Opportunistic Mobile IoT with Blockchain based Collaboration

Ravishankar Chamarajnagar
VMware Inc.,Atlanta,GA 30338
Email: ravshankcv@gmail.com

Ashwin Ashok
Georgia State University,Atlanta,GA 30303
Email: aashok@gsu.edu

***Abstract*—The proliferation of Internet-of-Things (IoT) devices has opened up plethora of opportunities for smart networking and connected applications. The large distribution of IoT devices within a finite geographical area and the pervasiveness of wireless networking presents an opportunity for such devices to collaborate. This paper proposes the idea of opportunistic collaboration among mobile IoT devices to share their services and excess computing resources. Opportunistic collaboration among devices over wireless networking requires proper coordination and agreements among the devices in a purely distributed manner. To facilitate the distributive collaboration, we propose a decentralized architecture design using blockchain technology. Through experimental evaluation of a prototype collaborative mobile–IoT system involving RaspberryPis and a Dell IoT edge gateway, we show that our proposed distributed collaborative approach is feasible and comparable to a non–collaborative edge–computing based approach from a latency perspective.**

I. Introduction

The pervasiveness of the Internet–of–Things (IoT) is leading the way to a world of smart systems, applications and services powered by mobile devices. Cloud based mobile IoT is paving the way for a large number of emerging applications and computing platforms. One of the key challenges in realizing mobile IoT systems using current day cloud solutions is the heavy dependence on centralized cloud infrastructures. Such centralized solutions are necessary to address the large scale of IoT, however, the performance of such systems degrade with the wireless network connectivity between the device and the cloud. Also, with the growing need for data creation, access and storage, the network backbones will be inundated with data if they try to handle every IoT application and service request using centralized data centers. This dependency on centralized architecture can lead to performance degradation and eventually unreliability in mobile IoT systems.

Even as we encounter computational inefficiencies and network limits with the cloud based architecture in dealing with the vast internet of things, the fact that mobile IoT machines are getting smarter and more powerful is encouraging. IoT devices are getting packed with resources such as sensors, computing, I/O, etc. However, these resources are being completely underutilized as most of the resources are used only for a certain duration of time for specific applications. The usage can vary with location and time. For example, a smartphone with quad–core processors has more computing power and storage than necessary for an average

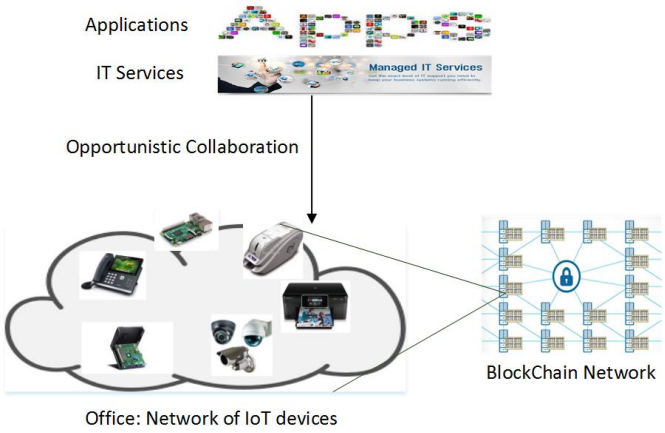

Fig. 1. Opportunistic Collaborative IoT using Blockchains

daily usage which mostly involves access to text and emails; a smart printer in an office is largely in sleep and usually handles jobs in aperiodic chunks; a smart TV is used more during the day in an office versus evenings at homes. Another manifestation of the excess and unused resources on a lot of these devices are the services available on the devices. For instance, the services on a printer and the video capabilities on a camera are unused for a large percentage of the time.

In essence, *there is excess resource and service capacity available in mobile IoT devices opening up a huge opportunity for sharing resources and services across IoT devices.* In this regard, we propose a system design for better utilization of resources and services in mobile IoT through opportunistic collaboration.

Towards opportunistic collaboration. As we look to address the above issues with network reliability, and resource and service underutilization, we realize the need for opportunistic collaboration of IoT devices to share resources and services when they are available. Opportunistic collaboration becomes necessary as application offloading to the cloud may not always be possible due to bandwidth and latency concerns. Also, the problem gets worse at scale. We seek to improve computational and network usage efficiency by mechanizing efficient ways to drive decisions and insights to execute applications and services closer to the devices, at the *edge*. Improving computation efficiency through collaboration

coupled with the opportunistic usage of unused capacity in devices and at the edge, motivates our proposed *opportunistic collaborative mobile IoT* design. The proposed solution uses the well–known paradigm of blockchains to facilitate the collaboration, as illustrated in Fig. 1.

Blockchain based collaboration. We design a novel collaborative mobile IoT architecture that lets mobile IoT devices come together in an adhoc manner, advertise their excess resource capacity and offered services using the blockchain framework. The collaborating devices are connected through a blockchain network that manages data dissemination in the network. The system uses *smart contracts* to advertise excess resources and network capacity, which are synchronized across a global network of blockchain nodes. These contracts are binding in nature and transparent to the network of nodes participating in the blockchain. Resources are made available to the seeking devices through containers (e.g. docker) so that applications can be executed in a sandboxed fashion and do not need rooting the device. Services are made available through peer–to–peer (P2P) communication protocols. Here, services can be those initiated by apps or broad IT infrastructure based services that cater to diverse applications and other dependent services.

This paper lays the foundation for a distributed computing solution for mobile IoT that lets one opportunistically use excess capacity on resources and services. In essence, our proposal is a distributed middleware design that leverages blockchain technology for smart resource/service discovery, coordination and management.

In summary, the key contributions of this paper are as follows:

(i) design of a distributed mechanism to allocate resources and execute services through smart blockchain contracts and application execution on sandboxed containers,

(ii) implementation of a prototype collaborative mobile IoT system on Raspberry Pis and a Dell IoT edge gateway, and

(iii) evaluation of network latency and benchmarking resource utilization under collaboration.

II. System Design

Blockchain Overview. Blockchain is a distributed database for an active list of records called *blocks*. Each block contains a timestamp and a link to a previous block making blockchain a chronological sequence of blocks with all transactions recorded up to that point in time. Just as transactions execute, they are mined, validated and added to blocks to be synchronized with the rest of the blockchain network. A blockchain serves as an open, transparent distributed ledger that can record transactions between parties efficiently and in a verifiable and permanent way. The ledger hosts transactions and smart contracts that can be triggered and executed automatically in software. A smart contract is a computerized transaction protocol that executes the terms of a contract. All of its code and data are housed in the blockchain and synchronized across the network.

A. System Overview

The core of this design is a distributed middleware that enables collaboration of mobile IoT elements using the capabilities of blockchain network. The blockchain framework helps to advertise, disseminate and make resources and services available to the network of IoT devices. Our design focuses on a localized blockchain network of nodes. We position that each network of IoT devices can form a small blockchain network that can expand organically as more devices are registered into this collaborative network. It also provides a virtual gateway to other blockchain networks making information on resource and service available on a global scale.

As shown in Figure 1, there are two key elements that comprise this distributed architecture: (a) applications and services that require resources, (b) distributed collection of IoT elements (devices and machines) with resources to share. Here, the IoT elements are termed as the *collaborating nodes*. Discovery of services and resources (computing and storage), consent for collaboration, invocation of services and allocation of resources are done using *smart contracts* that are deployed by each of the collaborating nodes in the blockchain. These contracts contain information of the list of available services and resources on each node. Through the blockchain framework these contracts are shared among all the nodes in the network.

A subset of the blockchain nodes, designated (through consensus) as *miners*, create and update smart contracts across the blockchain. For example, in an office room scenario of a collaboration network of a phone, printer, laptop, smart voice assistant and smart thermostat, the device with the highest computing power can be designated as the first miner. In essence, every device in the network can be designated as the miner. The downside is that the information dissemination time would increase as synchronization of each contract update will have to percolate through each miner node and arrive at a consensus. Depending on the number of miners the convergence can take few seconds to minutes. On the other hand, less number of miners will put a heavy load on a few miners. This lends itself to an interesting trade off between miner count and resource availability. We reserve addressing this trade off problem in more detail in our future work.

B. Workflow

The blockchain setup in a one–time process that involves all the mining and collaborating nodes. In this paper, we particularly focus on the collaborative execution process and not on the setup process. The setup process is analogous to setting up any IT infrastructure where registration of devices and information routing is checked through template test benches. In the office room example discussed above, the setup process is equivalent to registering the devices in the network of blockchain and designating the potential miners. In this section, we discuss the design details of the execution workflow of the process that happens once a blockchain is setup and that a resource requirement is found in the network. How the network of nodes collaborate among themselves

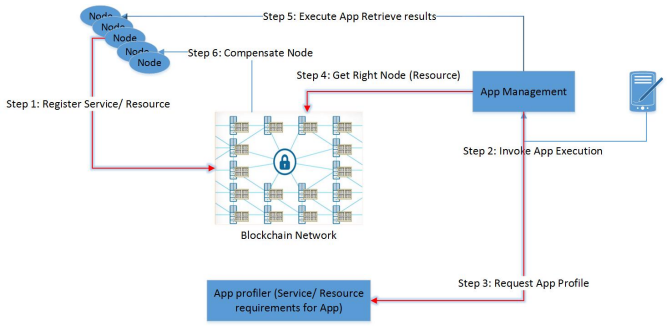

Fig. 2. System Workflow. The lines in red color highlight the design aspects we focus in this paper.

in a distributed manner to help each other achieve their tasks (applications and/or services) is the key notion of this workflow design. We discuss this workflow in more detail using the illustration in Fig. 2.

(Step 1) Service and Resource Registration. Any node with resources or services to offer registers the resources and/or services with the smart contract deployed in its blockchain network. The contracts contain resources and services available for rent/lease along with their cost and node identity in an encrypted manner. When the resources are advertised, they are recorded along with the identity of the node offering the same. This minimizes the resource discovery time as it accounts for locality in addition to keeping fragmentation to a minimum. Once a resource or service becomes available, it goes through a dissemination cycle, beginning with the blockchain contract getting invoked. As this request changes the state of the transaction, it gets synchronized with the rest of the blockchain that involves mining of the transaction, creation of a block and subsequently synchronization with the rest of the blockchain cluster. As the technologies move closer to autonomous operations, it is extremely important for resources and services to be discovered when needed. Having them on the blockchain makes them immediately available and easy to audit, two key aspects as we look into distributed autonomous systems.

(Step 2 & 3) Application Profiling. Once an application is kicked off, a typical application management system profiles the apps for its required and excess resources. It also acquires the list of services that might be invoked as part of the execution cycle of the apps and some of these services might be spread across the network. The apps and services to profile are invoked through an automated triggering process during the contract preparation.

(Step 4) Resource Allocation. Once distributed or queued for execution, the edge computing units check for the availability of the resources and/or services necessary to execute the apps and services. These edge units can be a single edge computer or a collection of nodes in the blockchain at the edge of the network. If the edge units do not have sufficient computing resources as requested in the contracts the request is turned to the blockchain nodes to execute the apps and services. Selection of the blockchain nodes to execute the apps and services comprises the resource allocation phase. Any traditional resource allocation mechanism that optimizes for latency based on available computing resources works in this case. However, the adhoc and distributed behavior of the blockchain approach enables to simplify the resource allocation process to a simple *resource matching* process by comparing contracts. This is possible because the contracts are agreed upon based on consensus and all collaborating nodes are informed of every other node's resource requirement and availability. This execution is analogous to pairing and agreement among the collaborating IoT devices in the office scenario to *help* each other with resources to execute the apps pertinent in the network. In the event, the nodes in the network do not have excess resources exposed for use, the nodes initiating the apps will fall back to the central edge or cloud computing framework.

(Step 5 & 6) Execution and Compensation. Once the node with the available resources is acquired the app is executed and the results gathered. App/service execution is done in sandboxed containers that get deployed in the blockchain nodes during the execution process. As we will describe later, our prototype system uses *docker* [1] containers, however, any type of sandboxing solution will work. In addition to satisfying the service and resource demands, it matches the rental price with the price the requesting entity is ready to pay for the resource or the service. Once either one is granted to the requesting entity, the node renting them is compensated for it with cryptocurrency (e.g. bitcoin [2], ethereum [3]).

Note the distinction between 'initiation'and 'invocation'; initiation of the app/service involves Resource discovery while invocations refer to execution which is instantaneous. For instance, deploying an HTTP service (initiation) versus accessing webpage using HTTP urls (invocation). Service invocation takes on a lot more importance in solutions that tie them together by cascading them and getting results to be fed as inputs to the next set of services. SingularityNet [4] has set an example of cascading the artificial intelligence services so they no longer as just speech or text AI services, but services that work in tandem to create a coherent solution. Such examples make a compelling case for the need of our decentralized collaborative platform.

III. Evaluation

We conducted experiments using a prototype implementation of our proposed system to evaluate the feasibility and benefits of the collaborative approach. We use latency and resource utilization efficiency as the metrics for our evaluation, where

- *App Execution Latency* is defined as the time it takes for all steps in the workflow including the latency for resource procurement, transfer and deployment, and initiation. This is equivalent to user response time or service delivery time.

- *Resource Utilization Efficiency* is defined as a combination of the dissemination efficiency to draw up a binding contract that makes the resource available on the blockchain network and the procurement efficiency to allocate resources to the requesting entity.

A. Experiment setup and methodology

Nodes	RAM (GB)	Storage (GB)	CPU
Raspberry Pis	1GB LPDDR2	32GB microSD	4× ARM 1.2GHz
Dell 5100 GW	2GB DDR3L-1067MHz	32GB SSD	Intel E3825 1.33GHz

Fig. 3. IoT nodes specifications.

We set up a blockchain network with 4 nodes forming a private network which are set to mine transactions and blocks, each on a single thread. The setup comprises 3 Raspberry Pi3 and a Dell 5100 IoT gateway, all of which have resources to spare (x RAM, y CPU cycles, z storage) and services available that are advertised on the blockchain as part of a contract with appropriate usage costs. Any app that wants to use these resources and/or services will need to request and acquire a handle to them and compensate the offering node. The nodes each have (pseudo) accounts set up to receive compensation for mining, resources leased and services invoked through pseudo cryptocurrency. The table in Fig. 3 outlines the configuration of the Raspberry Pis and the Dell gateway. Four containerized apps on Docker [1] with different data and storage requirements and execution priorities are used as sandboxes for executing the apps and services. All experiments were conducted in an office room environment with the 3 Raspberry Pi nodes and the Dell edge gateway placed in the same local area network within 5m radius.

We deploy 4 containerized apps using our architecture and they are set to be available as services in our prototype system. A Java program that simulates App Manager and Profiler invokes the blockchain contract to procure resources to invoke services. This setup helps evaluate the effects of sharing resources and services on the network. The containerized apps we use for our evaluation are:

- whoami: greets the invoking user: size: 2.1 MB
- busybox httpd: a full-fledged web server: size: 3.1 MB
- mysqld: a mysql database: size: 190MB
- dockerui: UI for docker management: size: 4.5 MB

B. App Execution latency

We consider 3 aspects when evaluating app execution latency: (a) Resource procurement, (b) App deployment, and (c) App invocation. We will compare these steps for our proposed *Collaborative* approach, purely computing on *Edge* and joint edge and collaborative when the system will *Fallback to Edge* when the resources are not available in the collaborating nodes. Here, edge implies a monolithic powerful edge with compute capabilities closer to the source of data, and our solution implies an "Elastic Edge" with multiple nodes collaborating. The Edge is not opportunistic and all resources

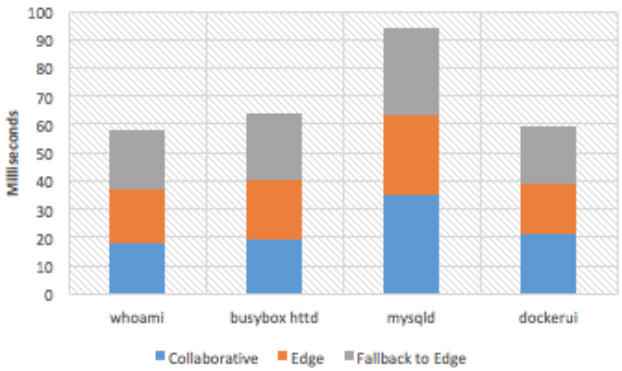

Fig. 4. App Execution Latency

necessary for the execution of apps and services have to be provisioned at the Edge when the app/service is initiated.

We observe from Fig. 4 that our collaborative system's execution latency is comparable to that of a purely centralized Edge computing system. This shows that a collaborative distributive approach to mobile IoT is feasible. We also note that the collaboration may not necessarily provide all resources requested, in which case the system is able to fallback to the edge. We can observe that the fallback to edge approach has a minimal overhead as resource discovery time is very small in the blockchain based architecture due to the information availability on all nodes through smart contracts.

Note that in our setup the Edge was placed in the same local network within a 5m radius of the collaborating nodes. The latency for app setup phase will scale with the distance between the requesting nodes and the edge unit. In a separate benchmark experiment, we observed a network latency (cellular) of the order of 50ms for an Edge computing device placed 10 miles away from the experiment office location. In the collaborative approach, the apps are present and available on the local network. It is important to note that once the resource is procured for a service app, it is available instantaneously for all other user apps that access the service.

We direct the reader to Section IV to understand, through an empirical model, how our evaluation results can help infer the performance of larger scale networks using our collaborative architecture.

C. Resource Utilization Efficiency

In contrast to our proposed collaborative approach, a purely edge computing environment does not take advantage of the device resource pool.

Table I shows the baseline resource usage of the 4 nodes in our network, and Table II shows the resource usage after deploying the apps using our collaborative model. We can observe from the usage statistics that collaborative approach enables to allocate resources through a proper matching of available with requested. In our test case, we ensured that the resources required for the 4 apps are available in the collaborating nodes. In reality, the system will have to fall

back to the edge or cloud in case the resource is not available. However, the collaboration presents a first hand opportunity to run the applications and services merely through mutual agreements and only approach the central edge/cloud units when no agreements can be reached. Due to the distributive nature of blockchains, such a case is rare as finding at least one node that has the available resource has a non-trivial probability.

Node	RAM	Storage	CPU
RP1	0.381 (38%)	5.922/29 (20%)	2%
RP2	0.329 (33%)	7.271/29 (25%)	2%
RP3	0.421 (42%)	8.372/29 (29%)	2%
Dell	0.597 (30%)	11.383/25 (46%)	3%

TABLE I
RESOURCE USAGE BEFORE DEPLOYING APPS – PERCENTAGE OF TOTAL CAPACITY OF THE NODE

Node (App)	RAM	Storage	CPU
RP1 (whoami)	0.395 (40%)	5.938/29 (20%)	2%
RP2 (httpd)	0.471 (47%)	7.422/29 (26%)	4%
RP3(dockerui)	0.489 (49%)	8.391/29 (29%)	3%
Dell(mysqld)	0.723 (36%)	12.123/25 (48%)	9%

TABLE II
RESOURCE USAGE AFTER DEPLOYING APPS– PERCENTAGE OF TOTAL CAPACITY OF THE NODE

To understand the resource utilization better we discuss a use–case to highlight the benefit of our approach: A gaming app needs resources from multiple nodes as a single node does not have the required resources. The request has to be supported by multiple nodes with partial resources. Suppose the request is for 200 MB of memory, a camera and a microphone, it is possible to fulfill the request from multiple nodes by allocating 50 MB from 4 different nodes and camera and microphone from individual nodes. However, the caveat here is that the app must be able to accept resources from multiple sources. E.g. an application that requires camera and microphone at the same location may not benefit, however, it can resolve to find a node that offers the RAM while using the camera and mic from a different node. Our framework supports such a management of resources. Our distributed model allows for handling heterogeneous set of resources from physically separated entities, which gets complicated using traditional central resource allocation techniques. This is made possible through the use of blockchain.

IV. DISCUSSION

Our evaluation in this paper involved a 4 node blockchain network. The key inferences from our evaluations translate well with the scale of the network – more number of nodes. Due to limitation of resources we have not run experiments at large scale, however, to help understand the performance of our approach at any scale we present a model for clarity. We will use the app execution latency as the metric of evaluation.

The app execution latency t_{AE} is the cumulative sum of the resource procurement time t_{RP}, app deployment time t_{AD} and app initiation time t_{AI}. The app deployment time is a function

App Size (MB)	App Name	Collaborative - IT Services			
		Resource Procurement	App Deployment	App Initiation	Total
2.1	whoami	14	3	1	**18**
3.1	busybox httd	14	4	1	**19**
190	mysqld	14	20	1	**35**
4.5	dockerui	14	6	1	**21**

Fig. 5. IT Service App Latency Matrix

of transmission rate and app size and the app initiation time is transmission latency coupled with message processing time to procure a resource handle. The procurement time is the same as the time it takes for the replicas to reach consensus on when a resource has been released for use or reacquired into the pool. The slope is dependent on a few factors; proportional to the queuing delay at each replica in a bigger network (>4 replicas) and inversely proportional to the transmission delay (increased transmission delay offsets the queuing delay at each replica).

The 3 latency measures can be expressed as,

$$t_{RP} = (slope * n_{replicas}) + b; slope = a * \frac{t_{queue}}{t_{trans}} \quad (1)$$

where, a and b are empirically measured constants that account for network factors that impact performance and compute capacity of nodes in a heterogeneous network, respectively.

$$t_{AD} = \frac{size_{app}}{rate_{trans}}; t_{AI} = t_{trans} + t_{msgprocess} \quad (2)$$

Case study. On a 10G Ethernet, we observed the numbers noted in Fig. 5 for a 4-replica blockchain network, but this is extendable to bigger clusters using the model presented above. The overall comparison to other deployment models in Fig. 4 takes into account these numbers for the collaborative blockchain approach. For the resource procurement stage, the consensus numbers were measured starting with 1 node and then replicas were added to the same to demonstrate decentralized usage. In t_{RP}, the empirical slope was observed to be 3.13 as the node count was increased from 1 to 4; t_{queue} and t_{trans} were not captured separately. Then, for a 4-replica network, $t_{RP} = 14 = 3.13 * 4 + b$ and from the observed values in Fig. 5, b was set to 1.48 for subsequent runs. For the deployment and initiation steps, we consider the *busybox httpd app*, with a size of 3.1 MB. The deployment time is $t_{AD} = size_{app}/rate_{trans} = (3.1 * 10^6 bytes)/(10 * 10^9) bits/sec = 3.1ms$. This is very close to our empirical observation of close to 4 ms. The Initiation time to request and procure a resource handle (total 20 bytes) is $t_{AI} = t_{trans} + t_{msgprocess} = 0.1ms + t_{msgprocess}$ (based on 10G network). This is close to our observation of 1ms and the $t_{msgprocess}$ was set to 1ms for subsequent runs.

This model can be used further to explore larger scale collaborative blockchain networks. Such a study is out of

scope for this paper and we aim to accomplish that as a part of our future work plan.

V. Related Work

We discuss related works from the areas of resource abstraction platforms, resource lookup algorithms, distributed resource sharing platform and service invocation platforms.

Ardalan Amiri Sani et. al. [5], proposed an architecture *RIO*, for the abstraction of resources using a separation of the application layer from the operating systems services and kernel layer. Their approach addresses the specific area of resource abstraction with cross memory mapping. Daniel J. Dubois et. al. [6], proposed a middleware *ShAir*, for P2P resource sharing. It elaborates on the abstraction of resources using an event bus. While the approach addresses resource abstraction, it does not address specifically the adhoc networking and resource registry aspects. Ion Stoica et al. [7] discuss the approach of P2P lookup of Internet applications. It uses a distributed lookup protocol that helps map a given key to a node with the right content and efficiently handles the dynamic registration and exits of the nodes in the network. Salem, et.al, [8] have proposed a mechanism for sharing resources at the edge. It uses registries and a central mediation to choose the right resources based on demand. Also, they introduce the concept of compensation for the resources used.

IBM and Samsung [9] have brought together in a proof of concept, the blockchain as a repository of assets, their artifacts and related services, along with the P2P fabric using Telehash that enables discovery and communication. This is testament to the fact that decentralized IoT is permeating the industry and there is a perceived need for it in the near term. One of the key aspects to fully realize it is the flexibility to execute any apps or services on any device using any resource in the pool. SingularityNet [4] has brought together multiple AI services talking to each other to drive synergies from a decentralized AI. The underlying platform to support this level of communication is a decentralized system on blockchain that support services and apps. The group led by Prof. Bhaskar Krishnamachari in USC [10] explores block chain technology for diverse areas. These are clear evidences that the decentralized platform to advertise, discover and instantiate services and apps using blockchains is key to materializing highly sophisticated concepts in diverse areas.

Our proposed architecture helps provide a one–stop solution to: (a) Sharing of system resources and services, (b) Decentralized mediator and decision making, and (c) Compensation for sharing. While such features have been discussed in many technologies and research works before, combining all into a single entity framework has not been well explored.

VI. Conclusion

This paper explored the idea of an opportunistic collaborative resource sharing for mobile IoT systems. We designed a novel architecture that uses blockchains for collaboration. We developed a mechanism that enables opportunistically identifying available resources and advertised services, and to invoke or utilize them based on a collaborative consensus. We implemented a preliminary prototype of a 4–node collaborative IoT network using RaspberryPis and an edge computing gateway. Through experimental evaluation we showed the feasibility of the collaborative architecture with overall app execution latency and resource utilization being comparable to the traditional centralized edge approach. We also presented a model to help determine the latency of the system at any scale.

References

[1] DockerOrg. https://www.docker.com/what-docker.

[2] BitcoinOrg. https://bitcoin.org/en/.

[3] Ethereum. https://ethereum.org.

[4] Cassio Pennachin Ben Goertzel, David Hanson. Singularitynet: A decentralized, open market and inter-network for ais.

[5] Min Hong Yun Ardalan Amiri Sani, Kevin Boos and Lin Zhong. Rio: A system solution for sharing i/o between mobile systems. *MobiSys '14 Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014.

[6] Konosuke Watanabe Daniel J. Dubois, Yosuke Bando and Henry Holtzman. Shair: Extensible middleware for mobile peer-to-peer resource sharing. *ACM 978-1-4503-2237-9/13/08*, 2008.

[7] David Karger M. Frans Kaashoek Ion Stoica, Robert Morris and Hari Balakrishnany. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM '01*, 2001.

[8] Ahmed Salem and Tamer Nadeem. Lamen: Leveraging resources on anonymous mobile edge nodes. *ACM 978-1-4503-4255-1/16/10*, 2016.

[9] IBM.com. Adept: Empowering the edge; practical insights on a decentralized internet of things.

[10] BlockChainUsc. http://blockchain.usc.edu.

[11] Dieter Hogrefe Benjamin Leiding, Parisa Memarmoshrefi. Self-managed and blockchain-based vehicular ad-hoc networks. *ACM 978-1-4503-4462-3/16/09*, 2009.

[12] Markus Claeens Michael Vogler(B), Fei Li. Colt collaborative delivery of lightweight iot applications. *IoT360 2014, Part I, LNICST 150, pp. 265/272, 2015*, 2005.

[13] Mukesh A Zaveri2 Sathish Kumar1. Clustering for collaborative processing in iot network. *ACM ISBN 978-1-4503-4204-9/16/05*, 2005.

[14] Thilo Kielmann Roelof Kemp, Nicholas Palmer and Henri Bal. Cuckoo: a computation offloading framework for smartphones. *MobiCASE 2010: Mobile Computing, Applications, and Services*, 2010.

[15] Dongwan Shin William Claycomb. Towards secure resource sharing for impromptu collaboration in pervasive computing. *SAC?07 March 1115, 2007, Seoul, Korea*, 2007.

[16] Mushtaq Ali Muhammad Wasif Nabeel, Abdullah Embong. Computation offloading for smart internet devices. *978-1-4673-6722-6/15/ 31.00 2015 IEEE*, 2015.

[17] Simon Jouet Jeremy Singer Fung Po Tso, David R. White and Dimitrios P. Pezaros. The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures. *Distributed Computing Systems Workshops (ICDCSW)*, 2013.

[18] S.Swamynathan S. Sasirekha. Collaboration of iot devices using semantically enabled resource oriented middleware. *VisionNet '16, Jaipur, India*, 2016.

[19] Christophe Huygens Jef Maerien, Sam Michiels and Danny Hughes. Enabling resource sharing in heterogeneous wireless sensor networks. *ACM 978-1-4503-3234-7/14/12*, 2012.

[20] Shelly Grossman Guy Golan Gueta, Ittai Abraham and Dahlia Malkhi. Sbft: a scalable decentralized trust infrastructure for blockchains. *arXiv:1804.01626*, 2018.

[21] Xiaolin Chang Harish Sukhwani, Joze M. Martinez and Kishor S. Trivedi. Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric). *Conference: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, 2017.