Глава 1

Разработка системы моделирования.

1.1 Разработка архитектуры приложения.

Разрабатываемая система позволяет пользователю строить модель локальной сети, используя объекты, имитирующие различное сетевое оборудование, проводить эксперименты в различных режимах работы, анализировать полученные результаты. Каждая из этих возможностей может быть реализована посредством подсистемы, которые являются компонентами проектируемой системы. Таким образом можно выделить следующие подсистемы.

- Подсистема моделирования. Данная подсистема отвечает за правильное взаимодействие компонентов модели, учитывая их параметры.
- Подсистема управления отвечает за сохранение(загрузку) параметров моделей, создание модели сети, а так же предоставляет методы для ее функционирования.
- Подсистема мониторинга отвечает за сбор данных в процессе моделирования и их анализ

Взаимодействие данных подсистем представлено на рисунке ().

1.1.1 Подсистема моделирования.

Для реализации подсистемы моделирования необходимо описать взаимодействие различных компонент, входящих в ее состав. Такими компонентами являются модели устройств, протоколов, приложений. Объединение сетевых устройств в проектируемой системе посредством моделей сетевых соединений. Модель локальной сети представляет собой совокупность моделей устройств и их взаимодействия.

Канал связи.

Для имитации передачи данных могут быть использованы различные модели канала связи. При использовании модели идеального канала без помех единственной характеристикой такого соединения является задержка при передачи данных. В случае необходимости анализа передачи данных в каналах связи с помехами необходима реализация алгоритма генерации помехи. Используемая в системе модель канала связи представлена на рисунке 1.1.

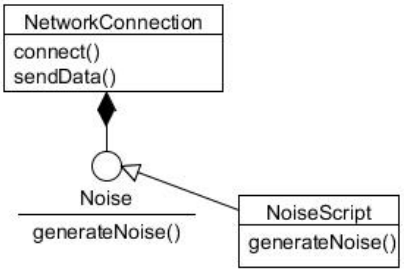

Рис. 1.1: Диаграмма классов модели канала связи.

Класс NoiseScript описывает алгоритм генерации помехи и связан с моделью канала связи через интерфейс Noise. Такая архитектура позволяет изменять использую модель канала связи без внесения изменений в архитектуру приложения.

Протоколы.

Для использования при моделировании различных сетевых протоколов они разбиты на четыре группы. К первой группе относятся протоколы, обрабатывающие данные приложений или вышестоящих протоколов. Каждый протокол, относящийся к этой группе, выполняет разделение данных на части и дополняет их служебными полями. В проектируемой модели, использующей стек протоколов TCP/IP, для каждого протокола эти фрагменты данных называются поразному. Для Ethernet - кадры, для IP - пакеты, для TCP - сегменты. Структура взаимодействия протоколов отражена на рисунке 1.2.

Анализ служебных полей фрагмента данных описывается в классе, реализующем интерфейс ProtocolScript. Данный механизм позволяет описывать различные протоколы для использования в системе, однако осложняет архитектуру приложения и описание процесса взаимодействия протокола.

Второй группой протоколов, выделенных на этапе моделирования, являются протоколы управления средой. В стеке протоколов TCP/IP таким протоколом является протокол MAC-подуровня, уровня передачи данных. Отличительной особенностью этих протоколов является связь с устройством передачи данных

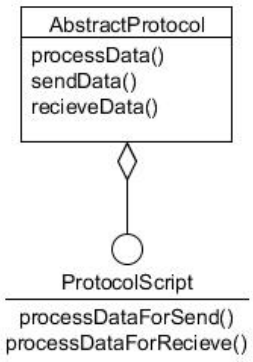

Рис. 1.2: Диаграмма классов модели протокола.

и возможность контроля за передачей. На рисунке 1.3 изображена диаграмма классов, используемая для описания этой группы протоколов.

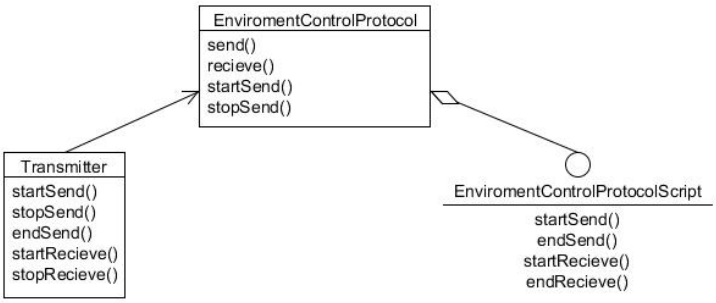

Рис. 1.3: Диаграмма классов модели протокола управления средой.

Интерфейс EnviromentControlProtocolScript предоставляет возможность описания различных протоколов управления средой. Для разрабатываемой модели будет реализован метод доступа с контролем несущей и обнаружением коллизий. Данный метод не используется в современных сетях с дуплексным каналом обмена информацией, однако необходим при использовании полудуплексного режима обмена данными.

Алгоритм, описанный в классе EnviromentControlProtocolScript, определяет возможно ли начать передачу в данный момент времени, возможно ли считать передачу или прием данных успешными и, в некоторых случаях, определяет задержку передачи данных.

Устройство передачи данных не выполняет никаких действий по контролю за процессом передачи данных, и является моделью передатчика, который генерирует необходимый сигнал и передает его в канал связи.

Третьей группой протоколов являются служебные протоколы. В стеке про-

токолов TCP/IP таким протоколом является ARP. Он используется для установления соответствия между физическими адресами и адресами, используемыми на более высоком уровне модели. Для использования этого протокола необходимо внести изменения в архитектуру протоколов первой группы. Для этого добавляется еще один метод для отправки данных, использование которого сопряжено с запросом адреса, используемого на более низком уровне модели, соответствующего адресу на данном уровне. Взаимодействие протоколов первой группы и служебного протокола изображено на рисунке 1.4.

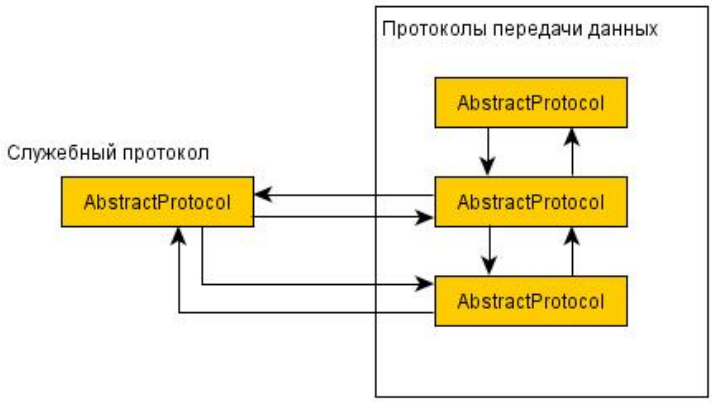

Рис. 1.4: Схема взаимодействия протоколов передачи данных и служебного протокола.

Протоколы первой и третьей группы являются экземплярами одного и того же класса, который был расширен методами для операций по установлению соответствия адресов. Для подобного рода операций, служебные протоколы связаны с протоколами более низкого уровня, что позволяет им взаимодействовать независимо от адресов более высоких уровней модели.

К четвертой группе протоколов относятся протоколы, связанные с приложениями. Данный вид протоколов предоставляет приложению интерфейс для обмена данными посредством локальной сети. В рамках используемой модели это протоколы четвертого уровня, содержащие в себе операции транспортного уровня, уровня представления данных и сеансового уровня. Учитывая сложность данных процессов, при реализации протокола TCP было принято решение выделить необходимые операции в отдельную подсистему. Диаграмма классов данной подсистемы представлена на рисунке 1.5.

Интерфейс UserTransportFace предоставляет приложению методы для взаимодействия с транспортной подсистемой. Класс ConnectionControlSubsystem отвечает за установление, поддержание и закрытие TCP соединения. В этом классе определяются алгоритмы обработки данных в зависимости от состояния в котором находится соединение. Схема состояний и переходов представлена на рисунке 1.6. На данном уровне абстракции используется понятие сегмен-

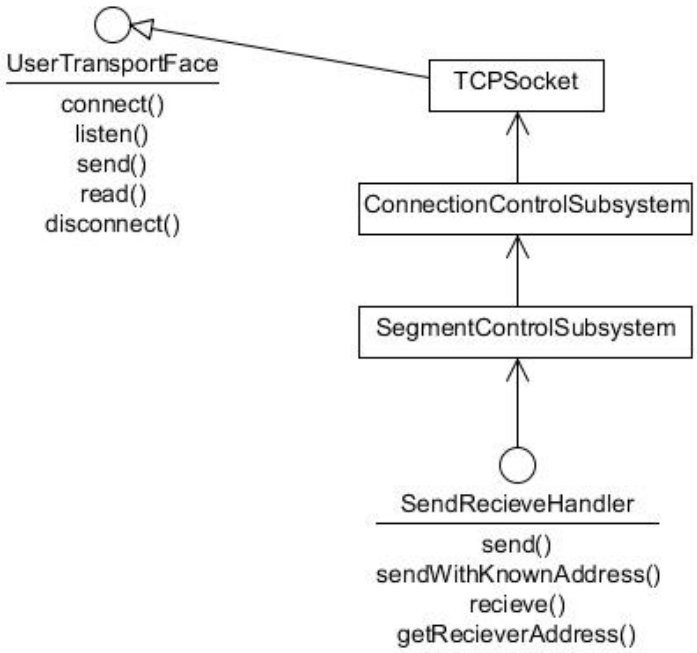

Рис. 1.5: Диаграмма классов транспортной подсистемы.

та данных. Начальное состояние системы - CLOSED. Управление соединением происходит посредством команд, поступающих от приложения и принятых служебных сегментов.

Типичный случай клиента, активно соединяющегося с сервером показан на жирными линиями - сплошными для клиента, прерывистыми для сервера. Тонкие линии обозначают необычные последовательности событий. Каждая линия маркирована парой событие/действие. Событие может представлять собой либо обращение пользователя к системной процедуре(CONNECT, LISTEN, SEND или CLOSE), либо прибытие сегмента(SYN, FIN, ACK, RST), либо, в одном случае, окончание периода ожидания, равного двойному времени жизни пакета. Действие может состоять в отправке управляющего сегмента(SYN, FIN или RST). Системные процедуры имеют следующее назначение:

- LISTEN - переводит систему в состояние ожидания соединений.
- CONNECT - попытка активного создания соединения.
- SEND - отправка данных.
- CLOSE - закрытие соединения.

Управляющие сегменты имеют следующее назначение:

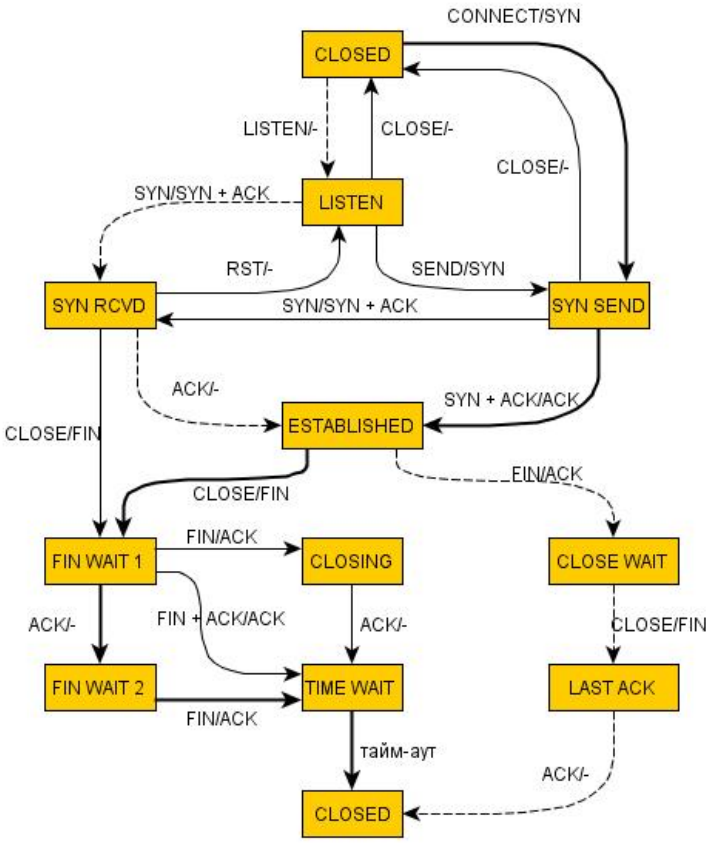

Рис. 1.6: Схема конечного автомата TCP-соединения.

- SYN - сегмент, отправляемый при установлении соединения.
- FIN - сегмент, отправляемый при закрытии соединения.
- RST - сегмент, отправляемый при сбросе соединения.
- ACK - сегмент, отправляемый для подтверждения получения определенного сегмента.

Назначение того или иного сегмента определяется наличием или отсутствием того или иного флага в TCP-заголовке сегмента.

Класс SegmentControlSubSystem отвечает за гарантированную доставку сообщений. Это достигается за счет отправки сегмента данных до тех пор, пока не будет получено подтверждение об успешном его приеме.

Модели приложений.

Приложения, использованные в системе для моделирования действий пользователя, могут использовать протоколы разных уровней. Алгоритмы маршрути-

зации, используемые в коммутаторах и маршрутизаторах работают на сетевом уровне, как и алгоритмы генерации трафика. Модели уязвимостей используют транспортную подсистему.

Приложение, используемое в коммутаторе, представляет собой алгоритм определения соединения, через которое будет направлен полученный пакет. Данный алгоритм имеет два режима работы. Первый - это режим обучения. В этом режиме составляется таблица соответствия физического адреса и порта коммутатора, к которому подключен абонент. При этом полученный пакет передается через все активные порты. Второй режим работы использует построенную на первом этапе таблицу маршрутизации для передачи пакета только указанному получателю. Исключением является широковещательный трафик.

Для моделирования процесса атаки на рабочую станцию, связанного с наличием уязвимости в приложении используем описанную модель активов и уязвимостей. Уязвимость отличается от актива только тем, что в случае удачной атаки, система считается скомпрометированной в то время, как удачная атака на один из активов подразумевает только получение некоторой информации. Для атаки на приложение злоумышленнику необходимо установить TCP-соединение с целевой рабочей станцией и послать соответствующий запрос. Приложение обрабатывает запрос и, с учетом параметров, влияющих на вероятность успеха, либо возвращает некоторую информацию, либо пустое значение, что означает, что атака была неудачной. Диаграмма классов, описывающая взаимодействие моделей приложений и транспортной подсистемы показана на рисунке 1.7

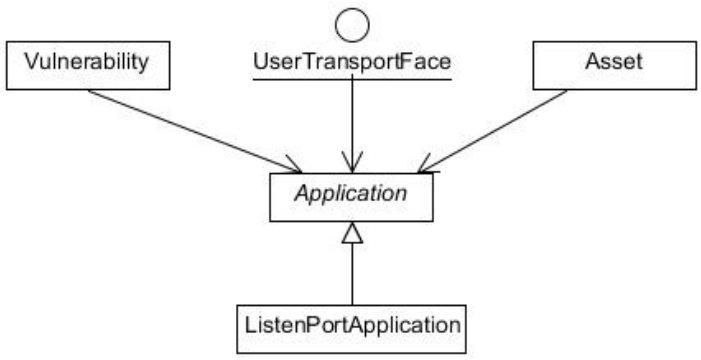

Рис. 1.7: Диаграмма классов модели пользовательского приложения.

Абстрактный класс Application отвечает за загрузку и передачу данных уязвимостям(Vulnerability) и активам(Asset). Для реализации некоторого приложения, алгоритм его действий необходимо реализовать в классе-наследнике класса Application. На приведенном рисунке таким классом является ListenPortApplication. Данный класс создает TCP-сокет и переводит его в режим ожидания входящего соединения. Подобное поведение наиболее удобно для демонстрации процесса атаки, однако возможна реализация и более сложных алгоритмов.

Модель поведения злоумышленника так же описывается в классе-наследнике от Application. При этом необходимо описать процесс формирования данных, которые будут отправлены приложению-жертве. В эти данные входит так же и параметры атакующего, такие как мотивация и уровень знаний.

Каждая модель уязвимости или актива содержит таблицу, в которой хранятся вероятности успеха атаки в зависимости от характеристик атакующего и таблицу необходимой для успешной атаки информации, которая должна содержаться в сообщении злоумышленника.

Таким образом, диаграмма классов, описывающая структуру модели сетевого устройства может быть изображена следующим образом(рисунок 1.8).

1.2 Описание работы приложения.

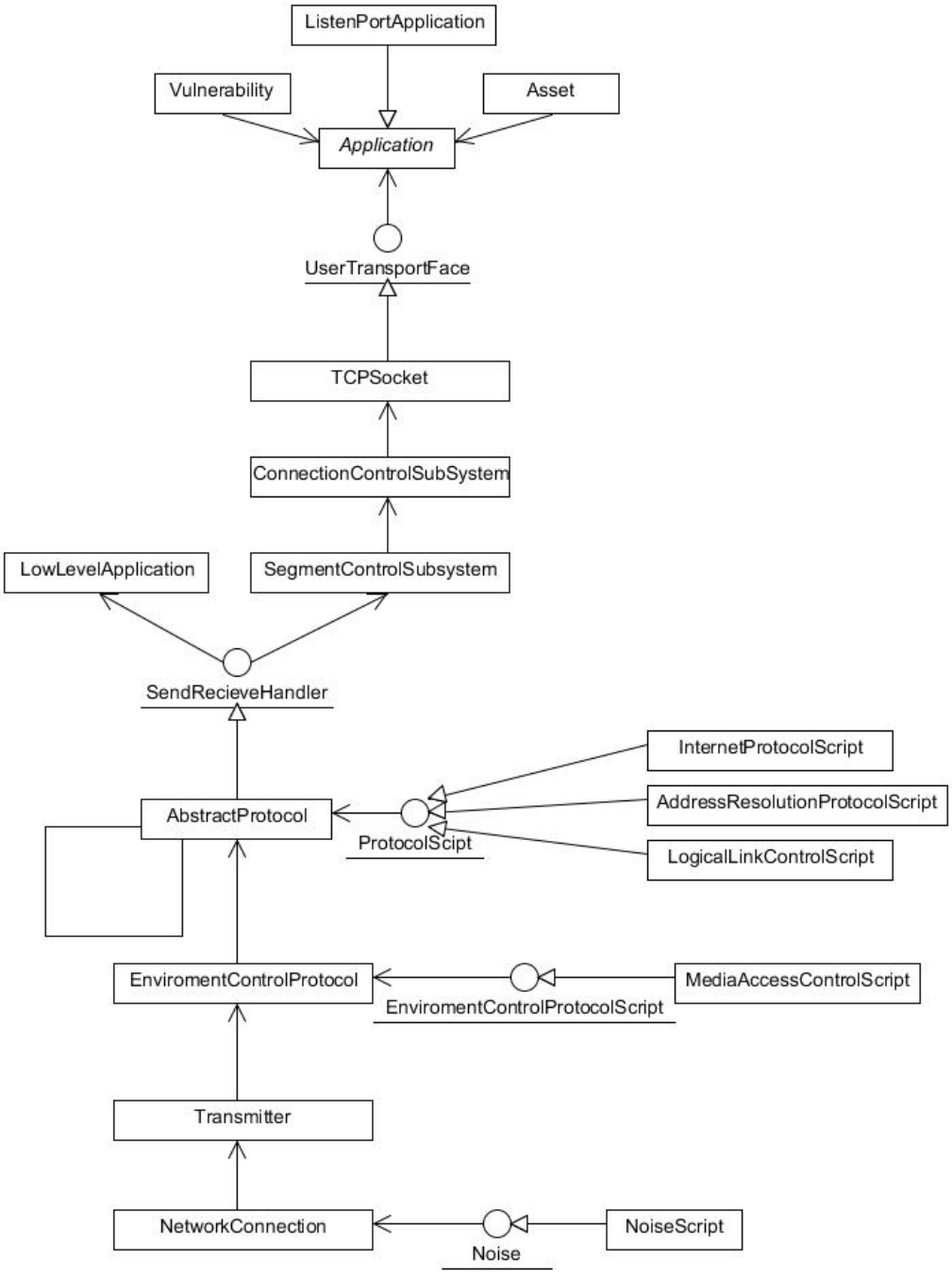

Рис. 1.8: Диаграмма классов модели сетевого устройства.