Dynamic Service Discovery as an Optimization Problem in Wireless Sensor Networks

Kaushik Lahiri[1], Amitava Mukherjee[1], Ayon Chakraborty[2], Subhajit Mandal[2], Dipankar Patra[2] and Mrinal K Nashkar[3]

[1]IBM India Pvt. Ltd., Kolkata-700091, India

[2]Dept. of CSE, Jadavpur University, Kolkata-700032, India

[3]Dept. of ETCE, Jadavpur University, Kolkata-700032, India

{lkaushik, amitava.mukherjee}@in.ibm.com, {jucse.ayon, msubhajitju, deepankarbcse}@gmail.com, mrinalnaskar@yahoo.co.in

Abstract. In a ubiquitous environment, e.g. smart home, a user is surrounded by a network of sensor nodes all around and also on his body or clothing. We modeled such a sensor network as a services network where the nodes exchange services for collaborating and smart decision taking. As the user moves around performing activities, the surrounding network also changes as wireless inter-node connections are made or broken. The challenge is to re-discover the network quickly and transparently to the user.

We used a two-step approach. First, Proximal Neighborhood Discovery identified nodes that formed the network. Second, Optimal Service Discovery determined, for each such network node, who were the best service provider nodes from the same network. We modeled this as an optimization problem and solved using a new and efficient algorithm.

We implemented the algorithm using nesC and simulated using TOSSIM interference-model. The results showed appreciable improvements over conventional approaches.

Key words: Ubiquitous environment, sensor network, smart home, Service Discovery, nesC, TOSSIM

1 Introduction

A Wireless Sensor Networks (WSN) consists of spatially distributed collections of small, smart and cheap, sensing and computing devices. In a traditional wireless sensor network, sensors are deployed in a specific geographical location and are required to send data to a sink. Some typical applications are habitat monitoring and military reconnaissance. Since these carry out simple tasks of remote sensing, they do not need a specific service discovery model, as routing data to the sink itself is sufficient to provide the required "service".

With the emergence of pervasive computing, users have gradually become surrounded by a variety of smart wireless computing devices. The ultimate goal is to realize a true ubiquitous computing paradigm where such devices weave themselves into the fabric of our everyday life by providing various different services and become invisible, as revealed by Weiser [1]. In order to realize this, we need many sensors, non-sensor smart devices and human beings moving and collaborating with each other. Together they perform multiple tasks like interfacing, querying, routing, and data acquisition.

An example is a medical application. Here, for detecting a patient's movement patterns, multiple sensors collaborate and exchange measurements perceived by light sensors and send this data to a processor sensor. This processor also receives signals from heartbeat and blood pressure monitoring devices implanted into the patient's

body. It combines and digitally processes all the signals received over a period of time and sends this to the nearest available analyzer sensor. The analyzer sensor takes help from a rule server sensor to quickly find patterns in the combined data that provide vital clues to the patent's health. It then routes only the analyzed summary data to a transmitter which further compresses it and securely sends it over the internet to a medical hub. The hub takes immediate action if necessary (e.g. notifying a doctor or ambulance) or simply stores the data in the patient's medical history for future analysis and action.

In the scenario just described, some sensors and devices are worn by the patient on his or her body and others are scattered around the patient's neighborhood (e.g. in the rooms of her home). There are two implications of this – 1) the relative positional co-ordinates of the sensors keep changing as the patient moves. 2) The set of collaborating sensors changes, as the most convenient and nearby sensors are used for quick analysis and transmission of data.

We can look upon the sensors as each providing some services of its own (service provider) and using some services provided by others (service consumer). From the point of view of a single sensor, it operates within a service neighborhood where it has to constantly collaborate with the neighbors to send service packets (where it is the service provider) and receive service packets (where it is consumer). Because of the implications described before, the neighbors keep changing their positions and their identities also change. Whenever such changes occur, each sensor has to re-discover the set of neighbors with whom it can collaborate most effectively (considering distance and packet loss) and who can also provide the set of services it requires. These changes can happen very frequently, e.g. as the patient moves about. As a result, we need extremely efficient and dynamic service discovery protocols.

Service discovery protocols for wireless networks [8] have been a sizzling subject for researchers over the past few years. These should allow devices to automatically detect useful services offered by other devices on a network along with their service attributes which help in determining their appropriateness in a given context. It also allows devices to advertise their own capabilities to the rest of the network. A few well known service discovery protocols are SLP, Sun Micro system's Jini and Microsoft's UPnP (Universal Plug and Play). However these protocols are not suited for ubiquitous environments which are dynamic, distributed and have energy constrained sensors. For such environments, highly efficient service discovery protocols are required which can adapt to the changing network topology. Such protocols should be able to quickly discover service providers in a dynamically changing network and also, more importantly, discover those providers which can provide the required services most efficiently, which is critical in a ubiquitous environment consisting of small sensor devices.

In this paper, we present a two level hierarchical approach for efficient service discovery. First, there is Proximal Neighborhood Discovery (PND) which a prerequisite for service discovery followed by the Optimal Service Discovery (OSD). Our algorithm is a based on modern heuristic techniques like Particle Swarm Optimization [2][3] and Simulated Annealing [4]. This algorithm finds out an optimal set of service providers from a potential set of sensors identified earlier by PND. It uses optimization parameters such as the distance of the provider and whether it is able to provide the required services. The main criteria used for choosing the parameters are 1) minimizing the use of communication power (e.g. by minimizing number of packet exchanges required) and 2) maximizing the packet reception probability (e.g. by using providers which have the best communication links with the consumer). Our results show appreciable improvements over conventional approaches, not only in terms of energy efficiency, but also minimizing packet loss.

We implemented our proposed algorithm using the nesC [5] programming language running on the TinyOS [6] software platform. Simulations were conducted using the TOSSIM [7] environment. The implementation of OSD in nesC not only shows the coding feasibility of the scheme, but also verifies that it can be run on real motes like MicaZ or Mica2. An interference model provided by the TOSSIM environment to simulate the unreliable wireless links is used for studying the successful packet delivery rates which made our simulation much more realistic.

The rest of the paper is organized as follows: Sections 2 introduces the rationale behind choosing the neighborhood and Service Discovery algorithms; Sections 3 and 4 elaborate on the working of the algorithms themselves; Section 5 discusses the implementation and compares the results obtained using our algorithm and other existing algorithms. We conclude in Section 6.

2 The rationale for Neighborhood and Service Discovery

As described in the introductory example, the network of sensors surrounding a particular sensor can change their positions quite frequently, e.g. as the patient embedded with a sensor enters or leaves a room. Whenever this happens, each sensor must quickly discover the new set of sensors surrounding it, i.e., its new neighbors. Discovering neighboring nodes (referred to as neighborhood discovery from now on) is a critical requirement for effective inter-node collaboration - inter-communication, routing and cluster formation.

Neighborhood discovery determines if direct single-hop radio communication is possible between two nodes. If the inter-node distance between the communicating nodes is greater than a certain threshold, then there is very high probability of packet loss due to larger interference and weaker signal strength. Although this situation can be partially improved by sending redundant packets or applying higher signal strength, these would drain out the resources (e.g. battery power) of the already energy-constrained sensor. Therefore it is critical to identify the right set of "neighbor" nodes (nodes with which a particular node can directly communicate by sending packets) with whom the best communication links can be established. This set of nodes can be termed as the "neighborhood" of the particular node.

Once each sensor has discovered its neighborhood, the network of sensors needs to collaborate and intercommunicate to achieve a set of objectives. When collaborating, each node acts as 1) a service provider by implementing a set of services itself and also 2) a service consumer where it depends on its neighbor nodes from whom it obtains a set of required services. In order to do this in the best possible manner, it is required that each service consumer node identifies the "optimal set of service providers" that can provide their required services. This optimal set can depend on several factors like the quality of connection with the providers, the number of bundled services that can be obtained from the same provider and the similarity of the service desired to the service provided. Considering all these factors and coming up with an optimal set of providers is the purpose of Optimal Service Discovery. Note that this optimal set should be arrived at in the fastest possible time. In fact, if there is a trade-off between the time taken for coming up with the solution and the quality of the solution itself, a higher priority should be given to a timely solution.

3 Proximal Neighborhood Discovery Algorithm

Proximal neighborhood discovery is the process by which a particular sensor discovers its neighboring nodes with which it has the best connectivity. The idea behind the discovery algorithm is simple.

Initialization*:* The foreign node n_0 (which enters into the network) broadcasts neighborhood request beacons periodically for $t_{MAX\_BEACONS}$ times. These beacon messages consist of the id of the transmitting node n_0. The idea behind multiple broadcasts is to account for unpredictable packet loss.

Acknowledgement*:* Whenever any node n_i receives a broadcast beacon from n_0, it replies back with an acknowledgement message containing its own id. Once n_0 receives back acknowledgement from n_i, it compares the received signal strength indication (RSSI) [13], S_i of the received acknowledgement message with a threshold value $S_{RSSI\_Threshold}$.

Selection: If $S_i > S_{RSSI\_Threshold}$, it selects n_i as a neighbor and adds it to the proximal neighborhood table t_{pn}. The threshold value of the signal strength is determined by factors such as the transmitting power of the node, the type of application and the physical environment in the network.

The discovery process is started immediately when the node enters a new network. Thereafter, for a particular node, inter-node communication is restricted to only those nodes that are in its proximal neighborhood table. As time passes, some nodes may leave the network or may drain out, or they might go out of range, also new nodes may join the network. Given the dynamical nature of the sensor networks and their topologies, nodes need to update themselves frequently about their neighbors by refreshing their local proximal neighborhood tables.

4 Optimal Service Provider Discovery Algorithm

As a pre-requisite for service discovery, a node first needs to know about its neighbors as described in Section 3. As a first step in service discovery, it needs to identify those neighbors that are capable for providing it with the services it needs. In a service-rich environment, more than one neighbor can provide each of the services that it needs. So the outcome of the first step can potentially be a large set of nodes. Therefore, as a second step in discovery, it needs to intelligently choose only those few service providers that are 1) necessary and sufficient for it to get all its required services and 2) able to provide the services most efficiently. Efficient service providers are chosen based on:

Distance of the provider: For service providers which are nearer, the quality of the link is also better, so there is a lesser chance of packets being lost.

The number of services provided by a single node: If the same provider can provide multiple services, then it is better, as there is a possibility for bundling of services, e.g. multiple types of information can be bundled on the same packet, thus reducing the number of packets that need to be exchanged.

"Appropriateness" of the services: Appropriateness of a service depends on how closely the consumer's requested service matches the provider's provided service. This is required as very often the services are similar but do not match exactly.

This can be looked upon as an optimization problem where, out of a set of available providers, the most efficient subset of providers must be identified.

The total set of neighbors have been already discovered (as in Section 3) and stored in the proximal neighborhood table t_{pn} as mentioned earlier. The initiator node n_0 starts by sending a service discovery beacon to each neighboring node n_i in table t_{pn}. These beacon messages consist of the id of the transmitting node n_0 and the set of services $S = \{S_{01} \ldots S_{0n}\}$ required by it. One a neighboring node gets this beacon, it checks if it can provide any of the services and replies by sending an acknowledgement message. After the initiator has waited for a sufficient period of time by which it was expecting to receive all acknowledgement messages, it prepares the Service Discovery Table.

As an input to the optimization problem, we have the Service Discovery Table, which is a relation from the set of services (S) to the set of service providers (SP). This is the set of all capable providers from which we need to identify the optimal subset of most efficient providers.

4.1 OSD Algorithm based on Particle Swarm Optimization

Problem Formulation: If the total number of required services is n, the solution space U can be said to be a collection of "arrangements" C, each of the form {SP_1, SP_2, .., SP_n}, where SP_i denotes the service provider selected for providing the i^{th} service in the particular arrangement. Let Q_i be the set of "all" providers capable of providing the i^{th} service. Thus, $\bigcup_{i=1}^{n} Q_i$ = SP which is nothing but the total set of service providers and U = {(SP_1, SP_2, .., SP_n) | SP_i □ SP for each i}. Our problem is to find the optimal arrangement C from our solution space U, i.e., the optimal way to assign service providers for each of n services that are desired by a service consumer sensor.

In order to solve this optimization problem, we start based on the idea of Particle Swarm Optimization (PSO) [2] and then apply to it the principle of Simulated Annealing (SA) [10], in order to arrive at an optimal solution in a short time. Using this idea, an arrangement C_i denotes the i^{th} *particle* in an n-dimensional system where each dimension represents a particular service. One restriction is that the *velocity* of a particle (the number of values it can have, i.e. the number of service providers that can be assigned) along a particular dimension (service) is finite and restricted to a set of possible values, that is, the i^{th} dimension of the particle has a domain restricted to set Q_i . We call the set Q_i the *velocity space* along a particular dimension i. If the velocity restriction is not adhered to, the arrangement would not be valid.

Formulation of the Energy Function: While formulating the energy function or cost function, we used two important thumb rules. First, the service consumer sensor should obtain as many services as it can get from a single provider, i.e., look for bundled services. This would make it possible for the single provider to combine more than one service (e.g. information on a patient) on the same packet, thereby reducing the overall number of data packets that need to be sent. Second, the service providing nodes should be as closer as possible. This is important because the signal strength between communicating nodes decreases quickly with the distance apart which means that either more energy or greater number of packets are needed to successfully send data.

Mathematically, the first goal is to minimize the 'average distance of a service' for a particular service provider node and the second goal is to minimize the sum of the service distances for all the service providers. With this goal, we formulate the energy function required for simulated annealing. For a particular arrangement C_i the energy function f(i) can be stated as,

$$f(i) = \sum_{i=1}^{\eta(sp)} d(i)/\gamma(i) \qquad (1)$$

Where d(i) is the distance of the service provider with id number i, and γ(i) is the number of services (bundled service) provided by that particular provider.

Note that a nearer service provider implies a smaller d(i) and a bundled service implies a larger γ(i). As the ratio of d(i) and γ(i) decreases, the arrangement C_i of service providers becomes more and more favorable. This is what we wanted to achieve in the first place. So equation (1) is a suitably formulated energy function.

As discussed before, any arrangement C_i can be looked upon as a particle in an n-dimensional space. When a particle updates its position from C_{old} to C_{new}, the energy gained is given by: $\Delta f = f(C_{new}) - f(C_{old})$. This is same as ΔE, *the energy difference* between the two states, which is an important parameter in SA.

Iterative approach to the solution based on SA: We start from an initial solution, which can be any valid arrangement and calculate its energy value. At the next step and then each subsequent step, we change the current solution to another valid solution and recalculate the energy value. We accept or reject the new solution based on a probability determined by the following *probability function* P:

$$P = 1 \quad \text{if} \quad \Delta E \leq 0$$

$$= e^{(\frac{-\Delta f}{\theta})} \quad \text{if} \quad \Delta E > 0 \qquad (2)$$

Here, Θ is a variable control parameter, called the *annealing temperature*, which is initially at a high value, but gradually decreases based on a *cooling schedule*, described in section IV.

If ΔE is negative (or zero), implying lower (or same) energy value of the new solution, it is always accepted, as P=1.

If ΔE is positive and P > random (0, 1), i.e., P is greater than a random number between 0 and 1, the new solution is accepted else it is rejected.

This second decision is of utmost importance, as we accept even higher energy value solutions with a certain probability, depending on the annealing temperature Θ. This ensures that our algorithm is not stuck at a local minimal value, which implies a sub-optimal solution. We proceed in this way, iteratively towards our ultimate solution till the annealing temperature has cooled down to its desired final value.

Cooling Schedule: An important control parameter in equation (2) is Θ, called the *annealing temperature*; a parameter which is decremented, every time the system of particles approaches a better solution (or a low energy state). If Θ_i be the *initial temperature* and Θ_f be the *final temperature*, and t is the cooling time, then the designed *cooling schedule* is given by: $\Theta(t) = \Theta_f + (\Theta_i - \Theta_f)*\alpha^t$. Here α is the rate of cooling, usually ($0.7 \leq \alpha < 1.0$) and t is the cooling time which in our case is the number of iterations. While the SA algorithm incorporates the concept of probability through the Metropolis acceptance rule [10], it can be slow. So in our proposed algorithm, we also utilize the fast optimal search capability of PSO.

Proposed algorithm based on PSO: Input: A set of N services and the set of Service providers pertaining to each instance of a service.

Step 1: Initialization: At first, a swarm of m particles C_1, C_2,....,C_m is initialized randomly:.

Here, a particle $C_i = \{SP_1^i, SP_2^i,..., SP_n^i\}$ is an arrangement of service providers where SP_k^i is the service provider which provides the k^{th} service in the i^{th} particle (arrangement) C_i. Here the number of services desired by the consumer is n.

The parameters Θ_i (initial temperature), Θ_f (final temperature), α (cooling rate) are initialized. A higher Θ_i gives a better result, but there is also a trade-off as this implies a larger number of iterations.

Step 2: Finding a local best solution: For each of the m particles, the next local best solution is found, at a particular temperature Θ, as described in section III. This is done by randomly selecting a member from the velocity space along a random dimension less than or equal to N. This means that any one of the N service providers is chosen randomly and replaced by another valid service provider, which also is chosen randomly. The new solution C_{new} probabilistically replaces the old solution based on the probability function described earlier in equation (2). The updated solution is referred to as C_{ilbest} or the local best.

Step 3: Updating the pbest and gbest values: For each particle, C_{ilbest} obtained in step 2 is compared to the *historically obtained best solution for that particle* C_{ipbest}. C_{ipbest} is updated by C_{ilbest} according to the following rule:

$C_{ipbest} = C_{ilbest}$ if $f(C_{ilbest}) - f(C_{ipbest}) < 0$

$= C_{ipbest}$ if $f(C_{ilbest}) - f(C_{ipbest}) \geq 0$

Now, comparing all the C_{ipbest} values, Cgbest (the historically obtained best solution globally across all particles) is updated by that C_{ipbest} which has the minimum energy state i.e. the minimum value of $f(C_{ipbest})$ among all particles.

Step 4: Finding new solution based on crossover: Based on the global knowledge of the swarm each particle forms a new solution from its local best (C_{ipbest}) by crossing a part of it with the global best (C_{gbest}). For example, say, C_{ipbest} = {1,5,**2,3,2**,1} and C_{gbest} = {1,3,**1,4,3**,3}. The portion {1,4,3} is randomly chosen from C_{gbest} and inserted in the same position in C_{ipbest} to obtain C_{inew}. Thus C_{inew} becomes {1,5,**1,4,3**,1}.
Note that this is slightly different from the crossover operator discussed in [11].
After the crossover, C_{ipbest} is replaced with C_{inew} if that has a lower energy state and is taken as the new individual best position. Crossover helps the particles jump out of local optimization by sharing global information about the swarm.

Step 5: Loop: The temperature Θ(t) is calculated. If either Θ(t) is less than Θ(f) or the number of iterations completed exceeds t, the algorithm comes to a halt. The best solution found is C_{gbest}. Else it goes back to step 2.

Possible extension: parameter matching and degree of association: The above algorithm can be extended to take into account the *appropriateness* of the service provider (how closely it matches the service desired by the consumer node) in cases where the types of parameters are variable. Suppose that a desired service S has parameters $\{p_1, p_2, \ldots, p_k\}$. Also suppose that the service S is provided by Service Providers SP_1, SP_2, …, where each of them provides the same service but with varying types of parameters. A very trivial example of varying types can be string versus integer versus floating point number. The most appropriate service provider would be the one whose parameter list matches most closely with the requested parameter list. Thus appropriateness of service provider i depends on the *degree of association* between the set $\{p_1, p_2, \ldots, p_k\}$ and the parameter set of SP_i. Degree of association is given by:

$$D = \sum_{j=1}^{k} (p_j \Phi SP_i(p_j)) \qquad (3)$$

Where, $SP_i(p_j)$ is the j^{th} parameter in set SP_i (parameter set for the service provider providing the i^{th} service to the service consumer) and p_j^i is the j^{th} parameter in the parameter set for the i^{th} service desired by the service consumer and $p_j^i \Phi SP_i(p_j)$ is the *correspondence*. The value of *correspondence* is given by:

$p_j^i \Phi SP_i(p_j) = 1$ if the type of p_j^i exactly matches the type of $SP_i(p_j)$

$= 0$ otherwise

The degree of association can be incorporated into the definition of distance d(i) that is used in the energy function (1). Alternatively, a strategy can be used to:

1) Produce several distinct sub-optimal particles.
2) Take the one having the maximum degree of association.

5 Simulation

Extensive simulations have been performed to judge the capability of the OSD algorithm. We have considered a sample network of 30 nodes and the existence of 10 network services. Each node provides a set of services and consumes a predefined set of services.

5.1 Implementation

We implemented the neighborhood discovery and service discovery using the nesC [5] programming language, hosted on the TinyOS [6] software platform which provides a component-based software model and an active message based communication model. The rationale behind choosing the implementation platform is to test its coding feasibility on the hardware platform (e.g., real motes like mica2, micaZ etc.) and also to utilize the interference-model provided by TOSSIM [7] to study packet-loss. NesC modules are software components that are wired together in a configuration file to form an application, much like hardware components in a schematic.

The Interference-Model: We have noticed the criticality of packet loss during transmission while simulating our scheme using the interference-model. The simulation in TOSSIM [7] considers the TOSSIM radio loss model, which is based on the empirical data (shown in Fig. 1). The loss probability captures transmitter interference using original trace that yielded the model. More detailed measurements would be required to simulate the exact transmitter characteristics; however experiments have shown the model to be very accurate.

After the nodes boot, they try to determine their neighbors. To avoid collision we have used TDMA-based approach for the nodes to transmit their neighborhood discovery messages. The implementation takes RSSI, where the signal strengths of the acknowledgement messages sent by the potential neighbors are sampled by the subject node. Our application uses a threshold value to help filter out nodes beyond a certain distance, which is directly related to the RSSI value. Results regarding relations between the RSSI values and distances have been studied in [9].

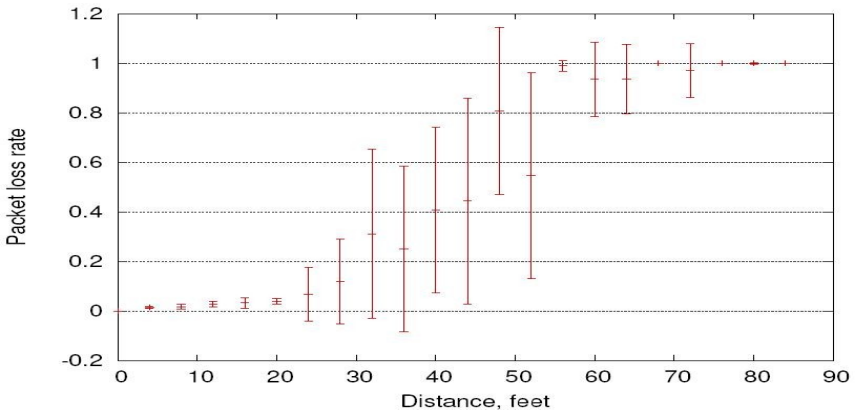

Fig. 1. TOSSIM Radio Loss Model based on empirical data

5.2 Simulation Results

The results obtained by the OSD algorithm (based on SA-PSO) were compared with results from simple PSO algorithm and a Greedy Algorithm. The PSO algorithm does not employ the notion of Simulated Annealing where as the Greedy Algorithm is more concerned only with the bundling of services from a peer.

Since all the algorithms employ an idea of random numbers the solutions produced are not at all always unique. However, the distribution of the obtained solutions varies with the algorithms. We conducted a study on this distribution, where we simulated the three algorithms for 100 times for a given network and plotted frequency of the obtained solution versus the value of energy (as in equation 1) of the obtained solution. The graph obtained is presented in figure 2.

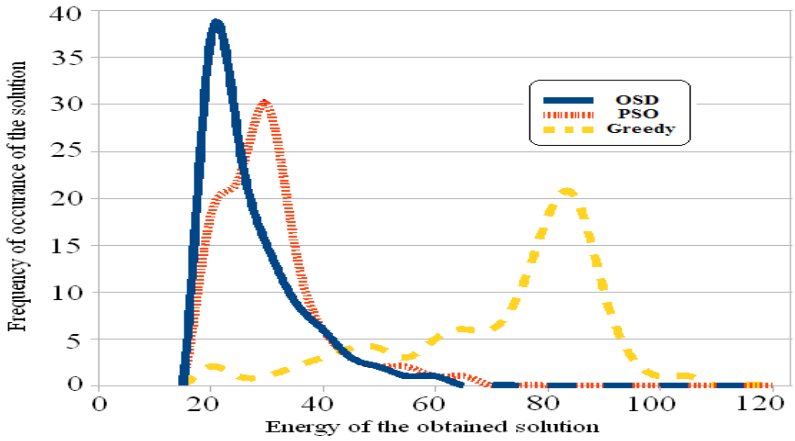

Fig. 2. Distribution of the solutions

The figure 2 shows that the graph for OSD has a greater measure of Kurtosis or *peak* than the other two. Hence the probability of the obtained optimal solution to be at a minimal energy level is greater in case of OSD, where the optimal solutions are more clustered towards the minimal energy solution. To study the merit of our approach, we performed the analysis based on the following criteria among the algorithms: a) percentage packet-loss and b) energy efficiency. The simulation was run for 50 rounds and the average number of packet losses for the three different algorithms were plotted graphically, as shown in figure 3. This simulation uses the interference-model introduced earlier. The OSD algorithm shows distinctively reliable communication compared to the others.

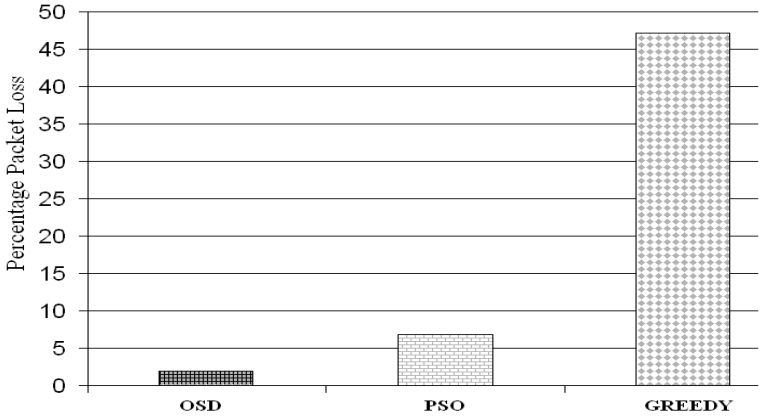

Fig. 3. Percentage packet loss for the schemes

In case of the greedy algorithm, packet loss is the maximum as communication distance is not considered. The relatively higher packet loss in PSO shows that the application of simulated annealing technique improves the OSD algorithm. The second simulation was performed for determining the energy-awareness of the three schemes. Here, we studied the residual energy of the initiator node the number of rounds increases gradually. The energy expenditure for communication follows the first order radio model as discussed in [12]. The initiator (consumer) node is assumed to have an initial energy of 1 J. The results are shown in figure 4. The cost of communication is best in the OSD scheme. Residual energy curves show a sharper decreasing trend in the other schemes. This increases the lifetime of individual nodes

resulting in increment of the network lifetime. It is clear from the above results that the OSD algorithm outperforms the other two algorithms, both in terms of higher energy conservation as well as lower packet-loss rates.

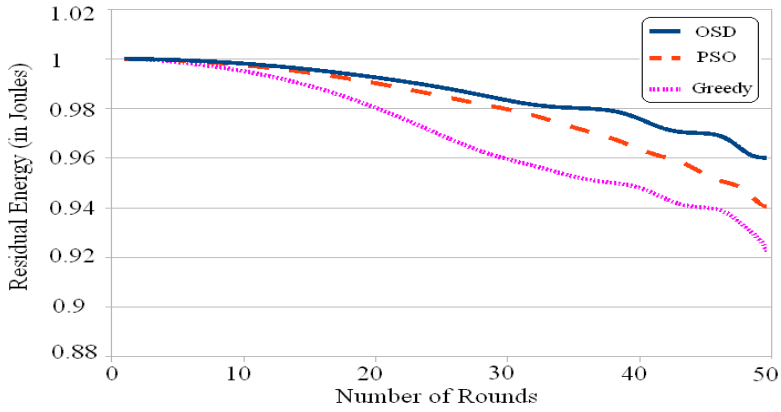

Fig. 4. Residual Mote Energy of the initiator after 30 rounds.

5 Conclusion and future work

In this paper, we have presented a combined two-level strategy for nearest neighborhood and optimal service provider discoveries that would provide a single solution to this challenge. We described the OSD algorithm, a new optimization approach based on a combination of SA and PSO algorithms, which prevents getting stuck in local optimal solutions while at the same time quickly converges towards the final solution. We implemented this using the nesC programming language running on the TinyOS platform. We have shown that our results show appreciable improvements over two other conventional approaches. Developing applications using our approach offers two major advantages:
1. Execution time is much shorter than when using more traditional approaches.
2. The algorithm is robust, being relatively insensitive to noisy and/or missing data, as it randomly searches for the best solution over the entire solution space.

To conclude, it is our belief that optimizing the service provider selection mechanism is an important step towards minimizing energy consumption when sensor nodes collaboratively exchange services or data in a Wireless Sensor Network. For greater applicability of wireless networks in a ubiquitous environment, one should also look at the context under which different services are required and take into account the security of services being provided over an open network. Moreover, in an actual WSN, there can be intermittent total disconnection/discontinuity of services and one should take proactive steps to continue operating even under such circumstances. Our future work would endeavor to enhance our programming framework to allow these enhancements.

References

1. Weiser, M.: The Computer for the Twenty-First Century, Scientific Am, vol. 265, no. 3, 1991, pp. 94–101.
2. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory, In the proceedings of the Sixth International Symposium on Micro machine and Human Science Nagoya, Japan, 1995, pp. 39 – 43.
3. Shi, YH., Eberhart, R.C.: A modified particle swarm optimizer, IEEE International Conference on Evolutionary Computation, 1998, pp. 63-73.

4. Chaojun, D., Zulian, Q.: Particle Swarm Optimization Algorithm Based on the Idea of Simulated Annealing, IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.10, October 2006
5. Gay, D., Levis, P., Behren, R.., Welsh, M., Brewer, E., Culler, D.: The nesC language - A holistic approach to networked embedded systems, ACM SIGPLAN Notices archive Volume 38 , Issue 5 (May 2003) .
6. Philip Levis et al.: TinyOS - An Operating System for Sensor Networks. Ambient Intelligence, 2005 – Springer.
7. P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS.
8. Lenders, V., May, M., Plattner, B.: Service discovery in mobile ad hoc networks: A field theoretic approach, Pervasive and Mobile Computing, 2005 – Elsevier.
9. Lim, J.C. and Wong, K.D. Exploring Possibilities for RSSI-Adaptive Control in Mica2-based Wireless Sensor Networks, ICARV 2006.
10. Kirkpatrick,. S., Sorkin, G.B.: Simulated Annealing, in The Handbook of Brain and Neural Networks, 1995, The MIT Press: Cambridge, MA.
11. Zhi-Feng Hao, Zhi-Gang Wang; Han Huang,: A Particle Swarm Optimization Algorithm with Crossover Operator, International Conference on Machine Learning and Cybernetics 2007, pp -19-22, Aug. 2007.
12. Rabiner, W., Heinzelman, Chandrakasan, A., and Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Micro sensor Networks. In the proceedings of the 33rd Hawaii International Conference on System Sciences – 2000.
13. K. Whitehouse, C. Karlof, and D. Culler. A practical evaluation of radio signal strength for ranging-based localization. SIGMOBILE Mob. Comput. Commun. Rev., 11(1):41–52, 2007.