International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue V, May 2018- Available at www.ijraset.com

A Survey on Device Discovery in Internet of Things

Vandana CP[1], Sivabalan[2], Maseera Shajid[3]
[1] *Professor, Dept. of ISE, New Horizon College of Engineering, Bangalore, India,*
[2]*Professor, Dept. of CSE, New Horizon College of Engineering, Bangalore, India*
[3]*Student, Dept. of ISE, New Horizon College of Engineering, Bangalore, India*

Abstract: Internet of Things (IoT) is gaining immense importance in today's world of fast connectivity. The Internet of things (IoT) enables us to connect and exchange data by forming a network of physical devices, vehicles, home appliances, and other items which are embedded with electronics, software, sensors, actuators, and connectivity. Internet of Things (IoT) has changed our lives by bringing a technological revolution. The imagination is the limit regarding the new devices that may arise in the market. With great diversity in the devices and the need to integrate new devices in our lives, there is no structure prepared for emerging diversity in IoT. This paper aims to provide some help by studying the various ways that contribute to the automation of part of the device integration process.

IoT devices like unmanned aerial devices, robots and wearable equipment, etc., are widely used which occur in an organizations' inner network and can create a potential threat to the security of the intranet. In this paper we also study a system named IoT Eye, which discovers IoT devices in real time and can discover vulnerabilities before malicious actors and reduces cyber menace to the organization's network.

Keywords: Internet of Things, device discovery, automation, integration, multiple pattern matching.

I. INTRODUCTION

The Internet of Things (IoT)[1] seamlessly connects physical objects with the Internet. This trend enables creating consumer centric applications and services in various domains like intelligent home automation (smart home), health care, intelligent transportation system, environmental monitoring etc. It is estimated that 50 billion smart objects will be connected to the Internet by 2020. The devices must communicate with the environment and among themselves to provide value added services to the end users. This interaction facilitates exchanging and processing metadata and reacting automatically to the environment. This communication may be made through various communication protocols; however, the aim is always the sending and receiving of data between devices and the ability to communicate. The purpose of the Internet of Things: (i) Transform the objects into connectable objects or smart objects providing them with some intelligence and the means to allow them to share their own information. For this to be accomplished it is necessary to integrate the hardware that will make it possible for the object to communicate with any platform. (ii) Another important goal is to create the necessary conditions for these objects to communicate with each other (within the same network) and globally (Internet), allowing to add even more value to the information obtained. This communication is one of the main challenges of IoT nowadays, given the diversity of objects, platforms (Iotivity, HomeKit, AllJoyn, etc.) and methods of communication (WiFi, Bluetooth, etc.). The growth in the number of existing objects is increasing the amount of data generated and the main goal will be to explore well this information and make all efforts to get various benefits through the IoT world.[2]

The definition of Internet of Things(IoT) is: "A pervasive and ubiquitous network which enables monitoring and control of the physical environment by collecting, processing and analyzing the data generated by sensors or smart objects." [3] "The basic idea of the IoT is that virtually every physical thing in this world can also become a computer that is connected to the Internet. To be more accurate, things do not turn into computers, but they can feature tiny computers. When they do so, they are often called smart things, because they can act smarter than things that have not been tagged."[4]

A. The architecture of INTERNET of Things Consists of four Different Layers

1) *Interaction layer*- This layer includes all the hardware devices physically attached to an object that enables its communication with other devices and the Internet. This layer allows the interaction of the objects with the real world and the information generated is sent to the upper layer.
2) *Representation layer*- responsible for managing each of the objects, providing an identification method (e.g. the Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)) and a way of establishing communication with each object using its own methods. Exchange of information between objects is possible through this layer, allowing its use by the top layer.

3) *Service layer*- provide the functionality and information from all the objects connected. This layer allows users to use this information in a standardized way.

4) *Application layer*- formed by many types of applications where the features and the information provided by the Service Layer are used.

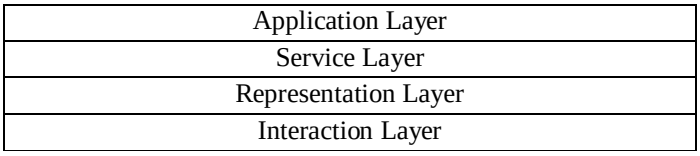

Figure 1: Architecture of Internet of Things.

B. *There are Several Connection Modes that Allow Objects to receive And send information. The Connection Modes can Be One of Four Types*

The most commonly used communication protocols include Bluetooth, Zigbee, Z-Wave, Ethernet, Wi-Fi, Cellular, NFC, etc. There are four of the main IoT platforms in the market. A brief summary of them is presented:

1) AllJoyn[3]- AllJoyn is an open source Application Programming Interface (API) which was initially developed by Qualcomm Innovation Center, Inc. Now the AllJoyn project is in the hands of AllSeen Alliance. It provides a simple work environment for the IoT objects which allows the objects to be easily discoverable and at the same time guarantees security. AllJoyn compromises to reduce the time, effort, and cost of adding advanced features to apps and help ensure interoperability across device types and operating systems.There are now more than 200 companies using AllJoyn API and that can show the success of this Platform.

2) IoTivity[5][6]- It is an Open Source Project hosted by the Linux Foundation and sponsored by the Open Interconnect Consortium (OIC). This project aims to develop seamless open source software framework to connect the billions of devices in the emerging IoT across multiple operating systems and network protocols. Services like smart home, automotive, industrial automation, and healthcare can be made interoperable by using open source implementation over various platforms. Billions of wired and wireless devices can be connected to each other and to the internet which is the architectural goal which is an extensible and robust architecture that works for smart and very small devices.

3) Apple HomeKit[7]- It is Apple's platform for home automation and it was introduced at the 2014 edition of the Apple Worldwide Developers Conference. In relation to the other 3 platforms, this platform covers less devices because it is designed only for the use of home automation. It is not an open source framework, so you have to submit an enrollment form to join the Home Kit program. This is not enough because the IoT world is scattered and this success is directly tied to the companies will to add Home Kit support in their devices. The main advantage of this platform is to guarantee that home automation accessories compatible with Home Kit, can all be integrated into a single coherent whole without vendors having to coordinate directly with each other.

4) Google Brillo[8]- Google is the latest entrant to the crowded IoT market. Presented on Consumer Electronics Show (CES) 2016 Google Brillo is meant to make tiny IoT applications, highly efficient, very fast. This open source platform is composed by a lightweight embedded OS based on Android, core services and a developer kit. Brillo uses Google's own communication protocol (Weave) that is based on Bluetooth and WIFI together to communicate with other devices. Google is already working with hardware partners to certify the boards that are compatible with Brillo knowing that the OS can run on low-end devices with at least 128MB of storage and 32MB of RAM. As well as Home Kit, Brillo success depends on companies' adoption and their hope relies on the widely used Android that can bring some advantage on this ride.

II. RELATED WORK

A. *Prototype on Device Discovery*

The IoT devices present in our homes connect to the internet using the network in our homes. In [2], a prototype is discussed which is software that can listen the home network and see communication of the devices. It analyses communication and with some techniques, it automatically discovers and identifies each device that is communicating in the home network. We can automate the integration process by identifying each device. The prototype will provide an example of this automation technique.

The architecture of the prototype has 3 main components which are- (i) IoT devices that send information through network. (ii) Computer that runs Charles Proxy and (iii) a device that receives information from the other devices.

Though, IoT devices send information to the destination (in this case the smart-phone), the computer will be able to see the data. Prototype dataset is prepared using this information and will also serve as input to test the prototype and try to identify the device that is communicating. Information received from a device is a communication file. eXtensible Markup Language (XML) or JavaScript Object Notation (JSON) formats are used by devices to send communication files. Some of the communication files, even with the Secure Socket Layer (SSL) certificate installed on Charles Proxy, remain encrypted and cannot be retrieved. Application for the prototype was developed using Iphone Operating System (iOS). The app receives the communication files and perform two main tasks: (i) create a database that will store the received communication files after some pre-processing; and (ii) identify the type of a device based on the received communication file. The application source code is available at [9].

Figure 2: Prototype Architecture

In the proposed system, importing the data is the most crucial step because we need to store the information received from communication files in the right way so that the information is ready to be used by the algorithm. A communication file can have more than one property.

In the pre-processing phase a total of nine categories were added: “Security”, “Other”, “Health”, “Activity”,” Sensor”,” Controller”,” Light”,” Display”,” Utility” based on an analysis of the gathered data. [2]

For each line of the database table we store the following parameters: “Property”, “Value”, “Device”, “Type (Category)”, “Value Type” and “Parent File”. After importing all the 37 communication files we were able to retrieve a total of 17547 proprieties into the database table.[2]

The next stage of the process is to identify the device, or its category. This identification can be made with different algorithms (detailed below). To evaluate the performance of the algorithms used, the following values are used: the number of correct matches that are correctly classified (true positives (tp)), the number of incorrect matches classified as positive (false positives (fp)), the number of correct matches classified as negative (false negatives (fn)) and the number of incorrect matches classified as negative (true negatives (tn)). From these values it is possible to derive the usual evaluation metrics (precision, recall and F-measure) [2]

1) *TF-IDF Table :* Besides the manual keyword table, we also use a special table that is based on a different statistic approach. Using the TF-IDF value we can generate a table where we can find the most significant words inside each category. TF-IDF algorithm value is calculated “by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TF-IDF numbers than common words such as articles and prepositions” [11]. To find the most relevant properties in each category, in this prototype, we calculate the frequency of a property in all communication files, comparing it with the frequency of a property in the communication files of a specific category. The algorithm generates a list of the devices and categories that have the highest TF-IDF values for that device.

As intermediate result we will have one table for each type. Each table have a list of keywords and their respective value calculated with TF-IDF algorithm, for simplicity we also add a column with the value discretized to 0, 1. We observe the top 5 output keywords for the algorithm applied to type “Light”. The keywords “intensity” and “color” are very good results and can help

improving the discovery precision, however "time" and "points" are generic words and not specific to "Light" type. The word "intensity" has a higher value because the keyword "intensity" doesn't exist in any document of other device type. In this case, if the discovery algorithm finds a match with "intensity", it will increment 0.28 points on type "Light".

The discovery of devices in the database reach 0.70 for precision and 0.67 for recall. The same cannot be said for the new devices discovery results, with a precision of 0.42 and recall 0.32. The main reason of this lower values is the fact that some keywords are not specific and are more general. [2]

2) *Levenshtein Algorithm :* The Levenshtein Distance Algorithm[10] is a string metric for measuring the difference between two sequences and "it calculates the minimal costs required to change a string of segments into another by means of insertions, deletions or substitutions". For instance, imagine that we have two property values (strings), one from the new device and one from the database, the Levenshtein Distance value will be the number of modifications needed to transform one string into another. The insertion and deletion of a character in the string have a cost of 1 unit and the substitution of a character has a cost of 2 units. This algorithm tends to identify similarly named properties.

After some trial and error tests it was found that the best results with a Levenshtein distance value of <4. The number of correct matches increases as the Levenshtein distance rises because it will discover more similar words on the correct properties. However, for distances higher than 3, the incorrect matches start to grow more quickly than the correct matches and therefore the precision and recall values start a decreasing curve.

The average precision value was 0.70 and the recall 0.63 giving a f-score of 0.65. Once again, we can verify more positives results on the correct type. The precision reduced slightly to 0.68 (-0.02) but the recall value rises to 0.66 (+0.03) leading to a better f-measure value of 0.67 (+0.02). In exception of the type" Display" all the recall values are above 0.5. The worst recall results (" Display"," Activity" and" Security") appear on types that only contained one device of its kind in database, this can indicate that, with more devices on test database, the results on type discovery could be better. [2]

3) *Synonyms Match:* As the objective was to determine similar properties a dictionary of synonyms is used in the proposed prototype. The main objective is to use each property name in communication file, and, on device discovery, "transform" it in multiple properties with the same meaning. We try to improve our results introducing synonyms on the matching algorithm. This way, we will try to match not onlyeach property but also all the synonyms related to each property. This leads to a drop on precision and recall values to 0.66 and 0.64 respectively (difference of - 0.02 in both cases comparing with Levenshtein Algorithm). Regarding the tests for new devices, we can say that the results were better. However, the matched synonyms words were more assertive on the correct devices types providing an increase of 0.02 on precision and recall comparing to Levenshtein tests. [2]

4) *Multi Property Matching*: All the above approaches were property-based, i.e. the attributes used both for training and for testing were those of a single communication of one property per example. This research path was pursued because data, although rich in properties, was poor in devices due to the encrypted communication formats used by many devices and also because the identification timing is important and the possibility of identifying a device using a single property set action had to be investigated. Of course, having more information on each device should entail better results, so a final experiment was necessary to assert how the use of accumulated information for one device over a certain time,
where several of its properties appear, can affect the results. We run the TF-IDF algorithm over the proprieties generated with this method to obtain the most type specific ones (the proprieties that appear frequently on a specific type and not on the remaining). Due to our number of devices used in tests, we have to manually revise these tables. This last step should not be necessary for a larger database. A technique named K-fold Cross Validation was used to evaluate the results. This technique divides data set into k folds. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k - 1 subsamples are used as training data [12].

The advantage of using this technique is that all the devices in the dataset are eventually used for both training and testing. With the Multi-Property matching method, we reach the best result of all the methods tested in this work. The test obtained an average precision of 0.91 and 0.90 of recall. The main factor for this result was the selection of the most specific proprieties for each device/type. Thanks to this detail, it was possible to reach a smaller number of False Positives and thus achieve higher values.[2]

B. IoT EYE

Due to the large number of uncontrolled dynamic connections to the intranet can result in the spread of Trojan, viruses, information leakage, etc. The complex structure of large network makes it very difficult to detect all the dynamic IoT devices accurately. In [13], a model is proposed called IoT Eye which is used to monitor these dynamic connections through high speed scanning and

fingerprint matching technique. The architecture comprises of two parts (i) high speed TCP scanning of IP segments during which open ports of each host are detected. (ii) equipment fingerprint identification. The system learns the communication protocol of each host and their services based upon their port status. A multi pattern algorithm is proposed called PI-AC which can also quickly modify the structure of automata.

The purpose of IoT Eye is to quickly discover IoT devices, discover vulnerabilities and reduce cyber threats in the organizations' intranet.

IoT Eye uses a TCP SYN probe for implementation. TCP is a connection-oriented protocol which uses three packet handshake process for establishment of TCP connection as follows:

1) The client sends SYN packet to server and waits for response.
2) The server sends a TCP packet with ACK=1 and SYN=1 to the client.
3) The client responds with ACK package to the server again.

The scan efficiency of TCP connect scan is too low for IoT Eye. High speed TCP scanner need not establish a complete TCP connection.

The proposed system uses Zero Copy technique to optimize local packet processing. The goal of zero copy technique is to avoid unnecessary system calls and context switches. By combining TCP SYN scan and Zero Copy stack a fast TCP port scanner is achieved.

The proposed system uses a high-performance pattern matching algorithm. PI-AC algorithm has an index array which consists of first character of strings in the pattern set which it extracts and stores in character array. In PI-AC automata, failure pointer is replaced with default transfer pointer. The pattern words are divided into several subsets according to their first character and processed by one worker thread after which, the thread will calculate first character ordinal and check whether the associated array element is null. If it's null, the pending pattern word is the first one started which needs to be processed then the associated array item will be written with the first-character and the sub tree will be constructed from the rest characters of the pattern word. Itf it is not null then it means that another pattern word led by the same character has been already added so the rest characters would be inserted into the sub-tree.

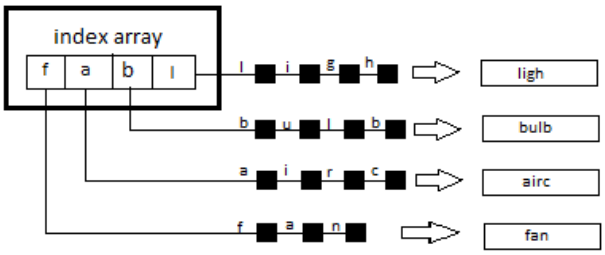

Figure 3 PI-AC automata of pattern set S.

To add a new pattern word into the existing PI-AC automata, check whether the first character of pending pattern is in the index array, if it is, then insert all associated characters into the subtree. If the first character is not present in index array then ordinal of the first character is calculated by the PI-AC algorithm and is inserted into the index array.

For delete operation, we need to locate the first character of the pattern word in index array, then the nodes associated with the pattern word are wiped out.

Experiments carried on TCP SYN probe in two applications: (i) Single port scanning and (ii) multi-port scanning, shows that with increase of number of detection ports, scanning period grows linearly. It also shows that the scanner of IoT Eye is able to complete the host and port probing in a short time.

Matching experiment is performed on AC algorithm, AC_BNFA algorithm and PI-AC algorithm on the same platform. The results of the test show that the matching performance of PI-AC is better than classic AC and AC_BFNA.

he third experiment is performed to see the efficiency of automata construction of classic AC, AC_BFNA, PI-AC. The experiment is performed on the same platform and the results show that the PI-AC's automata construction efficiency is improved by adopting parallel optimization method.

III.CONCLUSION

Due to the growth of IoT, there are many challenges like decentralization of the existent implementations and platforms. There is a need for effectively integrating the devices so that they can provide better services. In this paper we discussed the various techniques of device discovery that can contribute to automatic integration. This automatic integration process relies on an effective way to identify the device that is exchanging communication data.

The IoT devices were identified using only the properties of the device found in communicated data. In this paper we summarized the methods used to identify devices. The necessary communication data was gathered by listening to network traffic. This data was then used to generate a device information database that stores the properties of each device. Using this database, devices that are communicating inside a network were identified correctly. The techniques like the Levenshtein Distance algorithm, TF-IDF tables, Synonyms Match and Multi Property matching were discussed. The results for each of the tests mentioned above were also discussed. A multi-property matching technique over a database generated with the TF-IDF algorithm final results were good (precision of 0.91) and we can verify that, with correctly valued TF-IDF tables (type specific proprieties), it is possible to identify the device type with good precision.

IoT Eye is another technique that automatically discovers IoT devices in real time therefore reducing and controlling the risk associated with them.

REFERENCES

[1] http://blogs.flexerasoftware.com/elo/2014/12/big-numbers-50-billionconnected-devices-by-2020.html
[2] Pedro R J Pego and Luis Nunes "Automatic Discovery and Classifications of IoT Devices"
[3] C. Systems, "Internet of Things (IoT) in Real World," 2012.
[4] A. Alliance, "AllJoyn Proximal Connectivity Platform," 2015. [Online]. Available: https://developer.qualcomm. com/software/alljoyn
[5] S.-C. Arseni, S. Halunga, O. Fratu, A. Vulpe, and G. Suciu, "Analysis of the security solutions implemented in current internet of things platforms," in Grid, Cloud & High-Performance Computing in Science (ROLCG), 2015 Conference. IEEE, 2015, pp. 1–4.
[6] L. Foundation, "Iotivity architecture overview," 2016. [Online]. Available: https://www.iotivity.org/ documentation/architecture-overview
[7] E. Nimmermark and A. Larsson, "Comparison of iot frameworks for the smart home," 2016.
[8] H. Hexmoor, "Compelling use cases for the internet of things," in Proceedings on the International Conference on Internet Computing (ICOMP). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016, p. 20.
[9] Bitbucket, "Bitbucket thesis app fork," 2016.
[10] J. Ramos, "Using tf-idf to determine word relevance in document queries," in Proceedings of the first instructional conference on machine learning, 2003.
[11] K. Beijering, C. Gooskens, and W. Heeringa, "Predicting intelligibility and perceived linguistic distances by means of the levenshtein algorithm," Linguistics in the Netherlands, vol. 15, pp. 13–24, 2008
[12] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of kfold cross validation in prediction error estimation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 3, pp. 569– 575,2010.
[13] Jie Shen, Ying Li, Bo Li, Hanteng Chen, Jianxin Lee "IoT Eye: an efficient system for dynamic IoT devices auto-discovery on organizational level" 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing