A Blockchain-Based IoT Networks-on-Demand Protocol for Responsive Smart City Applications

Mohamed Alkalbani, Bechir Hamdaoui, Nizar Zorba† and Ammar Rayes‡
Oregon State University, Corvallis, OR 97331, alkalbmo,hamdaoui@oregonstate.edu
† Qatar University, Doha, Qatar, nizarz@qu.edu.qa
‡ Cisco Systems, San Jose, CA 95134, rayes@cisco.com

***Abstract*—This paper proposes a distributed resource sharing protocol for enabling dynamic deployment of IoT networks on-demand in smart cities. The proposed protocol leverages Blockchain technology to: (i) enable distributed and secure management of IoT devices; (ii) provide fast discovery of IoT resources and scalable on-demand networks; and (iii) ensure reliable fund transfers for service payment among the network entities. The protocol relies on a peer-to-peer network infrastructure to allow communication among the IoT devices in a distributed manner, and uses a self-recovery/self-healing mechanism to ensure robustness against device failure and maliciousness. The protocol also introduces and uses a reputation system to monitor and keep track of services delivered by registered devices for quality of service delivery assurance. We implemented and evaluated the proposed protocol intensively using simulations to assess its effectiveness in terms of scalability to network sizes and robustness to device failures.**

***Index Terms*—Smart City, Internet of Things (IoT), Blockchains Technology, Participatory IoT Networks on-Demand.**

I. Introduction

One key technology that can be potentially leveraged for promoting smart cities is the use of Internet of Things (IoT) devices [1], whose proliferation emerges as a key enabler for a new era of services and applications that urban cities can benefit from greatly, by enabling new city applications ranging from realtime surveillance and precision health to city traffic control and emergency management. These envisioned IoT applications would typically involve and rely on a collection of geographically distributed IoT devices, with sustained Internet connectivity, that collectively and interactively perform some task as dictated by the underlying application. Such a collection of IoT devices, forming what we term here *a participatory IoT networks-on-demand* instance, is created dynamically, on-demand, upon a request made by some end user, referred to as a *consumer*.

Albeit their great potentials in enabling networks on-demand in urban cities to enhance city services and responsiveness, the proliferation of IoT devices has also given rise to new unique challenges, specifically the need for having to deal with network congestion bottlenecks, resulting from the high demands of wireless bandwidths needed to provide Internet access to these IoT devices, and with the IoT devices' limited resources (power, CPU, etc.) that can potentially constrain their participation in IoT applications [2]. Such challenges must be overcome in order for IoT resource sharing protocols like the one proposed in this paper to be deployed, and thus for the IoT networks-on-demand to be enabled, thereby promoting and supporting various IoT applications and services beneficial to smart cities. The good news is that there have recently been lots of interest and work efforts from the research community, at both academic and industry levels, aimed at developing new innovative technological solutions to overcome these challenges. Examples of emerging technological solutions that tackle the high demands in wireless bandwidths and that are becoming fortunately mature and ready to be adopted include cognitive radio network access (e.g., [3–6]), MIMO (e.g. [7–9]), and wideband/mmWave spectrum usage (e.g., [10–13]). As for dealing with the limited resources of IoT devices in terms of energy, computation, and storage capacities, technologies such as edge cloud computing (e.g. [14–17]), in-network caching (e.g. [18, 19]), and device offloading (e.g. [20–22]) have emerged as potential solutions for such resource limitation problems, and are also becoming more mature and ready for adoption. For secure management of the IoT devices and safe communications among them, Blockchain [23] emerged as a suitable solution that can tackle such issues in a distributed manner. These new technologies will allow millions of IoT devices in large urban cities to be provided Internet access and connectivity, with enough data rates, computation resources, and storage capacities.

In this paper, we propose a distributed, blockchain-based protocol that enables dynamic deployment and mapping of networks-on-demand instances on top of IoT devices. Specifically, we leverage blockchains technology to allow distributed IoT resource sharing on-demand by easing the management of IoT devices, enabling the mapping of networks-on-demand requests on top of the IoT resources, and ensuring robustness against malicious behaviors using self-recovery/self-healing mechanisms. The proposed protocol also introduces and relies on a reputation system to monitor and keep track of services delivered by registered IoT devices, to ensure that quality of service (QoS) and service-level agreements (SLA) requirements are met. Using simulations, the proposed protocol is evaluated intensively to assess its scalability performance and its robustness to device failures.

The rest of the paper is organized as follows. In Section II, we describe the network infrastructure used by the proposed protocol. In Section III, we present our proposed IoT resource sharing on-demand protocol. In Section IV, we study and assess the effectiveness of the proposed protocol vis-a-vis of its scalability with network sizes and robustness to device failures.

Finally, we conclude the paper in Section V.

II. Network Model

We consider a city-wide network infrastructure, constituted of many IoT devices distributed randomly across the city, and a set of edge clouds/access points also spread across the city to provide Internet connectivity to the devices. We assume the city is divided into L regions. A device interested in participating in networks-on-demand needs to advertise, upon joining the network, its device characteristics or directory containing information such as its resource type, resource availability, bounty, etc. The system allows devices to submit networks on-demand requests, to be mapped on IoT devices already in the network. A device submitting a request is referred to as a consumer, and an example of a network on-demand request could be initiated to track a target (e.g., malicious person) that is moving through different regions in the city. Submitted requests are propagated to other IoT devices in the network using a peer-2-peer communication infrastructure. Our proposed protocol uses blockchains to enable distributed management of the IoT devices and the mapping of submitted requested onto these devices, and hence, a set $\mathcal{M}$ of IoT devices are designated to serve as miners, which are responsible for handling fund transfer and device payment through creation and addition of blocks; more on this will be provided later.

We assume that each IoT network on-demand request is submitted in the form of 5-tuple G=(V, $\mathcal{E}$, D, $\mathcal{C}$, B) where: V is the number of requested nodes (also referred to as request size), $\mathcal{E}$ is the set of edges between the requested nodes, D is the duration of request, $\mathcal{C}=$ $\{(Loc_1, Type_1, Cap_1), \ldots, (Loc_V, Type_V, Cap_V)\}$ is the (location, resource type, resource capacity) of requested nodes, and B is the bounty associated with the request.

When a device joins the network, its characteristics are added as an entry to a shared data structure, called `Directory`, which is updated every time a new device joins the network, and is stored and maintained by all devices through the blockchains (`BC`), to be described later. More specifically, each entry of `Directory` contains the following characteristics.

- *Device ID:* Serial number and/or IP/MAC addresses.
- *Resource Type(s):* Sensing (video, temperature, traffic density), computation, and/or communication.
- *Location:* GPS location and region ID.
- *Availability:* Time periods during which device is available for participating in networks on-demand.
- *Capacity:* Amount of resource available for sharing; i.e., computation (CPU power), sensing (sensed data per second), battery level (dBW) and/or bandwidth (bps).
- *Network access type:* Technology type used for connecting to the network; e.g., WiFi and/or cellular.
- *Bounty:* Cost of service.

Note that the terms device and node will be used interchangeably throughout to refer to the same thing.

III. The Proposed IoT Resource Sharing On-Demand Protocol

This section provides a detailed description of the proposed protocol, which can be described through three main phases: Initialization phase, Resource Reservation phase and Fund Transfer phase. Initialization phase deals with joining and registration of newly arrived devices to the network. Resource Reservation phase tackles the needed steps to be taken to allow for the requests' mapping, including finding and approving providers, monitors and miners. Fund Transfer phase is the phase where funds are transferred from the consumer to the provider nodes, once a request is fulfilled. Throughout this paper, we assume that neighboring nodes are nodes that are connected to the same access point. Next, we describe each of these three phases.

A. Initialization Phase

New nodes joining the network have to obtain the following information before they can become part of the ongoing peer-to-peer network.

- W: Private/Public Key pair associated with a device to serve as its wallet unique address.
- `BC`: The blockchains currently being maintained and used by the nodes in the peer-to-peer network.
- `Directory`: Directory file containing all devices' characteristics, currently being used by the nodes in the network .

New nodes start the initialization phase by obtaining a wallet. This could be obtained by simply downloading and running a piece of software to create a unique Private/Public Key pair; a random seed is used to obtain such a Private/Public Key pair combination. New nodes have to obtain a copy of `Directory`, which contains an up-to-date information about the different resources and nodes in the network. New nodes connect to a DNS, maintained by different provider and miner nodes in the network, to get information about highly reputed nodes in the network. The incentive to provide this service is to allow for more consumers to join the network and request resources. There could be multiple DNSs with multiple seeds that point to different nodes in the network. New nodes use DNS seeding to obtain a list of nodes in the network in which it can request `Directory` from. New nodes can limit themselves with a shortened version of `Directory` that only contains a handful of highly reputed nodes in each region. Also, connecting to neighboring nodes or any node in the network allows new nodes to obtain `Directory`. Depending on the nodes' purpose from joining the network, it can decide whether or not to download full `BC` version. This is done in a similar fashion to obtaining `Directory` by connecting to other miner nodes. Miner nodes are the only nodes that need full `BC` version. Once `BC` is obtained, new nodes can verify the integrity of all transactions throughout the network by verifying the signatures and public keys associated with each entry in `BC`. Since this is computationally intensive, a node can limit itself with a light `BC` version, where a *Light*

BC *is a blockchains of shortened length that includes a limited set of past mined blocks, plus a current state of all nodes' balances in the network*. If a new node desires to become a provider, it advertises its resources to neighboring nodes, and using gossip protocol [24], all nodes in the network add the new nodes' information to their local Directory. The initial value of R (the node's reputation) is set to 0, and its associated W address is included in Directory. New nodes can now listen to a specific port for transactions and for any new requests that are submitted to the network.

B. Resource Reservation Phase

A consumer node starts by creating a request G to be advertised to the network. It also has to sign the request to enable provider nodes to verify wallet address ownership and fund availability. Consumer nodes that decide to have full version of Directory will have full view of the network and can request specific nodes from Directory. To limit the overheard associated with reserving resources, a consumer node could opt for a lighter version of Directory and limit itself to connecting to a handful of nodes only. Consumer nodes that opted for the lighter version of Directory have to communicate their requests with other nodes using gossip protocol [24]. In the case of direct communications, provider nodes can accept or reject, depending on whether they can meet the constraints and/or bounty associated with the request. If a provider node does not reply due to it being down, the consumer node advertises that the node went down, and all other nodes update their Directory accordingly. Consumer nodes use first come first serve when picking provider nodes, and the provider nodes that reply to the request first gets to take the request. Also, if the request has to be satisfied with more than one node, then it must peform the following:

- Provider nodes start by accepting a percentage of the requests, a percentage that can be related to the location of resources requested, duration and percentage of data to be computed, etc. Provider nodes could also decide to reject the request and become monitor nodes instead by appending their info to the request as monitor nodes before propagating the request further.
- Accepting provider nodes append their acceptance info to the request and pass it to neighboring nodes.
- The process continues until the request is accepted or rejected. This can be done by limiting the number of hops until the request is mapped to some threshold, H, or by using a timeout threshold, T_{max}; i.e., if the request is not fully accepted within a certain time period, it is rejected.
- Once the request is accepted entirely (all of its requested nodes are found), then provider nodes collaborate to perform the assigned task. This can be achieved because all provider nodes will have accepting nodes information due to the appending process.
- First nodes to get the request are more likely to accept it, which makes it more likely for closer nodes to accept neighboring nodes' requests. This allows for real time resource reservation with minimum delays due to locality.

If the request is not accepted, then the requesting consumer node has to create a new request with lower constraints or higher bounty. If the request is accepted, a minimum number, M, of monitors is chosen. In the direct communication approach, 50% of the monitor nodes are chosen by the consumer node and the other 50% are chosen from neighboring provider nodes. This is done to limit the possibility of malicious monitor nodes. Also, if nodes were chosen using gossip protocol and from the lighter directory file, monitor nodes append their information to the request as it propagates, and the first M nodes to identify themselves as monitors are chosen. Monitor nodes keep track of provider nodes, including CPU usage, Uptime, Disk usage, RAM usage, network traffic, etc. to ensure SLAs are met and to prevent maliciousness. Monitor nodes have the ability to raise a flag indicating a misbehavior from a specific provider node. In this case, a voting round including all monitor nodes is done, during which it will be decided whether the said provider node violated the set SLAs or not. If the number of votes is higher than a threshold, V_{min}, then the node is flagged as misbehaved and its misconduct is broadcasted throughout the network. Reputation is set to 0 for the misbehaving node and the accepted request is re-advertised to neighboring nodes. If a provider node went down during task performance, its status in the directory is changed to inactive by monitor nodes and it is broadcasted throughout the network. Monitor nodes get to accept the resources first because of their proximity from the misbehaving node. This creates another incentive for monitor nodes to keep monitoring provider nodes to detect failures and misbehavior. Once request is over and if no flags were raised, monitor nodes get to vote to decide whether provider nodes met the request's SLA or not. If votes exceed V_{min}, a new entry is broadcasted to be added to BC that includes consumer node, provider nodes, monitor nodes, and the request, G. Monitors receive a Monitor Bounty, Mb, as a reward, which is a percentage of the bounty associated with the request. This provides an incentive to do the monitoring job. And the provider's reputation is increased for their successful request fulfillment.

C. Fund Transfer Phase

Miner nodes listen to broadcasts sent by consumer nodes, provider nodes and monitor nodes at all times. They add transactions that include fulfilled requests and reputation changes to their backlog once received. As transactions accumulate, miner nodes verify each transaction using the public key and signature associated with each transaction. At every mining period chosen by the protocol, a miner node is selected at random to add a block to the blockchains, BC. The probability of a miner node being selected is related to how much stake it has in the system (total funds). Miner nodes with more stake will have higher probability of winning the mining task. The equation used for determining the winner miner is

$$Hash(PrevHash, WalletAddr, Content, Time) \leq \frac{Balance(WalletAddr)}{TotalBalance} \times Difficulty$$

where:

- $PreviousHash$: Previous hash generated for the last block in the blockchains.
- $Content$: The string of transactions/reputation changes to be added to the block.
- $WalletAddr$: The public key associated with the miner node.
- $Time$: The period selected by the protocol to produce a miner node winner.
- $Balance(WalletAddr)$: Balance of the miner node.
- $TotalBalance$: Total funds/balance available for all nodes in the network.
- $Difficulty$: The current difficulty level set for network.

The hashing function produces a random value, based on the inputs, that is compared against the right side of the equation. The adjustable $Difficulty$ parameter of the network serves as a method to ensure that only one winner miner is selected. Every miner computes this function, and if it satisfies the equation, it advertises the new block to the rest of the network to be added. In case no miner meets the constraint and no nodes in the network receive a valid block by next the mining period (period selected by the protocol to produce a miner node winner), all nodes in the network decrease the value of $Difficulty$. This mechanism ensures that at the next mining period, the likelihood of a miner meeting the constraint increases. If two miners met the constraint and advertised the new block to be added to the rest of the network, the block is dropped by all nodes and $Difficulty$ is increased, thereby reducing the likelihood of two miners meeting the constraint by the next mining period.

New mined blocks are said to be unconfirmed until L_{min} blocks have been mined. Once the number of blocks that have been mined reaches L_{min}, all the previous transactions in current block are said to be confirmed and the winner miner receives a reward, R. This gives an incentive for miner nodes to participate in miner selection process. If one of the miner nodes attempted to create fake entries in the `BC` and broadcast it to the rest of the network, other nodes in the network would drop it if it does not satisfy the equation/constraint.

IV. Performance Evaluation

In this section, we present and analyze the performance results of the proposed protocol using simulation.

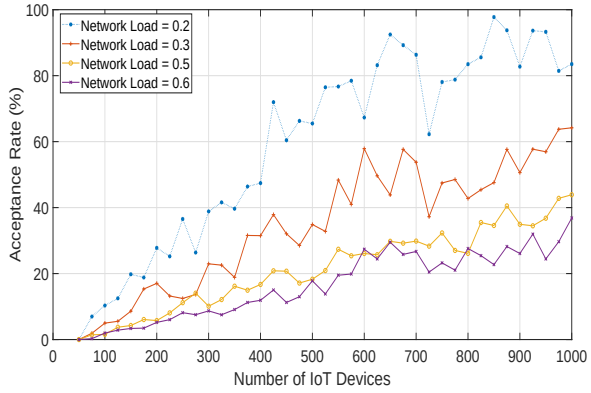
A. Simulation Setup

We consider a network of N nodes/IoT devices placed randomly in a city. The network is modelled as a graph whose vertex set is the set of all N nodes and the edge set consists of random connections among the nodes in the system. We assume that the city is split into $L = 9$ regions, and the devices are distributed randomly within the different regions. Only fully connected graphs, where each node can be reached from any other node, are simulated. The number of neighbors of each node is selected uniformly between 15 and 25. Each node is randomly assigned a resource type, which can be either of sensing or computing type. At every time period, a new miner is selected based off the miner's stake in the network as described in Section III-C. When a request is successfully accepted, it is propagated throughout the network and added to the miners' pending ledger. Every winning miner picks from the pending requests based off the highest bounty associated with the pending requests. We assume that nodes are faulty (whether intentionally or unintentionally), in that a provider accepting to serve a request may, with some probability, fail to deliver its service, thereby causing service disruption and violation of SLAs. The proposed protocol contains a recovery mechanism that allows recovery from such failures by promptly finding other nodes that can fulfil and replace the failing nodes. In the event when no nodes can be found, the request is dropped, and is categorized as a 'failed request'. Each node starts with zero balance, and the balance increases as the node participates more and more on requests and receive more and more rewards.

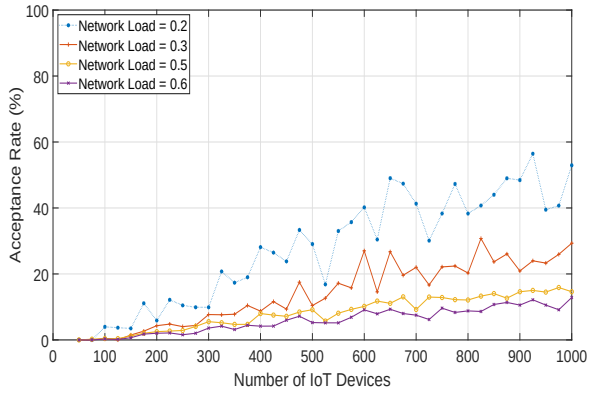
In this evaluation, we consider a time-slotted system, with the number of requests that arrive at each time step is Poisson distributed with mean r. For each arrived request G, the specifications are selected as follows: the request size, V, is varied between 5 and 20, the request duration, D, is drawn from a Bernoulli process with parameter q i.e., average duration (in number of time slots) equals $1/q$), the resource type of the request is set to 0 (sensing) or 1 (computing) with equal probability, the locations of requested nodes are selected randomly, and the request bounty, B, is selected uniformly between 100 and 1000. Each node has a minimum bounty, B_{min}, below which a request is denied. B_{min} is also drawn uniformly between 100 and 1000. We define the *network load* as $\frac{r}{q}$, which represents the average number of requests that would have been present in the network at a time slot had no arrived request been denied to the network. The average number of requests that are actually present in the network equals 'network load' times the 'accepted rate' of arrived requests. In our simulation, we varied the network load between 0.2 and 0.6. Finally, the size of mining period, τ_{mining}, (one bock is added to the blockchains every mining period) is set to 3 time slots, and the number of monitors, M, is set also to 3.

B. Scalability Analysis

We first study the performance behavior of the proposed protocol to assess how well it scales with the network size (number of IoT devices) by investigating the impact of the network size on its Acceptance Rate. More specifically, the Acceptance Rate is defined here as the rate at which requests are accepted into the network, and is calculated by dividing the number of accepted requests by the total number of requests submitted to the network. In Figure 1, we present the Acceptance Rate while varying the number of IoT devices

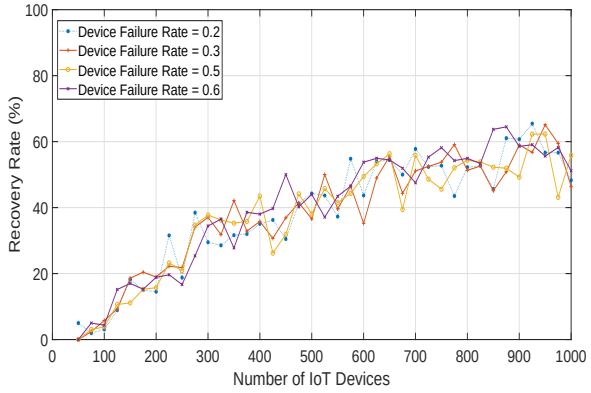


(a) Request size = 10

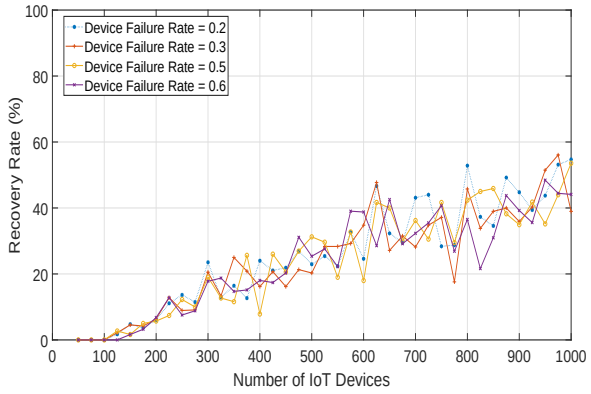


(b) Request size = 20

Fig. 1. Acceptance rates of requests when varying number of IoT devices under different network loads



(a) Request size = 10



(b) Request size = 20

Fig. 2. Recovery rates from device failures when varying number of IoT devices under different device failure rates: network load = 0.5

for different network loads and under two request sizes, 10 and 20, where again the request size is the number of IoT devices being requested. As expected, we observe from the figures that the acceptance rate decreases as the network load and/or the request size increases. This is because as the network load and/or request size increases, more nodes in the network become committed to requests, making it harder to find nodes that satisfy the new requests' requirements. We also note that as the network size grows (the number of IoT devices increases), the acceptance rate increases, merely because more nodes in the network implies higher chances of meeting new requests' requirements and hence higher acceptance rates.

C. Fault Tolerance

We now study the robustness of the protocol against node failures. We consider that nodes can fail before or after accepting a request, and assess how well our protocol recovers from such failures by measuring the Recovery Rate of the protocol under different failure rates and request sizes. The Recovery Rate is defined as the fraction of accepted requests that are recovered successfully from device failures, and is calculated by counting the number of successfully recovered requests and dividing it by the total number of failed requests.

Figure 2 shows the behavior of the recovery rate as a function of the number of IoT devices under different request sizes (10 and 20), and different node failure rates (probability of device failure is set to 0.2, 0.3, 0.5, and 0.6) when network load is 0.5. First, observe that as the number of IoT devices in the system increases, the recovery rate increases, regardless of the request size and the node failure rate. This is because the higher the number of nodes, the greater the likelihood of finding nodes that satisfy the failed nodes' requirements, thus increasing the overall recovery rate. For reasonable network sizes (e.g., 1000), the recovery rate ranges from 50% to 80%, depending on the node failure and request size. Second, note that the device failure rate has little effect on the recovery rate. The reasoning is that when increasing the device failure rate, the protocol is still able to recover from failures that happen to different nodes in the network. Since the load is constant, the likelihood of finding a node that satisfies the failed network requirement is the same. Our last finding is that as the request size increases, the recovery rate declines slightly, and this is regardless of the node failure rate. As the request size gets bigger, more nodes are committed, and hence, it becomes difficult to find replacement for failed nodes, but the effect is minimal as we only see a slight drop in the recovery rate.

This is due to the fact that there are still unoccupied nodes throughout the network that can be picked as a replacement for the failed nodes. To summarize, our results show that the proposed protocol is robust against faulty nodes and is able to fully recover from faulty instances for reasonable network and request sizes.

V. Conclusion

We presented a blockchain-based IoT resource sharing on-demand protocol that enables distributed mapping, management and maintenance of networks on-demand instances formed on top of and by IoT devices. The protocol can be used in smart cities to enable and support a variety of IoT-enabled applications and services, such as real-time surveillance, emergency operation management, traffic monitoring and control, and many others. We showed through simulations that our protocol scales efficiently under different system parameters and is resilient to faulty devices.

Acknowledgment

This work was supported in part by the Cisco Systems.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] B. Khalfi, B. Hamdaoui, and M. Guizani, "Extracting and exploiting inherent sparsity for efficient iot support in 5G: Challenges and potential solutions," *IEEE Wireless Communications Magazine*, vol. 24, no. 5, pp. 68–73, 2017.

[3] B. Hamdaoui and K. G. Shin, "Characterization and analysis of multi-hop wireless mimo network throughput," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2007, pp. 120–129.

[4] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE personal communications*, vol. 6, no. 4, pp. 13–18, 1999.

[5] M. NoroozOliaee, B. Hamdaoui, and K. Tumer, "Efficient objective functions for coordinated learning in large-scale distributed osa systems," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 5, pp. 931–944, 2013.

[6] P. Venkatraman, B. Hamdaoui, and M. Guizani, "Opportunistic bandwidth sharing through reinforcement learning," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 6, pp. 3148–3153, 2010.

[7] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive mimo for next generation wireless systems," *IEEE communications magazine*, vol. 52, no. 2, pp. 186–195, 2014.

[8] B. Hamdaoui and P. Ramanathan, "A cross-layer admission control framework for wireless ad-hoc networks using multiple antennas," *IEEE Transactions on Wireless Communications*, vol. 6, no. 11, 2007.

[9] D. Gesbert, M. Shafi, D.-s. Shiu, P. J. Smith, and A. Naguib, "From theory to practice: An overview of mimo space-time coded wireless systems," *IEEE Journal on selected areas in Communications*, vol. 21, no. 3, pp. 281–302, 2003.

[10] C. Fan, B. Li, C. Zhao, W. Guo, and Y.-C. Liang, "Learning-based spectrum sharing and spatial reuse in mm-wave ultradense networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4954–4968, 2018.

[11] B. Khalfi, A. Elmaghbub, and B. Hamdaoui, "Distributed wideband sensing for faded dynamic spectrum access with changing occupancy," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018.

[12] B. Hamdaoui, B. Khalfi, and M. Guizani, "Compressed wideband spectrum sensing: Concept, challenges, and enablers," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 136–141, 2018.

[13] B. Khalfi, A. Zaid, and B. Hamdaoui, "When machine learning meets compressive sampling for wideband spectrum sensing," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*. IEEE, 2017, pp. 1120–1125.

[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[15] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[17] M. D. Kristensen, "Execution plans for cyber foraging," in *Proceedings of the 1st workshop on Mobile middleware: embracing the personal communication device*. ACM, 2008, p. 2.

[18] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE, 2012, pp. 316–321.

[19] R. Abuhadra and B. Hamdaoui, "Proactive in-network caching for mobile on-demand video streaming," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[20] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, no. 5, pp. 2795–2808, 2016.

[21] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[22] S. Abdelwahab, S. Zhang, A. Greenacre, K. Ovesen, K. Bergman, and B. Hamdaoui, "When clones flock near the fog," *IEEE Internet of Things Journal*, 2018.

[23] B. Hamdaoui, N. Zorba, and A. Rayes, "Participatory IoT networks-on-demand for safe, reliable and responsive urban cities," *IEEE Blockchain Newsletter*, 2019.

[24] R. Chandra, V. Ramasubramanian, and K. Birman, "Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks," in *Distributed Computing Systems, 2001. 21st International Conference on.* IEEE, 2001, pp. 275–283.