Synchronization of Multiple Levels of Data Fusion in Wireless Sensor Networks

Wei Yuan, Srikanth V. Krishnamurthy, and Satish K. Tripathi
Department of Computer Science and Engineering, University of California, Riverside,
Riverside, CA, 92521

***Abstract*—In wireless sensor networks, in-network data fusion is needed for energy-efficient information flow from a plurality of sensors to a central server or sink. As data (either raw or fused) is propagated towards the sink, multiple levels of data fusion are likely. The data fusion at various levels should be synchronized in order to fuse data effectively. It is important that information from as many sensors as possible to be fused in order to increase the *credibility* of the aggregated report. However, there are trade-offs between fusing a large number of sensor reports and the latency incurred in the aggregation process. The paths taken by the data towards the sink determine where data can be fused, and thus, have an effect on the efficiency of the aggregation process. In this work, we propose a methodology by which the various levels of fusion are synchronized to ensure that the aggregated report has a desired trade-off between credibility and latency, regardless of the topology of the structure created by the integration of the paths on which data traverses towards the sink.**

I. Introduction

Due to rapid advances in technologies of VLSI, RF and embedded processors, the widespread use of wireless sensor networks to obtain physical quantities, such as temperature, pressure, etc., from the environment anywhere and at anytime is becoming the reality. Such a sensor network usually consists of hundreds or thousands of micro sensors with the capability of wireless communications and the ability to perform adequate processing to interpret the sensed data [1] [2]. The information generated by a plurality of sensors that sense a particular event is to be ultimately delivered to a central server or sink. Due to the severe energy constraints, sensor networks normally employ in-network data fusion [3][4]; the power-saving benefit of data fusion has been confirmed theoretically [8] and experimentally [5]. One might envision that at each sensor node at which data (probably already fused to certain extent) is received from multiple other sensors, information is fused to the extent possible. Thus,data will be fused multiple times,in stages, as it is relayed towards the sink.

During multiple levels of data fusion, at each level, multiple data reports are received. There is a possible time lag between the instances of reception of these multiple data reports. As an example, in Fig. 1, node B might receive the report from node C much later than when it receives the report from Node D. Thus, each sensor node has to decide on when to begin the process of fusion and how long to wait before the end of the fusion. Intuitively, the longer the time that the sensor node which performs the fusion waits, the larger the number of reports it will receive. We associate with a fused report a measure of its accuracy and we call this measure *credibility*. In this paper, we simply quantify the credibility of the aggregated report by the number of

This work was partially supported by HRL Laboratories, LLC and the UC Micro.

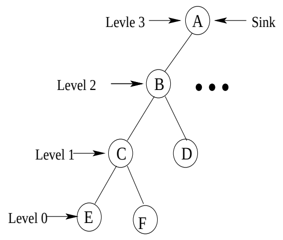

Fig. 1. A Structure to Perform Fusion at Multiple Levels

corroborative individual sensor reports that are fused in the aggregated report, our reasoning being, the higher this number, the higher the accuracy. However, a higher latency may be the cost of the higher credibility.

The process wherein, sensors detect an event, and the data related to the event is eventually fused at the sink via multiple levels of fusion en route, is called a “round” of aggregation. In this paper, we propose a protocol that allows a node to determine when to start and finish the fusion process during a round of aggregation in order to ensure the desired trade-off between the achieved credibility and the incurred latency. We call it the Multi-level Fusion Synchronization (MFS) protocol.

The rest of this paper is organized as follows. In section II, we describe related work on data aggregation in sensor networks. In section III, we describe the system model that we use in our simulation experiments. In section IV, we look at a simple but unrealistic network to understand the impact of various parameters. We describe the proposed protocol in section V. Our simulation results are presented in section VI. Finally, we conclude our work in section VII.

II. Related Work

The focus of our paper is unique and different as compared with previous work on sensor data aggregation. In WINS [2], a node can seek information from nearby sensors for data fusion (the weighted merging of detection decisions) or for coherence beamforming (the complex weighting of raw data from multiple sensors for making improved detection decisions). But the paper does not provide details of how data fusion is implemented.

In [6], a software architecture that supports in-network aggregation in a sensor network is described. The authors show by experiments that aggregation can significantly reduce network traffic. In [8], the authors provide analytical bounds on the energy saving by data aggregation.

In [5], *Directed Diffusion* is presented, in which, locally optimal paths between sources and sink are established. These paths form an aggregation tree rooted at the sink. Whenever similar data meets at a branching node in the tree, the copies of similar data are fused into a single message. To improve the energy-efficiency of this opportunistical aggregation, a greedy incremental tree is proposed by the same authors in [7]. This prior work however does not address issues related to synchronization among the aggregating nodes, nor does it examine the associated credibility of the aggregated content.

III. Sensor Network Model

In this section, we describe the basic framework of the sensor network considered in this paper. A sensor network consists of a large number of wireless micro-sensor nodes which are distributed over a certain area. The area may be divided into a number of regions based on the positioning precision requirements, sensing range of the sensors and other application specific requirements. Each micro-sensor node has at least one sensor, a computation unit and a radio transceiver. There are three circumstances that would cause a sensor node to send a report to the sink. First, sensor nodes periodically send reports to the sink and we call this periodical reporting. Second, the sink queries sensors in specific regions for current sensed information and we refer to this as sink inquiry response reports. Third, due to the occurrence of certain events, reports are triggered from sensors in the particular region in which the event occurs; we call these reports event triggered reports. In this paper, we limit ourselves to event-triggered scenarios; however, note that the schemes that we propose can easily apply to the other scenarios as well, with minor modifications. Depending on the target event (application) and the types of sensors deployed (temperature sensors, pressure sensors, motion sensors, etc.), the way in which data is fused may vary. Data fusion models for various types of sensed data may be found in [9]. The protocol we propose in this paper are independent of the fusion model used; however, to simplify our analyses and simulations, we assume the data that a sensor generates only represents whether or not an event occurs. A fused report would simply contain a count of the number of reports that either confirm or dispute the occurrence of the event[1].

Typically, the sink is distant from the area where the sensor nodes reside. The sensor data has to be ultimately relayed to the sink via multiple sensor node relays. One can then visualize the data being transferred via a structure that facilitates the *many-to-one* data transport. In some sense, building such a structure is akin to building a single source multicast tree, the difference is that instead of data propagating from the single source to the multicast group members, the data flows in the opposite direction, i.e., from the members to the sink. A second difference is that en route, data may be fused at various vantage points on the tree. The topology of this *aggregation tree* determines the efficiency with which data may be fused, to a certain extent. While the discussion of algorithms that help to generate and maintain this tree are beyond the scope of this paper, we show the effects of the topology on the fusion process and how our synchronization protocol can function independent of the topology of the tree. Our only requirement is that each sensor node is aware of its immediate neighbors; specifically it should know its *parent* i.e., the sensor node to which it sends data (either fused or raw) and its *children*, sensors from whom it receives data. Each *non-leaf* node or *internal node* is responsible for relaying (after possibly performing fusion) data received from its children towards the sink node. We assume that the aggregation tree is formed at the network initialization phase, and is dynamically re-organized as sensors sleep, wake up or fail. We note that in the aggregation tree, we refer to nodes as being at particular levels. The leaves are at the lowest level (Level 0) whereas the sink is at the highest level. Furthermore, note that multiple trees may be formed for gathering information from multiple (possibly geographically separate) sensor sets.

We assume a high density of sensor nodes and that multiple sensors detect each event. The *credibility* of the final report at the sink is directly reflected by the total number of reports that are fused and we denote this number by TN.

We restrict ourselves to the occurrence of a single event in this work. Multiple events can be treated individually by using the same method. Note that by either including query identifiers or by associating time-stamps and geographical position with events, one could identify a particular event. If a sensor is unable to fuse data (application layer function), it may still be able to simply concatenate reports to the extent possible to save on header overhead.

IV. A Simple Synchronization Scheme: The Event Triggered Scheme (ETS)

In this section we present a simple scheme that supports synchronization among the aggregation levels in the structure that is formed and call this scheme, the Event Triggered Scheme (ETS). Our objective here is to elucidate how the topology of the aggregation tree affects the multi-level fusion process. A leaf node reports a sensed event to its parent node immediately after it detects the event. At an internal node, the aggregation timer is triggered by one of the following two conditions: (a) the detection of the event (if at all the internal node itself detects it) or (b) the receipt of the first incoming report packet[2] from one of its direct children. The timer on an internal node will expire T seconds after it is triggered; late reports with regards to the same event are discarded[3]; T is the time-out value and can be configured. This scheme needs no global coordination and thus, no signaling is required once the initial structure is in place. We are interested in the impact of the value of T on the performance in terms of the number of fused reports, TN, and the latency incurred during a round of aggregation. The latency is defined as the time-interval from the point when the event occurs to the point when the sink receives the fused content, i.e. the duration of a round. As T increases, we might expect TN to increase as well. But increasing T beyond a certain threshold will not result in further increase of TN (there are only a limited number of reports that are generated). The latency will also increase with T.

[1] In contrary another example of data fusion might involve an attempt to estimate the precise location of a target by extrapolating its distance computations from multiple sensors whose co-ordinates are known. Even here, the more the reports the more precise will be the estimated position of the target.

[2] Note that this report packet may contain either raw or previously fused data.

[3] The node could forward late reports without performing any fusion. This would be ineffi cient. Furthermore, the delayed reports may not be received by the sink in time. Thus, in this work, we simply assume that they are discarded.

In order to understand the effect of the topology of the tree on the performance, we consider the example in Fig. 1. In this example, leaf nodes E, F and D detect an event at time 0. All of them report the event to their parent node at time 0. Let us ignore link delays due to propagation or due to the retransmissions because of packet collisions. Internal nodes B and C do not detect the event. However their timers are triggered at time 0 by the receipt of their children's packets. At time T, both nodes B and C time out, and fuse the data that they have received thus far and propagate the fused data further up the tree. Note that node B will miss the fused data from node C since its timer has already expired before the packet from node C arrives, and it has finished its fusion process. We observe that this effect is due to the imbalance in the tree and the lack of a good mechanism to synchronize the fusion operations at nodes B and C. One might intuitively suspect (as corroborated by simulation results) that a balanced tree structure is better than an unbalanced tree structure. However, it may be difficult to obtain a perfectly balanced tree for arbitrary distributions of sensor nodes. Thus, we would need a mechanism that provides synchronization between the fusion operations at different nodes. One would expect that it is essential for the time-outs at the different levels of the aggregation tree to be different; nodes at higher levels of the tree ought to have longer time-outs than those at the lower levels if the timers for fusion at the different levels were triggered concurrently. Note that the latency is proportional to the depth of the tree; thus a balanced tree might be expected to reduce the overall latency incurred in a round of fusion for a given credibility requirement.

V. The Multi-level Fusion Synchronization Protocol

In this section, We propose the Multi-level Fusion Synchronization (MFS) Protocol to synchronize the fusion processes at different nodes in the tree to achieve the desired trade-offs between the credibility and the latency incurred during a round of aggregation[4].

As seen earlier, if every node upon being triggered waits for the same fixed period of time to collect reports from descendents one would either have to make this waiting time large or suffer from either lost information or a degraded efficiency in fusing data. Nodes at higher levels will have to possibly receive information from a large number of lower levels. Hence, it would be appropriate for these nodes to wait for longer periods as compared to nodes at lower levels. We assume that depending on the application, the sink can choose the right trade-off between the desired credibility and the tolerable latency.

In MFS, first, the sink needs to determine a time interval MAX, for which it would like to wait before it attempts to fuse the received data. The actual latency incurred could potentially be larger than MAX, and in the worst case, could be proportional to the depth of the propagation tree[5]. The sink also needs to choose a parameter Δ, which represents the difference in the waiting periods at consecutive levels. We re-iterate that the leaf nodes are at the lowest level, the parents of these leaf nodes are at Level 1 and so on as shown in Fig. 1. The values of MAX and Δ may be indicated to each node during the set up phase. When a different level of credibility is needed, the sink might propagate new values of MAX and Δ to be used for "the particular round" of aggregation in the region of interest to reach the appropriate set of nodes.

A leaf node reports a sensed event to its parent node immediately after it detects the event. This triggers the timer of its parent node. When an internal node at distance K (in hops) from the sink initiates its timer, it sends out a START message to all its neighbors. If a neighbor is an internal node and its timer is not already triggered, it is triggered by the receipt of this START message. Thus, the timer on an internal node will be triggered by any of the three following events, (a) the detection of the event by the internal node, (b) the receipt of the first incoming report from one of its direct children, or (c) the receipt of a START message from a neighbor. The timer, thus triggered, will expire (MAX - K*Δ) seconds later. Upon the expiry of the timer, data received will be aggregated and passed further up in the tree. As mentioned earlier, we assume that late reports with regards to the event are simply discarded. There are some internal nodes whose timers are unnecessarily triggered. After the timer expires, they will have received no report from their children, nor will they have detected the event themselves. These nodes do not further participate in the aggregation process.

Note that the START messages may collide with each other or with other packets since they are simply broadcast. Thus, some of them may be lost. In the best case, there are no collisions and all internal nodes that are required to participate in data fusion and transport, towards the sink node, trigger their aggregation timers almost at the same time that the relevant event occurs (ignoring propagation delays). The latency then is approximately equal to MAX. In the worst case, none of the pertinent internal nodes receive the START message (due to collisions), and in this case the latency could be

$$L = \sum_{j=0}^{D-1} (MAX - j * \Delta) = D * MAX - \frac{(D-1)D}{2}\Delta,$$

where D is the depth of the propagation tree (in hops). If we ignore propagation delays, this is also the upper bound on the latency incurred during a round of aggregation with the MFS protocol.

It is to be noted that if $MAX < (D-1) * \Delta$, reports from sensors below a certain level may not reach the sink. In such a case, the fused content may not have the requisite credibility. The sink node will then have to choose new values for MAX and Δ, and query the network for a more credible response. If the sink knows the depth of the propagation structure, it should be easy for it to determine the values of MAX and Δ; otherwise a learning phase may be needed for the sink to find the reasonable values of MAX and Δ. In the learning phase, the sink queries the region with different values of MAX and Δ and adjusts these values based on the reports' credibility and the application requirements.

[4]Note that we do not need absolute clock synchronization between the sensor nodes. Needed is a loose event level synchronization between the fusion operations.

[5]We will compute an upper bound for the latency later.

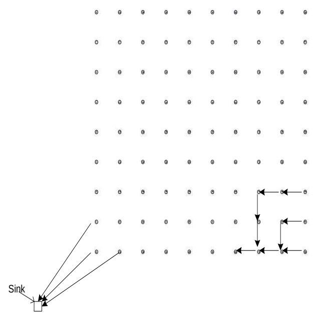

Fig. 2. Node Distribution for ETS

VI. Simulation and Results

We implemented ETS and MFS in ns-2 [12]. For our simulations we make use of the CMU Monarch group's mobility extensions [12]. The existing implementation of the IEEE 802.11 [13] MAC layer protocol is used. Our first objective is to demonstrate the effect of the topology on the efficiency of fusion. Towards this we use two protocols that can construct a tree rooted at the sink: (a) The Breadth First Search (BFS) algorithm [11] which creates a tree that is balanced to the extent feasible and (b) The On-Demand Multicast Routing Protocol (ODMRP) [10], a protocol proposed for creating source based multicast trees; the tree created by ODMRP could potentially be heavily unbalanced. We wish to then examine the correlation between the parameters MAX and Δ and evaluate the performance of our protocols when the tree is (i) balanced and (ii) unbalanced.

In the simulation of ETS, the nodes are distributed as shown in Fig. 2, in accordance to a 10*10 grid. The radio range of each node is such that its transmission can reach only its eight geographically nearest neighbors. The sink resides at the lower left corner; the nodes on the upper and the right boundaries are the only nodes that will detect the event of interest. Thus, there are totally 19 *raw* reports for the event. We first used a modified version of ODMRP to create the propagation tree. The modification we made was to turn off the periodic refreshment of the tree as required for multicast operation and maintenance. Due to the randomness associated with the MAC layer access, each time we ran ODMRP, it created a different tree. So we ran simulations over 300 different ODMRP trees and for each tree, we generated 200 events. Each event is detected by the 19 sensors as specified earlier, and the reports generated by these sensors are propagated towards the sink. At each intermediate node, data is fused and the fused message is propagated further. We take the average over the 60000 events. To see the impact of the structure of the tree, we used the BFS method to create a well-balanced tree, and performed the same simulation.

From Fig. 3, we can see that as T increases, the total number of responses included in the aggregation (TN) also increases, which is as expected. BFS creates a better-balanced tree and outperforms ODMRP most of the time. Thus, one might conclude that it is better to create a well balanced structure along which responses may be aggregated. Note that doing so also reduces bottlenecks of MAC layer contention. When ODMRP is used, some of the reports actually get lost due to such bottlenecks and this translates to the lower value for TN seen in Fig. 3. Fig. 4 shows that the latency is proportional to T, which also agrees with our analysis. Furthermore, notice from Fig. 3 and 4 that for a given latency, the sink can fuse more individual responses when using a propagation structure created by BFS than by ODMRP.

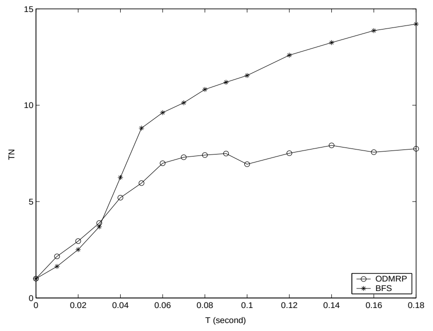

Fig. 3. Total number of fused raw sensor reports (TN) in ETS

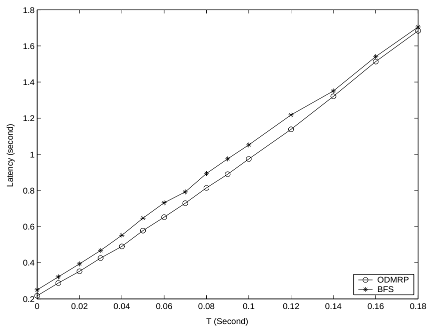

Fig. 4. Latency in ETS

We next simulated the MFS protocol and examined its performance with the ODMRP and the BFS trees. We randomly distributed one hundred nodes over a 1000m*1000m square area, as shown in Fig. 5. The sink resides at the lower left corner as shown in the figure. The radio range of each node is 250m. The eighteen sensor nodes that are closest to the upper right corner generate *raw* reports. The depth of the tree generated by BFS is 7. We set MAX to be 1.2 seconds[6].

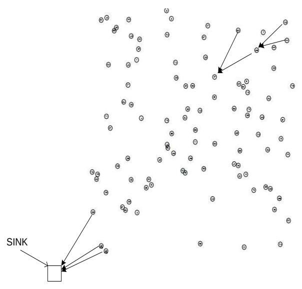

Fig. 5. The distribution of nodes for our simulation experiments

Fig. 6 shows that the MFS protocol works fine on both the balanced tree structures generated by BFS and the possibly unbalanced tree structures generated by ODMRP. We also can see

[6]Note that this is a system parameter that may be used to set a desired upper-bound on the latency incurred during a round of aggregation.

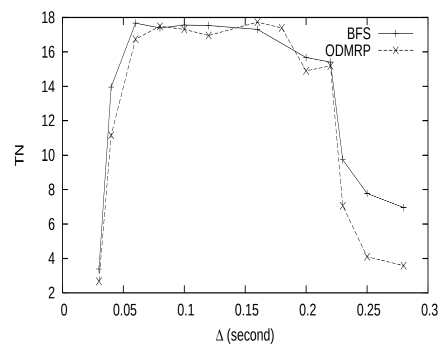

Fig. 6. Total Number of Nodes that respond (TN) in MFS

that as Δ increases, TN increases rapidly until a certain stage, when the sink gets an aggregated report that takes almost all the individual raw sensor reports into account. However if Δ increases further beyond a certain second stage, TN starts decreasing. If Δ is very small, then the time-out at each level is almost equal to the time-out at any other level ($\approx MAX$). Since the tree may be unbalanced, as mentioned earlier, this choice of equal time-outs at every level leads to a loss of information. Thus, the credibility of the aggregated report is low. Furthermore, notice that we incur a large latency (Fig. 7) since, now, the waiting period is large even at lower levels. On the other hand, if Δ is large, nodes at a given level do not wait long enough to collect information from all the children at the preceding lower level. Thus, even though we observe a low latency, the associated credibility in terms of the number of responses (TN) collected is low. We observe that the drop-offs in terms of credibility at the two extremes are fairly steep. Thus, by increasing Δ by a little less than (MAX/D) (in this case, 0.17 seconds), we can obtain a good credibility within a short period of time. The fairly flat behavior of the credibility between the steep drop-offs demonstrates that the credibility is not very sensitive to the value of Δ in this region. In fig. 7, we also note that in the tree generated by BFS, the latencies incurred for almost all values of Δ are less than 2*MAX. And since the randomly generated ODMRP trees are typically unbalanced as compared to the tree generated by BFS, they may have a larger depth, and hence, a larger latency is incurred.

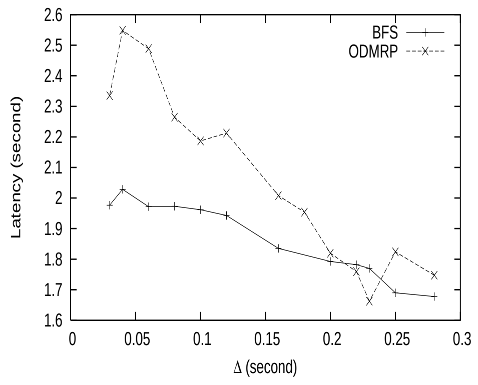

Fig. 7. Latency in MFS

VII. Conclusions

In this paper, we propose a protocol (Multi-level Fusion Synchronization or MFS protocol) that provides synchronization between multiple levels of data fusion in wireless ad hoc sensor networks. In such networks, typically raw data from a plurality of sensors is to be transported to a central operation center or sink. En route, data is fused multiple times to reduce the amount of communication overhead in terms of bandwidth and power. The flow of data from the sensors to the sink is akin to the flow of data along a tree, from the leaves and/or intermediate nodes, to the root of the tree. At each non-leaf node on the tree data may be fused. It is important to synchronize the fusion operations at various nodes in order to ensure that as many raw reports as possible are fused together within some latency constraint. The topology of the tree (whether it is balanced or unbalanced) has an effect on how effectively data may be fused. We show by means of simulations that our method provides effective synchronization between the various fusion levels and that a high credibility can be associated with the final fused report irrespective of whether the tree is balanced or unbalanced.

References

[1] D. Estrin, L. Girod, G.Pottie and M.Srivastava, "Instrumenting the World with Wireless Sensor Networks." In *Proceedings of ICASSP 2001*, May, 2001.

[2] G.J. Pottie, and W.J. Kaiser, "Wireless Integrated Network Sensors." *Communications of the ACM*, Vol. 43, No. 5, May 2000.

[3] G.J. Pottie, "Hierarchical Information Processing in Distributed Sensor Networks." In *Proceedings of the IEEE International Symposium on Information Theory, 1998*, pp.163, 1998.

[4] W.R. Heinzelman, A.C. Chandrakasan and H. Balakrishnan, "Energy Effi cient Communication Protocol for Wireless Microsensor Network." In *Proceedings of the IEEE Hawaii International Conference on System Sciences*, Jan, 2000.

[5] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks." In *Proceedings of the sixth annual international conference on Mobile computing and networking 2000*, Aug, 2000.

[6] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan "Building Effi cient wireless sensor network with low-level naming." In *Proceedings of the ACM Symposium on Operation System Principles*, Oct, 2001.

[7] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks." In *Proceedings of International Conference on Distributed Computing Systems*, July, 2002.

[8] B. Krishnamachari, D. Estrin and S. Wicker, "Impact of Data Aggregation in Wireless Sensor Networks." In *International Workshop on Distributed Event-Based Systems*, July, 2002.

[9] D.L.Hall and J.Llinas, *Handbook of Multisensor Data Fusion*, CRC Press, 2001.

[10] S.H. Bae, S.J. Lee, W. Su and M. Gerla, "The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocol in Multihop Wireless Networks.", *IEEE Network.*, Vol.14, Issue.1, Jan/Feb, 2000

[11] T.H. Cormen, C.E. Leiserson and R.L. Rivest, "Introduction to Algorithms.", *The MIT Press.* pp.469-477, Cambridge, Massachussatts, 1990.

[12] The Network Simulator - ns-2 "http://www.isi.edu/nsnam/ns/".

[13] "IEEE 802.11 Standard for Wireless LAN: Medium Access Control (MAC) and Physical Layer (PHY) Specifi cation.", New York, Approved on 26 June, 1997.