Протокол идентификации устройств Интернета вещей методом деградированной флэш-памяти

С.С. Владимиров

Интернет вещей (Internet of Things, IoT) — это современная концепция представления различных устройств, приборов и даже объектов, объединенных в единую глобальную сеть в рамках сетевой инфраструктуры, позволяющей этим вещам взаимодействовать друг с другом и с людьми через сети связи общего пользования (ССОП). На сегодня количество устройств IoT продолжает стремительно расти. Эти устройства используются в большинстве областей человеческой деятельности: образовании, медицине, сельском хозяйстве и других отраслях производства, решая важную задачу автоматизации взаимодействия их между собой и с человеком [1–4]. В рамках обеспечения автоматизации взаимодействия возникают и новые задачи: вопросы адресации такого огромного количества устройств и обеспечения безопасного обмена данными между ними.

Вопрос адресации может быть успешно решен посредством протокола IPv6, имеющего огромный запас свободного адресного пространства [5]. Вопрос информационной безопасности при обмене данными гораздо сложнее, ведь развитие современных технологий привело и к развитию средств для перехвата трафика, его изменения в реальном времени и дальнейшей передачи ложной информации. Например, злоумышленник может провести подмену адреса, представляясь другим участником сети, для осуществления атаки. В связи с этим в рамках общей задачи информационной безопасности встает вопрос надежной идентификации, то есть присвоения каждому сетевому устройству уникального идентификатора, чтобы его собеседник мог надежно определить, с кем он общается в данный момент [6].

Под идентификатором устройства будем понимать выделенный, общедоступный атрибут или набор атрибутов и имен. Как правило, идентификаторы работают в определенной области или конкретных сетях и не всегда применимы для идентификации вещей по всему миру. Устройства современного Интернета вещей обычно имеют более одного идентификатора в существующих сетях.

Следует отметить, что уникальный идентификатор не заменяет адресацию устройств, а дополняет ее. Если адреса устройства используются на сетевом и канальном уровнях для обеспечения передачи данных и маршрутизации пакетов, то идентификатор, как правило, работает на сеансовом уровне. Тем не менее, в зависимости от решаемой

задачи и требований безопасности, сетевые и канальные адреса могут быть использованы и в качестве идентификаторов устройства.

На сегодня для получения уникального глобального идентификатора предлагается большое количество различных программных и аппаратных решений. Условно их можно разделить на две группы:

1. Уникальные идентификаторы, присваиваемые пользователем, производителем или регулятором.
2. Истинно уникальные идентификаторы на основе уникальных характеристик или свойств самого устройства.

Характерным примером идентификатора из первой группы служат аппаратные MAC-адреса EUI-48 и EUI-64, присваиваемые производителем и записываемые в память сетевой карты устройства [7]. Ещё один пример — идентификация на основе архитектуры цифровых объектов DOA [8–10]. К этой же группе идентификаторов можно отнести идентификаторы на основе RFID меток [11, 12]. Идентификаторы первой группы просты в использовании, но обеспечивают меньший уровень безопасности из-за относительной простоты их подмены [11,13].

Вторая группа идентификаторов, как правило, сложнее в использовании, но обеспечивает больший уровень безопасности, поскольку уникальность идентификатора обеспечивается свойствами устройства или его элементов. Например, существует методика идентификации беспроводных устройств по особенностям передаваемого ими радиосигнала [14]. Другим перспективным вариантом является идентификация устройств по свойствам встроенного в них запоминающего устройства. В работах [15, 16] предлагается использовать встроенную в устройства флеш-память в качестве аппаратного генератора случайных чисел и источника уникальных идентификаторов устройства. В статье [17] был предложен метод использования слепка деградированной NAND флеш-памяти в качестве идентификатора сетевого устройства, а в работах [18–20] этот метод был расширен на использование более широко используемой NOR флеш-памяти.

Метод идентификации на основе слепка деградированной флеш-памяти

Большинство современных устройств IoT содержат на плате постоянное запоминающее устройство на основе флеш-памяти либо допускают возможность его размещения, видится логичным использовать это запоминающее устройство как источник уникального идентификатора. В основе идеи лежит тот факт, что при производстве все чипы памяти получаются уникальными из-за непредсказуемых микродефектов при производстве кремниевых чипов [15, 16].

Рассмотрим принцип данного метода на основе флеш-памяти типа NOR, которая представляет собой двумерный массив, состоящий из ячеек памяти, расположенных на матрице проводников. Каждая из этих ячеек памяти может хранить в себе от 1 до 4 бит информации в зависимости от типа ячейки. Структура NOR флэш-памяти представлена на рис. 1.

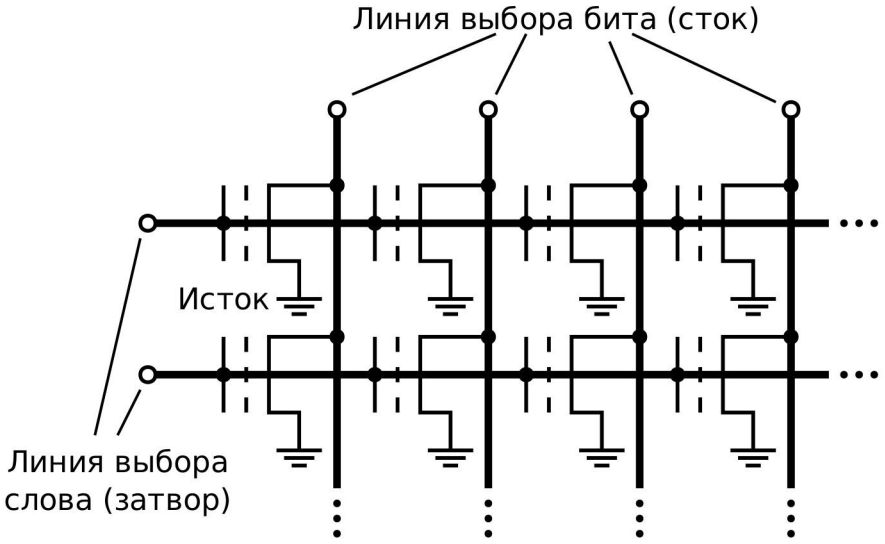

Рис. 1. Структура флеш-памяти типа NOR

Изначальное состояние каждой ячейки памяти равно 1, а при программировании памяти нужные ячейки изменяют заряд и получают нулевое значение. Каждое изменение состояния заряда влечет за собой накапливающиеся непоправимые изменения в структуру ячеек. После большого количества переопределений ячейка деградирует и больше не может вернуться в исходное состояние и ее значение всегда остается в неизменном состоянии 0, то есть создается так называемая плохая ячейка или плохой блок (bad-cell или bad-block). Из-за упомянутой уникальности чипов памяти появление таких деградировавших ячеек невозможно предсказать. На возникновение этих ячеек оказывает влияние и температура чипа, что является идеальным примером генерации физических случайных процессов. Если взять два различных чипа флэш-памяти из одной партии, то бэд-блоки в них тоже будут отличаться [17, 20].

Поскольку в чипах NOR-флеш памяти операции стирания и записи ячеек памяти производятся как правило с целым сектором памяти, то при деградации плохие ячейки начинают появляться по всему сектору памяти. Получается не просто плохая деградировавшая ячейка, а целый деградировавший сектор памяти. Слепок такого деградировавшего сектора памяти может быть использован в качестве уникального идентификатора, который не может быть повторен на другом чипе без дорогостоящего оборудования и значительных временных затрат [18–20].

Подобный же механизм работает и в микросхемах флеш-памяти типа NAND [17].

Необходимо отметить, что для успешной идентификации выбранный деградированный участок памяти должен содержать достаточное количество деградировавших ячеек. Предполагается, что при использовании данного метода устанавливаемые в устройство чипы памяти уже будут содержать участок памяти с заранее созданными бэд-блоками. Для дальнейшего использования данный блок памяти с идентификатором должен быть всегда доступен для чтения микропрограмме устройства и через специальный внешний интерфейс.

Метод идентификации сетевого устройства с использованием деградированного сектора флэш-памяти сводится к следующему алгоритму [18–20]:

1. Принудительная деградация микросхемы флэш-памяти путем циклического переписывания одного из ее секторов вплоть до регулярного появления бэд-блоков.
2. Формирование идентификатора как байтового массива S, соответствующего деградированному сектору микросхемы памяти после серии стираний подряд.
3. Запись массивов-идентификаторов S в базу идентификаторов.

Для идентификации устройства выбранный сектор чипа памяти стирается, его содержимое считывается в идентификатор, который далее сравнивается с соответствующим эталонным идентификатором S из базы идентификаторов.

Точность идентификации напрямую зависит от размера блока памяти, выбранного в качестве идентификатора. Чем больше блок, тем выше надежность идентификации. Однако по мере увеличения идентификатора увеличивается время обработки и время сравнения с базой данных идентификаторов. Для быстрой идентификации можно использовать укороченный идентификатор, например одну или две страницы памяти, а для важных задач — использовать более крупный идентификатор (целый сектор или несколько секторов памяти).

Протокол идентификации устройств IoT на основе деградированной флеш-памяти

При разработке протокола идентификации было принято решение использовать механизм подтверждения идентификаторов на отдельном доверенном устройстве — хранилище идентификаторов, которое позволяет хранить базу идентификаторов в одном месте при условии обязательного резервирования.

В зависимости от решаемой задачи и требований к безопасности хранилище идентификаторов размещается либо в одной локальной сети с идентифицируемыми устройствами, либо в отдельной сети с доступом через ССОП.

Рассмотрим механизм взаимной идентификации двух сетевых устройств IoT A и B через размещенное в сети хранилище идентификаторов C. Изначально в базу данных хранилища внесены эталонные идентификаторы $ID_A^Э$ и $ID_B^Э$ устройств A и B соответственно. В память устройств A и B внесен адрес хранилища.

При получении запроса на соединения устройства A и B обмениваются идентификаторами, чтобы подтвердить свою подлинность. После получения идентификатора ID_B собеседника, устройство A отправляет запрос с сетевым адресом B в хранилище C. Хранилище возвращает A эталонный идентификатор $ID_B^Э$, после чего A проверяет — совпадает ID_B с эталоном или нет. В случае положительного результата устройство A сохраняет ID_B в своей памяти. В дальнейшем сохраненный идентификатор используется для динамической идентификации устройств в процессе работы. Аналогичные действия производит устройство B. Процедура взаимной идентификации показана на рис. 2.

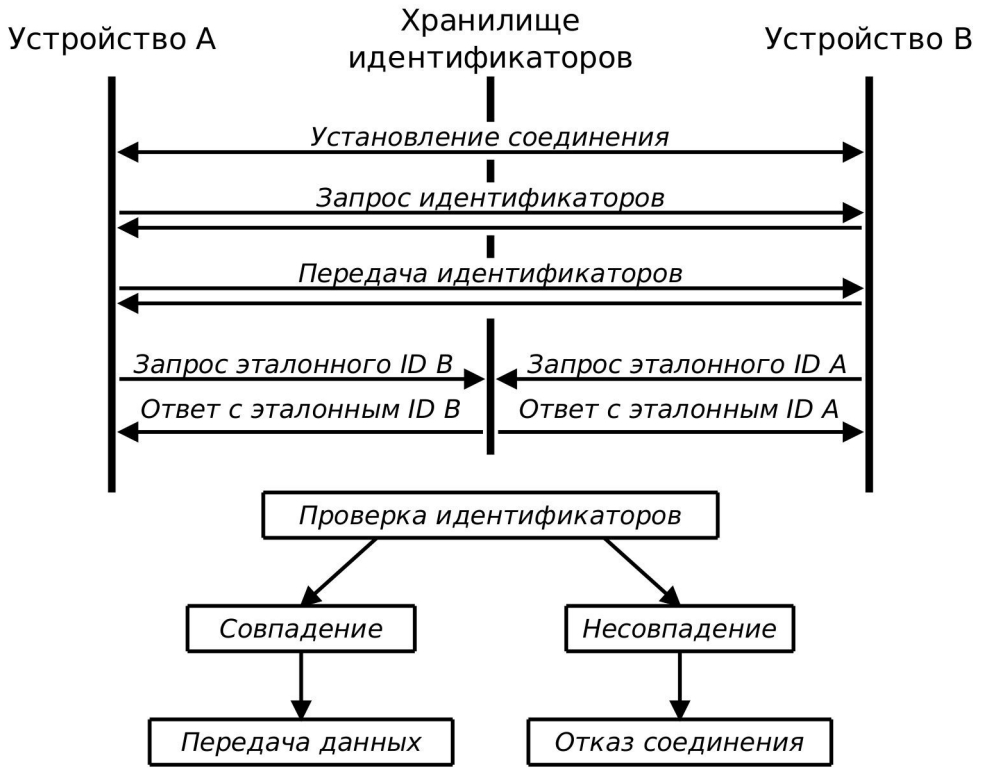

Рис. 2. Процедура взаимной идентификации сетевых устройств

В процессе передачи данных для периодического взаимного подтверждения подлинности собеседники используют укороченный идентификатор, что позволяет сократить временные и ресурсные затраты на идентификацию. При этом можно использовать как статический укороченный идентификатор, например, только первую страницу полного идентификатора, так и различные варианты динамической идентификации, к примеру чередуя укороченные идентификаторы по тому или иному алгоритму. Процедура упрощенной взаимной идентификации представлена на рис. 3.

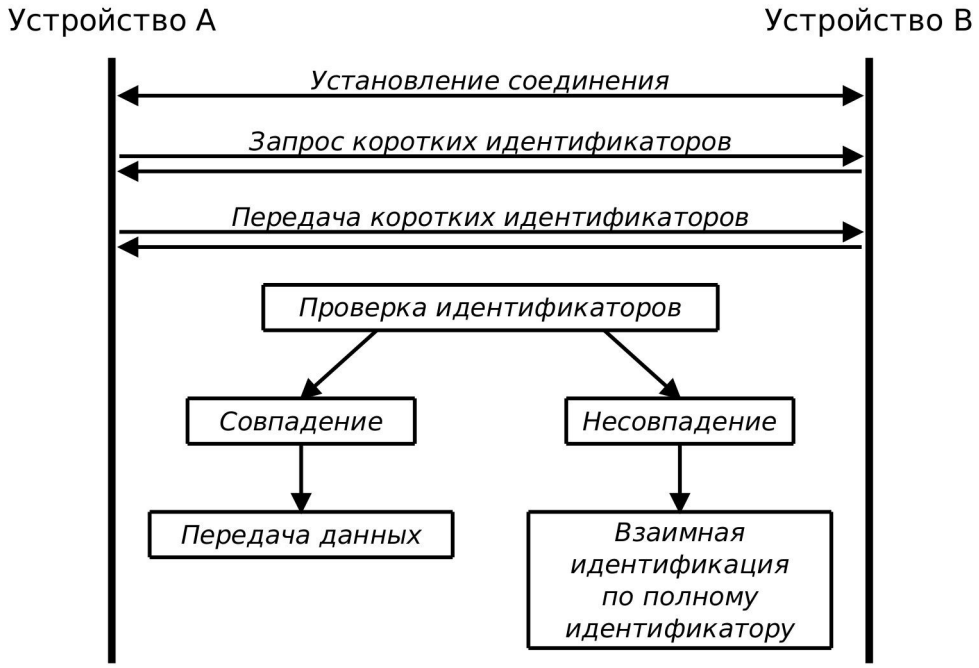

Рис. 3. Взаимная идентификация по укороченному идентификатору

По окончании сеанса связи, сохраненные образы полных идентификаторов собеседников сохраняются и используются в дальнейшем для идентификации, не требующей обращения к хранилищу.

В случае использования сетевых устройств IoT с ограниченным объемом памяти на устройстве лучше хранить вместо полного идентификатора только укороченный статический идентификатор. Это позволит сократить частоту обращения к хранилищу, увеличить количество устройств в собственной таблице идентификации и сэкономить память устройства.

В случае любых ошибок при взаимной идентификации устройств по укороченному идентификатору производится процедура полной взаимной идентификации через хранилище.

Пакет протокола идентификации

Общая структура пакета протокола идентификации представлена на рис. 4. Пакет включает поля: тип сообщения, номер сообщения, размер поля данных, данные.

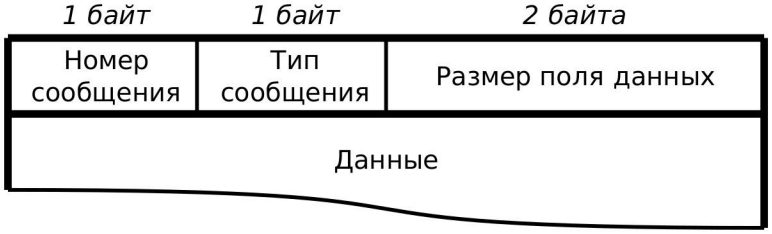

Рис. 4. Общая структура пакета протокола идентификации

Тип сообщения определяет дальнейшее поведение программы и назначение некоторых полей. Существует 3 основных типа сообщений:

1. Запрос. Используется, когда требуется получить от устройства или хранилища идентификаторов данные об идентификаторе.
2. Запрос короткого идентификатора. Используется при упрощенной схеме идентификации. Аналогичен запросу, но несет в себе дополнительные данные, например, положение укороченного идентификатора при динамической идентификации.
3. Ответ на запрос.

Номер сообщения используется для контроля последовательности сообщений. При передаче больших идентификаторов приходится разделять сообщение на фрагменты. Номер сообщения позволяет программе идентификации отслеживать фрагментированные сообщения и производить их дефрагментацию. Нумерация сообщений производится циклически по модулю 255.

Размер сообщения содержит длину данных в байтах, упрощает обработку сообщений, позволяет быстрее считывать данные и отсекать заполнители, добавляемые при малом размере пакета.

Содержание поля данных зависит от типа сообщения:

- Запрос устройству. При запросе идентификатора у другого устройства поле данных остается пустым.
- Запрос хранилищу идентификаторов. При запросе идентификатора у хранилища в поле данных записывается сетевой адрес устройства, для которого необходимо получить полный эталонный идентификатор.
- Запрос короткого идентификатора. При идентификации по укороченному идентификатору в поле данных записывается необходимый размер короткого идентификатора либо его позиция в случае использования укороченных идентификаторов фиксированного размера.
- Ответ на запрос. В поле данных ответа записывается полный или укороченный идентификатор. Если идентификатор полностью не помещается в один пакет, то он разбивается на участки максимально возможного размера и далее передается отдельными пакетами с увеличением значения номера сообщения.

Передача пакетов протокола идентификации по сети

Протокол разработан для передачи поверх протоколов различных уровней сетевой модели OSI от канального до транспортного. Целевыми протоколами являются Ethernet, IP, TCP и UDP.

Для тестирования работы протокола написана программная реализация на языке C++ с использованием кроссплатформенной библиотеки libtins. В дальнейшем для этой реализации проведено тестирование задержек при работе протокола и нагрузочное тестирование хранилища сообщений.

Тестовое хранилище сообщений при исследованиях размещалось на сервере в Амстердаме, а в роли клиентского IoT устройства работал микрокомпьютер Raspberry Pi 3, расположенный в Санкт-Петербурге.

Тестирование круговой задержки проводилось с увеличением размера идентификатора от 128 байт до 4096 байт с шагом в 128 байт и усреднением результата по пяти измерениям. Передача пакетов велась напрямую поверх протокола IP. Результаты тестирования задержки представлены на рис. 5. Точкой показано среднее по пяти измерениям. Линия демонстрирует разброс задержки от минимального до максимального. Эксперимент показал, что задержка практически не зависит от размера идентификатора и в среднем составляет 537 мс.

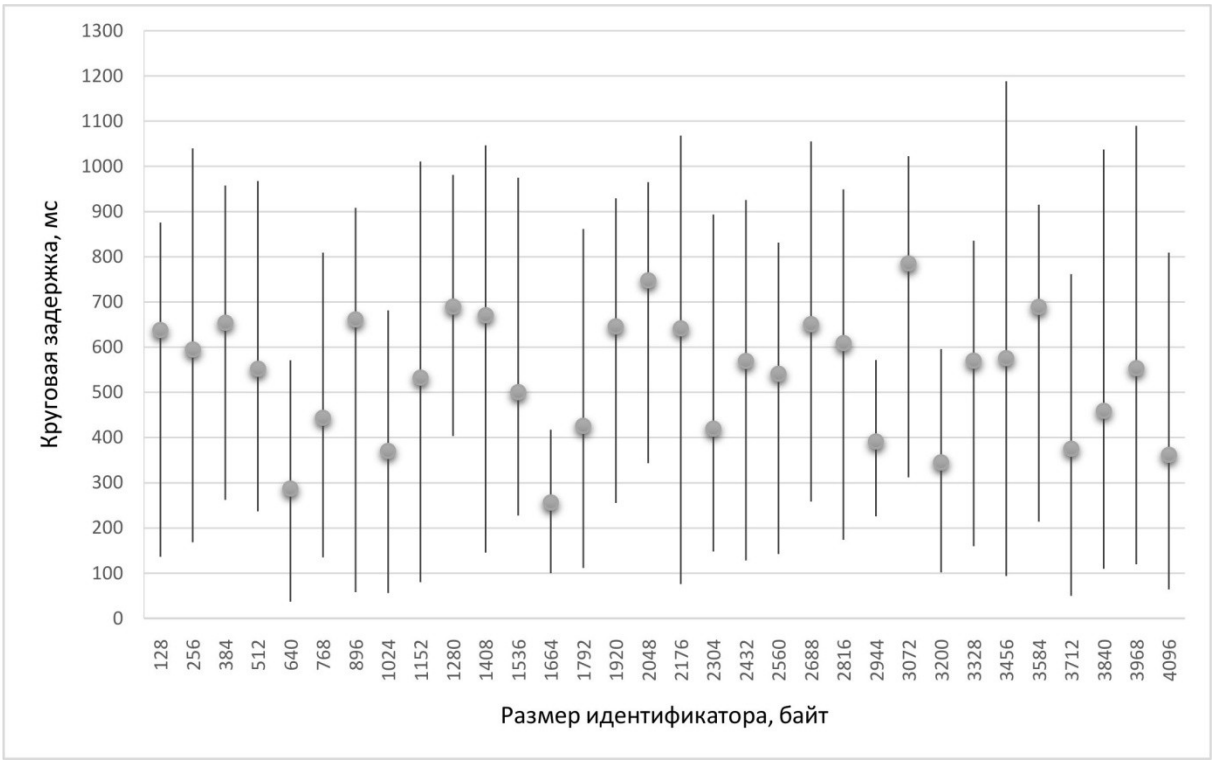

Рис. 5. Круговая задержка получения идентификатора в зависимости от его размера

На рис. 6 приведены результаты тестирования круговой задержки получения идентификатора при использовании протоколов TCP и UDP. Размер идентификатора был установлен равным 1024 байта. Проведено по 25 последовательных замеров для каждого протокола. Средняя задержка для протокола TCP составила 570 мс, а для протокола UDP — 411 мс, что объясняется особенностями работы протоколов.

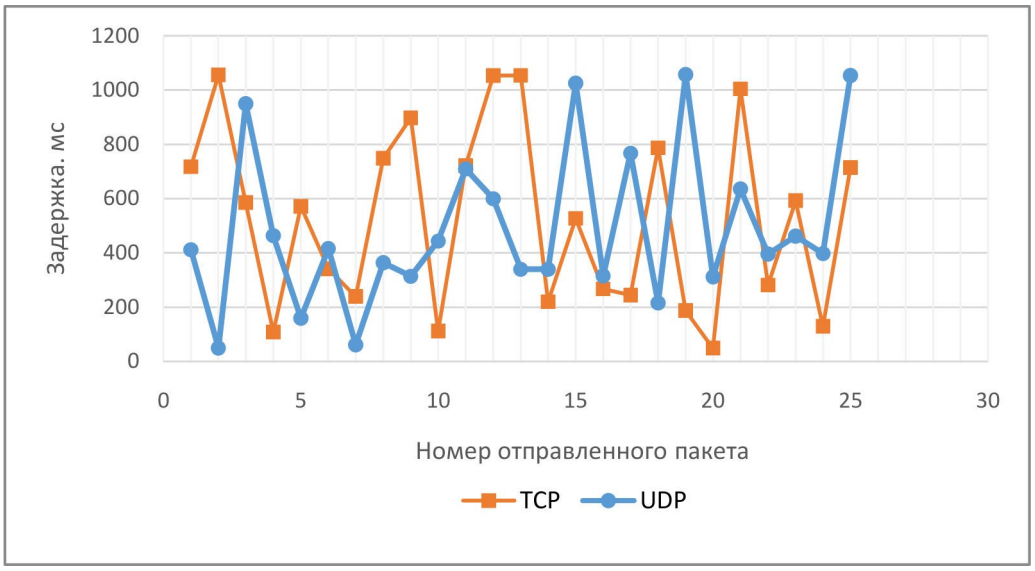

Рис. 6. Круговая задержка получения 1024-байтного идентификатора для протоколов TCP и UDP

Отдельно для протокола UDP была проведена оценка потерь пакетов. При тестировании использовался непрерывный поток запросов идентификаторов при длине идентификатора 1024 байта. На 100 000 запросов было получено 99 957 ответов, т.е. потери составили 0,043% или $4{,}3\times10^{-4}$.

Нагрузочное тестирование хранилища сообщений проводилось для протоколов TCP и UDP с увеличением размера идентификатора от 128 байт до 4096 байт с шагом в 128 байт при непрерывной передаче запросов. Результаты тестирования приведены на рис. 7. На графике показано, на каком запросе начали появляться отказы в обслуживании. Можно видеть, что в данной реализации тестового хранилища лучшие результаты показывает протокол TCP, как имеющий больший приоритет при обработке сообщений сетевой подсистемой сервера.

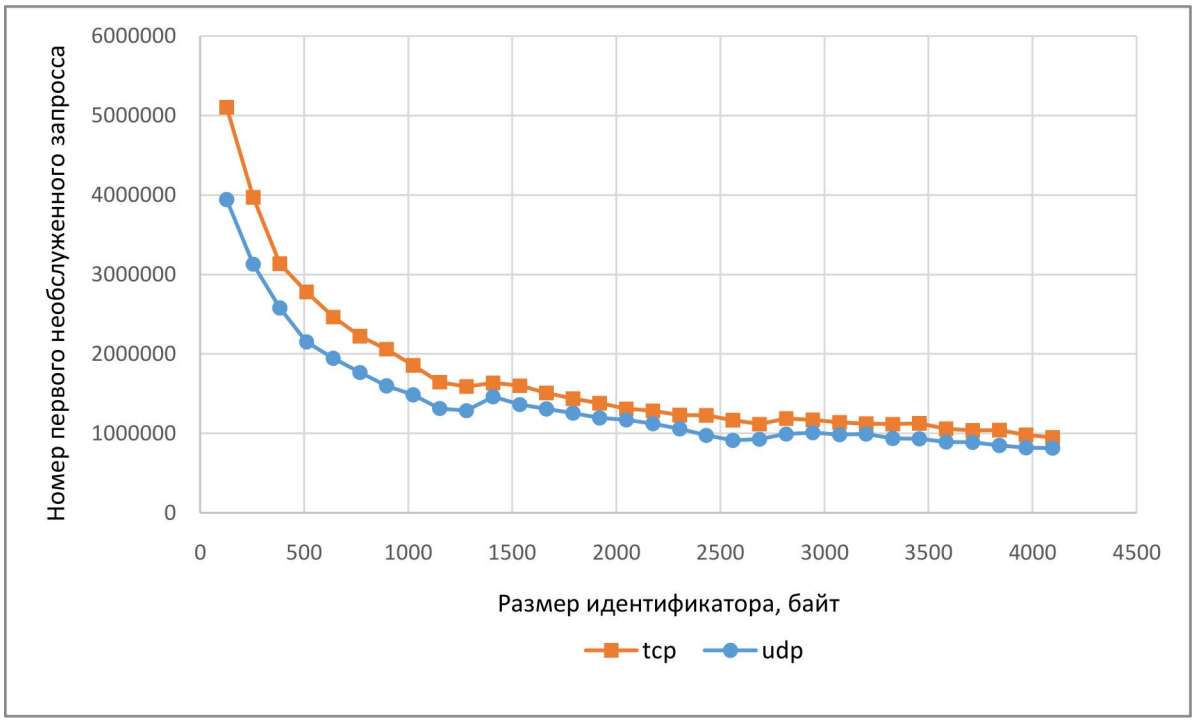

Рис. 7. Результаты нагрузочного тестирования хранилища сообщений

Литература

1. Кучерявый А.Е., Прокопьев А.В., Кучерявый Е.А. Самоорганизующиеся сети. СПб. : Любавич, 2011. 312 с.
2. Кучерявый А.Е. Интернет Вещей // Электросвязь. 2013. № 1. С. 21–24.
3. Jeschke S., Brecher C., Meisen T., Ozdemir D., Eschert T. Industrial internet of things and cyber manufacturing systems // Industrial Internet of Things. Cham : Springer, 2017. P. 3–19. DOI: 10.1007/978-3-319-42559-7_1.
4. Thibaud M., Chi H., Zhou W., Piramuthu S. Internet of Things (IoT) in high-risk Environment, Health and Safety (EHS) industries: A comprehensive review // Decision Support Systems. 2018. Vol. 108. P. 79–95.
5. Deering S., Hinden R. Internet Protocol, Version 6 (IPv6) Specification. RFC 8300. STD 86. IETF, 2017. DOI: 10.17487/RFC8200.
6. Соколов М.Н., Смолянинова К.А., Якушева Н.А. Проблемы безопасности интернет вещей: обзор // Вопросы кибербезопасности. № 5 (13). 2015. С. 32–35.
7. Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID). IEEE, 2017. 19 p.
8. Albahri M., Kirichek R., Muthanna A., Ateya A.A., Borodin A. Combating Counterfeit for IoT System Based on DOA // 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) 2018. P. 8631257. DOI: 10.1109/ICUMT.2018.8631257.
9. Аль-Бахри М.С., Киричек Р.В., Бородин А.С. Архитектура цифровых объектов как основа идентификации в эпоху цифровой экономики // Электросвязь. 2019. № 1. С. 12–22.
10. Al-Bahri M., Yankovsky A., Kirichek R., Borodin A. Smart System Based on DOA & IoT for Products Monitoring & Anti-Counterfeiting // 4th MEC International Conference on Big Data and Smart City (ICBDSC) 2019. P. 14–18. DOI: 10.1109/ICBDSC.2019.8645610.

11. Leloglu E A Review of Security Concerns in Internet of Things // Journal of Computer and Communications. 2017. Iss. 5. P. 121–136.
12. Internet of Things. EU-China Joint White Paper on Internet-of-Things Identification. European Research Cluster on the Internet of Things / Ed. by J. Soldatos, G. Yuming. 2014.
13. Hegde A. Spoofing Detection and Prevention // International Journal of Advanced Research in Computer and Communication Engineering. 2016. Vol. 5. Iss. 1. P. 229–232.
14. DeJean G, Kirovski D RF-DNA: Radio-Frequency Certificates of Authenticity // Lecture Notes in Computer Science. 2007. Vol. 4727. P. 346–363.
15. Wang Y., Yu W., Wu S., Malysa G., Suh G.E., Kan E.C. Flash Memory for Ubiquitous Hardware Security Functions: True Random Number Generation and Device Fingerprints // Proceedings of the 2012 IEEE Symposium on Security and Privacy. 2012. PP. 33−47. DOI: 10.1109/SP.2012.12.
16. Jia S., Xia L., Wang Z., Lin J., Zhang G., Ji Y. Extracting Robust Keys from NAND Flash Physical Unclonable Functions // Lecture Notes in Computer Science. 2015. Vol. 9290. PP. 437−454. DOI: 10.1007/978-3-319-23318-5_24.
17. Jakobsson M., Johansson K.-A. Unspoofable Device Identity Using NAND Flash Memory [Electronic resource] // SecurityWeek: [site]. URL: http://www.securityweek.com/unspoofable-device-identity-using-nand-flash-memory (Accessed date: 24.10.2017).
18. Владимиров С.С., Киричёк Р.В. Методика идентификации устройств интернета вещей на основе принудительной деградации участка флеш-памяти // Электросвязь. 2017. № 2. С. 32−35.
19. Vladimirov S., Kirichek R. The IoT Identification Procedure Based on the Degraded Flash Memory Sector // Lecture Notes in Computer Science. 2017. Vol. 10531. PP. 66−74. DOI: 10.1007/978-3-319-67380-6_6.
20. Vladimirov S.S., Pirmagomedov R., Kirichek R., Koucheryavy A. Unique Degradation of Flash Memory as an Identifier of ICT Device // IEEE Access. 2019. Т. 7. С. 107626–107634. DOI: 10.1109/ACCESS.2019.2932804.