6-1-2008

# Architecture and Implementation of a Trust Model for Pervasive Applications

Sheikh Iqbal Ahamed
*Marquette University*, sheikh.ahamed@marquette.edu

Mohammad Zulkernine
*Queen's University - Kingston, Ontario*

Sailaja Bulusu
*Marquette University*

Mehrab Monjur
*Marquette University*

Published version. *Journal of Mobile Multimedia,* Vol. 4, No. 2 (June 1, 2008): 84-103. Permalink. 

Journal of Mobile Multimedia, Vol. 4, No.2 (2008) 084-103


# ARCHITECTURE AND IMPLEMENTATION OF A TRUST MODEL FOR PERVASIVE APPLICATIONS

SHEIKH I. AHAMED

*Marquette University, Milwaukee, WI -53201, USA*
*iq@mscs.mu.edu*

MOHAMMAD ZULKERNINE

*Queen's University, Kingstone, Canada*
*mzulker@cs.queensu.ca*

SAILAJA BULUSU MEHRAB MONJUR

*Marquette University, Milwaukee, WI -53201, USA*
*{sbulusu, mmonjur}@mscs.mu.edu*

Received February 16, 2008
Revised April 24, 2008

Collaborative effort to share resources is a significant feature of pervasive computing environments. To achieve secure service discovery and sharing, and to distinguish between malevolent and benevolent entities, trust models must be defined. It is critical to estimate a device's initial trust value because of the transient nature of pervasive smart space; however, most of the prior research work on trust models for pervasive applications used the notion of constant initial trust assignment. In this paper, we design and implement a trust model called DIRT. We categorize services in different security levels and depending on the service requester's context information, we calculate the initial trust value. Our trust value is assigned for each device and for each service. Our overall trust estimation for a service depends on the recommendations of the neighbouring devices, inference from other service-trust values for that device, and direct trust experience. We provide an extensive survey of related work, and we demonstrate the distinguishing features of our proposed model with respect to the existing models. We implement a healthcare-monitoring application and a location-based service prototype over DIRT. We also provide a performance analysis of the model with respect to some of its important characteristics tested in various scenarios.

*Key words*: Pervasive computing, trust model, security level

## 1 Introduction

Heterogeneous, portable, lightweight devices like PDAs, cell phones, and smart phones communicate in a pervasive smart space. Collaborative efforts to share resources help these devices overcome

limitations such as battery power, memory, and computational capability [1, 19]. In order to optimize these sharing possibilities and to achieve pervasive environment-wide goals, we need to limit intrusions by malevolent entities. With increasingly transient and mobile environment where devices operate independently, the notion of a central authentication mechanism goes against the true nature of a pervasive environment. Distributed trust mechanisms can help distinguish which entities are trustworthy by sharing interaction experiences among devices.

Trust in pervasive computing environments has been defined by many researchers [37, 18, 8] from a number of different perspectives. Gambetta [8] defines trust as the probability of doing something benevolent by a trusted party A for the trustee party B. Almenarez et al. [18] describe trust as "the belief that an entity has about other entity, from past experiences, knowledge about the entity's nature and/or recommendations from trusted entities. This belief expresses an expectation on the entity's behaviour, which implies a risk" [12]. The level of trust value is normally used in order to make a risk-taking decision as stated by Mayer et al. [37]: "If the level of trust surpasses the threshold of perceived risk, then the trustor will engage in the risk-taking."

Most trust research focuses on entities and takes into account overall interactions across entities but disregards the nature of interactions, i.e., the actual service or content exchanged. This is important, since each service has associated security levels [23], and for each security level, the service provider will have varied opinions regarding a requester's reputation. In this paper, we look beyond actor-interactions and consider service-transactions for trust evaluation.

In this paper, we design and implement DIRT, a distributed trust-model and an extension of SSRD [23]. It has been named after the word 'dirt', which means private or personal information which if made public would create a scandal or ruin the reputation of a person, company, etc. [35]. Trust evaluation in DIRT requires initial trust estimation, trust-inference from other service-trust values for that device, recommendations from neighboring devices, and direct trust experience. A service provider (SP) assigns a trust value to a service requester (SR) for each category of services. We map services to different security levels and the trust estimation varies for each level of security. The initial trust value is critical for trust models to work effectively [34]. Most of the previous works [40], including our own work [23, 15, 26] used fixed trust values representing a service provider's initial disposition to trust, while some works tried to infer initial trust values from similar and known contexts [17]. For initial trust estimation, an SR needs to provide context data and must answer the relevant questions. Recommended trust among entities depends on a simple rule: take minimum recommended trust from the most valued recommender. Direct trust value depends on an SP's satisfaction level for an SR. Overall trust also infers ratings from other services that SR has accessed previously. We provide a performance analysis of our model evaluated with the help of OmNet++ [32] for various scenarios. For performance evaluation, we model a healthcare-monitoring application and a location-based service prototype.

We organize our paper as follows. Section 2 discusses the desired characteristics of a trust model that we have tried to achieve in our proposed model. Section 3 describes related research on trust, and compares and contrasts our proposed model with various other models. Section 4 shows the architecture of DIRT, and Section 5 presents the detail operation of DIRT. Section 6 presents an illustrative example to explain the operation of DIRT. In Section 7, we present simulation results with

various scenarios using OmNet++ and a prototype implementation. We conclude with Section 8 by providing some future directions.

## 2 Characteristics of the Trust Model

Most pervasive computing environments depend on sharing of services. For that, SRs make requests to SP for a service. Based on the trustworthiness of the requested device, the SP accepts or rejects. In this section, we discuss some scenarios and the desired characteristics of a trust model required for effective sharing of services.

Scenario1: John is a business analyst and commuter on a subway train to his office located in New York. Generally, his travel time to reach his office takes 1hr 30min. During his commute, John received a phone call indicating he has to attend a meeting in the office. For the meeting, he has to have some data ready. Since required information is in his email, he wants to check his email immediately. However, John does not have any internet service with his cell phone, and he wants to request service from any nearby devices.

Scenario 2: Andy visited the national historic museum of New York. The museum is so big that he will not be able to cover the whole museum within the time he has. Therefore, he would like to know about the important places to visit in the museum and some information about the performances that take place from time to time. He wants to get this information by chatting with someone who has already visited most of the museum.

Scenario 3: Dr. Jones, a resident doctor in a hospital, has fourteen patients in his care. He needs to pass detailed information about all of his patients to his on-call colleague Dr. Smith. Dr. Jones uses his hand-held device to write all the patient information, and with a healthcare application, Dr. Jones can simply transmit his patients' data from his handheld device to Dr. Smith who is working somewhere else in the medical center. Dr. Jones also makes his observations and comments about the patients available to the associated doctors instantly and effortlessly with the use of healthcare application assuming that other doctor's devices are trusted.

From the above scenarios, it is clear that the sharing of services in a pervasive network requires the device to be trusted. Moreover, the trust values are also not the same for all services. For example, the requesting and sharing of an internet service in Scenario 1 has a trust value and risk that is more than the chat application in Scenario 2. In Scenario 3, the doctor's device has to be trusted in order to share the information. As indicated in the scenarios, we encompass the following characteristics that are desired from a trust model used in a pervasive environment.

Dynamic

Trust value is dynamic in the sense that it has different values depending on the service requested and the past behaviour of the requester. For each service, service provider needs past relevant context information to make an inference about the present reputation or trust value.

Paula's note – you talk frequently about context information but up to this point in the paper it's unclear to me what you mean. At this point in reading, I skipped to Context Awareness and that helped. Perhaps you could move Context Awareness up and list under Dynamic. Or perhaps Context Information needs its own explanation.

Risk Awareness

To have more security in a pervasive environment, trust and risk always come hand in hand. Since the value of trust is not the same for all services, in service-oriented applications there is some risk involved when sharing services from one device to the other based on the calculated trust.

Adaptive Recommendation

As the service provider cannot have the trust values for all the services, it needs to depend on other devices in the network during the calculation of a trust value for the service requester. The trust values may increase or decrease with further experience.

Reliable Initial Trust

Whenever a new device joins the network, we use the user of the device's context information Depending on the service it will use, we calculate the initial trust value for the device. This value evolves over time as we get more acquainted, and we get more recommendations from other devices in the network.

Resource Efficient

Devices in a pervasive smart space like PDAs, cell phones, smart phones etc. have very low battery power and memory. Our model uses simple equations and an easy communication model to build trust and propagate it.

Context Awareness

A trust value mainly depends on only one context, i.e., service request. For initial value calculation, we consider context values like location, age, time, number of times service requested, etc.

Service-Oriented

A trust value is not the same among all devices for the different services categories. Each service requires different trust levels depending on security levels.

## 3 Related Work

In this section, we analyze each of the trust models with respect to the characteristics described in Section 2. Table I shows the comparison table for the related models. From the table, it is clear how our model is different from the other models.

Haque et al. [15] introduce a mechanism for handling multi-hop recommendations and at the same time, provide a unique way when considering contexts as vectors for trust calculation. Quercia et al. [7] show how mobile devices store a very limited subset of the intricacies of trust and modelled graph-based semi-supervised learning scheme. Here, trust is not dependent on context and any node can make public the ratings of all other nodes . In [34], they model a novel mechanism, named TRULLO (TRUst bootstrapping by Latently Lifting cOntext). TRULLO statistically analyzes the ratings of its users' past experiences to bootstrap unknown trust values and it relies only on local information. Therefore, TRULLO does not need to collect recommendations.

Sharmin et al. in their service discovery protocol SSRD [23] present a resource discovery and trust-based model which is developed to ensure security in a pervasive computing environment. The

trust model is a service specific model rather than device specific, which means that the same trust value is not used for all services. The trust value depends on the specific requester, the service requested, and the security level of the service. The model also has a prototype implementation for calculating the trust values. Based on these trust values, the service provider can decide whether the service is offered. While the model considers average trust value (which is given as an initial trust value for the newly joined device), no specific method is provided to achieve the reliable initial trust value. The SSRD+ [28] (extension of SSRD) focuses on the dynamic trust relationships of the devices while not having any performance degradation. They include trust and risk models for sharing services. The idea to include the risk model is to protect the network from malicious devices. Even though they include the risk model for an unknown device, they do not address how the reliable initial trust value is achieved as well as how the trust value changes with the context. They evaluate this model by implementing the prototype and using simulation tool OMNeT++ for the performance of the model.

Novel cloud model [36] focuses on the uncertainty between entities, referring to the trust between the entities as cloud. Further, they propose a cloud-based trust model which helps to put the model into real time and make the trust decisions in security-based mechanisms in a pervasive computing environment. They did not address how the trust decision is changed along time, nor is location addressed in this model. They did not mention the risk involved when a new device joins the network. Their focus is not towards having a reliable initial trust value and using the resources efficiently.

The PTM model [12] is a decentralized trust model based on a public key used in the infrastructure-less environment. Their main focus is on the dynamic behavior of the entities and the exchange of trust information by using recommendation protocol. Further, they did not address any risk and context in forming the model. However, they validated the model using the security level and variable parameter with respect to the trust values with the help of the behavior of the entity. The main focus of this model is to reduce the human intervention in calculating the trust. Yet they did not have the concept of initial trust value. Further they addressed risk assessment in a model [14]. However, this model is not service-oriented since all of the services for a specific device have the same trust value.

The service discovery protocol [13] is based on PTM [12]. They incorporate the trust in this protocol to get the trustworthy services from other devices. This protocol doesn't require any centralized server., The need for a server is eliminated by the use of components, user agents and a service agent. In addition, they address how the trust value is changed with the positive and negative experiences within the specific context. One of the pitfalls of this protocol is that the user agent of the device keeps on listening for the broadcasting messages and this consumes resources. The risk analysis is not used. Instead a security model is incorporated to work with the security levels of the services and to achieve security issues like mutual authentication, data integrity and confidentiality based on the level of trust.

English et al. [5] present a dynamic model, which supports recommendation trust. This model captures the dynamic aspects of the trust formation and evolution, and it considers human decisions in order to reduce the security vulnerabilities. The dynamic aspects of this model include trust formation, evolution, and exploitation. As a part of this project, the model being developed addresses the characteristics like context awareness and risk awareness.

In autonomic trust [21], the authors address the following issues: the autonomic ability of the model, the burden imposed on the user's device when the model is used, and the usage of large amount of user input and physical resources in computing the trust. This model overcomes the above mentioned issues by including an accurate and efficient trust predictor which is based on a basic Kalman filter. With the Kalman filter, they derive an autonomic, lightweight and thus the trust predictor in pervasive computing scenarios. In addition to the derivation of the trust predictor, the simulation results are also given to evaluate the performance of the trust predictor's behavior.

Social trust [3] is based on the social certificate for trust negotiation by using the secure trust negotiation agent (STNA). This model also uses the trust negotiation protocol in order to establish a trust relationship between the entities. To overcome the disadvantages of recommendation trust, this model used a property-based trust. In the property-based model, during the process, the two entities which are trying to establish the trust relationship need to agree about the activity on which they want to build the trust relation. Then the trust negotiation protocol is used to build the trust relationship. This model is basically developed for peer to peer environments that can be adapted to the pervasive computing environments.

| Model | Dynamic | Risk Awareness | Adaptive Recommendation | Reliable Initial trust | Resource efficient | Context Awareness | Service-Oriented |
|---|---|---|---|---|---|---|---|
| PTM[12] | Y | NA | Y | N | Y | NA | NA |
| SSRD[23] | Y | N | Y | N | Y | N | Y |
| SSRD+[37] | Y | Y | Y | N | Y | N | Y |
| SPDP[13] | Y | N | N | NA | Y | NA | Y |
| NCM[36] | Y | N | NA | NA | N | N | N |
| B-trust[6] | Y | P | Y | N | Y | NA | N |
| ST [3] | Y | NA | Y | NA | N | NA | NA |
| AT[21] | Y | NA | NA | P | Y | NA | NA |
| DTM [5] | NA | P | Y | N | NA | Y | N |
| TF [42] | Y | N | Y | N | NA | NA | N |
| TAG [39] | Y | NA | Y | N | NA | Y | NA |
| RTM [4] | NA | Y | N | N | NA | NA | NA |
| DIRT | Y | NA | Y | Y | Y | NA | Y |

Table I Comparison Table (Y – Yes, N – No, NA – Not Addressed, P – Partial)

The trust framework [42] presents a model which can be used in both 'purely adhoc environments' and 'managed adhoc environments'. The focus of this model is to categorize the device based on the security functionalities and the resources available for that device. The quantification of trust is done by using a recommendation value in this model. This model also has a scenario to show the calculation

of trust values for number of malicious devices and other specific devices such as devices with low battery power, etc.

The risk-based trust model [4] is an approach to integrate security and risk management with the help of a trust model, which in turn gains utility maximization. The objective of this model is not only to eliminate malicious devices but also to allocate permissions for the legitimate devices based on the risk levels. This model has developed a qualitative approach for integrating risk and security by using trust as a mediator. This trust model is invoked before the authorization module. If the value of the trust is lower than the threshold value, then it will be evaluated by the authorization module which in turn results in utility maximization. There is no approach given for the calculation of risk values and initial trust values. They use a mobile agent system as an example to study the approach.

The trust analysis grid [39] presents a trust analysis methodology and guidelines to design a pervasive computing system. Various trust issue categories from different aspects of the pervasive computing scenarios are listed in the form of a grid. The trust issue is categorized into the following groups: 1) Data categories – this group describes the property of the data. 2) System categories – this group describes the components and services of the system that can be used in the pervasive scenario. 3) Subjective categories – this group includes the categories which are related to the internal state and knowledge of the device.

They validate and analyze the scenarios. Then they peer review and refine the scenarios. Finally, they formalize the guidelines. This approach does not address any risk analysis issues; rather it does consider some context aware properties. They do the validation with the help of agent-based systems by which they achieve the dynamic property of the trust.

B–trust [6] is a distributed trust evaluation mechanism based on direct experience and recommendations. The whole process is dependent on the Bayesian formalization. Further, this model is robust to withstand the common attacks and thus supports the user anonymity. It is lightweight and does not include context awareness or the idea of calculating the initial trust for newly joined devices. They integrate risk awareness, while there is no mechanism for calculating the risk. Finally, this model shows positive results on packet delivery when some of the devices act as malicious devices.

## 4 DIRT Architecture

DIRT has a component-based architecture (See Figure 1 and 2). It has the following main components: Service Manager (SM), Direct Trust Builder (DTB), Recommendation Trust Builder (RTB), Recommended Value Accumulator (RVA), Initial Trust Calculator (ITC), and Update Manager (UM).

Service Manager (SM): The Service Manager of each device contains a list of trust values of all the neighboring devices for each service. Before that, SM maps each service to relevant security level [23] (See Table II). It is represented as SM (device_id, service_name, trust_value, security_level).

Direct Trust Builder (DTB): Upon experiencing a transaction with a requester, a SP (Service Provider) assings its satisfaction level. In order to share the service with the SR (Service Receiver), SP checks its trust list that is maintained by the SM and ITC. It uses both present and past direct trust values in calculating a new trust value.

Recommendation Trust Builder (RTB): This component is invoked when an SP does not have any previous interaction with an SR. In building the trust value with the SR (a new device in this case), as an initial step, SP must assign some initial trust value for SR which is done by ITC. In the next step, the RVA requests recommendation values from the neighboring devices and accumulates trust value for each service.

Recommended Value Accumulator (RVA): The RV Accumulator of a device gets all the recommendation values from the neighboring devices in the network. These values are used in calculating the direct trust in case of a known device and the recommended trust in case of a new device. Furthermore, RVA is responsible for the process of how the recommendation values are transferred from one device to the other.

Initial Trust Calculator (ITC): The ITC component of our model is invoked when a new device joins the network. This component assigns a trust value for a newly joined device in order to allow it communicate with other devices in the neighborhood and thus share services.

Update Manager (UM): The task of the UM is to calculate the total trust value for each service (Equation 5) and also total trust value without considering services (Equation 6). It updates the trust values in the SM whenever an interaction occurs. Once the trust values are calculated (either direct or recommended), it updates the trust/service list which is represented as UM (device_id, service_name, trust_value).

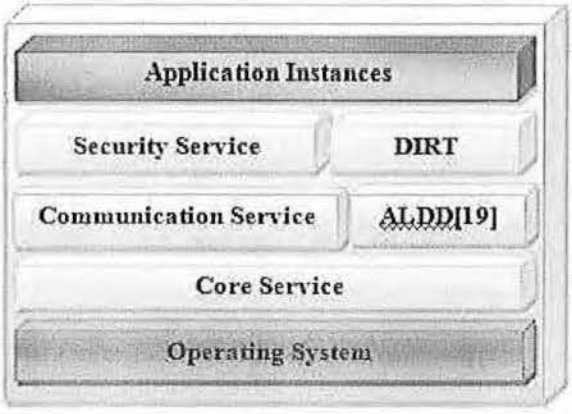



*Figure 1: Architecture of MARKS [21] with DIRT*

The components of the architecture are shown in Figure 2. When an SR (in case it is a new device and does not have any previous interactions with an SP) requests a service, the SM of the SP is invoked. Then RTB is called, followed by ITC and RVA in order to calculate the initial trust value and to accumulate the recommendations from the neighboring devices of the network respectively. Once the trust value is calculated, the UM updates the trust values.

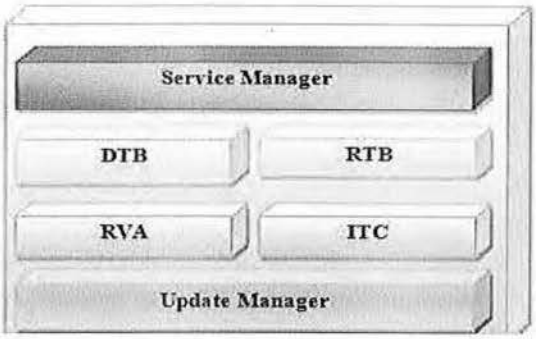



*Figure 2: Components of DIRT*

## 5 DIRT Operation

DIRT uses ITC to build initial trust, RTB to evaluate recommended trust, DTB to estimate direct trust experience, and UM to calculate overall trust with and without considering service. DIRT uses SM to categorize services to different security levels.

To calculate initial trust value, SM categorizes each service into different security-levels. Each security level has some requirements that SR needs to fulfill to access the service in that category. Requirements include relevant context information and four questions-answers. When a new device joins the network without any previous service request, initial trust value is calculated using Equation 1a. Again, if SR's service request is new but previously had low security-level service interaction, the initial trust value depends on previous average initial trust as well as meeting other requirements, according to Equation 1b.

RVA acquires all the recommended value from neighbouring devices. RTB makes recommended trust assignment using the statement: take minimum recommended value from most valued recommender. This is similar in nature to Advogato [38] calculation of recommended trust where "between any two nodes, the quantity of trust flowing from one node to another cannot be greater than the weakest rating somewhere on the path between the two nodes". Again, if there is no recommendation, RTB uses only initial trust value from ITC.

DTB calculates direct trust value for each service using SP's satisfaction level of SR and also takes into account old direct trust ratings.

UM calculates total trust value for each service considering three things, a) influence of total trust for other services, b) recommended trust value and c) direct trust experience. Finally, update manager also calculates overall trust without considering service specificity which serves as a recommendation for other devices.

### *5.1 ITC – New Device Enters Into the Environment*

For each service, an SR requires a trust value to access and use the service. Initial Trust Calculator (ITC) comes into play when there is no trust value for a SR for a service. Indeed, there may be cases when SR had other service experience with SP but not for this particular service for which ITC is invoked. We use the table of [23] and extended it (see Table II) to categorize services according to

their security level. SR needs to have a trust value more than 0.5 per security level to access services in that category. We use two parts to calculate initial trust: 1) Initial Trust Calculation and 2) Initial Requirement. Initial requirement includes question-answer requirements and context requirements.

*PART 1: Initial Trust Calculation*

Before providing a detail description of ITC algorithm, we define the following terms:

$T_{s_n}(SP, SR)$ = Trust value of SP in SR for accessing security level $n$ services (See Equation 7).

$T_{avg_n}$ = Average weighted-trust value for security-levels less than n.

*TS[1, 2, ..., 10]* = An array having all the trust value for the 10 security levels

$= \{ T_{s_1}, T_{s_2}, \ldots T_{s_{10}} \}$

We calculate $T_{avg_n}$ with the following equation:

$T_{avg_n} = \sum_{i=1}^{n-1}(i * T_{s_i}) / \sum_{i=1}^{n-1} i.$

Here is the algorithm where SR requires security n-level trust:

STEP 1: If $n$ is in *{1, 2, 3, 4}* and satisfies PART 2, $T_{s_n} = 0.5$. (1a)

STEP 2: If $n >= 5$ and $T_{avg_n} >= 0.5$ and satisfies PART 2, then $T_{s_n} = 0.5$ (1b)

What STEP2 does is permit accessing higher level services. If SR does not have any required average trust value for a lower level service, SR cannot ask for a higher level service and access it right away. SR can access services of level 1,2,3 and 4 by fulfilling STEP1 and with that, SR will have direct trust value. If SR performs okay in these levels, he has now chance to fulfill STEP2 and access higher level services.

*PART 2: Initial Requirement*

In this part, we present some requirements for SR to satisfy. As we observed in the previous section that PART 2 requirements are pre-requisites for an initial trust calculation, we have two requirements: 1) give answers to prepared questions and 2) give relevant context data.

STEP 1. Context Requirement:

Service Manager (SM) categorizes each service and maps it into the security level table. We used the table from [23] and we added some context requirement that SR needs to fulfill to have ITC work. Each device will define his satisfactory context domain. Hence, SR first needs to pass the context requirement of the pre-defined context domain. For example, to use a chat software, SR may need to be available in the smart space at least 5 times in a week. Again, if SR asks for a MIME type which SP does not provide or find it critical, SR will not pass this phase.

STEP 2. Question Requirement:

For accessing services in security levels 1 to 4, we emphasize the answering of four questions. Depending on the answers, we provide a value using function *rV()* from 0.1 to 0.5. (0.1 means dependency is minimum and 0.5 means dependency is maximum).

The questions are as follows:

Q1. How frequently is the service used by the device?

Q2. Where does the device need the service? For example, the services which can be used by students frequently in a school environment have lower values than other services.

Q3. How many times is the service requested by the requester?

Q4. How many devices can provide the recommendations for that newly joined device?

The response value to each answer from the SR is used to evaluate (rV). Now, the initial trust value of the SR for service *i* can be calculated as follows: $Ti(SR) = ( rV(Q1)+rV(Q2)+ rV(Q3) + rV(Q4))/4$.

| Service Name | Security Level (1-10) | Context Requirement |
|---|---|---|
| Date/Time | 1 | n/a |
| Weather | 1 | n/a |
| WordPad, PDA Viewer | 2 | n/a |
| Chat software | 4 | Location, Frequency of visit in the smart space |
| ... ... ... | ... | ... ... ... |
| Location Service | 7 | Location |
| Healthcare Service | 8 | Age, Location |
| Internet Service | 9 | Location, Round Trip Time, MIME |
| Address book | 10 | Location, Frequency of visit in the smart space. |

Table II List of services with associated security level and context requirement

### *5.2 RVA - Transferring recommendation values*

The idea of having an Recommended Value Accumulator (RVA) component is to get the recommendation values from all neighboring devices. If a new device requests a service from another device and SP does not have any previous interaction with SR, then SP requests recommendations from the neighboring devices. The message format for requesting the recommendations from the neighboring devices is: Reco_request (SP, SR_id, service_name, WT) SR_id is a service requester id, and WT is the waiting time that SP can wait for the recommendation. Once the request is received by the neighboring device, it responds with the recommendation value within WT time. If SP does not receive the recommendation within WT, it means that the neighboring device does not have the recommendation value for the SR. The message format for the response is: Reco_response (SP, SR_id, service_name, reco_value, WT).

### *5.3 RTB – Recommended Trust Calculation*

Recommendation Trust Builder (RTB) uses RVA to gather recommendation values and uses the initial trust value from ITC. We have two part to the calculation, first, we calculate the recommended value for a service i and second, we calculate recommended trust.

STEP 1. Recommended Value Calculation:

We calculate the recommended value for SR in SP for service i in three steps:

Find the set r of devices where for each device $r_i$ in $r$ the trust value in device SP is maximum.

$r \in Max(T(SP, x))$, where $x$ is the set of all the devices in the environment.

Find the minimum trust value of SR for service $i$ in the set of device $r$.

$$RV = Min\ (T(r, SR)) \tag{2}$$

$T\ (A, B)$ = Trust value of device B in device A for service.

STEP 2, Recommended Trust Calculation:

RTB consults Service Manager (SM) and learns the security-level of service $i$. If service $i$ has security level 1, then recommended trust value for service I is:

$$T_r(SP, SR, i) = \begin{cases} (T_{s_l} + RV)/2, where\ RV \neq \emptyset \\ T_{s_l}, otherwise. \end{cases} \tag{3}$$

$T_r(A, B, i)$ = recommended trust value of device B in device A for service $i$.

### *5.4 DTB – Direct Trust Calculation*

Direct Trust Builder (DTB) assigns SP's satisfaction level with SR as it interacts with SP. This direct trust level depends on a current satisfaction level and past direct experience. Direct trust is important since it does not depend on the recommendations of other trusted neighbors. The work in [15] uses a similar way to calculate direct trust.

$$T_d(SP, SR, i) = \begin{cases} \dfrac{T_{d_{old}}(SP,\ SR, i) + \tau}{2}, if\ there\ is\ previous\ experience \\ \tau \qquad\qquad\qquad\qquad\qquad\quad , otherwise \end{cases} \tag{4}$$

$T_d(SP, SR, i)$ = direct trust value of SR in SP for service i.

$T_{d_{old}}(SP, SR, i)$ = old direct trust value of SR in SP for service i.

$\tau$ = SP's satisfaction level of SR from direct experience.

### *5.5 Update Manager – Total Trust Calculation*

Update Manager (UM) calculates total trust for each service request by aggregating trust values from DTB and RTB. The total actual trust for a service makes use of trust-inference from other service-trust values for that device, recommendations from neighboring devices, and direct trust experience.

$$T(SP, SR, i) = \frac{\omega_1 \times \dfrac{\sum_{j=1}^{m} T(SP, SR, j)}{m} + \omega_2 \times T_r(SP, SR, i) + \omega_3 \times T_d(SP, SR, i)}{\omega_1 + \omega_2 + \omega_3} \tag{5}$$

$T(A, B, i)$ = total trust value of Device $B$ in device $A$ for service i.

$\omega_1$ = weight for trust inference from other accessed services.

$\omega_2$ = weight for recommended trust.

$\omega_3$ = weight for direct trust.

Update Manager also calculates the overall trust for SR in device SP without considering any service. This value is useful because it will serve as the recommendation for other devices regarding SR.

$$T(SP,SR) = \frac{\sum_{j=1}^{n} T(SP,SR,i)}{n} \quad (6)$$

Update Manager calculates SP's trust level of SR for security n level services. ITC also use this calculation while putting initial trust value to higher security levels.

$$T_{s_n}(SP,SR) = \frac{\sum_{i \in n} T(SP,SR,i)}{p} \quad (7)$$

$T_{s_n}(SP,SR)$ = Trust value of SP in SR for accessing security level $n$ services.

$p$ = number of services in security level $n$.

## 6 Illustrative Example

In this section, we present how our model works in one scenario described in Section 2. For Scenario 3, in which Dr. Jones is requesting for a healthcare service. let us suppose that there are other people, say A, B, and C, in the network and Dr. Jones is requesting a healthcare service of security-level 8 (See Table II). A has that service and is able to provide that service to Dr. Jones. However, as device A does not have any previous interaction with Dr. Jones, it needs to calculate an initial trust value as well as recommended trust value. It takes the recommendations from B and C, which are 0.5 and 0.7 respectively, as well as A's trust value for B and C which are 0.7 and 0.8, respectively. Because C is the most valued recommender of A, the recommendation value for Dr. Jones is 0.7 (Equation 2).

Device A needs to calculate an initial trust value for Dr. Jones when he is asking for security-level 9 service. As Dr. Jones has no previous average-weighted-initial-trust value for lower level security, he will have no initial security-level 9 trust assignment. Hence, $T_{s_9}$ = 0.0 (Equation 1b). This means Dr. Jones has recommended trust $T_r$*(A,* Dr. Jones, *healthcare service)* = (0.0+0.7)/2= 0.35 (Equation 3) with direct trust $T_d$ *(A,* Dr. Jones, *healthcare service)* = 0.0 (Equation 4). This makes the total trust value, T(A, Dr. Jones, healthcare service) = (25* 0.0 + 25*0.35 + 50 * 0.0)/100 = .07 (Equation 5) which means Dr. Jones will not get the service.

Instead, if Dr. Jones provides his context information and answers the questions, he will be given $T_{s_1} = T_{s_2} = T_{s_3} = T_{s_4} = 0.5$ (Equation 1a). This means that he can start thinking of accessing services of security-level 1, 2, 3, and 4. His trust value will again be calculated for those services, and if it becomes greater than 0.5, he can access that service. As Dr. Jones accesses a lower security level service, we can calculate his direct trust value (see equation 4), his trust value for that service (see equation 5), his total trust (see equation 6), and his trust value for that security level (see equation 7). Now Dr. Jones will be more reliable to A and is more likely to satisfy equation 1b and access his desired healthcare service.

## 7 Evaluation

We have evaluated our model using OmNet++ simulation tool with which we could measure the performance of the model. We varied the number of devices in the network to measure the performance. The graphs in this section were generated by using the simulation tool OmNet++ in which X – axis representing the number of devices in the network and Y – axis representing the response time for the simulation. We used different trust values for different devices and also for different services. We observed the model by varying the initial trust values and recommendation values. Figure 3 shows the graph for varying the initial trust values. In the graph, we observe that the values are the same for all the four series (each series represents the change in the value of initial trust). The similarity is due to not changing the mobility of the devices. We then tried to change the mobility of the devices, and this difference is displayed in the graph in Figure 4. The graph explains that when the requester is nearer to the service provider, the response time will be less. The node which is farthest from the provider has the higher response time. We also observed the behavior of the model for 10 devices in the network by changing the delay in broadcasting the packets. Figure 5 represents the graph for varying the delay in broadcasting. Finally, we observed the efficiency and scalability of the model by changing the number of devices in the network. Figure 6 represents this scenario. The representation of this graph depends on the number of devices that could give recommendations in favor of the service requester in the process of calculating the trust values.

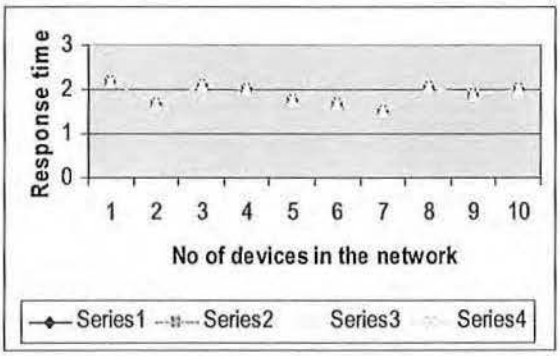



*Figure 3: Changing the values of initial trust*

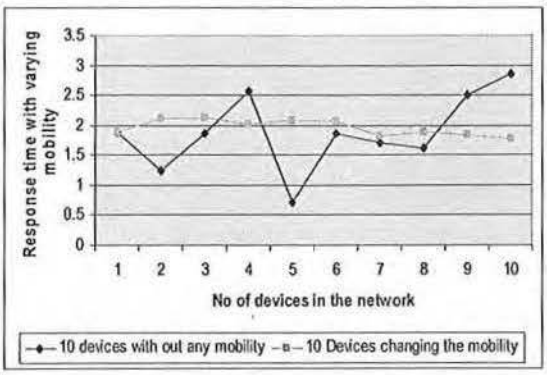


*Figure 4: Changing the mobility of the devices*

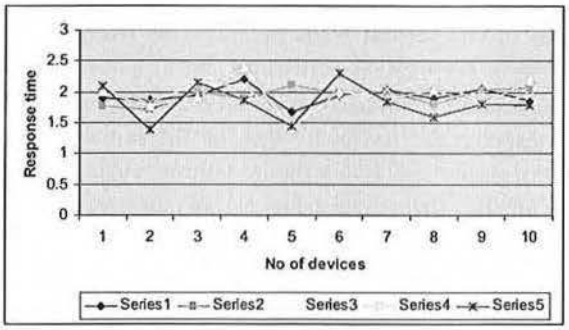



Figure 5: Varying the delay in broadcasting the packet

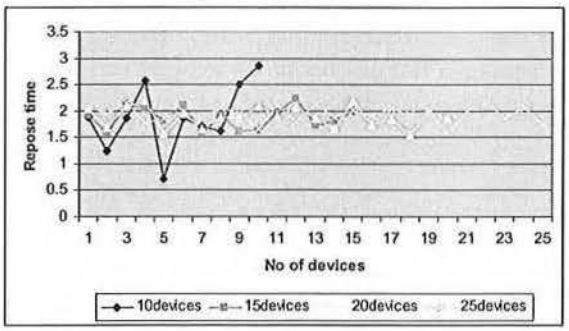
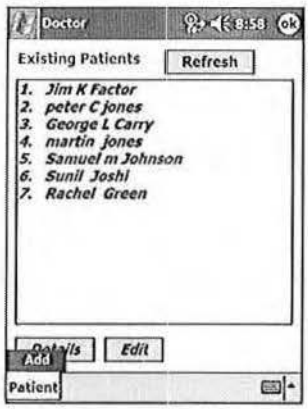


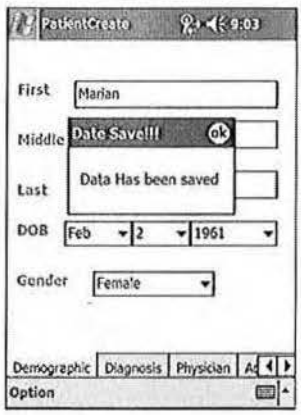

*Figure 6: Varying the No. Of devices*

We also developed the trust model on Dell Axim Pocket PC 50v using C#.NET compact framework. In addition, we developed a healthcare application [29] for the PDAs of nurses, doctors, and patients which use our trust model. We also implemented a location service prototype and used our trust model to evaluate DIRT's effectiveness. Indeed, location information is static and placed in an xml file. The idea was to implement it over our trust model. It is a prototype application of Scenario 3. Some of the screen shots of the application are provided in Figure 7.

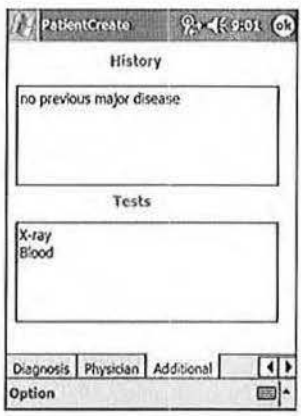

Try to access a file



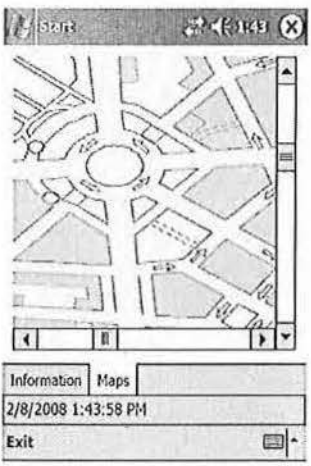

Patient record




Patient Data has been saved




Patients Additional information




Location Service showing Map


Selected Location Information

*Figure 7: Screenshots of Healthcare application [29] and Location Service application.*

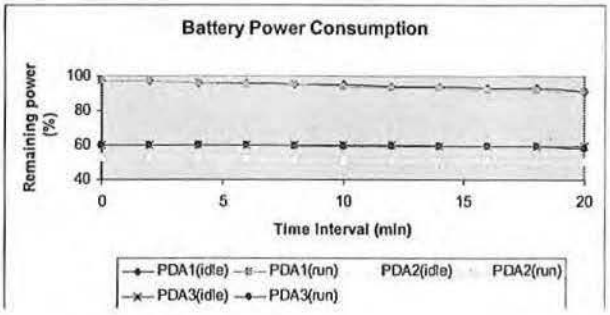



*Figure 8: Power consumption by all PDAs before and after running the Healthcare Aide*

Power consumption is one of the most important performance metrics of pervasive computing applications and services. Our healthcare application along with the trust model seems to be very power-conservative. It consumes the minimal amount of battery power possible. Figure 8 establishes this as a true statement. Here, we have shown two cases for each PDA: the idle case (when the PDA is ON but not doing anything) and the active case (when the application is running on the PDA).

## 8 Conclusion and Future Work

Instead of having a constant notion of initial trust value, our model showed how categorizing services into security levels, context information, and a simple questionnaire can decide initial trust. Our model uses recommendations from neighboring devices, direct trust experience, and inference from other service-trust values for that device to estimate the overall trust value. We defined characteristics of the pervasive environment and showed how these characteristics are satisfied in our model. Further, we reviewed the existing trust models and provided a comparison table with respect to the characteristics. Finally, we observed the performance of the model using OmNet++ simulation tool and implemented a prototype of healthcare and location-based service over our trust model.

In the future, we will work to categorize services more effectively and broadly according to different security-levels. We also plan to devise a communication-protocol to provide and share this classification data over a pervasive environment so that trust can be based on generic service-security-levels. We also need to map relevant context to each service category.

## References

1. A. Tripathi, T. Ahmed, D. Kulkarni, R. Kumar, and K. Kashiramka, "Context-Based Secure Resource Access in Pervasive Computing Environments," Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Vol. 00, Florida, 2004, pp. 159-163.
2. A. Josang, "The right type of trust for distributed systems," Proceedings of the 1996 workshop on New security paradigms, ACM, September, 1996.
3. J. Buford, I. P. Park, G. Perkins, "Social certificates and trust negotiation," Proceedings of the 3rd IEEE Consumer Communications and Networking Conference, Volume 1, Las Vegas, 8-10 January 2006, pp. 615-619.

4. C. Lin, V. Varadarajan, "Trust-Based Management for Distributed System Security – A New Approach," Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06), Vienna, Austria, April 2006.
5. C. English, P. Nixon, S. Terzis, A. McGettrick and H. Lowe, "Dynamic trust models for ubiquitous computing environments," Proceedings of the Fifth International Conference on Ubiquitous Computing, October 12-15, 2003, Seattle, USA.
6. D. Quercia, S. Hailes, and L. Capra, "B-trust: Bayesian Trust Framework for Pervasive Computing," iTrust. LNCS. May 2006. Pisa, Italy.
7. D. Quercia, S. Hailes, L. Capra: "Lightweight Distributed Trust Propagation", Proceedings of the 7th IEEE Int. Conference on Data Mining (ICDM 2007), September 2007, Omaha, USA
8. D. Gambetta, Can we trust?," In D. Gambetta (Ed.), Trust: Making and breaking cooperative relations, pp. 213-237. Oxford: Basil Blackwell.
9. F. Zhu, M. Mutka, L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services," Pervasive Computing and Communications, March 2003, pp. 235–242.
10. F. Zhu, M. Mutka, and L. Ni, "PrudentExposure: A Private and User-centric Service Discovery Protocol," Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), March 2004, pp. 329-340.
11. F. Zhu, M. Mutka, and L. Ni, "Expose or not? A progressive exposure approach for service discovery in pervasive computing environments," Proceedings of the third International Conference on Pervasive Computing and Communications, Mar 2005, pp. 225-234.
12. F. Almenarez, A. Marin, C. Campo, and C. Garcia, "PTM: A Pervasive Trust Management Model for dynamic open environments," Proceedings of the first workshop on Pervasive Security, Privacy, and Trust (pspt 2004), Boston, August 2004.
13. F. Almenarez and C. Campo, "SPDP: A Secure Service Discovery Protocol for Ad-hoc networks," Proceedings of the 9th open European summer school and IFIP workshop on net generation networks (EUNICE 2003), Hungary, September 2003.
14. F. Almenarez, A. Marin, D. Diaz and J. Sanchez, "Developing a model for trust management in pervasive devices," Proceedings of the 4th annual IEEE international conference on pervasive computing and communications workshops (PERCOMW'06), March 2006, Pisa, Italy
15. M. M. Haque; S. I. Ahamed, "An Omnipresent Formal Trust Model (FTM) for Pervasive Computing Environment," Proceedings of the 31st Annual Internatiobal Conference on 24-27 July 2007, pp. 49 –56.
16. O. K. Hussain; E. Chang; F.K. Hussain; T.S. Dillon; B. Soh;, "A Methodology for determining riskiness in peer to peer communications," 3rd IEEE International Conference on Industrial Informatics, Aug 2005, pages pp. 665-666.
17. J. Liu and V. Issarny, "Enhanced Reputation Mechanism for Mobile Ad Hoc Networks," Proceedings of the 2nd International Conference on Trust Management, vol. 2995. Oxford, UK: LNCS, March 2004, pp. 48–62.
18. K. Ranganathan, "Trustworthy Pervasive Computing: The hard Security Problems," Proceedings of the 2nd annual conference on pervasive computing and communications workshops (PERCOMW'04), 2004.

19. L. Kagal, T. Finin, A. Joshi, "Trust-based security in pervasive computing environments," IEEE Computer, volume 34, pages pp. 154-157, December 2001.
20. L. Kagal, T. Finin, A. Joshi, "Moving from security to distributed trust in ubiquitous computing environments," IEEE Computer, 2001.
21. L. Capra, M. Musolesi, "Autonomic trust prediction for pervasive systems," Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA 2006), 18-20 April 2006, Vienna, Austria. IEEE Computer Society 2006, pages pp 481 – 488.
22. M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," Fifteenth ACM Symposium on Principles of Distributed Computing, May 1996.
23. M. Sharmin, S. Ahmed, and S. I. Ahamed, "An Adaptive Lightweight Trust Reliant Secure Resource Discovery for Pervasive Computing Environments," Fourth Annual IEEE International Conference on Pervasive Computer and Communications (PerCom 2006), Pisa – Italy, Mar 2006.
24. Microsoft Corporation, "Universal Plug and Play Device Architecture," Version 1.0, Microsoft Co., 2000.
25. M. Sharmin, S. Ahmed, and S. I. Ahamed, "MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability, and Self Healing) in Pervasive Computing Environments," Third International Conference on Information Technology: New Generations, NV, USA, April 2006, pp. 306-313.
26. M. M. Haque, and S. I. Ahamed, "An Impregnable Lightweight Device Discovery (ILDD) Model for Pervasive Computing Environment," IEEE Transactions on Systems, Man, and Cybernetics, to appear, 2008.
27. M. Nidd, "Service Discovery in DEAPspace," IEEE Personal Communications, Aug 2001, pp. 39-45.
28. M. Sharmin, S. Ahmed, S. I. Ahamed, and H. Li , SSRD+: A Privacy-aware Trust and Security Model for Resource Discovery in Pervasive Computing Environment," Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC 2006), Chicago, USA, September 17-21, 2006, pp. 67-70.
29. M. Sharmin, S. Ahmed, S. I. Ahamed, M. Haque, and A. J. Khan, "Healthcare Aide: Towards a Virtual Assistant for Doctors, Patients, Nurses and Resident Doctors Using Pervasive Middleware," Proceedings of the 1st Workshop on Ubiquitous and Pervasive Health Care (UbiCare 2006), Pisa, Italy, Mar 2006, pp. 490-495.
30. M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," Communications of the ACM, Vol. 36, No. 7, Jul 1993, pp. 75-84.
31. N Shankar, W Arbaugh, "On trust for ubiquitous computing," Workshop on Security in Ubiquitous Computing, UBICOMP, 2002.
32. OMNet++ Discrete Event Simulation Systems, http://www.omnetpp.org.
33. P. Brezillon and G.K. Mostefaoui, "Context-based security policies: a new modeling approach," Second IEEE International Conference on Pervasive Computing and Communications Workshops, Florida, 2004, pp. 154–158.
34. D. Quercia and S. Hailes and L. Capra, TRULLO - local trust bootstrapping for ubiquitous devices. In: 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 6-10 Aug 2007, Philadelphia, USA.
35. Random House Unabridged Dictionary, Random House Inc., www.dictionary.com, 2006.

36. R. He, J. Niu, M. Yuan and J. Hu, "A novel cloud-based trust model for pervasive computing," 4th international conference on Computer and Information Technology, Sep 2004, pp. 693-700.
37. R.C.Mayer, J.H.Davis and F.D.Schoorman, "An integrative model of Organizational Trust," Academy of management review, 1995, Vol 20. No. 3, pp. 709 – 734.
38. R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In Proc. of USENIX Security, 1998.
39. S. L. Presti, M. Butler, M.Leuschel and C. Booth, "A trust analysis methodology for pervasive computing systems," Trusting Agents for Trusting Electronic Societies, Lecture Notes in Computer Science (LNCS), Springer – Verlag, 2005.
40. S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for P2P and mobile ad-hoc networks," Proceedings of the 2nd Workshop on the Economics of Peer-to-Peer Systems, Cambridge, USA, June 2004.
41. S. P. Marsh, "Formalising Trust as a computational concept," PhD thesis, University of Stirling, April 2004.
42. S. T. Wolfe, S. I. Ahamed, and M. Zulkernine, "A Trust Framework for Pervasive Computing Environments", Proceedings of the 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06), IEEE CS Press, Dubai, UAE, March 2006, pp. 312-319.
43. V. Cahill, E. Gray, J.M. Seigneur, C. Jensen, Y. Chen, "Using trust for secure collaboration in uncertain environments," IEEE CS and IEEE ComSoc, pp. 52 – 61, July – September 2003.