Reliable Data Transmission in Event-based Sensor Networks During Overload Situation

Charalambos Sergiou Vasos Vassiliou Andreas Pitsillides
Networks Research Laboratory
Department of Computer Science
University of Cyprus
75 Kallipoleos Str, 1678 Nicosia, Cyprus
{cspgsc2, vasosv, cspitsil} @ cs.ucy.ac.cy

ABSTRACT
In this paper, we describe a scalable and distributed framework for minimizing congestion and assuring reliable data transmissions in *event based networks*. Event based networks are a particular category of sensor networks on which reports are produced only upon the observation of a specific event. This event should satisfy a pre-specified condition. Whenever this condition is satisfied, a sudden traffic increase occurs which may lead the network into congestion. This is particularly undesirable because the data generated during this situation are of great importance, often critical, to the applications. We propose a novel algorithm which is able to control a congestion situation and which is efficient enough to safely transmit almost all the data, generated by the sensors due to an event, back to the sinks. The algorithm does that without throttling the source nodes' data rate. Throttling the data rate could prove fatal for critical networks, due to the fact that each data packet provides the network with updated information concerning the monitored event.

Categories and Subject Descriptors
C.2.1 **[Network Architecture and Design]**: Wireless Communication, Distributed Networks

General Terms
Algorithms, Performance

Keywords
Sensor Networks, Distributed Algorithms, Congestion Control, Overload Control, Hierarchical Tree, Alternative Path

1. INTRODUCTION
Wireless sensor networks (WSN) are wireless networks consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion, or pollutants, at different locations [1].

A special category of WSNs are the event-based networks. In event-based networks data packets are produced only upon the observation of a specific event that satisfies a pre-specified condition. In event-based environments there is a need for controlling the sudden traffic increase. Due to the nature of these environments, sudden traffic increase occurs when the monitoring event is happening. This high generation of data packets is usually uncontrolled and often leads to congestion.

When congestion occurs, the network may enter into an unstable state. In this state the networks' behavior is unpredictable. If there is no congestion control mechanism the network's reaction to congestion is the random drop of data packets. Besides the obvious energy consumption, the major drawback in this method is that the packets, which are produced during this state, are of great importance. So, the need for early congestion prediction and alleviation is obligatory.

In different studies [2,3] it is observed that the number of nodes with occupied queues grows if congestion gets worse. When congestion is detected, the sources should be notified in order to take action to face congestion. The most popular approach for this notification is the transmission of a control packet to the source, from the sink.

Congestion control approaches in WSNs [4,5,6,7,8] try to react in congestion with rate limiting techniques. Throttling the data rate in event-based WSNs is not acceptable due to the fact that the data packets which are produced during the monitoring of the event are of great importance and almost all of them need to be forwarded to the sink. An example of the use of these networks is the case of a fire in a forest. Data packets are forwarded to the sink to keep the fire stations updated for the fire's frontage

In this paper we present a scalable and distributed framework for assuring safe and reliable transmission of data packets to the sinks during an overload situation without reducing the sources' data rate. Our framework consists of the following algorithms:

(i) Hierarchical Flooding which is used initially for the network discovery and the placement of nodes in levels,

(ii) Hierarchical Tree Alternative Path (HTAP) algorithm in order to deal with the expected congestion situation and to safely forward the data packets to sinks, and

(iii) Handling of powerless (dead) nodes.

WICON 2007, October 22-24, Austin, USA
Copyright © 2007 978-963-9799-12-7
DOI 10.4108/pwsn.2007.2280

2. RELATED WORK

Wan et al. propose CODA [6], a congestion control system for sensor networks. CODA detects congestion by periodically sampling the channel load and comparing the fraction of time that the channel is busy to the optimal channel utilization. The system responds to congestion with a combination of hop-by-hop flow control and closed-loop regulation. In hop-by-hop flow control the node experiencing congestion broadcast backpressure messages upstream toward the source nodes informing them for the need to reduce their data rates. In closed- loop, acknowledgments (ACKs) are by the sinks in order to inform sources that their sending rates exceeded a predetermined threshold. In this case source reduces their sending rate.

Woo and Culler propose a rate control mechanism [7] that admits traffic into the network using an AIMD controller. When a node hears that a packet it had previously sent was forwarded, it additively increases its transmission rate. When it does not hear a previous transmission being successfully forwarded (presumably after a timeout), it multiplicatively reduces its transmission rate.

Sankarasubramaniam et al. propose ESRT [8], the Event to Sink Reliable Transport Protocol. Their system addresses congestion control in the context of reliable delivery. The sink uses congestion feedback from sensor nodes to broadcast a notification to reduce reporting rate. Feedback latency is dependent on the network's size and may not scale in very large sensor networks.

3. SCHEME DESIGN

Bearing in mind the related work and the unique features of WSN, especially the limited power and storage resources, a new algorithm is proposed attempting to solve the congestion problem in these networks. For the development of this algorithm we involve a major input, which exists in sensor networks. This is the plethora of unused recourses.

The algorithm consists of two parts, the Alternative Path Creation (APC) and Hierarchical Tree Creation (HTC). The philosophy of these two algorithms is similar. Both of them are based on the creation of alternative paths from the source to the sink, when congestion is going to take place. For the creation of these paths, nodes which are not used at that moment start being utilized. APC uses these nodes in a generally random way, compared to HTC where these nodes are placed in a Hierarchical tree from the source to the sink. The final algorithm is called HTAP (Hierarchical Tree Alternative Path) and is a combination of these two algorithms.

The HTAP (Hierarchical Tree Alternative Path) algorithm attempts to solve a congestion situation locally "by- passing" the congested node through the creation of alternative paths form the source to the sink. Initial simulation results show that the HTAP algorithm can cope with congestion and maintain the reliability of data packets transmission to the sink. In addition it achieves good performance in terms of energy dissipation, latency and transmission efficiency

3.1 Alternative Path Creation (APC)

The initial idea for the creation of this algorithm derived from a particular concept of the theory of Dynamic Alternative Routing (DAR) used in public telephony [9]. In this concept it is stated that if you have a good route without problems, stick to it, until something goes wrong with it. With some major changes in the implementation, this concept is adopted by us in the case of congestion control in WSNs.

According to [10,11] there are many nodes in wireless sensor networks, which, when a specific event is detected, are not taking part in the path from the source to the sink. This is due to the fact that these nodes are far away from the event. The main target of the APC algorithm is to take advantage of these nodes and use them for the creation of alternative paths from the source to the sink. The creation of these paths unloads the highly dense parts of the network and lead the data packets safely to a sink though other routes.

The basic theory of this algorithm is that a source node keeps transmitting data packets to a specific node at a level higher than itself, until it receives a control message from this node that is not able to handle any more packets. This is either because the downstream node is going to become congested or due to the fact that it will soon run out of power. In such a case, the source node will search in its neighbor table and find the most appropriate node to further transmit the data. To explain the concept let us consider Figure 1.

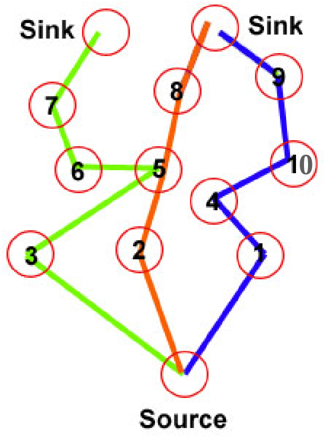

Fig. 1 Transmission from Source to Sinks

We consider that there is a source which is sending data packets. Initially, it is sending data to the node numbered 1. The data packets follow the right path to a sink. As soon as node 1 calculates that is going to be congested it sends a control packet to the source in order to inform it about the fact. When the source node receives the control packet it searches in its neighbor table, which includes nodes 1, 2 and 3, to find the proper node to keep sending data (how the neighbor table is created and which is the proper node is explained in the algorithm). The sending node may then choose to transmit through nodes 2 or 3 (middle or left path).

It must be stated that this procedure is taking place at all nodes and not only at the source. For example, if node 8 becomes congested, it will send a control message to the previous node (in this example is 5) to inform it about the fact and node 5 will apply the same procedure. It will continue to forward data through node 6 instead of 8.

3.1.1 APC implementation

The implementation philosophy of the algorithm follows the steps below:

- A simple hierarchical flooding protocol is used for the formation of the network's topology. Through this procedure, each node discovers its neighbor nodes and updates its neighbor table. In addition, through this protocol, sensor nodes are theoretically placed in levels from the source to the sink.

- At each packet transmission each node piggybacks its congestion state (buffer occupancy). The neighbor nodes overhear the packet transmission [6] and update their neighbor tables with this information.

- During the triggering of an event, the source node begins transmitting data packets creating flows to the sink. If the sending data rate is higher than the rate that the receiving node can transmit, the receiving node will soon face a buffer congestion situation and the results would probably be the random drop of data packets. In order to avoid this situation each candidate congested receiver is sending a backpressure packet to the sender to inform it that if it continues to transmit packets with the same rate it will soon be congested. This way the sender stops the transmission of packets to the candidate congested receiver and searches in its neighbor table to find the least congested receiver in order to continue the transmission of data.

- The transmitting node begins transmitting the data to the alternative node. The same phenomenon can happen at any level (between the neighbor nodes). The change of receivers leads to the creation of alternative paths.

3.2 Hierarchical Tree Creation (HTC)

This algorithm consists of two main steps:

- Route Creation: In this step a hierarchical tree is created beginning at the source node. Each node is assigned a level according to the hierarchical tree. The source node is assigned a level 0 and broadcasts a *level_discovery* packet. Sensors that receive this packet are handed as children to the transmitter and are set as level 1 (they will ignore subsequent *level_discovery* packets). Each of these nodes broadcasts a *level_discovery* packet, and the pattern continues with the level 2 nodes etc. The source when it receives the *level_discovery* packet updates its neighbor table.
- Flow Creation: Connection is established between each transmitter and receiver using a 2-way handshake. Packets are exchanged between each transmitter and receiver in the network, in order to get connected. Through this packet exchange, the congestion state of each receiver is communicated to the transmitter. This connection is performed using a 2-way handshake. Having a source node A and a receiver B, node A sends a first packet to B. When node B receives this packet, it sends an ack packet back to A. In this ack packet the node B piggybacks the congestion state at the moment. In this way, the source node is aware of the congestion state of all the children and is also able to forward them data packets. When the congestion state of children changes to a pre-specified limit this node updates its congestion state by sending a packet to the source node.

The congestion state limit is calculated as follows:
T_{PR} is the propagation time for the transmission of a packet from source A to node B (level 1).
K packets/sec is the transmission rate of node A.
In T_{PR} sec is possible to exist $b=T_{PR} \times K$ packets on fly.

Based on the above, when node's B buffer plus b equals full then node B is sending `Buffer_Full` to node A.

3.3 APC and HTC issues

We aim to evaluate the effectiveness of the Alternative Path Creation (APC) algorithm as a technique to alleviate the congestion problem in WSNs for event triggered networks.

The major advantage we anticipate is the fact that it utilizes nodes which in other ways would be in a dormant state. Basically, it increases the resource provisioning in the network with the energy and buffer capacity of these nodes.

This solution seems to create good results concerning congestion alleviation but its problem is that it scales poorly with dense networks where there are many nodes. More specifically the problem has to do with the mean time of the transmission of packets from the sources to the sink, which in some cases is highly increasing (Fig 2). This means that a packet could be received "late" causing problems in the accomplishment of the mission.

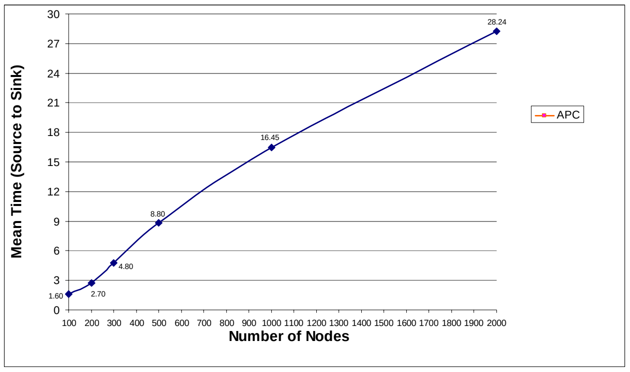

Fig. 2 APC's Mean Time (Source to Sink) vs Number of Nodes

Each node is transmitting event data packets without a perception about the congestion situation in other fields on the network, except of the level above. The lack of knowledge of the rest of the network may lead the packets through a longer path to the destination. In any case packets avoid hotspots due to the fact that they always choose the least congested node). Figure 3 explains this situation.

In Figure 3 node 1 (upper left) is the source and node 6 (lower right) is the sink. Node 38 is just one hop away from the sink. If, for example node 38 becomes congested, packets from node 30 can be forwarded to sink either through node 32 (the best case, one hop) or following the path 30, 9, 22, 36, 39, 32 following 6 hops. If exactly after the forwarding of packet to node 30, node 38 is relieved from congestion, the next packet which carries the latest information will be received by the sink in one hop, and the previous packet will become a "stale" packet.

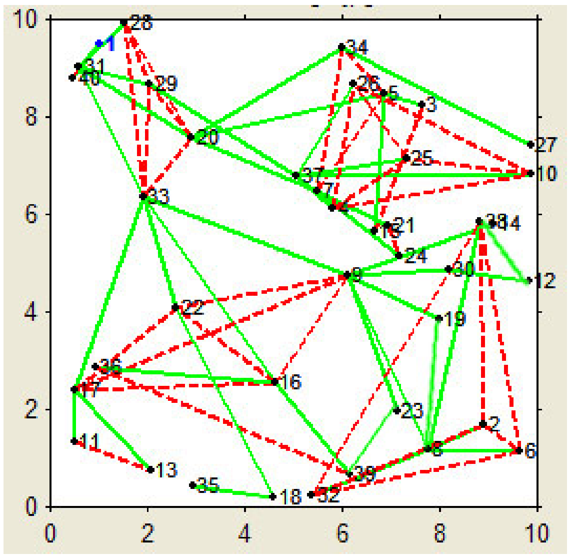
Fig. 3 APC snapshot

The Hierarchical Tree Creation (HTC) algorithm aims to solve this problem. In this case, the packets follow the hierarchical tree which is created at the network deployment. Through the nodes above it each node has perception of the whole network through the routes that have been created in the route discovery phase. Each packet follows specific flows in order to reach the sink.

When congestion is going to happen at a specific receiver node, this node sends a control packet to the transmitter node to inform it to change destination node. The transmitter node searches in its neighbor table and finds the most appropriate node, towards which it begins the transmission of data. The procedure is exactly the same as in the APC algorithm. The differences are that according to this algorithm the routes from the source to the sink, pre-exist and are followed by the packets.

However, as in APC so in this algorithm, a main disadvantage exist. This is the energy consumption compared with APC algorithm (Fig 4). The 2-way handshake requires each node to receive a packet and send one in response. For a source node A with children B and C, node A broadcasts level discovery packet and then connection packet. Nodes B and C receive both packets and then transmit an **ack** packet to node A, piggybacking also their congestion state. This example uses 4 transmissions and 4 receptions.

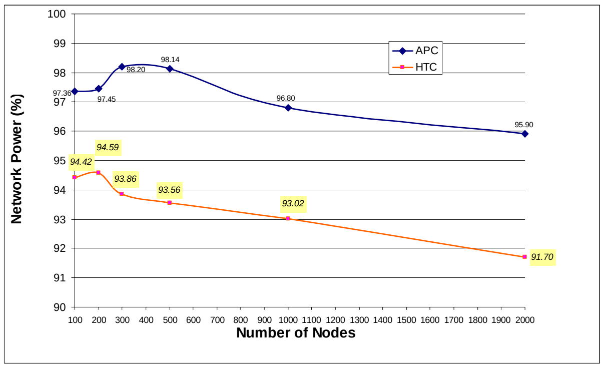

Fig. 4 Network's Power vs Number of Nodes for APC and HTC

3.4 Hierarchical Tree Alternative Path (HTAP) algorithm

The HTAP algorithm is a combination of the two previously mentioned algorithms. Due to the fact that the advantage of the first algorithm is the disadvantage of the second algorithm and vice versa, this algorithm embodies the advantages of both algorithms in order to eliminate the disadvantages.

This means that in this algorithm the APC is going to be applied in combination with the HTC when the network is densely deployed. The density or the sparseness of a network is not an abstract term. A specific threshold needs to be specified by which each node should be able to decide when to apply APC by itself or in combination with HTC. Many deployment scenarios envision that sensor nodes are dropped by airplanes or fill a place in a field etc. This means that some nodes may not become part of the network, due the fact that they fall too far away from other nodes and there is no other node in their transmission range. Some other nodes could be very densely deployed and other could be very sparse. This threshold results out of the number of nodes that a specific node has in its neighbor table. If the number is below the pre-specified (for the specific network) threshold the nodes apply the APC algorithm by itself. Otherwise, they apply it with the HTC algorithm. Threshold values can be extracted through simulations.

3.5 Powerless (dead) nodes

Special care should be taken in the HTAC algorithm concerning the nodes with exhausted batteries. These nodes are causing major problems to the network, especially when they are source nodes. Thus, when a node is going to lose its power, it should immediately be extracted from the network and the tables of their neighbor nodes should be updated. This procedure should be as simple as possible due to the fact that this can happen when the network is in a crisis state.

This algorithm deals with two cases. The first case is when the "dead" node is the source node and the other case is when the "dead" node is a child node. In the first case, when the remaining source's node power is diminished, the source node broadcast an *elect_packet* to its neighbor nodes. The neighbor nodes communicate their power levels with each other and the one with the most remaining energy is elected as the new node. Power is diminished, and the other nodes remove it from their neighbor tables.

4. DESCRIPTION OF ALGORITHMS

This section describes the four previous mentioned algorithms. These algorithms are activated when a congestion situation is about to appear in the network.

Moreover in this section the flooding algorithm is described. The flooding algorithm is used at the first deployment of the network in order for each node to discover its neighbor node and to update their network tables. In order to "assist" the Hierarchical Tree Algorithm a *level_discovery* functionality is also added to this algorithm

Flooding Algorithm with Level Discovery

```
set neighbor_nodes to 0
if current_node is source node
    Set level to 0
    Broadcast flood_packets with level
else if current_node receives flood_packets and is accepting them
    set current_node to level+1
    send ack_packet with current_node_id
    broadcast flood_packet with current_node_id and level
    ignore subsequent flood_packets
else if current_node receives ack_packet
    neighbor nodes+1
```

The APC algorithm is described next. After the application of the flooding algorithm, each node is aware of its neighbor nodes. As it was mentioned in the analysis of APC algorithm, the nodes are also aware of the congestion state of their neighbor nodes and are update their neighbor tables, by overhearing the transmitted packets of the other nodes, in which their congestion state is piggybacked.

APC: Alternative Path Creation Algorithm

```
if current_node receives ack with congestion_level full
    update neighbor_table
    search neighbor_table
    find node_id with min (congestion_level)
    send data packet
if current_node receives congestion_update_message
    update neighbor_table
else if current_node receives data packet and accepting them
    Set buffer to buffer+1
    if buffer+b=full //from Section 3.2
    send ack packet with congestion_level full
```

Figure 5 shows an execution of APC algorithm. In Fig 5 data are produced from a source node next to node 1 and are forwarded to sinks 6 and 49. In our simulations we use more than one sink. The same situation, concerning the neighbor table, applies for the Hierarchical Tree algorithm. Here, the differences are related with level discovery and the connection- oriented situation, in order to form the hierarchical tree. Flooding algorithm assists in level creation as it was described before.

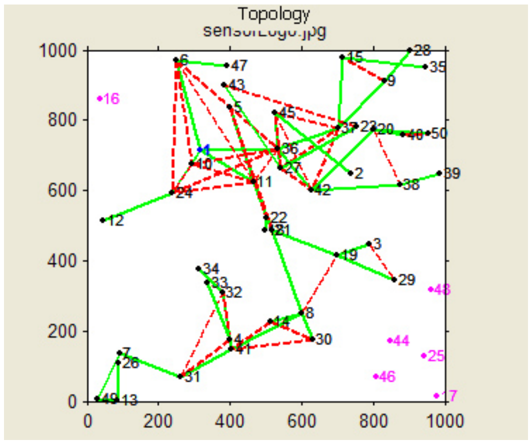

Fig. 5 APC snapshot

HTC: Hierarchical Tree Creation Algorithm

```
if current_node receives hello_message
send ack_hello
else if current_node sends hello_message
wait specific_time
    if current_node receives ack_hello
        update neighbor_table
    else if time_expires
re-send hello_message
/** IN CASE OF CONGESTION APPLY APC **/
```

The combination of the two algorithms to create the Hierarchical Tree Alternative Path (HTAP) algorithm is described below. As it was described before, when the neighbor nodes of a specific node is below a specified threshold the APC algorithm applies, the HT applies otherwise.

HTAP: Hierarchical Tree Alternative Path Algorithm

```
Set neighbor_nodes_threshold to [prespecified value]
    if neighbor_nodes< neighbor_nodes_threshold
            apply APC
    else
            apply HTC
```

Figure 6 shows an execution of HTAP algorithm

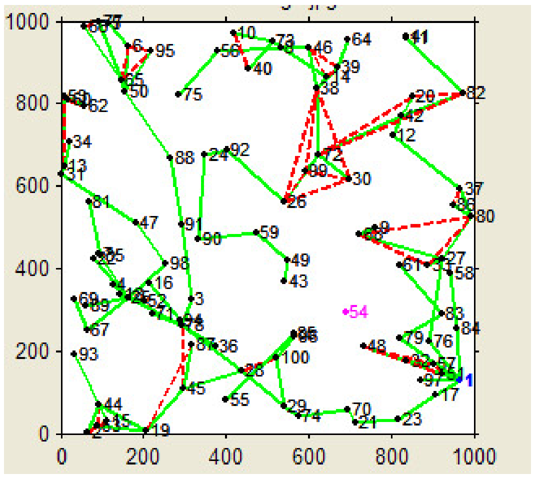

Fig. 6 HTAP snapshot

5. EVALUATION

To evaluate the proposed algorithm, scenarios were created to compare various network and nodes parameters. The proposed algorithm has been implemented in MATLAB, as a part of a specific simulator for Congestion Control in WSN.

5.1 Simulation Environment

In all simulation environments and scenarios we choose to randomly deploy nodes in a rectangular grid. The grid size is 1000m x 1000m a commonly used grid for modeling densely deployed networks [12]. In each run, the parameters in Table 1 were kept stable while increasing the number of nodes in the grid to make a dense network with strong connectivity. In order to trigger off a congestion situation, all the nodes in the network are almost congested, meaning that their buffer occupancy in near to 90%.

Table 1. Simulation Parameters

X distance (m)	**1000**
Y distance (m)	**1000**
Transmit Power (dBm)	**-2**
Sensitivity Threshold (dBm)	**-81**
Path Loss Coefficient	**3.5**
Node CPU (MHz)	**4**
Radio Freq. (MHz)	**433**
Buffer Occupancy (%)	**85**
Data packet	**1 packet**
Control Packet	**¼ packet**

In addition we introduce more than one sink. Each sink is able to handle a big amount of data. Each time the number of nodes in the network is increasing the sources produces proportional number of data packets.

5.2 Scenarios Analysis and Results

In order to get some reference results, the first simulation series run has been conducted with no congestion control algorithm. Basically, in this case the nodes were placed in a hierarchical tree (flooding algorithm) but no congestion prevention measures were taken when congestion happened (e.g no retransmissions). The only "extra" provision was the handling of "dead" (powerless) nodes.

The first parameter that has been investigated is the networks' energy consumption during a crisis state. As it was stated in Section 3.3, during a crisis state there is a sudden traffic increase which will certainly lead to congestion if no countermeasures are taken. This specific metric, simply adds the remaining power of all nodes in the network and divides it by the number of nodes**.**

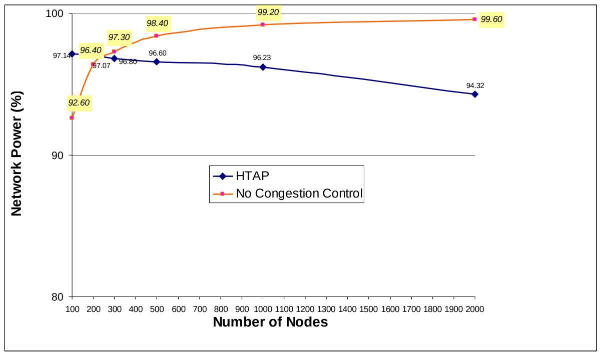

Fig. 7 Network power versus Number of Nodes

At first glance Figure 7 seems to give some "unexpected" results: as the number of nodes in the network is increasing, "no congestion control" algorithm appears to save more power. Of course, this is not true. The reason can easily be explained.

HTAP's main target is the involvement in congestion alleviation phase, a number of nodes which are in dormant state. These nodes will create alternative paths to the sinks. Bearing in mind that the network's dimensions are constant and the numbers of nodes are increasing, the more nodes are in the network, the more of them are involved in the congestion alleviation phase. This means that packets traverse to the sink through a higher number of hops, leading to energy consumption from many nodes.

On the other hand, when the "no congestion control" algorithm applies, the situation is clear. In this procedure is involved a specific number of nodes (only a path from the source to the sink). These nodes are, like all the nodes in the network, are almost congested. As long as they receive packets that are not able to handle, they drop them. Furthermore, the control packets that they exchange are limited. This situation will lead to extinction of a specific number of nodes, while the rest will have their power in full.

This leads to a subjective energy saving. Taking into account that the metric we examine, derives from the division of the remaining nodes' power by the number of nodes, it is easy to understand that as the number of nodes in the network is increasing, more nodes are in dormant state and of course more nodes have their power untouched, leading to increment of the overall power.

The next parameter that was studied was the number of packet drops during a crisis state. Keeping the parameters same as above, the following results were noted.

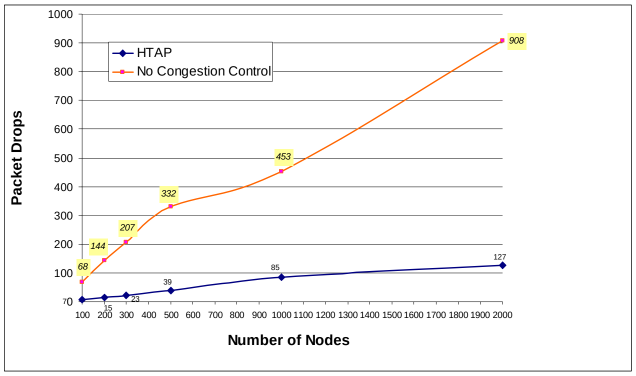

Fig. 8 Packet Drops versus Number of Nodes

As it is presented in Figure 8, the number of packets drops with HTAP, increase slightly as the number of nodes is rising. Yet, in all cases, the packet losses are up to ten times lower compared with the "no congestion control" algorithm. This fact is a good indication that the HTAP algorithm is able to minimize the congestion problem and minimize the packet loss. The problem concerning the "no congestion control" algorithm can become even worse, bearing in mind that in this simulation series, retransmissions have been disabled, and the "dead" nodes provision is active.

The following item of interest is the percentage of dropped packets, compared to transmitted packets. Figure 9 illustrates these results.

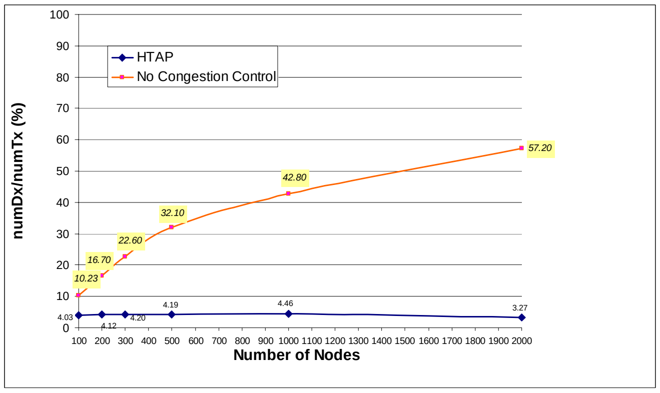

Fig. 9. numDx/numTx (%) versus Number of Nodes

The percentage of dropped packets to transmitted packets is generally stable meaning that HTAP alleviates congestion even under a big number of nodes (i.e a big load). On the other hand, if the "no congestion control" algorithm is applied, the percentage of lost packets is increasing as the number of nodes in the network is increasing.

The next parameter that has been simulated was the mean travel time of packets from the source to the sink. This parameter has not been compared with "no congestion algorithm", due to the fact that the results of "no congestion control" algorithm are not reliable in this case. The reason is that this metric measures the time of all received packets to the sink and calculates the mean time. Bearing in mind that a big number of packets with "no congestion control" algorithm are dropped, the results could not be true.

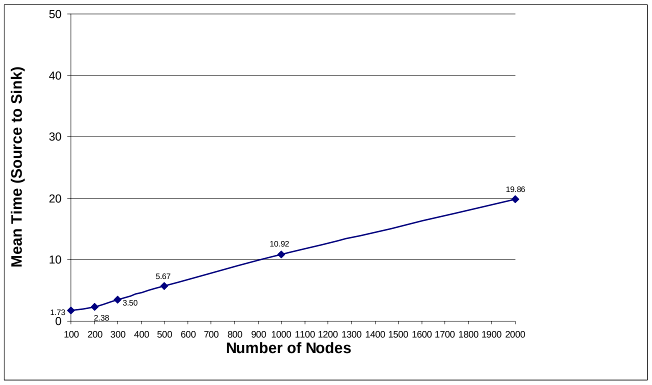

Fig. 10 Mean Time (Source to Sink) versus Number of Nodes

Basically, this metric shows the trend of mean time while the number of nodes in the network is increasing. As it is depicted in Figure 10 the time is slightly increasing as the number of nodes in the network is increasing. This is normal having in mind that the creation of alternative paths creates many hops between the source and the sink. This can be considered as a disadvantage of the proposed algorithm since the time is sometimes a critical factor for the success of the application.

6. Conclusions

In this paper we investigated the congestion problem that takes place in WSN. In our work we dealt with a specific category of WSNs, the event-based networks, which produce data only when the monitored event exceeds a pre-specified threshold

We propose a novel algorithm HTAP (Hierarchical Tree Alternative Tree), which takes advantage of unused nodes, involving in the congestion alleviation phase their resources, power and storage (buffer space).

The HTAP algorithm has been simulated in Matlab and has been compared to a situation without congestion control mechanism. The results showed that the HTAP algorithm is able to reliably lead a large percentage of data packets to the sinks. In contrast, the same network, without congestion control mechanism, looses almost half of its data packets.

Moreover, simulations show that HTAP can also contribute to the increment of network's life since it uses power from a big number of nodes. This means that networks are dying out, uniformly. In other cases, due to congestion, it is possible that a specific area of the networks gets exhausted, while other areas are "untouched". Moreover a special provision has been taken in our algorithm in order that the entire exhausted nodes to get removed immediately from the network's topology, because they may cause major problems.

7. REFERENCES

[1] K. Römer and F. Mattern, "The Design Space of Wireless Sensor Networks". *IEEE Wireless Communications* 11 (6): 54-61, December 2004

[2] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks", *In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, pp 221-235,* July 2001**.**

[3] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks", *In Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), p.55,* September 2002.

[4] C. Tien Ee, R. Bajcsy, "Congestion Control and Fairness for many-to-one Routing in Sensor Networks", *In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems,* November 2004.

[5] B. Hull, K. Jamieson, H. Balakrishnan, "Mitigating Congestion in Wireless Sensor Networks", *In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems,* November 2004.

[6] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks", *In Proceedings of the ACM SenSys*, 2003.

[7] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks". *In Proceedings of MobiCom 2001, pages 221–235,* 2001.

[8] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks" *In Proceedings of ACM MobiHoc`03,* pp. 177-188, June 2003,

[9] R.J. Gibbens, F.P. Kelly and P.B. Key "Dynamic Alternative Routing - Modelling and Behaviour", *In Proceedings of the 12th International Teletraffic Congress,* 1988.

[10] T. Yan, T. He, and J. A. Stankovic. "Differentiated Surveillance Service for Sensor Networks." *In Proceedings of the ACM SenSys 2003*, 2003.

[11] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. *In Proceedings of the 23rd International Conference on Distributed Computing Systems*, May 2003.

[12] Mingyan Liu "Modeling a Dense Wireless Sensor Network: Complexity, Stability and Robustness", *IPAM Mathematical Challenges and Opportunities in Sensor Networking (SN'07)*, Jan. 2007