ИМИТАЦИЯ ПОВЕДЕНИЯ КОМПЬЮТЕРНОЙ СИСТЕМЫ С ПОМОЩЬЮ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

DOI: 10.36724/2072-8735-2021-15-5-29-37

Шелухин Олег Иванович,
Московский технический университет связи и информатики, Москва, Россия, ***sheluhin@mail.ru***

Шариков Алексей Юрьевич,
Московский технический университет связи и информатики, Москва, Россия,
sharikov.it@gmail.com

Manuscript received 26 November 2020
Accepted 12 January 2021

Ключевые слова: *Компьютерное моделирование, имитационное моделирование, временные ряды, компьютерные системы, искусственные нейронные сети, генеративно-состязательные нейронные сети, GAN*

Рассматривается проектирование и реализация имитационной модели компьютерной системы (КС) с применением искусственных нейронных сетей (ИНС). Целью реализации является создание простой в освоении и реализации имитационной модели, позволяющей моделировать как нормальные, так и аномальные процессы в КС. Разработанная имитационная модель представляет собой программное обеспечение, состоящее из совокупности программных модулей объединённых с применением принципов клиент-серверной архитектуры, что позволяет использовать модель как в централизованном, так и в распределённом режимах работы. Модель позволяет имитировать поведение КС с различными топологиями: звезда, дерево и комбинации данных топологий. Основные элементы модели реализованы в виде четырёх модулей, выполняющих свои определённые роли: агент, генерирующий данные; пассивный элемент сети, передающий данные с возможными задержками и потерями; активный элемент сети, обрабатывающий и передающий поступающие на него данные, и ядро - центральный элемент модели, принимающий данные и отправляющий их в дополнительные модули для анализа. За счёт модульности обеспечивается высокий потенциал для дальнейшей модификации имитационной модели путём добавления новых рабочих модулей. Использование в модели модуля генерации данных на основе генеративно-состязательной ИНС позволяет генерировать данные, необходимые для моделирования поведения исследуемой КС. На основании вычисления евклидова расстояния между матрицами переходных вероятностей исходных и сгенерированных данных показано, что процессы, генерируемые с помощью разработанной имитационной модели, по характеру поведения подобны реальным. Разработанная модель может применяться для исследования работы реальных КС в том числе для имитации аномального поведения.

Информация об авторах:
Шелухин Олег Иванович, *д.т.н., профессор, заведующий кафедрой информационной безопасности, Московский технический университет связи и информатики, Москва, Россия*
Шариков Алексей Юрьевич, *магистрант, Московский технический университет связи и информатики, Москва, Россия*

Для цитирования:
Шелухин О.И., Шариков А.Ю. Имитация поведения компьютерной системы с помощью искусственных нейронных сетей // T-Comm: Телекоммуникации и транспорт. 2021. Том 15. №5. С. 29-37.

For citation:
Sheluhin O.I., Sharikov A.Yu. (2021) Simulation of the behavior of a computer system using artificial neural networks. *T-Comm*, vol. 15, no.5, pp. 29-37. (*in Russian*)

Введение

Компьютерное моделирование является одним из наиболее перспективных способов исследования работы и поведения сложных КС. При подобных исследованиях обычно применяются два вида моделей: аналитические, в которых поведение элементов задаётся в виде явно заданных функциональных зависимостей (различных уравнений и их систем), и имитационные, в которых используются математические модели, имитирующие поведение настоящих объектов исследования [1]. Аналитическое моделирование для исследования сложных систем практически не применяется из-за трудностей явного выделения функциональных зависимостей. Соответственно, для исследования поведения КС наилучшим образом подходит имитационное моделирование.

Имитационное моделирование применяется в тех случаях, когда затраты на использование реальных объектов больше, чем на реализацию модели; когда моделируемые процессы могут представлять угрозу при их реализации в рамках реальных объектов (например, имитационное моделирование критических состояний в химической промышленности); когда необходимо исследовать поведение системы в течение долгого периода времени и когда в исследуемой системе имеется множество различных параметров, поведение которых сложно рассчитать аналитическими способами, а также тяжело предусмотреть все возможные варианты поведения КС.

При проведении имитационного моделирования сложных КС обычно исследуемая система и явления, происходящие в ней, представляются в виде отдельных модулей или подсистем, что позволяет легко менять конфигурацию исследуемой системы и паттерны её поведения. Благодаря этому, имитационное моделирование позволяет всесторонне изучить исследуемую КС.

Имитационная модель представляет собой систему взаимодействующих логических элементов (сущностей), каждый из которых олицетворяет физический или логический элемент реальной компьютерной информационной сети. Основная цель создания имитационной модели – предоставить удобный и легко настраиваемый инструмент для дальнейшего исследования поведения компьютерных систем.

Предшествующие работы

В работе [2] при помощи имитационного моделирования на основе сетей Петри исследуется процесс передачи трафика в вычислительных сетях. Авторы дополняют свою имитационную модель аналитическими методами для того, чтобы «оптимальным образом решать широкий спектр задач моделирования». Для построения модели используются три типа данных: сведения о топологии сети, о маршрутах и характеристиках потоков трафика и о вычислительных мощностях сетевых устройств. Каждое сетевое устройство в системе моделируется как подсеть Петри.

В работе [3] описан подход к созданию системы для имитационного моделирования атак на компьютерные сети с мультиагентной архитектурой, где основное внимание уделяется концептуальному обоснованию выбранного подхода, спецификации основных компонентов, составляющих модель атаки, спецификации компонентов и их взаимодействию в процессе моделирования.

Имитационное моделирование не всегда сопровождается реализацией собственных моделей, для него могут применяться и уже готовые решения. Так, в [4] описано моделирование вычислительной сети в среде имитационного моделирования Cisco Packet Tracer. В работе имитационное моделирование применяется для исследования работы сети с точки зрения безопасности и на основе моделирования выявляются актуальные уязвимости сети.

В работе [5] предлагается метод, использующий имитационное моделирование эталонного состояния и поведения SCADA-систем, предназначенный для обнаружения заражения инфраструктуры SCADA-систем вредоносным ПО. В статье приведено подробное исследование проблем безопасности SCADA-систем и то, как имитационное моделирование предоставляет возможность раннего обнаружения вторжений в систему.

В данной работе приводится пример реализации имитационной модели компьютерной системы, использующей генерацию данных с применением искусственных нейронных сетей, в которой каждая моделируемая сущность представляет собой самостоятельный процесс. Взаимодействие между сущностями системы реализуется посредством сетевых сокетов в централизованном либо распределённом режимах запуска.

Описание имитационной модели

В рамках рассматриваемой задачи КС представляет собой систему, осуществляющую генерацию данных и их передачу данных между своими компонентами: компьютерами, серверами, активными сетевыми устройствами и иным оборудованием с учётом свойств среды передачи. Схема соединения всех этих компонентов в единую сеть представляет собой топологию сети. Модель позволяет имитировать топологии дерево и звезда, а также их комбинацию, которые являются наиболее часто встречающимися в современных КС.

Имитационная модель КС представляет собой набор элементов, сгруппированных в модули, взаимодействующие между собой и выполняющие функции генерации данных, их передачи и обработки, имитируя подобные процессы в реальных системах. Разработанная для имитации поведения компьютерной системы модель, состоит из следующих модулей: ядро модели, агрегирующее и обрабатывающее все данные, генерируемые моделью; модули сетевой структуры, имитирующие поведение различных элементов сети, и модуль генерации (имитации) данных, позволяющий генерировать данные для работы модели. Элементы имитационной модели реализуются в виде отдельного программного обеспечения и взаимодействуют между собой через сетевые сокеты. Данный подход позволяет как запускать процесс моделирования на одном устройстве, так и разделить процессы на несколько вычислительных машин, а также легко модифицировать алгоритмы работы отдельных компонентов модели.

На основании работающих в исследуемой КС элементов выделяются следующие модули, имитирующие сетевую структуру: агенты, имитирующие поведение конечных уст-

ройств системы; пассивные элементы сети (ПЭС), имитирующие среду передачи, и активные элементы сети (АЭС), имитирующие активные сетевые устройства, такие как маршрутизаторы и коммутаторы. Подробное описание работы данных модулей будет приведено в следующем разделе.

Для создания модели исследуемой КС используется модель на основе многоуровневой клиент-серверной архитектуры, как наиболее универсальной и легко поддающейся настройке и управлению из существующих архитектур [6]. При сетевом взаимодействии между компонентами каждый компонент имеет либо сокет, работающий в качестве TCP-сервера, прослушивающий входящие данные, либо сокет, работающий в качестве TCP-клиента, отправляющий данные, либо имеет оба сокета работающие одновременно. Компоненты модели запускаются в порядке от ядра к агентам, что позволяет организовывать создание прослушивающих TCP-серверов и подключение к ним в правильном порядке.

При централизованном запуске модели на одном устройстве, количество моделируемых элементов ограничено количеством свободных портов операционной системы, что описывается неравенством (1):

(1)

где – число пассивных элементов сети, – число активных элементов сети, – число агентов, 64512 – количество портов, вычисляемое как максимальное число портов (= 65536) за вычетом 1024 общеизвестных (well-known) портов. Если же модель запускается в распределённом режиме, она может включать в себя большее число элементов, однако возрастут и требования к вычислительным мощностям, требующимся для обработки данных.

После запуска всех компонентов работа имитационной модели КС осуществляется следующим образом:

1. Элемент, имитирующий поведение рабочей станции или выделенного сервера, генерирует данные, описывающие поведение данного элемента в каждый момент времени, и передаёт их через среду передачи данных.

2. Элемент, имитирующий среду передачи данных, получает данные, вносит возможные задержки и потери, после чего отправляет эти данные активному сетевому устройству.

3. Элемент, имитирующий активное сетевое устройство, получает данные и, при соответствующих настройках, может нести в них какие-либо изменения, после чего данные вновь передаются по среде передачи к следующему элементу.

4. Пункты 2 и 3 могут повторяться и комбинироваться в зависимости от исследуемой топологии КС.

5. Данные поступают в ядро модели, где может производиться их дальнейшая обработка в соответствии с исследуемой в имитационной модели задачей.

Вышеописанный процесс передачи данных в модели можно представить в виде схемы, изображённой на рисунке 1. Под блоком «Комбинация ПЭС/АЭС» подразумевается набор элементов, соединяющий агента сети с ядром модели. Примером такой комбинации может быть «ПЭС-АЭС-ПЭС», что означает: «Агент через среду передачи данных (первый ПЭС) передаёт данные коммутатору или маршрутизатору (АЭС), который через среду передачи данных (второй ПЭС) передаёт их в ядро модели».

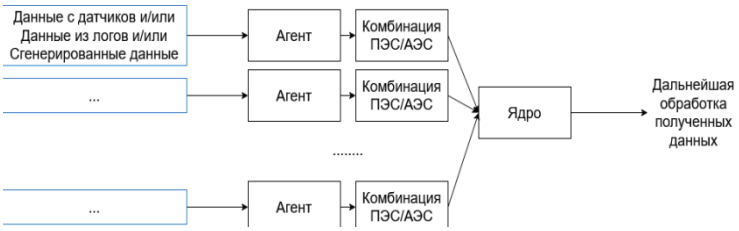

Рис. 1. Схема процесса передачи данных в модели

Разработанная структура модели позволяет легко перенести топологию построения реальной КС на имитационную модель. На рисунке 2 приведена иллюстрация, демонстрирующая реальную КС и её аналог в терминах имитационной модели.

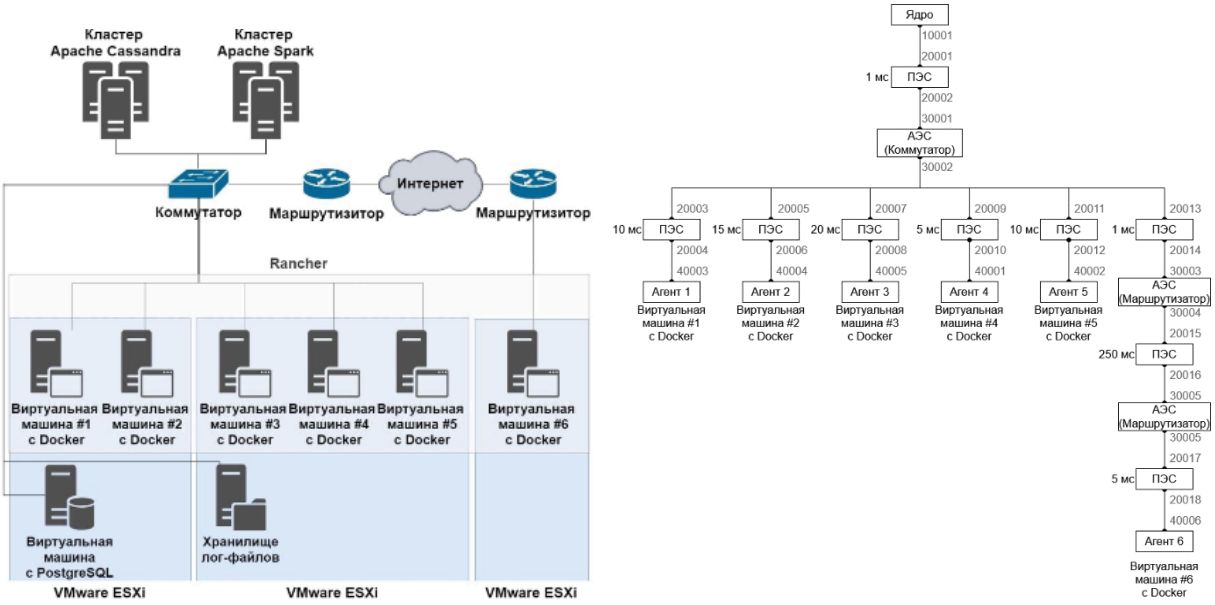

Рис. 2. Сетевая инфраструктура (слева) и её представление в терминах имитационной модели (справа)

На схеме имитационной модели помимо блоков, представляющих элементы сетевой структуры, приведены номера портов (схема приведена для централизованного запуска всех элементов), используемых для связи между элементами, а также дополнительно приведены искусственно вносимые задержки при передаче данных через пассивные элементы сети.

Модули, реализующие сетевую структуру

За имитацию сетевой структуры в модели отвечают три модуля: агент, активное сетевое устройство и пассивное сетевое устройство. Также к модулям сетевой структуры можно отнести ядро модели, так как оно является конечной точкой передачи данных в модели, хотя оно также выполняет и множество других задач, не связанных с передачей данных. Ядро похоже по принципу своей работы на активный элемент сети за исключением того, что ядро не посылает полученные данные дальше по сетевой структуре, а используется их для обработки в модулях ядра.

Агент представляет собой набор датчиков, отождествлённый с аппаратным обеспечением компьютерной системы: рабочими станциями и выделенными серверами. Задача агента – собрать данные с датчиков (подразумеваются различные сценарии работы: действительный сбор данных в реальном времени, получение данных из лог-файлов или искусственная генерация данных; последняя является основным сценарием при исследовании имитации поведения компьютерной системы) и передать их в ядро модели для дальнейшей работы. Агенты могут передавать как данные, характеризующие один признак, так и данные, характеризующие множество признаков. Именно агенты являются начальными звеньями в модели передачи данных.

Агент для своей работы создаёт соединение со следующим элементом, которое используется для передачи данных. Помимо самих данных, агент также передаёт в своих сообщениях следующую информацию: время и свой адрес. Алгоритм работы агента в виде псевдокода представлен в листинге 1.

Листинг 1 – Алгоритм работы агента

Обозначения: `local_ip` – IP-адрес агента, `local_port` – порт агента, `remote_ip` – IP-адрес следующего узла, `remote_port` – порт следующего узла.
Используемые функции: `IsData` – проверка на наличие данных для передачи, `GetData` – получение данных для передачи (с датчиков, из лог-файлов или сгенерированных), `PackData` – упаковка данных для передачи, `AddRoutingInfo` – добавление маршрутной информации, `SerializeData` – сериализация данных, `SendMessage` – отправка данных, `GetResponse` – получение подтверждения доставки.

```
function Agent(local_ip, local_port, remote_ip, re-
mote_port)
    reader, writer = OpenConnection(remote_ip, re-
mote_port,
                                    local_ip, lo-
cal_port)
    while IsData() do
        message = GetData()
        message = PackData(message)
        message = SerializeData(message)
        message = AddRoutingInfo(message)
        SendMessage(writer, message)
        GetResponce(reader)
    end while
end function
```

Пассивный элемент сети – это модуль, имитирующий среду передачи данных. Его задача – передача данных без внесения изменений от агента до активного элемента сети, от агента до ядра модели и между активными элементами сети. Во время передачи могут быть внесены задержки передачи, а также может имитироваться потеря данных.

ПЭС для своей работы создаёт соединение со следующим элементом, которое используется для передачи данных, а также TCP-сервер, принимающие данные от других элементов сети. Алгоритм работы пассивного элемента сети в виде псевдокода представлен в листинге 2.

Листинг 2 – Алгоритм работы пассивного элемента сети

Обозначения: `local_ip` – IP-адрес ПЭС, `local_input_port` – входящий порт ПЭС, `local_output_port` – исходящий порт ПЭС, `remote_ip` – IP-адрес следующего узла, `remote_port` – порт следующего узла, `drop` – вероятность потери пакета, `delay` – вносимая при передаче задержка.
Используемые функции: `StartServer` – запуск TCP-сервера, `OpenConnection` – открытие соединения с удалённым узлом, `Working` – проверка на отсутствие сигнала завершения работы, `GetData` – получение сообщения с прослушиваемого порта, `DropMessage` – потеря данных, `WaitUntilSend` – внесение задержки в передачу данных, `AddRoutingInfo` – добавление маршрутной информации, `SendMessage` – отправка данных, `GetResponse` – получение сообщения о доставке.

```
function PNE(local_ip, local_input_port, lo-
cal_output_port, remote_ip, remote_port, drop, delay)
    readerServ, writerServ = StartServer(local_ip,
local_input_port)
    reader, writer = OpenConnection(remote_ip, re-
mote_port,
                                    local_ip, lo-
cal_output_port)
    while Working() do
        message = GetData(readerServ)
        if DropMessage(drop) is False then
            message = AddRoutingInfo(message)
            WaitUntilSend(delay)
            SendMessage(writer, message)
            GetResponse(reader)
        end if
        SetResponse(writerServ)
    end while
end function
```

Активный элемент сети имитирует поведение активного сетевого оборудования, такого как коммутаторы и маршрутизаторы. Данный элемент позволяет обрабатывать данные в момент между их получением и передачей следующему элементу.

АЭС, также как и ПЭС, создаёт соединение со следующим элементом и запускает TCP-сервер для приёма данных. Алгоритм работы активного элемента сети в виде псевдокода представлен в листинге 3.

Листинг 3 – Алгоритм работы активного элемента сети

Обозначения: `local_ip` – IP-адрес АЭС, `local_input_port` – входящий порт АЭС, `local_output_port` – исходящий порт АЭС, `remote_ip` – IP-адрес следующего узла, `remote_port` – порт следующего узла.
Используемые функции: `StartServer` – запуск TCP-сервера, `OpenConnection` – открытие соединения с удалённым узлом, `Working` – проверка на отсутствие сигнала завершения работы, `GetData` – получение сообщения с прослушиваемого порта, `DeserializeData` – десериализация данных, `ChangeData` – внесение изменений в данные, `SerializeData` – сериализация данных, `AddRoutingInfo` – добавление маршрутной информации, `SendMessage` – отправка данных, `GetResponse` – получение сообщения о доставке, `SetResponse` – отправление сообщения о доставке.

```
function ANE(local_ip, local_input_port, lo-
cal_output_port, remote_ip, remote_port)
    readerServ, writerServ = StartServer(local_ip,
local_input_port)
    reader, writer = OpenConnection(remote_ip, re-
mote_port,
                                    local_ip, lo-
cal_output_port)
    while Working() do
        message = GetData(readerServ)
        message = DeserializeData(message)
        message = ChangeData(message)
        message = SerializeData(message)
        message = AddRoutingInfo(message)
        SendMessage(writer, message)
        GetResponse(reader)
        SetResponse(writerServ)
    end while
end function
```

Ядро модели, как часть сетевой структуры, – это модуль, который принимает все данные, поступающие от агентов. Ядро работает как TCP-сервер, принимающий данные от сетевых элементов и отправляющий их на дальнейшую обработку другим модулям модели. Алгоритм работы ядра в виде псевдокода приведён в листинге 4.

Листинг 4 – Алгоритм работы ядра

Обозначения: `local_ip` – IP-адрес ядра, `local_port` – входящий порт ядра.
Используемые функции: `StartServer` – запуск TCP-сервера, `Working` – проверка на отсутствие сигнала завершения работы, `GetData` – получение сообщения с прослушиваемого порта, `SetResponse` – отправление сообщения о доставке, `DeserializeData` – десериализация данных, `GetPayload` – получение полезной нагрузки из данных, `SendToNextModule` – отправка полученных даных другому модулю оделе для обработки и анализа.

```
function Core(local_ip, local_input_port)
    reader, writer = StartServer(local_ip, local_port)
    while Working() do
        message = GetData(reader)
        SetResponse(writer)
        message = DeserializeData(message)
        message = GetPayload(message)
        SendToNextModule(message)
    end while
end function
```

Данные сетевые компоненты объединяются в единую сеть, путём задания соответствующих настроек для имитации процесса передачи данных от агентов к ядру модели.

Модуль генерации данных

Модуль генерации данных в качестве основного элемента использует генеративно-состязательную нейронную сеть (GAN, Generative Adversarial Network) [7]. Преимущество генерации данных с использованием GAN состоит в том, что данная искусственная нейронная сеть (ИНС) позволяет генерировать новые значения, отличающиеся от исходных данных, которые использовались для обучения, применяя те же шаблоны поведения данных во времени, что были в этих данных.

Для обучения модуля генерации данных используется генеративно-состязательная ИНС, представляющая собой комбинацию сетей свёртки (для дискриминативной классифицирующей сети) и развёртки (для генеративной сети). Генеративная сеть считается обученной тогда, когда сгенерированные ею данные дискриминативная сеть не может отличить от реальных. На рисунке 3 приведена схема построения генеративно-состязательной ИНС с указанием подробной структуры генеративной и дискриминативной сетей.

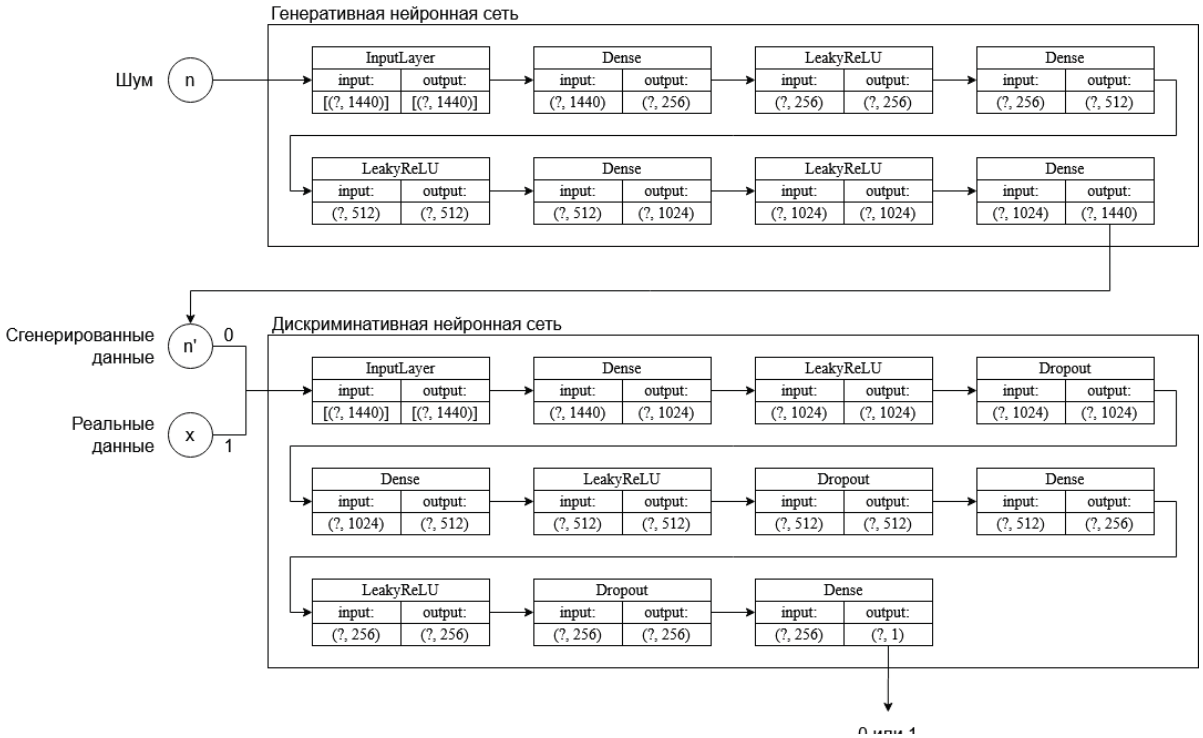

Рис. 3. Схема обучения модуля генерации на основе генеративно-состязательной ИНС

После обучения нейронных сетей по данной схеме, обученная генеративная нейронная сеть добавляется в алгоритм работы модуля генерации данных.

Модуль генерации данных устроен следующим образом. При соответствующем запросе от модуля-агента генеративная сеть на основе случайного шума генерирует данные в заданном временном диапазоне (например, суточные ежеминутные данные, как в примере с рисунка 3) и затем с заданным интервалом времен передаёт их элементу-агенту, как если бы эти данные поступали в реально существующих датчиков.

На рисунке 4 показан пример работы модуля генерации данных. На графике 4а изображены данные из реальной КС о значениях простоя центрального процессора, которые могут отправлять агенты и которые применялись для обучения, представленные в виде временного ряда. На графиках 4б и 4в представлены данные, сгенерированные нейронной сетью. По оси X на графиках представлена временная шкала в минутах, а по оси Y – значения простоя центрального процессора в каждый момент времени. Получаемые данные схожи по своему поведению с исходными, однако не являются идентичными.

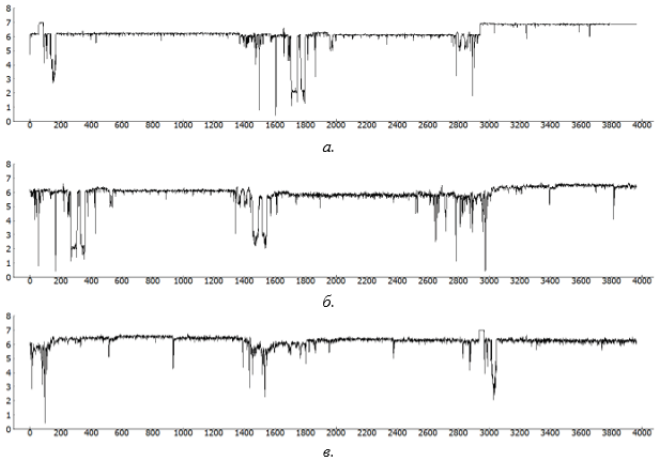

Рис. 4. Исходный (*а*) и два сгенерированных (*б*, *в*) временных ряда

Подобные сгенерированные данные могут дополнительно преобразовываться, например, путем добавления каких-либо отклонений в поведении или изменением характера поведения отдельных участков временного ряда. Эти преобразования могут быть реализованы путём добавления соответствующего модуля дополнительной обработки данных.

Методика оценки качества имитации данных

Для оценки качества генерации временных рядов при помощи генеративно-состязательной ИНС была применена следующая методика:

1. Производится квантование исходного временного ряда. При оценке использовалось количество уровней квантования (L) равное 5, 10 и 15.
2. На основе полученного квантованного временного ряда строится матрица переходных вероятностей (M) размером $i * j$, где $i = j = L$. В каждой ячейке матрицы M_{ij} указывается вероятность перехода значений временного ряда с уровня i на уровень j. Сумма вероятностей в каждой строке матрицы равна единице или 100%.
3. Производится генерация заданного количества временных рядов. Для оценки было выполнено 1000 генераций временных рядов продолжительностью 2 суток каждая.
4. Для каждого сгенерированного временного ряда строится своя матрица переходных вероятностей и затем данные матрицы усредняются, для получения средних вероятностей перехода в сгенерированных данных.
5. Производится сравнение матрицы переходных вероятностей исходного временного ряда и сгенерированных временных рядов. На основании сравнения оценивается качество генерации данных.

Оценка сходства матриц M выполняется путём преобразования двумерной матрицы в вектор значений путем последовательного объединения строк матрицы в вектор значений с сохранением взаимного расположения всех элементов и вычисления евклидова расстояния в многомерном пространстве по формуле (2):

$$d(t,g) = \sqrt{(t_1 - g_1)^2 + (t_2 - g_2)^2 + \cdots + (t_N - g_N)^2} = \\ = \sqrt{\sum_{i=1}^{N}(t_i - g_i)^2}, \quad (2)$$

где t и g – значения переходных вероятностей исходного и сгенерированного временных рядов, N – количество переходных вероятностей в векторе ($N = L^2$).

В процессе оценки качества генерации обученной модели была произведена оценка качества моделей, генерирующих временные отрезки различной длины. Модель, обучалась и предсказывала данные длительностью 24 часа, а оценка приводилась для моделей, работающих с длительностью в 2, 12 и 24 часа, что позволило обосновать выбранную длительность генерации.

Результаты имитационного моделирования

При исследовании работы имитационной модели и оценки качества её имитации данных были использованы 4 параметра, генерируемых объектами компьютерной сети описанной в [8]. В качестве эталонных значений, с которыми сравниваются сгенерированные данные, были взяты значения данных параметров из реальной КС. Эталонные временные ряды на рисунке 5 длительностью 2 суток представлены для следующих выбранных параметров:

- Значение простоя восьмиядерного центрального процессора – cpu_idle (рис. 5а).
- Значение загрузки жёсткого диска – disk_busy (рис. 5б).
- Значение средней загрузки системы за минуту – load_oneminute (рис. 5в).
- Значение средней задержки передачи пакетов по сети – ping_avg (рис. 5г).

На рисунке 5 по оси X представлена временная шкала в минутах, а по оси Y представлены следующие величины:

- Значение простоя центрального процессора в единицах. Под одной единицей понимается полный простой одного ядра процессора.
- Значение загрузки жёсткого диска в процентах.
- Значение средней загрузки системы за минуту в процентах.

- Значение среднего времени задержки пакетов по сети в миллисекундах.

В соответствии с приведённой методикой, было произведено квантование каждого временного ряда на 5, 10 и 15 уровней, после чего были построены матрицы переходных вероятностей *M* для каждого параметра. Матрицы переходных вероятностей для каждого исследуемого параметра при 5 уровнях квантования показаны на рисунке 6.

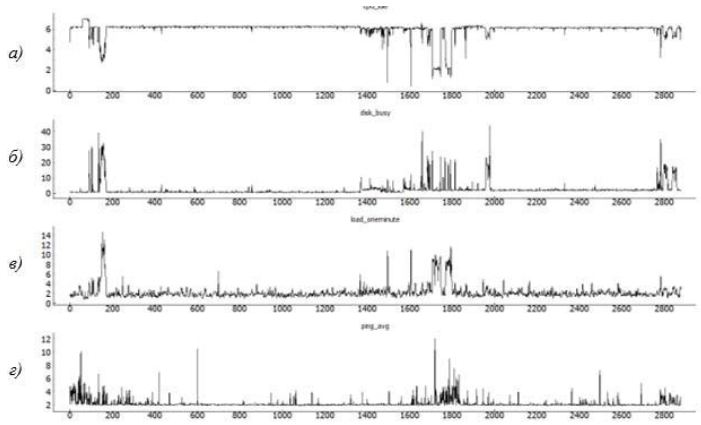

Рис. 5. Исходные временные ряды

	1	2	3	4	5
1	45.45%	27.27%	18.18%	9.09%	0.00%
2	4.55%	89.39%	4.55%	1.52%	0.00%
3	3.23%	9.68%	45.16%	38.71%	3.23%
4	0.69%	0.69%	5.52%	53.79%	39.31%
5	0.04%	0.00%	0.15%	1.98%	97.83%

cpu_idle

	1	2	3	4	5
1	98.84%	0.73%	0.22%	0.15%	0.07%
2	29.41%	47.06%	20.59%	1.47%	1.47%
3	19.44%	38.89%	30.56%	11.11%	0.00%
4	21.43%	7.14%	35.71%	35.71%	0.00%
5	66.67%	33.33%	0.00%	0.00%	0.00%

disk_busy

	1	2	3	4	5
1	98.63%	1.29%	0.07%	0.00%	0.00%
2	43.75%	47.50%	6.25%	2.50%	0.00%
3	1.54%	9.23%	80.00%	9.23%	0.00%
4	4.55%	4.55%	27.27%	54.55%	9.09%
5	0.00%	0.00%	0.00%	66.67%	33.33%

load_oneminute

	1	2	3	4	5
1	97.60%	2.05%	0.22%	0.04%	0.11%
2	72.15%	24.05%	3.80%	0.00%	0.00%
3	66.67%	33.33%	0.00%	0.00%	0.00%
4	100.00%	0.00%	0.00%	0.00%	0.00%
5	100.00%	0.00%	0.00%	0.00%	0.00%

ping_avg

Рис. 6. Матрицы переходных вероятностей для исходного временного ряда

При генерации временных рядов длительностью 24 часа, матрицы переходных вероятностей сгенерированных данных принимают вид, представленный на рисунке 7. Представленные матрицы содержат в себе информацию о переходных вероятностях при 2000 сутках наблюдения (1000 генераций, каждая по 2 суток, как в исходных данных).

Подобные матрицы для 5 уровней квантования также были построены для данных, генерируемых по 2 и по 12 часов, а также при 10 и 15 уровнях квантования. Рассчитанные значения евклидова расстояния между матрицами переходных вероятностей при различных длительностях генерации и уровнях квантования представлены в таблице 1.

	1	2	3	4	5
1	47.30%	35.98%	7.25%	9.44%	0.03%
2	9.58%	80.41%	6.67%	2.53%	0.81%
3	5.47%	16.22%	37.05%	30.94%	10.32%
4	0.26%	0.33%	2.66%	50.83%	45.93%
5	0.06%	0.03%	0.19%	5.21%	94.51%

cpu_idle

	1	2	3	4	5
1	98.69%	0.81%	0.30%	0.10%	0.09%
2	30.78%	49.26%	16.91%	1.82%	1.22%
3	22.45%	38.07%	30.35%	8.88%	0.25%
4	17.17%	9.70%	29.93%	34.23%	8.97%
5	44.91%	33.11%	16.51%	4.63%	0.83%

disk_busy

	1	2	3	4	5
1	95.79%	4.04%	0.15%	0.02%	0.00%
2	53.17%	42.05%	3.24%	1.48%	0.05%
3	5.80%	14.95%	56.79%	21.29%	1.18%
4	1.21%	5.24%	17.29%	64.78%	11.48%
5	6.92%	9.69%	17.16%	37.32%	28.90%

load_oneminute

	1	2	3	4	5
1	97.20%	2.40%	0.25%	0.06%	0.09%
2	71.05%	24.89%	4.00%	0.02%	0.03%
3	68.89%	28.85%	2.25%	0.01%	0.00%
4	95.57%	3.98%	0.40%	0.06%	0.00%
5	99.88%	0.12%	0.00%	0.00%	0.00%

ping_avg

Рис. 7. Матрицы переходных вероятностей для сгенерированных временных рядов длительностью 24 часа

Таблица 1

Значение евклидова расстояния при различной длительности генерации и уровнях квантования

Длительность генерации	Уровней квантования	Параметр			
		cpu_idle	_busy	load_oneminute	ping_avg
2 часа	5	1.227	0.882	1.312	0.528
12 часов		0.301	0.537	0.667	0.176
24 часа		0.249	0.305	0.492	0.082
2 часа	10	1.820	1.649	1.796	1.133
12 часов		1.112	1.278	1.191	0.926
24 часа		0.988	1.148	1.072	1.026
2 часа	15	2.133	2.061	1.967	1.901
12 часов		1.585	1.553	1.308	1.749
24 часа		1.279	1.463	1.199	1.886

На основании данных из вышеприведённой таблицы можно сделать вывод о том, что генерация временных рядов длительностью 24 часа даёт результаты, наиболее приближенные к исходному временному ряду. Евклидово расстояние между матрицами переходов вероятностей имеет тенденцию к уменьшению, при увеличении длительности генерации до значения равного длине сезонности временного ряда. Отсюда следует, что обучение генеративной сети на временных отрезках длительностью 24 часа даёт наилучшее качество имитации реального поведения КС. Сравнение сгенерированных временных рядов с исходным показало, что генерация отрезков длительность 24 часа позволяет генерировать данные в наибольшей степени повторяющие характер поведения, в том числе и аномальных исходных данных.

Заключение

Разработанная имитационная модель позволяет исследовать поведение реальных компьютерных систем в том числе и при аномальном режиме работы. Модель позволяет использовать как реальные данные с датчиков или из логов, так и генерировать новые на основе заранее собранных исторических данных.

Применение искусственных нейронных сетей внутри модуля генерации данных позволяет максимально качественно реализовывать процесс имитации поведения сети с возможностью дополнительного обучения используемых сетей, по мере необходимости, или полной их замены.

Модульная структура с элементами, представленными в виде самостоятельных процессов, и клиент-серверная архитектура позволяют легко масштабировать размеры модели, а также распределять её работу на несколько устройств. Разработанная имитационная модель легко реализуема и позволяет достаточно просто вносить изменения в её отдельные компоненты.

Реализованный в модели модуль генерации данных на основе применения генеративно-состязательной ИНС показал высокое качество имитации данных реальных КС. При генерации данных, длительность которых совпадает с длительностью сезонности ряда, модуль генерирует временных ряды, матрицы переходных вероятностей которых близки к матрицам переходных вероятностей исходных данных.

Полученные результаты имеют практический характер и могут быть использованы для тестирования, исследования и анализа поведения реальных КС. Имитационная модель может быть дополнена модулями анализа данных, поступающих от агентов, что позволит расширить область применения разработанной модели.

Литература

1. НОУ ИНТУИТ |Лекция| Компьютерное имитационное моделирование. Статистическое имитационное моделирование [Электронный ресурс] URL: https://intuit.ru/studies/courses/2260/156/lecture/27241 (Дата обращения: 20.10.2020).
2. *Гудов А.М., Семехина М.В*. Имитационное моделирование процессов передачи трафика в вычислительных сетях // Управление большими системами: сборник трудов. 2010. №. 31.
3. *Gorodetski V., Kotenko I*. Attacks against computer network: Formal grammar-based framework and simulation tool // International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg, 2002. С. 219-238.
4. *Кертов К.В*. и др. Моделирование защищенной локальной вычислительной сети организации в среде имитационного моделирования CISCO PACKET TRACER 6.2 // Современные наукоемкие технологии. 2017. №. 8. С. 19-24.
5. *Барчан К.А*. Разработка метода имитационного компьютерного моделирования эталонного состояния и поведения SCADA-систем на основе модели акторов // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2014. Т. 12. №.1.
6. *Таненбаум Э.С*. Компьютерные сети. 5-е изд. СПб.: Питер, 2013. 960 с.
7. *Goodfellow I*. et al. Generative adversarial nets //Advances in neural information processing systems. 2014. С. 2672-2680.
8. *Шелухин О.И., Костин Д.В*. Классификация аномальных состояний компьютерных систем средствами интеллектуального анализа системных журналов // Нейрокомпьютеры: разработка, применение. 2020. Т. 22. №. 1. С. 66-76.

SIMULATION OF THE BEHAVIOR OF A COMPUTER SYSTEM USING ARTIFICIAL NEURAL NETWORKS

Oleg I. Sheluhin, *Moscow Technical University of Communication and Informatics, Moscow, Russia,* ***sheluhin@mail.ru***
Aleksey Yu. Sharikov, *Moscow Technical University of Communication and Informatics, Moscow, Russia,* ***sharikov.it@gmail.com***

Abstract
The design and the implementation of the simulation model of a computer system (CS) using artificial neural networks (ANNs) are considered. The purpose of the implementation is to create the easy-to-learn and easy-to-implement simulation model that allows to simulate both normal and anomalous processes in computer systems. The developed simulation model is the software, which consists of various modules combined using principles of the client-server architecture, allowing to run the model in both centralized and distributed modes of operation. The model allows to simulate the behavior of the CS with various topologies: a star, a tree, and a combination of these topologies. The main elements of the model are implemented in the form of four modules that fulfill their specific roles: an agent that generates data; a passive network element that transmits data with possible delays and losses; an active network element that processes and transmits data arriving at it, and the core - the central element of the model that receives data and sends it to additional modules for analysis. The modularity provides a high potential for further modifications of the simulation model by adding new modules. Using the generative adversarial network-based data generation module in the model makes it possible to generate data required for modeling the behavior of the studied CS. Based on the calculation of the Euclidean distance between matrices of transition probabilities of initial and generated data, it is shown that processes generated using the developed simulation model have a similar behavior with real ones. The designed model can be used to study the work of a real CS, including the imitation of an anomalous behavior.

Keywords: *Computer modelling, simulation modeling, time series, computer systems, artificial neural networks, generative adversarial networks, GAN.*

References

1. NOU Intuit (2011), "Lecture 5: Computer simulation. Statistical simulation", available at: https://intuit.ru/studies/courses/2260/156/lecture/27241 (accessed 20.10.2020).
2. Goodov A.M. and Semekhina M.V. (2010). Traffic transfer processes simulation in computing networks. UBS, Vol. 31. P. 130-161.
3. Gorodetski V. and Kotenko I. (2002). Attacks against computer network: Formal grammar-based framework and simulation tool. *International Workshop on Recent Advances in Intrusion Detection,* Springer, Berlin, Heidelberg. P. 219-238.
4. Kertov K.V., Boliev Z.V., Shogenov A.A., Zhilokov A.Kh., and Kucherova V.Yu. (2017). Modeling of the protected local network of the organization in the media of imitation simulation Cisco Packet Tracer 6.2. Modern High Technologies. Vol. 8. P. 19-24.
5. Barchan K.A. (2014). The development of the actor model based computer simulation modeling method of SCADA standard state and behavior.Vestnik NSU. Series: Information Technologies, Vol. 12., No. 1, pp. 11-18.
6. Tanenbaum A.S. (2013). Computer Networks, 5th ed., Piter Publishing House, Saint-Petersburg.
7. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A. and Bengio Y. (2014). Generative adversarial nets. *Advances in neural information processing systems.* Vol. 27. P. 2672-2680.
8. Sheluhin O.I. and Kostin D.V. (2020). Classification of anomalous states of computer systems by means of intellectual analysis of system journals. *Neurocomputers: development, application.* Vol. 22. No. 1. P. 66-76.

Information about authors:
Oleg I. Sheluhin, *doctor of technical sciences, professor, head of the Department of Information Security, Moscow Technical University of Communication and Informatics, Moscow, Russia*
Aleksey Yu. Sharikov, *undergraduate, Moscow Technical University of Communication and Informatics, Moscow, Russia*