Integrating Encrypted Mobile Agents with Smart Spaces in a Multi-agent Simulator for Resource Management

Sherin M. Moussa
University of Illinois at Urbana-Champaign, Computer Science, Urbana, IL, USA
Ain Shams University, Faculty of Computer & Information Sciences, Cairo, Egypt
Email: sherin@cs.uiuc.edu

Gul A. Agha
University of Illinois at Urbana-Champaign, Computer Science, Urbana, IL, USA
Email: agha@ cs.uiuc.edu

Abstract—**The existence of advanced smart devices and related technologies such as pervasive computing, mobile wireless communications, sensor networks and agent technologies have supported the proliferation of smart spaces. In this paper, we present the design of "Bosthan", a multi-agent-based simulation tool that manages resources consumption in multi-inhabitants smart spaces. Bosthan is built on the top of ActorNet mobile agent platform to simulate different smart space topologies with varying numbers of residents. It allows strategies for resolution of conflicts between mobile agents, and for preserving inhabitants' anonymity and untraceability inside the smart spaces, to be studied. Bosthan will help us compare the efficiency of using mobile agents to allow smart spaces to act pro-actively and maintain anonymity and data privacy in multi-inhabitants environments**

Index Terms—**Multi-agent Systems; Smart Spaces; Simulators; Mobile Agents**

I. Introduction

The existence of advanced smart devices and related technologies such as pervasive computing, mobile wireless communications, sensor networks and agent technologies have supported the proliferation of smart spaces. Smart spaces are technically-enhanced physical environments that can sense the existence of its inhabitants and adapt to their behavior or preferences to provide several services, i.e., inhabitants' safety, cost reduction to maintain the environment, and resources optimization. With context awareness feature, smart spaces can also act according to the current context without explicit inhabitant intervention to increase usability and effectiveness, taking environment attributes into consideration, Ref. [1, 3, 5, 6]. Thus a smart space aims to construct an intelligent automation with the target to provide its inhabitants with maximum possible comfort and minimum resource consumption. A mobile agent is an agent that can transport its state and data from one executing environment to another, and is capable of performing appropriately in the new environment. Generally, mobile agents have five security requirements: confidentiality, availability, accountability, integrity and anonymity. There are some inherent threats associated with mobile agents because the agent's owner and system's operator are different.

In this paper, we describe the design of Bosthan, a modular simulation engine that supports agent-based smart space simulations. Bosthan is a tool for agent-based discrete event simulations that will enable the analysis of different smart space scenarios, ranging from small environments such as smart rooms, to large smart spaces such as smart homes, smart offices or even smart buildings. We use Bosthan to study how mobile agents can be used to provide a resource consumption management framework that preserves inhabitants' anonymity and untraceability rules through applying non-interactive evaluation of encrypted functions, without inhabitants' intervention, in multiple intersected contexts within a developed smart space. Our goal is to use Bosthan to investigate the effectiveness of strategies to resolve conflicts that may arise between competing agents due to these contexts intersections. This can be applied to simulate environments where residents' privacy is highly required.

The aim is to show that it is possible to provide a friendly environment with sufficient comfort to its inhabitants without revealing their identity. Specifically, we study if using mobile agents may have significant advantages in bandwidth consumption and reliability in resource-limited scenarios. This requires continuous mobility awareness ability and adaptive learning to automatically adjust smart space's behavior while inhabitants can still change their patterns and preferences through time. Bosthan uses ActorNet mobile agent platform that deploys mobile agents in wireless sensor networks.

II. Related Work

MAVHome ResiSim in Ref. [6] provides an abstract virtual home that focuses on simulations to investigate agent systems behavior that control an intelligent home. UbiWise in Ref. [10] is a 3D-based simulator that allows prototypes of new sets of devices and protocols to be simulated using a Java program. The goal of UbiWise is to explore the behavior of computation and communications devices. 3DSim in Ref. [8] allows including any UPnP device into the environment and allows arbitrary UPnP control points to interact with other devices. It aims to support the development of human-ambient-interaction systems such as PDA based control systems, adaptive user interfaces, multimedia output coordination, and goal-based interaction systems. Custodian in Ref. [11] allows untrained people to fully design and test smart homes for people with disabilities and older people. It is a visualization tool that enables users to test scenarios, set-up configurations and demonstrate the functional working design.

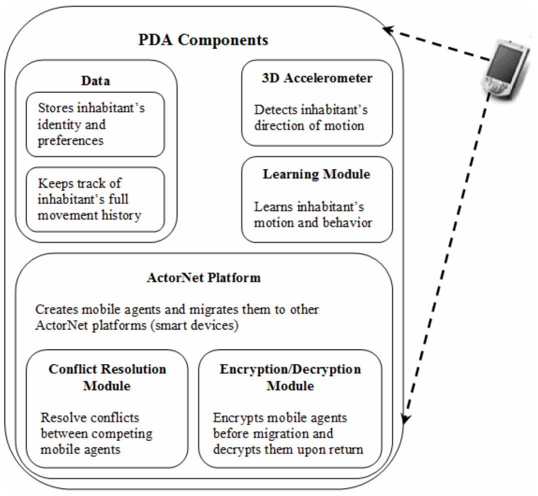
III. Motivating Scenario for a Smart Space

Consider the following scenario for a smart home. Several inhabitants are currently in a smart home. Each is carrying his PDA. PDAs are assumed to have a 3D accelerometer, which can detect the direction of motion. The smart home consists of many zones, i.e., master bedroom, children room, living room, kitchen …, etc, which are connected through pre-defined valid routes. When an inhabitant starts to move, the PDA detects that his user is trying to exit from his current location. The PDA checks the history stored in its knowledge base to predict what the next destination of his user can be within the possible valid routes. It obtains the priority level of its user in the predicted zone, the average duration he stays there, and his average number of entries to this zone in order to compute the inhabitant's weight function. The PDA evaluates the weight function except for the closeness to the median of preferences. It then creates a mobile agent for each smart device in the predicted zone, having the inhabitant's preference to this device and the partially computed weight function. The PDA encrypts the mobile agents using an encryption technique, and then migrates them – each to its assigned smart device.

A smart device usually keeps track of the median of preference values residing at its side. When a mobile agent arrives at a smart device, it checks the number of mobile agents residing there, if any. If their count is below the predefined public threshold, it uses the median of preferences to determine the closeness of its preference to this median. Finally, it evaluates the weight function while it is still encrypted in order to resolve conflicts between agents. The agent with the higher weight controls the device according to its preference. If the agents count exceeds the threshold, the agents vote for the preferences values. Thus by the time the inhabitant arrives to the predicted zone, all smart devices would be working already according to his preferences and ready for his arrival, without knowing any information about his identity.

When the inhabitant is in, manual interactions would be not needed unless there is a change in behavior that agents should detect in order to learn when they are back to their originator PDA. Thus, mobile agents remain at the smart devices detecting any changes in activities until their inhabitant representative starts to move again. When motion is detected, the PDA calls for its agents to migrate back. Accordingly, the mobile agents save any detected behavior changes and migrate back, leaving the rest of the agents to re-compete over the smart devices control and so on. Therefore, mobile agents keep following the inhabitant from one location to another as he moves throughout the environment in order to satisfy his preferences in advance by issuing automated service requests on his behalf while preserving his privacy and untraceability. Upon return, the PDA decrypts its agents, learns any updates in its inhabitant's preferences, and stores this move in his history logs. Hence, resources in the smart home are managed without inhabitants' intervention for maximum satisfaction and privacy. Fig. 1 shows the software add-ons that should be in a PDA to develop this scenario. In order to achieve this scenario, we should design a framework that considers several issues:

- What is the efficiency of smart homes computational environments?
- Why should we use mobile agents?
- What kind of multi-agent platforms to deploy?
- Why does a mobile agent need protection?
- How can we guarantee security?
- How can mobile agents resolve conflicts?
- How can we provide privacy?


PDA Components
Data
Stores inhabitant's identity and preferences
Keeps track of inhabitant's full movement history
3D Accelerometer
Detects inhabitant's direction of motion
Learning Module
Learns inhabitant's motion and behavior
ActorNet Platform
Creates mobile agents and migrates them to other ActorNet platforms (smart devices)
Conflict Resolution Module
Resolve conflicts between competing mobile agents
Encryption/Decryption Module
Encrypts mobile agents before migration and decrypts them upon return
Figure 1. Framework-related PDA components

IV. Why a Smart Space Simulator?

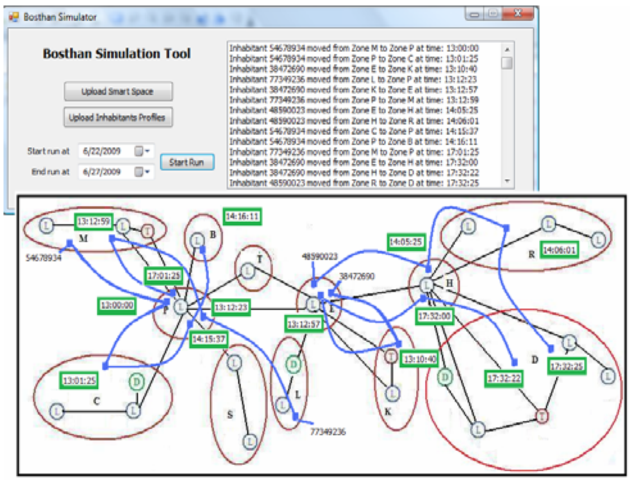
Consider a smart space where several inhabitants are moving from one zone to another. When an inhabitant leaves a zone, the smart space should predict what the next zone of this inhabitant can be, depending on his identity and learnt history of movements. The smart space can use this information pro-actively, i.e., switch on lights, air condition, or TV, while switching off all devices in the left zone according to inhabitant's learnt behavior and preferences. Hence, all inhabitants' preferences and tracking history should be continuously available to preserve this "smartness" feature, Ref. [3, 5]. Having many inhabitants in the same zone, the smart space should have the ability to resolve conflicts resulting from competing, conflicted preferences to control smart devices.

We propose using mobile agents on wireless sensor networks as an optimum architecture for such a smart space framework. Mobile agents are used to 1) maintain untraceability; once an inhabitant leaves a location, his related information should not be available afterwards, that is, no tracking information 2) overcome limited resources, i.e., connectivity, power and bandwidth) 3) control the framework structure. Therefore, the main concern in such a framework is how to create small, lightweight mobile agents that can manage resources consumption effectively in a limited resources computing environment. These lightweight mobile agents must act on behalf of their inhabitants to control smart devices depending on their preferences and adapt their processing results while learning inhabitants' changes in behavior through time. In addition, a mobile agent must be able to resolve any conflicts that may arise due to other mobile agents that coincidently may exist within the same context and try to control the same device at the same time.

Evaluation of mobile agent systems and smart spaces involves many concerns, Ref. [13]. This includes the ability to accurately investigate the effect of different interacting parameters on each other in an unpredictable environment leading to a large number of computations, besides, the difficulty to realistically model truly adaptive behavior in mobile agent systems within a static environment. Is it possible to know for certain that the runtime environment is identical from one run to another? In case of a problem, is it possible to know that it occurs at exactly the same time in two different runs? Many layers of performance are involved making it difficult to fix some parameters while changing others in a real-world test environment. Simulation would be easier to change parameters and run again. Thus, a general-purpose simulation tool would be more convenient as a test environment rather than real-world environment

V. BOSTHAN SIMULATOR ARCHITECTURE

We propose "Bosthan", a modular agent-based simulated environment for smart spaces that can model real inhabitants' interactions in a multi-inhabitant smart space to analyze different smart space scenarios and evaluate our proposed framework for resource management. Fig. 2 shows the main interface of Bosthan. In our simulated environment, our goal is to measure the efficiency of using mobile agents to manage resources and preserve anonymity of inhabitants. Bosthan should be generic; it can simulate any number of inhabitants with different motion patterns and can model any building topology or floor plan for a smart space. Accordingly, it can configure any sensors layout of different sensor types.



```
Inhabitant 54678934 moved from Zone M to Zone P at time: 13:00:00
Inhabitant 54678934 moved from Zone P to Zone C at time: 13:01:25
Inhabitant 38472690 moved from Zone E to Zone K at time: 13:10:40
Inhabitant 77349236 moved from Zone L to Zone P at time: 13:12:23
Inhabitant 38472690 moved from Zone K to Zone E at time: 13:12:57
Inhabitant 77349236 moved from Zone P to Zone M at time: 13:12:59
Inhabitant 48590023 moved from Zone E to Zone H at time: 14:05:25
Inhabitant 48590023 moved from Zone H to Zone R at time: 14:06:01
Inhabitant 54678934 moved from Zone C to Zone P at time: 14:15:37
Inhabitant 54678934 moved from Zone P to Zone B at time: 14:16:11
Inhabitant 77349236 moved from Zone M to Zone P at time: 17:01:25
Inhabitant 38472690 moved from Zone E to Zone H at time: 17:32:00
Inhabitant 38472690 moved from Zone H to Zone D at time: 17:32:22
Inhabitant 48590023 moved from Zone R to Zone D at time: 17:32:25
```

Figure 2. Bosthan Simulator

Bosthan is currently under development in the Open Systems Laboratory (OSL), University of Illinois at Urbana-Champaign (UIUC), using Microsoft Visual Studio C# 2005. As a first phase, Bosthan includes three sensor types—Light (L), Temperature (T), and Display (D)—but can be extended with others. Bosthan is designed on a modular basis to allow (1) seamless configuration of each module separately, (2) simulator scalability, and (3) ability to exchange any applied algorithm with another, i.e. different learning or encryption technique. Fig. 3 shows our proposed architecture for Bosthan agent-based simulator, which consists of ten modules:

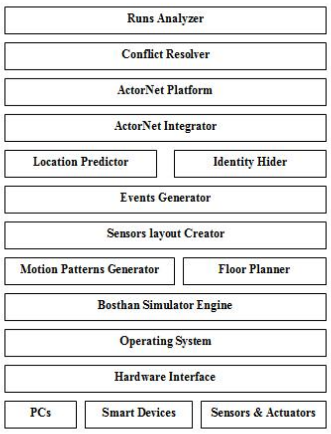


Figure 3. Bosthan high-level architecture

A. *Bosthan Simulator Engine*

This is the main component of Bosthan, which identifies how many inhabitants and zones will be in the simulated environment, maintains a full system log of history data, and integrates all modules together to guarantee coordination.

B. *Motion Patterns Generator*

Bosthan should be able to generate different motion patterns for each simulated inhabitant, based on his speed of motion, average duration he stays at each visited zone (minutes per day), and his average number of entries for that zone (per week). This module is also responsible for maintaining an inhabitant's profile, i.e. identity, preferences, and history of activities and movements.

C. *Floor Planner*

This module allows Bosthan to configure the smart space's floor plan on which the simulator will model. It can range from one zone to unlimited number of zones. It also allows the definition of the valid shortest routes between zones and the distances between them.

D. *Sensors Layout Creator*

Each zone is configured with the simulated sensors layout and the distances between each other on the defined floor plan. Light, temperature, and TV types are included in Bosthan current phase.

E. *Events Generator*

Bosthan is a discrete event simulator that generates a chronological sequence of events that forces mobile agents to react, based on ActorNet mobile-agents platform. The generated events simulate multi-inhabitants' entries to and exits from zones according to the defined valid shortest routes and inhabitants' pattern of motion, within a defined duration of run time. It also generates some manual interactions with devices to change their preferences in order to learn behavior change.

F. *Location Predictor*

Pro-activity requires prediction capabilities. To maintain functionality, a smart space should record inhabitants' daily movements and activities in order to learn a pattern out of these recordings for repeated behaviors. These patterns are used to predict what the next location of an inhabitant can be, in addition to what changes may occur to an inhabitants' preferences as time passes. For example, if a manager is used to having a weekly meeting with his team in conference room # 1 every Monday at 10:00 am, this piece of information can be used to adjust room temperature, warm the projector, and put down the blinds in preparation for the meeting in advance. In order to predict an inhabitant's next location, Bosthan considers the inhabitant's identity, all previous traversed zones, the time interval in which an inhabitant is moving, and the weekday, under the assumption that inhabitants usually tend to have different patterns on different weekdays at different times, i.e. weekends versus workdays, Mondays versus Wednesdays (a TV series to watch every Monday at 8:00 pm, a meeting to attend every Wednesday at 10:30 am… etc). Hence, Bosthan assumes its target function is a linear function of n+2 parameters as in (1);

$$\hat{V}(l) = w_0 + w_1 z_1 + \ldots + w_n z_n + w_{n+1} t + w_{n+2} d \qquad (1)$$

where n represents the number of previously traversed zones in addition to the current one, w_0 through w_{n+2} are numerical weights to be chosen by the learning algorithm, z_n is the previous zone, t is the time interval, and d is the day name. The weight w_0 provides an additive constant to the location value. Bosthan uses the Least Mean Square (LMS) training rule algorithm, where it adjusts the weights in small portions in the direction towards the reduction of the error in any predicted location. It minimizes the squared error *E* between the recorded history values and the values predicted by the hypothesis $\hat{V}$ as in (2);

$$w_i \leftarrow w_i + \eta (V_{history}(l) - \hat{V}(l)) x_i \qquad (2)$$

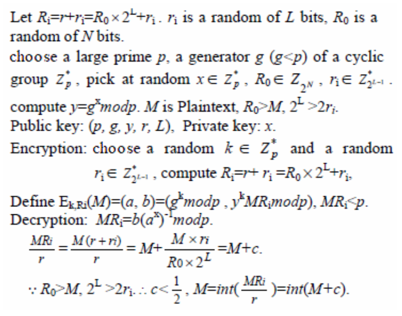
where η is a small constant number from 0 to 1 that adjusts the amount of the weight update, x_i is the attribute under study in the learning process, i.e. current location, time interval, etc. Location Predictor module can have its target function and training rule replaced by different approaches according to what is required to be applied in simulations.

G. *Identity Hider*

Intelligent environments maintain inhabitants' identity and preferences in order to adapt responses accordingly. As an inhabitant moves from one place to another, the new location should have access to the inhabitant's profile to control devices as it is stated in his preferences. Hence, manual actions can be avoided. Current smart environments used to store these profiles in a central repository, which is accessible by all zones inside a smart environment. In Bosthan, mobile agents move an inhabitant's identity and preference information from one zone to another as the inhabitant moves. Therefore, at any instance, the inhabitant's identity is only available in the zone where the inhabitant is actually in. When the inhabitant leaves, any data regarding his identity, preferences, or information about his stay at this zone, will leave with him. Thus, a current zone is the only location that can tell who is inside.

What if we create encrypted mobile agents having the inhabitants' identity that can migrate to control smart devices according to the inhabitants' preferences, and yet keep the inhabitants' anonymity? We then can hide the inhabitants' identity even in the same zone where they are. As a consequence, no adversary can trace the inhabitants while moving inside smart spaces. This feature is very essential for the places and buildings that intend to apply smart space applications, and are highly concerned with privacy and security matters. To the best of our knowledge, none of the current smart environments have conducted any research regarding inhabitants' privacy and data security in smart environments.

Many techniques have been explored to secure mobile agents, Ref. [4, 9, 12]. Eventually, most of these techniques would reveal the agent's contents, at some instance of a mobile agent's lifetime, away from the mobile agent's originator in order to allow for the mobile agent to execute successfully. If the executing platform is malicious, or it has a hidden adversary, then the inhabitant's identity is revealed. Identity Hider module applies one-round, non-interactive evaluation of encrypted functions (EEF) technique to maintain the inhabitant's anonymity and untraceability. A new ElGamal cryptosystem was proposed in Ref. [2], where new additive, multiplicative and mixed-multiplicative homomorphisms are defined. Fig. 4 presents the new ElGamal cryptosystem. Provided that Bosthan is designed on a modular basis, Identity Hider module can apply different security techniques according to the smart space policy.



Let $R_i=r+r_i=R_0\times 2^L+r_i$. r_i is a random of L bits, R_0 is a random of N bits.
choose a large prime p, a generator g ($g<p$) of a cyclic group Z_p^*, pick at random $x\in Z_p^*$, $R_0\in Z_{2^N}$, $r_i\in Z_{2^{L-1}}^*$.
compute $y=g^x modp$. M is Plaintext, $R_0>M$, $2^L>2r_i$.
Public key: (p, g, y, r, L), Private key: x.
Encryption: choose a random $k\in Z_p^*$ and a random $r_i\in Z_{2^{L-1}}^*$, compute $R_i=r+r_i=R_0\times 2^L+r_i$.
Define $E_{k,Ri}(M)=(a, b)=(g^k modp, y^k MR_i modp)$, $MR_i<p$.
Decryption: $MR_i=b(a^x)^{-1} modp$.

$$\frac{MR_i}{r}=\frac{M(r+r_i)}{r}=M+\frac{M\times r_i}{R_0\times 2^L}=M+c.$$

$$\because R_0>M,\ 2^L>2r_i.\ \therefore c<\frac{1}{2},\ M=int(\frac{MR_i}{r})=int(M+c).$$

Figure 4. New ElGamal. Cryptosystem in Ref. [2]

H. ActorNet Integrator

Bosthan integrates with ActorNet in Ref. [7] to deploy mobile agents and waits for their outcome. ActorNet is a mobile agent platform for wireless sensor networks that was developed in OSL at UIUC. Fig. 5 represents the high level architecture of ActorNet.

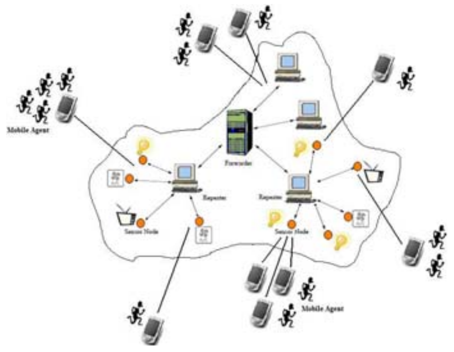


Figure 5. ActorNet architecture

ActorNet is a fine-tuned, multi-threaded embedded Scheme interpreter that provides an abstract environment for lightweight concurrent object-oriented mobile code with network interactive debugging, and high-level concurrent programming facility on a parallel wireless sensor network platform. This is a significant advantage over primitive stack-based virtual machines. Cygwin is used to run ActorNet platform, which is a Linux environment under Windows. An add-on is developed into ActorNet in order to be able to compute the median of all preferences for the agents residing at one ActorNet platform.

I. Conflict Resolver

One of the main characteristics of our framework is that it should address "multi-inhabitant" smart spaces. Each inhabitant's preference tries to gain control on smart devices over other competitors, causing conflicts that need to be resolved for the framework to keep functioning. Generally, smart spaces can be categorized as either public or private spaces. Examples for public spaces are: conference rooms, class rooms, meeting rooms… etc, whereas private spaces are like: office rooms, bedrooms, living rooms… etc. Spaces, both public and private, are usually shared between many inhabitants. In fact, we believe that public and private definitions can be interchanged; example: a meeting room in an office can be considered as a private place if the number of existents is below a certain predefined threshold, whereas a dining room in a house can be considered as a public place if there is a party and the number of invited people exceeds this threshold. Therefore, we suggest discriminating between public and private spaces depending on the number of existing inhabitants. We suggest using voting as a single peak preference function to resolve conflicts in public smart spaces. It allows each inhabitant to vote, say, for the first three preferable values for each of his preferences. When the number of inhabitants in a certain context exceeds a predefined threshold, voting technique investigates their voted values to get one value according to which smart devices will be controlled. Fig. 6 shows an example for agents voting for a room temperature as a single peak function.

Inhabitant1: 20 > 22 > 18
Inhabitant2: 18 > 22 > 20
Inhabitant3: 18 > 20 > 22

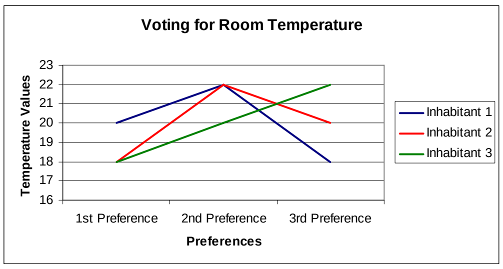


Figure 6. Voting as a single peak function

In private spaces, Bosthan assumes having several priority levels, i.e. 5, 10 levels… etc according to the smart space policy. Each inhabitant is assigned to a certain priority level per each zone, ex: an employee can be assigned to priority level 5 in his own office room, priority level 2 in other colleagues' office rooms, and priority level 3 in the break area. As many inhabitants may reside in the same priority level, additional factors should support conflict resolution. Inhabitants that tend to stay longer durations in a certain zone should have precedence over those who stay for shorter ones. In this sense, it would maintain better cost management and more inhabitants' comfort to dominate mobile agents of those longer stays inhabitants rather than switching smart devices control between shorter lifetime mobile agents for those inhabitants that tend to stay for short periods of time in that zone. Moreover, inhabitants that tend to frequently exist in certain zones should have precedence as well. On the smart devices side where competing mobile agents reside, it would be preferable to dominate mobile agents whose preferences are close to the median of preferences of other competing mobile agents so that when a mobile agent wins to control over a smart device, it still can be satisfactory for the majority of residents. This requires that inhabitants' priority computations should be performed on the smart devices side to get access to other inhabitants' preference values. Bosthan formulates a reasonable form for the sort of equation that can be used to determine an agent's weight depending on the factors discussed earlier. It then encrypts the equation as explained in section 4.3, and sends it through a mobile agent to be solved at the smart devices side to determine the dominant mobile agent. Bosthan has the ability to apply different conflict resolution approaches in the Conflict Resolver, depending on its modular structure. Equation (3) is an example for a reasonable form for the type of a weight function that can be used to determine an agent's weight:

$$W_i = 2v_i^3 + x_i + y_i^2 + 4c_i^2 \qquad (3)$$

where v is the priority level of inhabitant i in a certain zone, x is the average duration stayed in minutes per day, y is the number of entries per day, and c is the closeness of preference value to median of preferences. For example, if it is predicted that inhabitant7's next zone will be *Z*, where his assigned priority level in zone *Z* is 3, the agent deduces his average duration stayed at that zone is 240 minutes a day, and the average number of entries is 5 times. Then, the agent's weight equation would be as in (4):

$$f(c_{7Z}) = 2(3)^3 + 240 + (5)^2 + 4c_{7Z}^2 = 319 + 4c_{7Z}^2 \qquad (4)$$

Using the algorithm described in Ref. [2], this partially solved equation can be encrypted as follows:

Given M_1=319, M_2=1, M_3=4; p=47139889, L=7, g=3, x=1003, k=1002, r = 16384, r1 = 56, r2 = 62, r3 = 54, R_1= 16440, R_2=16446, R_3=16438;
$y = g^x \bmod p = 3^{1003} \bmod 47139889 = 19477797$
$a = g^k \bmod p = 3^{1002} \bmod 47139889 = 6492599$
$b_1 = y^k M_1 R_1 \bmod p = 43604023$
$b_2 = y^k M_2 R_2 \bmod p = 45520222$
$b_3 = y^k M_3 R_3 \bmod p = 38600605$

Therefore $b(c_{7Z}) = 43604023 + 45520222\, c_{7Z} + 38600605\, c_{7Z}^2$ is the encrypted form of $f(c_{7Z})$, which is sent to the smart devices as: $E(a, b(c_{7Z})) = (6492599, 43604023+45520222c_{7Z} +38600605c_{7Z}^2)$

J. Runs Analyzer

For each simulation run, system logs with a complete analysis for event interactions and ActorNet coordination should be maintained by Bosthan. This allows seamless generation of the simulation's statistics that can be used to study the experimental results based on the defined parameters under investigation.

VI. EXPERIMENTAL STUDY

We are developing Bosthan simulator to evaluate our proposed framework for resource management in multi-inhabitant smart space based on mobile agents' paradigm. Thus, the main questions that we will focus on to answer in this research can be summarized as follows:

- How can we effectively have lightweight adaptive mobile agents that can manage resources of smart devices, interact with other mobile agents, and resolve conflicts?
- How to ensure privacy and untraceability in smart spaces?
- What is the cost of applying privacy on smart spaces' performance?

A. Issues to Investigate

Bosthan will support us to perform an experimental study that investigates:

- The minimum size of a mobile agent that can perform its assigned task and estimate the power it consumes, since it should work in a limited computing environment. Bosthan measures size and consumed power in terms of number of instructions per agent, number of messages between agents, and time of computations and communications between agents.
- The average device response time using ActorNet multi-agents platform by measuring time of computations, communications, and idle time.
- Encryption overhead on the system by measuring agent size and device response time before versus after applying encryption.
- Conflict resolution overhead on the system on device response time by measuring device response time before versus after applying conflict resolution in case of (1) public space (2) private space in best case; all residents are of different priority levels (3) private space in worst case; all residents are of the same priority level.
- The optimum public/private threshold value that minimizes private conflict resolution overhead in comparison to public conflict resolution overhead.
- Inhabitants' patterns of motion effect on mobile agents' functionality by measuring the effect of speed of motion, average duration stayed, and average number of entries of each zone on migration speed of agents and switching control time for devices.
- The average prediction time versus average migration time of mobile agents in order to permit pro-activity using mobile agents.
- Inaccurate prediction overhead on mobile agents' interactions by measuring time taken to discover wrong prediction and redirecting of agents.

VII. CONCLUSION

In this paper, we presented "Bosthan", a modular multi-agent simulation tool for smart space simulations. Bosthan deploys ActorNet mobile agent platform for resources management. Inhabitants' anonymity and untraceability is preserved using EEF technique, while conflict resolution is handled using voting as a single peak function and an encrypted weight function for public and private spaces respectively. Techniques used for learning, security, and conflict resolution can be easily replaced for different simulated scenarios. We use Bosthan to study how our proposed solution affects the performance and efficiency of smart computing environments without revealing their identity.

ACKNOWLEDGMENT

This work is funded by the Egyptian Government in a Joint Supervision Grant between Ain Shams University and University of Illinois at Urbana-Champaign.

REFERENCES

[1] D. Cook, "Multi-agent smart environments," *Journal of Ambient Intelligence and Smart Environments*, Vol. 1, No. 1, pp. 47–51, 2009.

[2] L. Chen, Y. Xu, W. Fang, and C. Gao, "A New ElGamal-based Algebraic Homomorphism and Its Applications," *Proceedings of International Colloquium on Computing, Communication, Control, and Management*, Vol. 1, Issue 3-4, pp. 643–648, 2008.

[3] N. Saxena, A. Roy, and J. Shin, "CHASE: Context-Aware Heterogeneous Adaptive Smart Environments Using Optimal Tracking for Resident's Comfort," *J. Indulska et al. (eds.), Ubiquitous Intelligence and Computing, Vol. 4611 in LNCS*, Springer-Verlag, Berlin Heidelberg, pp. 133–142, 2007.

[4] R. Leszczyna, "Anonymity Architecture for Mobile Agent Systems," *Proceedings of the 3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems for Manufacturing, Artificial Intelligence, Vol. 4659 in LNCS*, Springer-Verlag, Germany, pp. 93–103, 2007.

[5] S. Das and D. Cook, "Designing and modeling smart environments," *Proceedings of the Workshop on Autonomic Computing and Communications*, 2006.

[6] D. Cook, G.M. Youngblood, and S.K. Das, "A multi-agent approach to controlling a smart environment," *AI and Smart Homes*, Springer-Verlag, pp. 165–182, 2006.

[7] Y. Kwon, S. Sundresh, K. Mechitov, and G. Agha, "ActorNet: An Actor Platform for Wireless Sensor Networks," *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1297–1300, 2006.

[8] N. Shirehjini, A.A., F. Klar, "3DSim: Rapid Prototyping Ambient Intelligence," *Joint sOc-EUSAI conference*, 2005.

[9] J.A. Foss, S.S. Harrison, and L. Hyungjick, "The use of encrypted functions for mobile agent security," *Proceedings of the 37th Hawaii International Conference on System Sciences, (HICSS)* Track 9 - Vol. 9, IEEE Computer Society, Washington, DC, USA, pp. 297–306. 2004.

[10] C. S. Design, "Ubiwise, a simulator for ubiquitous," 2003. URL citeseer.ist.psu.edu/701349.html

[11] G. Dewsbury, B. Taylor, and M. Edge, "Designing Safe Smart Home Systems for Vulnerable People," *in R. Proctor and M. Rouncefield (eds.), Dependability in Healthcare Informatics, Proceedings of the First Dependability IRC Workshop*, Edinburgh, 2001.

[12] W. Jansen, "Countermeasures for Mobile Agent Security," *Computer Communications*, Vol. 23, No. 17, Elsevier Press, 2000.

[13] V. Lesser, et al., "The Intelligent Home Testbed," *Autonomy Control Software Workshop*, 1999.