Исследование методов распределения ресурсов для систем умного города

М.С.А. Мутанна, М. М. Мутанна, А. Мутханна, С.С.С. Нассер, Ю. Т. Лячек
Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
muthanna@mail.ru, sakr85@mail.ru

***Аннотация*: В связи с широким использованием датчиков, для создания умного города необходимы надлежащие способы сбора данных, которые могут быть дополнительно проанализированы для предоставления нескольких услуг в умном городе. В данной статье рассматривается архитектура ПО сервиса IoTDM, также определены сущности, которые согласно спецификациям oneM2M используются для создания дерева ресурсов Интернет Вещей в качестве локальных серверов для распределения ресурсов в умном городе.**

Ключевые слова: умный город; граничные вычисления; Интернет вещей; распределения ресурсов

Введение

Разработка мобильных сетей пятого поколения 5G, а именно обеспечение связи и оптимального уровня трафика, вызывает трудности из-за огромного роста числа беспроводных устройств (например, смартфоны) [1, 2]. Ожидается, что мировой трафик, по сравнению с 2010 годом, возрастёт в 200 раз уже к 2020 году. А в 2030 ожидается подъём в 20000 раз. Одной из основных целей 5G является достижение скорость передачи данных около 10 Гбайт. Кроме того, сети пятого поколения обещают достичь высоких показателей в пропускной способности, малых задержках, надёжности, связности и мобильности. Для решения таких проблем, как загрузка трафика и проблемы подключения, а также для достижения ожидаемой скорости передачи данных со сверхнизкой задержкой, новая сотовая система 5G должна использовать новые технологии в различных точках сети. Некоторые технологии, такие как программно-определяемая сеть (SDN) и виртуализация сетевых функций (NFV), должны использоваться в основной сети [3].

Структура MEC производит облачные вычисления прямо рядом с пользователем – дословно в шаге от него [4]. Использование MEC, означает уход от централизованных больших центров обработки данных к распределённым небольшим центрам обработки данных с ограниченными возможностями по сравнению с централизованными единицами. Благодаря этому можно достичь больших успехов и выгоды. Мобильное облачное вычисление – это несомненно тренд в сфере облачных вычислений, т.к. все вычисления переносятся в мобильное [8] переносное устройство. Развитие коммуникационных технологий LPWAN так же имеет большие успехи [7]. С точки зрения лицензирования частотного спектра, технологии Интернета вещей [5, 6] можно разделить на такие составляющие: работающие в разрешённом и неразрешённом спектре. Первую категорию представляют Lora, Sigfox и т.д. Большинство из них являются нестандартными. Вторую составляют уже известные всем сотовые коммуникационные технологии 2G/3G (такие как GSM, CDMA, WCDMA и другие), LTE технология, поддерживающая разные виды терминалов. Стандарты для этих коммуникационных технологий, работающих в разрешённом спектре, разработаны международными организациями стандартов. Например, 3GPP (GSM, WCDMA, LTE и другие), а также 3GPP2 (CDMA и т.д.) [9].

В данной статье рассматривается дерева ресурсов Интернет Вещей в качестве локальных серверов в архитектуре ПО сервиса IoTDM как многоуровневая система граничных вычислений для распределения ресурсов Интернета вещей.

I. Архитектура дерева ресурсов IoTDM разработанной модели

Для централизованного сбора и управления Интернет Вещами требуется соответствующий сервис, который обеспечит необходимые функции менеджмента в условиях разработанной модели сервиса «Умного города». Так же для дальнейшего развития архитектуры «Умного города» требуется система, построенная, как уже говорилось ранее, на основе определенных стандартов взаимодействия Интернет вещей, сервиса менеджмента и соответствующих приложений. И в соответствии с данными требованиями к рассматриваемой модели, как было ранее определено, в качестве системы менеджмента «Умного города» для исследования возможности организации похожих систем именно на этом сервисе, а также тестирование протоколов Интернета Вещей, при организации передачи данных поверх сети SDN, был выбран сервис IoTDM (InternetofThingsDataManagement).

Платформа Opendaylight по-сути является брокером между Интернет Вещами и подключенными приложениями

и также выступает хранилищем данных. При этом, согласно спецификаций проекта oneM2M, IoTDM реализует хранение данных в виде дерева ресурсов, которое строится согласно построенным отношениям между Интернет вещами, а также приложениями. На рис. 1 приведена архитектура программного обеспечения сервиса InternetofThingData Management (IoTDM).

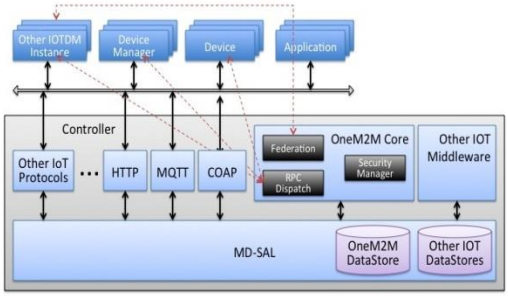

Рис. 1. Архитектура ПО сервиса IoTDM

Платформа Opendaylight была выбрана для реализации спецификаций oneM2M из-за множества заложенных в нее функций, таких как:

- MDSAL, транзакционное хранилище данных.
- Гибкая поддержка протоколов, заложена возможность масштабирования.
- В данной архитектуре предусмотрена кластеризация.
- Заложен потенциал для эффективного использования AAA (Authentication, Authorization, Accounting).

Согласно спецификациям, разработанным партнерским проектом oneM2M, каждая из физических Интернет вещей (M2M) представляется как «сущность» (определенное представление в цифровом мире) и имеет определенные отношения с другими сущностями, который в свою очередь определены спецификациями oneM2M. Тем самым разграничиваются права доступа на ту или иную Интернет Вещь или на целое множество подключенных Интернет Вещей.

Следует также учесть, что для удобства работы с сервисом, OpendaylightFoundation на странице своего проекта опубликовала разработанный API, как на языке Python 2.7, так и на Java. Что облегчает разработку, то есть не требуется работать напрямую с сетевыми библиотеками выбранного языка программирования и составлять запрос, начиная с формирования шаблона передачи данных — JSON. Вся работа сводится к «импортированию» (включению) данного разработанного API к своему программе (скрипту) и работы с соответствующими функциями, написанными в API.

Цель работы исследовать возможно создаваемый трафик Интернета вещей в концепции «Умного города» а именно трафик, который бы генерировался только от сервиса мониторинга экологической обстановки. Но для построения модели взаимодействия требовалось провести исследование на предмет количества требуемых для установки Интернет Вещей, поэтому за основу была взята модель стандартного перекрестка. За основу «Умного города» в данном исследовании рассматривается например Центральный район города Санкт-Петербурга.

Архитектурная модель умного города выглядит следующим образом:

- Для равномерного распределения нагрузки на сетевые элементы, за каждым openflow-коммутатором было привязано 2 муниципальных образования рассматриваемого района.
- В каждом муниципальном образовании было предложено задействовать около 40 перекрестков.
- Каждый стандартный перекресток имеет 4 светофора, которые играют роль Интернет Вещей.
- Каждая Интернет Вещь содержит 3 типа датчика для контроля экологического состояния окружающей среды, два из которых генерируют данные каждую секунду.

Работа модели построена по следующему алгоритму:

1. Построение дерева ресурсов oneM2M с учетом иерархического разделения на муниципалитеты, перекрестки, светофоры.

2. Инициализация Интернет Вещи. Каждая Интернет Вещь при инициализации к дереву ресурсов сервиса IoTDM отправляет запрос, который так же содержит информацию о готовности датчиков, при их корректном подключении. Информация с GPS датчика приходит единожды. Для регистрации Интернет Вещи также требуется пройти аутентификацию, в данном случае настройки были базовыми: логин — admin, пароль — admin.

3. Генерация трафика Интерна Вещей. В процессе работы, каждая Интернет Вещь посылает запросы каждую секунду (ContantInstance, по спецификации oneM2M) к дереву ресурсов, содержащие данные двух датчиков, время их регистрации, номер значения и другие метаданные. Также была организована передача с каждого перекрестка последовательно каждой Интернет Вещью, при этом запросы к дереву ресурсов (сервис IoTDM) производились одновременно от каждого перекреста. В результате получилась модель, содержащая 960 Интернет Вещей, при этом в один момент времени работало с сервисом 240 Интернет Вещей и происходила запись в Базу Данных.

Для построения дерева ресурсов, согласно разработанной модели был разработан соответствующий скрипт на языке python 2., крипт создания дерева ресурсов, а именно создание головного узла (сущность InCSE) с именем «Central_District_SPB», Также приведен второй скрипт, цель

которого — создать остальную часть дерева ресурсов, отвечающую за определенную часть модели, а именно муниципалитета. Также для работы второго скрипта был доработан API, разработанный сообществом Opendaylight и на основе его и разработанной исследуемой модели был разработан собственный API, который использовался как для создания части дерева и работы трафик генератора.

Так как реализуемая модель является слишком большой для полного ее отображения на экране монитора, на картинке конечными точками (овал №5) отображаются только контейнеры, описывающие перекрестки. Но согласно общей архитектуре каждый из перекрестков имеет 4 светофора (4 контейнера), каждый из которых имеет три типа датчика, соответственно 3 под-контейнера, в которых в свою очередь появляются новые данные, посылаемые каждой Интернет Вещью.

II. Разработка генераторов трафика M2M

В статье была описана модель взаимодействия Интернета вещей в концепции Умного города. Также были даны определения и фактические примеры на созданном дереве ресурсов каждой из сущностей.

Для разработанной модели взаимодействия M2M в рамках разработанной модели были разработаны генераторы трафика, эмулирующие соответствующую данной модели нагрузку как на рассматриваемый сервис менеджмента, так и на сеть SDN, которая выступала в качестве сетевой инфраструктуры.

Сервис IoTDM имеет поддержку трех протоколов Интернета Вещей: HTTP, CoAP, MQTT.

Средства разработки:

- язык программирования (ЯП): Python 2.7;
- IoTDMPython API;
- дополнительно также использовались такие библиотеки, как: threading, multiprocessing, os, time, random, unittest и др.;
- разработанный API ;
- операционная система: LinuxUbuntu 16.04 LTS;
- IDE: SublimeText 3

Для удобства разработки был написан собственный API, который также работал с основным API , разработанным сообществом Opendaylight. Разработка собственной API - прослойки позволила уменьшить основной код генератора, сделала его удобно читаемым, имена функций интуитивно понятно отражают суть задачи, которую они выполняют.

Генератор имеет модульную архитектуру, что позволяет за короткое время преобразовать его для распределения на большее количество машин.

- Глобальный модуль — муниципалитет (всего 6 модулей).
- Подмодуль глобального модуля — перекресток (40 подмодулей в каждом глобальном модуле).
- Подмодуль глобального модуля — модуль запуска генератора, то есть одновременного включения 40 перекрестков и соответственно Интернет Вещей.

Для примера приведем модуль перекрестка и модуль запуска генератора трафика для передачи данных по протоколу HTTP.

Алгоритм запуска генератора трафика на HTTP:

1. запуск скрипта MANAGER.py — построение основного дерева.
2. Запуск скрипта shell — например, StartTest78.sh. – инициализация интернет вещей и последующий процесс генерации данных.

Время работы генератора установлено жестким значением, а именно 90 минут или 1,5 часа.

III. Результаты натурного эксперимента

На локальных серверах (cse в случае дерево ресурсов IoTDM) собиралась статистика, и аппроксимировалась, и дальше уже отправлялась в удалённое облако. т.е сжатый формат. Работает по принципу микрооблак. В качестве эксперимента использовали генератор трафика, вкоторых генерировались 2 матрицы по 5*5, считались сумма, и потом дальше отправлялся средний результат сумм матриц от каждой пары.

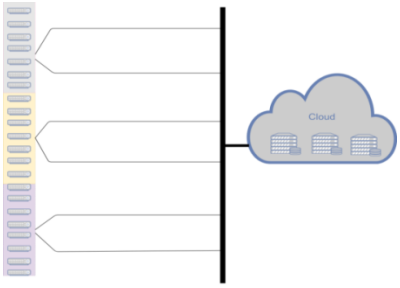

Рис. 2. Структура сети.

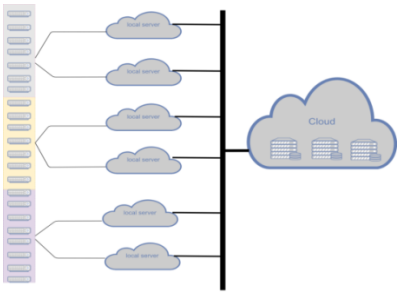

Рис. 3. Применение локальные сервисы.

Суммарная загруженность ЦПУ при использование локальные сервисы лучше.

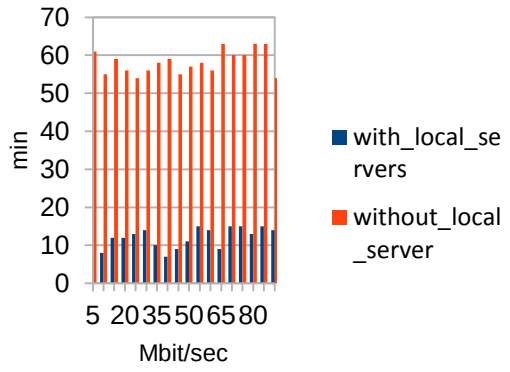

Рис. 4. Загрузка ЦПУ при использовании локального сервиса

IV. Заключение

В статье была рассмотрена архитектура ПОсервисаIoTDM, также определены «сущности», которые согласно спецификациям oneM2M используются для создания дерева ресурсов Интернет Вещей.

Также была рассмотрена модель взаимодействия клиент/сервер при использовании сервиса IoTDM (InternetofThingsDataManagement). Приведены примеры организации работы с сервисом, как с помощью REST API, так и с помощью предлагаемого сообществом OpendaylightPython API.

Приведены примеры создания основных видов сущностей с помощью Python API. Также подробно была разобрана архитектура разработанного дерева ресурсов IoTDM, построенного в рамках модели «Умного города».

Разобран пример разработанного генератора данных с устройств IoT по протоколу HTTP в рамках исследуемой модели.

Список литературы

[1] Recommendation ITU-R M.2083: IMT Vision, ―Framework and overall objectives of the future development of IMT for 2020 and beyond,‖ Sep. 2015.

[2] Muthanna A., Masek P., Hosek J., Fujdiak R., Hussein O., Paramonov A., Koucheryavy, A. Analytical Evaluation of D2D Connectivity Potential in 5G Wireless System // Lecture Notes in Computer Science. 2016. Vol. 9870. pp. 395–403.

[3] Klas G. I. Fog Computing and Mobile Edge Cloud Gain Momentum Open Fog Consortium // The Eleventh International Conference on Systems and Networks Communications (ICSNC). 2016. 122 p.

[4] Vladyko A., Muthanna A., Kirichek R. Comprehensive SDN Testing Based on Model Network // Lecture Notes in Computer Science. 2016. Vol. 9870. pp. 539–549

[5] Masek P., Fujdiak R., Zeman K., Hosek J., Muthanna A. Remote Networking Technology For IoT: Cloud-Based Access For Alljoyn-Enabled Devices. Proceedings of the 18th Conference of Open Innovations Association FRUCT and Seminar on Information Security and Protection of Information Technology 2016. Pp. 200-205.

[6] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017

[7] Low Power Wide Area Network – LPWAN Technology Decisions: 17 Critical Features.URL: http://www.weightless.org/membership/hvVs4ZGQqr5dwCDlBiYX

[8] V. Frascolla, F. Miatton, G. K. Tran, K. Takinami, A. D. Domenico, E. C. Strinati, K. Koslowski, T. Haustein, K. Sakaguchi, S. Barbarossa and S. Barberis, ―5G-MiEdge: Design, standardization and deployment of 5G phase II technologies: MEC and mmWaves joint development for Tokyo 2020 Olympic games,‖ In Standards for Communications and Networ

[9] 5G PPP Architecture Working Group white paper, View on 5G Architecture,‖ July 2016.