Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по дисциплине

Методы тестирования ПО

Выполнил:
студент 4 курса группы 22407

В. А. Баканов

Руководитель:

К. А. Кулаков, к.ф-м.н., доцент

Петрозаводск — 2020

Содержание

1 Объект тестирования

Тестируемое в рамках текущей дисциплины приложение решает задачу установления соединения двух удаленных устройств, находящихся в разных подсетях за маршрутизаторами с механизмом трансляции адресов (NAT [2]). Приложение работает в связке с аналогичным, с точки зрения логики, приложением, размещенным на удаленном устройстве под управлением ОС Linux.

Тестируемая программа работает на мобильном устройстве под управлением ОС Android и написана на языке программирования Java.

1.1 Функциональность объекта тестирования

В приложении реализованы следующие функции:

1. Установление соедения удаленных устройств в пределах локальной сети
2. Установление соединения удаленных устройств в разных подсетях за NAT

1.2 Требования к объекту тестирования

К приложению сформулированы следующие требования:

- Т1: Приложение должно работать на мобильном устройстве под управлением ОС Android версии 5 и выше.
- Т2: Приложение может быть оформлено в произвольном дизайне.
- Т3: Приложение должно содержать в себе не менее одной кнопки.

1.3 Функциональные требования

К приложению сформулированы следующие требования:

1. ФТ1: Приложение должно установить соединение с удаленным устройством, управляемым операционной системой Linux, при условии, что они подключены к одной точке доступа (в пределах локальной сети)
2. ФТ2: Приложение должно установить соединение с удаленным устройством, управляемым операционной системой Linux, при условии, что мобильное устройство подключено к сети мобильного оператора, а устройство Linux —к точке Wi-Fi. Точки доступа должны находиться в разных подсетях за NAT.

2 Стратегия тестирования

2.1 Используемые инструменты

- **Для блочного тестирования:** библиотека JUint в среде разработки Android Studio. Тестирование проводится в виде инструментальных тестов. Может проводится как на эмуляторе устройства Android, так и на физическом устройстве.
- **Для интеграционного тестирования:** инструменты и окружение аналогичны тем, которые используюся в блочном тестировании. Для тестов И-21, И-22 необходимо вмешательство тестировщика ручным методом.
- **Для аттестационного тестирования:** специальные инструменты не используются. Тестирование проводится ручным методом.

Оценка покрытия кода будет проводится средствами библиотеки JUnit в среде разработки Android Studio.

2.2 Отношение к тестированию структурных элементов

На рисунке 1 отображена архитектура тестируемой программы. Белым цветом выделены модули, участвующие в тестировании. Рядом с тестируемыми методами размещены литеры:

- **Б** — метод подвергнется блочному тестированию;
- **И** — метод подвергнется интеграционному тестированию.

Классы, которые не подвергнутся тестированию:
LocalAgent: единственная задача класса —вызов методов библиотеки ice4j. Класс не работает с входными данными и не изменяет их. В единственном методе класса тривиальная логика. Одна из схем интеграции методов библиотеки ice4j подвергнется тестированию (подразделы 2.4.1 - 2.4.3 настоящего отчета)
StateListener: данный класс является наследником интерфейса ChangeListener. Единственная задача данного класса - прослушивать состояния агента. Данный класс не изменяет данные.

Методы, которые не подвергнутся тестированию:
createSessionDescription(String s): данный метод извлекает данные из потока IceMediaStrean

и вызывает функцию, предоставляемую Java API, что не требует тестирования.

createVersion: данный метод является сеттером

createMedia: данный метод является сеттером

createOrigin: данный метод является сеттером.

2.3 Архитектура тестируемой программы

Архитектура программы с конкретными классами и методами приведена на рисунке 1. В приведенной схеме красными стрелками обозначены вызовы соответствующих методов. Зелеными — передаваемые результирующие данные.

В классе *LocalAgent* происходит инициализация агента, создание потока, в котором будет работать технология, генерация локального SDP.

Для обмена описаниями сессии между устройствами реализован и размещен на сервере cs.petrsu.ru *php*-скрипт.

Взаимодействие с удаленным скриптом осуществляет класс *URLConnectionReader*, который с помощью GET-запроса отправляет две переменные: SDP и строковое значение операционной системы ("android"или "linux").

Скрипт работает с двумя файлами (условно обозначим их, как androidFile и linuxFile). Принцип работы: если к скрипту обращается Android-устройство, то оно записывает SDP в androidFile, а считывает из linuxFile и после прочтения стирает данные. Для Linux-устройства аналогичен обратный порядок действий.

Полученное в результате обмена SDP удаленного устройства передается методу *parseSdp* класса *SdpUtils*. Здесь проводится парсинг SDP-сообщения с целью инициализации атрибутов экземпляра класса *SessionDescription*, с которым будет работать технология ICE. Парсер реализован в библиотечном классе *SDPAnnounceParser*, логика работы с инициализированными атрибутами реализована в методе *parseSdp*, который для парсинга части SDP-описания вызывает метод *parseCandidate*.

Все необходимые данные для работы ICE находятся в сформированном потоке, процесс установления соединения запускается методом библиотеки *startConnectivityEstablishment*. Подробное описание запуска ICE будет описано в подразделе 2.4.1-2.4.3 настоящего отчета.

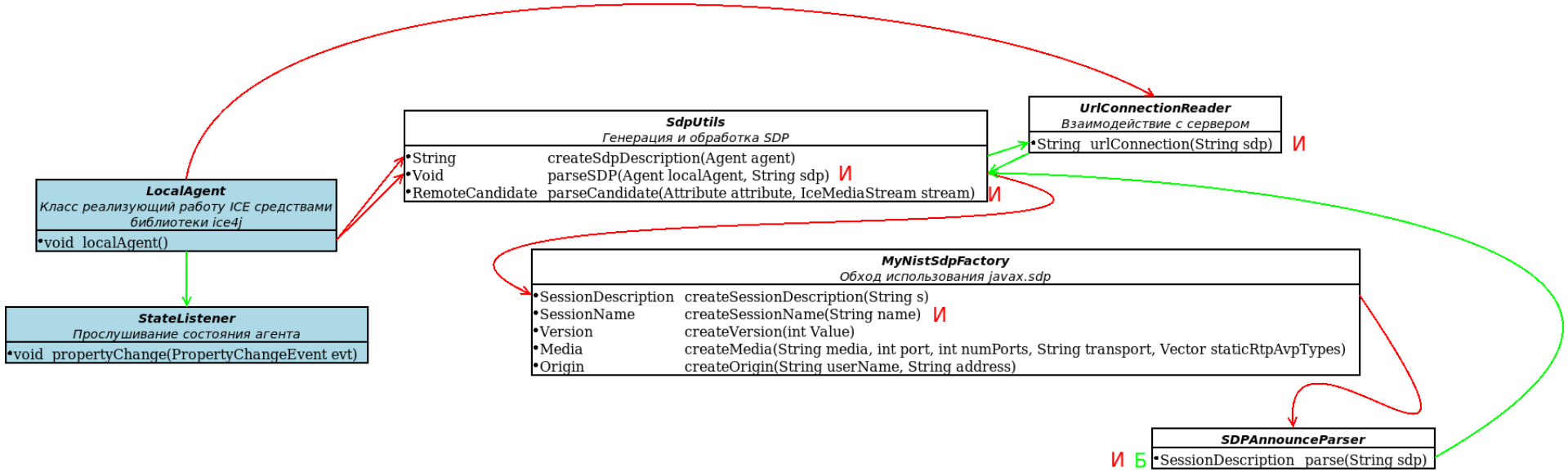


Рис. 1: Архитектура приложения

2.4 Стратегия блочного тестирования

Блочные тесты проверяют отдельные методы программы при условии, что их выполнение не зависит от взаимодействия с другими. При данном виде тестирования на вход метода подаются различные аргументы, корректность его выполнения оценивается по возвращаемому результату. Блочное тестирование будет проведено для нижеследующего метода:

- Класс: *SDPAnnounceParser* Метод: *parse(String sdp)*

  Данный метод в качестве аргумента принимает строку SDP-описания и возвращает объект класса SessionDescription с инициализированными атрибутами строкового типа:

  - ice-ufrag
  - ice-pwd
  - mediaDescription

Тестирование заключается в сравнении полученных значений атрибутов с конкретными ожидаемыми.

2.5 Стратегия интеграционного тестирования

Данный вид тестирования проверяет интеграцию модулей на соответствие требованиям, их взаимодействие между собой. Методы принимают аргументы и передают результирующие данные другим. В качестве интеграционных тестов будут протестированы три последовательности взаимодействующих методов. Две из них представлены на рисунках 2 и 3. Зеленым цветом отображены методы, белым - входные и выходные данные, синим - данные, которые изменяет метод в процессе выполнения. Зеленая стрелка отображает обращение к изменяемым данным.

На приведенных схемах черные пронумерованные стрелки демонстрируют последовательность выполнения шагов.

2.5.1 Стратегия для схемы взаимодействия методов № 1

Описание шагов:

- *Запуск теста*

- Шаг 0: вызов метода с аргументом строкового типа
- Шаг 1: вызов метода с аргументом строкового типа
- Шаг 2: возврат значения — объект класса SessionDescription
- Шаг 3: возврат значения — объект класса SessionDescription
- Шаг 4: вызов метода с аргументами текущего потока типа IceMediaStream и Attribute
- Шаг 5: возврат значения типа RemoteCandidate
- *Проверка результата*

Результат тестирования будет проверяться после выполнения работы приведенных на схеме методов.

Суть проверки: соответствие значений атрибутов объекта класса SessionDescription, извлеченного из потока, требуемым значениям конкретного SDP-описания.

Перечень проверяемых атрибутов (в скобках указан тип значения):

- ice-pwd (String)
- ice-ufrag (String)
- Тип медиапотока (String)
- Количество кандидатов (Int)
- Атрибуты кандидатов:
  - Foundation (String)
  - Транспортный протокол (Transport)
  - Приоритет (long)
  - Тип кандидата (CandidateType)
  - Транспортный адрес (String)
  - Порт (Int)
  - Локальный адрес, транслируемый в сетевой (String)

В схеме интеграции участвуют следующие методы:

- *parseSdp(Agent agent, String sdp)* — вызывает метод, запускающий парсинг входной строки, работает с инициализированными атрибутами в созданнном потоке, извлекаемом из объекта класса Agent.

  Для парсинга медиа-описания вызывает метод *parseCandidate(IceMediaStream stream, Attribute attribute)*

  Класс: SdpUtils

  Входные данные: строка SDP-описания, объект класса Agent.

- *CreateSessionDescription (String sdp)* — метод, запускающий парсинг входной строки и отлавливающий исключения работы парсера.

  Класс: MyNistSdpFactory

  Входные данные: строка SDP-описания

  Выходные данные: объект класса SessionDescription

- *parse(String sdp)* — метод, реализующий парсинг входной строки и инициализирующий атрибуты объекта класса SessionDescription.

  Класс: SdpAnnounnceParser

  Входные данные: строка SDP-описания

  Выходные данные: объект класса SessionDescription

- *parseCandidate(IceMediaStream stream, Attribute attribute)* — метод, реализующий парсинг атрибутов кандидатов объекта класса SessionDescription в текущем потоке. Инициализирует атрибуты объекта класса RemoteCandidate.

  Класс: SdpUtils

  Входные данные: Attribute - атрибуты кандидатов объекта класса SessionDescription, IceMediaStream - текущий поток.

  Выходные данные: Объект класса RemoteCandidate

2.5.2 Стратегия для схемы взаимодействия методов № 2

Описание шагов:

- *Запуск теста*

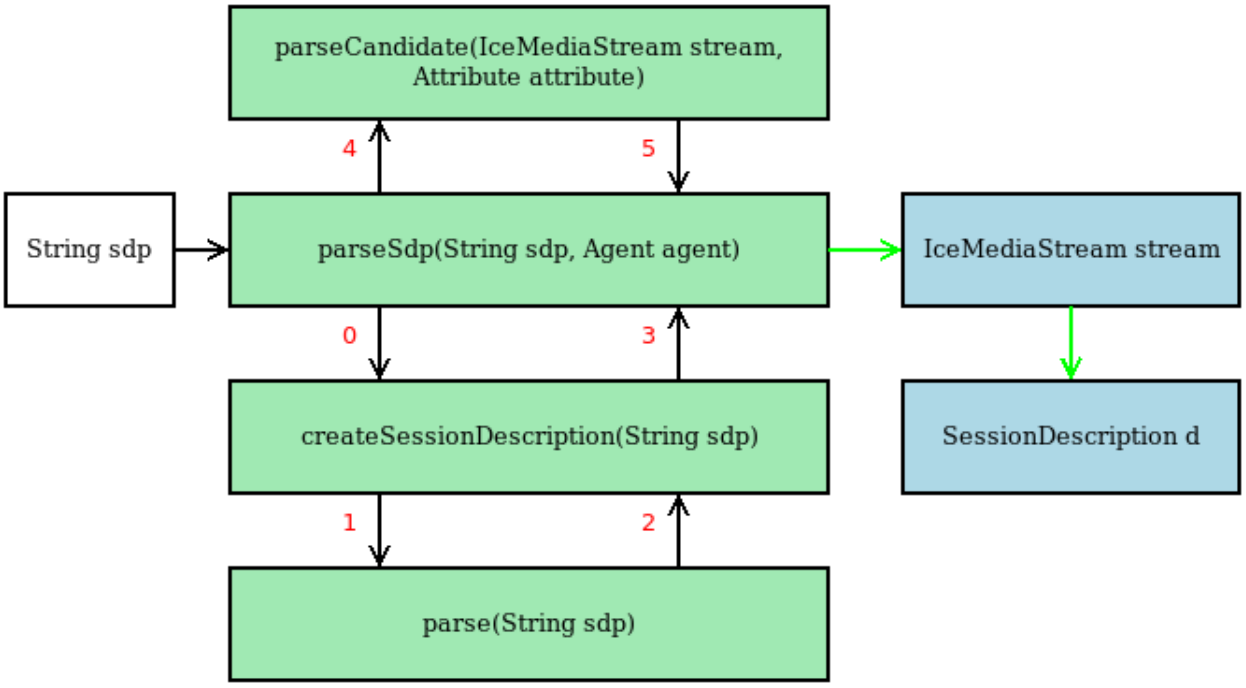


Рис. 2: Схема взаимодействия методов № 1

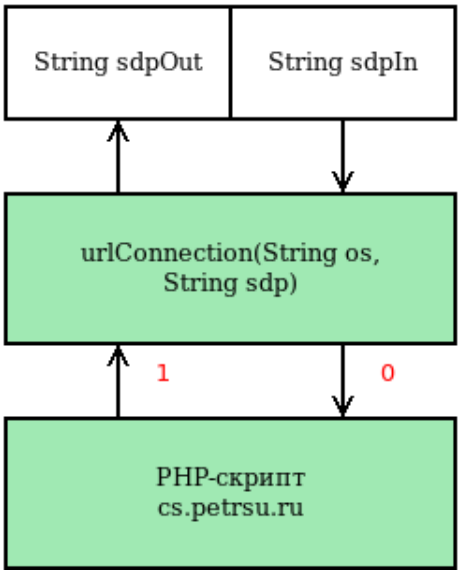


Рис. 3: Схема взаимодействия методов № 2

- Шаг 0: передача данных с помощью GET-запроса удаленному PHP-скрипту
- Шаг 1: чтение принятых данных
- *Проверка результатов теста*

В схеме интеграции участвуют следующие методы:

- *urlConnection(String os, String sdp)* — метод, который с помощью GET-запроса отправляет закодированные переменные *os* и *sdp* удаленному PHP-скрипту, считывает и декодирует ответные данные. *Класс:* URLConnectionReader

  Входные данные: строка SDP-описания, строка значения операционной системы.

- *server_http.php* — скрипт, осуществляющий обмен данными.

  Локация: cs.petrsu.ru\∼bakanov\http_server\server_http.php

  Входные данные: переменная SDP-описания, переменная значения операционной системы

  Выходные данные: объект класса SessionDescription

Переменная *os* может принимать следующие значения:

- "android"
- "linux"

Тестирование результата будет проводиться после выполнения всех составляющих схемы интеграции №2.

Суть проверки: соответствие возвращаемого значения SDP-описания ожидаемому.

2.5.3 Теоретическое описание тестирумого процесса библиотеки ice4j

Удаленные устройства в терминологии ICE принято называть *пользовательскими агентами* (далее ПА).

Каждый ПА имеет набор кандидатов, представляющих из себя, говоря не строго, набор транспортных адресов.

После того, как ПА №1 собрал данные о своих кандидатах, происходит их упорядочивание от высшего к низшему приоритету. ПА №1 отправляет информацию о кандидатах ПА №2, в атрибутах сообщения SDP [4]. ПА №2 выполняет аналогичный сбор и упорядочивание кандидатов и отправляет ответ SDP со своим списком. Каждый ПА рассматривает два

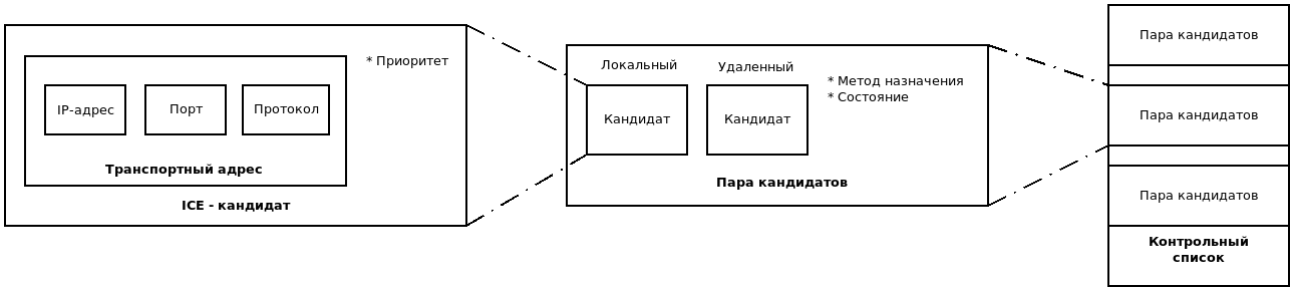


Рис. 4: Компоненты пар-кандидатов технологии ICE

списка кандидатов и создает из них *пары - кандидаты*. Пары собираются в *контрольные списки* и проводятся транзакции вида "запрос-ответ"к серверу STUN в целях выявления работоспособных.

Упрощенная схема компонентов пар-кандидатов приведена на рисунке 4.

Каждая пара кандидатов имеет свойство — *состояние*. Существует пять состояний [3], которые назначает агент в ходе проверки контрольного списка пар.

1. Заморожено (Frozen). Пара проверяется только после перехода в состояние ожидания. Чтобы войти в состояние ожидания, нужно дождаться прохождения другой текущей проверки.

2. Ожидание (Waiting). Будет выполнена проверка.

3. Выполняется (In-Progress). Транзакция в процессе выполнения.

4. *Успешно* (Success). Успешный результат проверки пары. Критерий успешности будет описан ниже.

5. Провалено (Failed). Неудачный результат проверки пары.

На рисунке 6 отображена диаграмма состояний проверки пар-кандидатов.

Критерий успешности проведения транзакций: четырехстороннее рукопожатие на портах, которые будут использоваться для передачи данных. Во всех остальных случаях транзакция считается неудачной.

2.5.4 Стратегия для схемы взаимодействия методов № 3

Архитектура тестируемой программы изображена на рисунке 7. Для компактности отображения некоторые методы изображены без аргументов. Для методов, которые вызываются в классах, инициализируемых данными, аргументы прописаны в описании ме-

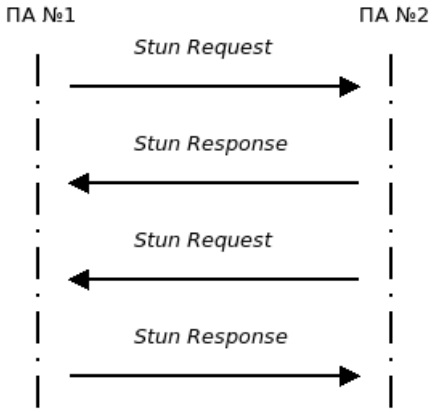


Рис. 5: Базовая проверка подключения

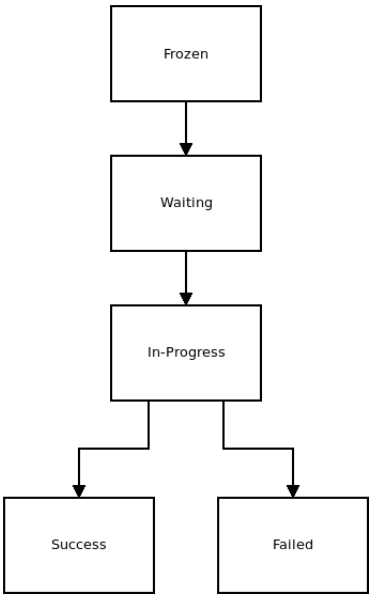


Рис. 6: Диаграмма состояний проверки пар-кандидатов

тодов с формулировкой *косвенные входные данные*. Стрелки черного цвета иллюстрируют вызов метода, зеленые — передачу результирующих данных.

Описание приведенных в схеме методов:

- *startConnectivityCheck()* — метод, запускающий процесс установления соединения. Работает с текущим потоком *IceMediaStream*

  Класс: Agent

  Косвенные входные данные: поток IceMediaStream

- *startCheck()* — метод, запускающий процесс установления соединения с извлеченным из потока контрольным листом пар-кандидатов. Работает с текущим потоком *IceMediaStream*

  Класс: ConnectivityCheckClient

  Косвенные входные данные: поток IceMediaStream

- *startCheck(CheckList checkList)* — метод, запускающий процесс установления соединения с извлеченным из потока контрольным листом пар-кандидатов. Работает с текущим потоком *IceMediaStream*

  Класс: ConnectivityCheckClient

  Входные данные: контрольный список —объект класса CheckList

- *startCheckForPair(CandidatePair pair, int originalWaitInterval, int maxWaitInterval, int maxRetransmissions)* — метод проверки пар-кандидатов. Из объекта класса CandidatePair извлекаются данные кандидата, формируется запрос, в который помещаются необходимые атрибуты. Проверяется, является ли проверяемый агент котролируемым или контролирующим. Формируется транзакция, в которую передаются данные о паре кандидатов. С помощью объекта класса StunStack отправляется запрос.

  Класс: ConnectivityCheckClient

  Входные данные: пара-кандидатов (CandidatePair), значения временных интервалов int originalWaitInterval, int maxWaitInterval, int maxRetransmissions, соответствующих RFC 5245.

  Выходные данные: объект класса TransactionID

- *processResponse(StunResponseEvent ev)* — метод, обрабатывающий ответ удаленного узла.

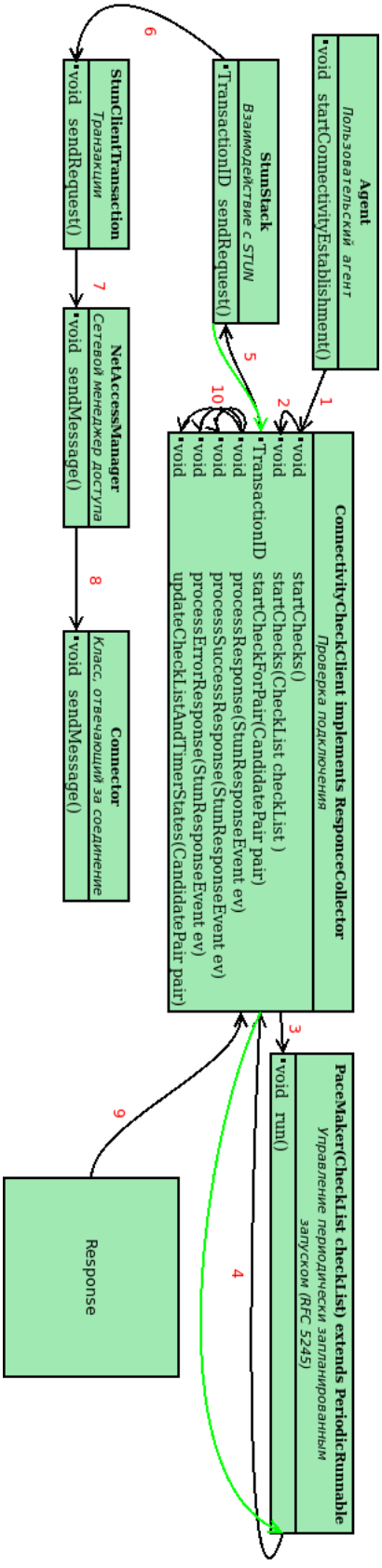


Рис. 7: Схема взаимодействия методов № 3

Класс: ConnectivityCheckClient

Входные данные: пара-кандидатов (CandidatePair), значения временных интервалов int originalWaitInterval, int maxWaitInterval, int maxRetransmissions, соответствующих RFC 5245.

Выходные данные: объект класса TransactionID

- *processResponse(StunResponseEvent ev)* — метод, обрабатывающий ответ STUN в соответствии с RFC 5245. В зависимости от типа возвращаемого сообщения (BINDING_ERR
или BINDING_SUCCESS_RESPONSE) вызывает методы processErrorResponse или processSuccessResponse соответственно.

  Класс: ConnectivityCheckClient

  Входные данные: объект класса StunResponseEvent

- *processSuccessResponse(StunResponseEvent ev)* — метод, обрабатывающий успешный ответ STUN в соответствии с RFC 5245 и реализующий логику изменения состояний объектов класса CandidatePair и CheckList.

  Класс: ConnectivityCheckClient

  Входные данные: объект класса StunResponseEvent

- *processErrorResponse(StunResponseEvent ev)* — метод, обрабатывающий неудачный ответ STUN в соответствии с RFC 5245 и реализующий логику изменения состояний объектов класса CandidatePair и CheckList.

  Класс: ConnectivityCheckClient

  Входные данные: объект класса StunResponseEvent

- *updateCheckListAndTimerStates(CandidatePair pair)* — метод, обновляющий состояние объекта класса CheckList.

  Класс: ConnectivityCheckClient

  Входные данные: объект класса StunResponseEvent

- *sendRequest(Request request, TransportAddress sendTo, TransportAddress sendThrough, ResponceCollector Collector)* — метод вызывающий метод генерации идентификатора транзакции и метод отправки запроса.

  Класс: StunStack

Входные данные: Request request, TransportAddress sendTo, TransportAddress sendThrough, ResponceCollector Collector

Выходные данные: объект класса TransactionID

- *sendRequest(Request request, TransportAddress sendTo, TransportAddress sendThrough, ResponceCollector Collector)* — метод вызывающий метод генерации идентификатора транзакции и метод отправки запроса.

  Класс: StunClientTransaction

  Косвенные входные данные: Request request, TransportAddress sendTo, TransportAddress sendThrough, ResponceCollector Collector

- *sendMessage(Request request, TransportAddress sendTo, TransportAddress sendThrough)* — метод вызывающий метод метод отправки запроса.

  Класс: NetAccessManager

  Входные данные: Request request, TransportAddress sendTo, TransportAddress sendThrough, ResponceCollector Collector

- *sendMessage(byte[] message, TransportAddress sendTo)* — метод, упаковывающий данные в DatagramPacket и отправляющий через сокет.

  Класс: Connect

  Входные данные: byte[] message, TransportAddress sendTo

- *run()* — метод, реализующий управление запланированным запуском проверок контрольного списка пар-кандидатов. Проверяет возвращаемое значение TransactionID, в случае null устанавливает состояние контрольного списка —failed

  Класс: PaceMaker

  Косвенные входные данные: CheckList из текущего потока IceMediaStream

Описание шагов, приведенных на рисунке 7:

1. *Запуск теста*
2. Вызов метода startChecks(), запускающего процесс установки подключения
3. Вызов метода startChecks, аргумент - объект класса CheckList
4. Вызов метода run()

5. Вызов метода StartChecksForPair для инициализации переменной типа TransportID
6. Вызов метода sendRequest для отправки запроса и возврата значения типа TransportID
7. Вызов метода sendRequest()
8. Вызов метода sendMessage()
9. Вызов метода sendMessage(), отправка запроса
10. Получение ответа в виде объекта класса StunResponseEvent, вызов метода processResponse
11. Вызов метода processSuccessResponse в случае удачного ответа STUN в соответствии с RFC 5245
12. Вызов метода processErrorResponse в случае неудачного ответа STUN в соответствии с RFC 5245
13. Вызов метода updateCheckListAndTimerStates(CandidatePair pair) для обновления состояния контрольного списка кандидатов.
14. *Проверка результата*

Результатом выполнения методов, приведенных на рисунке 8, является изменение состояния контрольного списка кандидатов.

Интеграционное тестирование заключается в моделировании ситуации, в результате которой состояние контрольного списка принимает конкретное значение. Результат проверяется на соответствие RFC 5245.

Допустимые значения состояния объекта класса CheckList:

- Frozen
- Waiting
- Running
- In-Progress
- Failed

2.6 Стратегия аттестационного тестирования

В ходе аттестационного тестирования будет проверена работоспособность приложения и его соответсвие функциональным требованиям. Метод проведения: вручную.

Тестирующий человек, по заранее заданным сценариям, проводит соответствующие действия. Полученные результаты сравниваются с требуемыми.

2.7 Способы оценивания результатов тестов

2.7.1 Критерий прохождения тестов

Любой тест считается *успешно* пройденным при совпадении ожидаемого и фактического результатов. Во всех остальных случаях тест считается *неудачным*. В случае неудачного завершения теста проводится проверка отсутствия логических ошибок в самом тесте. Если логические ошибки в тесте отсутствуют, то данные о найденной ошибке заносятся в журнал.

2.7.2 Критерий приостановления тестирования

Тестирование приостанавливается в случае, если количество неудачно завершенных тестов превысит 40% от общего количества.

2.7.3 Критерий возобновления работы

Тестирование возобновляется при исправлении обнаруженных на предыдущих этапах тестирования ошибок.

3 План тестирования

3.1 Блочные тесты

3.1.1 Входные данные для блочного тестирования

Для проведения блочного тестирования необходимо использовать несколько вариантов входных данных. Ожидаемый результат описываемых тестов будет сформулирован на основе приведенных примеров. В описании тестов будут использованы описанные ниже имена переменных.

String testSdp = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n

m=text 3840 RTP/AVP 0\r\n

c=IN 212.109.6.225 IP4\r\n

a=mid:text\r\n

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host\r\n

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host\r\n

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000\r\n"

String testSdp1 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag: \r\n

a=ice-pwd: 65i6644klsjvfoba3rsa1nrn31\r\n";

String testSdp2 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:\r\n";

String testSdp3 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n";

String testSdp4 = "v=0

o=ice4j.org 0 0 IN IP4 212.109.6.225

s=-

t=0 0

a=ice-options:trickle

a=ice-ufrag:baf8g1eo59h360

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31

m=text 3840 RTP/AVP 0

c=IN 212.109.6.225 IP4

a=mid:text

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000";

String testSdp5 = ;

String testSdp6 = "kadjfhlajkfhlaksjfhlaksfjsbcvhghakcbalsjdnakjdhbasdkjfalskfj";

String testSdp7 = "m=text 3840 RTP/AVP 0 c=IN 212.109.6.225 IP4

a=mid:text

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000";

3.1.2 Тест Б-1

Цель теста (описание): Проверка результата парсинга.

Тип теста: Общий

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaDescription = "m=text 3840 RTP/AVP 0
  c=IN 212.109.6.225 IP4
  a=mid:text
  a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host
  a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host
  a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000";

3.1.3 Тест Б-2

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается SDP-описание не имеющее раздела, описывающего медиа-данные и значения ice-ufrag

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp1

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaDescription = null;

3.1.4 Тест Б-3

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается SDP-описание не имеющее раздела, описывающего медиа-данные и значения ice-pwd

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp2

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = null;
- MediaDescription = null;

3.1.5 Тест Б-4

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается SDP-описание не имеющее раздела, описывающего медиа-данные.

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp3

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaDescription = null;

3.1.6 Тест Б-5

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается SDP-описание одной строкой.

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp4

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaDescription = null;

3.1.7 Тест Б-6

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается пустая строка.

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp5

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaDescription = null;

3.1.8 Тест Б-7

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается произвольная строка.

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp6

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaDescription = null;

3.1.9 Тест Б-8

Цель теста (описание): Проверка результата парсинга в случае, когда в качестве аргумента передается медиа-описание.

Тип теста: Негативный

Объект тестирования: метод parse(String sdp) класса SdpAnnounceParser

Входные данные: String testSdp7

Ожидаемый результат:

- ice-ufrag = null;

- ice-pwd = null;
- MediaDescription = null;

3.2 Интеграционные тесты

3.2.1 Входные данные для интеграционных тестов

Для проведения интеграционного тестирования в данном подразделе приведены примеры входных данных, которые будут использованы для формулировок ожидаемых значений. В качестве входных данных интеграционных тестов будут использоваться имена нижеследующих переменных. Переменная **testSdp** является эталонной. Значения всех остальных переменных — отклонения от формата SDP описания. **String testSdp** = "v=0\r\n
o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n
s=-\r\n
t=0 0\r\n
a=ice-options:trickle\r\n
a=ice-ufrag:baf8g1eo59h360\r\n
a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n
m=text 3840 RTP/AVP 0\r\n
c=IN 212.109.6.225 IP4\r\n
a=mid:text\r\n
a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host\r\n
a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host\r\n
a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000\r\n"

String testSdp1 = "v=0\r\n
o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n
s=-\r\n
t=0 0\r\n
a=ice-options:trickle\r\n
a=ice-ufrag: \r\n
a=ice-pwd: 65i6644klsjvfoba3rsa1nrn31\r\n";

String testSdp2 = "v=0\r\n
o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:\r\n";

String testSdp3 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n";

String testSdp4 = "v=0

o=ice4j.org 0 0 IN IP4 212.109.6.225

s=-

t=0 0

a=ice-options:trickle

a=ice-ufrag:baf8g1eo59h360

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31

m=text 3840 RTP/AVP 0

c=IN 212.109.6.225 IP4

a=mid:text

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000";

String testSdp5 = ;

String testSdp6 = "kadjfhlajkfhlaksjfhlaksfjsbcvhghakcbalsjdnakjdhbasdkjfalskfj";

String testSdp7 = "m=text 3840 RTP/AVP 0 c=IN 212.109.6.225 IP4

a=mid:text

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000";

String testSdp8 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n

m=text 3840 RTP/AVP 0\r\n

c=IN 212.109.6.225 IP4\r\n

a=mid:text\r\n

a=candidate:1 1 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host\r\n

a=candidate:2 1 2130706431 192.168.0.105 5000 typ host\r\n

a=candidate:3 1 1677724415 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000\r\n"

String testSdp9 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n

m=text 3840 RTP/AVP 0\r\n

c=IN 212.109.6.225 IP4\r\n

a=mid:text\r\n

a=candidate:1 1 udp fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host\r\n

a=candidate:2 1 udp 192.168.0.105 5000 typ host\r\n

a=candidate:3 1 udp 212.109.6.225 3840 typ srflx raddr 192.168.0.105 rport 5000\r\n"

String testSdp10 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n

m=text 3840 RTP/AVP 0\r\n

c=IN 212.109.6.225 IP4\r\n

a=mid:text\r\n

a=candidate:1 1 udp 2130706431 5000 typ host\r\n

a=candidate:2 1 udp 2130706431 5000 typ host\r\n

a=candidate:3 1 udp 1677724415 3840 typ srflx raddr 192.168.0.105 rport 5000\r\n"

String testSdp11 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n

m=text 3840 RTP/AVP 0\r\n

c=IN 212.109.6.225 IP4\r\n

a=mid:text\r\n

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ\r\n

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ\r\n

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ raddr 192.168.0.105 rport 5000\r\n"

String testSdp12 = "v=0\r\n

o=ice4j.org 0 0 IN IP4 212.109.6.225\r\n

s=-\r\n

t=0 0\r\n

a=ice-options:trickle\r\n

a=ice-ufrag:baf8g1eo59h360\r\n

a=ice-pwd:65i6644klsjvfoba3rsa1nrn31\r\n

m=text 3840 RTP/AVP 0\r\n

c=IN 212.109.6.225 IP4\r\n

a=mid:text\r\n

a=candidate:1 1 udp 2130706431 fe80:0:0:0:bee6:e4e3:12a9:6210 5000 typ host\r\n

a=candidate:2 1 udp 2130706431 192.168.0.105 5000 typ host\r\n

a=candidate:3 1 udp 1677724415 212.109.6.225 3840 typ srflx\r\n"

3.2.2 Тест И-1

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = [udp, udp, udp];
- Priority = [2130706431, 2130706431, 1677724415];
- CandidateType = [host, host, srflx];
- TransportAddress = [fe80:0:0:0:bee6:e4e3:12a9:6210, 192.168.0.105, 212.109.6.225];
- Port = [5000, 5000, 3840];
- ReflexiveAddress = [192.168.0.105];

3.2.3 Тест И-2

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP не содержит медиа-описания и ice-ufrag.

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp1, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.4 Тест И-3

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP не содержит медиа-описания и ice-pwd.

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp2, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = null;
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.5 Тест И-4

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP не содержит медиа-описания.
Тип теста: Негативный
Объект тестирования: схема взаимодействия методов № 1
Входные данные: String testSdp3, Agent agent
Косвенные входные данные: поток процесса установления соединения IceMediaStream
Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;

- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.6 Тест И-5

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrea
в случае, если входное SDP задано одной строкой.
Тип теста: Негативный
Объект тестирования: схема взаимодействия методов № 1
Входные данные: String testSdp4, Agent agent
Косвенные входные данные: поток процесса установления соединения IceMediaStream
Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.7 Тест И-6

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP задано пустой строкой.

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp5, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.8 Тест И-7

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP задано произвольным текстовым содержанием.

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp6, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.9 Тест И-8

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP задано медиа-описанием.
Тип теста: Негативный
Объект тестирования: схема взаимодействия методов № 1
Входные данные: String testSdp7, Agent agent
Косвенные входные данные: поток процесса установления соединения IceMediaStream
Ожидаемый результат:

- ice-ufrag = null;
- ice-pwd = null;
- MediaType = null;
- CountCandidate = null;
- Foundation = null;
- Transport = null;

- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.10 Тест И-9

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP без указания транспортного протокола кандидатов.
Тип теста: Негативный
Объект тестирования: схема взаимодействия методов № 1
Входные данные: String testSdp8, Agent agent
Косвенные входные данные: поток процесса установления соединения IceMediaStream
Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = null;
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.11 Тест И-10

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP без указания приоритета кандидатов кандидатов.

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp9, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = [udp, udp, udp];
- Priority = null;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.12 Тест И-11

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP без указания транспортных адресов кандидатов.

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 1

Входные данные: String testSdp10, Agent agent

Косвенные входные данные: поток процесса установления соединения IceMediaStream

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = [udp, udp, udp];
- Priority = [2130706431, 2130706431, 1677724415];;
- CandidateType = null;
- TransportAddress = null;
- Port = null;
- ReflexiveAddress = null;

3.2.13 Тест И-12

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP без указания типов кандидатов.
Тип теста: Негативный
Объект тестирования: схема взаимодействия методов № 1
Входные данные: String testSdp11, Agent agent
Косвенные входные данные: поток процесса установления соединения IceMediaStream
Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = [udp, udp, udp];

- Priority = [2130706431, 2130706431, 1677724415];;
- CandidateType = null;
- TransportAddress = [fe80:0:0:0:bee6:e4e3:12a9:6210, 192.168.0.105, 212.109.6.225];
- Port = [5000, 5000, 3840];
- ReflexiveAddress = null;

3.2.14 Тест И-13

Цель теста (описание): Проверка результата инициализации атрибутов потока IceMediaStrean
в случае, если входное SDP без указания транслируемого адреса.
Тип теста: Негативный
Объект тестирования: схема взаимодействия методов № 1
Входные данные: String testSdp12, Agent agent
Косвенные входные данные: поток процесса установления соединения IceMediaStream
Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = [udp, udp, udp];
- Priority = [2130706431, 2130706431, 1677724415];
- CandidateType = [host, host, srflx];
- TransportAddress = [fe80:0:0:0:bee6:e4e3:12a9:6210, 192.168.0.105, 212.109.6.225];
- Port = [5000, 5000, 3840];
- ReflexiveAddress = null;

3.2.15 Тест И-14

Цель теста (описание): Проверка того, что входной аргумент — SDP-описание не изменяется в процессе передачи данных. Метод urlConnection запускается в двух экземплярах: urlConnection(String testSdp, String os1) и urlConnection(String testSdp, String os2).

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 2

Входные данные: String testSdp12, String os1 = "android String os2 = "linux"

Ожидаемый результат:

Строковые значения, возвращенные вызванными методами эквивалентны testSdp

3.2.16 Тест И-15

Цель теста (описание): Проверка того, что входной аргумент — SDP-описание, не передается между одинаковыми устройствами. Методы запускаются параллельно в двух экземплярах: urlConnection(String testSdp, String os1) и urlConnection(String testSdp, String os2).

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 2

Входные данные: String testSdp12, String os1 = "android String os2 = "android"

Ожидаемый результат:

Строковые значения, возвращенные вызванными методами — пустая строка

3.2.17 Тест И-16

Цель теста (описание): Проверка того, что входной аргумент — SDP-описание, не передается между одинаковыми устройствами. Методы запускаются параллельно в двух экземплярах: urlConnection(String testSdp, String os1) и urlConnection(String testSdp, String os2).

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 2

Входные данные: String testSdp12, String os1 = "linux String os2 = "linux"

Ожидаемый результат:

Строковые значения, возвращенные вызванными методами — пустая строка.

3.2.18 Тест И-17

Цель теста (описание): Проверка того, что входной аргумент — SDP-описание, не передается при значениях переменной String os, отличной от значений "linux"и "android".

Тип теста: Негативный

Объект тестирования: схема взаимодействия методов № 2

Входные данные: String testSdp12, String os1 = "abracadabra String os2 = "abracadabra"

Ожидаемый результат:

Строковые значения, возвращенные вызванными методами — "Нет данных об устройстве!"

3.2.19 Тест И-18

Цель теста (описание): Проверка того, что входной аргумент — SDP-описание, не передается при значениях переменной String os, отличной от значений "linux"и "android".

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 3

Входные данные: String testSdp12, String os1 = "abracadabra String os2 = "abracadabra"

Ожидаемый результат:

Строковые значения, возвращенные вызванными методами — "Нет данных об устройстве!"

3.2.20 Тест И-19

Цель теста (описание): Проверка изменения состояния контрольного списка кандидатов при запуске процесса соединения. Изъятие данных о состоянии контрольного списка из потока IceMediaStream

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 3

Косвенные входные данные: Объект класса Agent, поток процесса установления соединения IceMediaStream, извлекаемый из потока объект класса CheckList checkList

Ожидаемый результат:

checkList.getState() == RUNNING

3.2.21 Тест И-20

Цель теста (описание): Проверка изменения состояния контрольного списка кандидатов при запуске процесса соединения в случае, когда удаленный агент не отвечает. Изъятие данных о состоянии контрольного списка из потока IceMediaStream. После запуска процесса установления соединения **ожидание: 10000 мс**.

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 3

Косвенные входные данные: Поток процесса установления соединения IceMediaStream, извлекаемый из потока объект класса CheckList checkList, String testSdp

Ожидаемый результат:

checkList.getState() == FAILED

3.2.22 Тест И-21

Цель теста (описание): Проверка изменения состояния контрольного списка кандидатов при запуске процесса соединения в случае, когда присутствует сетевое взаимодействие с удаленным узлом. Изъятие данных о состоянии контрольного списка из потока IceMediaStream. После запуска процесса установления соединения **ожидание: 6000 мс**.

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 3

Косвенные входные данные: Поток процесса установления соединения IceMediaStream, извлекаемый из потока объект класса CheckList checkList, String testSdp

Ожидаемый результат:

checkList.getState() == IN_PROGRESS

3.2.23 Тест И-22

Цель теста (описание): Проверка изменения состояния контрольного списка кандидатов при запуске процесса соединения в случае, когда присутствует сетевое взаимодействие с удаленным узлом. Тест выполняется с параллельным запуском аналогичного, с точки зрения логики, приложения на устройстве Linux. Изъятие данных о состоянии контрольного списка из потока IceMediaStream. После запуска процесса установления соединения **ожидание: 10000 мс.**

Тип теста: Общий

Объект тестирования: схема взаимодействия методов № 3

Косвенные входные данные: Поток процесса установления соединения IceMediaStream, извлекаемый из потока объект класса CheckList checkList, String testSdp

Ожидаемый результат:

checkList.getState() == COMPLETED

3.3 Аттестационные тесты

3.3.1 Тест А-1

Цель теста: Проверка соответствия работы приложения ФТ1.

Описание теста: Для тестирования требуется мобильное устройство с операционной системой Android и устройство, управляемое операционной системой Linux. Приложение запускается на мобильном устройстве, подключенном с помощью USB к компьютеру с заранее запущенной средой разработки Android Studio. Одновременно, с разницей не более 3 секунд, запускаются программы на устройствах Linux и Android. Устройства подключены к одной точке доступа Wi-Fi.

Ожидаемый результат:

В консоли выполнения приложения среды разработки AndroidStudio должна появиться надпись:

Checklist of stream is COMPLETED

ICE state changed from COMLETED to TERMINATED

3.3.2 Тест А-2

Цель теста: Проверка соответствия работы приложения ФТ2.

Описание теста: Для тестирования требуется мобильное устройство с операционной системой Android и устройство, управляемое операционной системой Linux. Приложение запускается на мобильном устройстве, подключенном с помощью USB к компьютеру с заранее запущенной средой разработки Android Studio. Одновременно, с разницей не более 3 секунд, запускаются программы на устройствах Linux и Android. Мобильное устройство подключено к мобильной сети, устройство Linux подключено в сети Wi-Fi. Точки доступа должны быть в разных подсетях за NAT.

Ожидаемый результат:

В консоли выполнения приложения среды разработки AndroidStudio должна появиться надпись:

```java
@Test
public void checkListRunningTest() throws Throwable {
    CheckList checkList = checkListInit();
    //Thread.sleep(20000);
    System.out.println("CHECKLIST: " + checkList);
    assertEquals(CheckListState.RUNNING, checkList.getState());
    assertEquals(CandidatePairState.WAITING, checkList.get(0).getState());
    assertEquals(CandidatePairState.WAITING, checkList.get(1).getState());
    assertEquals(CandidatePairState.WAITING, checkList.get(2).getState());
    assertFalse(checkList.allChecksCompleted());
}

@Test
public void checkListFailedTest() throws Throwable {
    CheckList checkList = checkListInit();
    System.out.println("CHECKLIST: " + checkList);
    Thread.sleep( millis: 10000);
    assertEquals(CheckListState.FAILED, checkList.getState());
    assertEquals(CandidatePairState.FAILED, checkList.get(0).getState());
    assertEquals(CandidatePairState.FAILED, checkList.get(1).getState());
    assertEquals(CandidatePairState.FAILED, checkList.get(2).getState());
    assertTrue(checkList.allChecksCompleted());
}
```

Рис. 8: Фрагмент теста И-19

Checklist of stream is COMPLETED

ICE state changed from COMLETED to TERMINATED

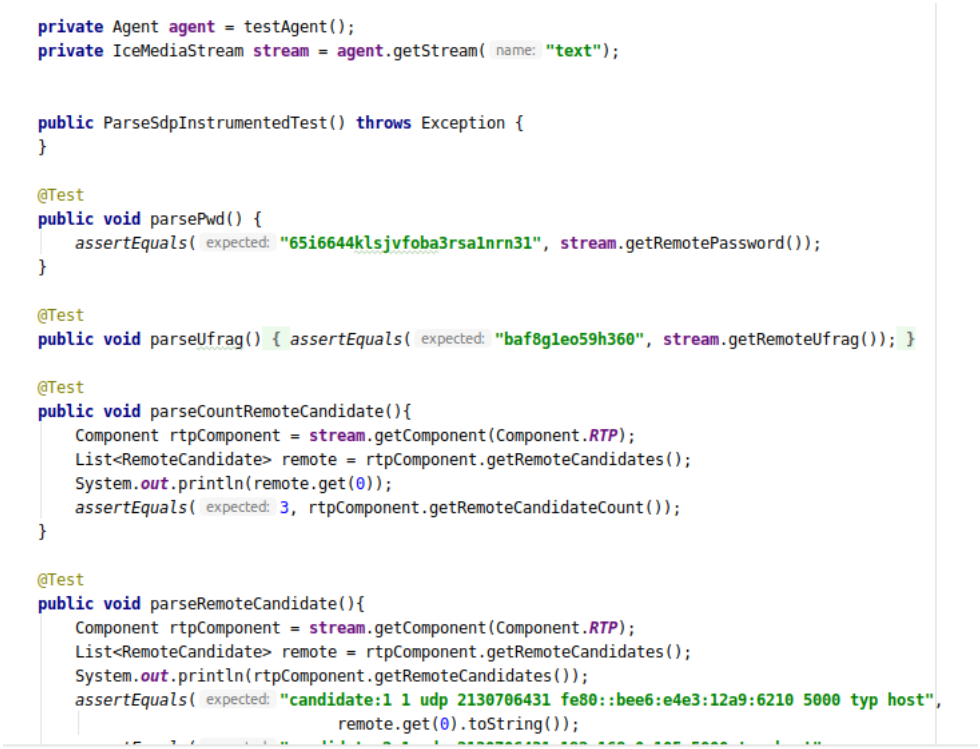
3.4 Примеры тестов

Примеры выполнения тестов И-1 и И-19 приведены на рисунках 8 и 9.

4 Отчет о проведении тестирования

4.1 Блочное тестирование

№ теста	Дата	Результат	Отчет об ошибке
Б-1	7.12.20	Пройден	
Б-2	7.12.20	Пройден	
Б-3	7.12.20	Пройден	
Б-4	7.12.20	Пройден	
Б-5	7.12.20	Пройден	
Б-6	7.12.20	Пройден	
Б-7	7.12.20	Пройден	
Б-8	7.12.20	Пройден	



```
private Agent agent = testAgent();
private IceMediaStream stream = agent.getStream( name: "text");


public ParseSdpInstrumentedTest() throws Exception {
}

@Test
public void parsePwd() {
    assertEquals( expected: "65i6644klsjvfoba3rsa1nrn31", stream.getRemotePassword());
}

@Test
public void parseUfrag() { assertEquals( expected: "baf8gleo59h360", stream.getRemoteUfrag()); }

@Test
public void parseCountRemoteCandidate(){
    Component rtpComponent = stream.getComponent(Component.RTP);
    List<RemoteCandidate> remote = rtpComponent.getRemoteCandidates();
    System.out.println(remote.get(0));
    assertEquals( expected: 3, rtpComponent.getRemoteCandidateCount());
}

@Test
public void parseRemoteCandidate(){
    Component rtpComponent = stream.getComponent(Component.RTP);
    List<RemoteCandidate> remote = rtpComponent.getRemoteCandidates();
    System.out.println(rtpComponent.getRemoteCandidates());
    assertEquals( expected: "candidate:1 1 udp 2130706431 fe80::bee6:e4e3:12a9:6210 5000 typ host",
                                remote.get(0).toString());
```

Рис. 9: Фрагмент теста И-1

4.2 Интеграционное тестирование

№ теста	Дата	Результат	Отчет об ошибке
И-1	15.11.20	Не пройден	Отчет об ошибке № 1
И-1	1.12.20	Пройден	
И-2	7.12.20	Пройден	
И-3	7.12.20	Пройден	
И-4	7.12.20	Пройден	
И-5	7.12.20	Пройден	
И-6	7.12.20	Пройден	
И-7	7.12.20	Пройден	
И-8	7.12.20	Пройден	
И-9	7.12.20	Пройден	
И-10	7.12.20	Пройден	
И-11	7.12.20	Пройден	
И-12	7.12.20	Пройден	
И-13	7.12.20	Пройден	
И-14	7.12.20	Пройден	

№ теста	Дата	Результат	Отчет об ошибке
И-15	7.12.20	Пройден	
И-16	7.12.20	Пройден	
И-17	7.12.20	Пройден	
И-18	7.12.20	Пройден	
И-19	7.12.20	Пройден	
И-20	7.12.20	Пройден	
И-21	7.12.20	Пройден	
И-22	7.12.20	Пройден	

4.3 Аттестационное тестирование

№ теста	Дата	Результат	Отчет об ошибке
А-1	10.12.20	Пройден	
А-2	10.12.20	Не пройден	Отчет об ошибке № 2

5 Журнал ошибок

5.1 Отчет об ошибке № 1

Описание: Тест, проверяющий инициализацию атрибутов на основе парсинга, реализованного вызываемой библиотечной функцией.

Ожидаемый результат:

- ice-ufrag = "baf8g1eo59h360";
- ice-pwd = "65i6644klsjvfoba3rsa1nrn31";
- MediaType = "text";
- CountCandidate = 3;
- Foundation = [1, 2, 3];
- Transport = [udp, udp, udp];
- Priority = [2130706431, 2130706431, 1677724415];
- CandidateType = [host, host, srflx];
- TransportAddress = [fe80:0:0:0:bee6:e4e3:12a9:6210, 192.168.0.105, 212.109.6.225];

- Port = [5000, 5000, 3840];
- ReflexiveAddress = [192.168.0.105];

Фактический результат: при запуске теста возникает следующее исключение:
java.lang.NullPointerException: Attempt to invoke virtual method 'java.lang.Object java.util.Hashtable.get(java.lang.Object)' on a null object reference
Данная ошибка является багом открытой библиотеки *java-sdp-nist-bridge*, вызывающей метод SIP-клиента, который неработоспособен на Android.
Статус ошибки: устранена

5.2 Отчет об ошибке № 2

Описание: Тест, проверяющий возможность подключения удаленных устройств в разных подсетях за NAT.
Ожидаемый результат: В консоли выполнения приложения среды разработки AndroidStudio должна появиться надпись:
Checklist of stream is COMPLETED
ICE state changed from COMLETED to TERMINATED
Фактический результат:
Checklist for stream FAILED
ICE state is FAILED
Ошибка возникает из-за превышения лимита ожидания ответа от удаленного устройства.
Статус ошибки: Не устранена

6 Покрытие кода тестами

Процентное значение покрытия кода тестами составило 42% классов (3 из 6, при условии что в подразделе 2.3 текущего отчета не рассматривался класс MainActivity).

Неубедительные цифры покрытия обясняются отношением реализованных методов к тестированию, описанном в подразделе 2.2. Следует учитывать, что тестирование библиотечных методов (класса SdpAnnounseParser, библиотеки ice4j) в покрытии кода не учитываются.

Результат, полученный в среде разработки Android Studio приведен на рисунке 10.

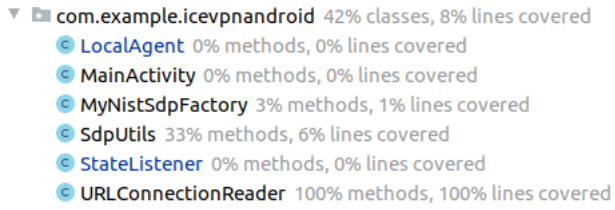


Рис. 10: Покрытие кода тестами

7 Заключение

В процессе тестирования было выявлено две ошибки, занесенные в журнал тестирования.

Первая ошибка (отчет об ошибке № 1) являлась багом используемой библиотеки с открытым исходным кодом, которую удалось исправить.

Вторая ошибка (отчет об ошибке № 2), связанная с особенностями установления соединения удаленных устройств не исправлена. Реализованная программа не соответствует ФТ.2

Исходя из процесса тестирования, и анализируя получившиеся значения покрытия кода теста (42% классов) можно прийти к выводу, что для тестирования подобного рода приложений необходимо создавать специальное окружение, способное в полной мере протестировать необходимые функциональности.

Список литературы

1. Олифер В.Г, Олифер Н.А *Компьютерные сети. Принципы, технологии, протоколы: Юбилейное издание* СПб.: Питер, 2020. — 1008 с.

2. Baruch Sterman, Ph.D. *NAT Traversal in SIP* — 16 с.

3. RFC 5245 Interactive Connectivity Establishment (ICE):[Электронный ресурс]. - Режим доступа: https://tools.ietf.org/html/rfc5245 (дата обращения 10.12.20)

4. RFC 4566 Session Description Protocol (SDP):[Электронный ресурс]. - Режим доступа: https://tools.ietf.org/html/rfc4566 (дата обращения 10.12.20)