестового сценария. При обходе графа автоматически отслеживается покрытие требований, описанных в спецификациях для генерации отчета о покрытии.

4.3. Медиаторы

Под медиаторами в UniTesK понимается промежуточный слой между оракулами и реализацией.

Необходимость введения медиаторов вызвана тем, что в UniTesK воздействия на целевую систему и реакции целевой системы описываются в терминах модели, содержащейся в спецификациях. Перед тем как оказать воздействие на

целевую систему, необходимо перевести параметры воздействия из модельного представления в представление реализации, а после того, как воздействие оказано, необходимо перевести реакции целевой системы в модельное представление. Также необходимо отобразить изменения состояния целевой системы в абстрактном состоянии.

В UniTesK медиаторы осуществляют необходимые преобразования, оказывают воздействия на целевую систему и собирают реакции целевой системы.

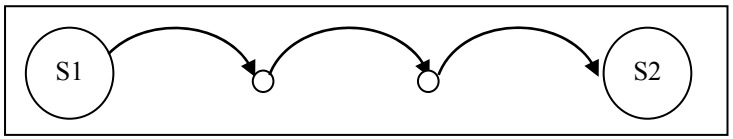


Рис. 3. *Пример цепочки реакций между двумя стационарными состояниями.*

5. Применение UniTesK к тестированию систем с отложенными реакциями

Как сказано выше, целевая система может демонстрировать реакции **спустя некоторое время** после того, как на неё было оказано воздействие. Мы называем такое поведение *отложенными реакциями*. Как правило, неудобно моделировать отложенные реакции посредством обычных конечных автоматов. Для облегчения моделирования систем с отложенными реакциями в UniTesK введено представление об *автоматах с отложенными реакциями* [11].

Автоматы с отложенными реакциями отличаются от обычных конечных автоматов тем, что переходы в между состояниями представляют собой цепочку реакций.

Как правило, целевая система демонстрирует реакции нескольких различных видов. Например, у реализации сетевого протокола есть как минимум два вида реакций – пакеты, которые отправляется в сеть, и пакеты, которые передаются на верхний уровень.

Для регистрации отложенных реакций целевой системы тестовая система содержит так называемые *кетчеры*. Кетчеры, как видно из названия, «хватают» (catch) реакции целевой системы и передают их в тестовую систему. В тестовой системе должны быть кетчеры для каждого вида реакций.

Реакции могут регистрироваться с запозданием, причем для реакций различных типов величина задержки может быть различной. Тестовой системе необходимо определить допустимый порядок реакций различных видов. Процедура определения допустимой последовательности реакций называется

сериализацией. В процессе сериализации тестовая система строит различные цепочки реакций и проверяет их допустимость. Каждая реакция рассматривается как переход между промежуточными состояниями, конечное состояние последней реакции рассматривается как последнее состояние в цепочке реакций и принадлежит множеству состояний конечного автомата. Если при сериализации не удалось найти ни одной допустимой последовательности реакций, то тестовая система выносит вердикт о рассогласовании модели и целевой системы – набор зарегистрированных реакций не соответствуют спецификации.

Сериализация реакций происходит после того, как все реакции целевой системы получены. Это накладывает ограничения на спецификации и технику тестирования. Во-первых, при разработке спецификаций используется подход «черного ящика», при котором внутреннее состояние целевой системы недоступно, и его приходится моделировать в спецификациях. Во-вторых, на целевую систему накладывается жесткое ограничение на *стационарность начальных и конечных состояний*. Рассмотрим последнее требование более подробно.

Сериализация производится после того, как собраны *все* реакции на воздействие. Это означает, что существует такой интервал времени $\mathbf{T_0}$, что в течение $\mathbf{T_0}$ с момента оказания воздействия целевая система продемонстрирует все реакции на воздействие. Состояние, в которое перейдет целевая система, обладает тем свойством, что нем система не демонстрирует спонтанных (то есть без воздействия извне) реакций. Такие состояний называются *стационарными*.

Реализованный в UniTesK метод тестирования систем с отложенными реакциями требует, чтобы в *любом* состоянии конечное состояние для *любого* допустимого в данном состоянии воздействия было стационарным, причем существует верхняя грань времени ожидания реакций по всем состояниям и воздействиям.

Кроме того, алгоритм обхода графа состояний автомата тестового сценария требует, чтобы воздействие на целевую систему оказывалось только из стационарного состояния.

Несмотря на жесткость указанных ограничений, UniTesK применим для тестирования широкого класса систем с отложенными реакциями. В частности, как показал анализ документации, протокол IPv6 удовлетворяет требованию на стационарность конечных состояний, поэтому UniTesK можно использовать для построения тестовых наборов для IPv6.

6. Использование UniTesK для тестирования MSR IPv6

Тестовый набор разрабатывался средствами реализации UniTesK для языка C – CTesK-lite. CTesK-lite поддерживает разработку спецификаций на

спецификационном расширении языка C – SEC (**S**pecification **E**xtension of **C** language).

В CTesK-lite реализована архитектура тестового набора UniTesK – тестовые сценарии, оракулы, медиаторы. Реализована поддержка тестирования систем с отложенными реакциями.

SEC – это ANSI C, к которому были добавлены ряд конструкций, характерных для академических языков формальных спецификаций. В частности, в SEC реализованы пред- и постусловия, инварианты типов данных и глобальных переменных, описатели доступа (access descriptors). Из спецификаций на языке SEC генерируется код на языке C, который затем компилируется обыкновенным компилятором C (в нашем случае MS VC 6.0).

Для разработки и сборки тестового набора мы использовали MS Visual Studio 6.0. Проверка синтаксиса и семантики для SEC не реализована, но синтаксические ошибки в спецификациях можно идентифицировать по ошибкам, которые компилятор C находит в сгенерированном коде.

Далее в данном разделе приведены краткие описания компонентов тестового набора.

6.1. Спецификации

Спецификации можно рассматривать как формальное представление требований к целевой системе. В наших спецификациях мы формализовали требования, взятые из стандартов IPv6. В основном, требования извлекались из [2-5], но использовался также ряд других RFC.

Спецификации представляют собой формализованную запись требований, изложенных в RFC. Чтобы сохранить *прослеживаемость* требований, в тесте спецификаций расставлялись ссылки на соответствующие полуформальные требования из RFC. Благодаря этому по расхождениям между реализацией и моделью, выявленным при прогоне тестов, можно определить, какое требование из RFC нарушено в реализации.

6.2. Абстрактное состояние

Абстрактное состояние – это множество структур данных, которые моделируют внутреннее состояние целевой системы. Все действия, которые выполняет целевая система, моделируются в терминах абстрактного состояния.

Документация на IPv6 не содержит требований к внутреннему устройству реализации IPv6. Абстрактное состояние было спроектировано таким образом, чтобы предоставить удобные концептуальные структуры данных для спецификаций стимулов и реакций.

Абстрактное состояние для MSR IPv6 включает следующие элементы:

- Множество исходящих пакетов IPv6. Это множество моделирует множество пакетов, которые реализация передает канальному уровню.

- Набор очередей входящих дейтаграмм UDP. Очереди используются для моделирования получения данных клиентами UDP.
- Пул фрагментов. Пул фрагментов используется для двух целей: (1) мы моделируем процедуру сборки пакетов в целевой системе; (2) мы проверяем, что из фрагментов, которые выпускает целевая система, получатель сможет воссоздать оригинальный пакет.
- Набор таблиц Neighbor Discovery. Эти таблицы используются при моделировании алгоритмов Neighbor Discovery.

6.3. Спецификации стимулов

6.3.1. Спецификация процедурных стимулов

Мы специфицировали два процедурных стимула:

- Отправить пакет UDP
- Отправить ICMPv6 Echo Request

Спецификация «Отправить пакет UDP» моделирует отправку данных через UDP сокет. У данного стимула есть следующие параметры: адрес узла назначения, адрес узла отправителя (так как узел IPv6 может иметь более одного адреса), номер порта получателя и данные, которые необходимо отправить. Спецификация утверждает, что в сеть будет отправлен пакет IPv6 с заданным исходным адресом и заданным адресом назначения, который содержит пакет UDP на указанный порт и с указанными данными.

Данная спецификация нацелена на проверку функции отправки пакетов в сеть. Так как точка доступа к упомянутой функции расположена глубоко внутри модуля IPv6, то для тестирования мы воспользовались функцией-посредником. В качестве посредника был выбран протокол UDP, так как UDP (в отличие от TCP) практически не обладает собственной внутренней функциональностью, и запрос на отправку пакета UDP почти точно отображается на запрос на отправку пакета IPv6.

Спецификация «Отправить ICMPv6 Echo Request» моделирует протокол ICMPv6 Echo, описанный в [4]. У данного стимула есть следующие параметры: адрес узла назначения, адрес узла отправителя (так как узел IPv6 может иметь более одного адреса) и данные, которые необходимо отправить в пакете ICMPv6 Echo Request. Спецификация утверждает, что в сеть будет отправлен пакет IPv6 с заданным исходным адресом и заданным адресом назначения, который содержит пакет ICMPv6 Echo Request с указанными данными, и что все пакеты Echo Reply, которые придут в ответ на запрос, будут доставлены в процесс, инициировавший запрос.

Спецификация «Отправить ICMPv6 Echo Request» также имеет отношение к функции отправки пакетов. Но, поскольку у данного стимула имеется сложная внутренняя функциональность, не связанная с отправкой пакетов, то

тестирование отправки пакетов производилось преимущественно посредством отправки пакетов UDP.

6.3.2. Спецификация непроцедурных стимулов

Алгоритм обхода графа состояний автомата тестового сценария, реализованный в CTesK-lite, требует, чтобы воздействия на целевую систему оказывались только в стационарных состояниях. Протокол IPv6 не удовлетворяет указанному ограничению, так как некоторые внешние воздействия (входящие пакеты) могут оказываться в нестационарных состояниях. Например, в протоколе ICMPv6 Echo узел испускает пакет ICMPv6 Echo Request и переходит в нестационарное состояние, в котором ожидает входящего пакета ICMPv6 Echo Reply. Таким образом, внешнее воздействие – пакет ICMPv6 Echo Reply – необходимо оказать в нестационарном состоянии.

Для того, чтобы обойти ограничение на стационарность начального состояния, непроцедурные стимулы были специфицированы как отложенные реакции на некоторые процедурные стимулы. Мы ввели стимул «Отправить пакет IPv6 в целевую систему». Спецификация для данного стимула тривиальна. Медиатор для данного стимула отправляет пакет IPv6 с удаленного узла. Собственно входящий пакет рассматривается как отложенная реакция на данный стимул.

6.4. Спецификации реакций

MSR IPv6 демонстрирует два вида отложенных реакций - исходящие пакеты IPv6 и входящие пакеты UDP, то есть данные, которые получают клиенты протокола UDP. Кроме того, по техническим причинам мы моделировали как реакции и входящие пакеты IPv6 (см. обсуждение выше).

Спецификация для исходящих пакетов IPv6 проверяет следующее:

- Все пакеты, которые целевая система отправляет в сеть, удовлетворяют требованиям на формат пакетов, изложенным в соответствующих RFC.
- Все фрагменты, которые целевая система отправляет в сеть, можно собрать в пакет IPv6.
- Корректность данных для некоторых видов пакетов (например, UDP и ICMPv6).

Спецификация для входящих пакетов IPv6 разбирает входящие пакеты IPv6. Если в пакете нет ошибок, то спецификация определяет получателя данного пакета и моделирует получение данных пакета получателем. Если в пакете обнаружена ошибка, то спецификация определяет, какое сообщение об ошибке следует послать.

Спецификация для входящих пакетов IPv6 также моделирует сборку пакетов из фрагментов. В спецификации восстанавливается исходный пакет, и моделируется доставка восстановленного пакета получателю.

Спецификация для входящих дейтаграмм UDP проверяет, что дейтаграмма получена в правильном сокете, и что данные получены без искажений.

6.4.1. Спецификация фрагментации

Функция фрагментации формально относится к функции отправки пакета в сеть. Тем не менее, спецификация стимулов не описывает фрагментацию. Проверка правильности фрагментации возложена на спецификацию для исходящих пакетов. Данная спецификация проверяет, что из фрагментов можно собрать оригинальный большой пакет, собирает большой пакет и затем проверяет, что полученный пакет соответствует одному из пакетов, которые были созданы по запросу протокола верхнего уровня.

6.4.2. Спецификация Neighbor Discovery

Подсистема Neighbor Discovery не содержит входных точек, доступных извне модуля MSR IPv6. ND обрабатывает входящие пакеты и сохраняет информацию о соседних узлах во внутренних таблицах. При отправке IPv6 пакета в сеть соответствующая подсистема IPv6 запрашивает у ND информацию об узле назначения пакета. В результате такого запроса ND может выслать один или более служебных пакетов.

Спецификация для входящих пакетов распознает пакеты Neighbor Discovery и определяет, какая информация должна сохраняться в таблицах Neighbor Discovery.

Спецификация для исходящих пакетов IPv6 проверяет следующее:

1. Реализация отправляет пакеты известным узлам.
2. Реализация производит поиск соседа перед отправкой пакетов незнакомым соседним узлам.
3. Реализация не отсылает пакеты, если Neighbor Discovery не в состоянии найти узел адресат.
4. Реализация правильно определяет, какие узлы находятся на том же сегменте локальной сети, а какие находятся вне сегмента локальной сети.
5. Реализация находит маршрутизаторы в локальной сети и корректно обновляет информацию о маршрутизаторах.

Заметим, что спецификация не старается предсказать, на какой узел локальной сети будет выслан пакет. Вместо этого спецификация проверяет, что узел, выбранный реализацией, соответствует информации, собранной Neighbor Discovery на момент отправки пакета.

6.5. Транспорт стимулов и реакций

В данном разделе описывается, как тестовая система оказывает воздействие на целевую систему (MSR IPv6) и собирает реакции целевой системы.

6.5.1. Медиаторы для процедурных стимулов

Тестовая система работает на том же компьютере, что и целевая система. Благодаря этому медиаторы для процедурных стимулов устроены просто.

Медиатор для отправки дейтаграммы UDP вызывает процедуру sendto для отправки указанных данных по указанному адресу через указанный сокет. После того, как sendto возвращает управление, медиатор анализирует возвращенное значение и, при необходимости, код ошибки и формирует собственное возвращаемое значение (`OK`/`FAIL`).

Медиатор для функции Echo Request обращается к соответствующей точке доступа транспортного драйвера MSR IPv6. После завершения вызова медиатор анализирует возвращенное значение и, при необходимости, код ошибки, сохраняет данные ответа (Echo Reply) и формирует собственное возвращаемое значение (`OK`/`FAIL`)

6.5.2. Медиаторы для непроцедурных стимулов

Как упоминалось выше, в интерфейс целевой системы входят непроцедурные стимулы – пакеты IPv6, которые поступают в целевую систему из сети.

Медиатор для непроцедурных стимулов представляет собой распределенное приложение, реализованное в рамках архитектуры клиент-сервер. Клиент – тестовая система – передает серверу запрос, в котором содержится IPv6 пакет и адрес узла назначения. Сервер, который работает на некотором узле в том же сегменте сети, что и целевой узел, пересылает указанный пакет по указанному IPv6 адресу.

Для того, чтобы обмен запросами между клиентом и сервером не оказывал воздействия на целевую систему, запрос серверу передается по сети IPv4.

6.5.3. Сбор реакций целевой системы

Сбор реакций целевой системы осуществляют специальные программы, которые называются кетчерами. Как было сказано выше, MSR IPv6 демонстрирует два вида отложенных реакций – исходящие IPv6 пакеты и UDP датаграммы, которые получают в сокетах.

6.5.3.1. Входящие и исходящие пакеты IPv6

Кетчер для входящих/исходящих пакетов IPv6 перехватывает кадры Ethernet, в которых передаются пакеты IPv6. Кетчер реализован как модуль расширения Microsoft Network Monitor. Кетчер не разбирает кадры Ethernet, а передает их в тестовую систему. Заметим, что кетчер не умеет отличать исходящие кадры от входящих.

В тестовой системе перехваченные кадры преобразуются. Тестовая система извлекает из кадра вложенный в него пакет IPv6 и определяет, является пакет входящим или исходящим. При этом формируется специальная структура

данных, которая используется в спецификациях для представления пакетов IPv6.

6.5.4. Входящие дейтаграммы UDP

При старте тестовая система открывает несколько сокетов и запускает кетчер, который слушает открытые сокеты. Когда в один из сокетов прибывает дейтаграмма, кетчер получает дейтаграмму и определяет ее исходный.

6.6. Тестовые сценарии

В ходе проекта были разработаны 15 тестовых сценариев. Приблизительно тестовые сценарии можно разделить на 4 группы:

- Тесты на отправку пакетов. Тесты из данной группы проверяют, как целевая система передает пакеты верхнего уровня в сеть. Также производится проверка того, как реализация проводит нарезку больших пакетов на фрагменты.
- Тесты на получение пакетов. Тесты из данной группы проверяют, как целевая система обрабатывает различные входящие пакеты. Тестируются функция сборки фрагментов и доставка пакетов протоколам верхнего уровня. Значительное место занимает проверка того, как система обрабатывает поданные на вход некорректные пакеты.
- Тесты для функции ICMPv6 Echo. Тесты из данной группы проверяют корректность реализации простого диагностического протокола Echo, встроенного в ICMPv6.
- Тесты для Neighbor Discovery. Тесты из данной группы проверяют, что реализация Neighbor Discovery соответствует требованиям, изложенным в RFC 2461 [5].

Заметим, что представленное разделение нестрогое, и отдельные тестовые сценарии можно отнести сразу к нескольким группам.

Тестовые сценарии для MSR IPv6 осуществляют два вида тестирования:

- тестовые сценарии проверяют корректность работы целевой системы на корректных входах;
- тестовые сценарии проверяют устойчивость целевой системы по отношению к ошибкам во входных пакетах IPv6.

6.7. Процесс разработки тестового набора

Процесс разработки тестового набора для MSR IPv6 можно разделить на несколько фаз:

1. Определение интерфейса.

2. Разработка спецификаций для отправки и получения пакетов без Neighbor Discovery.
3. Разработка и прогон тестовых сценариев для отправки и получения пакетов без Neighbor Discovery.
4. Пополнение спецификаций функцией Neighbor Discovery.
5. Разработка и прогон тестовых сценариев для Neighbor Discovery.
6. Прогон всего полученного тестового набора.

На фазе определения интерфейса был определен состав функций IPv6 для тестирования, и определены стимулы и реакции, относящиеся к выбранным функциям.

Разработка спецификаций проводилась в два этапа. На первом этапе были разработаны спецификации, в которых не моделировалось поведение Neighbor Discovery. Для полученных спецификаций были разработаны и отлажены тестовые сценарии.

Протоколы Neighbor Discovery предназначены для сбора и обновления информации о конфигурации сегмента локальной сети, к которой подключен узел IPv6. В спецификациях и тестовых сценариях, которые мы разработали на первом этапе, конфигурация сети предполагалась заданной и неизменной во времени. Это предположение позволило разработать и отладить довольно содержательные тестовые сценарии для функций отправки и получения пакетов и ICMPv6.

На втором этапе мы добавили спецификации протоколов Neighbor Discovery. Для отладки полученных спецификаций мы воспользовались предположением, что сценарии, разработанные для спецификаций без ND, должны корректно работать и для спецификаций с ND.

Избранная нами стратегия себя оправдала. При прогоне тестовых сценариев, разработанных на первом этапе, было выявлено и исправлено много недочетов в спецификациях Neighbor Discovery.

После того, как спецификации Neighbor Discovery были отлажены, были разработаны тестовые сценарии специально для Neighbor Discovery.

7. Результаты

Был разработан тестовый набор для реализации протокола IPv6. Тестовый набор был пропущен на Windows 2000, в качестве целевой системы выступала реализация IPv6 от Microsoft Research (MSR IPv6 version 1.4).

Проект по разработке тестового набора для MSR IPv6 показал применимость UniTesK для тестирования сложного API с непроцедурными стимулами и отложенными реакциями. Также показана применимость UniTesK для тестирования внутренних подсистем ядра Windows 2000.

В ходе тестирования были обнаружены отклонения от стандартов IPv6 и ошибки программирования.

Отклонения от стандартов заключаются в том, что в ряде случаев поведение MSR IPv6 отличается от требований, изложенных в стандартах на IPv6.

При тестировании мы выявили также дефекты, которые можно охарактеризовать как ошибки, допущенные при программировании.

7.1. Выявленные дефекты

При тестировании мы обнаружили, что в реализации алгоритма сборки фрагментов в MSR IPv6 присутствует ошибка. При получении последовательности фрагментов определенного вида в модуле MSR IPv6 возникает исключительная ситуация, которая приводит к краху ядра операционной системы и перезагрузке Windows 2000.

В MSR IPv6 некорректно спроектирован интерфейс доступа к функции ICMPv6 Echo Request. Клиентский процесс, инициировавший запрос, блокируется до прибытия первого ответа (Echo Reply) или до истечения таймаута. Если на запрос поступили ответы, то первый ответ возвращается клиенту, а все остальные отбрасываются. Такое поведение не соответствует требованию передавать клиенту все поступившие ответы [4].

В одном частном случае реализация алгоритма сборки фрагментов в MSR IPv6 нарушает одно из требований, изложенное в RFC 2460 [2]. RFC требует, чтобы служебная информация, которая передается во фрагментах, не попадала в восстановленный пакет. Тестирование показало, что для фрагментов определенного вида MSR IPv6 оставляет служебную информацию в восстановленном пакете.

Более подробное описание выявленных дефектов можно найти в [12].

7.2. Оценка покрытия кода целевой системы

Для точного определения покрытия кода целевой системы необходимо пропустить тесты на отладочной версии ядра Windows 2000 с включенным профилировщиком ядра. Это сделано не было, поэтому оценка покрытия кода получена косвенным методом.

Мы составили список всех процедур, которые есть в целевой системе, и для каждой процедуры оценили, будет ли она вызываться при прогоне тестов.

Оказалось, что 20% процедур в целевой системе не будут вызываться при прогоне тестов; как правило, эти процедуры реализуют функции IPv6, исключенные из рассмотрения при разработке тестового набора. 60% процедур реализуют выбранные нами функции IPv6; эти процедуры должны вызываться при пропуске тестов. Про остальные 20% мы затрудняемся сказать что-либо определенное; как правило, это вспомогательные процедуры.

Итого получается оценка покрытия исходных текстов 60-80%.

8. Заключение

Проект по разработке тестового набора для MSR IPv6 продемонстрировал применимость UniTesK для тестирования сложного API на платформе Win32.

UniTesK был использован для тестирования системы, расположенной в ядре Windows 2000. Интерфейс целевой системы содержит непроцедурные стимулы и отложенные реакции.

Были выявлены ошибки в системе, которая уже несколько лет находилась в эксплуатации и проходила испытания другими тестовыми наборами.

Литература

1. Information technology -- Open Systems Interconnection -- Basic Reference Model. *ISO/IEC 7498, 1994*
2. S. Deering, R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *RFC 2460, December 1998*
3. M. Crawford. Transmission of IPv6 Packets over Ethernet Networks. *RFC 2464, December 1998*
4. A. Conta, S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. *RFC 2463, December 1998*
5. T. Narten, E. Nordmark, W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). *RFC 2461, December 1998*
6. http://research.microsoft.com/msripv6/
7. R. P. Draves, A. Mankin, B. D. Zill. Implementing IPv6 for Windows NT. *Proceedings of the 2nd USENIX Windows NT Symposium, Seattle, WA, August 3–4, 1998*
8. I. Bourdonov, A. Kossatchev, V. Kuliamin, A. Petrenko. UniTesK Test Suite Architecture. *Proceedings of FME 2002. LNCS 2391, pp. 77-88, Springer-Verlag, 2002.*
9. I. Bourdonov, A. Kossatchev, A. Petrenko, D. Galter. KVEST: Automated Generation of Test Suites from Formal Specifications. *FM'99: Formal Methods.LNCS, volume 1708, Springer-Verlag, 1999, pp. 608–621.*
10. И. Б. Бурдонов, А. В. Демаков, А. С. Косачев, А. В. Максимов, А. К. Петренко. Формальные спецификации в технологиях обратной инженерии и верификации программ. *Труды Института системного программирования, 1999 г., том 1, стр. 35-47*
11. И.Б.Бурдонов, А.С.Косачев, В.В.Кулямин. Асинхронные автоматы: классификация и тестирование. *В данном сборнике*
12. I.Agamirzian, S.G.Groshev, A.V.Khoroshilov, G.N.Kluchnikov, A.S.Kossatchev, V.A.Omelchenko, N.V.Pakoulin, A.K.Petrenko, V.Z. Shnitman. MSR IPv6 verification project. *Technical report, December 2001*