Trust-based Service Management of Internet of Things Systems and Its Applications

Jia Guo

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Computer Science and Applications

Ing-Ray Chen
Konstantinos P. Triantis
Jeffrey J.P. Tsai
Chandan Reddy
Narendran Ramakrishnan

March 12th, 2018
Falls Church, Virginia

Keywords: Trust management, Internet of Things (IoT) systems, mobile cloud computing, service management, security, scalability, performance analysis.

Trust-based Service Management of Internet of Things Systems and Its Applications

Jia Guo

Abstract

A future Internet of Things (IoT) system will consist of a huge quantity of heterogeneous IoT devices, each capable of providing services upon request. It is of utmost importance for an IoT device to know if another IoT service is trustworthy when requesting it to provide a service. In this dissertation research, we develop trust-based service management techniques applicable to distributed, centralized, and hybrid IoT environments.

For distributed IoT systems, we develop a trust protocol called Adaptive IoT Trust. The novelty lies in the use of distributed collaborating filtering to select trust feedback from owners of IoT nodes sharing similar social interests. We develop a novel adaptive filtering technique to adjust trust protocol parameters dynamically to minimize trust estimation bias and maximize application performance. Our adaptive IoT trust protocol is scalable to large IoT systems in terms of storage and computational costs. We perform a comparative analysis of our adaptive IoT trust protocol against contemporary IoT trust protocols to demonstrate the effectiveness of our adaptive IoT trust protocol. For centralized or hybrid cloud-based IoT systems, we propose the notion of Trust as a Service (TaaS), allowing an IoT device to query the service trustworthiness of another IoT device and also report its service experiences to the cloud. TaaS preserves the notion that trust is subjective despite the fact that trust computation is performed by the cloud. We use social similarity for filtering recommendations and dynamic weighted sum to combine self-observations and recommendations to minimize trust bias and convergence time against opportunistic service and false recommendation attacks. For large-scale IoT cloud systems, we develop a scalable trust management protocol called IoT-TaaS to realize TaaS. For hybrid IoT systems, we develop a new 3-layer hierarchical cloud structure for integrated mobility, service, and trust management. This architecture supports scalability, reconfigurability, fault tolerance, and resiliency against cloud node failure and network disconnection. We develop a trust protocol called IoT-HiTrust leveraging this 3-layer hierarchical structure to realize TaaS.

We validate our trust-based IoT service management techniques developed with real-world IoT applications, including smart city air pollution detection, augmented map travel assistance, and travel planning, and demonstrate that our

trust-based IoT service management techniques outperform contemporary non-trusted and trust-based IoT service management solutions.

Trust-based Service Management of Internet of Things Systems and Its Applications

Jia Guo

General Audience Abstract

A future Internet of Things (IoT) system will consist of a huge quantity of heterogeneous IoT devices, each capable of providing services upon request. It is of utmost importance for an IoT device to know if another IoT service is trustworthy when requesting it to provide a service. In this dissertation research, we develop trust-based service management techniques applicable to distributed, centralized, and hybrid IoT environments.

We have developed a distributed trust protocol called Adaptive IoT Trust for distributed IoT applications, a centralized trust protocol called IoT-TaaS for centralized IoT applications with cloud access, and a hierarchical trust management protocol called IoT-HiTrust for hybrid IoT applications. We have verified that desirable properties, including solution quality, accuracy, convergence, resiliency, and scalability have been achieved.

Furthermore, we validate our trust-based IoT service management techniques developed with real-world IoT applications, including smart city air pollution detection, augmented map travel assistance, and travel planning, and demonstrate that our trust-based IoT service management techniques outperform contemporary non-trusted and trust-based IoT service management solutions.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Trust Management in Internet of Things

A future Internet of Things (IoT) system connects the physical world into cyberspace via radio frequency identification (RFID) tags, sensors, and smart objects owned by human beings [1, 10]. Physical objects can be equipped with RFID tags and electronically identifiable and tractable. Devices with sensing capability provide environmental information, body conditions, etc., which are remotely accessible. Smart objects like smart phones and consumer electronics with ample computing resources share information and provide billions of new services connecting everyone with everything.

In Service-Oriented Architecture (SOA) based IoT systems [54], each device is a service consumer and if desirable can be a service provider offering services or share resources and interacts with service consumers via compatible service APIs. SOA technologies (such as WS-*, REST, and CoRE) enable publishing, discovery, selection, and composition of services offered by IoT devices. The important application scenarios proposed for SOA-based IoT systems include e-health (continuous care) [17][63], smart product management, smart events for emergency management [89], etc.

The motivation of providing a trust system for an SOA-based IoT system is easy to see. There are misbehaving owners and consequently misbehaving devices that for self-interest may perform "discriminatory" attacks to ruin the reputation of other IoT devices which provide similar services. Furthermore, users of IoT devices are likely to be socially connected via social networks like Facebook, Twitter, Google+, etc. Therefore, misbehaving nodes with close social ties can collude and monopoly a class of services.

SOA-based IoT systems challenge trust management in the following aspects. First, an IoT system has a huge amount of heterogeneous entities with limited capacity. Existing trust management protocols do not scale well to accommodate this requirement because of the limited storage space and computation resources. Second, a SOA-based IoT system evolves with new nodes joining and existing nodes leaving. A trust management protocol must address this issue to allow newly joining nodes to build up trust quickly with a reasonable degree of accuracy [76]. Third, IoT devices are

mostly human carried or human operated [8]. Trust management must take into account social relationships among device owners in order to maximize protocol performance. Lastly and arguably most importantly, a SOA-based IoT system essentially consists of a large number of heterogeneous IoT devices providing a wide variety of services. Many of them (the owners) will be malicious for their own gain so they will perform attacks for self-interest. Many of them with close social ties will collude to ruin the reputation of other devices which provide similar services via bad-mouthing attacks, and conversely boost the reputation of each other via ballot-stuffing attacks. A trust management protocol for SOA-based IoT must be resilient to such attacks to be sustainable.

Despite the abundance of trust protocols for P2P and ad hoc sensor networks [22][24][52][44][47][48][49][50][67][97][100][101][102], there is little work on trust management for IoT systems [9][10][11][19][103]. We will survey related work in Chapter 2 and compare as well as contrast our approach with existing work. The problem we aim to solve is design and validation of a scalable and adaptive trust management protocol for SOA-based social IoT systems capable of answering the challenges discussed above. Our proposed IoT trust management protocols will be executed autonomously by IoT devices with little human intervention. The main idea is to combine peer evaluation with trust evaluation in IoT systems. The goals are two-fold: (a) trust bias minimization; (b) application performance optimization. This is achieved by adaptive trust management, i.e., adjusting trust protocol settings in response to environment changes dynamically.

A future Internet of Things (IoT) system will consist of tens of thousands, millions, or even billions of heterogeneous IoT devices (such as RFID tags, sensors, and smart objects owned by human beings), each capable of providing services upon request. It is of utmost importance for an IoT device to know if another IoT service is trustworthy when requesting it to provide a service. The motivation of providing a trust system for an IoT system is easy to see. There are misbehaving owners and consequently misbehaving devices that for self-interest may perform "discriminatory" attacks to ruin the reputation of other IoT devices which provide similar services. Furthermore, users of IoT devices are likely to be socially connected via social networks like Facebook, Twitter, Google+, etc. Therefore, misbehaving nodes with close social ties can collude and monopoly a class of services.

We survey existing IoT trust protocols to date in Chapter 2. In the literature, trust-based service management protocols for IoT systems can be categorized into distributed [9][10][11][23][27][67][94][102] and centralized (cloud-based) [19][86][87][91]. The basic issue of distributed trust-based service management protocols is scalability, i.e., an IoT device's communication and storage cost cannot scale with a large number of IoT

devices in the system. The basic issue of centralized trust-based service management protocols is that it is difficult if not impossible to maintain a consistent global view of social and interaction relationships for every pair of IoT devices dynamically in a large and rapidly changing IoT system. An inconsistent view of social and interaction relationships among IoT devices will make the trust prediction inaccurate and render trust-based service management ineffective. Since the cloud cannot physically collect social and interaction relationships itself, it needs to collect such information from individual IoT devices dynamically. The amount of traffic generated by a large number of IoT devices simultaneously to the cloud will not only consume IoT energy but also cripple the cloud communication network. Our research motivation is to address the scalability issues in existing distributed and centralized trust-based IoT service management protocols, without compromising desirable trust accuracy, convergence, and resiliency properties.

In our work, we propose scalable trust management protocols for supporting trust as a service (TaaS) in a large-scale IoT system. TaaS is made possible by a group of public/private cloud servers which are pre-trusted and whose size scales linearly with the number of IoT devices in the system. In our trust management protocol design, each user is mapped to a "home" cloud server using its unique id based on DHT techniques to achieve load balancing among all cloud servers. TaaS is further realized by following a simple report-and-query paradigm. Specifically, to know if an IoT device is trustworthy in providing a service, a user simply sends a query to its home cloud server who replies with a trust value in the range of 0 to 1 indicating the trustworthiness of the IoT device in providing the service requested. On the other hand, to contribute to accurate trust assessment, a user reports service quality experiences to its home cloud server reactively or proactively.

1.2 Research Objectives

The research objectives of this dissertation research are as follows:

1. Design and develop general trust management techniques applicable to distributed, centralized, and hybrid IoT environments. A distributed IoT system is typified by smart city IoT applications where IoT devices do not necessarily have access to a centralized trusted entity such as the cloud. A centralized IoT system is typified by cross-state and cross-country IoT applications where IoT devices have access to centralized trusted clouds. A hybrid IoT system is the mix of distributed and centralized IoT systems. The trust management techniques developed aim to achieve trust system desirable properties including solution quality, accuracy,

convergence, resiliency, and scalability in the presence of malicious nodes performing trust-related attacks.

2. Design trust protocols utilizing general trust management techniques for distributed, centralized, and hybrid IoT applications. Verify desirable properties including solution quality, accuracy, convergence, resiliency, and scalability, are achieved. Prove the designed protocols outperform contemporary centralized and distributed trust protocols.
3. Prove the validity of our trust protocols with real-world IoT applications running in distributed, centralized, and hybrid IoT environments.

1.3 Research Contributions

The contributions of this dissertation research are as follows:

1. We develop general solution techniques applicable to distributed, centralized and hybrid IoT environments to achieve trust system desirable properties including solution quality, accuracy, convergence, resiliency, and scalability in the presence of malicious nodes performing trust-related attacks. As illustrated in Figure 1.1, the design space includes (a) social similarity based collaborative filtering; (b) adaptive filtering; (c) multi-trust with dynamic weighted sum for trust formation; (d) cloud based service management; and (e) hierarchical trust management. The left hand side of Figure 1.1 illustrates the desirable system properties to which these techniques contribute.

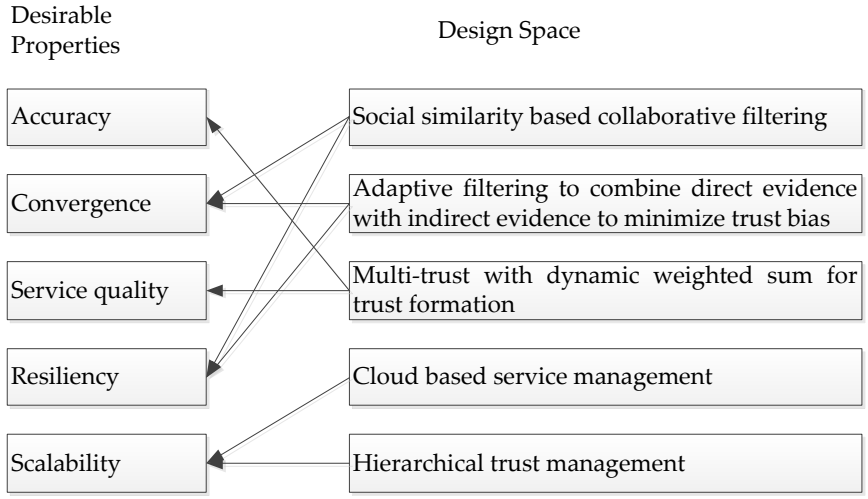


Figure 1.1: Desirable system properties and design space.

2. We develop a trust protocol called Adaptive IoT Trust [27] for distributed IoT systems. The novelty lies in the use of distributed collaborating filtering [62] to select trust feedback from owners of IoT nodes sharing similar social interests. We develop a novel adaptive filtering technique to adjust trust protocol parameters dynamically to minimize trust estimation bias and maximize application performance. Our adaptive IoT trust protocol is scalable to large IoT systems in terms of storage and computational costs. We perform a comparative analysis of our adaptive IoT trust protocol against four prevalent trust protocols, namely, ServiceTrust[94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] in trust convergence, accuracy and resiliency properties achieved to demonstrate the effectiveness of our adaptive IoT trust protocol. We also develop a dynamic weighted sum technique to dynamically adjust the weights associated with social similarity metrics to minimize trust bias and maximize the performance of trust-based service decomposition applications in terms of the solution quality of the composite service formulated. We validate the design by three real-world IoT applications, smart city air pollution detection, augmented map travel assistance, and travel planning.
3. We propose the notion of Trust as a Service (TaaS) for large-scale cloud based centralized IoT systems. We develop a scalable trust management protocol called IoT-TaaS to realize TaaS. Our protocol merges trust management and cloud computing techniques into one for scalability. IoT-TaaS preserves the notion that trust is subjective despite the fact that trust computation is performed by centralized cloud servers. That is, when a user queries the trust value of an IoT device through TaaS, it would receive a subjective trust value, as if the trust computation was performed by the user itself using self-observations (which is subjective) and recommendations (which is subjective and can be even malicious) toward the IoT device. This is different from ObjectiveTrust [86] which only computes the reputation (i.e. common belief) of an IoT device. To achieve true TaaS for IoT systems, we consider not only service quality but also social relationships for evaluating trust of IoT devices, as IoT devices are owned by humans who inherently have social relationship. We use social similarity for filtering recommendations and dynamic weighted sum to combine self-observations and recommendations to minimize trust bias and convergence time against opportunistic service and false recommendation attacks. We validate IoT-TaaS by comparing its performance against contemporary distributed Adaptive IoT Trust [27] (which outperforms ServiceTrust[94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86]) and centralized ObjectiveTrust [86] (which is the only centralized IoT trust management protocol to-date that considers social

standing and relationships for credibility rating and recommendation filtering) and demonstrate IoT-TaaS achieves scalability without compromising trust accuracy.

4. For hybrid IoT systems, we propose a hierarchical cloud structure for integrated mobility, service, and trust management. This architecture supports scalability, reconfigurability, fault tolerance, and resiliency against cloud node failure and network disconnection, and can benefit both network operators and cloud service providers. Applying to mobile cloud environments, we design and validate a trust protocol called IoT-HiTrust levering a 3-layer hierarchical structure with upper-level nodes as clouds providing cloud-based centralized trust service, middle-level nodes as IoT devices with cloud access, and low-level nodes as IoT nodes without cloud access. The middle-level nodes serve as cloud surrogates and act as "cloudlets" accessible to low-level IoT devices. We demonstrate the feasibility of the proposed hierarchical cloud structure with air-pollution detection and service composition IoT applications. IoT-HiTrust is a scalable trust protocol for mobile cloud IoT by means of a new "report-and-query" design in our proposed 3-tier cloud-cloudlet-device hierarchy (see Chapter 6 for detail) that allows an IoT device to report its service experience and social relationship with another IoT device dynamically and query the service trustworthiness of another IoT device locally through the local cloudlet to the cloud and the cost remains more or less constant. It achieves scalability because an IoT device's communication and storage cost will remain the same, regardless of the number of IoT devices in the system.

1.4 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we perform a literature survey for related work in the area of trust management for IoT systems. We classify existing approaches, position our work in the classification hierarchy, and discuss our unique contributions from existing work. In Chapter 3, we discuss the system and threat assumptions of IoT systems based on which our trust management protocols are designed. In Chapter 4, we develop a trust protocol called Adaptive IoT Trust for distributed IoT systems. In Chapter 5, we develop a scalable trust management protocol called IoT-TaaS for large-scale cloud based centralized IoT systems. In Chapter 6, we develop a hierarchical cloud structure for integrated mobility, service, and trust management of hybrid IoT systems, and develop a trust protocol called IoT-HiTrust for mobile cloud environments. Finally in Chapter 7, we discuss work completed, future work, and milestones for the dissertation research.

Chapter 2 Related Work

2.1 Classification Tree

In this section, we develop a classification tree for classifying trust computation techniques for IoT systems. The intent is to identify research gaps in IoT trust computation research. In this section we only classify existing IoT trust computation models. The detailed descriptions of existing IoT trust computation models for IoT systems will be given further in Section 2.2.

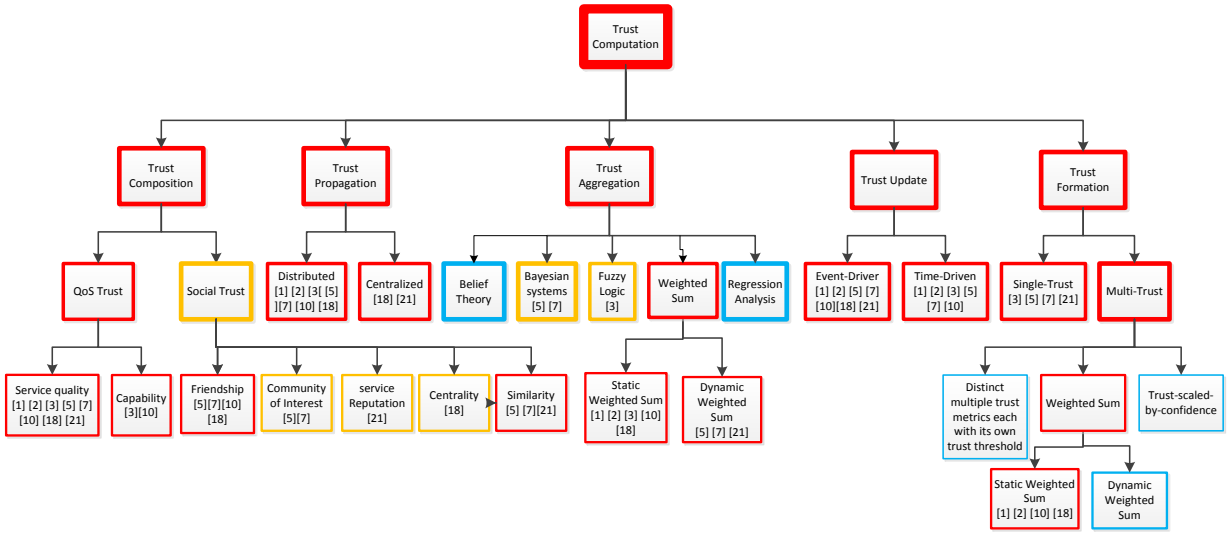


Figure 2.1: Classification Tree.

Figure 2.1 shows the classification tree based on five design dimensions: trust composition, trust propagation, trust aggregation, trust update, and trust formation. These 5 dimensions are considered essential for a trust computation model [45]. It is color coded with red indicating the most visited, yellow for least visited, and blue for little visited. In each color class, we use thick vs. thin line format to differentiate the amount of exposure. Also at the bottom leaf level, we put in references for works that have used the approach. Below we discuss each classification design dimension in detail.

2.1.1 Trust Composition

Trust composition refers to what components to consider in trust computation. Trust components include quality of service (QoS) trust and social trust.

2.1.1.1 **QoS Trust**

QoS trust refers to the belief that an IoT device is able to provide quality service in response to a service request. QoS trust in general refers to performance and is measured by competence, cooperativeness, reliability, task completion capability, etc. [86] used transaction performance to measure QoS trust. [19] used end-to-end packet forwarding ratio, energy consumption, and packet delivery ratio to measure QoS trust.

2.1.1.2 **Social Trust**

Social trust derives from social relationship between owners of IoT devices and is measured by intimacy, honesty, privacy, centrality, and connectivity. [27] made use of friendship, social contact, and community of interest (CoI) to rate a rater. [28] measured social trust by connectivity, intimacy, honesty and unselfishness. Social trust is especially prevalent in social IoT systems where IoT devices must be evaluated not only based on QoS trust, i.e., a device's capability to execute a service request, but also based on social trust, i.e., s device's commitment and good will to perform a service request. Moreover, when taking in a recommendation, an IoT device may trust its socially connected devices (of their owners) over unrelated devices.

2.1.2 Trust Propagation

Trust propagation refers to how to propagate trust evidence to peers. In general, there are two trust propagation schemes, that is, distributed and centralized.

2.1.2.1 **Distributed**

Distributed trust propagation refers to IoT devices autonomously propagating trust observations to other IoT devices they encounter or interact with without the use of a centralized entity. This is particularly the case in which it is difficult to setup or access a centralized entity in IoT environments mimicking a mobile ad hoc network (MANET) and/or a wireless sensor network (WSN). [27] proposed a distributed trust propagation scheme for social IoT systems. [86] proposed a distributed trust system called SubjectiveTrust by which IoT devices each store trust values of other IoT devices in the network. In [19], each node in the network maintains a data forwarding information table by overhearing activities of its neighboring nodes.

2.1.2.2 **Centralized**

Centralized trust propagation requires the presence of a centralized entity, either a physical cloud or a virtual trust service implemented by participating IoT devices. [86] proposed a centralized trust protocol called ObjectiveTrust that leverages a distributed hash table structure to store node trust feedbacks and answer queries for node trust. [91] proposed a centralized trust manager keeping trust information of IoT entities and selecting capable IoT devices for answering a service request.

2.1.3 Trust Aggregation

Trust aggregation refers to aggregating trust evidence collected through either self-observations or feedbacks from peers. Major trust aggregation techniques investigated in the literature [66] include weighted sum, belief theory, Bayesian inference (with belief discounting), fuzzy logic, and regression analysis.

2.1.3.1 **Weighted Sum**

Weighted sum is a popular technique to aggregate evidence. Many reputation systems aggregate ratings or feedbacks using weighted sum such that raters with a higher reputation or transaction relevance have a higher weight. [86] used *credibility* (derived from QoS and social trust) as the weight associated with the recommendation or feedback provided by a rater for indirect trust aggregation. [27] also used similarity (derived from social trust) as the weight for indirect trust aggregation. Weighted sum can also be used to aggregate direct trust (through self-observations) with indirect trust (through feedbacks or recommendations) for the same trust property (e.g., service quality). There is a further classification of whether the weights assigned to direct trust and indirect trust can be dynamically adjusted or just static at design time.

2.1.3.2 **Belief Theory**

Belief theory, also known as evidence theory or Dempster–Shafer theory (DST), is a general framework for reasoning with uncertainty, with connections to other frameworks such as probability, possibility and imprecise probability theories. [64] proposed a subjective logic which operates on subjective beliefs about the world, and used opinion metric to denote the representation of a subjective belief. [106] adopted Dempster-Shafer Theory as the underlying trust computational model to compute trust of agents in autonomous systems. The basic idea is model trust by belief, disbelief and uncertainty. A node's opinion in another node is denoted by (b, d, u, a) [66] where b, d, and u represent belief, disbelief, and uncertainty, respectively, with b+d+u=1, and a is the base rate probability in the absence of evidence. The average trust is therefore the probability expectation value computed as b+au. Subjective logic operators such as the

discount and consensus operators can be used to combine opinions (self-observations or recommendations) [64].

2.1.3.3 **Bayesian inference with Belief Discounting**

Bayesian inference treats trust as a random variable following a probability distribution with its model parameters being updated upon new observations. It is a popular trust computational model because of its simplicity and sound statistical basis. [65] proposed a Beta reputation system based on Bayesian inference with the trust value modeled as a random variable in the range of [0, 1] following Beta distribution; the amounts of positive and negative experiences are mapped to the (α, β) parameters in Beta distribution so that the average trust is computed as $\frac{\alpha}{\alpha+\beta}$. [52] applied Bayesian inference for a reputation system in a WSN, taking binary positive and negative ratings as input, and computing sensor node reputation scores. Belief discounting is applied to defend against bad-mouthing attacks (saying a good node as a bad node) and ballot-stuffing attacks (saying a bad node as a good node). Specifically [52][65], let node *i* be the trustor, node *j* be the trustee, and node k be a recommender. Also let $(\alpha_{i,j}, \beta_{i,j})$ be the trustor's (α, β) toward the trustee, $(\alpha_{k,j}, \beta_{k,j})$ be the recommender's (α, β) toward the trustee and $(\alpha_{i,k}, \beta_{i,k})$ be the trustor's (α, β) toward the recommender. Based on belief discounting (see the detail in [52][65]), node i will compute its new $(\alpha_{i,j}^{new}, \beta_{i,j}^{new})$ as follows:

$$\alpha_{i,j}^{new} = \alpha_{i,j} + \frac{2\alpha_{i,k}\alpha_{k,j}}{[(\beta_{i,k}+2)(\alpha_{k,j}+\beta_{k,j}+2)]+2\alpha_{i,k}} \quad (2.1)$$

$$\beta_{i,j}^{new} = \beta_{i,j} + \frac{2\alpha_{i,k}\beta_{k,j}}{[(\beta_{i,k}+2)(\alpha_{k,j}+\beta_{k,j}+2)]+2\alpha_{i,k}} \quad (2.2)$$

The basic idea is that if node i does not trust k, it will discount the recommendation provided by node k, so $\alpha_{i,j}^{new} \sim \alpha_{i,j}$ and $\beta_{i,j}^{new} \sim \beta_{i,j}$ as if the recommendation from k does not have any effect. This can be derived from (2.1) and (2.2). First of all, if node i does not trust node k then $\alpha_{i,k} \ll \beta_{i,k}$. In case node k is performing a bad-mouthing attack on node j, then $\alpha_{k,j} \ll \beta_{k,j}$.Applying these two conditions to (2.1) and (2.2), one can easily verify $\alpha_{i,j}^{new} \sim \alpha_{i,j}$ and $\beta_{i,j}^{new} \sim \beta_{i,j}$. In case node k is performing a ballot-stuffing attack on node j, then $\alpha_{k,j} \gg \beta_{k,j}$ and again one can easily verify $\alpha_{i,j}^{new} \sim \alpha_{i,j}$ and $\beta_{i,j}^{new} \sim \beta_{i,j}$. After trust aggregation, the trustor's (or node i's) trust toward the trustee (or node j) is then computed as $\frac{\alpha_{i,j}^{new}}{\alpha_{i,j}^{new}+\beta_{i,j}^{new}}$.

2.1.3.4 **Fuzzy Logic**

Fuzzy logic is a form of many-value logic; it deals with reasoning that is approximate rather than fixed and exact. Compared to traditional binary sets, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific membership functions. Reputation or trust is represented as a fuzzy measure with membership functions describing the degrees of trust, e.g., a trust value in the range (−1.25, 1.25) denotes very low trust, (0, 2.5) low trust, (1.25, 3.75) medium trust, (2.5, 5) high trust, (3.75, 6.25) high trust, and so on. Hence, a node with a trust value of 0.25 is 75% very low trust (a membership function) and 25% low trust (another membership function). Fuzzy logic provides rules for reasoning with fuzzy measures. [19] uses a fuzzy membership function taking into consideration of the number of positive and negative experiences together with uncertainty to compute trust. More detail is discussed in Section 2.2.4.

2.1.3.5 **Regression Analysis**

Regression analysis is a statistical process for estimating the relationships among variables. It can be applied to estimate the relationships between trust and a set of variables characterizing the behavior of a node. [98] applied logit regression to learn the relation between cumulative evidence gathered by node i toward node j and the corresponding environmental context variables including energy-sensitivity, capability-limitation, and cost-awareness. This behavior pattern is learned dynamically and can be used to predict node j's trust behavior, i.e., whether node j, from the perspective of node i, can provide good service when called for given an environment setting characterized by a set of context variable values as input.

2.1.4 Trust Update

Trust update concerns when trust is updated. In general, there are two schemes, namely, event-driven and time-driven.

2.1.4.1 **Event-driven**

In the event-driven scheme, all trust data in a node get updated after a transaction or event is made. This can be when a service is rendered and therefore a feedback regarding service quality is sent to the trust manager in the cloud, or recorded in each node cache for trust aggregation. A recommendation can also be sent upon request in encounter-based environments [24][30] where nodes run into each other and request for recommendations for other nodes.

2.1.4.2 **Time-driven**

In the time-driven scheme, evidence (self-observations or recommendations) is collected periodically and trust is updated by applying a trust aggregation technique. In case no evidence is collected, trust decay over time is frequently applied because one should trust recent information more than past information. The exponential decay function with a parameter adjusting the rate of trust decay over time can be used depending on the specific application needs [24][30].

2.1.5 Trust Formation

Trust formation refers to how to form the overall trust out of multiple trust properties. In the literature, trust formation is considered from the aspect of single-trust or multi-trust.

2.1.5.1 **Single-trust**

Single-trust refers to the fact that only one trust property is considered in a trust protocol. For example, service quality is deemed the single most important metric in service-oriented IoT systems [27]. Therefore, an IoT device is being evaluated on its ability to produce quality service when called for. In a social IoT system, the service quality may be affected by the relationship between the service requester and the service provider, so inherently trust in service quality in a social IoT system is pair-wise. In other words, a node is more concerned with one-to-one trust toward another node based on their relationship, rather than the general reputation of the node derived from the public belief.

2.1.5.2 **Multi-trust**

Multi-trust implements the common belief that trust is multidimensional, so multiple trust properties should be considered for trust formation. [28] considered multiple trust properties including intimacy, honesty, unselfishness and competence, to assess the overall trust of a MANET node. There are multiple ways to do trust formation:

- One can just use individual trust properties without combining them together but define a minimum threshold for each trust property depending on the application requirements. For example, honesty is important, so a high threshold is used but competence may not be very critical so a low threshold is set.
- One can use *weighted sum* to combine individual trust properties together into an overall trust metric. The weight assigned can reflect the application requirements. For example, honesty is important so a high weight is used. Furthermore, the weight

assignment may be dynamically adjusted to reflect environmental situation awareness. For example in a hostile environment where attacks are likely, the weight associated with honesty can be set high so as to effectively defend malicious attacks (such as bad-mouthing and ballot-stuffing attacks). On the other hand in a friendly environment such as in a club setting, competence is more important than honesty, so a higher weight can be placed for competence instead. [27] proposed to readjust the weights of direct and indirect trust to maximize the average user satisfaction experiences during the most recent time period. [91] proposed to change the weight associated with positive recommendations dynamically to derive the overall trust value. [74] proposed to use a penalty coefficient weight based on the authentication history to update trust.

One can use a *trust-scaled-by-confidence* technique for trust formation. The idea is to scale the most important trust property with less important trust properties which serve as confidence. [96] considered competence and integrity as two trust properties for rating a node with competence being the more important trust metric. It considered two scaling schemes: (a) competence trust drops to zero if integrity trust falls below a threshold; (b) competence trust scales up (to 1 maximum) or down (to 0 minimum), depending on whether integrity trust is higher or lower than the threshold.

2.2 Survey and Classification of Existing IoT Trust Computation Models

IoT trust computation is still in its infancy with limited work reported in the literature to date, possibly due to limited experiences with IoT platforms and experimentations. We only found [9] [23] [19] [11] [27] [40] [86] [91] and [95] in the literature to date. In this section, we follow the classification tree to classify existing IoT trust computation models based on the techniques used in trust composition / trust propagation / trust aggregation / trust update / trust formation. Based on the classification tree, we identify 5 classes as summarized in Table 2.1. It is noteworthy that the classification is based on the underlying trust techniques adopted in the five design dimensions. So works will fall into the same class only if they use identical trust techniques in all five design dimensions. A deviation of trust techniques used in just one design dimension will put the works in separate classes. By this way, we can compare trust computation models class by class and identify the most effective class for trust computation for service management (whether to select a device as a service provider) in service-oriented IoT systems.

For works that fall under the same class, we also survey defense mechanisms used (if any) to defend against malicious attacks discussed in Section 3.3.

We discuss these five classes identified in the 5 subsections below, with the subsection title reflecting the classification based on the techniques used in trust composition / trust propagation / trust aggregation / trust update / trust formation.

2.2.1 Class 1: QoS + Social / Distributed / Bayesian inference + Dynamic weighted sum / Event + Time-driven / Single-trust

Among all works listed in Table 2.1, only [11] [27] fall into this classification with [11] being the preliminary work of [27].

[11] [27] use service quality (a QoS trust metric) to rate a SP, and social similarity (a social trust metric) to rate a recommender based on the concept of collaborative filtering to select feedbacks using similarity rating of friendship, social contact, and community of interest relationships as the filter.

In trust propagation, every node acts autonomously to collect evidence (through self-observations or recommendations) and also serves as a recommender upon request. Hence it is based on distributed trust propagation. A node first collects evidence of the service quality trust and social similarity trust of adjacent nodes. Then it collects recommendations from qualified adjacent nodes about other nodes in the system.

In trust aggregation, [11] [27] use Bayesian inference to aggregate self-observations into direct trust. Social similarity-weighted sum is used to aggregate recommendations into indirect trust. A novel adaptive filtering technique is proposed to adjust weights associated with direct trust and indirect trust dynamically to minimize trust bias and maximize application performance.

In trust update, both event-driven and time-driven are considered. The direct trust is updated upon each service interaction while the indirect trust is updated periodically using peer recommendations collected during the period. The work also considers and analyzes the effect of trust decay on trust convergence rate.

Table 2.1: Five Classes of IoT Trust Computation Models based on the Techniques used in Trust Composition / Trust Propagation / Trust Aggregation / Trust Update / Trust Formation.

Classification	Work	SPA	BMA	BSA	OSA	OOA
QoS + Social / Distributed / Bayesian inference + Dynamic weighted sum / Event + Time-driven / Single-trust (Section 2.2.1)	2013 Bao, et al. [11], and 2016 Chen, et al. [27]	Direct service quality trust assessment and feedback propagation	Social similarity to rate a recommender	Social similarity to rate a recommender	Adaptive filtering to adjust the weights of direct and indirect service quality trust dynamically	NA

QoS + Social / Distributed + Centralized / Static weighted sum / Event-driven / Multi-trust with static weighted sum (Section 2.2.2)	2014 Nitti, et al. [86]	Direct service quality trust assessment and feedback propagation	Credibility to rate a recommender	Credibility to rate a recommender	Long-term and short-term direct service quality trust assessment	NA
QoS / Centralized / dynamic weighted sum / Event-driven / Single-trust (Section 2.2.3)	2014 Saied, et al. [91]	Direct service quality trust assessment and feedback propagation	Recommender trust to rate a recommender	Recommender trust to rate a recommender	NA	NA
QoS / Distributed / Fuzzy logic + Static-weighted sum / Time-driven / Single-trust with static weighted sum (Section 2.2.4)	2011 Chen, et al. [19]	Direct service quality trust assessment and feedback propagation	NA	NA	NA	NA
QoS + Social / Distributed / Static weighted sum / Event + Time-driven / Multi-trust with static weighted sum (Section 2.2.5)	2012, Bao, et al.[9] , and 2016 Chen, et al. [23]	Honesty trust assessment and feedback propagation	Honesty trust assessment and feedback propagation	Honesty trust assessment and feedback propagation	NA	NA
	2016 Chen, et al. [40]	Direct service quality trust assessment and feedback propagation	Recommender trust to rate a recommender	Recommender trust to rate a recommender	NA	NA

In trust formation, only a single service quality trust is considered, so it falls into the single-trust category. However, [11] [27] use several social similarity metrics, i.e., friendship, social contact, and community-of-interest, and apply the weighted sum technique to combine these social similarity metrics into one to rate a recommender. The best weighting scheme to combine the three metrics into one overall metric is identified for a service-oriented IoT application.

Entry 1 of Table 2.1 summarizes the defense mechanisms used by [11] [27] to defend against malicious attacks. SPA is detected in the protocol design and the feedback is propagated through trust propagation. BMA and BSA are tolerated by using social similarity to rate a recommender. OSA is resolved by adaptive filtering which dynamically adjust the weights associated with direct and indirect trust to capture the opportunistic service attack behavior. However, OOA is not considered.

2.2.2 Class 2: QoS + Social / Distributed + Centralized / Static weighted sum / Event-driven / Multi-trust with static weighted sum

[86] falls into this classification. In trust composition, [86] considers both QoS trust and social trust. QoS trust properties considered include transaction service quality and computational capability. Social trust properties considered include centrality, relationship factor (such as ownership, co-location, co-work, social and co-brand), and credibility. In particular, credibility is used to rate a recommender who provides indirect evidence and is computed by a weighted sum of the direct trust toward the recommender and the relative centrality of the recommender.

In trust propagation, [86] is rather unique in that it considers both distributed and centralized models. In the distributed model (called SubjectiveTrust), each node computes its own *subjective trustworthiness* toward another node. Transaction service quality trust is assessed by individual nodes after a transaction is completed, and feedbacks are propagated as indirect evidence from one node to another upon request. In the centralized model (called ObjectiveTrust), transaction service quality feedbacks are propagated to a centralized entity making use of a dynamic hash (DHT) table structure on the network to maintain the *objective trustworthiness* status (global reputation) of a node. A node can query the DHT to receive the trust value of other nodes in the network, and the DHT returns the objective trustworthiness scores after searching the database.

In trust aggregation, [86] applies *static weighted sum* to compute centrality trust, direct service quality trust, and indirect service quality trust separately. In particular, for direct service quality trust assessment, transaction relevance (along with relationship factor and computational capability to a lesser extent) is used as the weight. For indirect service quality trust assessment, credibility is used as the weight. The difference between subjective trustworthiness and objective trustworthiness is how evidence is collected. For subjective trustworthiness, a node uses the relative centrality for centrality trust assessment, self-observations for direct service quality trust assessment, and feedbacks provided to its peers for indirect service quality trust assessment of another node. For objective trustworthiness, the network centrality is used for centrality trust assessment, and all feedbacks are used for both direct and indirect service quality trust assessment.

In trust update, the event-driven scheme is taken. For the distributed model, a service receiver rates the service quality of a transaction at the end of the transaction, stores the rating its local storage, and provides a feedback to its peers upon request. For the centralized model, the DHT collects the feedback and updates its trust database after the end of each transaction.

In trust formation, [86] is considered a multi-trust scheme by applying *static weighted sum* to combine centrality trust (which is a social trust property) with service quality trust (which is a QoS trust property obtained from direct service quality trust and indirect service quality trust) into an overall trust value.

For defending against attacks, [86] uses direct service quality trust assessment and feedback propagation for SPA, credibility rating for BMA and BSA, and differentiating long-term and short-term direct service quality trust assessment to change credibility to defend against OSA. However OOA is not considered.

2.2.3 Class 3: QoS / Centralized / dynamic weighted sum / Event-driven / Single-trust

[91] proposes a reputation system for a service-oriented IoT system and falls into this classification. In trust composition, it considers service quality as the sole trust metric but uses context information such as service type and node capability (e.g., energy status) to associate a service quality rating. In trust propagation, it uses a centralized manager to store all reputation reports (with context information) sent by individual SRs, after service is rendered. Upon receiving a new service request, the centralized manager selects SP candidates based on the service context for servicing the request. It then uses only evaluation reports with similar service context for reputation assessment of each SP candidate. In trust aggregation, the service context similarity between a stored report and the target service is computed by a global contextual distance function. A higher weight is used if a higher service context similarity is found. The reputation score is then computed by *dynamic weighted sum* with the weight associated with a report corresponding to the *recommendation trust* of the recommender who supplies the report. The recommendation trust is updated dynamically based on if the recommender's report agrees or deviates from the majority of reports with a similar service context. Trust update is performed by the centralized manager via a learning process which presumably occurs whenever new reports are received, so it is based on event-driven. In trust formation, only service quality is considered.

[91] deals with SPA by detection of service quality and feedback propagation. The centralized manager rates a recommender dynamically based on the degree to which the recommender's report deviates from the majority reports. This can effectively defend against BMA and BSA, if the majority recommenders are not malicious. OSA and OOA are not considered.

2.2.4 Class 4: QoS / Distributed / Fuzzy logic + Static-weighted sum / Time-driven / Single-trust with static weighted sum

[19] falls into this classification. In trust composition, it considers direct QoS trust metrics such as end-to-end packet forwarding ratio (EPFR), energy consumption (AEC), and packet delivery ratio (PDR). In trust propagation, trust is propagated in a distributed manner. Each node in the network maintains a data forwarding information table of other nodes.

In trust aggregation, the overall trust of a node toward another node is aggregated using a static weighted sum of direct trust based on direct interaction experiences, and indirect trust based on recommendations. The direct trust is computed by aggregating EPFR, AEC and PDR direct interaction evidence using a static-weighted sum. If the aggregated trust value passes a threshold, the experience is a positive experience; otherwise, it is a negative experience. A fuzzy membership function taking into consideration of the number of positive and negative experiences together with uncertainty is used to compute the direct trust. The indirect trust on the other hand is computed by the product of the trustor's direct trust toward the recommender with the recommender's recommendation trust toward the trustee. The recommender's recommendation trust is computed in the same way as direct trust except that positive and negative recommendation experiences, together with uncertainty, are used in the fuzzy membership function definition. Since direct trust and recommendation trust are each defined by a fuzzy membership function, the overall trust can be aggregated by applying fuzzy logic "add" (for adding direct trust with indirect trust based on static weighted sum) and "multiply" (for multiplying direct trust with recommendation trust to obtain indirect trust) operators. In trust update, the local trust is updated periodically, so it is time-driven. In trust formation, only single-trust (service quality trust) is considered. No social trust is considered in this work.

SPA is detected in the protocol design and the feedback is propagated to the central manager through trust propagation. BMA, BSA, OSA and OOA are not considered.

2.2.5 Class 5: QoS + Social / Distributed / Static weighted sum / Event + Time-driven / Multi-trust

[9] [23] fall into this classification. In trust composition, it considers separate trust properties, including QoS trust properties such as honesty and cooperativeness, and social trust such as community-interest.

In trust propagation, [9] [23] follow the distributed scheme where each node maintains its own trust assessment towards other nodes and propagates its recommendation trust toward other nodes.

In trust aggregation, a trustor node aggregates its current trust toward the trustee node with new evidence based on weighted sum. The new evidence can be either direct evidence if the trustor node directly encounters and interacts with the trustee node or indirect evidence if it does not encounter the trustee node but receives a recommendation trust toward the trustee node from a recommender it encounters. In the latter case, the recommendation trust received is discounted by the trustor's direct trust toward the recommender. A novelty is that the weights associated with the past experience and the current evidence can be dynamically adjusted to tradeoff the trust convergence rate and trust fluctuation rate. Although the effect of weight parameters is analyzed, there is no discussion of how the weight parameters can be dynamically adjusted. Therefore they can be classified as static weighted sum at most.

In trust update, the trust management protocol is encounter-based as well as activity based, meaning that the trust value is updated upon an encounter event or an interaction activity. Two nodes encountering each other or involved in a direct interaction activity can directly observe each other and update their trust assessment. They also exchange their trust evaluation results toward other nodes as recommendations.

In trust formation, our work considers multiple trust properties: honesty, cooperativeness and community-interest trust. However, the issue of how to form the overall trust out of these separate trust properties is not discussed.

[9] [23] use honesty trust assessment and feedback propagation for defending against SPA, BMA and BSA. However, OSA and OOA are not considered.

[40] also falls into this classification. In trust composition, it considers both QoS trust (i.e., quality reputation and energy status) and social trust (i.e., social relationship using factors considered in [27] [86]). In trust propagation, our work adopts a distributed scheme where each node maintains its own trust assessment towards other nodes. Evidence for each trust component is propagated separately. In trust aggregation, each trust component is assessed separately based on static weighted sum. In particular, quality reputation is a static weighted sum of direct trust and indirect trust. The indirect trust is also a feedback-trust-weighted sum of feedbacks received from all recommenders with each recommender's feedback trust being updated based on how the recommender's feedback deviates from the average. The social relationship trust and the energy trust are also each assessed by a static weight sum function. In trust update, it is event-driven with the trust value being computed based on transaction completion and timer events. In trust formation, it falls into the multi-trust category with the overall trust being formed from the three trust components, quality reputation, energy, and social relationship, based on static weighted sum.

2.3 Research Gaps

In this section, we identify research gaps in trust computation for IoT systems. We first summarize the most visited, least visited, and little visited trust computation methods, as summarized in Table 2.2. The rationale behind this is to identify the most popular trust composition, trust propagation, trust aggregation, trust update, and trust formation techniques adopted by existing trust computation models, provide reasons for such popularity, and then identify research gaps that deserve more attention for IoT trust computation research.

Trust Composition: A modern IoT system is inherently socially oriented since IoT devices are owned by humans which are connected by social relationships. We see from Table 2.2 that most works indeed consider both QoS trust and social trust for trust composition. We take the view that a valid trust model for IoT must consider social trust metrics because social relationships between human operators control the way an IoT device would behave toward another IoT device. Among social trust properties, similarity and friendship are the most visited among all. The reason is that these two social properties arguably are the most important among all because friendship implies good service, and high social similarity implies high creditability of a recommendation. However, we argue that for certain IoT applications such as smart city air pollution detection and augmented map travel assistance [11], social metrics such as centrality and community of interest especially for rating recommenders, should be further explored to improve trust computation performance, including convergence, accuracy, and resiliency against malicious attacks.

Table 2.2: Most, Least and Little Visited IoT Trust Computation Models in the Literature.

Most visited	Trust composition	QoS trust [9] [23] [19] [11] [27] [40] [86] [91]
		Social trust [9] [23] [11] [27] [40] [86]
	Trust propagation	Distributed [9] [23] [19] [11] [27] [40] [86]
		Centralized [86] [91]
	Trust aggregation	Static weighted sum [9] [23] [19] [40] [86]
		Bayesian inference [11] [27]
	Trust update	Event-driven [9] [23] [11] [27] [40] [86] [91]
		Time-driven [9] [23] [19] [11] [27] [40]
	Trust formation	Single-trust [19] [11] [27] [91]
		Multi-trust with static weighted sum [9] [23] [40] [86]
Least visited	Trust aggregation	Fuzzy logic [19]

		Dynamic weighted sum [11] [27] [91]
Little visited	Trust aggregation	Belief theory
		Regression analysis
	Trust formation	Multi-trust with dynamic weighted sum
		Multi-trust each with its own minimum threshold
		Multi-trust with trust scaled by confidence

Trust Propagation: Table 2.2 shows that existing work mostly considered distributed trust propagation for "subjective" trust computation without relying on a centralized entity. That is, each node propagates trust information to other nodes upon request, and each node also aggregates trust information (including self-observations) for trust assessment toward other nodes in the system. This is feasible for IoT systems where IoT devices (e.g., smart phones, vehicles, etc.) are mobile with no access to a centralized entity such as a cloud. With the emergence of cloud services, however, centralize trust propagation for "objective" reputation computation (common belief of the public) makes more sense. The literature is limited in investigating cloud-based trust propagation methods. Only [86] [91] discussed centralized trust propagation. Centralized trust propagation is implemented in [86] by means of a distributed hash table structure, while [91] simply assumes the existence of a trusted central entity. Neither discusses the potential of integrating cloud computing with centralized trust propagation. The challenge lies in the cloud-based infrastructure design that facilitates trust information propagation from IoT devices to the cloud which aggregates trust feedbacks and answers user queries regarding the service trustworthiness of any particular IoT device in the system. This is a research gap that demands attention.

Trust Aggregation: Static weighted sum and Bayesian inference are the two most visited methods, dynamic weighted sum and fuzzy logic are least visited, while belief theory and regression analysis have not been investigated in the literature. There is no clear reason why fuzzy logic, belief theory, or regression analysis based trust aggregation cannot perform comparably or even better than weighted sum or Bayesian inference based trust aggregation. A comparative analysis is called for to compare these competitive trust aggregation methods for IoT systems. The comparative analysis should consider specific IoT applications and/or specific trust propagation models (distributed vs. centralized). For example, complex statistical analysis is required for regression analysis and hence it is particularly appealing when a centralized cloud is available for cloud-based IoT applications. This is a research gap that deserves attention.

Trust Update: Event-driven trust update is frequently performed when a service or transaction is completed, or a node encountering another node, while time-driven trust update occurs periodically to preserve energy. The hypothesis is that event-driven is more suitable under centralized trust propagation since the centralized entity (i.e., a cloud) is powerful, while time-driven is more suitable under distributed trust propagation since most, if not all, IoT devices are power limited. For the latter case, there is a tradeoff between trust accuracy and energy conservation. Hence, there exists an optimal trust update interval under which an IoT application performance is maximized depending on the trust accuracy and energy consumption requirements which collectively determine the success or failure of the IoT application. These hypotheses remain to be verified or refuted.

Trust Formation: there is a big gap from most visited methods in single-trust and multi-trust with static weighted sum, to little visited methods in multi-trust with dynamic weighted sum, multi-trust each with its own minimum threshold, or multi-trust with trust scaled by confidence. Multi-trust refers to the trust protocol that considers more than one trust properties, each being assessed separately and a trust formation method is applied to form the overall trust out of these multiple trust properties. IoT systems inherently are multi-trust based because both QoS trust (for service quality) and social trust (for social relationship) must be considered, taking into considerations that device-to-device interaction behaviors derive from owner-to-owner relationships. We see from Table 2.2 that only [9] [23] [40] [86] considered multi-trust with static weighted sum. There is a pressing need to investigate more effective multi-trust formation methods and a comprehensive comparative analysis to identify the best trust formation technique to maximize IoT application performance.

2.4 Baseline Trust Protocols for Performance Comparison

Based on our literature survey, we select a set of contemporary IoT trust protocols as baseline IoT trust protocols against which Adaptive IoT Trust, IoT-TaaS, and IoT-HiTrust (our distributed, centralized, and hybrid IoT trust protocols developed in Chapter 4, Chapter 5, and Chapter 6 respectively) will be compared for a comparative performance analysis, as follows:

- Baseline IoT trust protocols for distributed IoT trust management performance comparison: ServiceTrust[94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86].
- Baseline IoT trust protocols for centralized IoT trust management performance comparison: ObjectiveTrust [86].

Below we summarize the design characteristics of these baseline trust protocols and provide the reasons why we select them for performance comparison.

2.4.1 Baseline Trust Protocols for Distributed IoT Trust Management Performance Comparison

EigenTrust [67] is a reputation scheme for P2P/IoT systems. Its basic idea is to aggregate trust recommendations towards a trustee node weighted by the trustor's opinion toward the recommenders. It assumes that there are pre-trusted peers that can provide trusted recommendations to guarantee trust convergence and break up malicious collectives.

PeerTrust [102] is also a reputation system for P2P/IoT systems. Its basic idea is also to aggregate feedbacks weighted by the recommender's trustworthiness. It includes a coherent adaptive trust model for quantifying and comparing the trustworthiness of peers based on a transaction-based feedback system. It introduces three basic trust parameters and two adaptive factors in computing trustworthiness of peers, namely, feedback a peer receives from other peers, the total number of transactions a peer performs, the credibility of the feedback sources, transaction context factor, and the community context factor.

ServiceTrust [94] is a quality sensitive and attack resilient trust management protocol for service provision P2P/IoT networks. First, it encapsulates quality-sensitive feedbacks by multi-scale rating scheme and incorporates the variances of user's behaviors into the local trust algorithm. Second, ServiceTrust measures the similarity of two users' feedback behavior and aggregate the local trust values into the global trust algorithm by exploiting pairwise feedback similarity scores to weight the contributions of local trust values towards the global trust of a participant. Finally, pairwise feedback similarity weighted trust propagation is utilized to further strengthen the robustness of global trust computation against malicious or sparse feedbacks.

SubjectiveTrust [86] is a distributed IoT trust management protocol with each node maintaining its own trust and service experience data. It relies on a friendship social network graph as input to know the "centrality" of a node with respect to another node or to all nodes in the network. SubjectiveTrust is executed by each individual node to assess its "subjective trust" toward a peer IoT device, since each node maintains its own data. A node assesses the trust score of another node through a weighted sum of the following three scores: centrality, own service experiences, and recommendations filtered by credibility. The centrality score (in the range of 0 to 1) of j as evaluated by i is computed by the degree of common friends between i and j. The credibility score (also in the range of 0 to 1) of k (a recommender that provides a recommendation to i about j) as

evaluated by *i* is computed by a weighted sum of own service experiences of *i* toward *k* and the centrality score of *k* as evaluated by *i.* For scalability, each node only stores trust information about its neighbor nodes in the social network graph. When a node wants to know the trust value of a remote node, a search procedure is invoked to first find a path leading to the remote node [19] whose trust value is then computed through the trust chain found.

The reason we consider ServiceTrust[94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] as baseline protocols for performance comparison against our Adaptive IoT Trust protocol (developed in Chapter 5) is that like in our protocol, each IoT device evaluates other IoT devices using both direct service experiences and indirect recommendations. Also, as in our protocol, social relationships are being considered for recommendation filtering and credibility rating.

2.4.2 Baseline Trust Protocols for Centralized or Hierarchically Structured IoT Trust Management Performance Comparison

ObjectiveTrust [86] is a centralized IoT trust protocol. It requires a set of pre-trusted nodes be in place for storing trust information of all nodes in the system. ObjectiveTrust assesses the trust score of a node through a weighted sum of the centrality score and the average opinion score (long term and short term) after applying the recommender's credibility score to filter untrustworthy recommendations. Specifically, ObjectiveTrust computes the centrality score (in the range of 0 to 1) of *j* based on if *j i*s central in the network and if it is involved in many transactions. It computes a recommender's credibility score by taking into consideration of possible collusion attacks. So the credibility score of *k* (a recommender that provides opinions about *i)* is proportional to *k*'s trust score, but reversely proportional to the capability of *k,* the strong object relationship (including ownership, co-location, co-work, social, and parental) between *i* and *k*, and the number of transactions between *i* and *k*. A problem of ObjectiveTrust is that it computes the "objective trust" (common belief or reputation), not the "subjective trust" of an IoT device, so it does not preserve the notion that trust is subjective and is inherently one-to-one. This is especially problematic for IoT systems since IoT devices are owned by humans who have social relationships among themselves and the trust of one user toward another user is inherently one-to-one and subjective.

We select Adaptive IoT Trust [27] and ObjectiveTrust [86] as baseline IoT trust protocols against which IoT-TaaS and IoT-HiTrust (our centralized and hybrid IoT trust protocols developed in Chapter 5 and Chapter 6, respectively) are compared for a comparative performance analysis. The reason we select Adaptive IoT Trust [27] (our distributed IoT trust protocol developed in Chapter 4) is that it is a proven protocol for dis-

tributed IoT trust management and it outperforms existing IoT/P2P trust protocols including ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86]. We want to know if cloud-based IoT trust management can perform better than distributed IoT trust management. The reason we select ObjectiveTrust [86] is that it is the only centralized IoT trust management protocol to date that considers social standing and relationships for credibility rating and recommendation filtering.

2.5 Connection between IoT Trust Management Systems and Recommendation Systems

Our IoT trust management systems have similarities with recommendation systems in that both collect ratings from members in a community and use collaborative filtering to filter recommendations. However, there are two major differences as first observed in [66]. First, a recommendation system takes ratings subject to taste as input, whereas our IoT trust management system takes ratings assumed insensitive to taste as input. The assumption behind a recommendation system is that different people have different tastes and rate things differently according to subjective taste. If two users rate a set of items similarly, they share similar tastes, and this information can be used to recommend items that one user likes to the other user. On the other hand, our IoT trust management system uses social similarity, rather than taste, to rate recommenders because people sharing social similarity can provide more trustworthy recommendations. Second, recommendation systems often assume participants are trustworthy and sincere and opinions are offered based on taste, while our IoT trust management system is designed to sustain malicious recommendation attacks defined in the threat model.

2.6 Differences and Relationships of the Research Ideas or Designs for IoT Trust Management with Existing Ones

The methods of Bayesian framework [65] and social similarity in our designs are not new concepts. They are adopted from other fields to implement our trust management.

We adopt Bayesian framework as the underlying model for evaluating direct trust from direct user satisfaction experiences. The reason we choose Bayesian because it is well-established and because of its popularity in trust/reputation systems. We also adopt the design concept of distributed collaborating filtering to select trust feedback from nodes sharing similar social interests. A node will first measure its "social similarity" with a recommender in friendship, social contact and CoI and then decide if the recommendation is trustable. The reason we consider these metrics is that these metrics are core social metrics for measuring social relationships which are multifaceted. We adopt

cosine similarity to measure the distance of two social relationship lists. Computational efficiency is the main reason why we choose cosine similarity to measure the similarity of two vectors in high-dimensional positive spaces because of limited computational capacity of IoT devices.

Unlike ObjectiveTrust [86] which must rely on the existence of a friendship social network graph as input for specifying social relationships, we collect social relationships between each pair of IoT devices dynamically when IoT devices encounter each other. Our social similarity calculation method is more scalable than ObjectiveTrust [86] as it is difficult if not impossible to construct an accurate friendship social network graph when there is a large number of IoT devices arriving and leaving the system dynamically.

We also introduce a new design concept called *application performance maximization* by which the best weights assigned to the three similarity metrics are identified to optimize application performance, when given a node population characterized by friendship, social connection, and community of interest relationships as input.

2.7 Summary

In this chapter, we developed a classification tree based on five design dimensions considered essential for trust computation, namely, trust composition, trust propagation, trust aggregation, trust update, and trust formation, to classify existing IoT trust computation models. We discussed the pros and cons of existing IoT trust computation models in terms of the class they fall within. In particular for each class of IoT trust computation models, we discussed the defense mechanisms used and their effectiveness against malicious attacks aiming to disrupt the trust system.

Through the pros and cons analysis of IoT trust computation model classes, we identified the most effective trust computation techniques (and thus trust computation classes) as applying to IoT systems. We further identified research gaps in IoT trust computation research for supporting future IoT applications. In view of these research gaps, we discussed deficiencies of existing trust computation research in IoT systems and set the goals (as discussed in Chapter 1) in our dissertation research to address these deficiencies. To achieve these goals, we identified prevalent, contemporary IoT trust protocols as baseline IoT trust protocols against which our distributed, centralized, and hybrid IoT trust protocols will be tested and compared for a meaningful performance comparative analysis to prove their validity.

Chapter 3 System Model

3.1 Social IoT Network Model

We consider a user-centric social IoT [7][8] environment where nodes are physically connected via communication networks and socially connected via social networks (see Figure 3.1). Each node has a unique address to identify (i.e., URI). There is no centralized trusted authority. There are two types of nodes: devices and users (or owners). The user-device relationship is a one-to-multiple relationship. In our trust management, the trustor is a user and the trustee is a device (owned by another user). For each user, the trust evaluation information is computed and stored in a designated high-end device owned by the user.

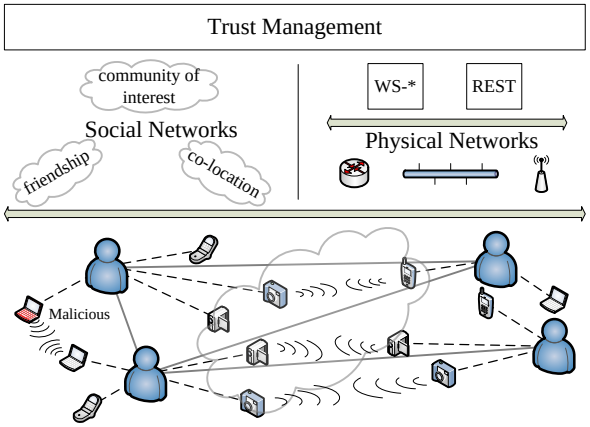


Figure 3.1: User-Centric Internet of Things Systems.

Trust is evaluated based on both direct user satisfaction experiences of past interaction experiences and recommendations from others. In particular, for recommendations from others, we utilize the design concept of distributed collaborating filtering [62][104] to select trust feedback from nodes sharing similar social interests. We consider the following three social relationships: friendship, social contact, and community of interest (CoI). More specifically, we use the social relationships between the trustor and the recommender for the trustor to weigh the recommendation provided by the recommender

toward a trustee. The reason is that two users sharing similar social relationships including friendship (representing intimacy), social contact (representing closeness) and CoI (representing knowledge and standard on the subject matter) are likely to have similar subjective trust views towards services provided by a trustee IoT device. A similar concept to the social contact relationship is proposed in [77], where familiar strangers are identified based on colocation information in urban transport environments for media sharing.

These social IoT relationships are represented by three lists: a friend list (F) with current friends, a location list (S) with locations frequently visited for social contact, and a CoI list (C) with devices (services) directly interacted with. Each user has at least one designated high-end device (i.e., smart phone and laptop) storing these lists in the user's profile (see Figure 3.2). Other devices of the same user have the privilege to access the profile. By delegating the storage and computation of social networks to a high-end device for each user, many low-end devices (i.e., sensors) are able to share and utilize the same social information to maximize its performance. Energy spent for maintaining the lists and executing matching operations is negligible because energy spent for computation is very small compared with that for communication, and matching operations to identity a friend, social contact, or a CoI member are performed only when there is a change to the lists.

The privacy preservation issue which we address in the dissertation research is not about anonymity because we need to know the identity of a user to know whether the user's recommendation is trustworthy. The privacy preservation issue which we address in the dissertation research is about not disclosing private information when two users encounter each other and exchange their (F, S, C) profiles for social similarity computation. Each user maintains its (F, S, C) profile separately. When user u_x encounters user u_y, they exchange their (F_x, S_x, C_x) and (F_y, S_y, C_y) profiles so as to measure their mutual social similarity. To preserve privacy, they only want to reveal common elements in the *F, S,* and *C* lists (If any) but do not want to let the other party know their entire (F, S, C). To achieve this users u_x and $u_y$can first authenticate each other using standard PKI. User u_x can then use a cryptographic hash function in combination with a secret session key *K* (established via PKI during user authentication) to generate a hash-based message authentication code HMAC(*K, p*) for $p \in (F_x, S_x, C_x)$ and then transmit HMAC(*K, p*) along with HMAC(*K,* HMAC(*K, p*)) to u_y. When u_y receives the message, it can unilaterally generate HMAC(*K,* HMAC(*K, p*)) using HMAC(*K, p*) sent by u_x. If this matches with HMAC(*K,* HMAC(*K, p*)) sent by u_x, then u_y verifies the message received is indeed sent by u_x. Then u_y can compare HMAC(*K, p*) with HMAC(*K, q*) for $q \in (F_y, S_y, C_y)$. If HMAC(*K, p*)=HMAC(*K, q*) then *p*=*q* and a common friend, location, or CoI (corresponding to *F, S,* or *C*) is identified. If HMAC(*K, p*)≠HMAC(*K, q*), it prevents

the identities of uncommon friends/locations/CoIs from being revealed.

In the physical networks, devices provide and/or consume services utilizing SOAP-based techniques or RESTful APIs [54]. Each time when device d1 requests a service from device d2, d1 updates the user satisfaction experience record (in the user satisfaction experience list in Figure 3.2) towards d2 stored in the designated device of d1's user. Similarly, d1 can query the trust information (in the trust list in Figure 3.2) towards d2 from the designated device of d1's user. Note that elements in the user interaction experience list correspond to devices in the CoI list.

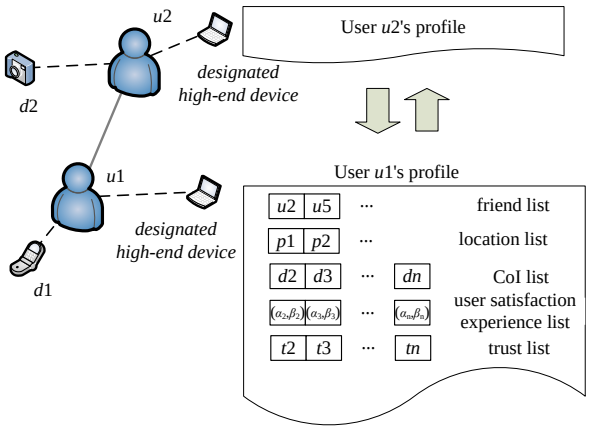


Figure 3.2: User Profile.

We consider a large IoT system in which a device with limited storage space cannot accommodate the full set of trust values towards all other devices. We address this scalability issue with a storage management design.

In the context of SOA, an owner provides services via its IoT devices. An IoT device providing a service will have to compete with other IoT devices which provide a similar type of service.

3.2 IoT Architecture Model

In general, IoT systems can be distributed, centralized, or hybrid. A distributed IoT system is typified by smart city IoT applications where IoT devices do not necessarily access to a centralized trusted entity such as the cloud. A centralized IoT system is typified by cross-state and cross-country IoT applications where IoT devices have access to centralized trusted clouds. A hybrid IoT system is the mix of distributed and centralized IoT systems organized in a cloud hierarchy. An example is a 3-tier "device-cloudlet-cloud" structure in a mobile cloud environment. A cloudlet can be viewed as a mobility-enhanced small-scale data center that brings the cloud closer. The main pur-

pose of the cloudlet is supporting resource-intensive and interactive mobile services by providing powerful computing resources to mobile devices to lower the latency. There are two important differences between a cloud and a cloudlet: (a) cloudlets need to be much more agile in their provisioning, and their association with IoT devices is highly dynamic due to user mobility; and (b) cloudlets must provide seamless data handoff services for a mobile device moving from one cloudlet to another for maintaining end-to-end service quality.

3.3 Threat Model

A malicious node in general can perform communication protocol attacks to disrupt network operations. We assume such attack is handled by intrusion detection techniques [2][3][4][5][22][36][44][46][78][79][80][81][82][83][84] and is not addressed in our work. In the context of SOA, we are concerned with trust-related attacks that can disrupt the trust system. Bad-mouthing and ballot-stuffing attacks are the most common forms of reputation attacks. Self-promoting and opportunistic service attacks are the most common forms of attacks based on self-interest [75][86][91]. Thus, a malicious IoT device (because its owner is malicious) can perform the following trust-related attacks:

1. *Self-promoting attacks*: a malicious node can promote its importance (by providing good recommendations for itself) for it to be selected as an SP, but then can provide bad or malfunctioned service.
2. *Bad-mouthing attacks*: a malicious node can ruin the reputation of a well-behaved device (by providing bad recommendations against it) so as to decrease the chance of that good device being selected as an SP. Bad-mouthing attack is a form of collusion attack when multiple malicious nodes perform bad-mouthing attacks to a good node to ruin the service trustworthiness of the good node.
3. *Ballot-stuffing attacks*: a malicious node can boost the reputation of a malicious node (by providing good recommendations) so as to increase the chance of that bad device being selected as an SP. Ballot-stuffing attack is also a form of collusion attack when multiple malicious nodes perform ballot-stuffing attacks to boost the service trustworthiness of one another.
4. *Opportunistic service attacks*: : a malicious node can provide good service to gain high reputation opportunistically especially when it senses its trust standing is dropping because of providing bad service. With a good trust standing, it can effectively collude with other bad nodes to perform bad-mouthing and ballot-stuffing attacks. We assume that a malicious node has a low trust score threshold below which it will behave (providing good service) in order to raise its trust standing, and a high trust

score threshold above which it will misbehave (providing bad service) to take advantage of its high trust standing for self-gain.

5. *Discriminatory attacks (or conflicting behavior attacks)*: a malicious node can discriminatively attack non-friends or nodes without strong social ties (without many common friends) because of human nature or propensity towards friends in social IoT systems. While serving as a recommender, if the target node is a friend or a malicious node, it can provide a good service recommendation (i.e., ballot stuffing attacks) even if the target node does not provide good service. On the other hand, if the target node is a non-friend, it can provide a bad service recommendation (i.e., bad-mouthing attacks) even if the target node provides good service. We address this issue by considering trust being formed not only from recommendations, but also from self-observations. Self-observations are one-to-one in nature, so the service behavior of a malicious node performing discriminatory attacks on a service requester will be remembered by the service requester. Consequently, our trust formation design has inherently addressed discriminatory attacks.

Table 3.1 summarizes the attack behavior of a malicious node as a rater, depending on the nature of the trustor and trustee nodes. If the trustor is non-malicious and the trustee is malicious, a malicious rater will perform ballot-stuffing attacks. If the trustor is non-malicious and the trustee is also non-malicious, a malicious rater will perform bad-mouthing attacks. The discriminatory attack is not included in the table because our trust protocol design has inherently addressed discriminatory attacks.

Table 3.1: Behavior of a Malicious Rater.

Trustor	Trustee	Bad-Mouthing	Ballot-Stuffing
malicious	malicious		
malicious	non-malicious		
non-malicious	malicious		√
non-malicious	non-malicious	√	

Table 3.2: Behavior of a Malicious Service Provider.

Service Requester	Self-Promoting	Opportunistic Service
malicious		
non-malicious	√	√

Table 3.2 summarizes the attack behavior of a malicious node as a SP, depending on the nature of the service requester. If the service requester is non-malicious, a malicious SP will perform both self-promoting and opportunistic service attacks. In particular, opportunistic service attacks are to be performed depending on the current reputation standing of the malicious SP itself.

In IoT sytems, we considered these 5 malicious behaviors. Since we are concerned with trust-related attacks that can disrupt the trust system. Bad-mouthing and ballot-stuffing attacks are the most common forms of reputation attacks. Self-promoting and opportunistic service attacks are the most common forms of attacks based on self-interest. Thus, our assumptions about malicious behaviors are realistic.

Our solution to cope with these malicious attacks is to use "social similarity based recommendation filtering" to filter out untrustworthy service rating recommendations and "adaptive filtering" to dynamically combine *direct trust* based on a target node's own experience and *indirect trust* based on other nodes' service rating recommendations weighted by the target node's subjective social relationship and trust view toward them.

3.4 Performance Metrics

In this section, we list performance metrics for measuring trust accuracy, convergence, resiliency, scalability, application performance.

Trust accuracy is measured by trust bias, i.e., the difference between the predicted trust value vs. ground truth trust. We collect a set of trust values over time and compare the set with ground truth using the mean square error (MSE) to measure trust bias.

Trust convergence is measured by the rate at which trust converges upon a change of node status, e.g., after being compromised or performing opportunistic service attacks. Let the difference between the predicted trust values of a node at t1 and t2, we say the trust value is converged when the difference is smaller than a tolerance limit. We measure convergence by the amount of time taken.

Trust resiliency is measured by the behavior of our trust protocol in terms of trust accuracy and convergence against increasing malicious population. The trust resiliency is measured by the highest malicious population beyond which trust accuracy and convergence can no longer be achieved.

Scalability is measured by the computational and communication overhead of an IoT device as the total number of IoT devices increases. Let CompOver(n) and CommOver(n) be the amount of overhead spent by an IoT node when the number of

nodes is n. Scalability is measured by CompOver(n) and CommOver(n), given n as the input.

Application performance is measured by the prediction accuracy of whether trustworthy service will be provided for the application. We measure application performance by a utility score representing the goodness of quality service after the application selects trustworthy SPs out of a myriad of IoT devices providing similar service. The application performance is related to trust formation, i.e., how trust is formed out of multi-dimensional trust components, as well as how trust is formed out of direct evidence and indirect evidence. Thus trust formation is application dependent and can affect application performance.

We measure the effectiveness of our IoT trust management protocol design by trust accuracy, trust convergence, trust resiliency, and application performance, as discussed above. We measure the efficiency of our IoT trust management protocol design by scalability, which is measured by the computational and communication overhead of an IoT device, as discussed above.

Chapter 4 Distributed IoT Trust Management

A future Internet of Things (IoT) system will connect the physical world into cyberspace everywhere and everything via billions of smart objects. On the one hand, IoT devices are physically connected via communication networks. The service-oriented architecture (SOA) can provide interoperability among heterogeneous IoT devices in physical networks. On the other hand, IoT devices are virtually connected via social networks. In our dissertation research, we propose adaptive and scalable trust management to support service composition applications in SOA-based IoT systems. We develop a technique based on distributed collaborative filtering to select feedback using similarity rating of friendship, social contact, and community of interest relationships as the filter. Further, we develop a novel adaptive filtering technique to determine the best way to combine direct trust and indirect trust dynamically to minimize convergence time and trust estimation bias in the presence of malicious nodes performing opportunistic service and collusion attacks. For scalability, we consider a design by which a capacity-limited node only keeps trust information of a subset of nodes of interest and performs minimum computation to update trust. We demonstrate the effectiveness of our proposed trust management through service composition applications with a comparative performance analysis against contemporary distributed IoT/P2P trust protocols, including ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86].

4.1 Trust Management Protocol

Our adaptive IoT trust management protocol is distributed. Each user maintains its own trust assessment towards devices. For scalability, a user just keeps its trust evaluation results towards a limited set of devices of its interests. Each user stores its profile in a designated high-end device (Figure 3.2). The profile of user u_x includes:

(1) A "friend" list including all friends of u_x, denoted by a set $F_x = \{u_a, u_b, \dots\}$;
(2) Locations that u_x frequently visited for social contact, denoted by a set $P_x = \{p_{x,1}, p_{x,2}, \dots\}$;
(3) List of devices that u_x has directly interacted with and the corresponding user satisfaction experience values, denoted by set $D_x = \{d_i, d_j, \dots\}$ and set $B_x = \{(\alpha_{x,i}, \beta_{x,i}),$

$(\alpha_{x,j}, \beta_{x,j}), \ldots\}$, where $\alpha_{x,i}$ and $\beta_{x,i}$ are the accumulated positive and negative user satisfaction experiences of user u_x towards device d_i;

(4)Trust values of user u_x towards IoT devices, denoted by a set $T_x = \{t_{x,i}, t_{x,j}, \ldots\}$.

4.1.1 Direct Interaction Experiences

We adopt Bayesian framework [65] as the underlying model for evaluating direct trust from direct user satisfaction experiences. The reason we choose Bayesian because it is well-established and because of its popularity in trust/reputation systems. In service computing, a service requester could rate a service provider after direct interaction based on nonfunctional characteristics. The nonfunctional characteristics include user-observed response time, failure probability, prices, etc. The current user satisfaction experience of user u_x toward device d_i is represented by a value, $f_{x,i}$. We consider the simple case in which the direct user satisfaction experience $f_{x,i}$ is a binary value, with 1 indicating satisfied and 0 not satisfied. Then, we can consider $f_{x,i}$ as an outcome of a Bernoulli trial with the probability of success parameter $\theta_{x,i}$ following a Beta distribution (a conjugate prior for the Bernoulli distribution), i.e., Beta($\alpha_{x,i}, \beta_{x,i}$). Then, the posterior p($\theta_{x,i} | f_{x,i}$) has a Beta distribution as well. Equation (4.1) shows how the hyper parameters $\alpha_{x,i}$ and $\beta_{x,i}$ are updated considering trust decay.

$$\begin{aligned} \alpha_{x,i} &= e^{-\varphi \Delta t} \cdot \alpha_{x,i}^{(old)} + f_{x,i} \\ \beta_{x,i} &= e^{-\varphi \Delta t} \cdot \beta_{x,i}^{(old)} + 1 - f_{x,i} \end{aligned} \tag{4.1}$$

In Equation (4.1), $f_{x,i}$ contributes to positive observations and $1 - f_{x,i}$ contributes to negative observations. When updating $\alpha_{x,i}$ and $\beta_{x,i}$, we consider an exponential decay, $e^{-\varphi \Delta t}$, on $\alpha_{x,i}^{(old)}$ and $\beta_{x,i}^{(old)}$, where φ is the decay factor which is normally is a small number to model small trust decay over time, and Δt is the trust update interval.

The direct trust of user u_x to device d_i, $t_{x,i}^d$, is calculated as the expected value of $\theta_{x,i}$, i.e.,

$$t_{x,i}^d = E[\theta_{x,i}] = \frac{\alpha_{x,i}}{\alpha_{x,i} + \beta_{x,i}} \tag{4.2}$$

In the literature, $\alpha_{x,i}$ and $\beta_{x,i}$ are often set to 1 initially since no prior knowledge available. In our work, we consider the social relationships (if available) between u_x and the user of d_i (say u_y) as the prior knowledge and set the initial values of $\alpha_{x,i}$ and $\beta_{x,i}$ to $sim(u_x, u_y)$ and $1 - sim(u_x, u_y)$, respectively, where $sim(u_x, u_y)$ is the similarity between u_x and u_y characterizing their social connection (discussed below).

4.1.2 Recommendations

When the devices of two users have direct interactions, they can exchange their profiles and provide trust recommendations. In addition, a device can also aggressively request trust recommendations from another device belonging to a friend when necessary. To preserve privacy, one can use a hash function (with session key) to prevent the identities of uncommon friends/devices from being revealed.

We utilize the design concept of distributed collaborating filtering [62][104] to select trust feedback from nodes sharing similar social interests. A node will first measure its "social similarity" with a recommender in friendship, social contact (representing physical proximity) and CoI (representing knowledge on the subject matter) and then decide if the recommendation is trustable. The reason we consider these metrics is that these metrics are core social metrics for measuring social relationships which are multifaceted [93]. We adopt cosine similarity to measure the distance of two social relationship lists (see Figure 3.2), with 1 representing complete similarity and 0 representing no similarity. Computational efficiency is the main reason why we choose cosine similarity to measure the similarity of two vectors in high-dimensional positive spaces because of limited computational capacity of IoT devices. In our work we further introduce a new design concept called *application performance maximization* by which the best weights assigned to the three similarity metrics are identified to optimize application performance, when given a node population characterized by friendship, social connection, and community of interest relationships as input. Later in Section 4.3 we will deal with the subject of the effect of social similarity in friendship, social connection, and community of interest on application performance and identify the best way of combining these metrics to maximize the service composition application performance.

We describe how these social similarity measures may be estimated dynamically as follows:

- **Friendship Similarity** (sim_f): The friendship similarity is a powerful social relationship (intimacy) for screening recommendations. After two users u_x and u_y exchange their friend lists, F_x and F_y, they could compute two binary vectors, $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$, each with size $|F_x \cup F_y|$. An element in $\overrightarrow{VF_x}$ (or $\overrightarrow{VF_y}$) will be 1 if the corresponding user is in F_x (or F_y), otherwise 0. Let $\|\vec{A}\|$ be the norm of vector $\vec{A}$ and $|B|$ be the cardinality of set B. Then, we could use the "cosine similarity" of $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$ (giving the cosine of the angle between them) to compute sim_f as follows:

$$sim_f(u_x, u_y) = \frac{\overrightarrow{VF_x} \cdot \overrightarrow{VF_y}}{\left\|\overrightarrow{VF_x}\right\| \left\|\overrightarrow{VF_y}\right\|} = \frac{\left|F_x \cap F_y\right|}{\sqrt{|F_x| \cdot \left|F_y\right|}} \tag{4.3}$$

- **Social Contact Similarity** (sim_l): The social contact similarity presents closeness and is an indication if two nodes have the same physical contacts and thus the same sentiment towards devices which provide the same service. The operational area could be partitioned into sub-grids. User u_x records the IDs of sub-grids it has visited in its *location list* P_x for social contact. After two users u_x and u_y exchange their location lists, P_x and P_y, they could compute sim_l in the same way of computing sim_f as follows:

$$sim_l(u_x, u_y) = \frac{\left|P_x \cap P_y\right|}{\sqrt{|P_x| \cdot \left|P_y\right|}} \tag{4.4}$$

- **Community of Interest Similarity** (sim_c): Two users in the same CoI share similar social interests and most likely have common knowledge and standard toward a service provided by the same device. Also very likely two users who have used services provided by the same IoT device can form a CoI (or are in the same CoI). After two users u_x and u_y exchange their device lists, D_x and D_y, they could compute sim_c in the same way of computing sim_f as follows:

$$sim_c(u_x, u_y) = \frac{\left|D_x \cap D_y\right|}{\sqrt{|D_x| \cdot \left|D_y\right|}} \tag{4.5}$$

The social similarity between two users can be a weighted combination of all social similarity metrics, i.e., friendship, social contact, and community of interest, considered in our work:

$$sim(u_x, u_y) = \sum_{v \in \{f,l,c\}} w_v \cdot sim_v(u_x, u_y) \tag{4.6}$$

where $w_f + w_l + w_c = 1$ and $0 \leq w_f, w_l, w_c \leq 1$. Each user can send trust recommendations request to its friends periodically (in every Δt interval) or before requesting a service. Upon receiving recommendations, user u_x selects top-k recommendations from k users with the highest similarity values with u_x and calculates the indirect trust ($t_{x,i}^r$) towards device d_i as follows:

$$t_{x,i}^{r} = \sum_{u_y \in U} \frac{sim(u_x, u_y)}{\sum_{u_y \in U} sim(u_x, u_y)} \cdot t_{y,i}^{d} \tag{4.7}$$

Here, U is a set of up to k users whose $sim(u_x, u_y)$ values are the highest, and $t_{y,i}^{d}$ is the direct trust of user u_y toward device d_i serving as $u_y's$ recommendation toward d_i provided to u_x. Each recommendation is weighted by the ratio of the similarity score of the recommender to the sum of the similarity scores of all recommenders. We also note that if u_y is malicious, then it can provide $t_{y,i}^{d}$=0 against a good device for bad-mouthing attacks, and $t_{y,i}^{d}$=1 for a bad node for ballot-stuffing attacks.

4.1.3 Adaptive Control of the Weight Parameter

The trust value of user u_x toward d_i is denoted as $t_{x,i}$ and is obtained by combining direct trust and indirect recommendations (if available) as follows,

$$t_{x,i} = \mu \cdot t_{x,i}^{d} + (1 - \mu) \cdot t_{x,i}^{r} \tag{4.8}$$

Here, μ is a weight parameter ($0 \leq \mu \leq 1$) to weigh the importance of direct trust relative to indirect trust feedback. The selection of μ is critical to trust evaluation. A contribution of the dissertation research is that we propose a method based on adaptive filtering [62] to adjust μ dynamically in order to effectively cope with malicious attacks including self-promoting, bad-mouthing, ballot-stuffing, and opportunistic attacks and to improve trust evaluation performance. The basic design principle is that a successful trust management protocol should provide high trust toward devices who have more positive user satisfaction experiences and, conversely, low trust toward those with more negative user satisfaction experiences. Specifically, the current trust evaluation (i.e., $t_{x,i}(\mu)$ as a function of μ) should be as close to the average user satisfaction experiences observed over the last trust update window Δt as possible. Therefore, we formulate the selection of μ as an optimization problem as follows:

$$\begin{aligned} &\underline{\text{Find}}: \mu, 0 \leq \mu \leq 1 \\ &\underline{\text{Minimize}}: \text{MSE}(\mu) = \sum_i \left(t_{x,i}(\mu) - \overline{f_{x,\iota}^{(new)}} \right)^2 \end{aligned} \tag{4.9}$$

Here, $t_{x,i}(\mu)$ is obtained from Equation (4.8) using past direct user satisfaction experiences and indirect trust feedback, and $\overline{f_{x,\iota}^{(new)}}$ is the most recent direct user satisfaction experiences observed by user u_x within the last trust update interval Δt. The objective can be achieved by minimizing the mean square error (MSE) of trust evaluations against actual user satisfaction experiences towards all applicable devices, such that the

trust value could be a good indicator or predictor for quality of service (with direct user satisfaction experiences considered as ground truth). After user u_x obtains new user satisfaction experiences over Δt, it can compute the average user satisfaction experience value $\overline{f_{x,\iota}^{(new)}}$ and update μ by minimizing MSE in Equation (4.9). The optimization problem in Equation (4.9) can be solved by plugging $t_{x,i}(\mu)$ in Equation (4.8) into Equation (4.9) and minimizing $\mathrm{MSE}(\mu)$ as follows:

$$\mathrm{MSE}(\mu) = \sum_i \left(\mu \cdot t_{x,i}^d + (1-\mu) \cdot t_{x,i}^r - \overline{f_{x,\iota}^{(new)}}\right)^2 \tag{4.10}$$

The minimum value of $\mathrm{MSE}(\mu)$ is obtained at the point where the derivative is zero, i.e., $\mathrm{MSE}'(\tilde{\mu}) = 0$. Thus, $\tilde{\mu}$ is obtained as follows,

$$\tilde{\mu} = \frac{\sum_i \left(\overline{f_{x,\iota}^{(new)}} - t_{x,i}^r\right)\left(t_{x,i}^d - t_{x,i}^r\right)}{\sum_i \left(t_{x,i}^d - t_{x,i}^r\right)^2} \tag{4.11}$$

The optimal value of μ (i.e., $\hat{\mu}$) should be in the range of [0, 1] because it is a weight parameter. Therefore,

$$\hat{\mu} = \begin{cases} 0 & \tilde{\mu} < 0 \\ \tilde{\mu} & 0 \le \tilde{\mu} \le 1 \\ 1 & \tilde{\mu} > 1 \end{cases} \tag{4.12}$$

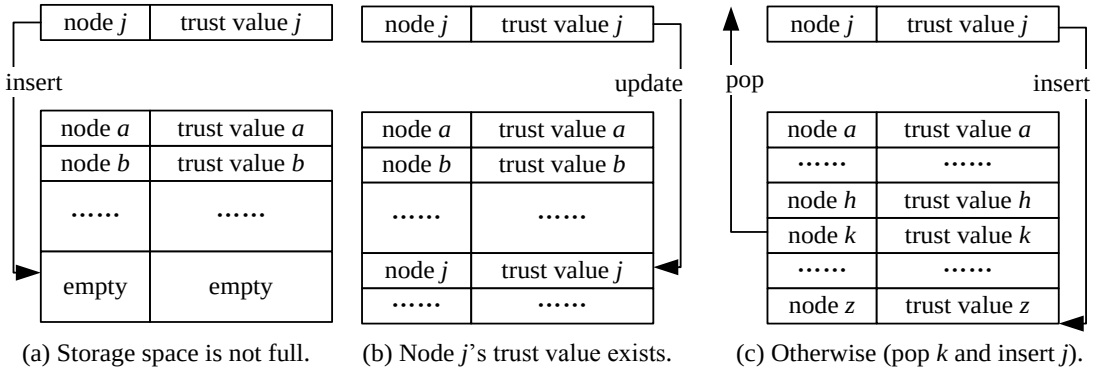
Each user computes its own optimal value of μ (i.e., $\hat{\mu}$) and updates it dynamically in every trust update time interval Δt, based on Equations (4.11) and (4.12), using the historical data collected in its storage, so there is essentially no extra overhead. This adaptive design is applicable to other trust parameters (i.e., φ and (w_f, w_l, w_c)) as well. However, introducing these trust parameters in Equation (4.9) leads to a more complex optimization problem and may not be feasible for IoT devices with limited resources. In our dissertation research, we focus on adaptive control of μ and leave adaptive control of other trust parameters as future work.

Our dynamic weight adjustment scheme is driven by minimizing the difference between $t_{x,i}(\mu)$ (obtained from Equation (4.8)) and $\overline{f_{x,\iota}^{(new)}}$ (obtained in the last trust update interval Δt). If d_i is a malicious node and it retains high reputation either because it performs opportunistic service attacks to gain high reputation, or because other nodes provide ballot-stuffing attacks to boost its reputation, then our trust system will be temporarily deceived of its true status because the difference between these two quantities will be small. However, the moment d_i performs self-promoting attacks and provides bad service to user u_x, this bad user experience will be immediately observed

by user u_x and, as a result, the difference between these two quantities will be large enough to drive the change of μ to minimize MSE(μ) in Equation (4.10). It is noteworthy that μ is dynamically adjusted based on minimizing the sum of the differences of all devices observed by user u_x over Δt, so adjusting μ to minimize MSE(μ) moves toward the right direction of minimizing the difference between "the subjective trust" vs. "what service quality is actually provided" for all devices with which user u_x has interaction experiences over Δt.

4.1.4 Storage Management for Small IoT Devices

Considering a large-scale IoT system in which each node has limited storage space to keep direct user satisfaction experiences and trust values of a small set of nodes with which it shares interests. A node has to decide which trust values to keep. In general, nodes are more interested in others with higher trust values. However, simply saving the trust values towards the most trustworthy nodes cannot make the trust evaluation process converge and is not adaptive to dynamic environments since there is little chance to accumulate trust towards newly joining nodes. Our storage management strategy considers nodes with the highest trust values and recent interacting nodes as these nodes are most likely to share common interests.



(a) Storage space is not full. (b) Node *j*'s trust value exists. (c) Otherwise (pop *k* and insert *j*).

Figure 4.1: Storage Management for Small IoT Devices.

Figure 4.1 illustrates how our approach works conceptualizing the storage size of each node as n (meaning that there is space to save trust values of up to n nodes). When a slot is needed, for a node's trust value to be kept it must be in the top Ω of the n trust values, or this node is one of the most recent interacting nodes. We consider Ω = 50% in our work and the selection of optimal Ω value in dynamic IoT systems can be solved using the same adaptive control in Section 4.1.3.

When node i obtains the trust value towards node j, if the storage space is not full or node i does have the trust information of node j in its storage space, then node i will

simply save the trust value towards node *j*. If the storage space is full and node *i* does not have the trust information of node *j* in its storage space, node *i* will put the trust value towards node *j* and pop out the trust value towards the earliest interacting node among those with trust values below the median (Ω = 50%). By using a max-min-median heap, find medium, maximum or minimum operations can be performed in O(1) constant time, while all others operations (find, insert and delete) can be performed in O(log n) logarithmic time.

4.2 Trust Protocol Performance

In this section, we report simulation results obtained as a result of executing our proposed autonomous trust management protocol by IoT devices. We choose ns-3 [108][109] as the simulator as it emerges as the de facto standard open simulation platform for networking research; it is a discrete-event network simulator, targeted primarily for research and educational use.

The focus in this section is to demonstrate our protocol's desirable convergence and accuracy properties, as well as its resiliency property against malicious attacks. In Section 4.3, we will apply it to service composition and compare its performance against the baseline trust management schemes.

Our simulation results have three parts. First, we demonstrate trust convergence, accuracy and resiliency properties of our adaptive IoT trust protocol design against malicious attacks. We then demonstrate the effectiveness of our storage management protocol design for IoT devices with limited storage space. Lastly, we perform a comparative analysis of our adaptive IoT trust protocol against four baseline schemes: ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] for which we have discussed in detail in Chapter 2.

4.2.1 Environment Setup

Table 4.1: Parameter List and Default Values Used.

parameter	value	parameter	value	parameter	value
N_T	400	$m \times m$	16×16	T	200hrs
N	40	P_M	30%	φ	0.001
Ω	50%	σ_ϵ	0.01	λ	1/day
Δt	2hrs				

Table 4.1 lists the default parameter values. We consider an IoT environment with N_T = 400 heterogeneous smart objects/devices. These IoT devices are randomly assigned to N = 40 users. Users are connected in a social network represented by a friendship matrix [72]. We consider these users moving according to the SWIM mobility model [71] modeling human social behaviors in an $m \times m$=16×16 operational region for the purpose of assessing the social contact similarity metric between any pair of users. Direct trust of node i toward node j is assessed upon completion of a service request from node i to node j. Each node requests services from a selected device with a time interval following an exponential distribution with parameter λ, with 1/day being the default unless otherwise specified. The trust update interval Δt is 2 hours at which time if there is no direct trust update due to service request and completion, direct trust will be decayed according to Equation (4.1). Indirect trust is always updated in every Δt interval according to Equation (4.7). The system runs continuously although we often can observe trust convergence in less than 200 hours, given that bad nodes follow the attack behaviors specified in Section 3.3.

The user satisfaction levels of service invocations are generated based on a real dataset [107] and are used as "ground truth" based on which the accuracy of our trust protocol is assessed. As the direct trust of user u_x toward device/service provider d_i (i.e., $t_{x,i}^d$) is calculated by Equation (4.1) with "ground truth" user satisfaction experiences as input, $t_{x,i}^d$ essentially is equal to ground truth. However, we account for the presence of noise in the IoT environment (i.e., error of assessing user satisfaction level received) by considering a standard deviation parameter σ_c (set to 1% as default) to reflect the deviation of the actual user satisfaction level as recorded in the database from the direct trust evaluation outcome in terms of $t_{x,i}^d$.

Initially, $t_{x,i}$ is set to 0.5 (ignorance) by user u_x for all i's. Then, trust is updated dynamically as nodes encounter each other, as services are requested and rendered, and as trust feedback are acquired. We consider $w_f = w_l = w_c = 1/3$ (in Equation (4.6)) as we assess the convergence and accuracy properties of our trust protocol in this section. Later in Section 4.3 we will identify the best weight assignment (w_f, w_l, w_c) for social similarity computation for the service composition application.

We test the resiliency of our trust protocol against malicious node behavior (i.e., performing self-promotion, bad-mouthing and ballot-stuffing attacks) by randomly selecting a percentage P_M out of all as dishonest malicious nodes with P_M=30% as the default. A normal or good node follows the execution of our trust management protocol faithfully, while a malicious node provides false trust feedback by means of ballot-stuffing, bad-mouthing, and self-promoting attacks to gain advantage.

4.2.2 Trust Convergence, Accuracy and Resiliency against Malicious Attacks

In this section, we examine the trust convergence, accuracy and resiliency properties of our adaptive IoT trust protocol design. We first compare static control (i.e., μ is fixed at a constant) vs. adaptive control (i.e., μ is changed dynamically based on Equation (4.12).

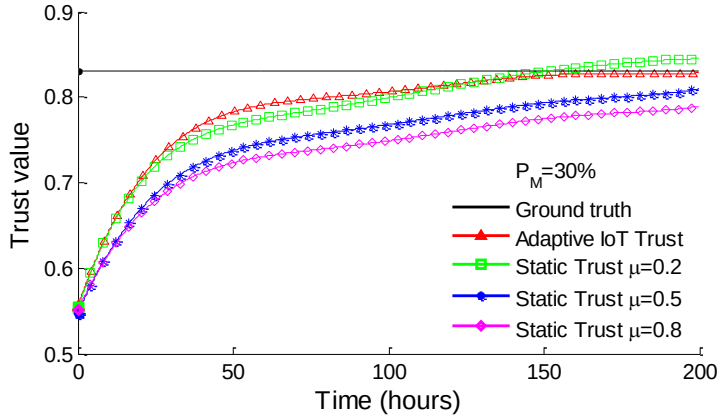


Figure 4.2: Trust Value of a Good Node with P_M = 30%.

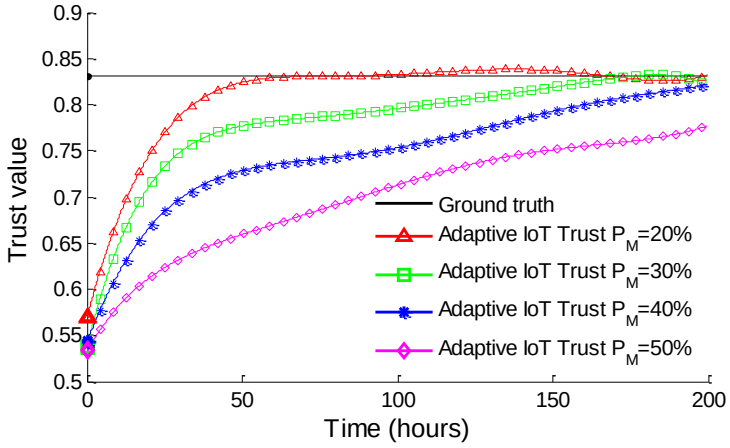


Figure 4.3: Trust Value of a Good Node under Adaptive Control with P_M ranging from 20% to 50%.

Figure 4.2 shows trust evaluation results for a trustor node toward a "good" trustee node randomly picked. We see that trust convergence behavior is observed for either fixed or adaptive control. There is a tradeoff between convergence time vs. trust bias. With static control, when a higher μ value is used, the trust convergence time is longer, but the trust bias is smaller, i.e., the trust value is closer to ground truth after

convergence. With adaptive control, on the other hand, the trustor node is able to adjust μ dynamically to minimize both the convergence time and the trust bias after convergence. Here we note that the trust value of a "good" trustee is not 1 because we use the user satisfaction levels of service invocations based on a real dataset [107] with a standard deviation parameter σ_c (set to 1% as default) reflecting the deviation of the actual user satisfaction level recorded in the database from the direct trust evaluation outcome.

An interesting observation in Figure 4.2 is that if μ is too small (e.g., 0.2) the trust value is over-estimated upon convergence, which is not a desirable outcome as trust overshoot is considered a bad property detrimental to the stability of a trust system [43]. Our adaptive protocol dynamically adjusts μ for fast convergence without incurring trust overshoot.

Figure 4.2 is for the case in which the percentage of malicious nodes P_M = 30%. We conduct experiments to test the residency of our trust protocol against increasing malicious node population. Figure 4.3 shows that as the population of malicious nodes increases, both the convergence time and trust bias increase. However, the system is found to be resilient to malicious attacks for P_M as high as 40%, with proper convergence and accuracy behaviors exhibited. In general we observe that the trust bias is minimum, e.g., < 5% when $P_M \leq 40\%$ and the trust bias becomes more significant, e.g., > 10% when $P_M \geq 50\%$. This demonstrates the resiliency property of our trust protocol against malicious attacks.

Correspondingly, Figure 4.4 shows how our trust-based adaptive control protocol adjusts μ in Equation (4.12) in response to increasing malicious node population.

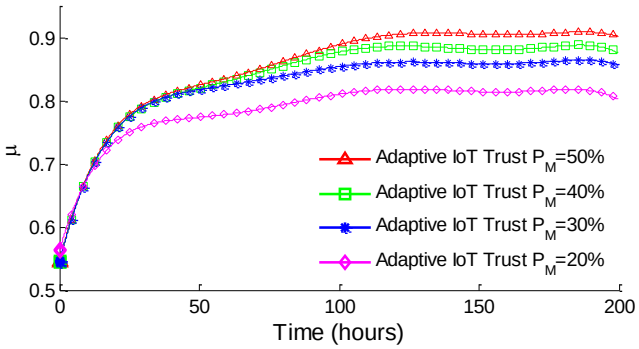


Figure 4.4: Adjustment of μ against Increasing Malicious Node Population.

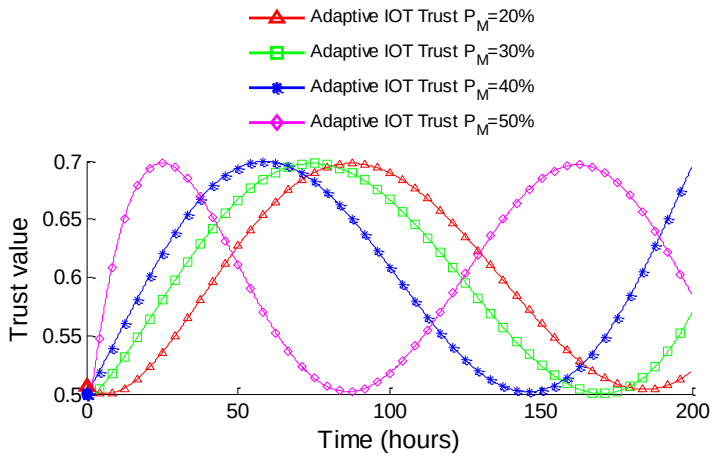


Figure 4.5: Trust Value of a Bad Node under Adaptive Control with P_M ranging from 20% to 50%.

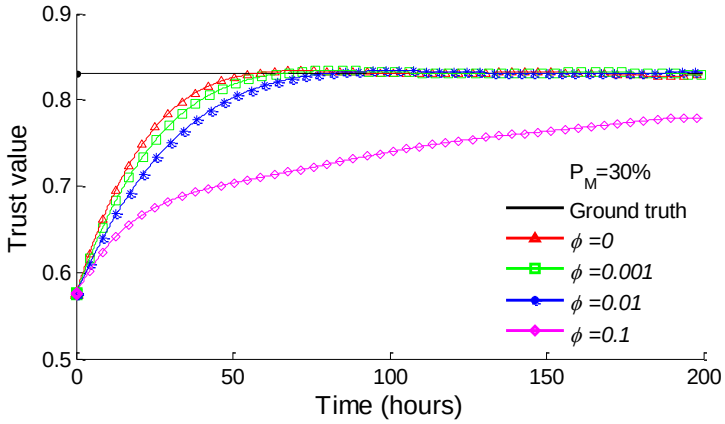


Figure 4.6: Effect of Decay Parameter on Trust Convergence.

We observe that as the malicious node population increases, the system will have to rely more on direct trust by increasing μ and conversely rely less on indirect trust by decreasing $1-\mu$ so as to mitigate the effect of bad-mouthing and ballot-stuffing attacks by malicious nodes. Figure 4.4 shows that when P_M = 20%, the optimal converged μ value is 0.78, while when P_M = 50%, the optimal converged μ value is 0.90. This follows the design principle of "go up slowly, reduce quickly," that is, when a node acts maliciously, its trust value should reduce quickly, and when a node acts cooperatively, its trust should just go up slowly. When a node is being observed maliciously, its trust value will be reduced quickly because in this case a high μ value will be used by our trust protocol and a high μ value means that the trust value of the malicious node will be very close to direct trust which is low as the node is being observed maliciously. Conversely, when a node is being observed cooperatively, its trust value will just go up

slowly because in this case a low μ value will be used by our adaptive protocol and a low μ value means that both direct trust and indirect trust will contribute to the overall trust based on Equation (4.8). Although in this case, the direct trust observed is high as the node is being observed cooperatively, it will only increase the overall trust value slowly by a weight of μ, with the indirect trust contributing to the overall trust by a weight of $1-\mu$. The system cannot rely on direct trust 100% because malicious nodes can perform opportunistic service attacks and there is an error of assessing direct trust due to noise in the environment. Figure 4.4 demonstrates that our adaptive control mechanism is effective in terms of convergence of μ to its optimal value under which trust bias is minimized.

Figure 4.5 shows trust evaluation results for a trustor node toward a "bad" trustee node. Among all attacks, the bad node performs *opportunistic service attacks* with the high trust threshold being 0.7 and the low trust threshold being 0.5. Specifically, the bad node provides good service to gain high reputation opportunistically when it senses its reputation drops below 0.5. Once it reputation rises to 0.7, it provides bad service again. We see from Figure 4.5 that our adaptive trust protocol is able to accurately track the trust fluctuation of the bad node performing opportunistic service attacks. We observe that the rate of trust fluctuation is higher when P_M is higher because more malicious nodes can collude to quickly bring the trust level of the bad node to 0.7.

The effect of the decay parameter φ is analyzed in Figure 4.6. A smaller φ means a slower trust decay rate with φ=0 meaning no trust decay. We choose φ=0.001 to achieve the desirable convergence behavior. We see that as φ increases, it takes longer to achieve trust convergence. This is because a good node remains good for its lifetime so a larger trust decay rate requires a good node to become more socially and service active over time in order to regain its trust status. In this case, we see that φ=0 produces the fastest convergence rate. This is not necessarily true for cases in which a good node may be compromised dynamically for which φ>0 may become the best setting. The determination of the optimal φ to trade convergence with accuracy as dictated by environment conditions is a future research area.

4.2.3 Trust Evaluation with Limited Storage Space

The results presented in Section 4.2.1 are based on the assumption that each node has sufficient storage to save trust values of all nodes. In this section, we consider a more realistic scenario in which many small IoT devices only have a limited storage space. A trustor node in this case would run the trust storage management strategy described in Section 4.1.4 to store trust values considered important to the node.

Figure 4.7 compares the trust value obtained by a trustor node toward a good trustee node randomly picked, when P_M = 30% and each node has 10%, 50% or 100% space to accommodate all trust values. We first note that the curve labeled with "adaptive trust-based 100% storage" in Figure 4.7 is the same as the curve labeled with "adaptive trust-based" in Figure 4.2. We observe that the convergence time and trust bias after convergence are comparable for the 10% and 50% storage cases and they don't deviate much from those for the 100% storage case. This demonstrates the effectiveness of our management strategy for limited storage. We attribute this to its ability to function like a filter, thus excluding highly deviated trust feedback coming from untrustworthy nodes to shield the system from false recommendation attacks.

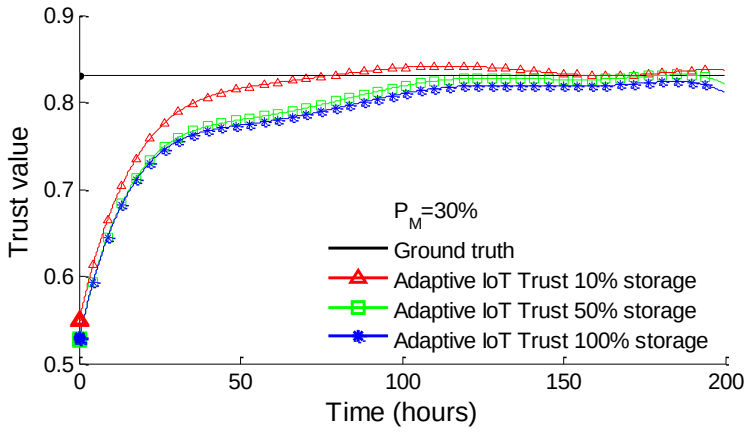


Figure 4.7: Adaptive Control with Limited Storage.

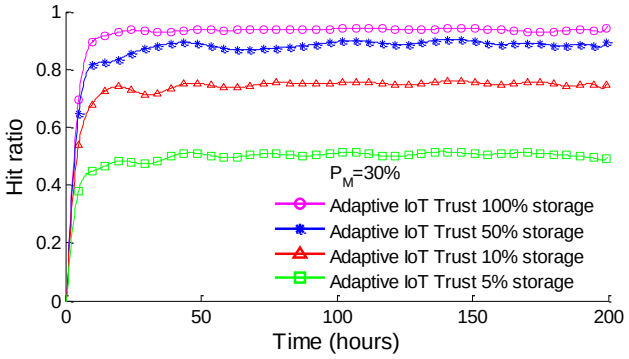


Figure 4.8: Hit Ratio with Limited Storage.

Lastly, we examine the effect of our management strategy for limited storage on hit ratio. We define the "top-*m* hit ratio" as the percentage of the top-*m* most trustworthy nodes having their trust values stored in the limited *n* slots. Figure 4.8 shows the top-20 hit ratio as a function of time for a randomly selected node. We can see that initially the hit ratio is zero because there is no trust information stored for any node. As the trust

value converges, the hit ratio quickly increases and approaches its peak. We see that the maximum achievable hit ratios are 90%, 85%, 75% and 50% under 100%, 50%, 10% and 5% storage spaces, respectively. Even with as little storage space as 10%, the hit ratio only deteriorates from 90% to 75%. This again demonstrates the effectiveness and high space utilization of our management strategy for limited storage.

4.2.4 Comparative Analysis

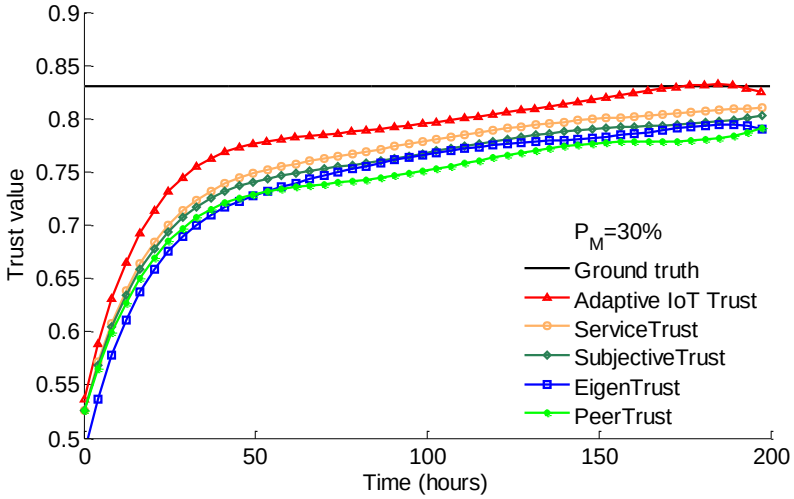


Figure 4.9: Performance Comparison of Trust Convergence, Accuracy and Resiliency when P_M =30%.

Figure 4.9 shows head-to-head performance comparison data of our adaptive IoT trust protocol against ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86], for the trust evaluation of a good node randomly selected. The environment conditions are setup the same way as in Figure 4.2 with P_M=30%. We see that while all protocols converge at about the same rate, our protocol achieves accuracy but ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] suffer inaccuracy.

Figure 4.10 shows the corresponding 3-dimensional view with P_M varying in the range of 20% to 40%. We see that the trust bias gap (difference to ground truth) for ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] widens as P_M increases, while it remains minimum for our adaptive IoT trust protocol against increasing malicious node population. This demonstrates the resiliency property of our trust protocol against malicious attacks. We attribute the superiority of our adaptive IoT trust protocol over these 4 baseline schemes to our protocol's adaptability. That is, our protocol is able to adjust the best trust parameter (μ) dynamically to achieve trust

accuracy despite the presence of a high percentage of malicious nodes performing opportunistic service attacks to boost their own reputation scores opportunistically and colluding (via bad-mouthing attacks) to ruin the reputation of this good node.

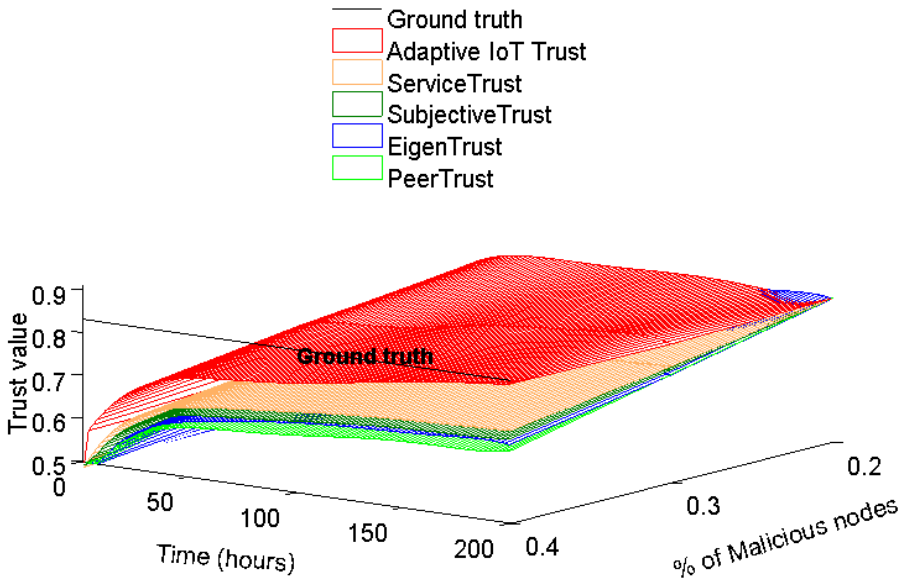


Figure 4.10: Performance Comparison of Trust Convergence, Accuracy and Resiliency in 3-D View with P_M ranging from 20% to 40%.

4.3 Applicability to Real World IoT Systems

In this section, we apply our trust management protocol to three trust-based IoT service composition applications. We compare the performance of *trust-based service composition* with two baseline approaches:

1. *Ideal service composition* which returns the maximum achievable utility score derived from ground truth or global knowledge.
2. *Random service composition* which randomly selects service providers for service composition without regard to trust.

We also compare the performance of our protocol with the 4 baseline distributed IoT/P2P trust protocols, ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86].

4.3.1 Smart City Air Pollution Detection

We consider a smart city IoT application running on Alice's smartphone for air pollution detection [16]. Alice tries to avoid stepping into high air pollution areas (in terms of the levels of carbon dioxide, PM10, etc.) for health reasons. Alice's smartphone is a member of the air pollution awareness social network. She decides to invoke her smartphone to connect to sensor devices in an area she is about to step (or drive) into. Alice knows that many IoT devices will respond, so she needs to make a decision on which sensing results to take. She instructs her smartphone to accept results only from n=5 most "trustworthy" sensors and she will follow a trust-weighted majority voting result. That is, each yes or no recommendation is counted as 1 weighted by Alice's trust toward the recommender. If the total trust-weighted "yes" score is higher than the total trust-weighted "no" score, Alice will step into the area; otherwise, she will make a detour to avoid the area.

This smart city air pollution detection application is essentially a simple trust-based service composition IoT application in which Alice will simply select n=5 IoT devices for which she trusts the most. Therefore, the *trustworthiness* score of this service composition application which it aims to maximize is simply the average of the 5 individual trustworthiness scores Alice has toward the 5 IoT devices, as given in Equation (4.8).

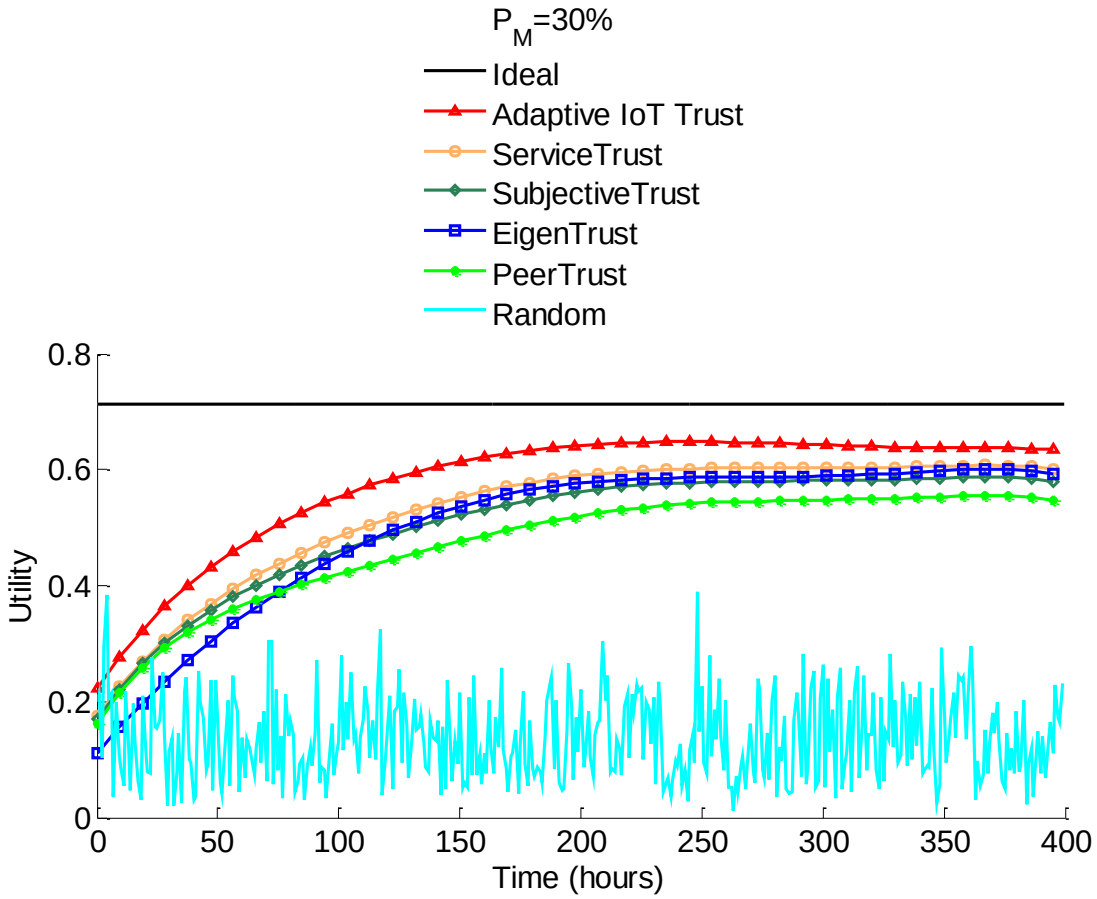


Figure 4.11: Utility Score of the Smart City Air Pollution Detection Application.

Figure 4.11 shows the ns-3 simulation results with PM=30%. We observe that trust-based service composition with our adaptive IoT trust protocol significantly outperforms random service composition and upon convergence approaches the performance of ideal service composition based on ground truth. Further, our adaptive IoT trust protocol outperforms ServiceTrust[94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] as the underlying trust protocol for trust-based service composition.

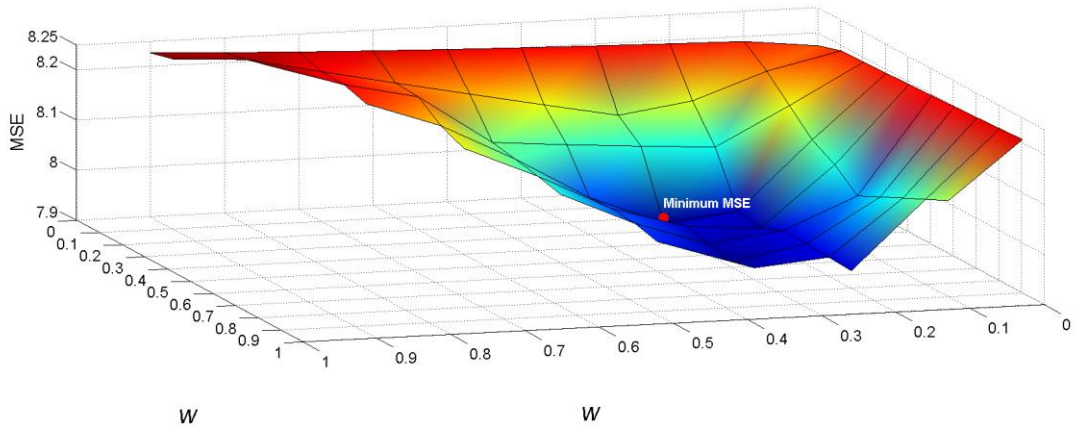


Figure 4.12: Mean Square Error of Utility Difference vs. (w_f,w_l,w_c). Minimum MSE at (w_f,w_l,w_c) = (0.3, 0.3, 0.4) for the Air Pollution Detection Application.

Figure 4.12 shows the simulation results of the MSE of the difference between the utility obtainable under trust-based service composition and the ideally achievable utility for the service composition application vs. (w_f, w_l, w_c). Note that w_c=$1 - w_f - w_l$ and is not shown in the 3-D diagram. One can see clearly from Figure 4.12 that there exists an optimal weight assignment (w_f, w_l, w_c)= (0.3, 0.3, 0.4) under which MSE is minimized, i.e., the utility obtainable via trust-based service composition is closest to the ideally achievable utility with perfect global knowledge of node status. The methodology developed in this document will allow each service requester to dynamically decide and apply the optimal weight combination (w_f, w_l, w_c) that will lead to the most credible trust feedback to minimize trust bias and as a result maximize the utility or the user satisfaction level of the application.

4.3.2 Augmented Map Travel Assistance

Bob has never traveled to Washington DC so he is excited but also nervous about the quality of service he will receive during his visit. He is aware of the fact that DC is a smart city so he registers his smartphone to the *travelers-in-Washington-DC* social network. He also downloads an *augmented map* social IoT application [6] to run on his smartphone, allowing his Near Field Communications (NFC) equipped smartphone to browse a tag-augmented DC map wherever he goes sightseeing. This tag-augmented map automatically connects Bob's smartphone to IoT devices available upon encounter, which provide information, food, entertainment (e.g., ticket purchasing), and transportation services [6]. Bob instructs his smartphone to make selection decisions dynamically, so it can leverage new information derived from direct experiences and

recommendations received from IoT devices it encounters. In response to a service request issued by Bob, his smartphone must:

- gather sensing data or information collected from the physical environment based on either self-observations or recommendations;
- formulate a service plan based on the results gathered; and
- invoke necessary services to meet Bob's service demand and requirements.

To this end, the augmented map travel assistance application running on his smartphone composes a service workflow plan as shown in Figure 4.13 in response his service request "Fill me with the best grilled hamburger within 20 minutes under a $30 budget." With the service plan formulated, Bob's smartphone selects the best service providers out of a myriad of service providers to execute the service plan. The objective of this trust-based service composition application running on Bob's smartphone is to select the most trustworthy IoT nodes for providing services specified in the flow structure subject to the time and budget constraints (20 minutes and 30 dollars) such that the overall *trustworthiness* score representing the goodness of the service composition is maximized.

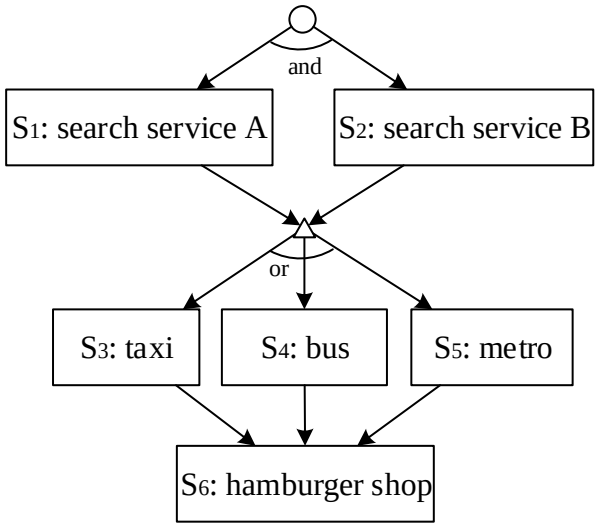


Figure 4.13: A Service Flow Structure for the Augmented Map Travel Assistance Application.

In Figure 4.13, there are 6 atomic services connected by three types of workflow structures: *sequential*, *parallel* (AND), and *selection* (OR). Each service would have multiple service provider candidates. In this case, the overall trustworthiness score of this service composition application can be calculated recursively in the same way the reliability of a series-parallel connected system is calculated. Specifically, the

trustworthiness score of a composite service (whose trustworthiness score isT_s) that consists of two subservices (whose trustworthiness scores are T_1 and T_2) depends on the structure connecting the two subservices as follows:

a) Sequential structure: $T_s = T_1 \times T_2$;

b) Selection structure (OR): $T_s = \max(T_1, T_2)$;

c) Parallel structure (AND): $T_s = 1 - (1 - T_1) \times (1 - T_2)$.

For the flow structure in Figure 4.13, the outmost structure is a sequential structure connecting (S_1 S_2), (S_3, S_4 S_5), and S_6 out of which (S_2 S_2) is a parallel structure and (S_3, S_4 S_5) is a selection structure.

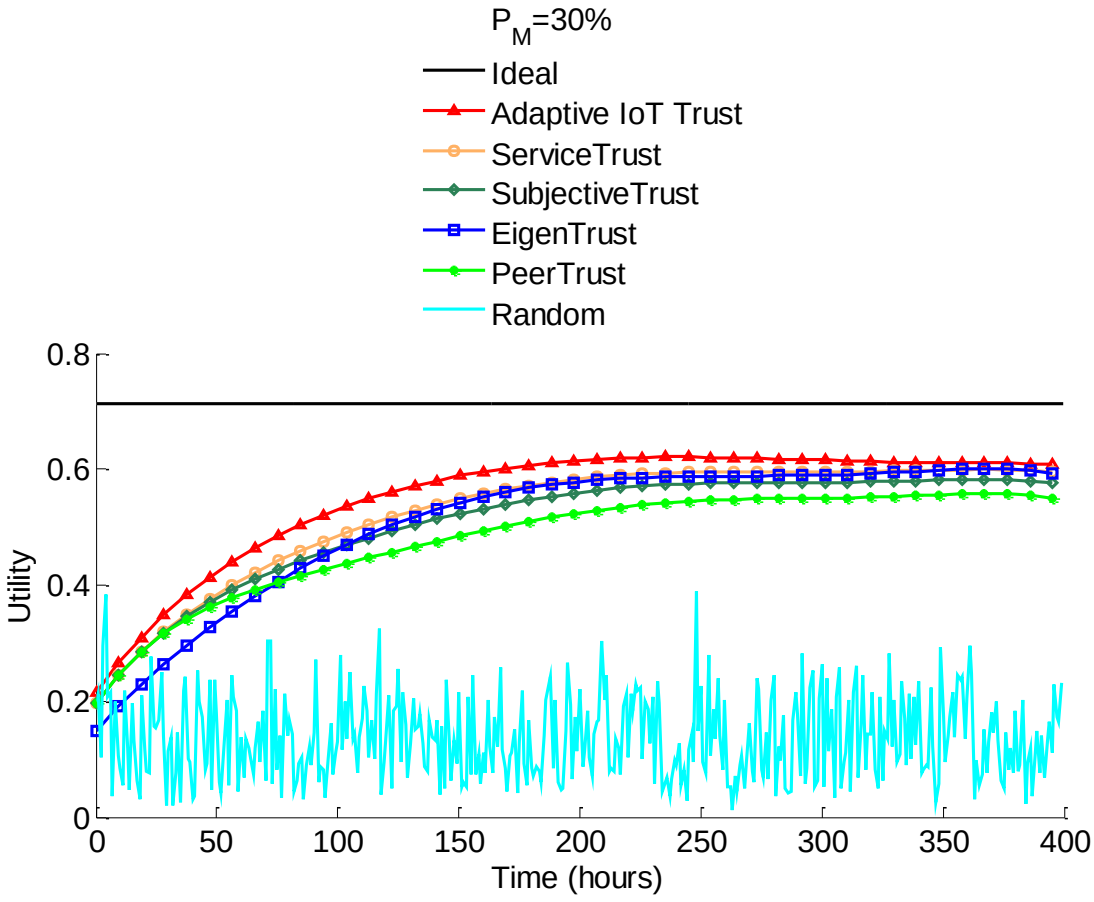


Figure 4.14: Utility of the Augmented Map Travel Assistance Application.

Figure 4.14 shows the ns-3 simulation results with PM=30%. We observe that for the augmented map travel assistance application, trust-based service composition with our

adaptive IoT trust protocol again significantly outperforms random service composition and upon convergence approaches the performance of ideal service composition based on ground truth. Further, our adaptive IoT trust protocol outperforms ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] in terms of trust convergence rate and trust accuracy.

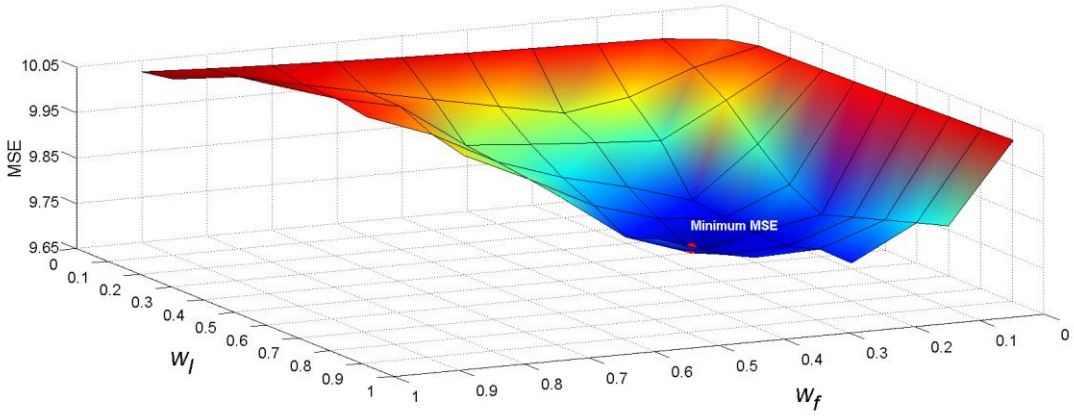


Figure 4.15: Mean Square Error of Utility Difference vs. (w_f,w_l,w_c). Minimum MSE at (w_f,w_l,w_c) = (0.3, 0.5, 0.2) for the Augmented Map Travel Assistance Application.

Figure 4.15 shows the simulation results of the MSE of the difference between the utility obtainable under trust-based service composition and the ideally achievable utility for the service composition application vs. (w_f, w_l, w_c). One can see clearly from Figure 4.15 that there exists an optimal weight assignment (w_f, w_l, w_c)= (0.3, 0.5, 0.2) under which MSE is minimized. Here we note that the optimal weight assignment is application dependent.

4.3.3 Travel Planning

Ed has never visited New York City. He wants to plan his travel early on from Seattle, Washington, including airline reservation from Seattle to New York, ground transportation after reaching New York, hotel reservation at New York, entertainments and attractions while in New York, hotel shuttle to the airport, etc. Ed instructs his smart phone to first construct a workflow structure for the travel and then select from a myriad of IoT SPs to populate the workflow structure. Figure 4.16 shows the workflow structure constructed by his smartphone. There are 9 atomic services connected by three types of workflow structures in this example, namely, *sequential*, *parallel* (AND), and *selection* (OR). Each service would have multiple SP candidates.

We use the "true" user satisfaction levels received from the SPs selected for the service composition application to derive the overall user satisfaction level, called the *utility* score, to evaluate the performance of service composition. The utility score of a candidate service composition is calculated recursively. Specifically, the utility score of a composite service (whose utility score is us_s) comprising two subservices (whose utility scores are us_1 and us_2) depends on the structure connecting the two subservices as follows:

- Sequential Structure: $us_s = us_1 \times us_2$;
- Selection Structure: $us_s = \max(us_1, us_2)$;
- Parallel Structure: $us_s = 1 - (1 - us_1) \times (1 - us_2)$.

We also use the percentage of malicious nodes selected as SPs for providing the travel service as an additional performance metric. For *trust-based service composition*, the goal is to select service providers based on trust evaluation such that the composite service utility score is the best.

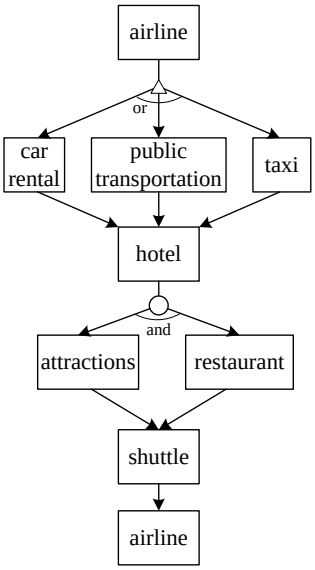


Figure 4.16: A Service Composition Example (Travel Planning).

We differentiate two types of service composition applications: without constraints and with constraints, i.e., a budget limit for travel planning. In both scenarios, we compare the performance of *trust-based service composition* running on top of our adaptive IoT trust protocol against that running on top of ServiceTrust, EigenTrust, and PeerTrust.

4.3.3.1 *Service Composition without Constraints*

In *trust-based service composition without constraints*, the SR selects the SP with the highest trust value for each required service.

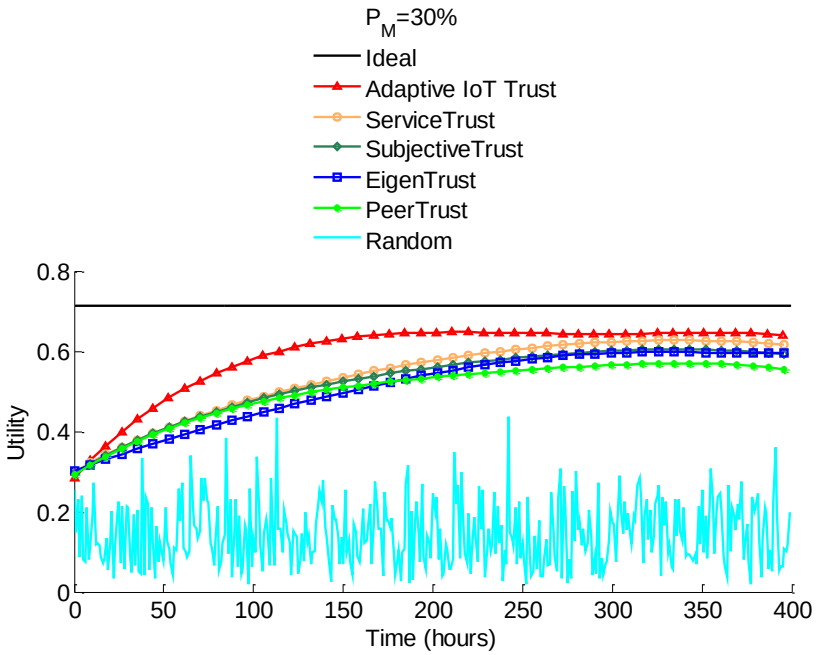


Figure 4.17: Utility of Service Composition without Constraints.

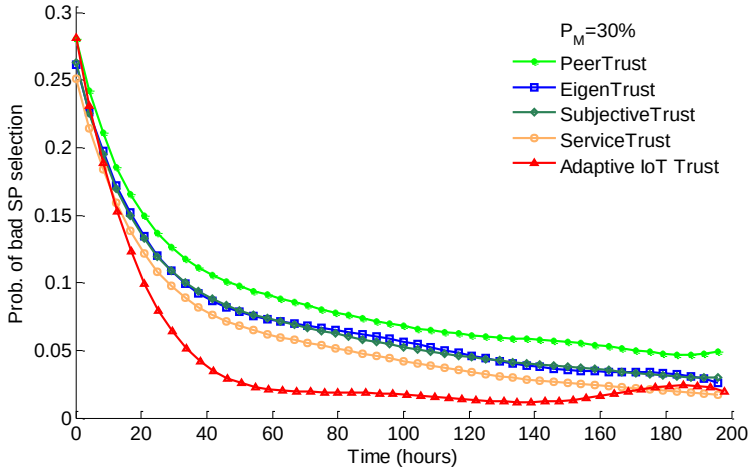


Figure 4.18: Probability of a bad SP being selected for Service Composition without Constraints.

Figure 4.17 shows the ns-3 simulation results with P_M=30%. We observe that trust-based service composition with our adaptive IoT trust protocol significantly outperforms random service composition and upon convergence approaches the performance of ideal service composition based on ground truth. Further, our adaptive IoT trust protocol outperforms ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] as the underlying trust protocol for trust-based service composition. In addition, we also observe that the performance gap widens as P_M increases.

Figure 4.18 shows the percentage of bad nodes selected for service composition without service constraints. Our adaptive IoT trust protocol again outperforms ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86].

We attribute the superiority of Adaptive IoT Trust over ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] to our protocol's adaptability to adjust the best trust parameter (μ) dynamically to minimize trust bias, and, consequently, maximize the performance of the service composition application.

4.3.3.2 Service Composition with Constraints

One example of service constraints is budget limit. Simply selecting the most trustworthy SPs may lead to infeasible solutions. Suppose that each SP announces its price when publishing the service and the SR has a budget limit for service composition. In *trust-based service composition with constraints*, the SR calculates the overall utility score and the overall price for each candidate configuration, using the trust value it has toward a SP to predict the utility score for that SP, and selects the configuration with the highest utility score among those with the overall price below the budget limit.

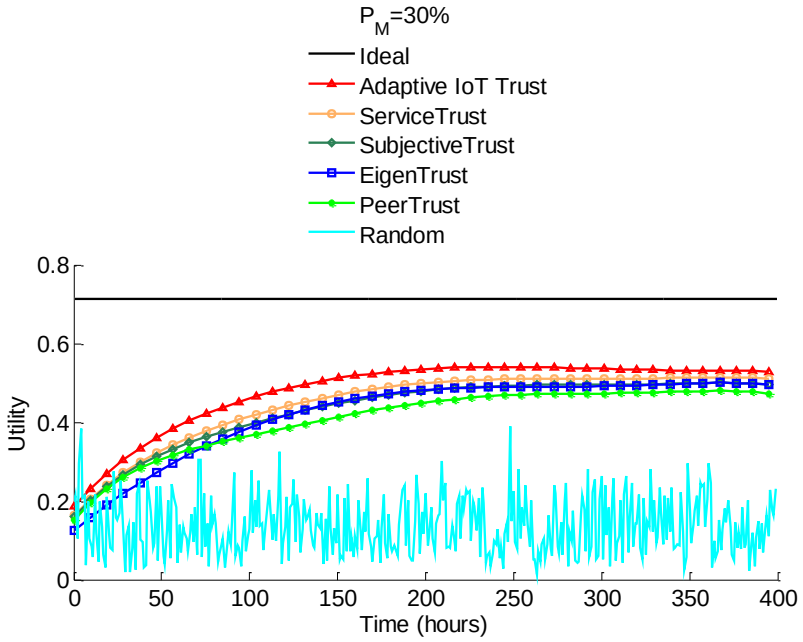


Figure 4.19: Utility of Service Composition with Constraints.

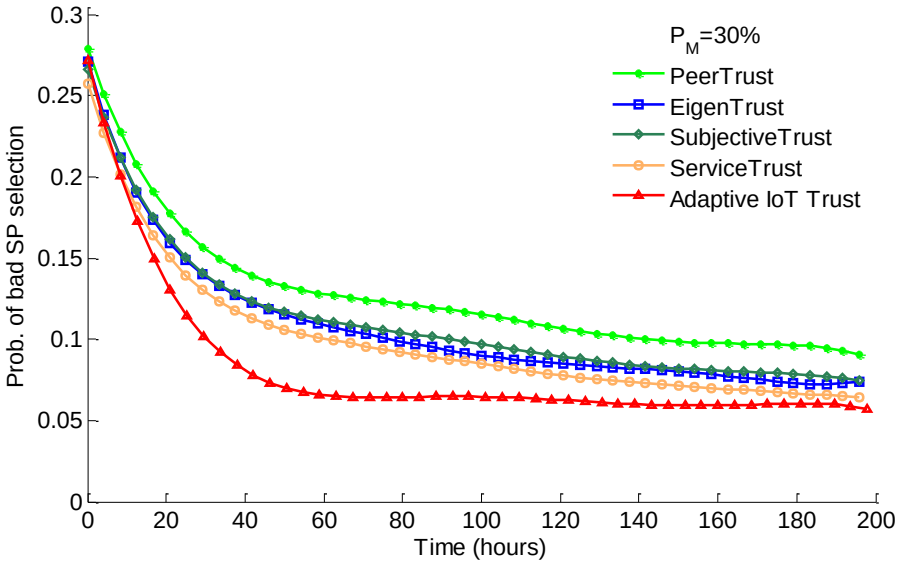


Figure 4.20: Probability of a bad SP being selected for Service Composition with Constraints.

Figure 4.19 shows the ns-3 simulation results with P_M=30%. We first observe that the utility scores are lower than those without budget constraints since good service providers may post high price, thus preventing them from being included. We again

observe that the trend is similar to Figure 4.17 in terms of performance ranking, with *trust-based service composition* with our adaptive IoT trust protocol outperforming that with ServiceTrust [94], EigenTrust [67], PeerTrust [102], or SubjectiveTrust [86]. Figure 4.20 shows the percentage of bad nodes selected for service composition with budget limit constraints. Our adaptive IoT trust protocol again outperforms ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] by a significant margin nearly cut in half in the percentage of bad nodes selected for service composition. We again attribute the superiority of our protocol over ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] to our protocol's adaptability in response to a high percentage of nodes performing malicious attacks.

4.3.3.3 Effects of Social Similarity on Trust Feedback

So far we have assumed $w_f = w_l = w_c = 1/3$ (in Equation (4.6)) for computing social similarity, considering there is an equal contribution from friendship, social contact, and CoI. However, in some application environments (say remote travel agent service) in which nodes that are friends or in the same CoI may be more credible than nodes that are co-located in providing trust feedback, while in another environment (say local restaurant service), it is the other way around. So there is an optimal weight assignment (w_f, w_l, w_c) that can provide the most credible trust feedback. In this section, we examine the effect of (w_f, w_l, w_c) on protocol performance with the service composition application with constraints as our test case.

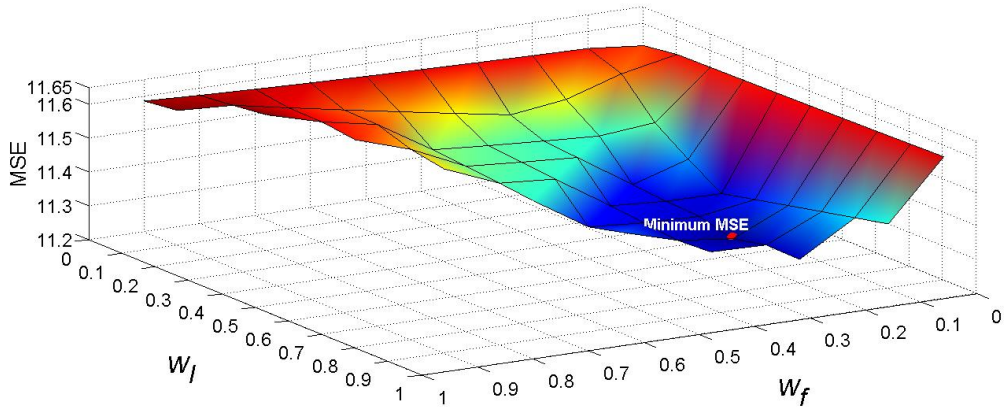


Figure 4.21: Mean Square Error of Utility Difference vs. (w_f,w_l,w_c). Minimum MSE at (w_f,w_l,w_c) = (0.2, 0.6, 0.2)

Figure 4.21 shows the simulation results of the MSE of the difference between the utility obtainable under trust-based service composition and the ideally achievable

utility for the service composition application vs. (w_f, w_l, w_c). Note that w_c=$1 - w_f - w_l$ and is not shown in the 3-D diagram. One can see clearly from Figure 4.21 that there exists an optimal weight assignment (w_f, w_l, w_c)= (0.9, 0.0, 0.1) under which MSE is minimized, i.e., the utility obtainable via trust-based service composition is closest to the ideally achievable utility with perfect global knowledge of node status. Here it is worth noting that the social contact similarity metric is not a factor in this application scenario for trust feedback because all services except one (restaurant in Figure 4.16) do not require social contact similarity. However, this is not universally true should another service composition flowchart be given as input.

4.4 Summary

In this chapter, we designed and analyzed an adaptive and scalable trust management protocol for SOA-based IoT systems. We developed a distributed collaborating filtering technique to select trust feedback from owners of IoT nodes sharing similar social interests. We considered three social relationships, i.e., friendship, social contact, and community of interest, for measuring social similarity and filtering trust feedback based on social similarity. Further, we developed an adaptive filtering technique by which each node adaptively adjusts its best weight parameters for combining direct trust and indirect trust into the overall trust to minimize convergence time and trust bias of trust evaluation. We demonstrated via simulation the superiority of our adaptive IoT trust protocol over ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86] in trust convergence, accuracy and resiliency against malicious nodes performing self-promoting, bad-mouthing, ballot-stuffing, and opportunistic service attacks.

For scalability we proposed a storage management strategy for small IoT devices to effectively utilize limited storage space. By using the proposed method, our trust protocol with limited storage space is able to achieve a similar performance level as that with unlimited storage space. To demonstrate the applicability, we applied our trust management protocol to a service composition application, with or without service constraints in SOA-based IoT systems. Our simulation results demonstrated that with our adaptive trust protocol design, the application running on top of the trust protocol is able to approach the ideal performance upon convergence and can significantly outperform the counterpart non-trust-based random selection service composition, as well as service composition running on top of ServiceTrust [94], EigenTrust [67], PeerTrust [102], and SubjectiveTrust [86]. We also demonstrated that our technique is effective in deciding and applying the best weight combination (w_f, w_l, w_c) for

combining social similarities that will lead to the most credible trust feedback to minimize trust bias and maximize the utility of the application.

Chapter 5 Centralized IoT Trust Management

A future Internet of Things (IoT) system will consist of a huge number of autonomous IoT devices capable of providing services upon request. It is of utmost importance for an IoT device to know if another IoT service is trustworthy when requesting it to provide a service. We propose a centralized scalable trust management protocol for supporting trust as a service (TaaS) in a large-scale IoT system. TaaS is made possible by a group of cloud servers whose size scales linearly with the number of IoT devices. In our trust protocol design, each user will map to a home cloud server using its unique id based on Distributed Hash Table techniques to achieve load balancing among all cloud servers. TaaS is realized by following a simple report-and-query paradigm. Specifically, a user upon a service completion will report to its home cloud server of the user satisfaction result. To know if another IoT device is trustworthy in providing a service, a user will send a query to its home cloud server even if the user has not had any service experience with the IoT device. The server will return a trust value formed by considering self-observations from the user as well as recommendations from other users with similar social interest. We conduct a cost analysis of our scalable trust management protocol to demonstrate its scalability to a large number of IoT nodes. We demonstrate our trust protocol's convergence, accuracy, and resiliency properties against self-promotion, discriminatory, bad-mouthing, and ballot-stuffing attacks. We prove that our protocol can achieve scalability without compromising subjective trust evaluation accuracy, when compared with contemporary IoT trust management protocols, including Adaptive IoT Trust [27] and ObjectiveTrust [86]. We also demonstrated its applicability by applying IoT-TaaS to two real-world IoT applications. Our results support its superiority over contemporary, non-scalable distributed subjective trust management protocols in selecting trustworthy nodes and resulting in the highest trustworthiness score of the application.

5.1 System Model

5.1.1 IoT TaaS Model

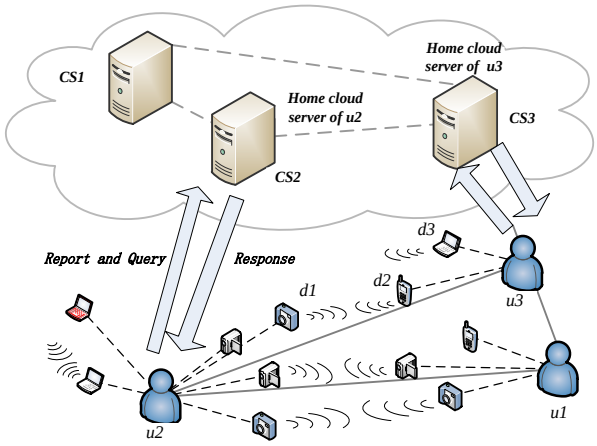


Figure 5.1: IoT TaaS Model.

5.1.2 Report-and-Query Model

TaaS is supported by a simple report-and-query model. The home cloud server of u_x is responsible for storing service experiences of u_x. Every time device u_x requests and subsequently receives a service from device $d_{i,}$ u_x reports its user satisfaction experience toward $d_{i,}$to its home cloud server. For example in Figure 5.1 when u_2 receives a service from d_2, it will report whether it is satisfied with the service of d_2 to its home cloud server CS_2. Every time u_x wants to know the service trustworthiness status of d_j, it can simply query its home cloud server. The home cloud server after consulting with other cloud servers will return the trust value requested. Afterward the user can make an application level decision, e.g., whether to select the device for providing a requested service. For example, in Figure 5.1, u_3 can simply send a query to its home cloud server CS_3 to query the subjective trust value of d_1. After retrieving service experience reports of u_3 toward d_1 stored in its local store, and communicating with other cloud servers for retrieving feedback reports of other users toward d_1 stored in other cloud servers, CS_3 can compute the subjective trust of u_3 toward d_1 (to be described later in Section 5.2 below), and return d_1's subjective trust value as a reply to u_3.

To facilitate subjective trust computation based on social similarity, user u_x after encountering user u_y also reports to its home cloud server social similarity between u_x and u_y in friendship, social contact, and community of interest. Hence, the home cloud server of u_x will store the social similarity information between u_x and u_y. This social similarity information stored subsequently allows u_x to discount or even filter out feedbacks from users (who serve as recommenders) it does not bear much social

similarity with. For example, in Figure 5.1 after u_2 encounters u_3, u_2 reports its social similarity with u_3 to u_2's home cloud server CS_2. Similarly u_3 reports its social similarity with u_2 to u_3's home cloud server CS_3. Here we note that for scalability, each IoT device does not keep a copy of any of these reports in its limited memory.

5.2 IoT-TaaS Protocol Description

In this section we describe IoT-TaaS protocol design. One can view it as the centralized version of Adaptive IoT Trust [27] developed in Chapter 4 as it preserves the notion of one-to-one "subjective" trust evaluation.

5.2.1 Reporting

Whenever a service is rendered, a user (using its primary IoT device) reports whether it is satisfied with the service provided by an IoT device to the user's home cloud server via a service rating report. Let the current user satisfaction experience of user u_x toward device d_i be represented by a value, $f_{x,i}$ which can be a real number in the range of 0 to 1 indicating the user satisfaction level, or simply a binary value, with 1 indicating satisfied and 0 not satisfied. Here $f_{x,i}$ is the first piece of information sent from u_x to its home cloud server. For example in Figure 5.1, u_3 will send $f_{3,1}$ to CS_3, the home cloud server of u_3, whenever a service is rendered by d_1. A timestamp is also sent in the report to indicate the time at which this service rating happens. This allows cloud servers to know the event occurrence times of reports for regression analysis if necessary.

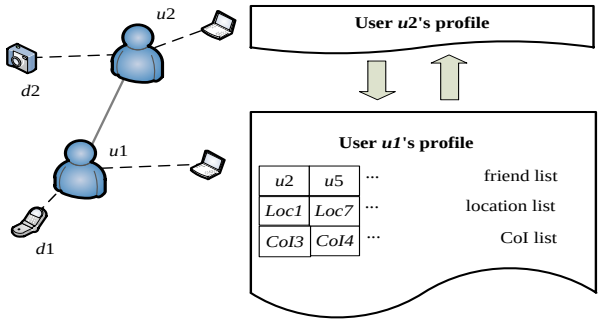


Figure 5.2: Friend, Location, and CoI Lists for Social Similarity.

When user u_x encounters user u_y they exchange their $(F_x, S_x C_x)$ profiles so as to measure their mutual social similarity. We adopt cosine similarity to measure the

distance of two social relationship lists (see Figure 5.2), with 1 representing complete similarity and 0 representing no similarity. Computational efficiency is the main reason why we choose cosine similarity to measure the similarity of two vectors in high-dimensional positive spaces because of limited computational capacity of IoT devices. To preserve privacy, one can use a one-way hash function (with session key) to encrypt friend/location/CoI lists so the comparison is performed on encrypted data to prevent the identities of uncommon friends/devices from being revealed. Specifically, the following three similarity metrics are measured as follows:

- **Friendship Similarity** (sim_f): The friendship similarity is a powerful social relationship (intimacy) for screening recommendations. After two users u_x and u_y exchange their friend lists, F_x and F_y, they could compute two binary vectors, $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$, each with size $|F_x \cup F_y|$. An element in $\overrightarrow{VF_x}$ (or $\overrightarrow{VF_y}$) will be 1 if the corresponding user is in F_x (or F_y), otherwise 0. Let $\|\vec{A}\|$ be the norm of vector $\vec{A}$ and $|B|$ be the cardinality of set B. Then, we could use the "cosine similarity" of $\overrightarrow{VF_x}$ and $\overrightarrow{VF_y}$ (giving the cosine of the angle between them) to compute sim_f as follows:

$$sim_f(u_x, u_y) = \frac{\overrightarrow{VF_x} \cdot \overrightarrow{VF_y}}{\|\overrightarrow{VF_x}\| \|\overrightarrow{VF_y}\|} = \frac{|F_x \cap F_y|}{\sqrt{|F_x| \cdot |F_y|}} \tag{5.1}$$

- **Social Contact Similarity** (sim_s): The social contact similarity represents closeness and is an indication if two nodes have the same physical contacts and thus the same sentiment towards devices which provide the same service. The operational area could be partitioned into sub-grids. User u_x records the IDs of sub-grids it has visited in its *location list* S_x for social contact. After two users u_x and u_y exchange their location lists, S_x and S_y, they could compute sim_l in the same way of computing sim_f as follows:

$$sim_s(u_x, u_y) = \frac{|S_x \cap S_y|}{\sqrt{|S_x| \cdot |S_y|}} \tag{5.2}$$

- **Community of Interest Similarity** (sim_c): Two users in the same CoI share similar social interests and most likely have common knowledge and standard toward a service provided by the same device. Also very likely two users who have used services provided by the same IoT device can form a CoI (or are in the same CoI). After two users u_x and u_y exchange their device lists, C_x and C_y, they could compute sim_c in the same way of computing sim_f as follows:

$$sim_c(u_x, u_y) = \frac{|C_x \cap C_y|}{\sqrt{|C_x| \cdot |C_y|}} \tag{5.3}$$

The above three social similarity measures (sim_f, sim_s, sim_c) are computed upon encountering of user u_x and user u_y, and are stored in the home cloud servers of user u_x and user u_y. For example, in Figure 5.1 after u_2 encounters u_3 they will each compute the three social similarity measures (sim_f, sim_s, sim_c) through Equations(5.1), (5.2) and (5.3) and store the results in the home cloud servers CS_2 and CS_3, respectively.

When the home cloud server of u_x receives $sim_i(u_x, u_y)$, $i \in \{f, s, c\}$, from user u_x which just encounters user u_y, it computes the social similarity between users u_x and u_y (who now serves as a rater or recommender) as a weighted combination of all social similarity metrics, i.e., friendship, social contact, and community of interest, as follows:

$$sim(u_x, u_y) = \sum_{i \in \{f,s,c\}} w_i \cdot sim_i(u_x, u_y) \tag{5.4}$$

where $0 \leq w_f, w_s, w_c \leq 1$, with $w_f + w_s + w_c = 1$, are social similarity weight parameters to be dynamically adjusted by IoT-TaaS to maximize trust protocol performance. This is to be analyzed in Section 5.3.

5.2.2 Querying and Replying

Whenever a user wants to know the trust value of an IoT device, it simply sends a query to its home cloud server. For example, in Figure 5.1, u_2 will send a query to its home cloud server CS_2 to know the "subjective" trust value of user u_2 toward d_2 which belongs to u_3.

Let the "subjective" trust value of user u_x toward d_i be denoted by $t_{x,i}$. The home cloud server of u_x computes $t_{x,i}$ by combining $u_x's$ direct trust toward d_i ($t_{x,i}^d$) based on self-observation reports, and $u_x's$ indirect trust toward d_i ($t_{x,i}^r$) based on other users' ratings, as follows:

$$t_{x,i} = \mu_{x,i} \cdot t_{x,i}^d + (1 - \mu_{x,i}) \cdot t_{x,i}^r \tag{5.5}$$

Here, $\mu_{x,i}$ is a weight parameter ($0 \leq \mu \leq 1$) to weigh the importance of direct trust relative to indirect trust. The selection of $\mu_{x,i}$ is critical to trust evaluation. We apply adaptive filtering [27][62] to adjust $\mu_{x,i}$ dynamically to effectively cope with malicious attacks and to improve trust evaluation performance, which we will discuss in more detail in Section 5.2.3.

To compute direct trust $t_{x,i}^{d}$, we adopt Bayesian framework [65] as the underlying model. The reason we choose Bayesian because it is well-established and because of its popularity in trust/reputation systems. In service computing, a service requester would rate a service provider after a service is rendered based on nonfunctional characteristics. The nonfunctional characteristics include user-observed service delay, service quality received, prices, etc. Then, we can consider the service rating $f_{x,i}$ as a Bernoulli trial with the probability of success parameter $\theta_{x,i}$ following a Beta distribution (a conjugate prior for the Bernoulli distribution), i.e., Beta($\alpha_{x,i}$, $\beta_{x,i}$). Then, the posterior p($\theta_{x,i} | f_{x,i}$) has a Beta distribution as well. The model parameters $\alpha_{x,i}$ and $\beta_{x,i}$ are updated as follows:

$$\begin{aligned} \alpha_{x,i} &= \alpha_{x,i}^{(old)} + f_{x,i} \\ \beta_{x,i} &= \beta_{x,i}^{(old)} + 1 - f_{x,i} \end{aligned} \tag{5.6}$$

In Equation (5.6), $f_{x,i}$ contributes to positive service experience and $1 - f_{x,i}$ contributes to negative service experience. The direct trust $t_{x,i}^{d}$ of user u_x toward device d_i then can be computed as the expected value of $\theta_{x,i}$, i.e.,

$$t_{x,i}^{d} = E\left[\theta_{x,i}\right] = \frac{\alpha_{x,i}}{\alpha_{x,i} + \beta_{x,i}} \tag{5.7}$$

Specifically, the home cloud server of u_x updates $\alpha_{x,i}$and $\beta_{x,i}$whenever it receives $f_{x,i}$ (a service rating report) from user u_x based on Equation (5.6) and then computes $t_{x,i}^{d}$ based on Equation (5.7).

To compute indirect trust $t_{x,i}^{r}$, the home cloud server of u_x first locates social similarity records $sim\left(u_x, u_y\right)'s$ in its local storage. The home cloud server of u_x then selects top-R recommendations from R users with the highest similarity values with u_x and calculates the indirect trust ($t_{x,i}^{r}$) towards device d_i as follows:

$$t_{x,i}^{r} = \sum_{u_y \in U} \frac{sim\left(u_x, u_y\right)}{\sum_{u_z \in U} sim(u_x, u_z)} \cdot t_{y,i}^{d} \tag{5.8}$$

Here, U is a set of up to R users whose $sim(u_x, u_y)$ values are the highest, and $t_{y,i}^{d}$ is the rating or recommendation provided by user u_y toward device d_i, which is stored in the home cloud server of u_y but obtainable after the home cloud server of u_x communicates with the home cloud server of u_y.

In Equation (5.8), the feedback from u_y toward d_i $\left(\text{i.e.}, t_{y,i}^{d}\right)$ is weighted by the ratio of the similarity score toward the rater to the sum of the similarity scores toward all

raters. Here we note that if u_y is malicious, then it can provide $t_{y,i}^d$=0 against a good device for bad-mouthing attacks, and $t_{y,i}^d$=1 for a bad node for ballot-stuffing attacks.

5.2.3 Dynamic Control to Minimize Trust Bias

As in Adaptive IoT Trust [27], we dynamically control $\mu_{x,i}$ (with $0 \leq \mu_{x,i} \leq 1$) in Equation (5.5) to weigh the importance of direct trust $t_{x,i}^d$ relative to indirect trust $t_{x,i}^r$ so as minimize trust bias in the centralized cloud environment. We first note that $\mu_{x,i}$ is a one-to-one parameter, i.e., it is from user u_x to device d_i. Further, the dynamic control function is performed by $u_x's$ home cloud server during a query-and-reply process. We apply adaptive filtering [27][62] to adjust $\mu_{x,i}$ dynamically in order to effectively cope with malicious attacks. The basic design principle is that a successful trust management protocol should provide high trust toward devices who have more positive user satisfaction experiences and, conversely, low trust toward those with more negative user satisfaction experiences. Specifically, the current trust evaluation (i.e., $t_{x,i}(\mu_{x,i})$ as a function of $\mu_{x,i}$) should be as close to the average user satisfaction experiences observed by user u_x toward d_i over the last trust update window Δt as possible. Therefore, we formulate the selection of $\mu_{x,i}$ as an optimization problem as follows:

$$\begin{aligned} &\text{Find: } \mu_{x,i}, 0 \leq \mu_{x,i} \leq 1 \\ &\text{Minimize: } \mathrm{MSE}(\mu_{x,i}) = \textstyle\sum_{\Delta t}\big(t_{x,i}(\mu_{x,i}) - f_{x,i}\big)^2 \end{aligned} \tag{5.9}$$

Here, $t_{x,i}(\mu_{x,i})$ is obtained from Equation (5.5), and $f_{x,i}$ is a service rating reported by user u_x within the last trust update interval Δt. The objective can be achieved by minimizing the mean square error (MSE) of trust evaluations against actual user satisfaction experiences by user u_x toward d_i over the last trust update window Δt, such that the trust value could be a good indicator or predictor for quality of service (with direct user satisfaction experiences considered as ground truth). After plugging in the expression of $t_{x,i}(\mu_{x,i})$ from Equation (5.5) to Equation (5.9) and setting the derivative to zero, we obtain the optimal value of $\mu_{x,i}$, denoted by $\widetilde{\mu_{x,i}}$, at which $\mathrm{MSE}(\mu_{x,i})$ is minimized, as:

$$\widetilde{\mu_{x,i}} = \frac{\sum_{\Delta t}\big(f_{x,i} - t_{x,i}^r\big)\big(t_{x,i}^d - t_{x,i}^r\big)}{\sum_{\Delta t}\big(t_{x,i}^d - t_{x,i}^r\big)^2} \tag{5.10}$$

The true optimal value of $\mu_{x,i}$ (i.e., $\widehat{\mu_{x,i}}$) should be in the range of [0, 1] because it is a weight parameter. Therefore,

$$\widehat{\mu_{x,\iota}} = \begin{cases} 0 & \widetilde{\mu_{x,\iota}} < 0 \\ \widetilde{\mu_{x,\iota}} & 0 \le \widetilde{\mu_{x,\iota}} \le 1 \\ 1 & \widetilde{\mu_{x,\iota}} > 1 \end{cases} \quad (5.11)$$

More specifically, $u_x's$ home cloud server computes the optimal value of $\mu_{x,i}$ (i.e., $\widehat{\mu_{x,\iota}}$) and dynamically updates it in every trust update time interval Δt based on Equations (5.10) and (5.11), using the historical rating and trust data stored in its storage, so there is essentially no extra overhead.

Here we against note that our dynamic control design is driven by minimizing the difference between the subjective trust $t_{x,i}(\mu_{x,i})$ and the new user satisfaction experiences $f_{x,\iota}'s$ obtained in the last trust update interval Δt. If d_i is a malicious node and it retains high trust either because it performs opportunistic service attacks to gain high trust, or because other nodes provide ballot-stuffing attacks to boost its trust, then our trust system will be temporarily deceived of its true status because the difference between these two quantities will be small. However, as soon as d_i performs self-promoting attacks and provides bad service to user u_x, this bad user experience will be immediately observed by user u_x and, as a result, the difference between these two quantities will be large enough to drive the change of $\mu_{x,i}$ to minimize $\mathrm{MSE}(\mu_{x,i})$ in Equation (5.9).

5.3 IoT-TaaS Protocol Performance

In this section, we analyze performance characteristics of IoT-TaaS and compare IoT-TaaS performance with two baseline IoT trust protocols, Adaptive IoT Trust [27] and ObjectiveTrust [86]. See Chapter 2 Related Work for the detail of these two baseline trust protocols chosen for our comparative analysis.

5.3.1 Environment Setup

Table 5.1: Parameter List for Performance Evaluation.

Parameter	Meaning	Default
N_T	Number of IoT devices	2000
N_u	Number of users	500
N_C	Number of cloud servers	10
P_M	% of malicious users	30%
P_C	% of high centrality nodes	0%
λ	Service request rate per node	1/day
σ_c	Standard deviation of error	1%
$m \times m$	Regional area	16×16
$SWIM_S$	SWIM slope	1.45

$SWIM_{pt}$	SWIM maximum pause time	4 hrs
$SWIM_{npp}$	SWIM # popular places per node	[10, 25]
$n_{f,lc}$	# friends per low-centrality node	[10, 50]
$n_{f,hc}$	# friends per high-centrality node	[100, 500]
N_{CoI}	# communities of interest	50
n_{CoI}	# communities of interest per node	[10, 25]
n_r	# of recommenders accepted	10
$w_f : w_s : w_c$	Weight ratio of social relations	1/3:1/3:1/3

The experiment setup follows the model parameters as listed in Table 5.1 and is explained as follows. We consider a large IoT $m \times m$ regional area with N_T = 2000 IoT devices, where each region (location) is also a square with the width and height equal to wireless radio range so that nodes within a grid can communicate with the local region in the grid. The boundary locations are wrapped around (i.e., a torus is assumed) to avoid end effects. These IoT devices are randomly assigned to N_U = 500 users, with each user having 4 devices on average. The number of cloud servers in the data cloud center is N_C = 10 such that each cloud server can approximately handle N_T/N_C = 200 IoT devices.

The % of malicious users (P_M) is a model parameter whose effect will be analyzed via a sensitivity analysis. Malicious users are chosen randomly from N_U = 500 users. A malicious user will perform self-promoting, bad-mouthing, ballot-stuffing, discriminatory, and opportunistic service attacks as described in Chapter 3.3. In particular, a malicious user u_y can provide a bad recommendation $t_{y,i}^d$=0 (see Equation (5.8)) against a good device i for bad-mouthing attacks, and conversely a good recommendation $t_{y,i}^d$=1 for a malicious device i for ballot-stuffing attacks. Our protocol handles ballot-stuffing and bad-mouthing attacks by recommendation filtering during the computation of the indirect trust $t_{x,i}^r$ in Equation (5.8).

The % of high centrality users (P_C) is also a model parameter whose effect will be analyzed. Users are connected through social networks represented by a friendship matrix and a CoI matrix where an entry xy is 1 meaning that user x and user y are friends and members of a CoI, respectively. Each low centrality user has $n_{f,lc}$ = [10, 50] friends, and each high centrality user has $n_{f,hc}$ = [100, 500] friends populated randomly. There are N_{CoI} = 50 CoIs and each user has n_{CoI} = [10, 25] CoIs. We consider these users moving according to the small world in motion (SWIM) mobility model [71] modeling human social behaviors for the purpose of assessing the social contact similarity metric between any pair of users. In SWIM [71], a node has a home location and $SWIM_{npp}$= [10, 20] popular places populated randomly out of the $m \times m$ locations in the system. A node makes a move to one of the population places based on a prescribed pattern. The

probability of a location being selected is higher if it is closer to the node's home location or if it has a higher *popularity* (visited by more nodes). When reaching the destination, the node pauses at the destination location for a period of time following a bounded power law distribution. We set the slope of the SWIM mobility model ($SWIM_S$) to 1.45 (as in [71]) and the upper-bound pause time ($SWIM_{pt}$) to 4 hours. The encounter time interval for any two nodes is a bounded power-law distribution between [10 minutes, 2 days], which models the social contact behavior of any two nodes.

Direct trust of node i toward node j is assessed upon completion of a service request from node i to node j. Each node requests services from a selected device with a time interval following an exponential distribution with parameter λ, with 1/day being the default unless otherwise specified. The trust update interval Δt is 2 hours. The system runs continuously although trust convergence is achieved in less than 200 hours.

The user satisfaction levels of service experiences, i.e., $f_{x,i}$ in the range of [0, 1] from user u_x about d_i's service quality, are from a real web service dataset [107] and are used as "ground truth" based on which the accuracy of our trust protocol is assessed. Since the direct trust of user u_x toward service provider d_i (i.e., $t_{x,i}^d$) is calculated by Equation (5.5) with "ground truth" user service experiences as input, $t_{x,i}^d$ essentially is equal to ground truth. However, we account for the presence of noise in the IoT environment (i.e., error of assessing user satisfaction level received) by considering a standard deviation parameter σ_c (set to 1% as default) to reflect the deviation of the actual user satisfaction level recorded in the database from the direct trust evaluation outcome $t_{x,i}^d$. Initially, $t_{x,i}$ is set to 0.5 (ignorance) by user u_x for all i's. Then, trust is updated dynamically as nodes encounter each other, as services are requested and rendered, and as trust feedback are acquired. n_r =10 for the set size of U in Equation (5.8) for calculating $t_{x,i}^r$. We consider $w_f = w_l = w_c = 1/3$ considering friendship, social contact, and community of interest are equally important.

5.3.2 Cost Analysis

In this section, we perform a cost analysis of the communication and storage requirements for IoT-TaaS. For comparison, we also perform a cost analysis of Adaptive IoT Trust [27] for cost performance comparison to demonstrate the scalability of IoT-TaaS.

The communication cost per node (from node i's perspective) for IoT-TaaS is of complexity $O(\sum_{j=1}^{N_T} \lambda_{ij} + q_{ij} + \pi_{ij})$ where λ_{ij} is the service request rate of node i to node j, q_{ij} is the query rate of node i about node j's trust status, and π_{ij} is the encountering rate of node i with node j which can be derived by analyzing the encounter or interaction

pattern, e.g., a power-law distribution, as supported by the analysis of many real traces [12]. Upon requesting and receiving a service from node *j*, node *i* sends its service quality assessment to its home cloud server. Whenever node *i* wants to know node *j*'s trust status, it needs to send a query to its home cloud server. Upon encountering node *j*, node *i* exchanges its user profile with node *j's* for computing social similarity (i.e., through hashed friend, social contact, and community-interest lists) and then sends computed social similarity values to its home cloud server. The one extra message needed for sending computed social similarity values to its home cloud server does not change the complexity. For Adaptive IoT Trust [27], the communication cost per node (from node *i*'s perspective) is just $O(\sum_{j=1}^{N_T} \pi_{ij})$ because node *i* stores all service quality assessment results as well as social similarity results in its local storage. When node *i* needs to know node *j*'s trust status it simply looks up its trust value in the local storage. Comparing IoT-TaaS with Adaptive IoT Trust [27], the extra message cost for IoT-TaaS (from a user's perspective) is for storing and querying trust data, the magnitude of it is proportional to node encountering, service request, and trust status query rates.

The storage cost per node (from node *i*'s perspective) for IoT-TaaS is O(1) for storing the user profile (i.e., friend, social contact, and community-interest lists) since all user satisfaction experiences, social similarity, and trust data are stored in the cloud. The storage cost per cloud server is $O(N_T/N_C)$ where N_T is the number of IoT devices and N_C is the number of cloud servers since the load is shared by all servers (through the use of a fair hash function). For Adaptive IoT Trust [27], the storage cost per node is $O(N_T)$. For each IoT device, a storage space is needed for storing user satisfaction experiences, social similarity, and trust data. Apparently Adaptive IoT Trust is not scalable when N_T is sufficiently large.

5.3.3 Trust Convergence, Accuracy and Resiliency against Malicious Attacks

In this section, we examine the trust convergence, accuracy and resiliency properties of our IoT-TaaS protocol design. We first compare static control (i.e., $\mu_{x,i}$ is fixed at a constant) vs. dynamic control (i.e., $\mu_{x,i}$ is changed dynamically based on Equation (5.10)). When the context is clear (which trustor x and which trustee i), we simply call $\mu_{x,i}$ as μ henceforth.

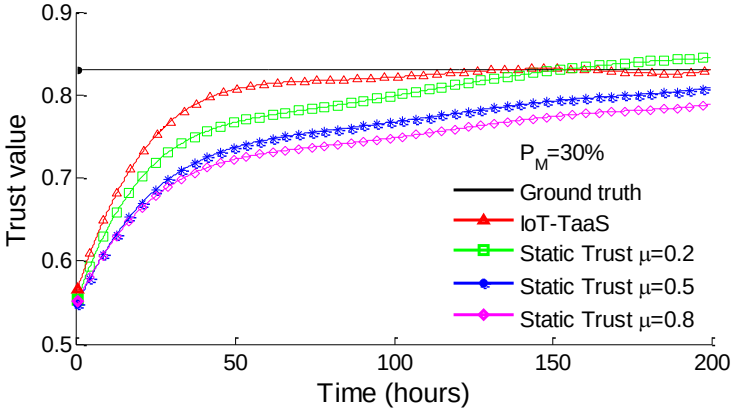


Figure 5.3: Trust Value of a Good Node with PM = 30%.

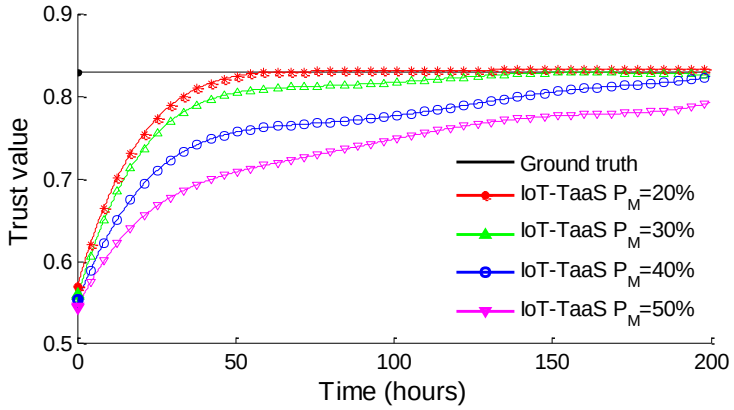


Figure 5.4: Trust Value of a Good Node under varying PM.

Figure 5.3 shows trust evaluation results for a trustor node toward a "good" trustee node randomly picked. We see that trust convergence behavior is observed for either static or dynamic control. There is a tradeoff between convergence time vs. trust bias. With static control, when a higher μ value is used, the trust convergence time is longer, but the trust bias is smaller, i.e., the trust value is closer to ground truth after convergence. With dynamic control, on the other hand, the trustor node is able to adjust μ dynamically to minimize both the convergence time and the trust bias after convergence. Here we note that the trust value of a "good" trustee is not 1 because we use the user satisfaction levels of service invocations based on a real dataset [107] with a standard deviation parameter σ_c (set to 1% as default) reflecting the deviation of the actual user satisfaction level recorded in the database from the direct trust evaluation outcome.

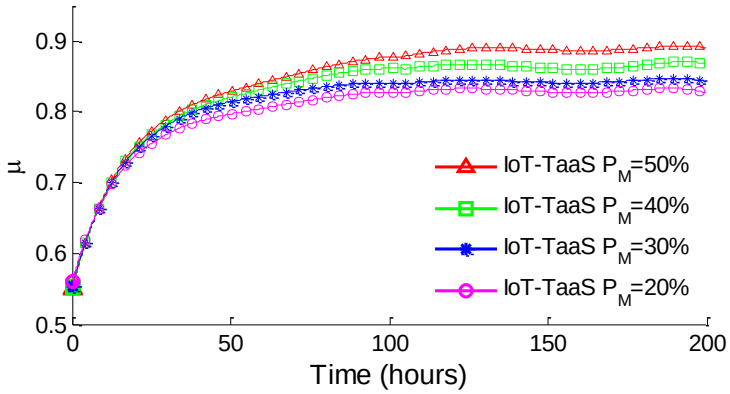


Figure 5.5: Adjustment of μ against Increasing Malicious Node Population.

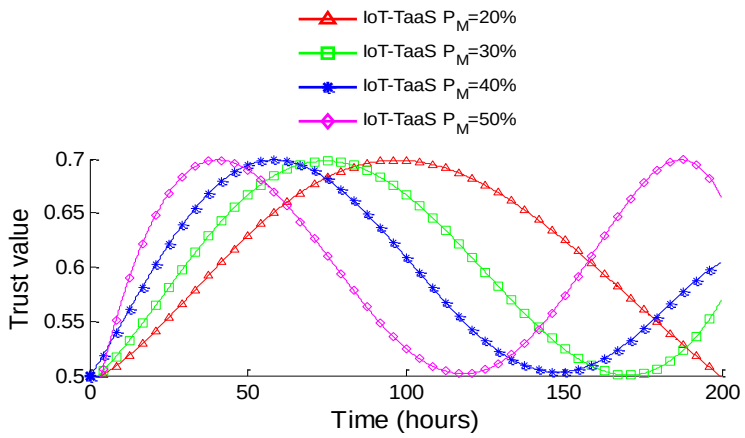


Figure 5.6: Trust Value of a Bad Node under varying P_M.

An interesting observation in Figure 5.3 is that if μ is too small (e.g., 0.2) the trust value is over-estimated upon convergence, which is not a desirable outcome as trust overshoot is considered a bad property detrimental to the stability of a trust system [43]. Our protocol dynamically adjusts μ for fast convergence without incurring trust overshoot.

Next we conduct experiments to test the residency of our trust protocol against increasing malicious node population. Figure 5.4 shows that as the population of malicious nodes increases, both the convergence time and trust bias increase. However, the system is found to be resilient to malicious attacks for P_M as high as 40%, with proper convergence and accuracy behaviors exhibited. In general we observe that the trust bias is insignificant (less than 3%) when $P_M \leq 40\%$ and the trust bias becomes more significant (larger than 10%) when $P_M \geq 50\%$. This demonstrates the resiliency property of our trust protocol against malicious attacks.

Correspondingly, Figure 5.5 shows how IoT-TaaS adjusts μ in Equation (5.10) in response to increasing malicious node population. We observe that as the malicious node population increases, the system will have to rely more on direct trust by increasing μ and conversely rely less on indirect trust by decreasing $1-\mu$ so as to mitigate the effect of bad-mouthing and ballot-stuffing attacks by malicious nodes.

Figure 5.5 shows that when P_M = 20%, the optimal converged μ value is 0.8, while when P_M = 50%, the optimal converged μ value is 0.9. When a node is being observed maliciously, its trust value will be reduced quickly because in this case a high μ value will be used by our trust protocol and a high μ value means that the trust value of the malicious node will be very close to direct trust which is low as the node is being observed maliciously. Conversely, when a node is being observed cooperatively, its trust value will just go up slowly because in this case a low μ value will be used by our dynamic control protocol and a low μ value means that both direct trust and indirect trust will contribute to the overall trust based on Equation (5.5). Although in this case, the direct trust observed is high as the node is being observed cooperatively, it will only increase the overall trust value slowly by a weight of μ, with the indirect trust contributing to the overall trust by a weight of $1-\mu$. The system cannot rely on direct trust 100% because malicious nodes can perform opportunistic service attacks and there is an error of assessing direct trust due to noise in the environment. Figure 5.5 demonstrates that our dynamic control mechanism is effective in terms of convergence of μ to its optimal value under which trust bias is minimized.

Figure 5.6 shows trust evaluation results for a trustor node toward a "bad" trustee node. Among all attacks, the bad node performs *opportunistic service attacks* with the high trust threshold being 0.7 and the low trust threshold being 0.5. Specifically, the bad node provides good service to gain high reputation opportunistically when it senses its reputation drops below 0.5. Once it reputation rises to 0.7, it provides bad service again. We see from Figure 5.6 that IoT-TaaS is able to accurately track the trust fluctuation of the bad node performing opportunistic service attacks. We observe that the rate of trust fluctuation is higher when P_M is higher because more malicious nodes can collude to quickly bring the trust level of the bad node to 0.7.

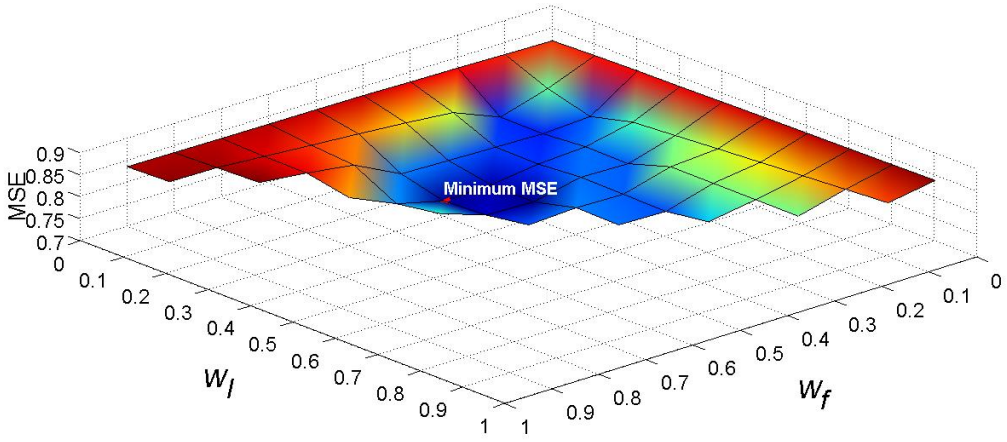


Figure 5.7: Effect of Social Similarity Weights on Protocol Performance.

Figure 5.7 shows the simulation results of the MSE of trust bias (i.e., the difference between the calculated trust and ground truth) vs. (w_f, w_l, w_c) for a "good" trustee node randomly chosen. Note that w_c=1 $- w_f - w_l$ and is not shown in the 3-D diagram. One can see clearly from Figure 5.7 that there exists an optimal weight assignment (w_f, w_l, w_c)= (0.5, 0.3, 0.2) under which MSE is minimized, i.e., the subjective trust obtained from IoT-TaaS is closest to the ground truth. The optimal weight assignment largely depends on a trustee node's social profile which can be learned by a user's home cloud server which aggregates the user's own assessment results stored locally with other users' assessment results toward the trustee node stored remotely in other cloud servers.

5.3.4 Sensitivity and Comparative Analysis

In this section, we report the sensitivity of IoT-TaaS performance with respect to the % of malicious users (P_M) and the % of high centrality users (P_C). We compare IoT-TaaS performance with two baseline trust protocols, Adaptive IoT Trust [27] and ObjectiveTrust [86], which we have described in detail in Chapter 2.

We first analyze the effect of the % of malicious users (P_M). Figure 5.8 compares IoT-TaaS (red surface) with Adaptive IoT Trust (orange surface) and ObjectiveTrust (green surface) as P_M (% of malicious nodes) varies from 20% to 40% for a non-malicious node randomly chosen. All other parameters use the default settings as in Table 5.1. The ground truth trust value of the non-malicious node is marked by a solid line at 0.83. We first note that for all protocols, the estimated trust score of the non-malicious nodes approaches the ground truth value as time progresses as more evidence is collected. Also, the deviation from the ground truth trust score increases as P_M increases because

more malicious nodes perform attacks. We see that IoT-TaaS (red) consistently outperforms Adaptive IoT Trust (orange) and ObjectiveTrust (green), especially when the % of malicious nodes is high (40%). We attribute IoT-TaaS performing better than Adaptive IoT Trust to the fact that IoT-TaaS can leverage cloud service to aggregate broad evidence from all nodes to achieve the minimum trust bias. We attribute IoT-TaaS performing better than ObjectiveTrust to the fact that unlike ObjectiveTrust, IoT-TaaS considers "subjective trust" which takes a customer's own service experiences into consideration and it can dynamically adjust the weight associated with won service experiences to effectively cope with malicious attacks and to improve trust accuracy, as it computes the trust score of a customer toward a target service provider.

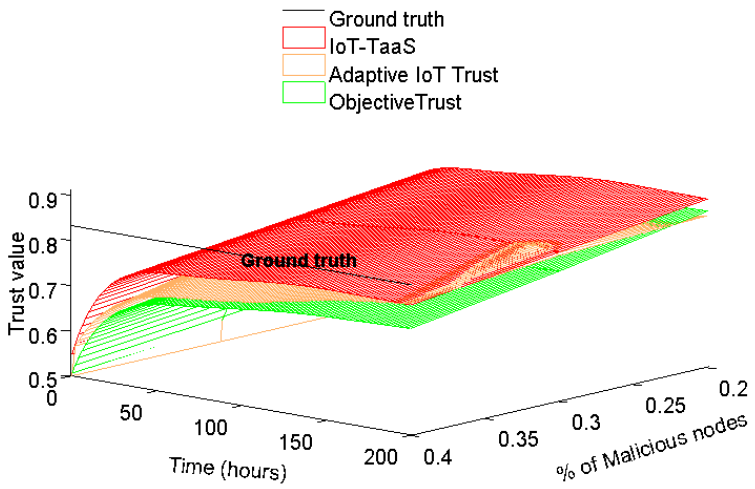


Figure 5.8: Performance Comparison of Trust Convergence, Accuracy and Resiliency in 3-D View with P_M (% of Malicious Nodes) ranging from 20% to 40%.

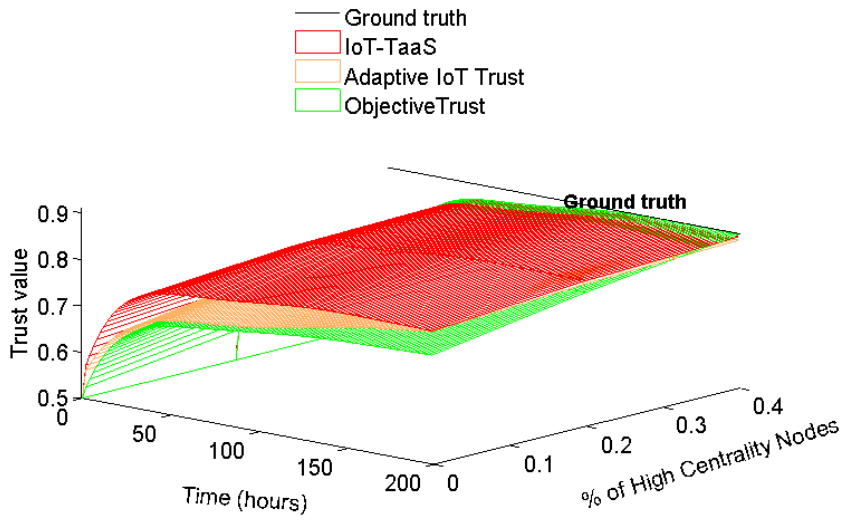


Figure 5.9: Performance Comparison of Trust Convergence, Accuracy and Resiliency in 3-D View with P_C (% of High Centrality Nodes) ranging from 0% to 40%.

Next we analyze the effect of % of high centrality users (P_C). Figure 5.9 compares IoT-TaaS (red surface) with Adaptive IoT Trust (orange surface) and ObjectiveTrust (green surface) as P_C (% of high centrality nodes) varies from 0% to 40% for a non-malicious node randomly chosen. All other parameters use the default settings as in Table 5.1. We see IoT-TaaS (red) also consistently performs better than Adaptive IoT Trust (orange) and ObjectiveTrust (green), especially when P_C is low. All protocols perform comparably when P_C is high. The reason is that all protocols consider social relationships (including friendship which maps to centrality) for recommendation filtering. So when P_C is high, all have excellent recommendation filtering taking place, resulting in the estimated trust score approaching the ground truth trust value. When P_C is extremely high (40%), ObjectiveTrust has a slight edge over IoT-TaaS and Adaptive IoT Trust because ObjectiveTrust uses a weighted sum of centrality and filtered opinions for the overall trust computation, and directly uses centrality (by means of the overall trust score computation) as credibility for recommendation filtering. One should note that, however, a social network normally has only a small number of nodes having high centrality [87], so for a typical social network with less than 20% high centrality nodes, IoT-TaaS (red) shall outperform ObjectiveTrust (green).

5.4 Applicability to Real World IoT Applications

In this section, we apply IoT-TaaS to two real-world IoT applications. We again compare IoT-TaaS performanace against Adaptive IoT Trust [27] and ObjectiveTrust [86] in our performance comparative analysis.

5.4.1 Case Study 1: Trust-based IoT Cloud Participatory Sensing of Air Quality

In this section, we present an IoT participatory sensing application [1], [70], [85] where IoT devices (e.g., smart phones carried by humans or smart cars driven by humans) can act as participants to collect air quality data and submit to a processing center located in the cloud for environmental data analysis. It is especially applicable to a health IoT group where the main concern is about a pollutant (O3 in our case study). Users in the group report their O3 sensing results upon receiving a query from a member who wishes to find out a location's O3 level at a particular time to decide if it should enter the location based on its susceptibility to the O3 level detected.

The major challenge in IoT cloud participatory sensing is the selection of trustworthy participants because not all IoT devices will be trustworthy and some IoT devices may behave maliciously to disrupt the network or service (e.g., in a terrorist attack scenario) or just for their own gain (e.g., in an evacuation scenario following a disaster). We leverage IoT-TaaS to addresses the issue of selecting trustworthy participants. We compare the performance of IoT-TaaS against two contemporary trust protocols, Adaptive IoT Trust [27] and ObjectiveTrust [86] in selecting trustworthy participants. We show that IoT-TaaS can provide better performance than these contemporary IoT trust protocols. Using real traces of O3 levels and mobility traces of users in the O3 community of interest (O3COI) group in the city of Houston [88], we demonstrate that with the help of a TaaS cloud utility with IoT-TaaS as the underlying trust protocol, a user in this O3COI group is able to obtain close to the ground truth O3 level.

We use real traces of O3 levels and mobility traces of users in the O3 community of interest (O3COI) group in the city of Houston and apply it to our participatory sensing case study. The original dataset in [88] covers the socio-demographically relevant activity sequences and the movements of each individual in 4.9 million synthetic individuals in the Houston metropolitan area. We extract a portion of this huge database to cover a smaller set of members in the O3COI group along with their mobility and activity data around a smaller area. Figure 5.10 shows the synthetic individuals in the Houston metropolitan area and the zoomed view of a small region covering the locations of the target user. The coordinates in the figure represents the longitude and latitude of each synthetic individual. The zoomed area is divided into 8x8

regions. The red curve in the zoomed area represents the mobility of the target user. In the case study, we assume a percentage of nodes, denoted by P_M in the range of [0, 30%], are malicious. Every day this "good" member issues queries to its home cloud server before it enters a particular location to know the O3 level in the location it is about to step into. After collecting a number of O3 reports from other members, it then performs a trust-weighted computation to deduce the O3 reading (described later in this section). If the O3 level is below a threshold, it would follow its route; otherwise, it will not enter the location or it will detour to avoid the location because the location has a high O3 level that can harm its owner's health. After the query-and-response event is completed, this "good" member will assess if an O3 sensing report submitted by another member is satisfactory and will submit the service experience to its home server so as to facilitate the implementation of the TaaS cloud utility with IoT-TaaS as the underlying trust protocol.

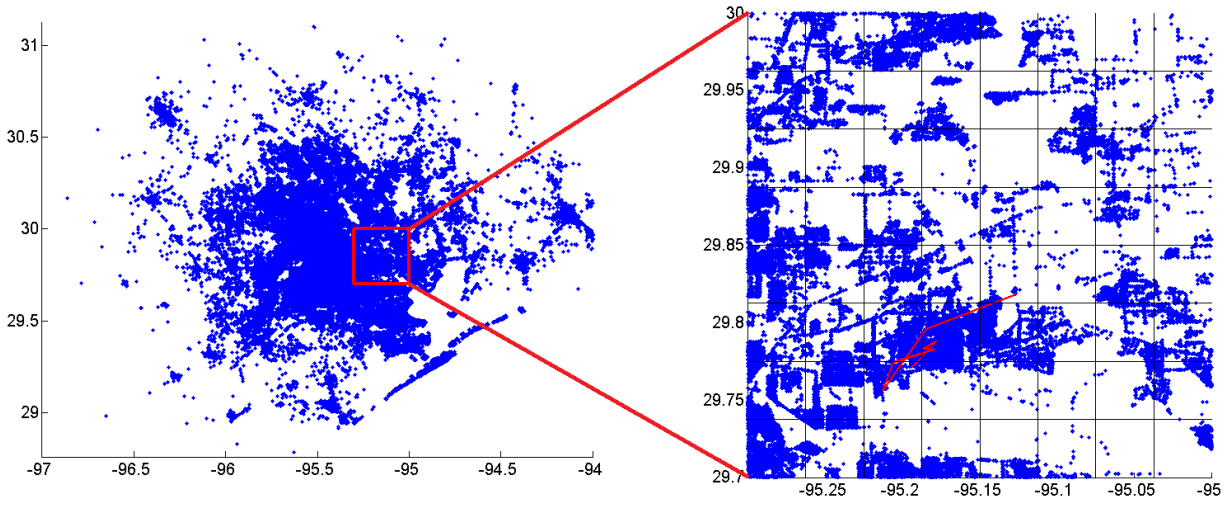


Figure 5.10: the Zoomed View of A Small Region Covering the Locations Visited by A Target User in Two Days.

The Houston area has an extensive air monitoring network including 47 monitors measuring ozone. The O3 sensing data are not released in real time, but on an hourly basis. For our case study, hourly readings of ozone concentration levels (O3 ug/m^3 hourly) across 39 monitors in the Houston metropolitan area are used and served as "ground truth" against which a node's O3 sensing report is checked for service quality and the service experience is reported to the cloud server for TaaS service management.

A node (node i) in the O3COI group can query the ozone level in a particular location and at a particular time via a mobile IoT cloud application [5][99] installed in its smartphone. The mobile application would send the query to all O3COI members that are in this particular location via the mobile cloud application. Upon receiving O3

sensing reports from other members, node i sends queries via IoT-TaaS to get the trustworthiness scores of these IoT devices who had reported sensing reports. To filter out untrustworthy O3 sensing reports, node i first accepts a sensing report (S_j) from j only if j is deemed trustworthy for O3 sensing service (i.e., i's trust score toward j, t_{ij}, is higher than 0.5 as determined by IoT-TaaS by Equation (5.5). Then it computes a trust-weighted O3 level average as follows:

$$S = \sum_{j=1}^{N} (t_{ij} / \sum_{j=1}^{N} t_{ij}) \times S_j. \tag{5.12}$$

where N is the number of trustworthy members providing O3 sensing reports in the particular location. If the average O3 level exceeds a maximum threshold defined by i's owner, node i will decide not to visit the location because the ozone level will cause harm to its owner's health. Figure 5.11 shows a scenario in which the target node (node i) before moving to a new location asks for the ozone level through trust-based participatory sensing. In this scenario, j and k provide O3 levels 180 and 40, respectively. However since j's trust score is only 0.2 (supplied by IoT-TaaS), the O3 level reported by j (180) is filtered out. Node k's trust score is 0.9 (supplied by IoT-TaaS), so the O3 level (40) reported by k is accepted. Node i then applies Equation (5.12) to compute the average O3 level and since it is below the maximum threshold, node i decides to step into the location. Node i later checks the ground truth O3 level against sensing readings reported by j and k. As a result, node i reports a positive service rating for k because k provided a satisfactory O3 level, but a negative service rating against j because of the large discrepancy between the ground truth O3 level and the high O3 level reported by j.

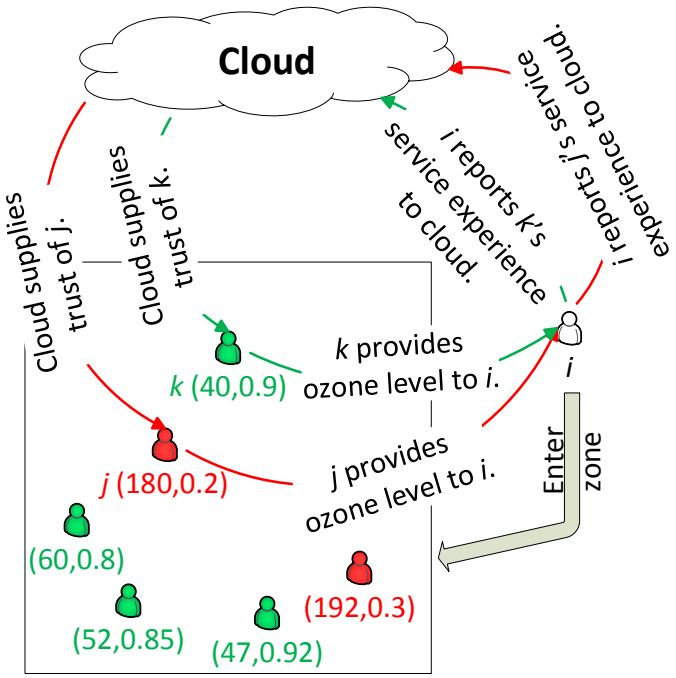


Figure 5.11: A Scenario Illustrating How A Target Node Invokes Trust-based Participatory Sensing Application Before Moving to A New Location.

Using the ns3 simulator, we simulate the participatory sensing system. We use real traces of O3 levels and mobility traces of users in the O3COI group in the city of Houston [88]. The O3 level can be classified as good condition [0, 50] ug/m^3, medium condition [51, 168] ug/m^3 (unhealthy for sensitive groups), poor condition [169, 208] ug/m^3, and severe condition [209+] ug/m^3. The percentage of bad nodes is set at P_M in the range of [0, 30%]. A malicious node always reports O3 readings in poor condition range [169, 208] ug/m^3 regardless of location with the intention to break the system. Also a malicious node will perform bad-mouthing attacks (saying a good node's sensing result is not trustworthy in the user satisfaction report) and ballot-stuffing attacks (saying a bad node's sensing result is trustworthy) when it submits a service rating report recording its satisfaction experience s_i toward device d_i.

We compare IoT-TaaS with ObjectiveTrust [86] and Adaptive IoT Trust [27]. We measure two performance metrics for performance evaluation:

- the trust-weighted average O3 reading vs. ground truth (i.e., the actual O3 level at a specific location and a particular time);
- the accuracy of selecting trustworthy participants.

The goal is to prove that IoT-TaaS provides O3 readings closer to ground truth than these two very recent, contemporary IoT trust protocols.

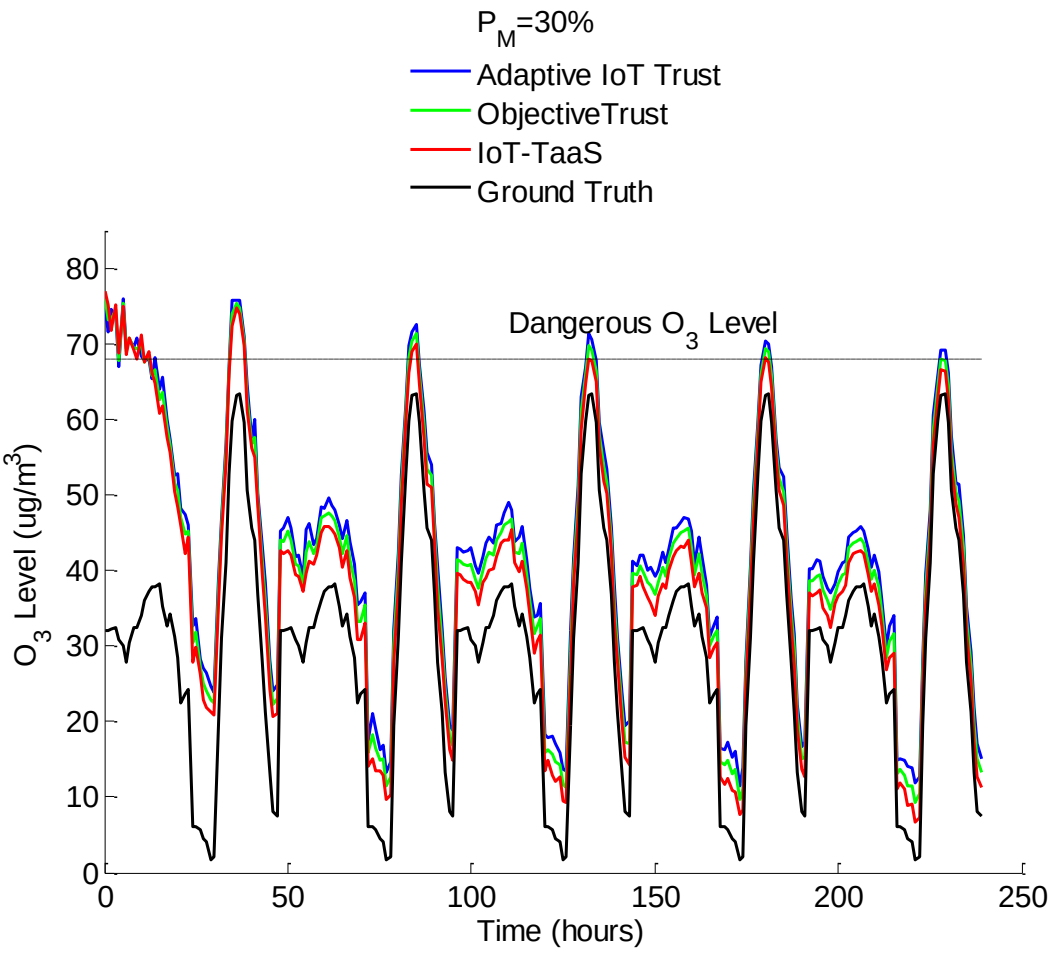


Figure 5.12: Performance Comparison of Trust-Weighted Average O3 Readings.

Figure 5.12 shows the trust-weighted average O3 readings vs. time of a selected target node (each time point is an O3 detection service request) with the percentage of bad nodes P_M set at 30%. In the experiment, the target node repeatedly queries the ozone level in the location that he will visit next over a 250 hour span. Each data point under a particular trust protocol is the average O3 level obtained from Equation (5.12). For example, at time t = 10 hours, the target node sends queries via IoT-TaaS to get the trustworthiness scores of those IoT devices that have supplied O3 readings in the particular location. The target node accepts results (S_j) from 557 trustworthy IoT devices (for which the trust score is higher than 0.5) for the O3 sensing service out of all 764

members in that particular location at that particular time and it then computes the average O3 level based on Equation (5.12).

The results indicate that IoT-TaaS (red line) can provide O3 readings very close to ground truth (black line) as time progresses. Further, IoT-TaaS outperforms Adaptive IoT Trust (orange line) and ObjectiveTrust (green line) in terms of accuracy (i.e., the difference between ground truth and the average O3 levels) and resiliency (against malicious attacks of 30% bad nodes). We attribute this to its ability to effectively and adaptively aggregate trust evidence from all nodes in the system through our effective and efficient localized report-and-query paradigm. We draw a line "Dangerous O3 Level" for a user whose "dangerous O3 level" is 68 as diagnosed by his/her doctor as vulnerable to O3 exposure for long hours. We see that at time t=130, 180, or 235 (the last three peaks in the figure) only IoT-TaaS will correctly identify the fact that O3 level is below the dangerous level, while either Adaptive IoT Trust or ObjectiveTrust will raise a false alarm that the dangerous O3 level for this user is already reached.

Figure 5.13 shows the percentage of bad nodes selected to provide sensing results to a selected target node. IoT-TaaS outperforms Adaptive IoT Trust and ObjectiveTrust as time progresses. The results can be explained as follows: Compared with Adaptive IoT Trust, IoT-TaaS is not being limited by encountering experiences and can leverage cloud service to aggregate broad evidence from all nodes who have had sensing service experiences with a target IoT device. Compared with ObjectiveTrust which is based on "objective trust" (i.e., common belief), IoT-TaaS is based on "subjective trust" (one-to-one trust evaluation) and can adaptively put a higher weight on a participant if it has had good O3 sensing experiences with the particular participant. This allows IoT-TaaS to more effectively select trustworthy participants among all participants that had submitted O3 sensing reports.

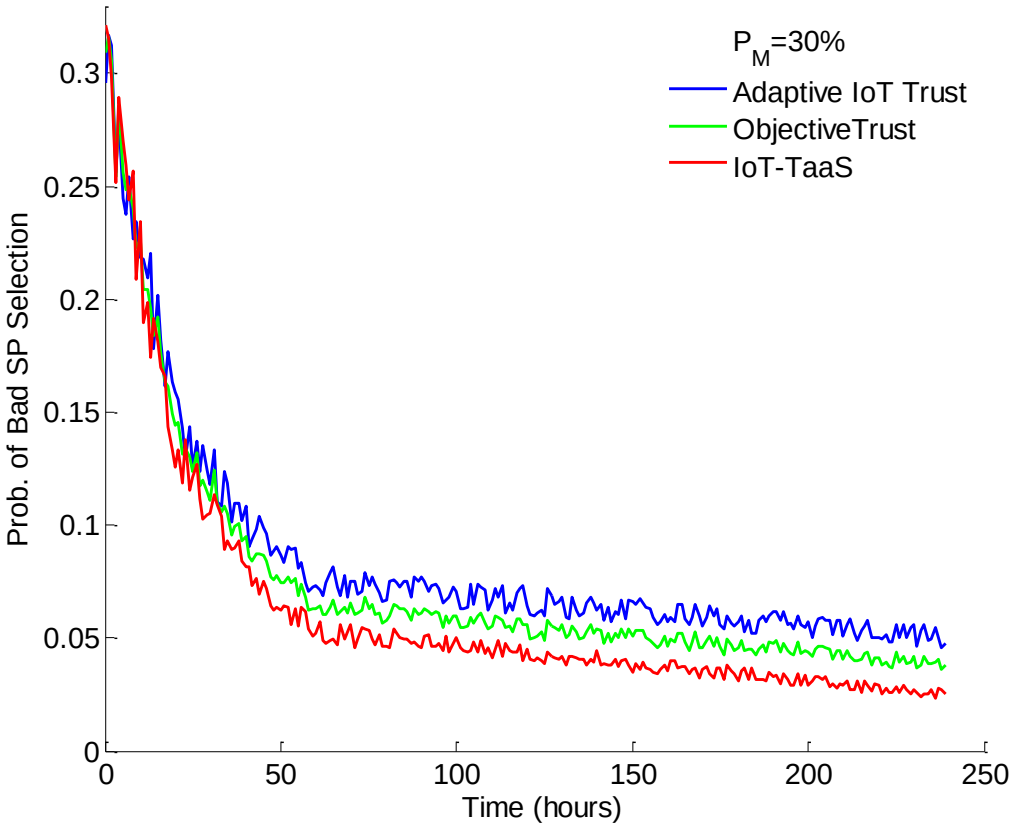


Figure 5.13: Percentage of Bad IoT Devices Selected to Provide O3 Sensing Service.

5.4.2 Case Study 2: Travel Planning

Ed has never visited New York City. He wants to plan his travel early on from Seattle, including airline reservation from Seattle to New York, ground transportation after reaching New York, hotel reservation at New York, entertainments and attractions while in New York, hotel shuttle to the airport, etc. Ed instructs his smart phone to first construct a workflow structure for the travel and then select from a myriad of IoT SPs to populate the workflow structure. Figure 5.14 shows the service flow structure constructed by his smartphone. There are 9 atomic services connected by three types of workflow structures, namely, *sequential*, *parallel* (AND), and *selection* (OR). Each service would have multiple SP candidates. The trustworthiness score of a candidate service composition based on the service flow structure in Figure 5.14 is calculated recursively as explained earlier. In this application we consider service constraints such as a budget limit. Simply selecting the most trustworthy SPs may lead to infeasible solutions.

Suppose that each SP announces its price when publishing the service and the SR has a budget limit for service composition. In *trust-based service composition with constraints*, the SR calculates the overall utility score and the overall price for each candidate configuration, using the trust value it has toward a SP to predict the utility score for that SP, and selects the configuration with the highest utility score among those with the overall price below the budget limit.

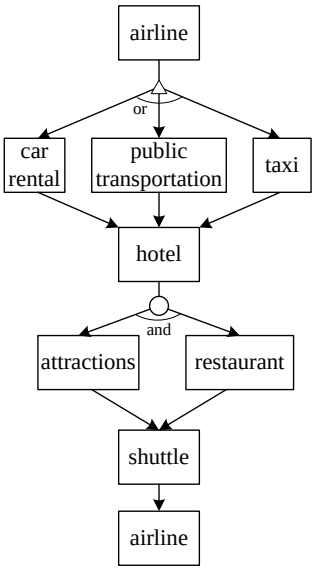


Figure 5.14: A Service Flow Structure for the Travel Planning Application.

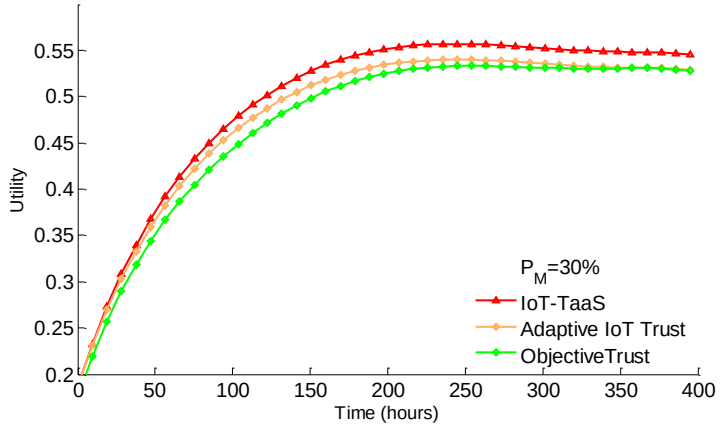


Figure 5.15: Utility Score of the Travel Planning Application.

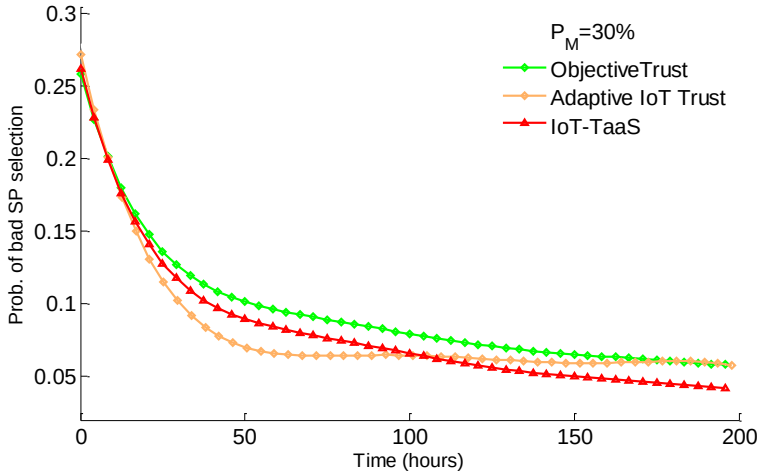


Figure 5.16: Probability of a Bad SP Being Selected for the Travel Planning Application.

Figure 5.15 shows the ns-3 simulation results with P_M=30%. We again observe that the trend is similar in terms of performance ranking, with *trust-based service composition* with IoT-TaaS outperforming Adaptive IoT Trust [27] and ObjectiveTrust [86] (in this order).

Figure 5.16 shows the percentage of bad nodes selected for service composition with budget limit constraints. IoT-TaaS again outperforms Adaptive IoT Trust [27] and ObjectiveTrust [86] by a significant margin. We again attribute the superiority of IoT-TaaS over Adaptive IoT Trust [27] and ObjectiveTrust [86] to its adaptability in response to a high percentage of nodes performing malicious attacks.

5.5 Summary

In this chapter, we designed and analyzed a scalable trust management protocol called IoT-TaaS for supporting a Trust-as-a-Service (TaaS) cloud utility for large IoT systems. We demonstrated the superiority of our IoT-TaaS protocol over Adaptive IoT Trust [27] and ObjectiveTrust [86] in trust convergence, accuracy and resiliency against malicious nodes performing self-promoting, bad-mouthing, ballot-stuffing, and opportunistic service attacks. We also demonstrated its applicability by applying IoT-TaaS to two real-world IoT applications. Our results support its superiority over Adaptive IoT Trust [27] and ObjectiveTrust [86] in selecting trustworthy nodes and maximizing application performance.

Chapter 6 Hierarchical Trust Management for Hybrid IoT Systems

In this chapter, we develop trust management protocols for hybrid IoT systems. We first propose a hierarchical cloud architecture with moderately powerful base stations and routers (owned by mobile network operators) in the lower layers and more powerful mini and big data centers (owned by cloud service providers) at the higher layers for supporting integrated mobility, service, and trust management of service-oriented IoT systems. Then we propose and analyze a 3-tier cloud-cloudlet-device hierarchical trust-based service management protocol called IoT-HiTrust for large-scale mobile cloud IoT systems. Our mobile cloud hierarchical service management protocol allows an IoT customer to report its service experiences and query its subjective service trust score toward an IoT service provider following a simple localized report-and-query paradigm. We verify IoT-HiTrust's convergence, accuracy, and resiliency properties against self-promotion, bad-mouthing, ballot-stuffing, discriminatory, and opportunistic attacks despite intermittent network disconnection to the cloud. We test the feasibility by applying IoT-HiTrust to two case studies: a smart city travel service composition and binding application, and an air pollution detection and response application. The results demonstrate that IoT-HiTrust outperforms contemporary distributed and centralized trust-based IoT service management protocols in selecting trustworthy nodes to maximize application performance, while achieving scalability.

6.1 A Hierarchical Cloud Architecture for Integrated Mobility, Service and Trust Management

Figure 6.1 illustrates our proposed cloud hierarchy. We label the clouds from top to bottom as mega, macro, micro, and nano clouds. The nano and micro clouds can be base stations and routers owned by network operators, while macro and mega clouds can be mini and big data centers owned by cloud service providers. Each nano cloud at the bottom layer can be just a base station covering a geographical region, providing a communication path for IoT devices (e.g., sensors, smart phones, vehicles) in a region to interact with the cloud via wireless communication.

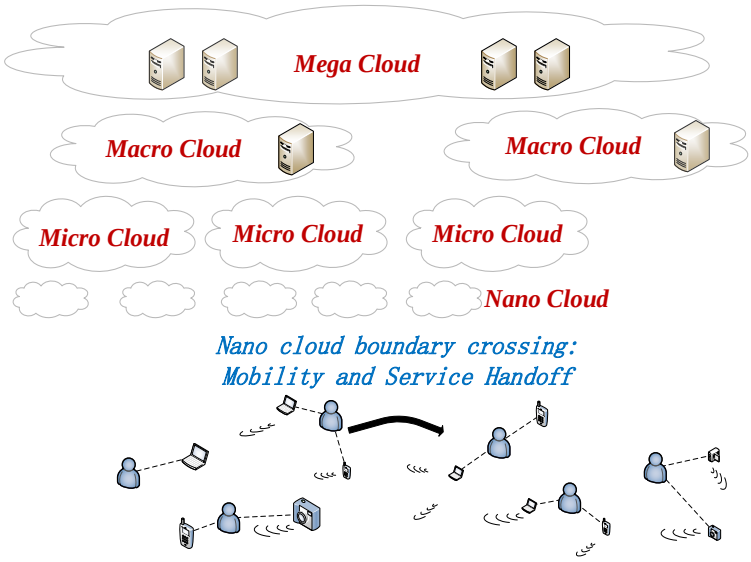


Figure 6.1: Hierarchical Cloud Architecture.

6.1.1 Mobility Management

Mobility management is the basic functionality of a mobile network. Since the bottom levels are base stations and routers in a mobile network, the location information (or the region a user is in) is maintained automatically as an added benefit. Specifically, following the concept of using a home location server for tracking the location of a mobile node in a mobile network [35][37][55][34], we track the location of an IoT device using a family of "home" clouds, starting from the home nano cloud (base station) at the bottom layer, home micro cloud (router) at the second bottom layer, home macro cloud at the second top layer, and home mega cloud at the top layer. These home clouds are assigned based on the "home" geographical location of an IoT device similar to the home location register (HLR) in mobile networks [35][37][55][34]. When an IoT device moves from one region to another region (if the IoT device is mobile), a "mobility handoff" ensues by which the nano cloud which the IoT device just roams into will inform all home clouds of the IoT device of the new location. When a caller IoT device wants to locate a callee IoT, it simply sends a query to the least common "home" cloud of the caller IoT device and the callee IoT device and this least common "home" cloud will return the callee IoT's current location that was recorded in its database. Essentially, the location information is known to the system all the time, thus facilitating location-based service to be invoked by the cloud as IoT devices move from one region to another. One cloud application that can benefit from this capability is participatory sensing [70] allowing a huge amount of location-based sensing information to be

collected by IoT devices (e.g., smart phones, public transportation vehicles, city utility poles, etc.) and analyzed in the cloud for situation awareness and decision making.

6.1.2 Service Management

Service management is the basic functionality of cloud computing service. Nano and micro clouds in the bottom layers of the cloud hierarchy augment macro and mega clouds in the top layers to collectively provide cloud service to IoT devices. An IoT device will only interact with its current nano cloud for service invocation to minimize energy consumption and service latency. The current nano cloud will examine the service request to decide if it can process locally. If yes, it will complete the request-response cycle locally without disturbing the upper layer clouds. This is true if the request is just to do computation offloading, or if the service request only requires local service data to answer the service request. If the service request is to report new service data such as a feedback or a sensing outcome, the current nano cloud will follow the store-process-forward procedure. i.e., it will store a replicated copy [38][42] of the service data, process it locally as needed, and pass the new service data to the home clouds of the IoT device. If the service request involves several IoT devices some of which are not under the current nano cloud, then the nano cloud will pass the request to the least common "home" cloud of these IoT devices for processing because the least common "home" cloud will store location and service data of these IoT devices. If the service request is a query regarding a target IoT device, then the current nano cloud will follow the forward-wait-reply procedure. That is, the query will be forwarded to the least common "home" cloud of the requesting IoT device and the target IoT device. Then it will wait for a reply to return to it after which it will forward the reply to the requesting IoT device. Last, when an IoT device moves from one region to another region, a "service handoff" ensues. Among other things, the IoT device's Virtual Machine (VM) [99][92][69] is migrated from the old nano cloud to the new nano cloud so as to maintain service continuity. A forwarding link is connected between the old nano cloud and the new nano cloud, so if the old nano cloud receives a reply, it will forward to the new nano cloud.

6.1.3 Trust Management as a TaaS Cloud Utility

Trust management of IoT devices is of fundamental importance for any service-oriented IoT application that must incorporate feedbacks or recommendations for decision making because one must identify trustworthy raters or recommenders and apply filtering mechanisms to filter out untrustworthy feedbacks before data analysis and decision making. With the cloud architecture, trust management can be realized as

a TaaS cloud utility using standard store-process-forward and forward-wait-reply procedures described earlier.

An IoT device can report user satisfaction in the range of [0, 1] toward another IoT device who just completed a service of a specific service type (e.g., sensing noise or sensing air pollution) to its current nano cloud. The nano cloud upon receiving a user satisfaction report would follow the standard store-process-forward procedure described earlier.

An IoT device (on behalf of its owner) can simply query its current nano cloud about the trustworthiness of a target IoT device for providing service of a specific service type. The current nano cloud would follow the standard forward-wait-reply procedure described earlier. The least common home cloud of the requesting IoT device and the target IoT upon receiving the query will simply use reports in its local store and apply a trust protocol to assess the trustworthiness of the target node. When the trust assessment is completed, the home cloud will return the response (i.e., the trustworthiness of the target IoT device for providing service) to the nano cloud forwarding the query who in turn will forward the response to the requesting IoT device for decision making.

6.1.4 Reconfigurability, Fault Tolerance, and Resiliency

Failure recovery is critical for service-oriented IoT applications [15][20][21][53][33]. The proposed hierarchical cloud architecture is highly reconfigurable and fault tolerant to failures. When a nano cloud fails, one of its neighbor nano clouds can take over its duty to continue with the ongoing services. This may involve restarting the VM process originally running on the failed nano cloud. When a micro cloud fails, its function can be covered by all nano clouds under it, since the service data stored in the failed micro cloud can be recovered from all service data stored in all nano clouds under it based on the store-process-forward procedure. The same fault tolerance process is applied hierarchically up.

The proposed hierarchical cloud architecture is also highly resilient to network disconnection. When answering a query involving a requesting IoT device and a target IoT device, the nano cloud needs to find the least common home cloud by following the standard forward-wait-reply procedure. If the last common home cloud cannot be found because of disconnection, then it can always go to the mega cloud which is the common home cloud of all IoT devices, at the expense of energy consumption and service latency. If the nano cloud cannot find any home cloud because of a total disconnection, then it can attempt to answer the query using the service data stored locally. The accuracy of the reply largely depends on the amount of service data for the

target IoT device stored locally in the nano cloud. If the target IoT device does not move often and stay in the nano cloud over the recent past, then the nano cloud would store most of the service data of the target IoT device (because nearly all reports would come to this nano cloud) which would be sufficient for accurately answering the query. This robust response strategy improves resiliency against disconnection to maintain service continuity.

6.2 Mobile Cloud Hierarchical IoT Trust Management

6.2.1 3-Tier Cloud-Cloudlet-Device Architecture

Leveraging the cloud hierarchy discussed in Section 6.1, we develop a trust protocol called IoT-HiTrust for managing IoT devices in mobile cloud environments.

As illustrated in Figure 6.2, we leverage a 3-tier mobile cloud hierarchy [92] for hierarchical IoT trust management. IoT devices are sitting at the bottom tier of the hierarchy. Cloudlets are at the middle tier. Cloud servers are at the top tier. We use the term "*node*" to loosely refer to an IoT device.

The cloud at the top tier is a logical entity consisting of many cloud servers each serving a number of users (carrying IoT devices) and is assumed infallible with tight security and reliability protection. The number of cloud servers is changed dynamically for scalability and load balance reasons. Each user with its unique id is internally assigned to a "home" cloud server based on distributed hash table (DHT) techniques for load balancing. The home cloud server of a user will manage this particular user's data internally. While a user's VM may migrate from one place to another, the user's home cloud server remains the same. The mapping of a user to its home cloud server is not known to the user and can be migrated dynamically for scalability and load balance reasons. A user's queries and reports are simply sent to the cloud through the local cloudlet with the system routing them to the user's home cloud server.

At the bottom tier sits "lightweight" IoT devices owned by users (e.g., smart phones, sensors, PDAs, etc.). Lightweight IoT devices typically are intermittently connected to the Internet. They are typically carried by their owners and can move from one cloudlet to another due to mobility. When a lightweight IoT device is disconnected from the Internet, it can simply connect to a regional cloudlet for service continuity. To save energy and bandwidth, an IoT device always communicates with the cloud through its regional cloudlet.

Cloudlets (each occupying a physical region) are sitting in the middle layer, bringing users closer to the cloud. A cloudlet is formed by a set of "heavyweight" IoT devices

(e.g., PCs, servers, notebooks, etc.) with moderate communication, computational and storage capability to offload lightweight IoT devices sitting at the bottom layer. Heavyweight IoT devices typically are well connected to the Internet. Using IoT-HiTrust, the cloud periodically evaluates trustworthiness of all IoT devices in a cloudlet region and selects a group of IoT devices to form the region's cloudlet. In return for the surrogate services provided by these IoT devices, users owning these IoT devices are granted access privileges to cloud resources. We will call these IoT devices selected to govern a region's cloudlet as "cloudlet devices," with the understanding that cloudlet devices are just heavyweight IoT devices.

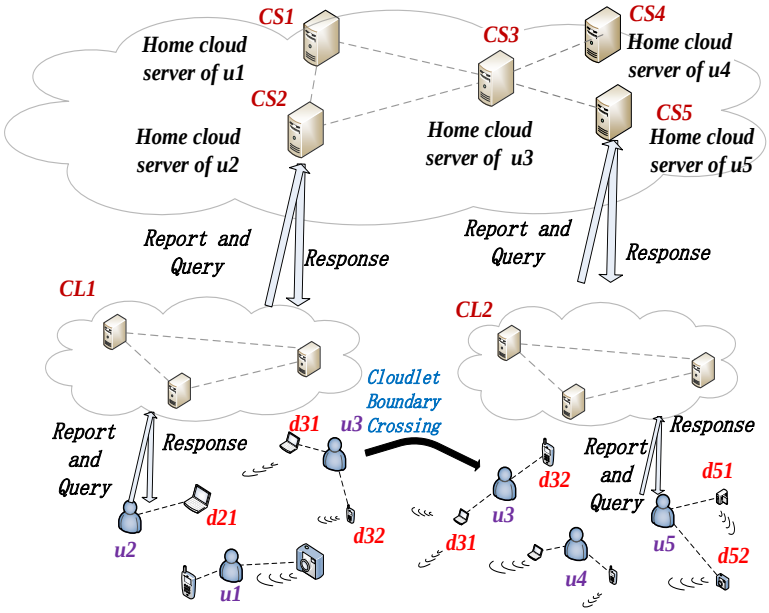


Figure 6.2: Cloud-Cloudlet-Device Architecture.

Figure 6.2 shows two cloudlets, CL_1 and CL_2, each with three heavyweight IoT devices serving as cloudlet devices. The "logical" cloud has 5 cloud servers, CS_1, CS_2, CS_3, CS_4 and CS_5. In Figure 6.2, CS_2 is the home cloud server of user u_2 and CS_3 is the home cloud server of user u_3. Since user u_3 owns two IoT devices, d_{31} and d_{32}, these two IoT devices' home cloud sever is also CS_3.Each cloudlet only relays requests/responses from/to IoT devices under its region. In addition, each cloudlet caches trust information. In case of Internet disconnection, a cloudlet can operate in disconnection mode [92] to answer user queries issued from IoT devices in its region. The regional size covers the radio range to ensure that mobility and instability of radio environments will not be a major factor to prevent nodes under a cloudlet region from communicating directly with the cloudlet. When an IoT device in one cloudlet region moves to another cloudlet region, it performs a registration/deregistration action to the two in-

volving cloudlets. As illustrated in Figure 6.2, user u_3 moves across the cloudlet boundary, causing deregistration with the old cloudlet CL_1 and registration with the new cloudlet CL_2. A pointer is recorded by CL_1 so that a message for u_3 can be redirected to CL_2.

6.2.2 Collaborative Filtering based on Social Similarity

Our trust model is based on social relationships among humans who are owners of IoT devices. Hence the trustor is a user and the trustee is a device (owned by another user). The trust relationship is not between a user trustor and a user trustee because a user may own several IoT devices with vastly heterogeneous capabilities.

We use the social relationships between a trustor and a recommender (or rater) for the trustor to weigh the service rating provided by a rater toward a trustee. We consider three core social metrics for measuring social relationships which are multifaceted: friendship (representing intimacy), social contact (representing closeness), and community of interest (representing knowledge and standard on the subject matter). The idea is that two users sharing similar social relationships are likely to have similar subjective views towards services provided by a trustee IoT device. Social relationships between owners are translated into social relationships between IoT devices as follows:

1. *Friendship*: Each owner has a list of friends (i.e., other owners), representing its social relationships. This friendship list varies dynamically as an owner makes or denies other owners as friends. If the owners of two IoT devices are friends, then it is likely they will be cooperative with each other. For ease of discussion, we do not differentiate friends from acquaintances. We can easily extend IoT-HiTrust to take this finer granularity into consideration.

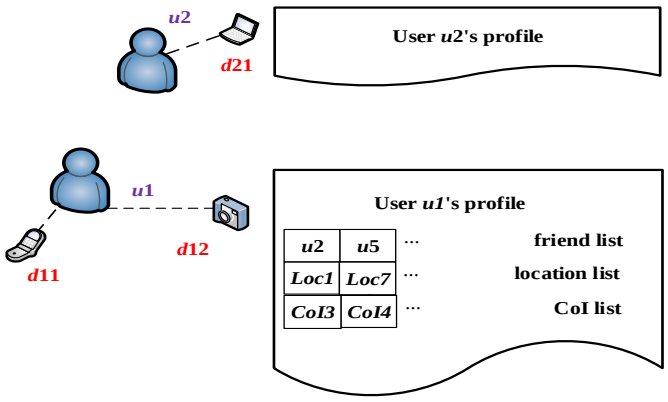


Figure 6.3: Each User Stores Its Friend, Location, and CoI Lists for Detecting Social Similarity with other Users.

2. *Social Contact*: A device may be carried or operated by its owner in certain environments (e.g., work vs. home or a social club). Two devices have high social contact opportunities when their owners have similar mobility patterns.
3. *Community of Interest* (CoI): Each owner has a list of communities of interest such as health, sport, travel, etc. Nodes belonging to a similar set of communities likely share similar interests or capabilities [67].

To facility measuring social similarity with other owners, an IoT device belonging to user u_x maintains three lists in its profile (as illustrated in Figure 6.3):

1. Friends of u_x, denoted by a set $F_x = \{u_1, u_2, \dots\}$;
2. Locations that u_x frequently visited for social contact, denoted by a set $S_x = \{Loc_1, Loc_2, \dots\}$;
3. Communities of interest that u_x is a member of, denoted by a set $C_x = \{CoI_1, CoI_2, \dots\}$.

A user may designate one of its IoT devices to update such information and share the information with other IoT devices it owns. Although social similarity is between two users, it is propagated to IoT devices owned by the two users, so social similarity is also between two devices.

6.3 IoT-HiTrust Protocol Design

The IoT-HiTrust protocol design is similar to IoT-TaaS protocol design discussed in Chapter 5, except that the former is for a centralized cloud architecture, whereas the latter is for a hierarchical cloud architecture. To be self-contained, the IoT-HiTrust protocol design is described in this section in full detail at the expense of somewhat repetition.

6.3.1.1 Report-and-Query Design for Scalability

Whenever a service is received, a user (using its primary IoT device) reports whether it is satisfied with the service provided by an IoT device to the user's home cloud server via a service rating report. Let the current user satisfaction experience of user u_x toward device d_i be represented by a value, $f_{x,i}$ which can be a real number in the range of 0 to 1 indicating the user satisfaction level, or simply a binary value, with 1 indicating satisfied and 0 not satisfied. Here $f_{x,i}$ is the first piece of information sent from u_x to its home cloud server. A timestamp is also sent in the report to indicate the time at which this service rating happens. This allows cloud servers to know the event occurrence times of reports so it can decay the credibility of the report over time.

Each user maintains its (F, S, C) profile separately. When user u_x encounters user u_y, they exchange their (F_x, S_x, C_x) and (F_y, S_y, C_y) profiles so as to measure their mutual social similarity. To preserve energy, they can exchange the profile information the very first time they encounter or periodically. This is especially so if user profile information does not change much over time. To preserve privacy, they only want to reveal common elements in the *F, S,* and *C* lists (If any) but do not want to let the other party know their entire (F, S, C). To achieve this users u_x and $u_y$can first authenticate each other using standard PKI. User u_x can then use a cryptographic hash function in combination with a secret session key *K* (established via PKI during user authentication) to generate a hash-based message authentication code HMAC(*K, p*) for $p \in (F_x, S_x, C_x)$ and then transmit HMAC(*K, p*) along with HMAC(*K,* HMAC(*K, p*)) to u_y. When u_y receives the message, it can unilaterally generate HMAC(*K,* HMAC(*K, p*)) using HMAC(*K, p*) sent by u_x. If this matches with HMAC(*K,* HMAC(*K, p*)) sent by u_x, then u_y verifies the message received is indeed sent by u_x. Then u_y can compare HMAC(*K, p*) with HMAC(*K, q*) for $q \in (F_y, S_y, C_y)$. If HMAC(*K, p*)=HMAC(*K, q*) then *p*=*q* and a common friend, location, or CoI (corresponding to *F, S,* or *C*) is identified. If HMAC(*K, p*)≠HMAC(*K, q*), it prevents the identities of uncommon friends/locations/CoIs from being revealed.

We adopt cosine similarity to measure the distance of two social relationship lists (see Figure 6.3), with 1 representing complete similarity and 0 representing no similarity. The physical meaning of cosine similarity is the cosine of the angle between the two vectors deriving from the two lists, with the cosine value of 1 meaning totally identical lists and the cosine value of 0 meaning totally non-overlapping lists. Computational efficiency is the main reason why we choose cosine similarity to measure the similarity of two lists in high-dimensional positive spaces because of limited computational capacity of IoT devices. Specifically, the following three similarity metrics are measured as follows:

- **Friendship Similarity** (sim_f): The friendship similarity is a powerful social relationship (intimacy) for screening recommendations. After two users u_x and u_y exchange their friend lists, F_x and F_y, they compute the "cosine similarity" sim_f as follows:

$$sim_f(u_x, u_y) = \frac{|F_x \cap F_y|}{\sqrt{|F_x| \cdot |F_y|}} \tag{6.1}$$

  Where the notation $|A|$ represents the cardinality of set *A*.

- **Social Contact Similarity** (sim_s): The social contact similarity represents closeness and is an indication if two nodes share the same location-based social contacts (e.g.,

co-workers at work, classmates at school, and co-residents at home) and thus share the same sentiment towards devices which provide the same service. The operational area could be partitioned into sub-grids. User u_x records the IDs of sub-grids it has visited in its *location list* S_x for social contact. After two users u_x and u_y exchange their location lists, S_x and S_y, they could compute sim_l in the same way of computing sim_f as follows:

$$sim_s(u_x, u_y) = \frac{|S_x \cap S_y|}{\sqrt{|S_x| \cdot |S_y|}} \tag{6.2}$$

- **Community of Interest Similarity** (sim_c): Two users in the same CoI share similar social interests and most likely have common knowledge and standard toward a service provided by the same device. Also very likely two users who have used services provided by the same IoT device can form a CoI (or are in the same CoI). After two users u_x and u_y exchange their device lists, C_x and C_y, they could compute sim_c in the same way of computing sim_f as follows:

$$sim_c(u_x, u_y) = \frac{|C_x \cap C_y|}{\sqrt{|C_x| \cdot |C_y|}} \tag{6.3}$$

The above three social similarity measures (sim_f, sim_s, sim_c) are computed upon the encountering event of user u_x with user $u_{y,}$ and are reported by user u_x and user u_y through their local cloudlets to the home cloud servers of user u_x and user u_y, respectively.

When the home cloud server of u_x receives $sim_i(u_x, u_y)$, $i \in \{f, s, c\}$, from user u_x which just encounters user u_y, it computes the social similarity between users u_x and u_y (who now serves as a rater or recommender) as a weighted combination of all social similarity metrics as follows:

$$sim(u_x, u_y) = \sum_{i \in \{f,s,c\}} w_i \cdot sim_i(u_x, u_y) \tag{6.4}$$

where $0 \leq w_f, w_s, w_c \leq 1$, with $w_f + w_s + w_c = 1$, are social similarity weight parameters to be dynamically adjusted by IoT-HiTrust to maximize trust protocol performance.

Whenever a user wants to know the trust value of an IoT device, it simply sends a query to its home cloud server. For example, in Figure 6.2, u_2 will send a query to its home cloud server CS_2 to know its "subjective" trust toward d_{31} which belongs to u_3.

Let the "subjective" trust value of user u_x toward d_i be denoted by $t_{x,i}$. The home cloud server of u_x computes $t_{x,i}$ by combining $u_x's$ direct trust toward d_i ($t_{x,i}^d$) based on self-observation reports, and $u_x's$ indirect trust toward d_i ($t_{x,i}^r$) based on other users' ratings, as follows:

$$t_{x,i} = \mu_{x,i} \cdot t_{x,i}^d + (1 - \mu_{x,i}) \cdot t_{x,i}^r \tag{6.5}$$

Here, $\mu_{x,i}$ is a weight parameter ($0 \leq \mu \leq 1$) to weigh the importance of direct trust relative to indirect trust. The selection of $\mu_{x,i}$ is critical to trust evaluation. We apply adaptive filtering developed in [27] to adjust $\mu_{x,i}$ dynamically to effectively cope with malicious attacks and to improve trust accuracy.

The direct trust $t_{x,i}^d$ in Equation (6.5) is computed by Beta Reputation [65] under which the trust value is modeled as a random variable in the range of [0, 1] following the Beta (α, β) distribution. The numbers of positive and negative experiences are modeled as binomial random variables. Since the beta-binomial is a conjugate pair, this leads to a posterior beta distribution with updated parameters. Specifically, we can calculate direct trust of u_x toward device d_i, $t_{x,i}^d$, as follows:

$$t_{x,i}^d = \alpha/(\alpha + \beta) \tag{6.6}$$

where α is the amount of positive service experience with time decay and β is the amount of negative service experience with time decay, calculated as $\alpha = \sum_{all} f_{x,i}(t)\, e^{-\lambda_d(t_{now}-t)}$ and $\beta = \sum_{all}(1 - f_{x,i}(t))\, e^{-\lambda_d(t_{now}-t)}$ where $f_{x,i}(t)$ is a service rating received from user u_x at time t about d_i's service quality, t_{now} is the current time, and λ_d is the decay parameter to discount old service experiences. Here $f_{x,i}$ contributes to positive service experience and $1 - f_{x,i}$ contributes to negative service experience. If $t = t_{now}$ then the service rating has the highest credibility of 1; otherwise, the credibility of the service rating decays over time exponentially. The summation is over all service ratings received from user u_x, including old and new service ratings maintained in user $u_x's$ cloud server.

To compute indirect trust $t_{x,i}^r$, the home cloud server of u_x first locates social similarity records $sim(u_x, u_y)'s$ in its local storage. The home cloud server of u_x then selects top-R raters from R users with the highest similarity scores with u_x and calculates the indirect trust ($t_{x,i}^r$) towards device d_i as follows:

$$t_{x,i}^r = \sum_{u_y \in U} \frac{sim(u_x, u_y)}{\sum_{u_z \in U} sim(u_x, u_z)} \cdot t_{y,i}^d \tag{6.7}$$

The indirect trust model as indicated in Equation (6.7) is essentially a weighted sum of service ratings reported by other IoT devices (acting as recommenders) with a higher weight giving to a recommender with a higher social similarity. It is based on a widely accepted concept that mobile IoT systems can be characterized as a hybrid of P2P and social networks and IoT trust management must take into account social relationships among device owners in order to maximize protocol performance [27][86]. Here, U is a set of up to R raters whose $sim(u_x, u_y)$ scores are the highest, $u_y \in U$ is a rater selected, and $t_{y,i}^d$ is the service rating provided by u_y toward device d_i. We note that $t_{y,i}^d$ is stored in the home cloud server of u_y but it is obtainable after the home cloud server of u_x communicates with the home cloud server of u_y. In Equation (6.7), the service rating provided from u_y toward d_i $(\text{i.e.}, t_{y,i}^d)$ is weighted by the ratio of the similarity score of u_x toward u_y to the sum of the similarity scores toward all raters. That is, if the similarity score of u_x toward u_y is high relative to that of u_x toward other raters, then the home cloud server of u_x will put a relatively high weight on the rating $t_{y,i}^d$ provided by u_y to compute $t_{x,i}^r$.

6.3.1.2 Protocol Execution Description

In this section, we elaborate in detail the actions taken by the entities in our 3-tier mobile cloud hierarchy (IoT devices, cloudlets, and cloud servers) while executing IoT-HiTrust. For user u_x, we use $CL_x$and CS_x to refer to its local cloudlet and home cloud server, respectively, in the 3-tier mobile cloud hierarchy. We note that while CS_x is fixed, CL_x is dynamically changed as the user roams from one region to another.

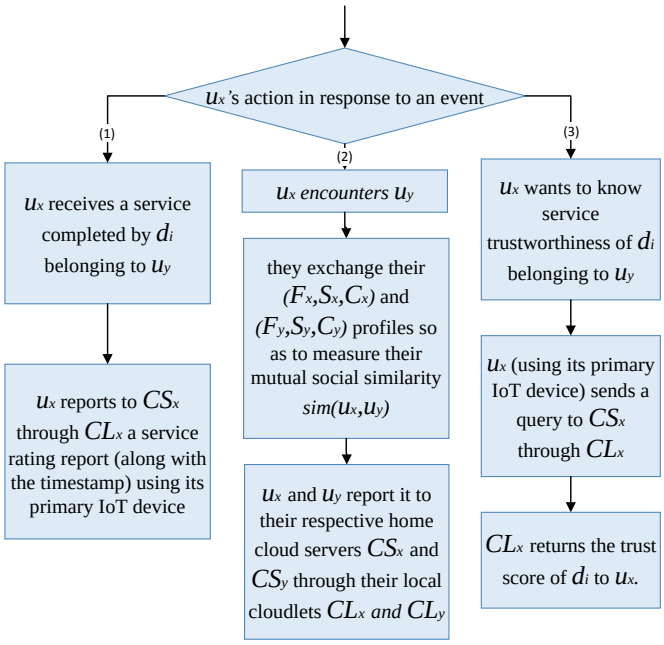


Figure 6.4: Trust Protocol Execution Action Flowchart for User u_x.

Figure 6.4 shows a flowchart of our trust management protocol execution from the perspective of user u_x. As illustrated in Figure 6.4, the actions performed by $u_x$are as follows:

(1) When u_x just receives a service completed by d_i belonging to u_y, u_x (using its primary IoT device) reports to CS_x through CL_x a service rating report ($f_{x,i}$ along with the timestamp at which service is rendered) indicating the extent to which it is satisfied with the service provided by d_i.

(2) When u_x encounters u_y, they exchange their (F_x, S_x, C_x) and (F_y, S_y, C_y) profiles so as to measure their mutual social similarity $sim(u_x, u_y)$ (Equations (6.1), (6.2), (6.3) and (6.4)). Then u_x and u_y report it to their respective home cloud servers CS_x and CS_y through their local cloudlets CL_x and CL_y, respectively. Energy constrained devices may report $sim(u_x, u_y)$ only the first time or periodically to conserve energy.

(3) When u_x wants to know service trustworthiness of d_i belonging to u_y, u_x (using its primary IoT device) sends a query to CS_x through CL_x and waits for a reply back from CL_x returning the trust score of d_i.

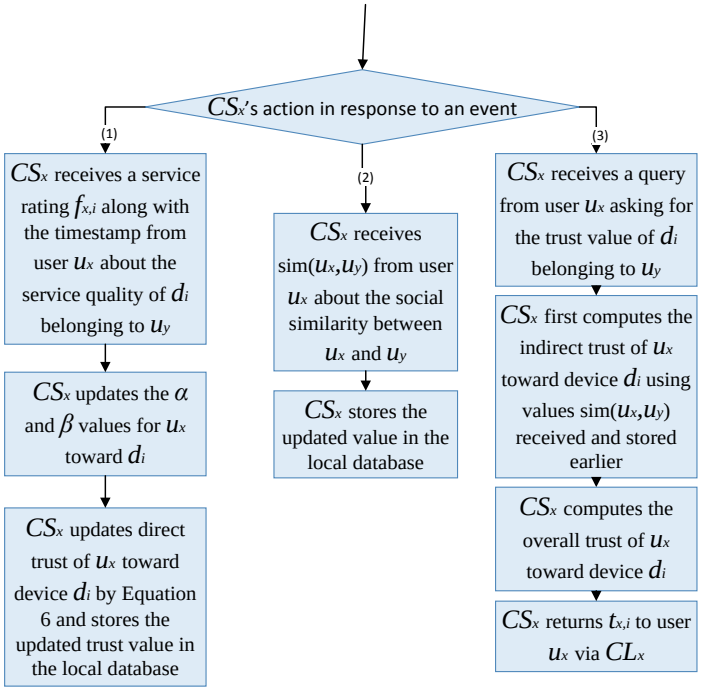


Figure 6.5: Trust Protocol Execution Action Flowchart for Cloud Server CS_x.

Figure 6.5 shows a flowchart of our trust management protocol execution from the perspective of CS_x, the home cloud server of user u_x. As illustrated in Figure 6.5, the actions performed by $CS_x$are as follows:

(1) When CS_x receives a service rating $f_{x,i}$along with the timestamp t from user u_x about the service quality of d_i belonging to u_y, CS_x updates the α and β values for u_x toward $d_i$based on $u_x's$ own old and new service ratings toward d_i with trust decay over time. Then, CS_x updates direct trust of u_x toward device d_i ($t_{x,i}^d$) by Equation (5.6) and stores the updated $t_{x,i}^d$value in the local database.

(2) When CS_x receives $sim(u_x, u_y)$from user u_x about the social similarity between u_x and u_y, CS_x stores the updated $sim(u_x, u_y)$ value in the local database.

(3) When CS_x receives a query from user u_x asking for the trust value of d_i belonging to u_y, CS_x first computes the indirect trust of u_x toward device d_i ($t_{x,i}^r$) by Equation (6.7) using $sim(u_x, u_y)$ values received earlier and stored in its local database. Then, CS_x computes the overall trust of u_x toward device d_i ($t_{x,i}$) by Equation (6.5) using $t_{x,i}^d$ and $\mu_{x,i}$ values stored in the local database. Lastly, CS_x returns $t_{x,i}$to user u_x via CL_x.

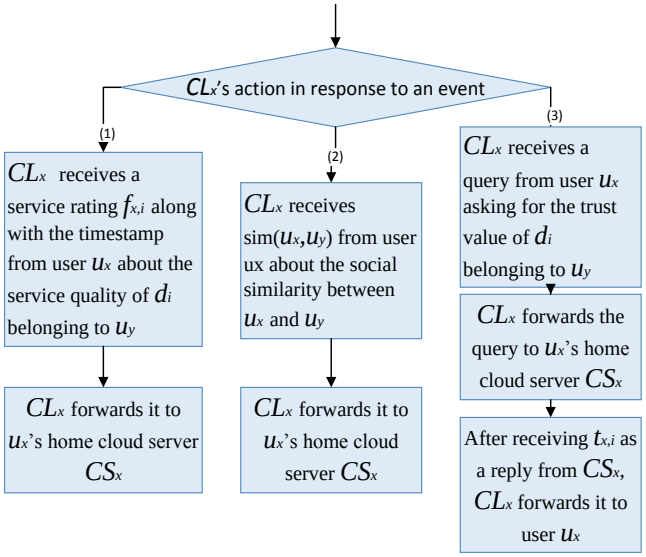


Figure 6.6: Trust Protocol Execution Action Flowchart for Cloudlet CL_x.

Figure 6.6 shows a flowchart of our trust management protocol execution from the perspective of CL_x, the local cloudlet of user u_x. As illustrated in Figure 6.6, the actions performed by $CL_x$are as follows:

(1) When CL_x receives a service rating $f_{x,i}$along with the timestamp from user u_x about the service quality of d_i belonging to u_y, CL_x forwards it to $u_x's$ home cloud server CS_x.

(2) When CL_x receives $sim(u_x, u_y)$from user u_x about the social similarity between u_x and u_y, CL_x forwards it to $u_x's$ home cloud server CS_x.

(3) When CL_x receives a query from user u_x asking for the trust value of d_i belonging to u_y, CL_x forwards the query to $u_x{}'s$ home cloud server CS_x. Then, after receiving $t_{x,i}$ as a reply from CS_x, CL_xforwards it to user u_x.

6.3.1.3 Dealing with Network Disconnection

A cloudlet may lose connectivity with the cloud if all heavyweight IoT devices selected as the cloudlet devices for the cloudlet experience network disconnection in emergency situations such as a network service disruption. A cloudlet must maintain service continuity to IoT devices during network disconnection. In case the cloudlet can still communicate with a neighbor cloudlet which still has connectivity with the cloud, then all reports, queries, and responses can route through the neighbor cloudlet. Otherwise, it will have to operate in disconnected mode [92] using cached data. For a heavy IoT device that is typically not mobile or only moves within the region, the home regional cloudlet has all the data it needs for answering a user query regarding the trustworthiness of that IoT device, since it caches all reports, responses, and social similarity reports with that IoT device pass through the local cloudlet. So all the local cloudlet has to do is to follow the computational procedure described earlier to assess the trustworthiness of that IoT device, as if the computation is performed by the cloud itself. For a lightweight IoT device, the home region cloudlet may lose some precision in estimating the trustworthiness of the IoT device because it does not have all the data needed. For example it may miss some reports of that IoT device when that IoT device moves away from its region. The trust accuracy is largely affected by the mobility of the user carrying the IoT device. In Section 6.2.3 we will assess the extent to which the trust accuracy is impacted under various mobility scenarios.

6.4 IoT-HiTrust Protocol Performance

In this section, we analyze IoT-HiTrust performance. We first perform a complexity analysis of the communication and storage cost for evaluating scalability. Then we conduct a performance analysis using ns-3 network simulator [108]. We compare IoT-HiTrust performance with two baseline trust protocols, Adaptive IoT Trust [27] and ObjectiveTrust [86]. See Chapter 2 Related Work for the detail of these two baseline trust protocols chosen for our comparative analysis.

6.4.1 Scalability Analysis

Table 6.1 summarizes the complexity analysis results. We evaluate the scalability of IoT-HiTrust against Adaptive IoT Trust [27] and ObjectiveTrust [86] based on the complexity analysis results.

Table 6.1: Complexity Analysis of IoT-HiTrust against Adaptive IoT Trust [27] and ObjectiveTrust [86].

Protocol	Communication	Storage
IoT-HiTrust	$O(\sum^{N_T} \lambda_{ij} + q_{ij} + \pi_{ij})$	$O(1)$
Adaptive IoT Trust	$O(\sum^{N_T} \pi_{ij})$	$O(N_T)$
ObjectiveTrust	$^*O\left(\sum^{N_T} \lambda_{ij} + q_{ij}\right)$	$O(1)$

Communication Cost Complexity: For IoT-HiTrust, the communication cost per IoT device (from node *i*'s perspective) is of complexity $O(\sum_{j=1}^{N_T} \lambda_{ij} + q_{ij} + \pi_{ij})$ where N_T is the number of IoT devices in the system, λ_{ij} is the service request rate of node *i* to node *j*, q_{ij} is the query rate of node *i* about node *j*'s trust status, and π_{ij} is the encountering rate of node *i* with node *j* which can be derived by analyzing the encounter or interaction pattern, e.g., a power-law distribution, as supported by the analysis of many real traces [68]. Upon completing a service from node *j*, node *i* sends its service quality assessment toward node *j* to its home cloud server. When node *i* wishes to find out if node *j* is trustworthy, it sends a query to its home cloud server. Lastly, upon encountering node *j*, node *i* exchanges its user profile with node *j's* user profile while preserving privacy to compute three social similarity measures (sim_f, sim_s, sim_c) between node *i* and node *j* according to Equations (6.1), (6.2), and (6.3), after which node *i* sends (sim_f, sim_s, sim_c) to its home cloud server for storage. The one extra message needed for sending computed social similarity values to its home cloud server does not change the complexity.

For ObjectiveTrust [86], the communication cost per IoT device (from node *i*'s perspective) is of complexity $O\left(\sum_{j=1}^{N_T} \lambda_{ij} + q_{ij}\right)$. Upon completing a service from node *j*, node *i* sends its service quality assessment toward node *j* to the cloud. When node *i* wishes to find out if node *j* is trustworthy, it sends a query to the cloud. There is no need for node *i* to exchange social relationships upon encountering each other, or report the social similarity between node *i* and node *j* to the cloud because ObjectiveTrust assumes the existence of a friendship social network graph as input for specifying social relationships (e.g., centrality and credibility) and such information is already loaded in the cloud. Due to this assumption, ObjectiveTrust's communication cost complexity is lower than that of IoT-HiTrust by $O(\sum_{j=1}^{N_T} \pi_{ij})$.

For Adaptive IoT Trust [27], the communication cost per node (from node *i*'s perspective) is $0(\sum_{j=1}^{N_T} \pi_{ij})$ because upon node *i* encountering node *j*, node *i* exchanges its user profile with node *j's* user profile for computing social similarity and also exchanges its service quality experiences with node *j's* service quality experiences toward other nodes in the system for computing trust scores toward other nodes in the system. There is no communication cost for node *i* to send its own service quality assessment results and social similarity results to the cloud because node *i* stores all service quality assessment results (including from itself and from all other nodes) as well as social similarity results in its local storage. When node *i* wishes to know node *j*'s trust status it simply looks up its trust data in the local storage. Compared with IoT-HiTrust, Adaptive IoT Trust communication cost complexity is lower by $0\left(\sum_{j=1}^{N_T} \lambda_{ij} + q_{ij}\right)$ since node *i* sends neither trust data nor trust status queries to the cloud.

Storage Cost Complexity: For IoT-HiTrust, the storage cost per node (from node *i*'s perspective) is O(1) for storing its own user profile only (i.e., friend, social contact, and community-interest lists) because all service quality experience, social similarity, and trust data are stored in the cloud. The storage cost per cloud server is $O(N_T/N_C)$ for storing service quality experiences, social similarity, and trust data of N_T/N_C devices, where N_T is the number of IoT devices and N_C is the number of cloud servers, as the load is shared by all cloud servers (through the use of a fair hash function).

For ObjectiveTrust [86], the storage cost per node is also O(1) because all data are also stored in the cloud.

For Adaptive IoT Trust [27], the storage cost per node is $O(N_T)$ because for every other IoT device, a storage space is needed for storing service quality experiences, social similarity, and trust data. Apparently Adaptive IoT Trust is not scalable when N_T is sufficiently large.

Scalability Evaluation: Adaptive IoT Trust is not scalable in the storage cost when N_T is sufficiently large. IoT-HiTrust and ObjectiveTrust are both scalable in the communication and storage cost. However, IoT-HiTrust achieves better scalability than ObjectiveTrust for two reasons: (a) with the hierarchical mobile cloud IoT architecture under IoT-HiTrust, an IoT device only communicates with its local cloudlet over a short radio-range distance when forwarding service quality experience results and social similarity information to the cloud or querying trust data from the cloud, whereas under ObjectiveTrust, an IoT device communicates with the cloud over a long haul distance; (b) IoT-HiTrust collects social relationships between each pair of IoT devices dynamically when IoT devices encounter each other, while ObjectiveTrust must rely on the existence of a

friendship social network graph for specifying social relationships. Point (b) above is especially problematic for scalability of ObjectiveTrust because it will be costly if not impossible to obtain a global friendship social network graph when a huge number of IoT devices change their social profiles dynamically.

6.4.2 Environment Setup

Table 6.2: Parameter List for Performance Evaluation.

Parameter	Meaning	Default
N_T	Number of IoT devices	2000
N_u	Number of users	500
N_C	Number of cloud servers	10
P_M	% of malicious users	30%
P_C	% of high centrality nodes	0%
λ	Service request rate per node	1/day
σ_c	Standard deviation of error	1%
$m \times m$	Cloudlet regional area	16×16
$SWIM_S$	SWIM slope	1.45
$SWIM_{pt}$	SWIM maximum pause time	4 hrs
$SWIM_{npp}$	SWIM # popular places per node	[10, 25]
$n_{f,lc}$	# friends per low-centrality node	[10, 50]
$n_{f,hc}$	# friends per high-centrality node	[100, 500]
N_{CoI}	# communities of interest	50
n_{CoI}	# communities of interest per node	[10, 25]
n_r	# of recommenders accepted	10
$w_f : w_s : w_c$	Weight ratio of social relations	1/3:1/3:1/3

The environment setup for performance evaluation of IoT-HiTrust is the same as that in Chapter 5 except that Chapter 5 is for a centralized cloud IoT system and Chapter 6 is for a hierarchically structured 3-tier IoT system. To be self-contained, the environment setup is described again in this section in full detail at the expense of somewhat repetition. The environment setup follows the model parameters as listed in Table 6.2 and is explained as follows. We consider a large IoT $m \times m$ "cloudlet" regional area with N_T = 2000 IoT devices, where each "cloudlet" region (location) is also a square with the width and height equal to wireless radio range so that nodes within a grid can communicate with the local cloudlet in the grid. The boundary locations are wrapped around (i.e., a torus is assumed) to avoid end effects. These IoT devices are randomly assigned to N_U = 500 users, with each user having 4 devices on average. The number of cloud servers in the data cloud center is N_C = 10 such that each cloud server can approximately handle N_T/N_C = 200 IoT devices.

The % of malicious users (P_M) is a model parameter whose effect will be analyzed via a sensitivity analysis. Malicious users are chosen randomly from N_U = 500 users. A malicious user will perform self-promoting, bad-mouthing, ballot-stuffing, discriminatory, and opportunistic service attacks as described in Chapter 3.3. In particular, a malicious user u_y can provide a bad recommendation $t_{y,i}^d$=0 (see Equation (6.7)) against a good device i for bad-mouthing attacks, and conversely a good recommendation $t_{y,i}^d$=1 for a malicious device i for ballot-stuffing attacks. Our protocol handles ballot-stuffing and bad-mouthing attacks by recommendation filtering during the computation of the indirect trust $t_{x,i}^r$ in Equation (6.7).

The % of high centrality users (P_C) is also a model parameter whose effect will be analyzed. Users are connected through social networks represented by a friendship matrix and a CoI matrix where an entry xy is 1 meaning that user x and user y are friends and members of a CoI, respectively. Each low centrality user has $n_{f,lc}$ = [10, 50] friends, and each high centrality user has $n_{f,hc}$ = [100, 500] friends populated randomly. There are N_{CoI} = 50 CoIs and each user has n_{CoI} = [10, 25] CoIs.

We consider these users moving according to the small world in motion (SWIM) mobility model [71] modeling human social behaviors for the purpose of assessing the social contact similarity metric between any pair of users. In SWIM [71], a node has a home location and $SWIM_{npp}$= [10, 20] popular places populated randomly out of the $m \times m$ locations in the system. A node makes a move to one of the population places based on a prescribed pattern. The probability of a location being selected is higher if it is closer to the node's home location or if it has a higher *popularity* (visited by more nodes). When reaching the destination, the node pauses at the destination location for a period of time following a bounded power law distribution. We set the slope of the SWIM mobility model ($SWIM_S$) to 1.45 (as in [71]) and the upper-bound pause time ($SWIM_{pt}$) to 4 hours. The encounter time interval for any two nodes is a bounded power-law distribution between [10 minutes, 2 days], which models the social contact behavior of any two nodes.

Three IoT devices in a region are selected as cloudlet devices periodically responsible for caching and relaying service experience reports, trust score queries, and responses for IoT devices in the region. Direct trust of node i toward node j is assessed upon completion of a service request from node i to node j. Each node requests services from a selected device with a time interval following an exponential distribution with parameter λ, with 1/day being the default unless otherwise specified. The trust update interval Δt is 2 hours. The system runs continuously although trust convergence is achieved in less than 200 hours.

The user satisfaction levels of service experiences, i.e., $f_{x,i}$ in the range of [0, 1] from user u_x about d_i's service quality, are from a real web service dataset [107] and are used as "ground truth" based on which the accuracy of our trust protocol is assessed. Since the direct trust of user u_x toward service provider d_i (i.e., $t_{x,i}^d$) is calculated by Equation (6.5) with "ground truth" user service experiences as input, $t_{x,i}^d$ essentially is equal to ground truth. However, we account for the presence of noise in the IoT environment (i.e., error of assessing user satisfaction level received) by considering a standard deviation parameter σ_c (set to 1% as default) to reflect the deviation of the actual user satisfaction level recorded in the database from the direct trust evaluation outcome $t_{x,i}^d$. The decay parameter λ_d is set to 10^{-4} as in [24] for heavy discounting old experiences. Initially, $t_{x,i}$ is set to 0.5 (ignorance) by user u_x for all i's. Then, trust is updated dynamically as nodes encounter each other, as services are requested and rendered, and as trust feedback are acquired. n_r=10 for the set size of U in Equation (6.7) for calculating $t_{x,i}^r$. We consider $w_f = w_l = w_c = 1/3$ considering friendship, social contact, and community of interest are equally important.

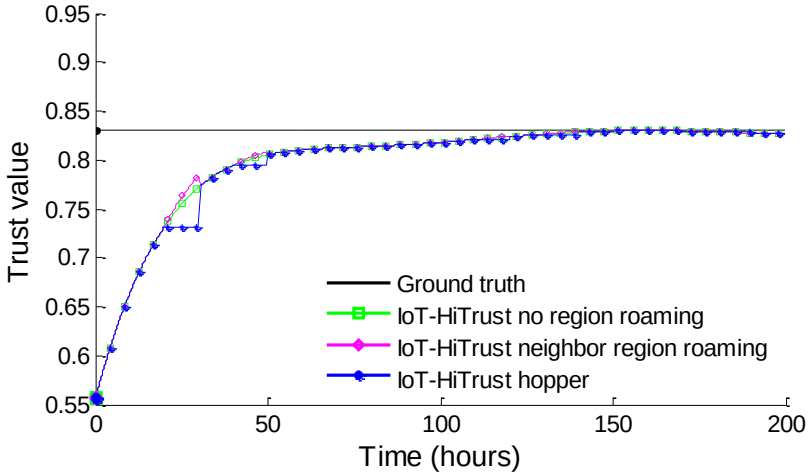


Figure 6.7: Trust Value of a Good Node under Intermittent Disconnection.

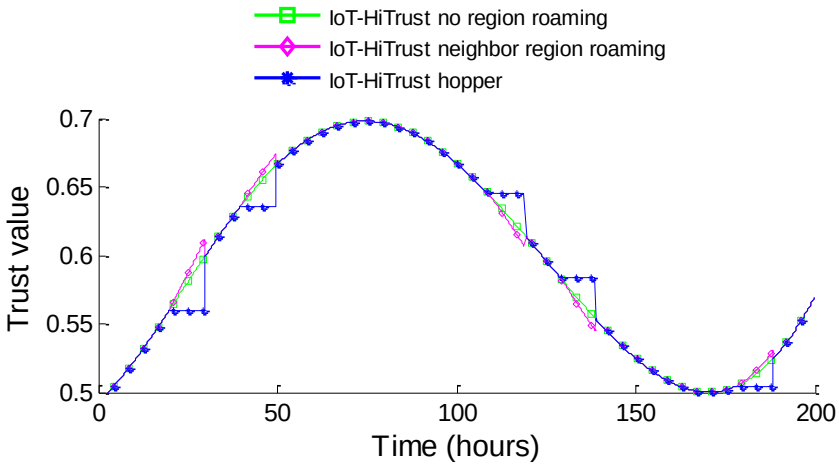


Figure 6.8: Trust Value of a Bad Node under Intermittent Disconnection.

6.4.3 IoT-HiTrust Performance Characteristics

In this section, we report ns-3 simulation results on IoT-HiTrust performance characteristics in response to user and environment dynamics.

Figure 6.7 shows IoT-HiTrust performance in terms of trust accuracy, convergence, and resiliency against attacks. It shows the progressive trust value of a "good" IoT device as assessed by the cloudlet in the home region where the IoT device initially resides vs. time for the case in which P_M = 30%. The "ground truth" trust value of this good device is depicted by the solid line at 0.83. Note that the "ground truth" trust value of this good node is not 1 because $f_{x,i}$ (service satisfaction experience of user u_x toward d_i) retrieved from the trace dataset [107] is not necessarily 1. The progressive trust value measured by IoT-HiTrust is depicted by dashed lines. We consider the scenario in which the "good" IoT device's cloudlet is disconnected due to network disconnection at times (marked by zig-zag patterns) during which it can only perform disconnected trust assessment. The label "no region roaming" means that the target IoT device stays in the same region all the time. The label "neighbor region roaming" means that the target IoT device stays in the home region but roams to neighbor regions from time to time. The label "hopper" means that the target IoT device moves across region boundary frequently. We see that trust accuracy decreases as the target "good" IoT device moves across the regional boundary more frequently. However, given time, all cases eventually converge after sufficient data are collected to allow accurate trust assessment.

Correspondingly, Figure 6.8 shows IoT-HiTrust performance in terms of the trust value of a "bad" IoT device vs. time when P_M = 30%. This bad node's trust value is up

and down because of opportunistic service attacks performed by the bad node. We again confirm that trust accuracy decreases as the target "bad" IoT device moves across the regional boundary more frequently. However, trust accuracy is restored as soon as the home regional cloudlet is reconnected. Figure 6.7 and Figure 6.8 verify that our IoT-HiTrust protocol is effective to deal with intermittent disconnection for providing service continuity, especially for IoT devices that do not move much such as heavyweight IoT devices.

In summary, relative to prior work [9][10][11][23][27], Figure 6.7 and Figure 6.8 demonstrate that our hierarchical trust management achieves efficiency and scalability, without compromising trust accuracy.

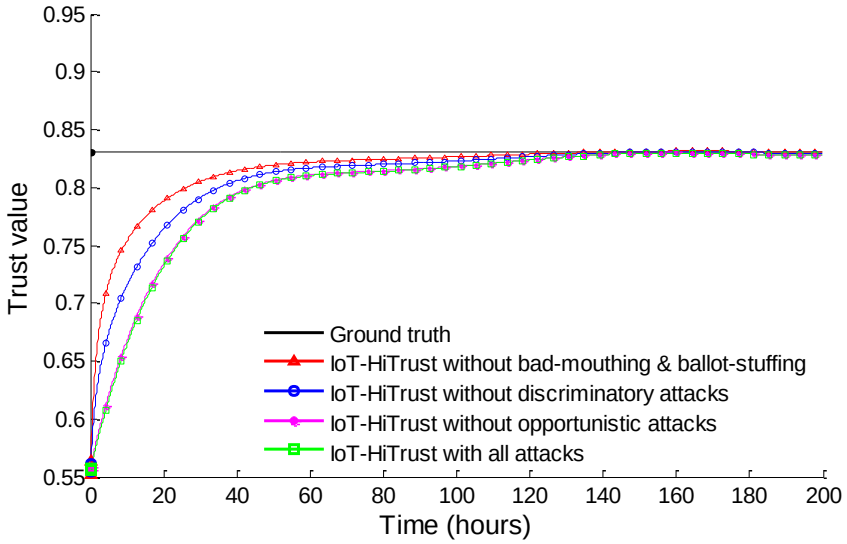


Figure 6.9: Trust Value of a Good Node under Various Attack Types.

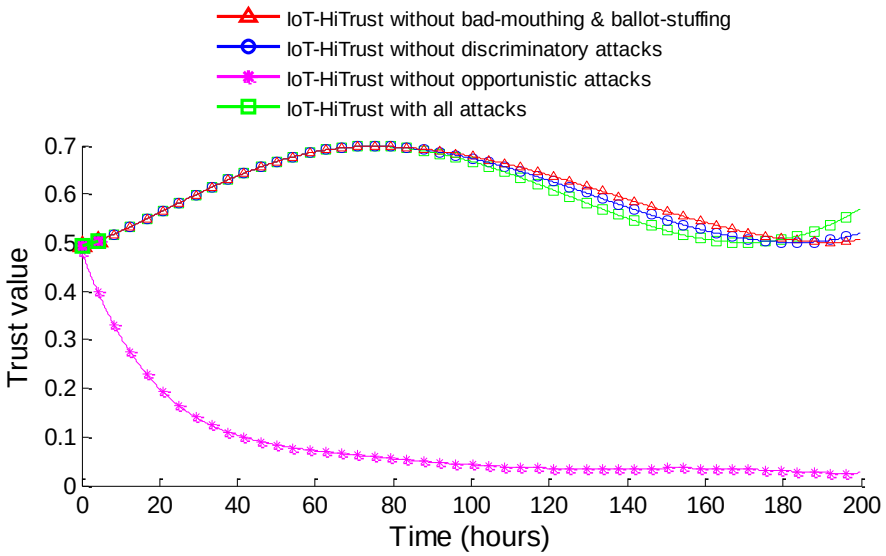


Figure 6.10: Trust Value of a Bad Node under Various Attack Types.

We test the sensitivity of the results w.r.t. the attack type in Figure 6.9 and Figure 6.10.

Figure 6.9 demonstrates the effect of the attack type on trust accuracy, convergence, and resiliency of a "good" target node that roams between neighbor regions. The curve labeled by "with all attacks" (green curve) corresponds to the curve in Figure 6.7. There are three other curves, i.e., "without bad-mouthing and ballot-stuffing attacks," "without discriminatory attacks," and "without opportunistic attacks," each with a particular attack type being removed so as to see its absence on accuracy, convergence, and resiliency. We see from Figure 6.9 that recommendation attacks (i.e., bad-mouthing and ballot-stuffing attacks combined) have the biggest effect on accuracy, convergence, and resiliency because without them (red curve) the system can approach ground truth (black curve) in the shortest amount of time. Discriminatory attacks also have some effect on accuracy, convergence, and resiliency because without them (blue curve) the system can converge faster than with them (green curve), although the sensitivity is not as high. Lastly opportunistic attacks have the least effect on accuracy, convergence, and resiliency because without them (pink curve) the system does not converge any faster than with them (green curve). This is so because a bad node performing opportunistic attacks will only affect its own trust value, not the "good" target node's trust value. Therefore the most severe attack type appears to be the recommendation attack as it will ruin the reputation of the "good" target node.

Figure 6.10 demonstrates the effect of the attack type on trust accuracy, convergence, and resiliency of a "bad" node that roams between neighbor regions. Similarly the

curve labeled by "with all attacks" (green curve) corresponds to the curve in Figure 6.8. There are three other curves, i.e., "without bad-mouthing and ballot-stuffing attacks," "without discriminatory attacks," and "without opportunistic attacks," each with a particular attack type being removed so as to see its absence on accuracy, convergence, and resiliency. Since the target node is a "bad" node, we see from Figure 6.10 that opportunistic attacks have the most severe effect on accuracy, convergence, and resiliency because without them (pink curve) the system converges to ground truth (zero for a bad node) while with them the system converges to the up and down curve (green curve). This is so because the "bad" target node performing opportunistic service attacks will alternate between behave and misbehave to keep its trust value between the high threshold and low threshold. We see from Figure 6.10 that the other two attack types do not have a high impact on the trust value of this "bad" target node. In practice, a bad node will always perform opportunistic service attacks to disguise itself as a good node without being caught. The best the system can do is to accurately track a bad node's trust status to refrain it from providing bad service and to decrease the chance of selecting bad nodes for providing service. This is illustrated by two real-world mobile cloud IoT applications described in Section 6.5 below.

6.4.4 Sensitivity and Comparative Analysis

In this section, we report the sensitivity of IoT-HiTrust performance with respect to the % of malicious users (P_M) and the % of high centrality users (P_C). We again compare IoT-HiTrust performance with Adaptive IoT Trust [27] and ObjectiveTrust [86].

We first analyze the effect of the % of malicious users (P_M). Figure 6.11 compares IoT-HiTrust (red surface) with Adaptive IoT Trust (orange surface) and ObjectiveTrust (green surface) as P_M (% of malicious nodes) varies from 20% to 40% for a non-malicious node randomly chosen. All other parameters use the default settings as in Table 6.2. The ground truth trust value of the non-malicious node is marked by a solid line at 0.83. We first note that for all protocols, the estimated trust score of the non-malicious nodes approaches the ground truth value as time progresses as more evidence is collected. Also, the deviation from the ground truth trust score increases as P_M increases because more malicious nodes perform attacks. We see that IoT-HiTrust (red) consistently outperforms Adaptive IoT Trust (orange) and ObjectiveTrust (green), especially when the % of malicious nodes is high (40%). We attribute IoT-HiTrust performing better than Adaptive IoT Trust to the fact that IoT-HiTrust can leverage cloud service to aggregate broad evidence from all nodes to achieve the minimum trust bias. We attribute IoT-HiTrust performing better than ObjectiveTrust to the fact that unlike ObjectiveTrust, IoT-

HiTrust considers "subjective trust" which takes a customer's own service experiences into consideration and it can dynamically adjust the weight associated with a customer's own service experiences to effectively cope with malicious attacks and improve trust accuracy.

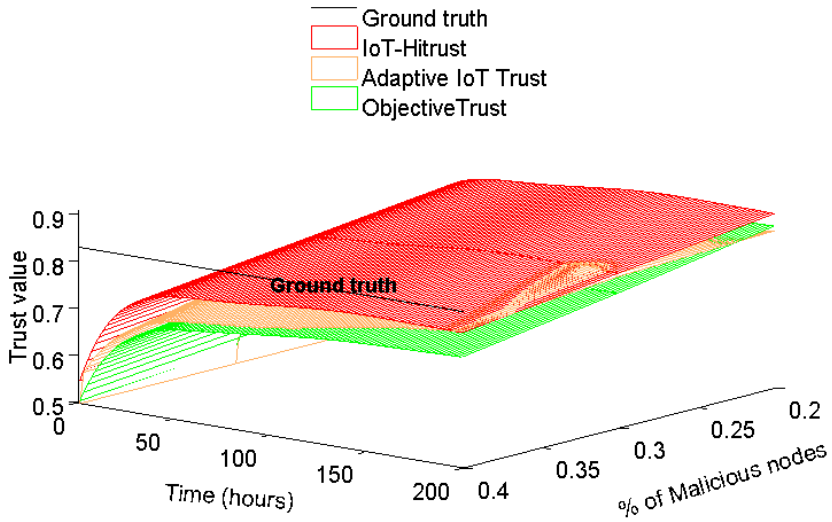


Figure 6.11: Performance Comparison of Trust Convergence, Accuracy and Resiliency in 3-D View with PM (% of Malicious Nodes) ranging from 20% to 40%.

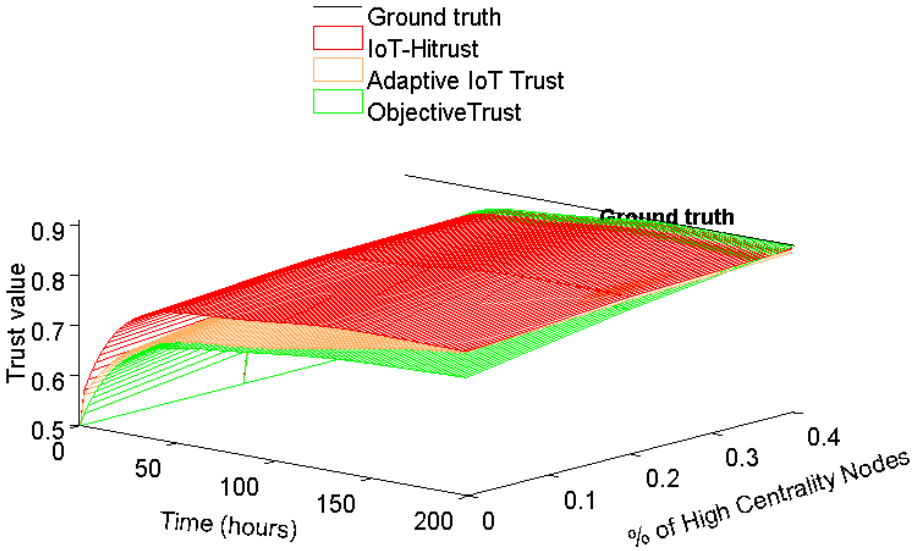


Figure 6.12: Performance Comparison of Trust Convergence, Accuracy and Resiliency in 3-D View with PC (% of High Centrality Nodes) ranging from 0% to 40%.

Next we analyze the effect of % of high centrality users (P_C). Figure 6.12 compares IoT-HiTrust (red surface) with Adaptive IoT Trust (orange surface) and ObjectiveTrust (green surface) as P_C (% of high centrality nodes) varies from 0% to 40% for a non-malicious node randomly chosen. All other parameters use the default settings as in Table 6.2. We see IoT-HiTrust (red) also consistently performs better than Adaptive IoT Trust (orange) and ObjectiveTrust (green), especially when P_C is low. All protocols perform comparably when P_C is high. The reason is that all protocols consider social relationships (including friendship which maps to centrality) for recommendation filtering. So when P_C is high, all have excellent recommendation filtering taking place, resulting in the estimated trust score approaching the ground truth trust value. When P_C is extremely high (40%), ObjectiveTrust has a slight edge over IoT-HiTrust and Adaptive IoT Trust, because ObjectiveTrust uses a weighted sum of centrality and filtered opinions for trust computation, and directly uses centrality (by means of the overall trust score computation) as credibility for recommendation filtering. One should note that, however, a social network normally has only a small number of nodes with high centrality [87], so for a typical social network with less than 20% high centrality nodes, IoT-HiTrust (red) still outperforms ObjectiveTrust (green).

In summary, relative to Adaptive IoT Trust [27] and ObjectiveTrust [86], our work addresses scalability while achieving "subjective trust" evaluation accuracy.

6.5 Applicability to Real World IoT Applications

6.5.1 Case study 1: Smart City Travel Service Composition

The first case study is taken from [23]: Bob has never traveled to Washington DC, so he is excited but also nervous about the quality of service he will receive during his visit. He is aware of the fact that DC is a smart city so he registers his smartphone to the *travelers-in-Washington-DC* social network. He also downloads an *augmented map* social IoT application [6] to run on his smartphone, allowing his Near Field Communications (NFC) equipped smartphone to browse a tag-augmented DC map wherever he goes sightseeing. This tag-augmented map automatically connects Bob's smartphone to IoT devices available upon encounter, which provide information, food, entertainment (e.g., ticket purchasing), and transportation services. Bob instructs his smartphone to make selection decisions dynamically, so it can leverage new information derived from direct experiences and recommendations received from IoT devices it encounters. In response to a service request issued by Bob, his smartphone must (a) gather sensing data or information collected from the physical environment based on either self-observations or

recommendations; (b) formulate a service plan based on the results gathered; and (c) invoke necessary services to meet Bob's service demand and requirements.

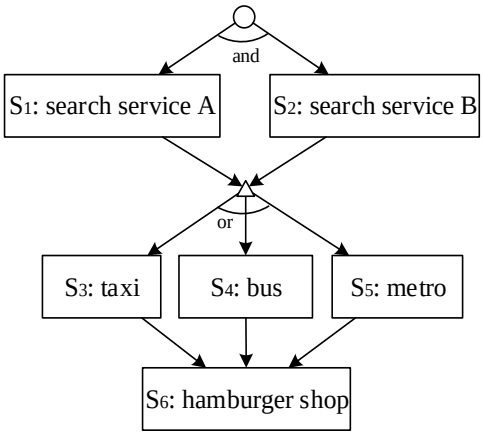


Figure 6.13: A Service Flow Structure for the Smart City Travel Service Composition Application.

To this end, the augmented map travel service composition application running on Bob's smartphone composes a service workflow plan as shown in Figure 6.13 in response his service request "Fill me with the best grilled hamburger within 20 minutes under a $30 budget." With the service plan formulated, Bob's smartphone selects the best service providers out of a myriad of service providers to execute the service plan. The objective of this trust-based service composition application running on Bob's smartphone is to select the most trustworthy IoT nodes for providing services specified in the flow structure subject to the time and budget constraints (20 minutes and 30 dollars) such that the overall *trustworthiness* score representing the goodness of the service composition is maximized.

In Figure 6.13, there are 6 atomic services connected by three types of workflow structures: *sequential*, *parallel* (AND), and *selection* (OR). Each service would have multiple service provider candidates. In this case, the overall trustworthiness score of this service composition application can be calculated recursively in the same way the reliability of a series-parallel connected system is calculated. Specifically, the trustworthiness score of a composite service (whose trustworthiness score isT_s) that consists of two subservices (whose trustworthiness scores are T_1 and T_2) depends on the structure connecting the two subservices as follows:

a) Sequential structure: $T_s = T_1 \times T_2$;
b) Selection structure (OR): $T_s = \max(T_1, T_2)$;
c) Parallel structure (AND): $T_s = 1 - (1 - T_1) \times (1 - T_2)$.

For the flow structure in Figure 6.13, the outmost structure is a sequential structure connecting (S_1 S_2), (S_3 S_4 S_5), and S_6 out of which (S_1 S_2) is a parallel structure and (S_3 S_4 S_5) is a selection structure.

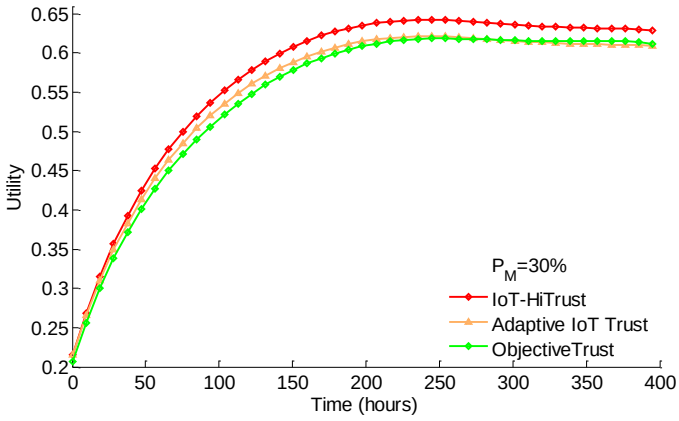


Figure 6.14: Utility Score of the Smart City Travel Service Composition Application.

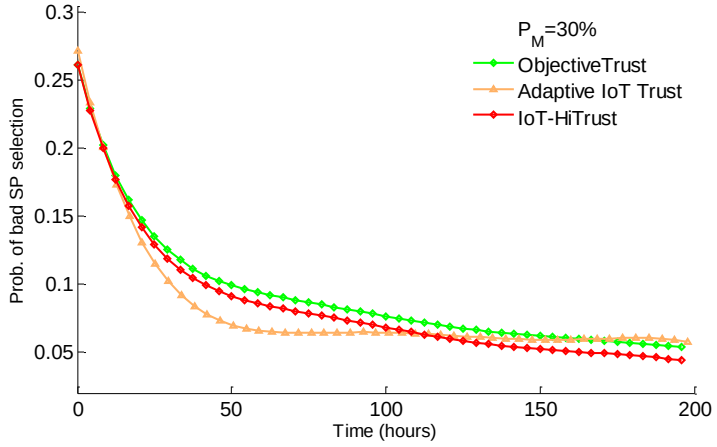


Figure 6.15: Probability of a Bad SP Being Selected for the Smart City Travel Service Composition Application.

Figure 6.14 shows the ns-3 simulation results. The 3-tier hierarchical mobile cloud environment is setup the same way as discussed in Chapter 6.3 (see Table 6.2). We observe that trust-based service composition with IoT-HiTrust (red line) outperforms both Adaptive IoT Trust (yellow line) and ObjectiveTrust (green line) as time progresses when more trust evidence is gathered. Figure 6.15 shows the percentage of bad nodes selected for the augmented map travel service composition application. Again as time progresses, IoT-HiTrust outperforms all baseline schemes in selecting a good SP to exe-

cute the service request. Initially Adaptive IoT Trust (which is a distributed trust protocol) performs better than both IoT-HiTrust and ObjectiveTrust (both are centralized trust protocols) because centralized trust protocols initially do not have access to a broad range of trust evidence collected from other IoT devices. The advantage of IoT-HiTrust over Adaptive IoT Trust increases as time progresses after the system gathers sufficient service data from all nodes having service experience with a target node. We also note that the advantage persists even the percentage of malicious nodes is 30% in the system. On the other hand, IoT-HiTrust performs better than ObjectiveTrust because it considers "subjective trust" which takes a customer's own service experiences with a target service provider into consideration as it computes the trust score of a customer toward a target service provider. In the case in which the service experience is plenty, IoT-HiTrust will dynamically put a higher weight on direct service experience and conversely a lower weight on the recommendations to compute the trust score for the purpose of minimizing trust bias. In the case in which the service experience is not plenty but recent, IoT-HiTrust will adaptively adjust the weights on direct service experiences and recommendations such that the computed score would match the recent service satisfaction outcome. In both cases, it shields the customer from malicious recommendation attacks, which is a common problem for reputation-based trust management protocols like ObjectiveTrust.

6.5.2 Case study 2: Air Pollution Detection and Response

The second case study is for the *Fairfax County Hazard Detection and Response Team* charged to monitor the pollution levels of CO, NO_2, SO_2, and O_3 for all cities under the county so as to take appropriate actions if the air pollution level is above a tolerance threshold. Since the area to be covered is rather large, the county officials only install a few county-sensors in more strategic and populated areas to collect air pollution data. To cover the whole county area air quality detection, the county officials also encourage environment-health-conscious civilians driving or carrying air pollution detection capable vehicles or smartphones [51] to report air pollution data.

In case of emergency, the county officials can request IoT devices in a particular location to immediately report their sensing results to their home cloud servers through their local cloudlets. Because the county officials have registered this cloud service, a home cloud server upon receiving a sensing report will inform the county officials (running as an IoT device) of the sensing report. Also the county officials send queries via IoT-HiTrust to get the trustworthiness scores of these IoT devices who had reported sensing results. To know if a location has acceptable air quality, the county officials (running as node i) accept results (S_j) from 200 most trustworthy IoT devices (which have the highest t_{ij} trust values as determined by IoT-HiTrust) for the air quality detec-

tion service out of a total of 2000 nodes, and compute a trust-weighted average $\sum_{j=1}^{200}(t_{ij}/\sum_{j=1}^{200} t_{ij}) \times S_j$ for each air pollutant (e.g., CO). If the level exceeds a minimum threshold (e.g., above 70 ppm for CO), the county officials push alerting text to IoT devices in the affected area.

Using the ns3 simulator, we simulate the above system populated with 2000 IoT devices capable of detecting and reporting CO air pollutant levels. The 3-tier hierarchical mobile cloud environment is setup the same way as discussed in Chapter 6.3 (see Table 6.2). The CO level is simulated to be in the range of [60, 70 ppm] in various locations. The percentage of bad nodes is set at P_M =30%. A malicious node always reports CO readings above 70 ppm in the range of [70, 120 ppm] regardless of location in order to confuse the county official. Also a malicious node always performs bad-mouthing attacks (saying a good node's sensing result is not trustworthy in the user satisfaction report) and ballot-stuffing attacks (saying a bad node's sensing result is trustworthy).

We again compare IoT-HiTrust with Adaptive IoT Trust and ObjectiveTrust. We measure two performance metrics for performance analysis: (a) the trust-weighted average CO reading vs. ground truth (i.e., the actual CO level at a specific location and a particular time); (b) the accuracy of selecting trustworthy participants.

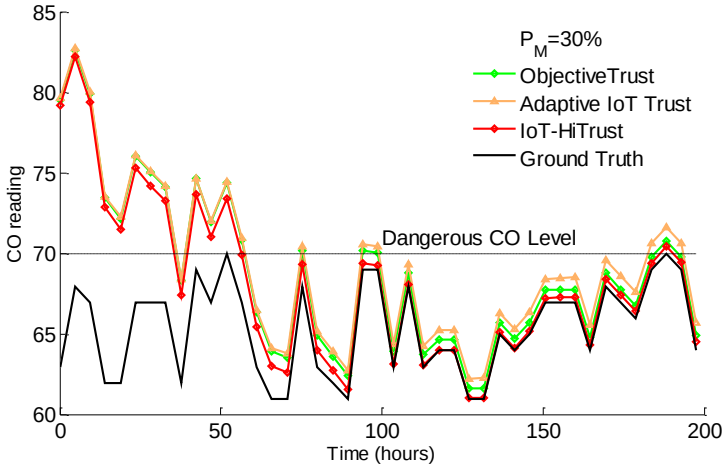


Figure 6.16: Performance Comparison of Trust-Weighted Average CO Readings for the Air Pollution Detection and Response Application.

Figure 6.16 shows the trust-weighted average CO readings vs. time (each time point is a CO detection service request) with the percentage of bad nodes P_M set at 30%. We observe that IoT-HiTrust (red line) leveraging the proposed mobile cloud hierarchy can provide CO readings very close to ground truth (black line) as time progresses. Further, IoT-HiTrust outperforms Adaptive IoT Trust (yellow line) and ObjectiveTrust (green line) in terms of accuracy, convergence, and resiliency. We mark a "Dangerous CO Lev-

el" line at which the CO reading is equal to or above 70 in the graph. We see that in many sensing time points such as at 65, 75, 90, 100, and 185, IoT-HiTrust would report the CO level is not dangerous as the ground truth is, but either Adaptive IoT Trust or ObjectiveTrust would falsely report the CO level is dangerous since the trust-weighted CO level average computed is above 70. This demonstrates that IoT-HiTrust is more resilient to malicious attacks (30% are malicious) than either Adaptive IoT Trust or ObjectiveTrust in this application. Figure 6.17 shows the percentage of bad nodes selected to provide sensing results. We see again IoT-HiTrust (red line) outperforms Adaptive IoT Trust and ObjectiveTrust as time progresses.

We attribute the superiority of IoT-HiTrust over Adaptive IoT Trust to its ability to effectively aggregate trust evidence from all nodes who have had sensing service experiences with a target IoT device, leveraging our effective and efficient localized report-and-query paradigm, not being limited by node encountering experiences as in Adaptive IoT Trust. We attribute the superiority of IoT-HiTrust over ObjectiveTrust to its ability to accurately compute the "subjective trust" which takes a customer's own service experiences into consideration as opposed to the "objective trust" which only takes the common belief or reputation into consideration as in ObjectiveTrust, and also to its ability to dynamically adjust the weights associated with direct trust and indirect trust to minimize trust bias, based on the customer's past and recent own service experiences.

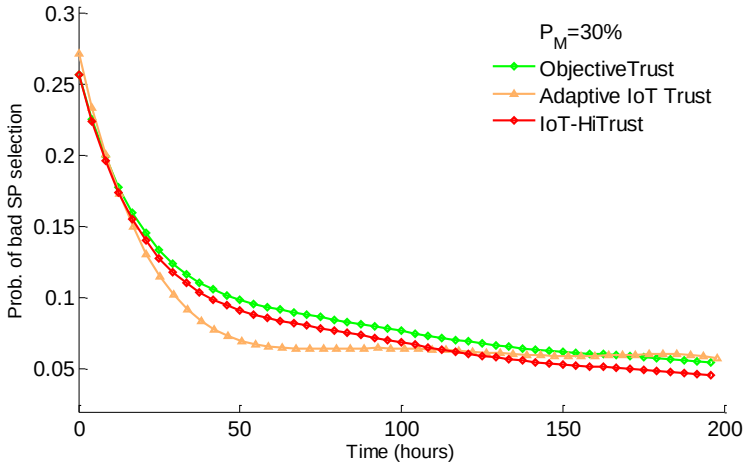


Figure 6.17: Percentage of Bad IoT Devices Selected to Provide CO Sensing Service for the Air Pollution Detection and Response Application.

6.6 Summary

In this chapter, we designed and analyzed a scalable 3-tier hierarchical cloud architecture for supporting integrated mobility, service, and trust management of service-oriented IoT systems, and exemplified it with the design of a hierarchical trust management protocol called IoT-HiTrust for large mobile cloud IoT systems. We verified that IoT-HiTrust is effective for dealing with intermittent disconnection and cloud failure while preserving desirable trust accuracy, convergence and resiliency properties, especially for IoT devices that do not move much such as heavyweight IoT devices. We also demonstrated its applicability by applying IoT-HiTrust to a smart-city travel service composition application, and an air pollution detection and response application. Our results support its superiority over Adaptive IoT Trust [27] and ObjectiveTrust [86] in achieving scalability and maximizing application performance, without compromising trust accuracy, convergence and resiliency properties.

Chapter 7 Conclusion

The goals of the dissertation research are to:

1. develop trust protocols utilizing general trust management techniques for distributed, centralized, and hybrid IoT applications;
2. verify desirable properties (including solution quality, accuracy, convergence, resiliency, and scalability) having been achieved;
3. prove the designed protocols outperform contemporary trust protocols;
4. prove the validity of our trust protocols with real-world IoT applications running in distributed, centralized, and hybrid IoT environments.

This chapter summarizes completed work reflecting the goals above, future work, and the schedule for completing the dissertation research.

7.1 A Summary of Dissertation Research Publications

We have developed a distributed trust protocol called Adaptive IoT Trust for distributed IoT applications (Chapter 4), a centralized trust protocol called IoT-TaaS for centralized IoT applications with cloud access (Chapter 5), and a hierarchical trust management protocol called IoT-HiTrust for hybrid IoT applications (Chapter 6). We have verified that desirable properties, including solution quality, accuracy, convergence, resiliency, and scalability have been achieved through formal proof or simulation. Furthermore, we have tested Adaptive IoT Trust, IoT-TaaS, and IoT-HiTrust with real-world IoT applications for validity. Through model-based analysis with simulation validation, we have demonstrated that our proposed IoT trust protocols outperform contemporary IoT trust protocols, and are resilient to trust related attacks.

The work completed so far has resulted in 6 journal publications, 8 conference publications, and 2 journal/conference paper submissions, listed below. At the end of each paper publication, we annotate the dissertation chapter in which part of the paper appears.

Papers Published:

Journal Publications:

1. J. Guo, I.R. Chen, and J.J.P. Tsai "A Survey of Trust Computation Models for Internet of Things Systems," *Computer Communications*, vol. 97, 2017, pp. 1-14. ([57] appearing in Chapter 2.)
2. I. R. Chen, J. Guo, and F. Bao, "Trust Management for SOA-based IoT and Its Application to Service Composition", *IEEE Transactions on Services Computing*, vol. 9, no. 3, 2016, pp. 482-495. ([27] appearing in Chapter 4.)
3. I. R. Chen, F. Bao, and J. Guo, "Trust-based Service Management for Social Internet of Things Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, Nov-Dec 2016, pp. 684-696. ([23] appearing in Chapter 4.)
4. I.R. Chen, J. Guo, and J.J.P. Tsai, "Trust as a Service for SOA-based Internet of Things," *Services Transactions on Internet of Things,* vol 1, no 1, June 2017, pp. 43-52. ([29] appearing in Chapter 5.)
5. I.R. Chen and J. Guo, "Hierarchical Trust Management of Community of Interest Groups in Mobile Ad Hoc Networks," *Ad Hoc Networks*, vol. 33, Oct. 2015, pp. 154-167. ([26] appearing in Chapter 6.)
6. I.R. Chen, J. Guo, F. Bao and J.H. Cho, "Trust Management in Mobile Ad Hoc Networks for Bias Minimization and Application Performance Maximization," *Ad Hoc Networks*, vol. 19, 2014, pp. 59-74. ([25] appearing in Chapter 6.)

Conference Publications:

1. J. Guo, I.R. Chen, D.C. Wang, J.J.P. Tsai, and H. Al-Hamadi, "A Case Study of IoT Participatory Sensing of Hazardous Ozone using Traces," *The 7th International Conference on Electronics, Communications and Networks*, Hualien, Taiwan, Nov. 2017. ([61] appearing in Chapter 5.)
2. J. Guo, I.R. Chen, and J.J.P. Tsai, "A Mobile Cloud Hierarchical Trust Management Protocol for IoT Systems," *5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, San Francisco, April 2017. ([60] appearing in Chapter 6.)
3. J. Guo, I.R. Chen, J.J.P. Tsai, and H. Al-Hamadi, "A Hierarchical Cloud Architecture for Integrated Mobility, Service, and Trust Management of Service-Oriented Internet of Things," *6th IEEE international conference on Innovative*

Computing Technology (Intech 2016), Dublin, Ireland, August 2016. ([58] appearing in Chapter 6.)

4. J. Guo, I.R. Chen, J.J.P. Tsai, and H. Al-Hamadi, "Trust-based IoT Participatory Sensing for Hazard Detection and Response," *1st International Workshop for IOT Systems Provisioning & Management in Cloud Computing (ISYCC 2016)*, Banff, Canada, Oct. 2016. ([59] appearing in Chapter 6.)
5. J. Guo and I.R. Chen, "A Classification of Trust Computation Models for Service-Oriented Internet of Things Systems," *12th IEEE International Conference on Services Computing (SCC 2015)*, New York, June 2015. ([56] appearing in Chapter 2.)
6. I.R. Chen and J. Guo, "Dynamic Hierarchical Trust Management of Mobile Groups and Its Application to Misbehaving Node Detection," *28th IEEE International Conference on Advanced Information Networking and Applications (AINA 2014)*, Victoria, Canada, May 2014. ([28] appearing in Chapter 6.)
7. I. R. Chen, J. Guo and F. Bao, "Trust Management for Service Composition in SOA-based Internet of Things Systems," *IEEE 2014 WCNC*, Istanbul, Turkey, April 2014, pp. 3486-3491. ([31] appearing in Chapter 4.)
8. I.R. Chen, J. Guo, F. Bao, and J.H. Cho, "Integrated Social and Quality of Service Trust Management of Mobile Groups in Ad Hoc Networks," *9th IEEE Conference on Information, Communications and Signal Processing (ICICS 2013)*, Tainan, Taiwan, Dec. 2013. ([32] appearing in Chapter 4.)
9. F. Bao, I. R. Chen, and J. Guo, "Scalable, Adaptive and Survivable Trust Management for Community of Interest Based Internet of Things Systems," *11th International Symposium on Autonomous Decentralized System (ISADS 2013), Mexico City*, Mexico, 2013. ([14] appearing in Chapter 4.)

<u>Papers Submitted:</u>

1. J. Guo, I.R. Chen, and J.J.P. Tsai, "Scalable Trust-based Service Management for Mobile Cloud IoT Systems," *IEEE Transactions on Network and Service Management*, Jan 2018. (Chapter 6.)
2. J. Guo, I.R. Chen, D.C. Wang, J.J.P. Tsai, and H. Al-Hamadi, "Trust-based IoT Cloud Participatory Sensing of Hazardous Ozone," *Wireless Personal Communications,* Jan 2018. (Chapter 5.)

7.2 Future Research Directions

There are several future research directions extending from this dissertation research:

1. The first and foremost research direction is to explore and apply untouched trust aggregation techniques based on belief theory or regression analysis to IoT systems. Regression analysis especially is applicable when IoT nodes can access a centralized trust manager, say, located on a cloud because of the limited computing power of IoT devices for executing computationally expensive statistical analysis. Feedback data reflecting service context information, such as capability and energy of the SP, traffic congestion conditions of the network, and service quality feedback itself, can be propagated to the cloud for complex statistical analysis to better connect context information with service quality, thus providing a more accuracy estimate of service quality trust of a SP in question.

2. The second research direction is to further explore innovative social trust metrics and the best way to combine them for IoT trust computation. The reason is that IoT systems are inherently social oriented because IoT devices are owned by humans. In the literature, using social similarity to rate a trustee or a recommender is emerging. However, how to combine several social metrics such as friendship, centrality, social contact, and community-of-interest, into social similarity is still an open problem. Further, other than similarity, there are also many IoT social properties such as selfishness, cooperativeness, integrity, and honesty that need to be further explored. Properly exploring social relationship between IoT devices can effectively defend bad-mouthing and ballot-stuffing attacks which happen commonly in social IoT systems. This dissertation research has developed a *dynamic weighted sum* method based on *adaptive filtering* to combine social relationships and demonstrated its effectiveness, but there may be new social trust metrics yet discovered or a better way to combine social trust metrics for IoT trust computation. Potential methods to investigate further include (a) context-sensitive dynamic weighted sum, (b) setting a minimum threshold for each trust property without combining multiple properties into one, and (c) using one trust property as the main one which is scaled by other trust properties serving as confidence. A potential technique to use for context-sensitive dynamic weighted sum is regression analysis [98] to link context information with trust accuracy and/or application performance so as to determine the best weight assignment

3. The third research direction is to devise and validate a trust computation model that can defend against all attacks. In this dissertation research, we have considered ways to defend against self-promotion attacks, bad-mouthing attacks, ballot-stuffing attacks, opportunistic service attacks, and discriminatory attacks by

leveraging social trust metrics. However, more efforts are required to verify with emerging service-oriented social IoT systems. Also, there is a need to further model malicious behaviors (e.g., [83][78][46][43]) which can happen in service-oriented social IoT systems, and devise as well as validate effective defense mechanisms against such malicious behaviors.

4. The fourth research direction is to integrate cloud service with trust management service to create an IoT service-community cloud utility, aka, Trust as a Service (TaaS), for centralized or hierarchically structured IoT systems. An IoT service-community for example can be an e-health group paying particular attention to air pollution for the welfare of a group of users who may suffer from polluted air quality, an intelligent your-ride-on-demand IoT group (like Uber), or a smart city group consisting of visitors, merchants, restaurants, and entertainment business entities, etc. TaaS would be a perfect service provided by the cloud to members in each of these groups. Service reputation feedbacks along with service context information can be fed into the cloud for a complex yet complete statistical analysis. Users requesting a service or a composite service (i.e., several services bundled together via service composition and binding) can be assured of trustworthy, high-quality service, as a result of TaaS being applied to such an IoT service-community group. More IoT applications need to be further explored to fulfill the full potential of such an IoT service-community TaaS cloud utility.

5. Lastly, there is a lack of a holistic design for scalable, adaptive and survivable trust computation for social IoT systems. The fifth research direction is to consider a more holistic design to manage "integrated" mobility, service and trust information of a large number of IoT devices, in a scalable, secure, reliable, and efficient manner. A possible solution is to integrate the design concepts currently existing in hierarchical trust management [26] [28] [13] [22], hierarchical mobility management [55] [73] [35] [37], admission control [39][41][42][105], and tiered cloud architectures [90]. While a node in a hierarchical mobility management architecture is a router responsible for keeping track of location information only (where and how to route), a node in a hierarchical cloud management architecture is a cloud server responsible for keeping track of "integrated" information including location, trust, and service information. A lower-level cloud server (e.g., a cloudlet or a private cloud) keeps track of IoT devices in its directly covered service area. A higher level cloud server (e.g., a public cloud) in the architecture keeps track of status of all IoT devices covered by all local cloud servers below it. Should an IoT device roam from one cloud service area to another, a "service handoff" ensues causing this IoT device's location, trust and service information to be transferred between the two involving cloud servers. Such an IoT

framework can track IoT devices not only in trust status, but also in service and mobility status dynamically to achieve the potential of anytime anywhere service-oriented IoT applications in the 21th century. The hierarchical cloud architecture for integrated mobility, service, and trust management with IoT-HiTrust as the underlying IoT trust management protocol (which we proposed in Chapter 6 of the dissertation research) is a first step toward this research direction, but more efforts are required to fulfill the potential.

7.3 Research Milestones

Table 7.1 below lists the milestones accomplished in the dissertation research.

Table 7.1: Dissertation Research Milestones

Date	Milestone
Feb. 2013	Survey related work on trust management of IoT systems.
May 2014	Develop a classification tree for classifying existing trust management protocols for IoT systems, with the intent to identify research gaps and future research areas.
Aug. 2014	Complete trust-based service management for distributed IoT systems and its applications to IoT service composition applications.
Dec. 2014	Complete a hierarchical trust management protocol for mobile group management and its applications to misbehaving node detection in mobile ad hoc networks.
May 2015	Complete a trust management protocol for a cloud-based centralized IoT system.
May 2016	Leverage experience learned from hierarchical trust management protocol for mobile group management to the design and analysis of a hierarchical trust management protocol for large scale hybrid IoT systems.
Nov. 2016	Ph.D. Preliminary Proposal Exam
May 2017	Complete the verification of protocol designs of Adaptive IoT

	Trust, IoT-TaaS, and IoT-HiTrust
Sept. 2017	Complete the validation of Adaptive IoT Trust, IoT-TaaS, and IoT-HiTrust with real-world IoT applications.
Oct. 2017	Ph.D. Research Defense
Feb 2018	Complete the responses toward the comments raised by the committee members.
March 2018	Ph.D. Final Defense

Acronym Table

Acronym	Definition
BMA	Bad-mouthing attacks
BSA	Ballot-stuffing attacks
CoI	Community of interest
DHT	Dynamic hash table
DST	Dempster–Shafer theory
MANET	Mobile ad hoc network
MSE	Mean square error
IoT	Internet of Things
O3COI	O3 community of interest
OOA	On-off attacks
OSA	Opportunistic service attacks
QoS	Quality of service
RFID	Radio frequency identification
SOA	Service-Oriented Architecture
SP	Service provider

SR	Service requester
SPA	Self-promoting attacks
TaaS	Trust as a service
WSN	Wireless sensor network

Notation Table

Notation	Definition
N_T	Number of IoT devices
N_u	Number of users
N_C	Number of cloud servers
P_M	% of malicious users
P_C	% of high centrality nodes
λ	Service request rate per node
σ_c	Standard deviation of error
$m \times m$	Regional area
$SWIM_S$	SWIM slope
$SWIM_{pt}$	SWIM maximum pause time
$SWIM_{npp}$	SWIM # popular places per node
$n_{f,lc}$	# friends per low-centrality node
$n_{f,hc}$	# friends per high-centrality node
N_{CoI}	# communities of interest
n_{CoI}	# communities of interest per node

n_r	# of recommenders accepted
$w_f : w_s : w_c$	Weight ratio of social relations
Δt	trust update interval
CompOver(n)	computational overhead
CommOver(n)	communication overhead
sim_f	Friendship Similarity
sim_l	Social Contact Similarity
sim_c	Community of Interest Similarity
$t_{x,i}$	trust value of user u_x toward device d_i
CL_x	local cloudlet of user u_x
CS_x	home cloud server of user u_x

Bibliography

[1] H. Amintoosi, S.S. Kanhere, M. Allahbakhsh, "Trust-based Privacy-aware Participant Selection in Social Participatory Sensing," *Information Security and Applications*, vol.20, pp.11-25, 2015.

[2] H. Al-Hamadi and I.R. Chen, "Redundancy Management of Multipath Routing for Intrusion Tolerance in Heterogeneous Wireless Sensor Networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, 2013, pp. 189-203.

[3] H. Al-Hamadi and I.R. Chen, "Adaptive Network Management for Countering Smart Attack and Selective Capture in Wireless Sensor Networks," *IEEE Transactions on Network and Service Management,* vol. 12, no. 3, 2015, pp. 451-466.

[4] H. Al-Hamadi and I.R. Chen, "Integrated Intrusion Detection and Tolerence in Homogeneous Clustered Sensor Networks," *ACM Trans Sensor Networks,* vol. 11, no. 3, 2015.

[5] H. Al-Hamadi and I.R. Chen, "Trust-Based Decision Making for Health IoT Systems," *IEEE Internet of Things Journal*, vol. 4, no. 5, 2017, pp. 1408-1419.

[6] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks,* vol. 54, no. 15, Oct. 2010, pp. 2787-2805.

[7] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The Social Internet of Things (SIoT) - When social networks meet the Internet of Things: Concept, architecture and network characterization," *Computer Networks,* vol. 56, no. 16, Nov. 2012, pp. 3594-3608.

[8] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a Social Structure to the Internet of Things," *IEEE Communication Letters,* vol. 15, no. 11, pp. 1193-1195, Nov. 2011.

[9] F. Bao, and I. R. Chen, "Dynamic Trust Management for Internet of Things Applications," *2012 International Workshop on Self-Aware Internet of Things*, San Jose, California, USA, 2012.

[10] F. Bao, and I. R. Chen, "Trust Management for the Internet of Things and Its Application to Service Composition," *IEEE WoWMoM 2012 Workshop on the Internet of Things: Smart Objects and Services*, San Francisco, CA, USA, 2012.

[11] F. Bao, I. R. Chen, and J. Guo, "Scalable, Adaptive and Survivable Trust Management for Community of Interest Based Internet of Things Systems," *11th International Symposium on Autonomous Decentralized System, Mexico City,* Mexico, 2013.

[12] F. Bao, Dynamic Trust Management for Mobile Networks and Its Applications, ETD, Virginia Polytechnic Institute and State University, May 2013.

[13] F. Bao, I. R. Chen, M. Chang, and J. H. Cho, "Hierarchical Trust Management for Wireless Sensor Networks and Its Applications to Trust-Based Routing and Intrusion Detection," *IEEE Trans. on Network and Service Management,* vol. 9, no. 2, 2012, pp. 161-183.

[14] F. Bao, I. R. Chen, and J. Guo, "Scalable, Adaptive and Survivable Trust Management for Community of Interest Based Internet of Things Systems," *11th IEEE International Symposium on Autonomous Decentralized System*, Mexico City, March 2013.

[15] F.B. Bastani, I.R. Chen, and T. Tsao, "Reliability of Systems with Fuzzy-Failure Criterion," *Annual Reliability and Maintainability Symposium*, 1994, pp. 442-448.

[16] E. Borgia, "The Internet of Things vision: Key features, applications and open issues," Computer Communications, vol. 54, 2014, pp. 1-31.

[17] N. Bui, and M. Zorzi, "Health Care Applications: A Solution Based on The Internet of Things," *the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, Barcelona, Spain, Oct. 2011, pp. 1-5.

[18] C. Chen, and S. Helal, "A Device-Centric Approach to a Safer Internet of Things," *the 2011 International Workshop on Networking and Object Memories for the Internet of Things*, Beijing, China, Sep. 2011, pp. 1-6.

[19] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "TRM-IoT: A Trust Management Model Based on Fuzzy Reputation for Internet of Things," *Computer Science and Information Systems,* vol. 8, no. 4, Oct. 2011, pp. 1207-1228.

[20] I.R. Chen and F.B. Bastani, "Effect of Artificial-Intelligence Planning Procedures on System Reliability," *IEEE Trans Reliability*, vol. 40, no. 3, pp. 364-369, 1991.

[21] I.R. Chen, F.B. Bastani, and T.W. Tsao, "On the Reliability of AI Planning Software in Real-Time Applications," *IEEE Trans Knowledge and Data Engineering*, vol. 7, no. 1, pp. 4-13, 1995.

[22] I. R. Chen, F. Bao, M. Chang, and J.H. Cho, "Trust-based intrusion detection in wireless sensor networks," *IEEE International Conference on Communications*, Kyoto, Japan, June 2011, pp. 1-6.

[23] I. R. Chen, F. Bao, and J. Guo, "Trust-based Service Management for Social Internet of Things Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, Nov-Dec 2016, pp. 684-696.

[24] I.R Chen, F. Bao, M.J. Chang, and J.H. Cho, "Dynamic trust management for delay tolerant networks and its application to secure routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, 2014, pp. 1200-1210.

[25] I.R. Chen, J. Guo, F. Bao and J.H. Cho, "Trust Management in Mobile Ad Hoc Networks for Bias Minimization and Application Performance Maximization," *Ad Hoc Networks*, vol. 19, August 2014, pp. 59-74.

[26] I.R. Chen, and J. Guo, "Hierarchical Trust Management of Community of Interest Groups in Mobile Ad Hoc Networks", *Ad hoc Networks,* May vol. 33, 2015, pp. 154-167.

[27] I.R. Chen, J. Guo, and F. Bao, "Trust Management for SOA-based IoT and Its Application to Service Composition", *IEEE Transactions on Services Computing*, vol. 9, no. 3, 2016, pp. 482-495.

[28] I.R. Chen, and J. Guo, "Dynamic Hierarchical Trust Management of Mobile Groups and Its Application to Misbehaving Node Detection", *28th IEEE International Conference on. Advanced2Information Networking and Applications*, Victoria, Canada, May 2014, pp. 1-6.

[29] I.R. Chen, J. Guo, and J.J.P. Tsai, "Trust as a Service for SOA-based Internet of Things," *Services Transactions on Internet of Things,* vol. 1, no. 1, June 2017, pp. 43-52.

[30] I.R. Chen, F. Bao, M. Chang, J.H. Cho, "Trust management for encounter-based routing in delay tolerant networks," *Global Telecommunications Conference*, Miami, USA, 2010, pp. 1-6.

[31] I. R. Chen, J. Guo and F. Bao, "Trust Management for Service Composition in SOA-based Internet of Things Systems," *IEEE 2014 WCNC*, Istanbul, Turkey, April 2014, pp. 3486-3491.

[32] I.R. Chen, J. Guo, F. Bao, and J.H. Cho, "Integrated Social and Quality of Service Trust Management of Mobile Groups in Ad Hoc Networks," *9th IEEE Conference on Information, Communications and Signal Processing (ICICS 2013)*, Tainan, Taiwan, Dec. 2013.

[33] I.R. Chen, B. Gu, S.E. George, and S.T. Cheng, "On failure recoverability of client-server applications in mobile wireless environments," *IEEE Trans. Reliability*, vol. 54, no. 1, pp. 115-122, 2005.

[34] I. R. Chen, T.M. Chen, and C. Lee, "Agent-based forwarding strategies for reducing location management cost in mobile networks," *Mobile Networks and Applications*, vol. 6, no. 2, 2001, pp. 105-115.

[35] I.R. Chen, T.M. Chen, and C. Lee, "Performance evaluation of forwarding strategies for location management in mobile networks," *The Computer Journal*, vol. 41, no. 4, 1998, pp. 243-253.

[36] I.R. Chen, A.P. Speer, and M. Eltoweissy, "Adaptive Fault-Tolerant QoS Control Algorithms for Maximizing System Lifetime of Query-Based Wireless Sensor Networks," *IEEE Trans. on Dependable and Secure Computing*, vol. 8, no. 2, 2011, pp. 161-176.

[37] I.R. Chen and N. Verma, "Simulation study of a class of autonomous host-centric mobility prediction algorithms for wireless cellular and ad hoc networks," *36th annual symposium on Simulation*, 2003, pp. 65-72.

[38] I.R. Chen and D.C. Wang, "Analysis of Replicated Data with Repair Dependency," *The Computer Journal*, vol. 39, no. 9, 1996, pp. 767-779.

[39] I.R. Chen, O. Yilmaz, and I.L. Yen, "Admission control algorithms for revenue optimization with QoS guarantees in mobile wireless networks," *Wireless Personal Communications*, vol. 38, no. 3, 2006, pp. 357-376.

[40] Z. Chen, R. Ling, C.M. Huang and X. Zhu, "A scheme of access service recommendation for the Social Internet of Things," *International Journal of Communication Systems*, Feb. 2015.

[41] S.T. Cheng, C.M. Chen, and I.R. Chen, "Dynamic quota-based admission control with sub-rating in multimedia servers," *Multimedia Systems*, vol. 8, no. 2, 2000, pp. 83-91.

[42] S.T. Cheng, C.M. Chen, and I.R. Chen, "Performance evaluation of an admission control algorithm: dynamic threshold with negotiations," *Performance Evaluation*, vol. 52, no. 1, pp. 1-13, 2003.

[43] J. H. Cho, A. Swami, and I. R. Chen, "Modeling and Analysis of Trust Management for Cognitive Mission-Driven Group Communication Systems in Mobile Ad Hoc Networks," *International Conference on Computational Science and Engineering*, vol. 2, 2009, pp. 641-650.

[44] J.H. Cho, A. Swami and I.R. Chen, "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks," *Network and Computer Applications*, vol. 35, 2012, pp. 1001-1012.

[45] J. H. Cho, A. Swami, and I. R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, 2011, pp. 562-583.

[46] J.H. Cho, et al., "Effect of Intrusion Detection on Reliability of Mission-Oriented Mobile Group Systems in Mobile Ad Hoc Networks," *IEEE Trans. on Reliability*, vol. 59, 2010, pp. 231-241.

[47] J.H. Cho, I.R. Chen, and A.Swami, "Modeling and analysis of trust management for cognitive mission-driven group communication systems in mobile ad hoc networks," *Inter. Conf. on Computational Science and Engineering, 2009,* pp. 641-650.

[48] J.H. Cho, A. Swami, and I.R. Chen, "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1001-1012, May 2012.

[49] J.H. Cho and I.R. Chen, "Performance Analysis of Hierarchical Group Key Management integrated with Adaptive Intrusion Detection in Mobile Ad Hoc Networks," *Performance Evaluation*, vol. 68, no. 1, 2011, pp. 58-75.

[50] J.H. Cho and I.R. Chen, "PROVEST: Provenance-Based Trust Model for Delay Tolerant Networks,"*IEEE Transactions on Dependable and Secure Computing,* vol. 15, no. 1, 2018, pp.151-165.

[51] S. Devarakonda, "Real-time Air Quality Monitoring Through Mobile Sensing in Metropolitan Areas," *UrbComp*, Chicago, Illinois, USA 2013.

[52] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-Based Framework for High

Integrity Sensor Networks," *ACM Transactions on Sensor Networks,* vol. 4, no. 3, pp. 1-37, May 2008.

[53] S.E. George, I.R. Chen, and Y. Jin, "Movement-based checkpointing and logging for recovery in mobile computing systems," *5th ACM International Workshop on Data Engineering for Wireless and Mobile Access,* 2006.

[54] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services," *IEEE Transactions on Services Computing,* vol. 3, no. 3, pp. 223-235, 2010.

[55] B. Gu and I. R. Chen, "Performance analysis of location-aware mobile service proxies for reducing network cost in personal communication systems," *Mobile Networks and Applications,* vol. 10, no. 4, 2005, pp. 453-463

[56] J. Guo and I.R. Chen, "A Classification of Trust Computation Models for Service-Oriented Internet of Things Systems," *12th IEEE International Conference on Services Computing (SCC 2015)*, New York, June 2015.

[57] J. Guo, I.R. Chen, and J.J.P. Tsai "A Survey of Trust Computation Models for Internet of Things Systems," *Computer Communications*, vol. 97, 2017, pp. 1-14.

[58] J. Guo, I.R. Chen, J.J.P. Tsai, and H. Al-Hamadi, "A Hierarchical Cloud Architecture for Integrated Mobility, Service, and Trust Management of Service-Oriented IoT Systems," *6th IEEE International Conference on Innovative Computing Technology*, Dublin, Ireland, 2016.

[59] J. Guo, I.R. Chen, J.J.P. Tsai, and H. Al-Hamadi, "Trust-based IoT Participatory Sensing for Hazard Detection and Response," *1st International Workshop for IOT Systems Provisioning & Management in Cloud Computing (ISYCC 2016)*, Banff, Canada, Oct. 2016.

[60] J. Guo, I.R. Chen, and J.J.P. Tsai, "A Mobile Cloud Hierarchical Trust Management Protocol for IoT Systems," *5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, San Francisco, April 2017.

[61] J. Guo, I.R. Chen, D.C. Wang, J.J.P. Tsai, and H. Al-Hamadi, "A Case Study of IoT Participatory Sensing of Hazardous Ozone using Traces," *The 7th International Conference on Electronics, Communications and Networks*, Hualien, Taiwan, Nov. 2017.

[62] Z. Huang, D. Zeng, and H. Chen, "A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce," *IEEE Intelligent Systems,* vol. 22, pp. 68-78, 2007.

[63] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An Internet of Things-Based Personal Device for Diabetes Therapy Management in Ambient Assisted Living (AAL)," *Personal and Ubiquitous Computing,* vol. 15, no. 4, 2011, pp. 431-440.

[64] A. Jøsang, "A logic for uncertain probabilities", I*nternational Journal of Uncertainty, Fuzziness and Knowledge-Based System*s, vol. 9, no. 3, June 2001, pp. 279– 311.

[65] A. Jøsang, and R. Ismail, "The Beta Reputation System," *Bled Electronic Commerce Conference*, Bled, Slovenia, 2002, pp. 1-14.

[66] A. Josang, R. Ismail, and C. Boyd, "A Survey of Trust and Reputation Systems for Online Service Provision," *Decision Support Systems*, vol. 43, no. 2, March 2007, pp. 618-644.

[67] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," *12th International Conference on World Wide Web*, Budapest, Hungary, May 2003.

[68] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović, "Power Law and Exponential Decay of Intercontact Times between Mobile Devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, 2007, pp. 1377-1390.

[69] A.U.R. Khan et al., "A Survey of Mobile Cloud Computing Application Models," *IEEE Communi-*

cations Surveys & Tutorials, Vol. 16, No. 1, Nov. 2014, pp. 393-413.

[70] W.Z. Khan, Y. Xiang, M.Y. Aalsalem, and Q. Arshad, "Mobile Phone Sensing Systems: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 402–427, 2013

[71] S. Kosta, A. Mei, and J. Stefa, "Small World in Motion (SWIM): Modeling Communities in Ad-Hoc Mobile Networking," *7th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, Boston, MA, USA, 2010.

[72] X. Li, F. Zhou, and J. Du, "LDTS: A Lightweight and Dependable Trust System for Clustered Wireless Sensor Networks," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, 2013, pp. 924-935.

[73] Y. Li, and I.R. Chen, "Design and performance analysis of mobility management schemes based on pointer forwarding for wireless mesh networks," *IEEE Trans Mobile Computing*, vol. 10, no. 3, 2011, pp. 349-361.

[74] Y. Liu, Z. Chen, F. Xia, X. Lv, and F. Bu. "A trust model based on service classification in mobile services", *IEEE/ACM international conference on cyber, physical and social computing*, 2010, pp. 572–577.

[75] Z. Malik and A. Bouguettaya,"RATEWeb: Reputation Assessment for Trust Establishment among Web Services," *The VLDB Journal*, vol. 18, 2009, pp. 885-911.

[76] P. Massa, and P. Avesani, "Trust-aware Recommender Systems," *ACM Recommender Systems Conference, Minneapolis, Minnesota*, USA, 2007.

[77] D. A. Menasce, "QoS Issues in Web Services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72-75, 2002.

[78] R. Mitchell and I.R. Chen, "Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems," *IEEE Transactions on Reliability*, vol. 62, no. 1, March 2013, pp. 199-210.

[79] R. Mitchell and I. R. Chen, "Behavior-Rule Based Intrusion Detection Systems for Safety Critical Smart Grid Applications," *IEEE Trans. Smart Grid*, vol. 4, no. 3, pp. 1254-1263, 2013.

[80] R. Mitchell and I. R. Chen, "Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles using Behavior Rule Specifications," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 44, no. 5, 2014, pp. 594-604.

[81] R. Mitchell and I.R. Chen, "A survey of intrusion detection in wireless network applications," *Computer Communications*, vol. 42, 2014, pp. 1-23.

[82] R. Mitchell and I.R. Chen, "A Survey of Intrusion Detection Techniques in Cyber Physical Systems," *ACM Computing Survey*, vol. 46, no. 4, 2014.

[83] R. Mitchell and I.R. Chen, "Modeling and Analysis of Attacks and Counter Defense Mechanisms for Cyber Physical Systems," *IEEE Trans. on Reliability*, 2015.

[84] R. Mitchell and I.R. Chen, "Behavior Rule Specification-based Detection for Safety Critical Medical Cyber Physical Systems," *IEEE Trans Dependable and Secure Computing*, vol. 12, no. 1, pp. 16-30, 2015.

[85] H. Mousa, et al. "Trust management and reputation systems in mobile participatory sensing applications: A survey" *Computer Networks*, vol. 90, 2015, pp. 49-73.

[86] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness Management in the Social Internet of Things," *IEEE Transactions on Knowledge and Data Management*, vol. 26, no. 5, 2014, pp. 1253-1266.

[87] M. Nitti, L. Atzori, and I.P. Cvijikj, "Friendship Selection in the Social Internet of Things: Challenges and Possible Strategies," *IEEE Internet of Things Journal*, vol. 2, no. 3, 2015, pp. 240-247.

[88] B. Pires, et al, "Towards an in silico Experimental Platform for Air Quality: Houston, TX as a Case Study," *Computational Social Science Society of America Conference* (2015), Santa Fe, NM, USA.

[89] M. Presser, and S. Krco, *The Internet of Things Initiative D2.1: Initial report on IoT applications of*

strategic interest, 2011.

[90] M.R. Rahimi1, N. Venkatasubramanian1, S. Mehrotra1, and A.V. Vasilakos, "MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture," *IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, 2012, pp. 83-90.

[91] Y.B. Saied, A. Olivereau, D. Zeghlache and M. Laurent, "Trust Management System Design for the Internet of Things: A Context-aware and Multi-service Approach," *Computers and Security*, vol. 39, part B, Nov. 2013, pp. 351–365.

[92] M. Satyanarayanan, et al., "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, Oct. 2013, pp. 40-49.

[93] W. Sherchan, S. Nepal, and C. Paris, "A Survey of Trust in Social Networks," ACM Computing Survey, Vol. 45, No. 4, Article 47, August 2013.

[94] Z. Su, L. Liu, M. Li, X. Fan, and Y. Zhou, "ServiceTrust: Trust Management in Service Provision Networks," *IEEE International Conference on Services Computing*, Santa Clara, 2013, pp. 272-279..

[95] R.R.S. Verma, D. O'Mahony, and H. Tewari, "NTM - Progressive Trust Negotiation in Ad Hoc Networks," *IEI/IEE Symposium on Telecommunications Systems Research*, Dublin, Ireland, Nov. 2001, pp. 1-8

[96] Y. Wang, I.R. Chen, J.H. Cho, K.S. Chan and A. Swami, "Trust-based Service Composition and Binding for Tactical Networks with Multiple Objectives," *32th IEEE Military Communications Conference*, San Diego, CA, Nov. 2013.

[97] Y. Wang, I.R. Chen, J.H. Cho, A. Swami, and K.S. Chan, "Trust-based Service Composition and Binding with Multiple Objective Optimization in Service-Oriented Ad Hoc Networks," *IEEE Transactions on Services Computing*, vol. 10, no. 4, 2017, pp. 660-672.

[98] Y. Wang, Y.C. Lu, I.R. Chen, J.H. Cho, and A. Swami, "LogitTrust: A Logit Regression-based Trust Model for Mobile Ad Hoc Networks," 6th *ASE International Conference on Privacy, Security, Risk and Trust*, Boston, MA, Dec. 2014.

[99] Y. Wang, I.R. Chen, and D.C. Wang, "A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges," *Wireless Personal Communications*, vol. 80, no. 4, 2015, pp. 1607-1623.

[100] Y. Wang, et al., "CATrust: Context-Aware Trust Management for Service-Oriented Ad Hoc Networks," *IEEE Transactions on Services Computing*, 2018.

[101] Y. Wang, I.R. Chen, J.H. Cho, and J.J.P. Tsai, "Trust-Based Task Assignment with Multi-Objective Optimization in Service-Oriented Ad Hoc Networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, March 2017, pp. 217-232.

[102] L. Xiong and L. Liu, "PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, 2004, pp. 843–857.

[103] Z. Yan, P. Zhang, and A.V. Vasilakos, "A Survey on Trust Management for Internet of Things," *Journal of Network and Computer Applications*, vol. 42, pp. 120-134, 2014.

[104] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, 2014, pp. 1-10.

[105] O. Yilmaz and I.R. Chen, "Utilizing Call Admission Control for Pricing Optimization of Multiple Service Classes in Wireless Cellular Networks," *Computer Communications*, vol. 32, no. 2, pp. 317-323, 2009.

[106] B. Yu and M.P. Singh, "An evidential model of distributed reputation management", *1st ACM Int. Joint Conference on Autonomous Agents and Multiagent Systems*, July 2002.

[107] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of Real-World Web Services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32-39, 2014.

[108] ns-3 Simulator, http://www.nsnam.org.
[109] ns-3 Tutorial, http://www.cpe.ku.ac.th/~anan/myhomepage/wp-content/uploads/2012/01/ns3-part1-introduction.pdf.