International Conference on Sensors and Related Networks (SENNET'07),
VIT University, Vellore, India. Dec. 12-14, 2007. pp.448-453.

Simulation of Sensor Network Protocols and their Comparison

Arun Kumar and Ashok Kumar Turuk

Abstract— **This project aims to evaluate and compare the performance of two of the routing protocols for wireless sensor networks. The two routing protocols chosen for comparison are Flooding and Directed Diffusion. The project thus involves the study of wireless sensor networks, simulation of Flooding and Directed Diffusion protocols in the NS simulator and finally analyzing the trace files to gather information required for comparison. Comparison is performed on a set of metrics and the results are presented in the form of graphs showing the impact of network size on each metric. The metrics chosen for comparison are Average delay, Duplication overhead, Packet delivery ratio, Routing overhead, Data delivery cost and throughput. On the basis of results obtained after comparison, it is finally concluded that Directed Diffusion can perform much better than the traditional Flooding scheme in similar conditions of network size and work load.**

I. Introduction

Wireless sensor networks are potentially one of the most important technologies of this century. Recent advancement in wireless communications and electronics have enabled the development of low-cost, low-power, multifunctional miniature devices for use in remote sensing applications. The combination of these factors has improved the viability of utilizing a sensor network consisting of a large number of intelligent sensors, enabling the collection, processing analysis and dissemination of valuable information gathered in a variety of environments. A sensor network is composed of a large number of sensor nodes which consist of sensing, data processing and communication capabilities. These nodes are densely deployed either inside the phenomenon or very close to it. The position of sensor nodes need not be engineered or predetermined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities. Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an onboard processor. Instead of sending the raw data to the nodes responsible for the fusion, they use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. Sensor networks can be classified into two broad types; homogeneous and heterogeneous sensor networks. In homogeneous networks all the sensor nodes are identical in terms of battery energy and hardware complexity. On the other hand, in a heterogeneous sensor network, two or more different types of nodes with different battery energy and functionality are used. The fundamental objectives for sensor networks are reliability, accuracy, flexibility, cost effectiveness and ease of deployment.

Dept. of CSE, NIT Rourkela, Orissa, India.
Email: arun_iert009@yahoo.com, arun.nit.rkl@gmail.com

II. A Sensor Network Example

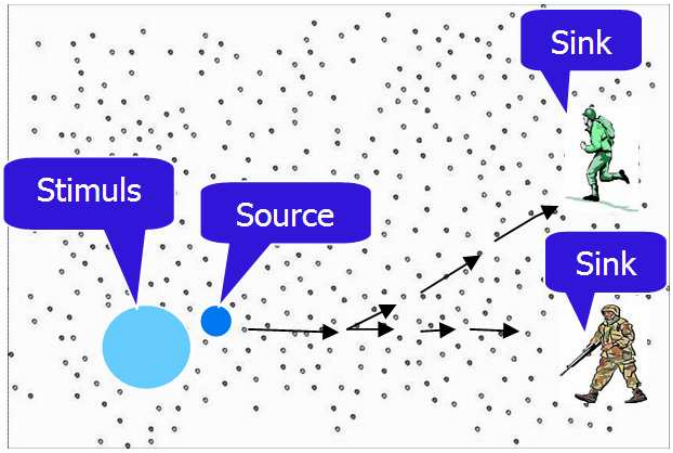


Fig. 2. A sensor network example. Soldiers use the sensor network to detect tank locations

In a sensor field, a *source* refers to a sensor node that generates sensing data to report about a *stimulus*, which is a target or an event of interest. A sink is a user that collects these data reports from the sensor network. Both the number of stimuli and that of the sinks may vary over time. For example, in fig 2, a group of soldiers collect tank movement information from a sensor network deployed in a battlefield. The sensors surrounding a tank detect it and collaborate among themselves to aggregate data, and one of them generates a data report. The soldiers collect these data reports. In a sensor network sensor nodes may either be stationary or mobile. In this case the sensor field has stationary sensor nodes only, whereas sinks may change their

locations dynamically i.e. the soldiers may move around, but must be able to receive data reports continuously.

A. Flooding protocols

The broadcast of messages is a frequently used operation to spread information to the whole network. It is the simplest building block used by network algorithms and is often required by higher level protocols such as most routing algorithms. For this reason, it is important for the broadcast to be implemented in the most efficient way. Its performance is likely to affect the global efficiency of any protocol using it.

Flooding is an old technique that can also be used for routing in sensor networks. In flooding, each node receiving a data or management packet repeats it by broadcasting, unless a maximum number of hops for the packet is reached or the destination of the packet is the node itself. Flooding is a reactive technique, and it does not require costly topology maintenance and complex route discovery algorithms.

1) Architecture

The flooding algorithm is build on top of the MAC layer protocol as shown in Fig 2.1.

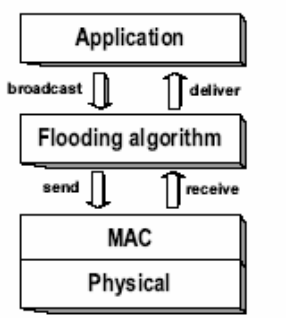


Fig. 2.1. (a) Algorithm protocol stack

The application layer, not described here, accesses the Flooding algorithm (FA) by the broadcast and deliver primitives. The broadcast primitive is called each time the application wishes to initiate a new broadcast. The deliver primitive is called by the FA whenever a new broadcast message is received. The send primitive floods a message by locally broadcasting it to the node's neighborhood. When a new message is sensed by the MAC layer, it is forwarded to the FA by the mean of the receive function. All messages exchanged during the simulations are sent to a broadcast MAC address. This means that each node that senses an incoming message will always deliver it to the FA layer. Finally, no transport protocol such as UDP or TCP is used and the FA is directly connected to the MAC layer. This minimal protocol stack ensures that the comparison between simulators will only depend of the differences between the MAC and physical layers modelisations.

2) Flooding example

A rather direct and simple way to implement broadcast is to flood the message over the network. When a node initiates a broadcast, it transmits the message to its neighborhood. By neighborhood, we mean all the nodes within the sender's transmission range. Then, when the message is received for the first time, the recipient re-broadcasts it. An example is shown in Figure 1 with a network composed of fie mobile nodes labeled from A to E.

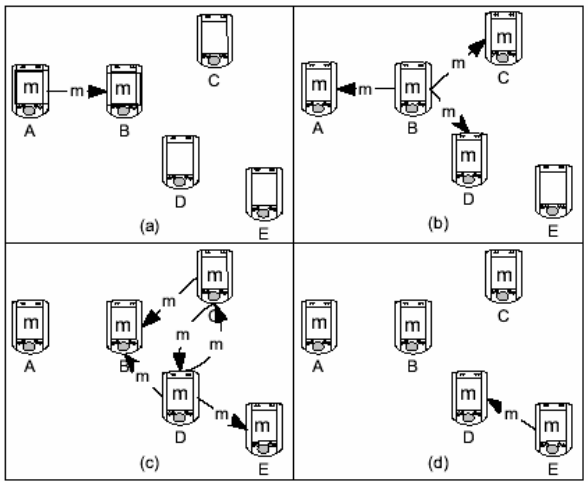


Fig. 2.1. (b) Flooding Example

Node A initiates a broadcast by flooding a message m to its surrounding nodes. In step (a), A floods m to its single neighbor B. Then, in step (b), node B, which has received m for the first time, rebroadcasts m to nodes C and D and so on. In (c), m is completely flooded through the whole network and delivered to every node. In this example, at least three steps are required in order to reach node E on the right. The last step (d) is useless as every neighbor of E has already received m.

This technique has an important drawback : it leads obviously to an overhead of flooded messages in the network. With ideal conditions (i.e. all node receive the broadcast) in a network of N nodes, a single broadcast will generate exactly N copies of itself which are likely to increase the probability of collisions. Moreover, most nodes will receive the same message several times keeping the shared medium unnecessarily busy.

B. The directed diffusion protocol

1) Naming

In directed diffusion, task descriptions are *named* by, for example, a list of attribute-value pairs that describe a task. A vehicle- tracking task might be described as this is a simplified description :

type = wheeled vehicle // detect vehicle location

interval = 20 ms // send events every 20 ms

duration = 10 s // for the next 10 s

rect = [100,100, 200, 400] // from sensors within rectangle.

For ease of exposition, we choose the sub region representation to be a rectangle defined on some coordinate system; in practice, this might be based on GPS coordinates. Intuitively, the task description specifies an interest for data matching the attributes. For this reason, such a task description is called an *interest*. The data sent in response to interests are also named using a similar naming scheme. Thus, for example, a sensor that detects a wheeled vehicle might generate the following data:

type = wheeled vehicle // type of vehicle seen

interval = truck // instance of this type

location = [125, 220] // node location

intensity = 0:6 // signal amplitude measure

intensity = 0:85 // confidence in the match

timestamp = 01 : 20 : 40 // event generation time.

Given a set of tasks supported by a sensor network, then, selecting a naming scheme is the first step in designing directed diffusion for the network. For our sensor network, we have chosen a simple attribute-value based interest and data naming scheme. In general, each attribute has an associated value range. The value of an attribute can be any subset of its range. In our example, the value of the attribute in the interest is that corresponding to wheeled vehicles. To some extent, the choice of naming scheme can affect the expressivity of tasks and may impact performance of a diffusion algorithm.

2) Interests and Gradients

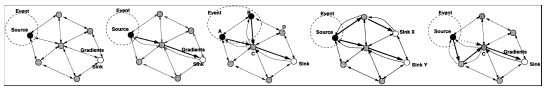
An interest is usually injected into the network at some (possibly arbitrary) node in the network. We use the term *sink* to denote this node.

3) Interest propagation

For each active task, the sink periodically broadcasts an interest message to each of its neighbors. This initial interest contains the specified rect and duration attributes, but contains a much larger interval attribute. Intuitively, this initial interest may be thought of as *exploratory*; it tries to determine if there indeed are any sensor nodes that detect the wheeled vehicle. To do this, the initial exploratory interest specifies a low data rate.

4) Gradient establishment

Fig. 2.2(a) shows the gradients established in the case where interests are flooded through a sensor field. Notice that every pair of neighboring nodes establishes a gradient towards each other. This is a crucial consequence of local interactions. When a node receives an interest from its neighbor, it has no way of knowing whether that interest was in response to one it sent out earlier, or is an identical interest from another sink on the "other side" of that neighbor. Such two-way gradients can cause a node to receive one copy of low data rate events from each of its neighbors. Note that for our sensor network, a gradient specifies both a data rate and a direction in which to send events. More generally, a gradient specifies a *value* and a direction.



(a) (b) (c) (d) (e)

Fig. 2.2(a). Illustrating different aspects of diffusion. (a) Gradient establishment.(b) Reinforcement. (c) Multiple sources. (d) Multiple sinks. (e) Repair

In summary, interest propagation sets up state in the network (or parts thereof) to facilitate "pulling down" data toward the sink. The interest propagation rules are *local* and bear some resemblance to join propagation in some Internet multicast routing protocols.

5) Data propagation

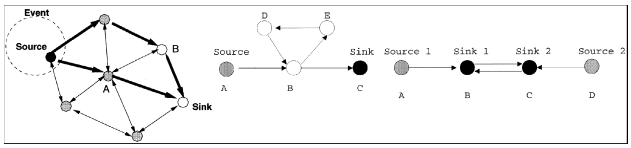
A sensor node that is within the specified rect processes interests as described in the previous section. A sensor node that detects a target searches its interest cache for a matching interest entry. In this case, a matching entry is one whose rect encompasses the sensor location and the type of the entry matches the detected target type. When it finds one, it computes the highest requested event rate among all its outgoing gradients. The node tasks its sensor subsystem to generate event samples at this highest data rate. In our example, this data rate is initially one event per second (until reinforcement is applied). The source then sends to each neighbor for whom it has a gradient an event description every second.By examining its data cache, a node can determine the data rate of received events. To resend a received data message, a node needs to examine the matching interest entry's gradient list. If all gradients have a data rate that is greater than or equal to the rate of incoming events, the node may simply send the received data message to the appropriate neighbors. However, if some gradients have a lower data rate than others (caused by selectively reinforcing paths), then the node may *downconvert* to the appropriate gradient. For example, consider a node that has been receiving data at 100 events per second, but one of its gradients is at 50 events per second. In this case, the node may only transmit every alternate event toward the corresponding neighbor.

6) Reinforcement for Path Establishment and Truncation

In the scheme we have described so far, the sink initially and repeatedly diffuses an interest for a low-rate event notification. We call these *exploratory* events, since they are intended for path setup and repair. We call the gradients set up for exploratory events *exploratory* gradients. Once a source detects a matching target, it sends exploratory events, possibly along multiple paths, toward the sink. After the sink starts receiving these exploratory events, it *reinforces* one particular neighbor in order to "draw down" real *data* (i.e., events at a higher data rate that allow high quality tracking of targets). We call the gradients set up for receiving high-quality tracking events *data* gradients.

Path Establishment for Multiple Sources and Sinks: In describing reinforcement so far, we may have appeared to implicitly describe a single-source scenario. In fact, the rules we have described work with multiple sources. To see this, consider Fig. 2.4(c). Assume initially that all initial gradients are exploratory. According to this topology, data from both sources reaches the sink via both of its neighbors **C** and **D**. If one of the neighbors, say, **C** has consistently lower delay, our rules will only reinforce the path through **C** (this is depicted in the figure). However, if the sink hears **B**'s events earlier via **D**, but **A**'s events5 earlier via **C** , the sink will attempt to draw down high-quality data streams from *both* neighbors (not shown). In this case, the sink gets both sources' data from both neighbors, a potential source of energy inefficiency. Such problem can be avoided with some added complexity. Similarly, if two sinks express identical interests, our interest propagation, work correctly. Without loss of generality, assume that sink **Y** in Fig.2.4 (d) has already reinforced a high-quality path to the source. Note, however, that other nodes continue to receive exploratory events. When a human operator tasks the network at sink **X** with an identical interest, **X** can use the reinforcement rules to achieve the path shown. To determine the empirically best path, **X** *need not wait* for data—rather, it can use its data cache to immediately draw down high-quality data toward itself.

Local Repair for Failed Paths: So far, we have described situations in which reinforcement is triggered by a sink. However, in directed diffusion, *intermediate* nodes on a previously reinforced path can apply the reinforcement rules. This is useful to enable *local repair* of failed or degraded paths. Causes for failure or degradation include node energy depletion and environmental factors affecting communication (e.g., obstacles). Consider Fig. 2.4(e), in which the quality of the link between the source and node **C** degrades and events are frequently corrupted. When **C** detects this degradation-either by noticing that the event reporting rate from its upstream neighbor (the source) is now lower, or by realizing that other neighbors have been transmitting previously unseen location estimates—it can apply the reinforcement rules to discover the path shown in the figure. Eventually, **C** negatively reinforces the direct link to the source (not shown in the figure).



(a) (b) (c) (d) (e)

Fig. 2.2. (b) Negative reinforcement for path truncation and loop removal. (a) Multiple paths. (b) Removable loop. (c) Unremovable loop

C. Comparison

1) Average delay

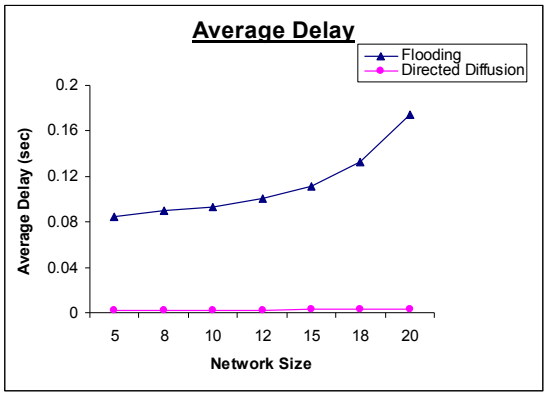


Fig. 2.3.1. Impact of Network size on Average Delay

Average Delay measures the average one way latency observed between transmitting an event and receiving it at each sink. Ideally, *Average Delay* should have a rather constant value. Fig 2.3.1 shows the average delay observed as a function of network size.

2) Duplication overhead

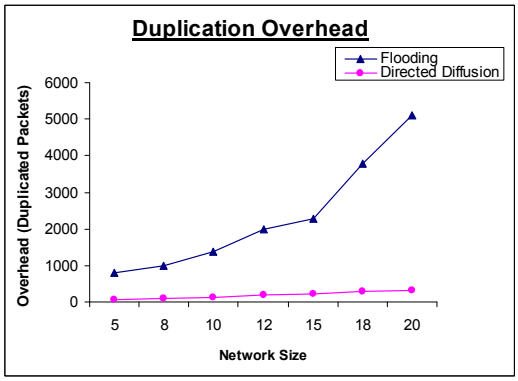


Fig. 2.3.2. Impact of Network size on Duplication Overhead

For a node n, *Duplication overhead* (or simply Overhead) is the sum of duplicated packets received at n. Duplication of packets implies wastage of energy, so it should be as small as possible.

3) Packet delivery ratio

Fig 2.3.3 shows the packet delivery ratio as the function of network size. Though the ratio increases above the ideal ratio for both the protocols with the increase in network size, the rate of increase for directed diffusion is much lower than for flooding and becomes almost constant after a certain network size. The sharp increase in case of flooding is due to the large number of duplicate packets received at the sinks

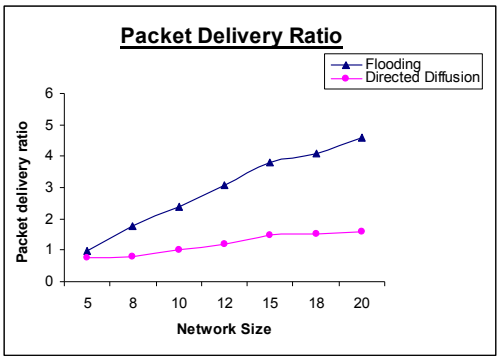


Fig. 2.3.3. Impact of Network size on Packet Delivery Ratio

4) Routing overhead

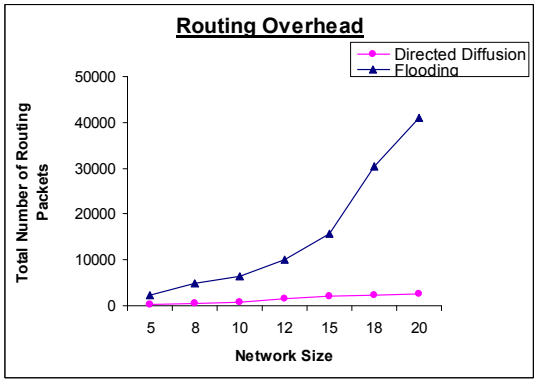


Fig. 2.3.4. Impact of Network size on Routing Overhead

Fig 2.3.4 shows the graph of total number of routing packets transmitted for each network size considered. It can be seen that in flooding, number of routing packets increase at a much higher rate than it does in the case of Directed Diffusion where it increases at a negligible rate.

5) Data delivery cost

The graph shows that the *data-delivery cost* increases with the increase in the number of sinks for both the protocols due to increase in the reception cost of an event for each new sink. However, the data delivery cost of flooding is several orders of magnitude higher than that of directed diffusion. This is because as the number of sinks increase, the cost saving due to in-network processing (e.g., duplicate suppression) of directed diffusion becomes more evident thus increasing the cost at a lower rate than in case of flooding.

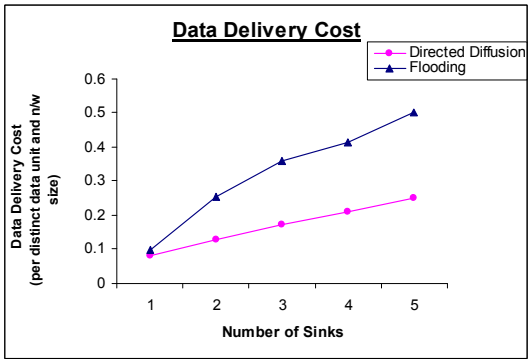


Fig. 2.3.5. Impact of Number of sinks on Data Delivery Cost

6) Throughput

Throughput against Simulation time (50 sec) (5 nodes)

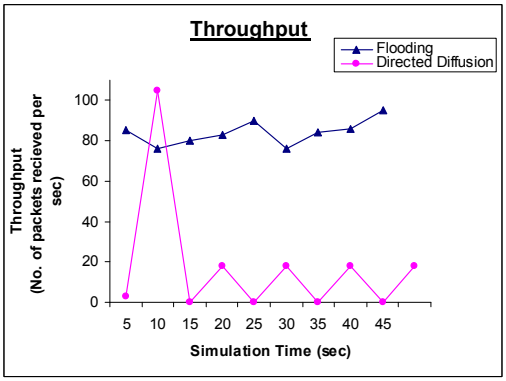


Fig. 2.3.6.1. Throughput against Simulation time (50 sec) (5 nodes)

Throughput of a routing protocol is a measure of number of packets transmitted over a period of time. Fig 2.3.6.1 shows the throughput as the total number of packets received by all the five nodes in 1 sec at intervals of 5 sec for the entire simulation time of 50 sec.

Throughput against Simulation time (50 sec) (20 nodes)

Fig 2.3.6.2 shows the throughput calculated in a sensor field of 20 nodes. The points plotted, give the total number of packets received by all the 20 nodes in 1 sec calculated at intervals of 5 sec for the entire simulation time of 50 sec.In case of Directed diffusion, throughput again increases to a large value in the initial simulation period due to the interest

propagation done by sinks by broadcasting it periodically to its neighbors to finally reach the sources causing a sharp increase in the number of packets received per unit time in the network. It then fluctuates between 0 and a small constant value in equal time intervals when the data is transmitted by the sources to the sinks along the reinforced path after every definite period of time.

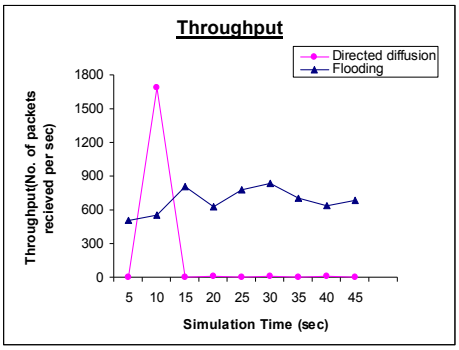


Fig. 2.3.6.2. Throughput against Simulation time (50 sec) (20 nodes)

III. CONCLUSION

In this project we have done an intensive and detailed study on the wireless sensor networks and compared the performances of two of its protocols namely, '*Flooding*' and '*Directed Diffusion*' by implementing them on the NS-2 (Network Simulator-2) platform. We have chosen six metrics to analyze the performances and compare the two protocols. The results of comparison have been presented in the form of graphs. Our analysis of comparison results confirmed that due to its data-centric approach and in-network aggregation, *Directed Diffusion* can more effectively deliver data from multiple sources to multiple, mobile sinks with much lower average delay and routing overhead than in *Flooding*. The large duplication overhead in *Flooding* is a major reason for its poor performance than *Directed Diffusion* where the increase in duplication overhead with increase in the network size, is negligible. Moreover, the packet delivery ratio of *Directed Diffusion* is quite close to the ideal ratio of 1 proving it to have better path reliability than *Flooding* where this ratio increases to a much larger value due to the large number of duplicate packets being received by the nodes. Also, the data delivery cost for *Flooding* is higher than that for *Directed Diffusion*. Thus it has been proved that *Directed Diffusion* can perform much better than the traditional *Flooding* scheme in similar conditions of network size and work load.

REFERENCES

[1] Q. Jiang and D. Manivannan, "Routing Protocols for Sensor Networks", *IEEE*.

[2] C. Intanagonwiwat, R.Govindan, D. Estrin, J. Heidemann and F.Silva "Direct Diffusion for Wireless Sensor Networking" *IEEE?ACM Transactions on Networking*, Vol.11, No.1,February 2003.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks" *IEEE Communications Magazine*, Vol. 40, August 2002.

[4] I. Stojmenovic and X.lin, "Power-Aware Localized Routing In Wireless Network", *IEEE Transactions on Parallel and Distributed System*, Vol. 12 No.10, October 2001.

[5] K. Akkaya and M. Younis, " A survey on Routing Protocols for Wireless Sensor Networks", *Elsevier Ad Hoc Network Journel*, Vol.3, 2005.

[6] M. Maroti, "Directed Flood-Routing Framework for Wireless Sensor Networks".

[7] T. Bokareva, N.Bulusu and S. Jha, "A Performance Comparison of Data Dissemination Protocols for Wireless Sensor Networks", November 2004.

[8] Y.KO, J. Choi and J. Kim, "A New Directional Flooding Protocols For Wireless Sensor Networks", *ICOIN 2004*.

[9] D. Cavin, Y. Sasson and A. Schiper, "On the Accuracy of MANET SIMULATION", *AcM* , 2002.

[10] K.Fall and K. Varadhan, "The ns Manual", 2005.

[11] E. Altman and T. Jimenez, "NS Simulator for Beginners", *Lecture notes*, 2003-2004.