

Исследование мультиагентной модели в системе NETLOGO (модель DDOS атаки)

Ковалева Ирина Валерьевна

*Приамурский государственный университет имени Шолом-Алейхема
Студент*

Баженов Руслан Иванович

*Приамурский государственный университет имени Шолом-Алейхема
К.п.н., доцент, зав. кафедрой информационных систем, математики и
методик обучения*

Аннотация

В статье рассматривается исследование мультиагентной модели в системе NETLOGO. В данной работе описано создание DDoS-атаки, подробно описаны коды и проведено исследование на основе созданной модели.

Ключевые слова: модель, NetLogo, DDoS-атака, распределенный отказ в обслуживании, сервер

The study of the multiagent model in the NETLOGO system (the DDOS attack model)

Kovaleva Irina Valerievna

*Sholom-Alechey Priamursky State University
Student*

Bazhenov Ruslan Ivanovich

*Sholom-Alechey Priamursky State University
Candidate of pedagogical sciences associate professor, Head of the Department of
Information System, Mathematics and teaching methods*

Abstract

The article examines the study of the multiagent model in the NETLOGO system. In this paper, we describe the creation of a DDoS attack, the codes are described in detail, and a study based on the model created.

Keywords: model, NetLogo, DDoS-attack, distributed denial of service, server

Во времена информационных технологий более популярными становятся распределенные атаки на глобальные компьютерные сети. Большая часть таких атак направлена на нарушение доступности или «Распределенный отказ в обслуживании» (Distributed Denial of Service, DDoS) и выведение из строя сервера путем наполнения системы большим количеством сетевых пакетов. Реализация таких атак может привести не

только к выходу из строя отдельных хостов, но и остановить работу корневых DNS-серверов и вызвать частичное или полное прекращение работы Интернета.

Для того чтобы лучше рассмотреть поведение таких атак, в разных средах для агентного моделирования создаются разнообразные модели, которые показывают как сервер выходит из строя. В данном исследовании модель реализована в среде мультиагентного моделирования NetLogo.

Исследованиями в данной теме занимались очень многие авторы. Например, К.Н. Мезенцев изучал мультиагентное моделирование в среде NetLogo[1], а В.А. Векслер моделировал экологические системы в среде NetLogo на уроках информатики в средней школе [2]. Я.О. Росоха исследовал проблематику распределенных атак типа «Отказ в обслуживании» (DDoS)[3]. Е.В. Жилина, Д.О. Ромашкин изучали признаки сетевых DDoS-атак и методы их обнаружения[4], а Е.В. Пальчевский разрабатывал защиту от DDoS-атаки HTTP-трафиком на веб-системы управления сайтом[5]. Я.В. Тарасов имел опыт использования технологий нейронных сетей для обнаружения низкоинтенсивных DDoS-атак[6]. Е.Д. Бычков и В.В. Кладов разрабатывали защиту Web-сервера от атак типа DDoS на основе модели нечеткого вывода[7]. М.В. Бурса, А.О. Григорий, Н.И. Баранников и К.А. Разинкин занимались управлением риска успешной реализации DDoS-атак на мультисервисные сети[8]. Р.Б. Рашевский и А.С. Шабуров на практике применяли нейронные сети для защиты информационно-управляющих систем критически важных объектов от DDoS-атак [9], а О.С. Тернова, А.В. Жариков и А.С. Шатохин применяли метод Хертса для определения сезонности сетевого трафика с целью раннего обнаружения DDoS-атак[10]. Bing Wang, Yao Zheng, Wenjing Lou и Y. Thomas Hou разрабатывали защиту от DDoS-атак в эпоху облачных вычислений и программно-определяемых сетей[11]. Sunny Behal и Krishan Kumar занимались обнаружением DDoS-атак и внезапных событий, используя обобщенный алгоритм обнаружения, который использует разность энтропии между потоками трафика для обнаружения различных типов атак [12]. Monika Sachdeva, Krishan Kumar и Gurvinder Singh занимались изучением комплексного подхода к распознаванию DDoS-атак [13].

Среда программирования NetLogo предназначена для моделирования ситуаций и феноменов, которые происходят в природе и обществе. В данной программе можно давать указания и управлять тысячами независимых «агентов» действующих параллельно. NetLogo отлично подходит для проведения исследовательских работ, а библиотека моделей программы содержит множество моделей по математике, биологии, химии и других наук. Программа открывает возможность для понимания и объяснения связей между поведением отдельных индивидуумов, природными явлениями и т.д.[14,15].

Изначально следует запустить NetLogo. Интерфейс программы довольно простой и содержит игровой мир, состоящий из двумерной сетки прямоугольных участков – неподвижных агентов, окно для создания модели,

окно командного центра, где могут вводиться команды для агентов, а также содержит кнопки для управления и изменения настроек для агентов. Сверху в окне программы находятся 3 вкладки: интерфейс, где происходит создание модели; вкладка «инфо», где существует возможность прописать функции модели и ее свойства и вкладка «код», где соответственно будет находиться программный код для модели (рис. 1).

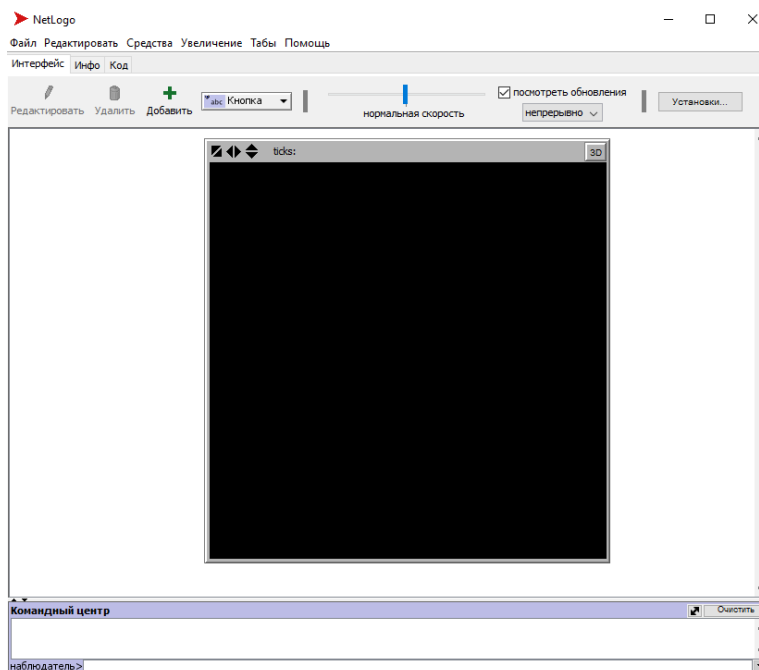


Рисунок 1. Интерфейс программы NetLogo

Следующий шаг – создание кнопок для управления моделью. В поле следует нажать на список объектов интерфейса и выбрать объект «кнопка» (Рис. 2).

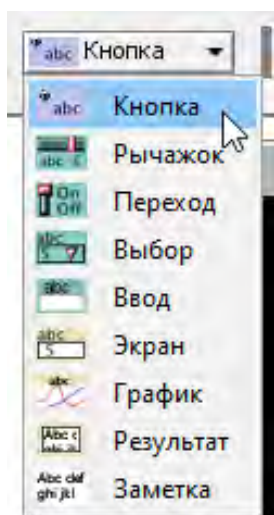


Рисунок 2. Объекты интерфейса для добавления

В редактировании кнопки следует прописать название и команду. Данная кнопка будет предназначена для обновления модели и установки новых параметров, которые будут задаваться агентам (Рис. 3).

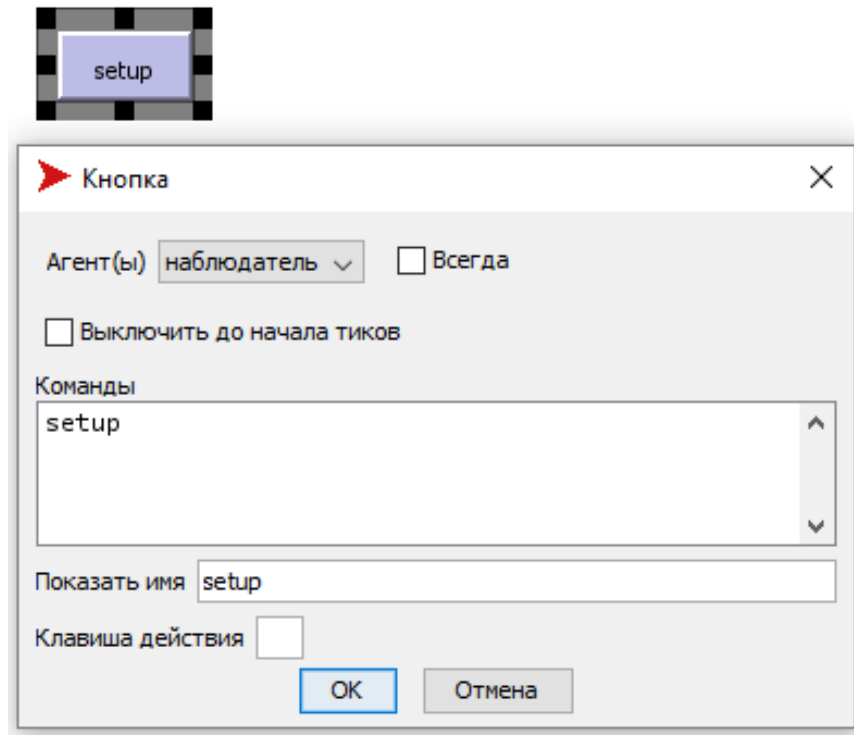


Рисунок 3. Редактирование кнопки setup

Таким же образом создается кнопка запуска модели «go». Но в редактировании данной кнопки следует отметить, что при запуске модель сама не остановится, пока на кнопку не нажмут еще раз (рис. 4).

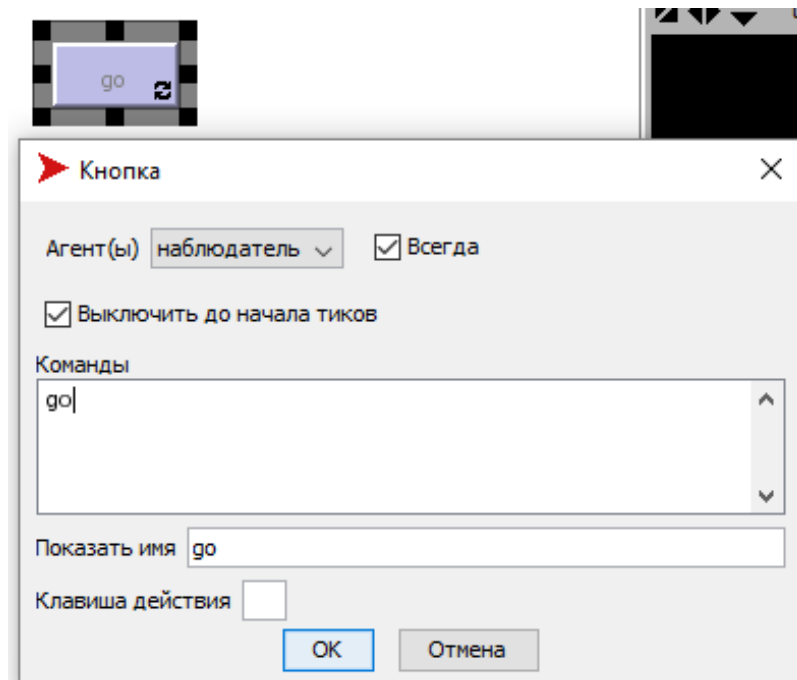


Рисунок 4. Редактирование кнопки «go»

Далее создаются рычажки-переключатели, с помощью которых можно изменять различные параметры для агентов и тем самым изменять модель. В

списке объектов интерфейса сразу после кнопки находится рычажок. В модели будет расположено 5 таких рычажков.

Первый рычажок – количество стеков у сервера. В поле редактирования прописывается название, начальное значение, можно отметить максимум и минимум (рис. 5).

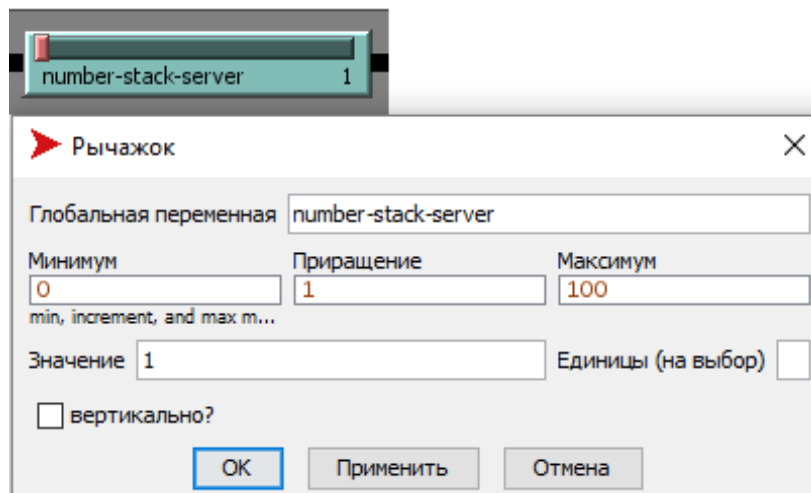


Рисунок 5. Редактирование рычажка «number-stack-server»

Второй рычажок – количество атак (зараженных компьютеров). Начальное значение равно единице (рис. 6).

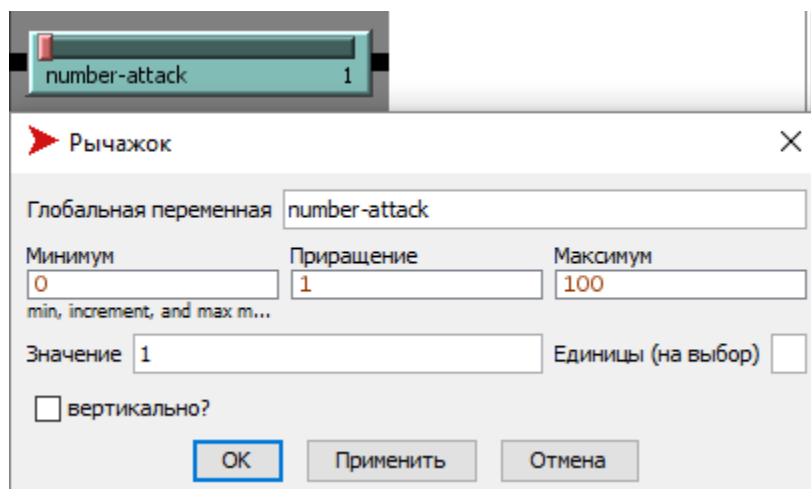


Рисунок 6. Редактирование рычажка «number-attack»

Третий и четвертый рычажки будут отвечать за энергию сервера (количество атак, которые способен выдержать сервер) и энергию атаки (модель количества одновременных атак зараженного компьютера) (рис. 7,8).

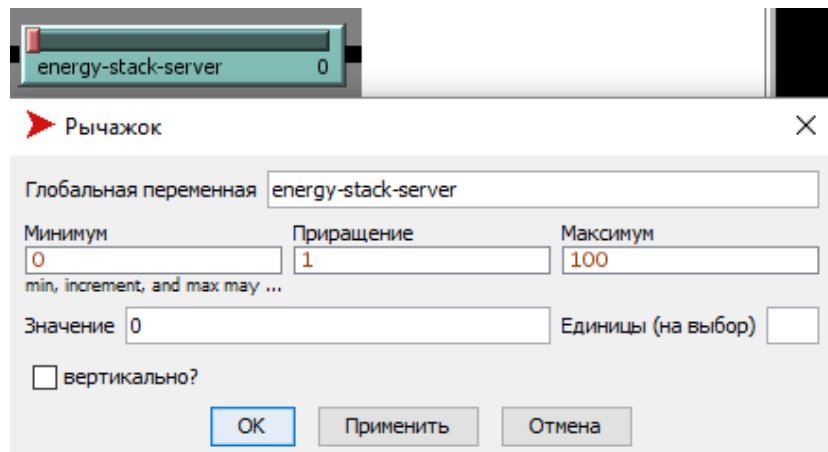


Рисунок 7. Редактирование рычажка «energy-stack-server»

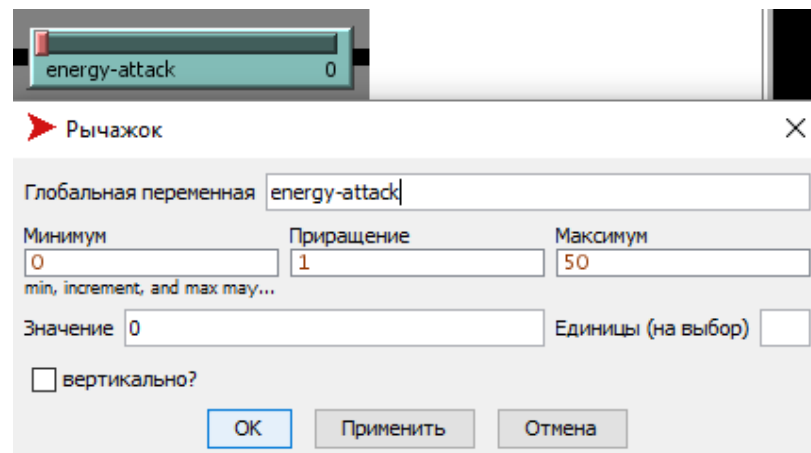


Рисунок 8. Редактирование рычажка «energy-attack»

Пятый рычажок будет применяться для воспроизведения новых атак. Изменяя параметры, изменяется вероятность воспроизведения новых зараженных компьютеров.

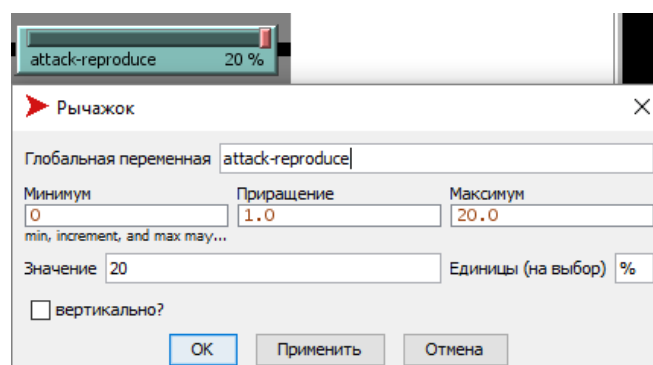


Рисунок 9. Редактирование рычажка «attack-reproduce»

Для того чтобы посмотреть поведение сервера и атаки следует добавить график, выбрав его из списка объектов интерфейса. В окне редактирования прописывается название графика, указывается имена осей X, Y и их минимальное и максимальное значения. Также следует добавить 2 пера, которые будут обозначать сервер и атаку, при этом еще выбрать цвет

линий и прописать перьям команды, в которых будет прописано, к какому агенту относится перо (рис. 10)

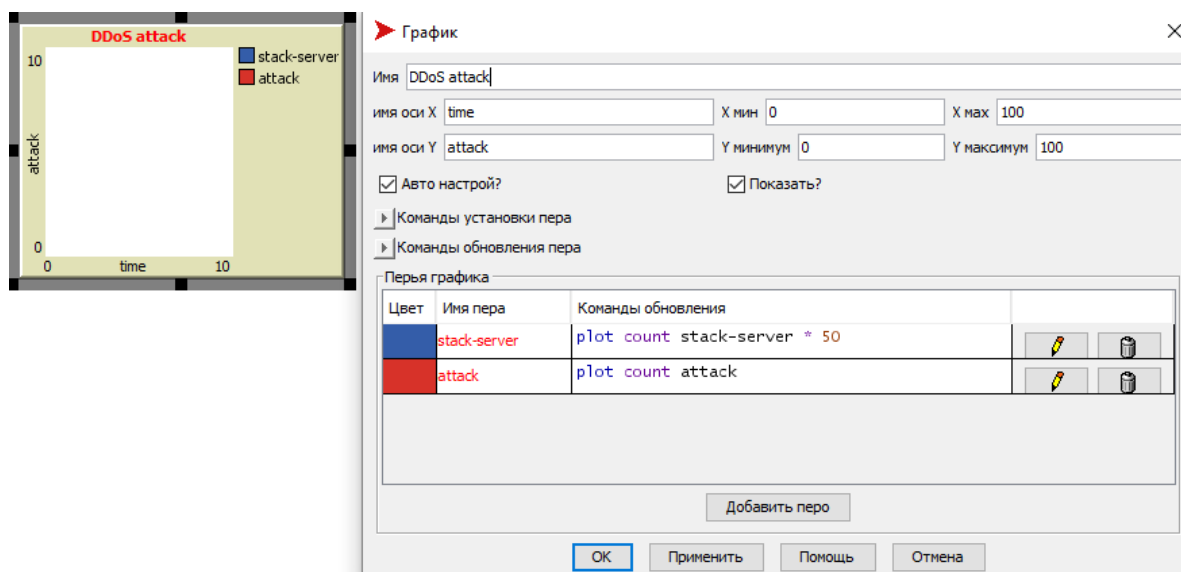


Рисунок 10. Редактирование графика

Для того чтобы отследить количество атак и количество стеков у сервера следует добавить объект под названием «экран». Он находится в списке объектов интерфейса для добавления. В окне редактирования прописывается датчик, имя экрана, размер шрифта и количество десятичных знаков. Создается 2 таких объекта, один для сервера, второй для количества атак (рис. 11, 12).

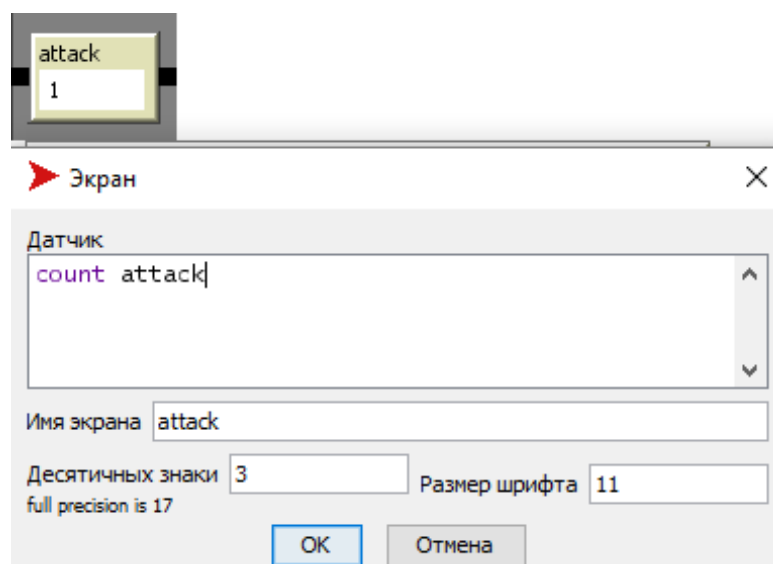


Рисунок 11. Редактирование объекта для показа количества атак

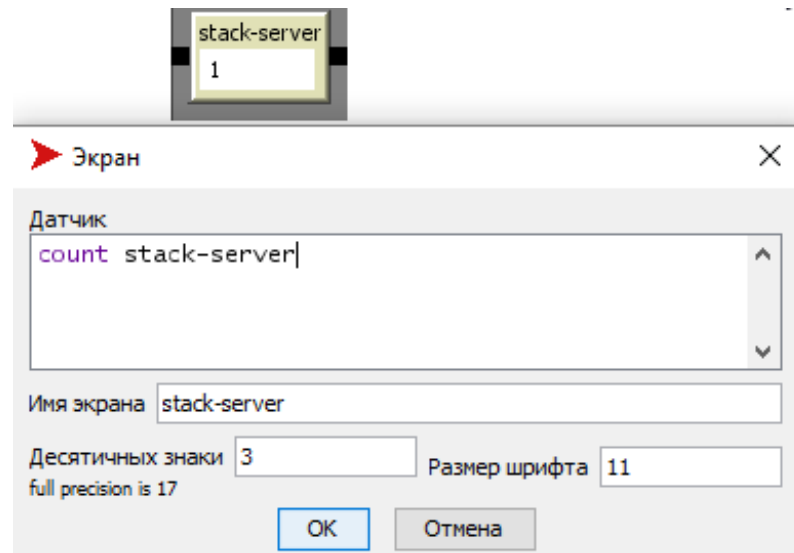


Рисунок 12. Редактирование объекта для показа количества стеков у сервера

После добавления объектов интерфейс модели должен выглядеть, как показано на рисунке 13.

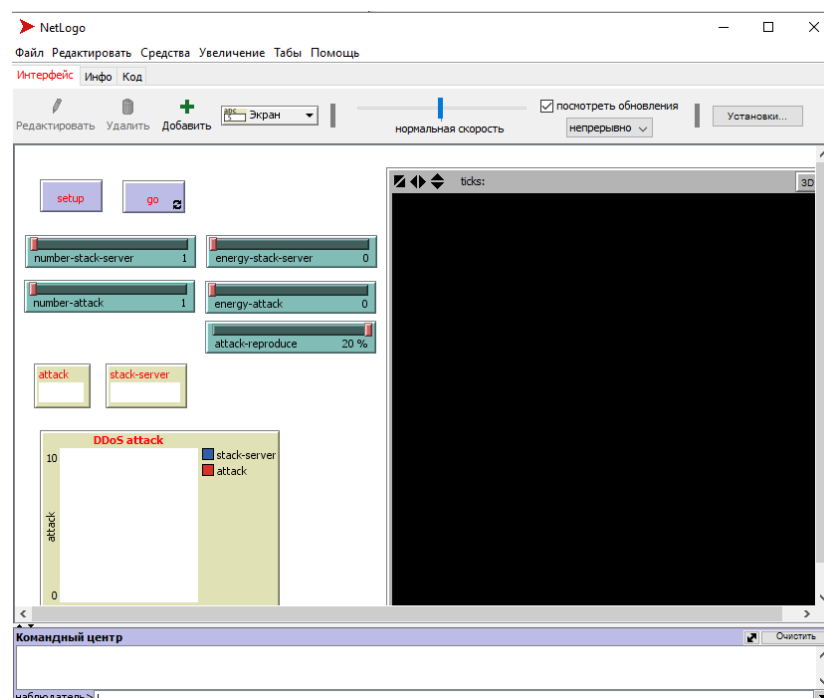


Рисунок 13. Интерфейс модели после добавления объектов

Следующий шаг в создании модели состоит в создании самих агентов. Для этого в верхней панели нужно выбрать вкладку «Средства» и после «Редактор форм черепах». В редакторе форм черепах находятся различные формы, которые можно присвоить агентам, например животные, растения, фигуры и т.д. (рис. 14). Но в библиотеке таких форм намного больше, поэтому внизу следует нажать на кнопку «Импорт из библиотеки».

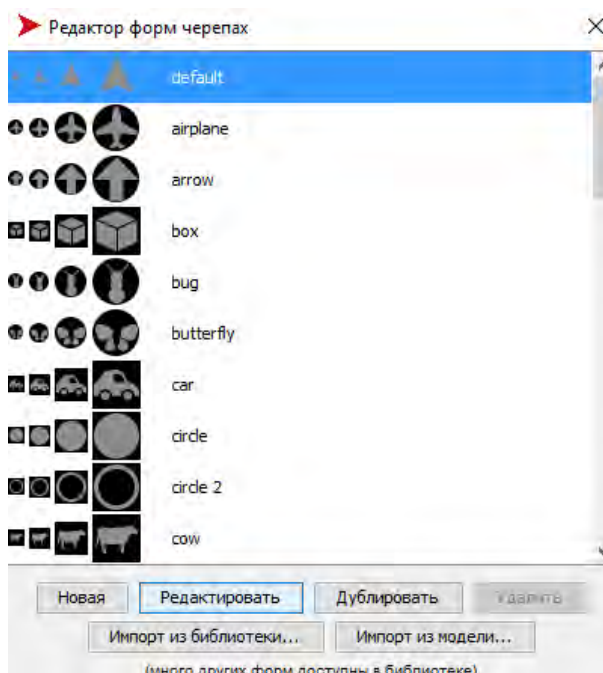


Рисунок 14. Редактор форм черепах

В библиотеке нужно найти формы «computer server», «computer workstation» и импортировать их в редактор форм. Формы можно дублировать, редактировать, менять цвет, форму, уменьшать и увеличивать, а также менять название.

В форме «computer server» изменяется только название на «stack-server» (рис. 15). А у формы «computer workstation» изменяется не только название на «attack», но и меняется цвет на красный, для того, чтобы было понятно, что этот компьютер заражен (рис. 16).

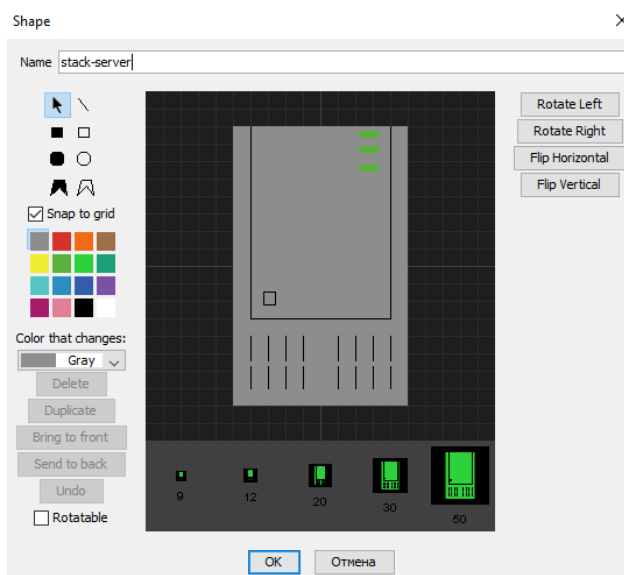


Рисунок 15. Редактирование формы «stack-server»

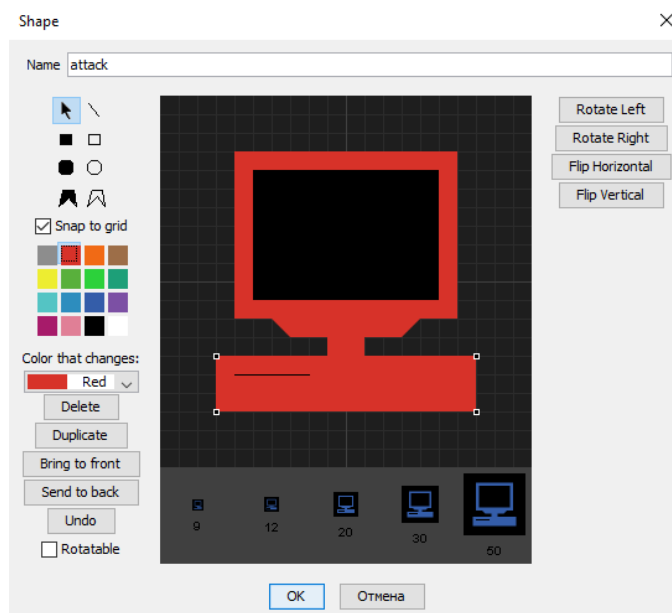


Рисунок 16. Редактирование формы «attack»

После закрытия формы редактора черепах, необходимо перейти к самому коду, где будут описаны все процедуры, поэтому следует открыть вкладку «код» в верхней панели. Вначале в коде следует описать переменные (рис. 17).

```
globals [system max-stack-server]
breed [stack-server]
breed [attack]
turtles-own [energy]
```

Рисунок 17. Описание переменных

Переменная `system max-stack-server` является глобальной. Если переменная глобальная, то она имеет единственное значение, и каждый агент имеет доступ к этому значению.

С помощью ключевого слова «breed» можно определить породу агентов-черепах. Здесь записывается имя переменной такое же, как и при изменении агентов в редактировании форм черепах.

Переменная `energy` относится к агентам и означает, что каждая черепашка обладает собственной энергией.

Для того чтобы модель обновлялась в программном коде следует прописать процедуру «setup», в которой будет содержаться описание параметров сервера и зараженного компьютера, цвет фона, а также форма агентов (рис. 18).

С помощью процедуры «go» модель можно воспроизвести. В коде прописываются, какие действия могут выполнять сервер и зараженные компьютеры (рис. 19).

```

to setup
  clear-all
  set max-stack-server 100000
  ask patches [ set pcolor white ]
  set-default-shape stack-server "stack-server"
  create-stack-server number-stack-server
  [
    set size 4
    set energy 1000
    setxy 0 0
  ]
  set-default-shape attack "attack"
  create-attack number-attack
  [
    set size 2
    set energy random (2 * energy-attack)
    setxy 0 0
  ]
  display-labels
  set sistem count patches with [pcolor = white]
  reset-ticks
end

```

Рисунок 18. Процедура «setup»

```

to go
  if not any? turtles [ stop ]
  ask stack-server [
    death
  ]
  ask attack [
    catch-stack-server
    death
    reproduce-attack
  ]
  tick
  display-labels
end

```

Рисунок 19. Процедура «go»

Для того чтобы зараженных компьютеров становилось больше, следует записать в коде процедуру их размножения, в которой описано, что размножение происходит в случайном порядке и каждый раз когда происходит появление новых атак, то энергия делиться пополам (рис. 20).

```

to reproduce-attack
  if random-float 100 < attack-reproduce [
    set energy random (energy-attack / 2)
    hatch 1 [ rt 0 fd 0 ]
  ]
end

```

Рисунок 20. Процедура «reproduce-attack»

Следующая процедура отвечает за выведение сервера из строя. Когда зараженные компьютеры атакую, у сервера уменьшается энергия. В данной

процедуре описывается, что если сервер начинают атаковать, то у сервера отнимается энергия на единицу и прибавляется к энергии атаки (рис. 21). Конструкция «let prey one-of stack-server-here» создает переменную prey, которая соотносит себя с любым агентом типа stack-server существующим в рамках вселенной, на которую наткнулся другой агент в данной точке.

```
to catch-stack-server
  let prey one-of stack-server-here
  ifelse prey != nobody
    [ ask prey [
      set energy energy - 1 ]
      set energy energy + energy-attack ]
    [set energy energy - energy-attack]
end
```

Рисунок 21. Процедура «catch-stack-server»

Процедура «death» отвечает за гибель агента. Если энергия сервера будет равна 0, то сервер соответственно выведен из строя и агент исчезает из окна «мира».

В последней процедуре «display-labels» записано обращение к агентам и вывод их количества на объекты «экран» и создание графика в поле интерфейса модели.

```
to death
  if energy < 0 [ die ]
end

to display-labels
  ask turtles [ set label "" ]
end
```

Рисунок 22. Процедуры «death» и «display-labels»

Дополнительно в коде можно прописать небольшую защиту для сервера (случайное увеличение стека на небольшой параметр). С помощью данной защиты сервер сможет продержаться дольше и для того, чтобы вывести его из строя необходимо больше зараженных компьютеров (рис. 23). Тем самым, когда у сервера будет энергия приближаться к 0, то случайным образом энергия будет увеличиваться на 10.

```
to save-stack-server
  if random-float 1 < 0.1
    [ set energy energy + 10 ]
end
```

Рисунок 23. Процедура для защиты сервера

В ходе выполнения всех действий, модель должна выглядеть как на рисунке 24.

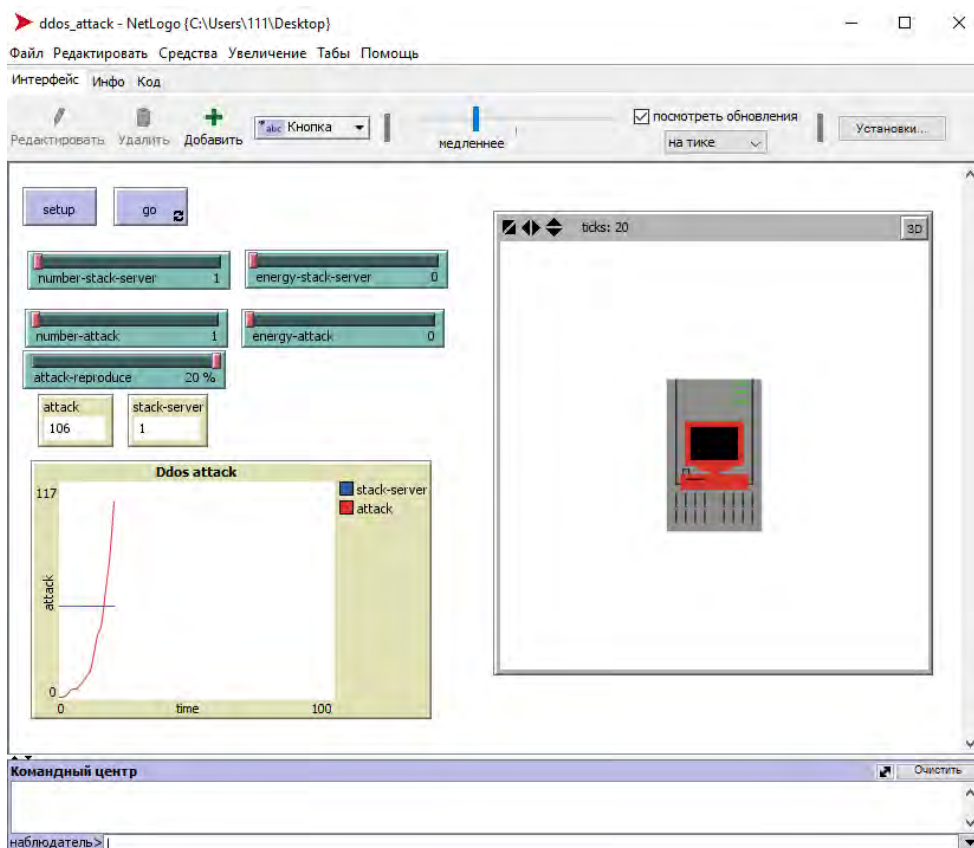


Рисунок 24. Интерфейс модели DDoS-атаки

После создания модели были проведены 2 исследования, в которых было определено за сколько тиков атака выведет сервер из строя. В первом исследовании защита для сервера отсутствовала, а во втором исследовании сервер отбивался вместе с защитой. В таблице 1 представлены результаты исследования без защиты для сервера при разном количестве первоначальных атак.

Таблица 1. Данные исследования №1

№	Первоначальное количество атак	Количество атак после того как сервер упадет	Количество тиков
1.	1	282	34
2.	15	346	19
3.	25	357	16

Таким образом, можно сделать вывод, чем выше количество первоначальных атак, тем меньше времени занимает выведение сервера из строя.

В таблице 2 представлены результаты исследования при таком же количестве первоначальных атак, но при этом к коду добавлена защита для сервера.

Таблица 2. Данные исследования №2

№	Первоначальное количество атак	Количество атак после того как сервер упадет	Количество тиков
1.	1	340	41
2.	15	370	21
3.	25	409	18

Исходя из результатов второго исследования, можно сказать, что если добавить серверу дополнительную защиту, то количество тиков немного увеличивается и возрастает количество атак. Тем самым, сервер сможет продержаться дольше, нежели как в предыдущем исследовании, когда у сервера отсутствовала защита.

Таким образом, в курсовой работе была разработана простая модель DDoS-атаки, которая реализована в мультиагентной среде NetLogo. Кроме этого были проведены исследования, с помощью которых можно увидеть поведение сервера и зараженных компьютеров (атак) и посмотреть за какое время сервер будет выведен из строя.

Данное исследование можно охарактеризовать, как учебное пособие для изучения мультиагентной среды NetLogo и при рассмотрении теоретической стороны, понять, что представляет собой DDoS-атака. Данную систему можно использовать в курсах «Интеллектуальные системы и технологии» и «Защита информации».

Библиографический список

1. Мезенцев К.Н. Мультиагентное моделирование в среде netlogo // Автоматизация и управление в технических системах. 2015. № 1 (13). С. 10-20.
2. Векслер В.А. Моделирование экологических систем в среде netlogo на уроках информатики в средней школе // NovaInfo.Ru. 2017. Т. 3. № 62. С. 327-335
3. Росоха Я.О. Исследование проблематики распределенных атак типа «Отказ в обслуживании» (DDoS) // В сборнике: Телекоммуникационные и вычислительные системы Труды конференции. 2015. С. 50-51.
4. Жилина Е.В., Ромашкин Д.О. Признаки сетевых DDoS-атак и методы их обнаружения // В сборнике: Современные проблемы гуманитарных и естественных наук Материалы XXXIII международной научно-практической конференции. В 2 частях. 2017. С. 20-26.
5. Пальчевский Е.В. Защита от DDoS-атаки HTTP-трафиком на веб-системы управления сайтом // В сборнике: Теоретические и практические аспекты развития науки и образования материалы международной (заочной) научно-практической конференции. Научно-издательский центр «Мир науки». 2016. С. 36-39.

6. Тарасов Я.В. Опыт использования технологий нейронных сетей для обнаружения низкоинтенсивных DDoS-атак // В сборнике: Безопасные информационные технологии (БИТ-2016) Сборник трудов Седьмой Всероссийской научно-технической конференции. Под редакцией В.А. Матвеева. 2016. С. 270-274.
7. Бычков Е.Д., Кладов В.В. Защита Web-сервера от атак типа DDoS на основе модели нечеткого вывода // Актуальные проблемы гуманитарных и естественных наук. 2013. №5. С. 44-49.
8. Бурса М.В., Григорий А.О., Баранников Н.И. Разинкин К.А. Управление риском успешной реализации DDoS-атак на мультисервисные сети // Информация и безопасность. 2015. Т. 18. №4. С. 524-527.
9. Рашевский Р.Б. Шабуров А.С. Практическое применение нейронных сетей для защиты информационно-управляющих систем критически важных объектов от DDoS-атак // Нейрокомпьютеры: разработка, применение. 2015. №10. С. 16-20.
10. Тернова О.С., Жариков А.В., Шатохин А.С. Применение метода Хертса для определения сезонности сетевого трафика с целью раннего обнаружения DDoS-атак // Динамика систем, механизмов и машин. 2016. Т. 4. №1. С. 57-61.
11. Bing Wang, Yao Zheng , Wenjing Lou , Y. Thomas Hou. DDoS attack protection in the era of cloud computing and Software-Defined Networking // Computer Networks. 2015. Т. 81. С. 308-319
12. Sunny Behal, Krishan Kumar. Detection of DDoS attacks and flash events using novel information theory metrics // Computer Networks. 2017. Т. 116. С. 96-110.
13. Monika Sachdeva, Krishan Kumar, Gurvinder Singh. A comprehensive approach to discriminate DDoS attacks from flash events // Journal of Information Security and Applications. 2016. Т. 26. С. 8-22.
14. NetLogo: И взрослым, и детям ту URL: <https://habrahabr.ru/post/220589/> (дата обращения: 15.05.2017).
15. NetLogo URL: <http://letopisi.org/index.php/NetLogo> (дата обращения: 15.05.2017).