Internet of Smart Things – IoST

Using Blockchain and CLIPS to make Things Autonomous

Mayra Samaniego
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
mayra.samaniego@usask.ca

Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
deters@cs.usask.ca

***Abstract*— Current networking integrates common "Things" to the Web, creating the Internet of Things (IoT). The considerable number of heterogeneous Things that can be part of an IoT network demands an efficient management of resources. With the advent of Fog computing, some IoT management tasks can be distributed toward the edge of the constrained networks, closer to physical devices. Blockchain protocols hosted on Fog networks can handle IoT management tasks such as communication, storage, and authentication. This research goes beyond the current definition of Things and presents the Internet of "Smart Things." Smart Things are provisioned with Artificial Intelligence (AI) features based on CLIPS programming language to become self-inferenceable and self-monitorable. This work uses the permission-based blockchain protocol Multichain to communicate many Smart Things by reading and writing blocks of information. This paper evaluates Smart Things deployed on Edison Arduino boards. Also, this work evaluates Multichain hosted on a Fog network.**

Keywords- IoT; Management; Blockchain; Multichain; Smart Things; Autonomy; Self-inferencing; Self-monitoring; Fog; Edge.

I. Introduction

The Internet of Things adds sensing and actuating capabilities to common Things to capture data from the real world [1]. Initially, the Things did not have the computational resources to process/analyze data before sending it to the Cloud. Typical IoT systems follow a Cloud-centric approach [2], which considers the Things as static collectors of data from the environment. As shown in Figure 1, Cloud-centric systems (e.g. [3][4]) consist of three primary layers:

1. Things
2. Services
3. Applications

The Things layer corresponds to the constrained devices (e.g. sensor networks). The Applications hosted in the Cloud access the data sensed in the Things layer through virtualized components hosted in the Services layer. Sensing and Actuation as a Service (SAaaS) [5], Data as a Service (DaaS) [6], and Sensor Event as a Service (SEaaS) [7] are examples of Cloud-centric systems in IoT. The robustness and flexibility of the Cloud make the data processing efficient and reliable [8]; however, the time data streams take to reach the Cloud may affect the accurate decision-making over that data [9].

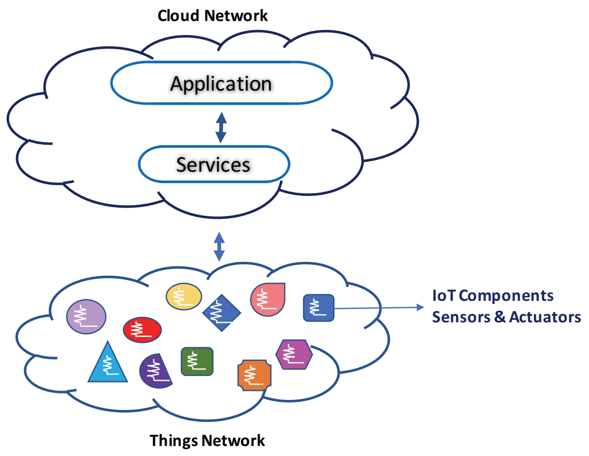


Figure 1. The three layers of Cloud-centric IoT systems [10].

The obvious drawback of the Cloud-centric design is a static sensor and data stream-centric IoT. Enabling more advanced IoT systems requires the possibility of processing data in the Things layer. This, in turn, introduces the need to include concepts for autonomy support in IoT.

The enhancement of the computing capabilities of IoT devices makes it possible to perform autonomous tasks. This work implements autonomic computing at the Things layer by creating Smart Things with self-inferencing and self-monitoring capabilities.

In the IoT space, it is mandatory to have low latency when managing the geographically distributed Things. According to a study by Cortés et al. [11] about IoT in the health field, the centralized Cloud storage cannot handle the velocity of the data flow generated by sensing devices in real-time. Using Fog computing [9], some IoT management tasks can be distributed toward the edge of the IoT networks improving efficiency and decreasing latency. Many studies have proposed virtual solutions hosted on Fog layers to manage constrained networks (e.g. virtual sensors [12] and gateways [13]). However, these approaches tend to focus on virtualizing individual components avoiding real-time communication among Things. Unlike Cloud-centric systems, Fog systems enable direct and real-time communication with the Things layer. This introduces the need to include a decentralized mechanism that manages the communication among Things. This paper proposes a blockchain cluster configured in a Fog network to support the communication management of Smart Things in real time. A Fog layer with blockchain introduces the following management capabilities:

- Decentralized communication management
- Low latency communication
- Real-time communication
- Time-effective event management

This paper is organized as follows. Section II discusses autonomy in IoT. Section III introduces expert systems in IoT. Blockchain protocols are discussed in Section IV. Section V explains the Smart Things architecture. Section VI presents the experiments and evaluations. Finally, this paper concludes with a summary and future work described in Section VII.

II. AUTONOMY & IOT

Autonomic Computing refers to architectures in which computers can monitor and manage themselves [14]. Autonomic architectures rely on the capacity of systems to perform self-management. Thus, systems can react to unknown events without requiring any human participation. According to Kephart and Chess [15], autonomic systems should fulfill the following "self-management" principles:

1. Self-configuration and re-configuration
2. Self-optimization
3. Self-healing
4. Self-protection

Table I presents the differences in management between current computing and autonomic computing.

Autonomic systems have been studied in the context of traditional Internet networks. IBM has presented a hierarchical autonomic computing framework [15] in which high-level components act as autonomic managers of either individual or groups of low-level components. IBM's framework has two main parts; first, the managed resource, which represents a common hardware component like a CPU, database, or any other component or group of components; second, the autonomic manager, which is an entity that monitors the managed resource and analyzes its surroundings to execute proper management tasks.

In the IoT context, autonomic systems are required to deal with the characteristics of constrained networks [16][17]:

- Heterogeneous platforms
- Large set of devices
- Limited computational capabilities
- Limited energy consumption
- Geographical distribution
- Real-time operations

These characteristics make the enabling of autonomy in IoT a challenge. Furthermore, the autonomy concept in IoT can be focused on supporting specific features depending on the context in which resources are working.

IBM's autonomic computing framework described above has been the basis of many IoT studies (e.g. [18][19][20]). Overall, in an IoT network, the managed resource is a sensing/actuating device, and the autonomic manager is a middleware that monitors the device and performs actions based on its current state.

The Autonomous Decentralized Peer-to-Peer Telemetry (ADEPT) Proof of Concept (PoC) [21] is another example of autonomic IoT systems. Unlike IBM's Autonomic Framework, ADEPT has a hybrid and decentralized architecture. ADEPT integrates Telehash [22] for peer-to-peer messaging, BitTorrent [23] for distributed file sharing, and Ethereum [24] blockchain for autonomous device coordination functions such as storing the configuration of devices and authentication.

TABLE I. DIFFERENCES IN SELF-MANAGEMENT BETWEEN CURRENT COMPUTING AND AUTONOMIC COMPUTING [15].

Concept	**Current Computing**	**Autonomic Computing**
Self-configuration	Corporate data centers have multiple vendors and platforms. Installing, configuring, and integrating systems is time-consuming and error proof	Automated configuration of components and systems follows high-level policies. Rest of system adjusts automatically and seamlessly
Self-optimization	Systems have hundreds of manually set, nonlinear tuning parameters, and their number increases with each release	Components and systems continually seek opportunities to improve their performance and efficiency
Self-healing	Problem determination in large, complex systems can take a team of programmers weeks	System automatically detects, diagnoses, and repairs localized software and hardware problems
Self-protection	Detection and recovery from attacks and cascading failures is manual	System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent systemwide failures

III. EXPERT SYSTEMS & IOT

An Expert System emulates the decision-making behavior of a human being [25]. The C' Language Integrated Production System (CLIPS) [26] is a public rule-based programming language for developing expert systems. CLIPS was developed by NASA. Because CLIPS is written in C, it can be installed on many platforms. CLIPS has been the foundation of many hybrid applications (e. g., [27][28]).

According to Stankovic [29], the development of inference techniques is a challenge to creating knowledge and intelligent systems in IoT. Also, according to Perera et al. [30], intelligence should be a major characteristic of IoT systems. More research is necessary to evaluate the potential benefits of Expert systems in IoT.

IV. Blockchain & IoT

Introduced by Bitcoin in 2009 [31], blockchain represents the public ledger that stores Bitcoin transactions in the form of blocks. Blocks are connected through a hash value. A blockchain is a peer-to-peer network that allows the execution of direct transactions without any central verification authority. Transactions are validated by a consensus mechanism in which participants must invest computation to show trustworthiness.

Public blockchain protocols give open access to transactions and blocks [31]. Any user on the Internet can interact with a public blockchain. However, a public blockchain protocol does not trust any participant; participants must validate their transactions by a proof-of-work mechanism. The proof-of-work process involves considerable time and computation.

Private blockchain protocols only trust a set of registered participants [32]. Even though registered participants do not have to do proof-of-work, this participation has to be managed by a consensus mechanism, such as Practical Byzantine Fault Tolerance (PBFT) [33], Round Robin (RR) scheduling [34], or other consensus algorithms.

Including blockchain protocols in the IoT space creates new management possibilities. In this work, a permission-based blockchain protocol (private blockchain protocol) is used to manage the real-time communication among Smart Things. In this study, the use of a blockchain protocol enables an efficient, secure and economic distributed communication architecture for Smart Things in a Fog network.

V. Smart Things Architecture

This section introduces the architectural design of Smart Things. Following the ADEPT PoC idea of building hybrid and decentralized autonomic solutions, this work seeks to make Things autonomous by creating a hybrid architecture. The hybrid architecture proposed by this research has two main components, Smart Things and blockchain protocol.

1. *Smart Things*

A Smart Thing is a software artifact, which can analyze its current state, infer knowledge and monitor possible changes.

Please see Samaniego et al. [35] *for additional information about the definition and implementation of Software-defined IoT components.*

The main goal of a Smart Thing is to keep the monitoring and decision-making in the Things layer by developing artificial intelligence (AI) features directly onto IoT devices. Figure 2 shows the design of a Smart Thing. A Smart Thing has three main components:

1. A Reader
2. A Self-inferencing resource
3. A Self-monitoring resource

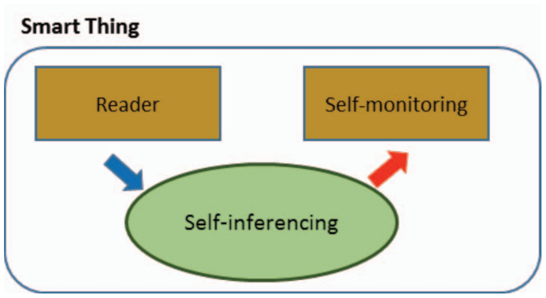


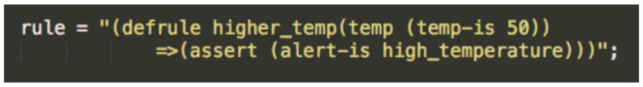
Figure 2. The design of a Smart Thing.

A. *Reader*

The Reader senses data from the environment. This work uses GOBOT [36] to face the physical sensors. The Reader sends every collected data to the Self-inferencing resource.

B. *Self-inferencing Resource*

The Self-inferencing resource integrates a rule-based expert system built on top of CLIPS. The Self-inferencing resource performs AI reasoning based on pre-configured rules. A set of initial rules and facts are asserted when the Smart Thing starts working. Figure 3 shows an example of a basic pre-configured rule. This rule evaluates the received temperature values and assets a new Fact in case the evaluated condition is true.



```
rule = "(defrule higher_temp(temp (temp-is 50))
        =>(assert (alert-is high_temperature)))";
```

Figure 3. An example of a basic rule defined in CLIPS.

The received data in the Self-inferencing resource is transformed into facts and asserted into the Facts List of the system. The inferencing resource analyses the facts and executes actions. Finally, the results of the analysis are sent to the Self-monitoring resource.

C. *Self-monitoring Resource*

The Self-monitoring resource receives the results from the Self-inferencing resource and evaluates accepted standards. The Self-monitoring resource interacts with the blockchain and when necessary sends data to the blockchain to be distributed onto the Things layer. The rules and the inferencing process in the expert system determine whether there is a need to send data to the Multichain cluster.

All the components of the Smart Things are programmed using Go language [37]. The three components communicate via RESTful micro services. All transmitted data is formatted using JSON notation. The integration of an expert system in the Things layer presents the following benefits:

- Things can analyze data without forwarding it to any gateway
- Things can self-monitor and infer knowledge locally
- Things can act over data in a time-effective manner

2. *Permission-based blockchain protocol Multichain*

This work uses the permission-based blockchain protocol Multichain [38] to manage the communication among Smart Things. Multichain is a private blockchain protocol that manages the access to the blocks using a list of registered participants. Only participants who have been previously registered have access to read and write blocks in the chain. The consensus method that Multichain implements to approve transactions is the Round Robin (RR) scheduling algorithm. The RR algorithm states that every block must have a signature from the participant who intends to create it [39]. The participant who created the block must wait a certain time to send a new request to create a new block.

Figure 4 shows the architecture of Smart Things working with the Multichain cluster. The Multichain cluster is hosted in a Fog network, closer to the Things layer. The multichain cluster interacts with the self-monitoring resource of the Smart Thing. Multichain receives the data, stores it, and distributes it to all the nodes in the cluster so that the nodes can communicate the results of the decision-making process to the Smart Things in real time.

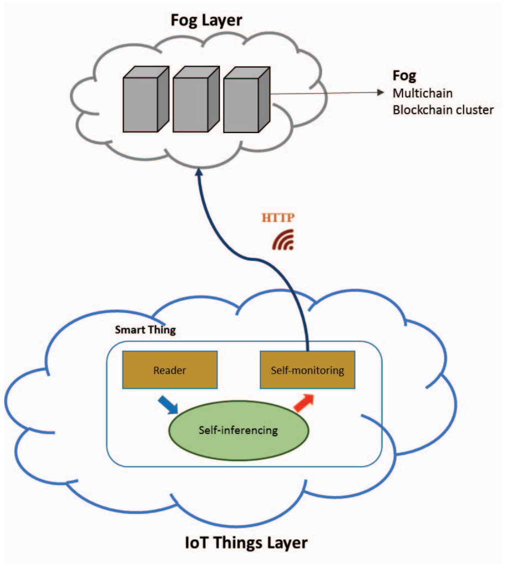


Figure 4. The architecture of a Smart Thing including Multichain.

VI. Experiments & Evaluations

This section evaluates the Smart Things and Multichain blockchain.

A. *Evaluation of the Smart Things*

To evaluate the performance of the Smart Things an Edison Arduino board (Figure 5) has been used. Table II details the characteristics of the Edison board.

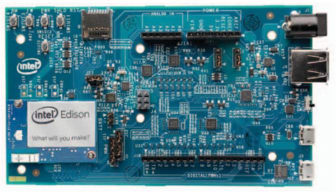


Figure 5. Edison Arduino board [40].

TABLE II. Specification of Edison Board

Edison Board	
Operating System	Linux Yocto
CPU	500 MHz dual-core, dual threaded Intel Atom and a 100 MHz 32-bit Intel Quark microcontroller
RAM	4GB LPDDR2 SDRAM

The Edison board runs the three components of the Smart Thing; the Reader, the Self-inferencing resource, and the Self-monitoring resource. 1000 temperature values are read from the environment and sent to the expert system.

The payload of each request is encrypted using a synchronous AES encryption method with a key of 16 bytes. The total payload size is 36 bytes. The expert system decrypts the data, transforms it into a fact, asserts the fact in the rule engine, makes inferencing processes and analysis, and sends the results to the Self-monitoring resource.

Different delay intervals have been added to the experiment to test the performance of the Smart Things under different request levels.

Figures 6 to 9 show the performance of the Smart Thing. The X axis in the graphs represents the requests sent to the expert system. The Y axis represents the response times from the expert system measured in milliseconds (ms).

Overall, the graphs show some picks. Because there is a minimal difference between the response times, those picks can be attributed to the noise of the network (university wireless connection), memory allocation or background processes of the device. That is why some response values increase up to almost 3.7 ms.

Adding delay intervals to the emissions of data does not affect the performance of the Smart Things. The average response time of the Smart Thing is 1.7 ms. In all the experiments, the response times are greater than 1.2 ms. This means that no matter the concurrency of requests, the response time will not be less than 1.2 ms.

It is very important to note that the time for encrypting and decrypting the transmitted data is included in the experiments, which evidences a good performance of the design of Smart Things presented in this study.

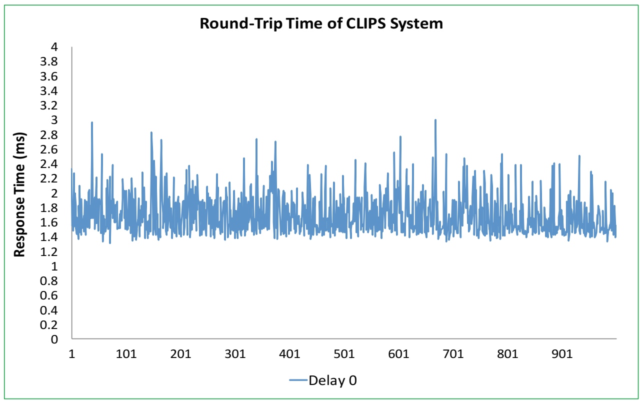


Figure 6. Evaluation of the Smart Thing. 0 delay.

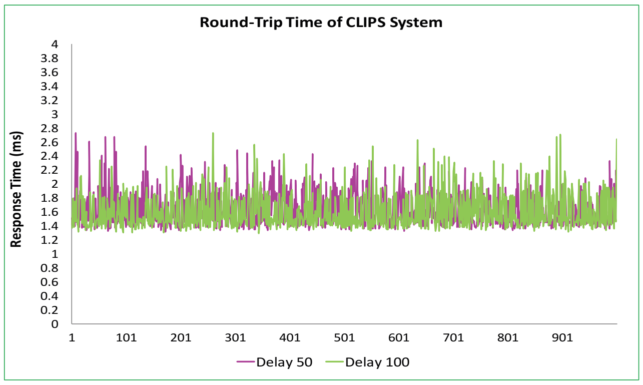


Figure 7. Evaluation of the Smart Thing. 50 ms and 100 ms delay.

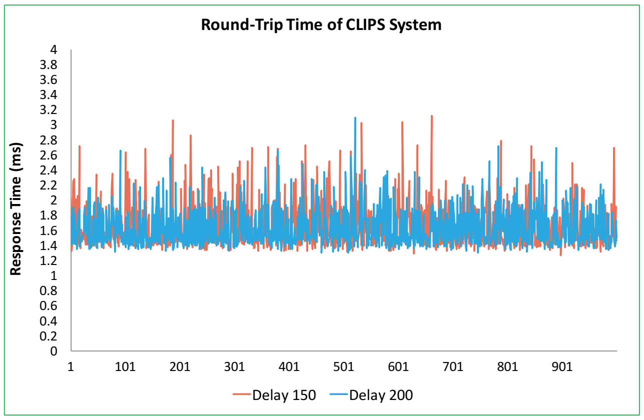


Figure 8. Evaluation of the Smart Thing 150 ms and 200 ms delay.

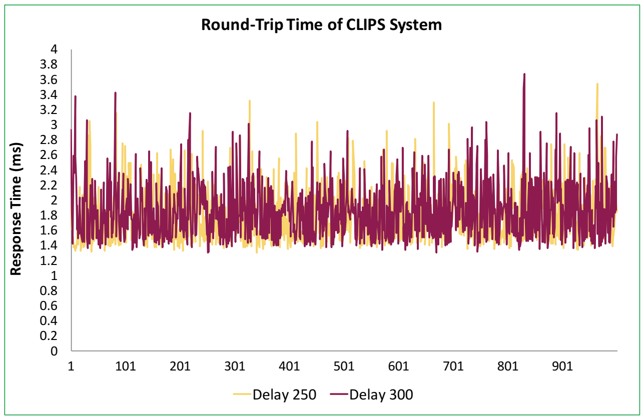


Figure 9. Evaluation of the Smart Thing. 250 ms and 300 ms delay.

B. *Evaluation of Multichain*

The performance of a Multichain cluster hosted in a Fog network is evaluated. Figure 10 shows the setup for these experiments. A cluster with three nodes has been configured. The characteristics of the Multichain nodes are described in Table III.

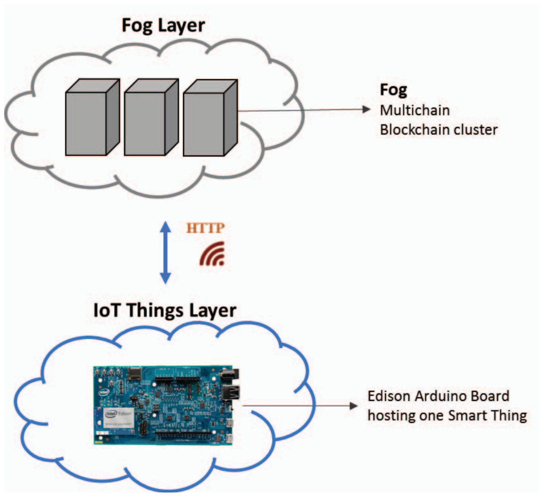


Figure 10. Setup of the evaluation of the Multichain cluster.

TABLE III. SPECIFICATION OF THE MULTICHAIN BLOCKCHAIN NODES

Hardware	Details
Operating System	Linux Debian 8.5 (Jessie)
CPU	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
RAM	14 GB

The Edison board hosts a Smart Thing, which after doing the inferencing and analysis process, sends 1000 results to the blockchain. The payload of each request includes the participant identification information. The payloads are encrypted using a synchronous AES encryption method with a key of 16 bytes. The total payload size is 148 bytes. The time for encrypting and decrypting the payload is included in the experiments.

Figures 11 to 14 show the performance of the Multichain cluster. The X axis represents the number of requests sent to Multichain. The Y axis represents the response times of the Multichain cluster in milliseconds. This experiment also includes different delay intervals to test the performance of the Multichain cluster under different request levels.

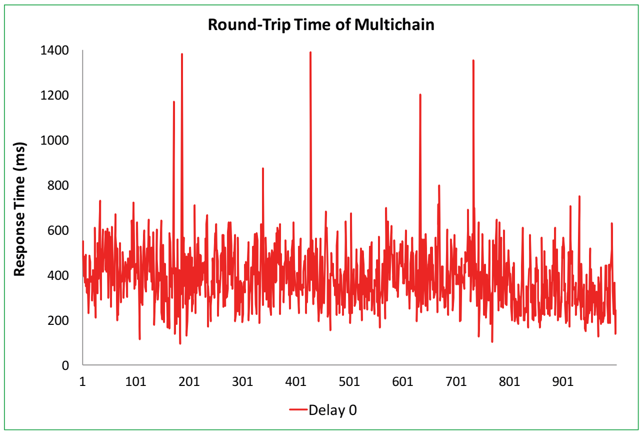


Figure 11. Evaluation of Multichain. 0 delay.

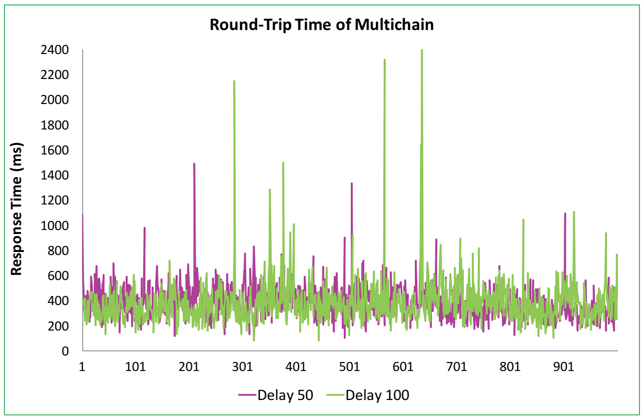


Figure 12. Evaluation of Multichain. 50 ms and 100 ms delay.

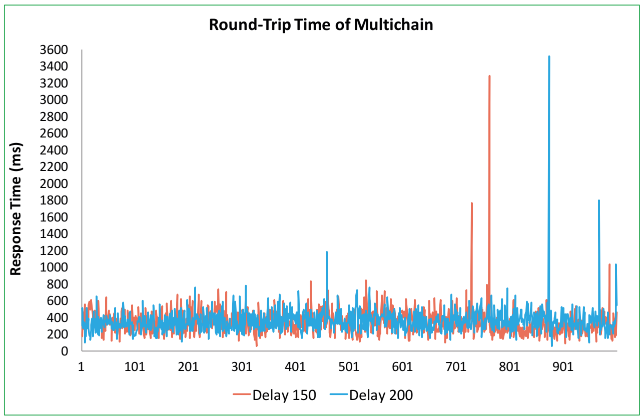


Figure 13. Evaluation of Multichain. 150 ms and 200 ms delay.

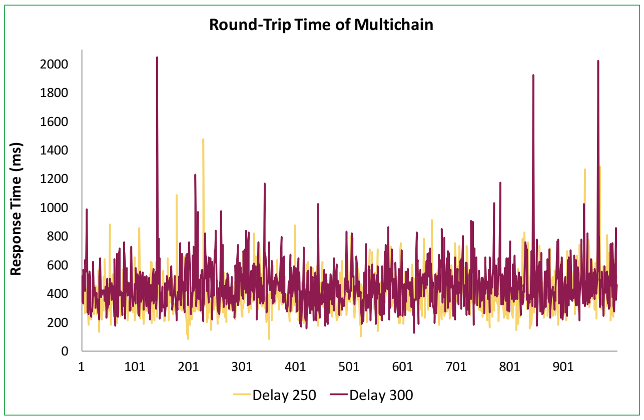


Figure 14. Evaluation of Multichain. 250 ms and 300 ms delay.

The graphs show that the average response time of Multichain is 389 ms. The delay intervals introduced in the experiments do not alter the performance of the Multichain cluster. In the series with no delay intervals, the average response time is 391 ms. In the series with a delay interval of 50 ms the average response time is 387 ms. In the series with a delay interval of 100 ms the average response time is 383 ms. In the series with a delay interval of 150 ms the average response time is 343 ms. In the series with a delay interval of 200 ms the average response time is 358 ms. In the series with a delay interval of 250 ms the average response time is 407 ms. Finally, In the series with a delay interval of 300 ms the average response time is 457 ms.

The previous graphs evidence very high picks in all delay intervals. Some of the observed picks are the result of the timeout responses received from the Multichain cluster. The other picks can be attributed to the noise in the network.

These experiments show that Blockchain can store IoT data and distribute it among all the nodes of the chain.

Figure 15 shows the percentage of successful transactions processed in the blockchain cluster. The average throughput of the Multichain cluster is 98.47%. Overall, the average of unprocessed requests is 98 requests in all the experiments.

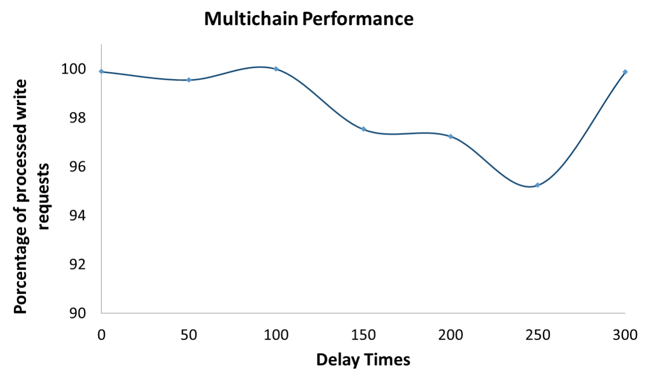


Figure 15. Processed transactions in Multichain.

VII. Conclusions and Future Work

This paper presented Smart Things as a means of enabling autonomy in IoT networks. Smart Things are formed by three main components, the reader, the self-inferencing resource, and the self-monitoring resource. By integrating a CLIPS-based expert system in the self-inferencing resource, the IoT focus is shifted away from centralized architectures to individual Smart Things capable of doing self-inferencing and self-monitoring in real time, which means that Things become autonomous. Smart Things communicate with other resources via permission-based blockchain protocol Multichain. In general, the communication cost via Multichain was low.

The entire development of Smart Things and the integration of the CLIPS system has been done using RESTful micro services programmed in Go language.

The results of the evaluations showed that hybrid solutions are a good option to enable autonomous features in IoT networks. The resources of the Smart Things demonstrated a satisfactory performance analyzing and inferring knowledge from every data received.

Overall, this research made the following contributions to IoT systems:

- Designed and developed Smart Things, which are software artifacts that can perform self-inferencing and self-monitoring tasks in real time.
- Designed and developed an expert system on top of CLIPS to perform data analysis and self-inferencing.
- Designed and implemented a decentralized Fog architecture that integrates permission-based blockchain protocols to communicate Smart Things in real time and distribute management tasks toward the edge of the Things layer.

Future work will focus on evaluating Smart Things on different boards.

References

[1] K. Ashton, "That 'Internet of Things' Thing - RFID Journal," *RFiD J.*, vol. 22, no. 7, pp. 97–114, 2009.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] C. Doukas and I. Maglogiannis, "Bringing IoT and cloud computing towards pervasive healthcare," in *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*, 2012, pp. 922–926.

[4] A. R. Biswas and R. Giaffreda, "IoT and Cloud Convergence: Opportunities and Challenges," *2014 IEEE World Forum Internet Things*, pp. 375–376, 2014.

[5] S. Distefano, G. Merlino, and A. Puliafito, "Sensing and actuation as a service: A new development for clouds," *Proc. - IEEE 11th Int. Symp. Netw. Comput. Appl. NCA 2012*, vol. 1, pp. 272–275, 2012.

[6] O. Terzo, P. Ruiu, E. Bucci, and F. Xhafa, "Data as a Service (DaaS) for sharing and processing of large data collections in the cloud," *Proc. - 2013 7th Int. Conf. Complex, Intelligent, Softw. Intensive Syst. CISIS 2013*, pp. 475–480, 2013.

[7] S. Alam, M. M. R. Chowdhury, and J. Noll, "SenaaS: An event-driven sensor virtualization approach for internet of things cloud," *2010 IEEE Int. Conf. Networked Embed. Syst. Enterp. Appl. NESEA 2010*, pp. 1–6, 2010.

[8] P. Parwekar, "From Internet of Things towards cloud of things," *2011 2nd Int. Conf. Comput. Commun. Technol. ICCCT-2011*, pp. 329–333, 2011.

[9] Cisco Systems, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," 2015.

[10] M. Samaniego, "Virtual Resources & Internet of Things," University of Saskatchewan, 2016.

[11] R. Cortés, X. Bonnaire, O. Marin, and P. Sens, "Stream Processing of Healthcare Sensor Data: Studying User Traces to Identify Challenges from a Big Data Perspective," *Procedia Comput. Sci.*, vol. 52, pp. 1004–1009, 2015.

[12] S. Alam, M. M. R. Chowdhury, and J. Noll, "SenaaS: An event-driven sensor virtualization approach for internet of things cloud," in *Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on.*, 2010, pp. 1–6.

[13] M. Aazam and E. N. Huh, "Fog computing and smart gateway based communication for cloud of things," *Proc. - 2014 Int. Conf. Futur. Internet Things Cloud, FiCloud 2014*, pp. 464–470, 2014.

[14] P. Horn, "Autonomic computing: IBM's Perspective on the State of Information Technology," 2001.

[15] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing," *IEEE Comput.*, vol. 36, no. 1, pp. 41–50, 2003.

[16] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[17] M. Wu, T. Lu, F.-Y. Ling, L. Sun, and H.-Y. Du, "Research on the architecture of Internet of Things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, 2010, vol. 5, pp. V5-484-V5-487.

[18] J. Gombak, "Introducing Autonomy in Internet of Things," in *Recent Advances in Computer Science*, 2015, pp. 215–221.

[19] M. A. Rajan, P. Balamuralidhar, K. P. Chethan, and M. Swarnahpriyaah, “A Self-Reconfigurable Sensor Network Management System for Internet of Things Paradigm,” *2011 Int. Conf. Devices Commun.*, pp. 1–5, 2011.

[20] G. Pujolle, “An Autonomic-oriented Architecture for the Internet of Things,” *Mod. Comput. 2006. JVA’06. IEEE John Vincent Atanasoff 2006 Int. Symp.*, pp. 163–168, 2006.

[21] V. Pureswaran, S. Panikkar, S. Nair, and P. Brody, “Empowering the Edge: Practical Insights on a Decentralized Internet of Things,” 2015. [Online]. Available: https://www-935.ibm.com/services/multimedia/GBE03662USEN.pdf. [Accessed: 22-Nov-2016].

[22] “https://github.com/telehash/telehash.github.io.” .

[23] “BitTorrent.” [Online]. Available: http://www.bittorrent.com/. [Accessed: 18-Mar-2017].

[24] “Ethereum Project.” [Online]. Available: https://www.ethereum.org/. [Accessed: 18-Mar-2017].

[25] Jackson and P., “Introduction to expert systems.” Addison-Wesley Pub. Co.,Reading, MA, 1986.

[26] “About CLIPS | CLIPS.” [Online]. Available: http://www.clipsrules.net/?q=AboutCLIPS. [Accessed: 16-Mar-2017].

[27] R. Sárfi and A. Solo, “Development of a hybrid knowledge-based system for multiobjective optimization of power distribution system operations,” *Proc. Natl.*, 2005.

[28] J. Jarmulak, E. J. H. Kerckhoffs, and P. P. Van Veen, “Hybrid knowledge based system for automatic classification of B-scan images from ultrasonic rail inspection,” in *AAAI/IAAI*, 1998, pp. 1121–1126.

[29] J. A. Stankovic, “Research directions for the internet of things,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, 2014.

[30] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the internet of things: A survey,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[31] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” p. 9, 2008.

[32] K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[33] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *Proc. Symp. Oper. Syst. Des. Implement.*, no. February, pp. 1–14, 1999.

[34] W. Assumptions, “Scheduling: Introduction.” [Online]. Available: http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched.pdf. [Accessed: 08-Oct-2016].

[35] M. Samaniego and R. Deters, “Using Blockchain to push Software-Defined IoT Components onto Edge Hosts,” in *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, 2016, p. 58.

[36] “Gobot - Golang framework for robotics, drones, and the Internet of Things (IoT).” [Online]. Available: https://gobot.io/. [Accessed: 01-May-2017].

[37] “The Go Programming Language.” [Online]. Available: https://golang.org/. [Accessed: 19-Mar-2017].

[38] “MultiChain | Open source private blockchain platform.” [Online]. Available: http://www.multichain.com/. [Accessed: 19-Mar-2017].

[39] A. Lewis, “In a nutshell: MultiChain (Epicenter Bitcoin interview – Nov 2015) | Bits on blocks,” 2016. [Online]. Available: https://bitsonblocks.net/2016/03/07/in-a-nutshell-multichain-epicenter-bitcoin-interview-nov-2015/. [Accessed: 27-Nov-2016].

[40] “The Intel® Edison Module | IoT | Intel® Software.” [Online]. Available: https://software.intel.com/en-us/iot/hardware/edison. [Accessed: 19-Mar-2017].