An Effective Policy Sharing Mechanism for Smart Home Networks

Carol J. Fung* and Bill McCormick†
* Virginia Commonwealth University, Virginia, USA.
Email: cfung@vcu.edu
† Email: billmcc@runbox.com

***Abstract*—The rapid advancement of the Internet of Things has brought new technologies to smart homes and smart hospitals. In a smart home environment, devices may be configured to accomplish certain tasks along with other devices and digital services through user configuration. The configured policy by one user can be further shared to the other users in the network. In this work, we study the policy control application and propose a novel user policy propagation mechanism that allows quality IoT policies to spread to more users than their ineffective counterparts, by integrating user and policy reputation tracking. We evaluated the proposed mechanism using a simulated framework. Experimental results demonstrate that under our designed policy propagation mechanism, high quality policies spread much more widely than low quality policies when the system is configured properly. At the same time, reputable users can have a higher impact and thus their policies can spread further.**

***Index Terms*—Internet of Things, IoT Policies, Smart Homes, Network management**

I. Introduction

The advancement of the Internet of Things (IoT) has triggered a new wave of technological innovations. Devices in smart home environments have brought plenty of convenience and controllable options into people's daily lives. For example, smart locks and smart surveillance cameras can be configured in a way that users can turn them on or off using the owner's PCs or smartphones [1]. Users can also configure devices to take actions based on certain conditions through logical programming. For example, users can configure their A/C system to turn down when the owner leaves the home or turn up when the owner is on the way back home. A new policy configuration app IFTTT [2] provides a platform for IoT users to create simple policies using the template "if this then that". For example, "if the owner is leaving home on a weekday then turn down the A/C system". Figure 1 shows a few examples of IoT policies that involves thermostats.

However, configuration and policy creation for home devices may not be an easy task for all users. Expert users may be able to create new policies based on their needs and share it with their friends who may find the policies useful. Inexperienced users may benefit from the policies shared by other expert users and pick the policies they find useful. However, without quality control in a policy sharing environment, low quality policies may be rampant and it may discourage users from adopting policies using the network. A centralized quality control design may use authentic screening

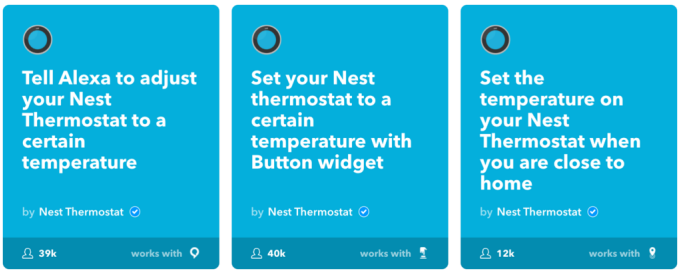


Fig. 1: Examples of IoT policies

to filter low quality policies but it will be a bottleneck for large scale networks where a large number of new policies are being created every day. The centralized server can also be a single-point-of-failure. The design of an effective and scalable IoT policy sharing mechanism to maximally benefit IoT users is a challenging problem.

To address the challenge mentioned above, we propose an IoT policy sharing mechanism with the goal of automatically pushing the most useful set of policies to potential users. The mechanism is designed based on a fully decentralized device policy sharing social network where expert users and novice users mingle together to support each other. Our mechanism contains several components including reputation monitoring for each policy and user, a feedback collection system that provides input to update the reputations, and a decision model that determines whether to propagate the policy further. Our simulation demonstrates that our design can effectively propagate high quality policies much further than low quality policies and the reputation scores of users converges to their true expertise levels.

The contribution of this work can be summarized as follows: 1) we propose a simple, scalable, and effective IoT policy sharing and propagation mechanism for smart home networks; 2) we design a reputation tracking mechanism for both users and policies based on the feedback from policy adopters; 3) we developed a simulation framework for IoT policy sharing networks to evaluate the proposed mechanism design.

The rest of the paper is organized as follows. In Section II we briefly go through some existing work related to IoT policies and policy sharing mechanisms. We present our policy sharing mechanism design in Section III and our evaluation results are described in Section IV. After discussion in Section V we conclude this paper in Section VI.

978-3-903176-31-7 © 2020 IFIP

II. Related work

In the past few years, some work has been done in IoT policy execution, policy design, and policy management.

A. Policy Execution

Most existing works on IoT policy focus on the execution of policies. For example, IoTGuard [3] is a system that can block unsafe apps that violate the IoT policies configured to protect the users. Xu et. al [4] proposed an IoT network security situation awareness model using a situation reasoning method based on semantic ontology and user-defined policies. Li et. al. [5] proposed a policy-based secure and trustworthy sensing scheme for IoT named RealAlert, in which the trustworthiness of both data and the IoT devices are evaluated based on both the reporting history and the context in which the data are collected using policy rules. All the above works focus on how to execute user policies in the system.

B. Policy design

There are several works focusing on privacy protection policy design so that policies can be created by smartphone users. For example, Enck et al. proposed Kirin [6] for smartphone users in 2008. Kirin takes both permission combinations and permission and action string combinations into consideration. This type of design can only be used by experts. To alleviate users from the complex policy configuration job, Fitzgerald et. al. [7] proposed an automated smartphone security configuration based on a threat-based model. In his model users only need to specify high level policies such as which resources are critical and what wifi connections are trusted. The system automatically generates security policies based on the recent threats found. The IFTTT [2] is a policy design that allows users to configure their IoT policy using a simple "If This Then That" grammar structure to define their policies. This type of policy is user friendly to set up and interpret. Therefore, it is considered usable by inexperienced users.

C. Policy management and sharing

In the literature there are few works about policy sharing in the IoT field. Most IoT policies are created by experts and stored either on a cloud server or fog device [8], [9]. In recent years there is an increasing trend on user defined IoT policy platforms such as IFTTT [2], Zapier [10], and Microsoft Flow [11], which are used to bridge the divide between IoT devices and digital services such as e-mail services and social media platforms. These platforms allow users to write policies that connect IoT devices with the events and actions of digital services. Policies are managed by a centralized system and users are allowed to search them when needed. This type of system allows user defined policies to be shared with others in the system. However, a centralized system requires a large amount of resources allocated to control the quality of policies and it is vulnerable to the single-point-of-failure.

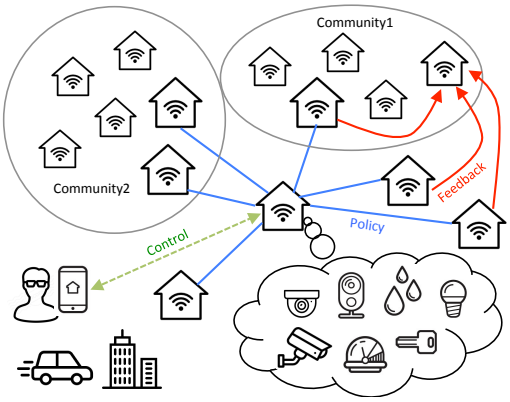


Fig. 2: Smart home Policy sharing network Illustration

Based on the best of our knowledge, our work is the first of the same kind that addresses how to do controllable user-defined IoT policy sharing and management through a fully distributed network model.

III. Policy Management and sharing mechanisms

In this section we first address the motivation of this work and then present our model design for policy sharing and management.

A. Motivation

We envision a smart home environment where users can create their own IoT policies or adopt policies created by other users. A fully centralized policy management model is simple but it faces many challenges. For instance, a centralized model require a huge amount hardware and network resources reserved to store and maintain the policies when the network size grows to a large scale. The centralized resources can also become the target of malicious attacks and single-point-of-failure. In this work we propose a different approach - a fully distributed policy sharing and management network, to over come the aforementioned problems.

As shown in Figure 2, a fully distributed policy sharing overlay network consists of a group of smart home users who create, share, and maintain policies that are related to the devices installed in the homes. For instance, a smart home device user may want to create a home device policy to turn off security cameras when the owner is at home so that the owner's privacy can be protected. The owner can set "deactivate <the security cameras> if the GPS of <Phone> is less than 100 feet away from <home>" (Figure 3), where parameters in the angled brackets are to be connected to corresponding IoT devices. The new policy can be recommended to the neighbors who share the same privacy concerns and own security cameras. Users can adopt policies from their neighbors if they cannot create their own. However, due to the various expertise levels of users, the quality of the privacy policies can be different. Our objective is to design a policy propagation mechanism so that good policies can be shared to more users and low quality policies will stop propagating quickly. In this

section we describe the design of a privacy policy sharing mechanism that fulfills our goal.

B. Network Model

In order to accurately describe the system, we first introduce some notations used in the paper. Let $\mathcal{G} = \{\mathcal{U}, \mathcal{E}\}$ denote a device policy sharing social network which consists of n IoT users $\mathcal{U} = \{u_1, u_2, ...u_n\}$. Suppose each user i is connected to a number of neighbors denoted by $\mathcal{N}_i$, through an IoT policy management app on their smartphones. Then $\mathcal{E} = \{e_{ij} | \forall i \leq n, j \in \mathcal{N}_i\}$. In this model, when a user creates or adopts a policy passed by a neighbor, the policy will be further passed on to their neighbors who share the same IoT devices. We use E_i to denote the expertise level of a user i. The expertise level is defined by the capability of a user in terms of creating quality policies. The set of policies generated by user i is denoted by $\mathcal{R}^i = \{r_1^i, r_2^i, ..., r_m^i\}$, where m is the total number of policies user i creates. We use q_k to denote the quality of policy k. The probability for user i to accept a policy k is denoted by p_k^i. The notations used in this paper are summarized in Table I.

C. Security/privacy policy Design

IoT policies are operational policies for IoT devices that instruct devices on how to operate (e.g., turn on/off) when the condition or the status of the devices change. For example, a user may want to protect their privacy from an IoT device such as a surveillance camera when it is not needed. This can be realized by creating a policy that turns off interior surveillance cameras automatically when the owner is at home, based on owner's phone location. An example is shown in Figure 3. A policy can involve multiple devices and the policy can utilize the information from multiple devices to decide what actions a particular device may execute.

An IoT policy should have several features that help users to understand its usefulness and effectiveness : 1) The purpose of the policy. This is the description of the policy that states what the policy can be used for. 2) The functional part of the policy which formally describes device actions and the conditions that trigger those actions. 3) A record of the policy – for example, what is the reputation of the policy and how many people installed this policy. 4) Information about the policy creator – for example, the creator's reputation score and the number of policies developed by the creator. 5) Metadata such as the time and location where the policy was created.

TABLE I: Notations used in this paper

Notation	Description
$\mathcal{G} = \{\mathcal{U}, \mathcal{E}\}$	Smarthome network consisting of users and their connections
U	Set of smarthome users in the network
n	Number of users in the policy sharing network
$\mathcal{N}_i$	The number of neighbors for user i
E_i	The expertise level for user i
$\mathcal{R}_i$	The set of IoT policies created by user i
q_k	Quality of policy k
R_k	The reputation of policy k
U_i	The reputation of user i
p_k^i	Probability for user i to accept a policy k
λ	The average policy creation rate for users
θ_t	The probation time of a policy
K	The propagation factor

Action	Condition	Comment
Turn off $sc1-$sc4	Dis($p1, $Home) < 100	Turn off surveillance cameras when in home
Turn on $sc1-$sc4	Dis($p1, $Home) > 200	Turn on surveillance cameras when leaving home

Fig. 3: An Example of IoT policies content

An IoT policy useful to one user may also be useful to other users who share similar devices. For example, if the policies in Figure 3 are adopted by another user, the only thing the user needs to do is to connect the device parameters to their own home devices. Such customization can be guided through a simple wizard program. Once the connections are set, the new user may further customize some other variables such as the distance at which the policy becomes active or inactive.

An effective privacy policy can propagate to a large number of other users in the network using multiple hops by spreading through neighbors. The propagation mechanism should be such that good policies spread to most users and bad policies should stop propagation quickly. In the next subsection we will present a simple policy propagation design that aims at meeting this objective.

D. Policy Propagation Mechanism Design

Policy propagation can be controlled through a policy management application on users' smartphones. In this subsection we describe our design of the IoT policy propagation mechanism that aims at propagating useful policies and constraining ineffective policies.

In the smarthome global network, each node represents a home user or business network user. Each node maintains a list of neighbors. In our design, polices are propagated through users' social connections. When a user receives a push of a new policy which is potentially useful, the user can choose to try out the policy. During this time, we say the policy is in a probation period (e.g., 1 day). When the probation period expires, the user will be asked a question regarding whether the policy meets their expectation. The user can choose to accept the policy, reject the policy or continue evaluating the policy. The user's decision of accept or reject a policy will be submitted as a feedback to update the reputation scores of the policy as well as the creator.

1) Reputation of the policy: When a policy is evaluated by a user after the probation period, the feedback is collected by a node in the network which maintains the reputation score of the policy. The ID of the node is determined by the hash value of the policy content $I_k = h(r_k.content)$, where $h : * \rightarrow \{ID\}$ is the hash function that maps any content to a set of node IDs in the network. When another user would like to check the reputation score of a specific policy, it can send

a request to the corresponding handling node to retrieve the score. Whenever a handling node receives new feedback from a user regarding a specific policy i, the updated score of policy i by the creator j is computed as follows:

$$R_i(m) = \beta R_i(m-1) + (1-\beta)F(m) \quad (1)$$

$$R_i(0) = U_j \quad (2)$$

Where equation (1) triggers after an evaluation has been received. $R_i(m)$ is the reputation score of policy i after feedback m is received. $\beta \in [0,1]$ is a *forgetting factor* that controls the decay rate of the accumulated reputation scores. Smaller forgetting factors place less emphasis on the historical reputation values. m is the total number of feedback received for policy i. $F(m) \in \{0,1\}$ is the feedback provided for policy i. The initial reputation score of a policy is the reputation score of the creator, which will be further explained in the following subsection.

2) Reputation of the creator: In addition to tracking the reputation score of each policy, the system also maintains the reputation score for each policy creator. The reputation of a policy creator is an aggregated score of all the reputation scores of the policies that user has created. The reputation score of a user i is handled by node $h(ID_i)$, where h is a hash function that maps one ID to another ID in the network. Wherever a feedback for a policy created by user i is submitted, the corresponding user reputation handler also receives a notice and will update the reputation score of the creator as follows:

$$U_j(n) = \alpha U_j(n-1) + (1-\alpha)F(n) \quad (3)$$

$$U_j(0) = 0.5 \quad (4)$$

Where equation (3) is triggered every time a policy feedback is submitted by a user. $U_j(n)$ is the reputation score of user j after nth feedback has been received about the policies created by user j. $F(n) \in \{0,1\}$ is the nth feedback and $\alpha \in [0,1]$ is the forgetting factor. Each user has an initial reputation 0.5 when they join the network.

3) policy propagation mechanism: When a user registers into the policy sharing network, the system will ask the user to register all IoT devices in the home automatically. Based on the type of available devices and location, the system will recommend a list of neighbors. The user can also choose to follow some specific users in the network that they already know. An initial list of relevant device policies will be recommended to the new user, based on what the neighbors have in their database.

When a user creates a new IoT policy, the policy is immediately available to all the neighbors of the user and other users who follow this user. The metadata of the policy will also be available to neighbors such as the reputation of the creator. Neighbors can decide whether or not to try the policy.

If a user chooses to try out a policy, the policy will be added into the neighbor's database after a configuration process to specify the how the policy will be applied to the devices on the user's network. At this time, the *probation period* θ_t of the policy starts. After the probation period, the system will pop up a question to ask for feedback from the user. The user can choose to accept the policy, reject the policy or continue evaluation, which will trigger another round of probation time. The feedback will be recorded to update the reputation scores of the policy and the creator.

After the user makes a decision after the probation period. The system will further propagate the policy to the next K neighbors with a probability equal to the reputation of the policy. For example, if the current reputation of of policy r_1 is 0.5, the probability of recommending the policy to each of the K neighbor is also 0.5. If the user does not try the policy after a period of time, then we considered the user is not interested and the policy will be passed to a randomly chosen neighbor to continue the evaluation.

E. Storage for reputation scores

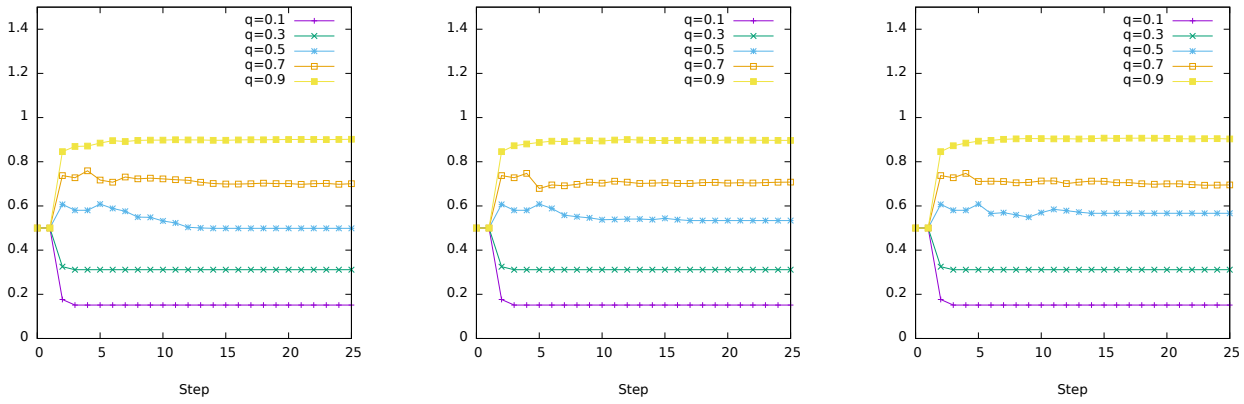
As we mentioned there will be reputations calculated for each policy and each user. However, where to store those scores in a fully distributed network? Our solution is to construct the policy sharing network into a peer-to-peer Chord network [12] and utilize the Chord structure to search for the peer that stores the reputation score of a policy or user. The node ID that stores the reputation score can be found through a mapping function $h : x \rightarrow N$, where h is the hash function that performs the mapping, x is the policy content or the user ID, and N is the set of nodes in the network. When a new policy feedback is generated, it will be delivered to the corresponding nodes that manage the reputation score of the policy and the reputation score of the policy creator.

IV. Evaluation

A. Simulation setup

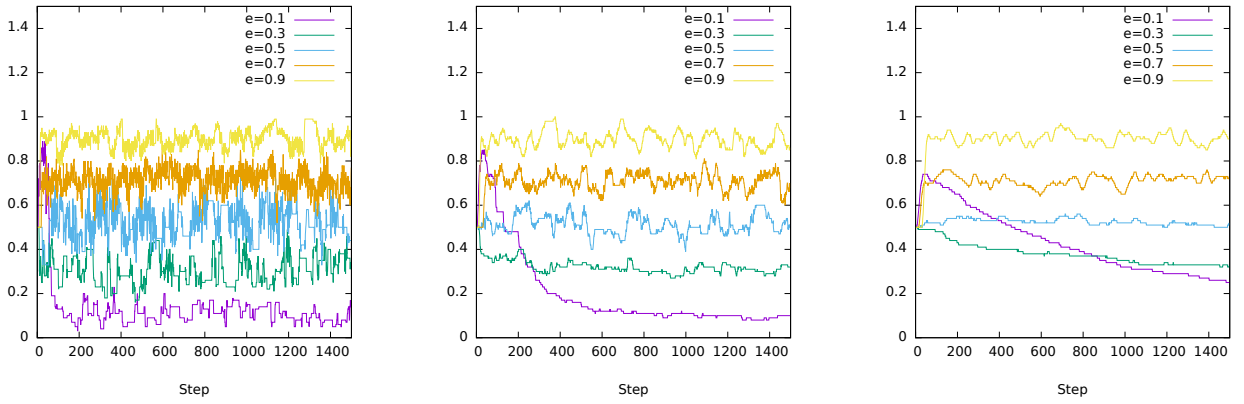
In the simulation we create a set of smart home users. These users create policies and share them with their neighbors in the network through a network-based message propagation. Each user i is created and assigned with a level of random expertise E_i in the range of $[0,1]$. The initial reputation level of each user is set to be 0.5. As a user creates and disseminates policies through the network, its reputation score is updated using a update law described in last section so that the user's reputation increases when a positive feedback is received, and decreases when a negative feedback is received.

In the simulation each user randomly creates a set of IoT policies. Each policy j has a quality metric on $[0,1]$, which is determined randomly using a uniform distribution with mean equal to the user's expertise level. Each policy has an initial reputation score which is assigned the value of the user's reputation score when the policy is created. As the policy is propagated through the system, the policy's reputation score is updated when feedback from the users are collected on the policy. Connections between users are built based on user proximity. Users are assigned a random location in a square zone. Each user builds a connection to the closest K neighbours in the system. When propagating a new policy, a user will forward the policy its connected neighbours.



(a) $\beta = 0.9$ (b) $\beta = 0.99$ (c) $\beta = 0.999$

Fig. 4: The reputation of policies under three different β settings



(a) $\alpha = 0.9$ (b) $\alpha = 0.99$ (c) $\alpha = 0.999$

Fig. 5: The reputation of users under three different α settings

B. Implementation of the policy propagation mechanism

When a new policy is created, the following steps take place as it is propagated through the system:

1) Creator: Each users receives a reputation score of 0.5 initially, which will be updated later over time. We let the originating user create policies following a Poisson process at a rate λ. The originator forwards the policy to all of its connected neighbours.

2) Receiver: When a user receives a policy from its neighbour, it first checks if it has already seen the policy. If the user has not seen the policy, it will take one unit time to assess the policy (probation period). After that the receiver decides whether to accept the policy or reject the policy. The decision in the simulation is modeled using Equation 5, where p_k^i is the probability that policy k is accepted by user i. q_k is the quality of policy k and E_i is the expertise level of user i.

$$p_k^i = 1 - (1 - q_k)E_i \tag{5}$$

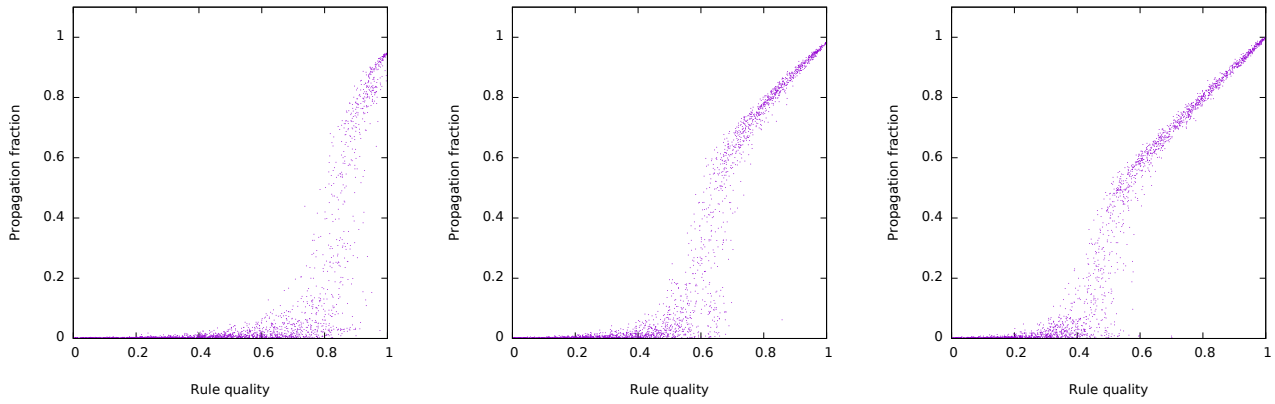
3) Policy: Once a user decides to accept or reject a policy, the system makes a stochastic decision about whether to forward the policy on to it's neighbour set with probability of the reputation value of the policy R_k. That is, the higher reputation a policy is, the higher likelihood it will be propagated to the next neighbor.

This mechanism propagates policies through the system using a form of stochastic flooding. The intent of the stochastic decisions is to limit the spread of low quality policies while giving high quality policies the ability to reach most users.

C. The Reputation of Policies

In the first experiment we study how the reputation of policies evolve in the system. We set up a network of 1000 users and the number of neighbors of each user is set to 6. We created 5 different policies from the same user and each policy was assigned a different quality level of 0.1, 0.3, 0.5, 0.7 and 0.9, respectively. We track the reputation of the Policy after each time unit under three different β values ($\beta = \{0.9, 0.99, 0.999\}$).

As shown in Figures 4 (a) - (c), in all the cases the reputation of policies converged to their quality levels quickly in less than 25 steps under all β settings. Based on the above results, we fix the parameter $\beta = 0.99$ in the rest of the experiments.



(a) $K = 4$ (b) $K = 6$ (c) $K = 8$

Fig. 6: The propagation factor vs. the fraction of users receiving the policy

D. The Reputation of Users

Next we ran a set of simulations to assess change of user reputations over time. Just like we did in the last experiment, we used 1000 users and each users has 6 neighbors. Among the set of users we select 5 users and set their expertise levels to be $0.1, 0.3, 0.5, 0.7$ and 0.9, respectively. The remaining 995 users were assigned random expertise levels. At each step one of the 1000 users was randomly selected to generate a new policy. The average quality of the policies generated by a user i is the same as their expertise level E_i. The reputation scores of all users start from 0.5 and they are updated when the feedback of their policies are collected. We ran the experiment for 1500 steps each time with a different setting of forgetting factor $\alpha = \{0.9, 0.99, 0.999\}$ and monitor the reputation scores. The results are plot in Figure 5 (a) - (c).

From the results we can see that under all α settings the reputation score of users converges towards their expertise levels. With a larger forgetting factor the convergence takes longer time but the reputation scores are more stable. When the forgetting factor is 0.9 the reputation scores oscillate greatly which is a sign of instability. At the same time the reputation score approach their expertise level fast.

Based on the results from these simulations we used values of $\alpha = 0.99$ and $\beta = 0.99$.

E. The impact of the propagation factor K

The purpose of this simulation is evaluate how much impact the propagation factor K can bring to the effectiveness of policy propagation and to assess an effective K value for each user. Remind that the propagation factor is the number of neighbors that the system decides to propagate to after the probation period. We conducted simulations for K =4, 6 and 8 respectively. In the simulation we created 1000 nodes and ran for 2500 steps. At each step, a node was randomly selected to create a new policy and the policy quality value is fixed to be q. The policy was propagated through the network according to the propagation policies depicted earlier. We vary the K value and the policy quality value q in each run and track the number of users that each policy has spread to after 2500 unit time.

The results are shown in the Figures 6 (a) - (c). In each figure the x-axis shows the policy quality q and the Y-axis shows the proportion of users that a policy can spread to. In all three figures we can see a consistent pattern - the higher quality a policy is the more nodes it can spread to. We can also see that when $K = 4$, only high quality policies ($q > 0.9$) can spread to most users (80% users). When the K increases to 6 or 8, the policy quality requirement is relaxed to 0.8 to spread to majority users. We also observed that in all three figures there exists a "crashing point" where the impact of a policy drops significantly while the policy quality decreases. It represents the transition period to the definition of bad quality policy, which will be suppressed to spread far. We call the end of transition period a *transition point*. We can see that when $K = 4$ the transition point is around $q = 0.9$. When K increases to $K = 6$ and $K = 8$ the transition point shifted to lower spots at around 0.7 and 0.6 respectively. As a summary, with a smaller K value only high quality rules can spread in the network. When K is larger, more policies can have chance to be propagated to other nodes.

V. Conclusion

In this work we proposed a novel IoT policy sharing and management mechanism. More specifically we build a fully distributed overlay network where each user are allowed create IoT policies and share their policies to their neighbors. They can also adopt policies from neighbors. Policies will propagate naturally through social connections based on the reputation of policies and the propagation factor K is used to determine the suppression threshold. No central facilities or authorities are needed in this model. We use a simulation approach to evaluate the effectiveness and the controllability of such network. Our simulation results demonstrate that the impact of policies are proportional to their quality levels, and several parameters can be used control the propagation mechanism.

REFERENCES

[1] "The best smart locks for 2019," https://www.pcmag.com/article/344336/the-best-smart-locks.

[2] "Ifttt app," ifttt.com.

[3] Z. B. Celik, G. Tan, and P. D. McDaniel, "Iotguard: Dynamic enforcement of security and safety policy in commodity iot." in *NDSS*, 2019.

[4] G. Xu, Y. Cao, Y. Ren, X. Li, and Z. Feng, "Network security situation awareness based on semantic ontology and user-defined rules for internet of things," *IEEE Access*, vol. 5, pp. 21 046–21 056, 2017.

[5] W. Li, H. Song, and F. Zeng, "Policy-based secure and trustworthy sensing for internet of things in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 716–723, 2017.

[6] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 235–245.

[7] W. M. Fitzgerald, U. Neville, and S. N. Foley, "Automated smartphone security configuration," in *Data Privacy Management and Autonomous Spontaneous Security*. Springer, 2012, pp. 227–242.

[8] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proceedings of the 2014 IEEE 15th international conference on information reuse and integration (IEEE IRI 2014)*. IEEE, 2014, pp. 16–23.

[9] S. B. Calo, M. Touna, D. C. Verma, and A. Cullen, "Edge computing architecture for applying ai to iot," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 3012–3016.

[10] "Zapier," https://zapier.com/.

[11] "Microsoft flow," https://flow.microsoft.com/.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.