DAlPaS: A Performance Aware Congestion Control Algorithm in Wireless Sensor Networks

Charalambos Sergiou and Vasos Vassiliou
Networks Research Laboratory
Department of Computer Science
University of Cyprus
Nicosia, Cyprus
Email: {sergiou,vasosv}@cs.ucy.ac.cy

***Abstract*—As Wireless Sensor Networks are evolving to applications where high load demands dominate and performance becomes a crucial factor, congestion remains a serious problem that has to be effectively and efficiently tackled. Congestion in WSNs is mitigated either by reducing the data load or by increasing capacity (employing sleep nodes). In either case due to the energy constraints and low processing capabilities of sensor nodes, congestion control and avoidance algorithms has to be kept as simple and efficient as possible while overhead must be limited. In this paper we propose a novel and simple Dynamic Alternative Path Selection Scheme (DAlPaS) attempting to face congestion by increasing capacity while attempts to maintain performance requirements. DAlPaS can efficiently and adaptively choose an alternative routing path in order to avoid congested nodes, by taking into consideration a number of critical parameters that affect the performance of a WSN while maintaining overhead in minimal levels. Simulation results show that DAlPaS algorithm can perform significant performance achievements over comparable schemes.**

***Keywords*-Wireless Sensor Networks, Multipath Routing, Congestion Control, Energy Utilization**

I. Introduction

Wireless Sensor Networks (WSNs) 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]. Sensor nodes are low cost, light-weight, tiny devices with energy constraints. Frequently, sensor nodes are densely deployed near the event sources and sinks in a redundant manner [2]. Since power is a severe limitation for these nodes, dense deployment assists in the networks' robust operation (e.g. due to node or link failure) and significantly contributes in the overall network performance. Congestion happens when the offered load is more than the available capacity of the network. Currently there are two ways to face congestion. Either by reducing the offered load (traffic control) or by increasing capacity (resource control) [3]. Both methods presents advantages and disadvantages under specific scenarios. In general, traffic control methods can be considered as more effective methods when transient overload situation exists while resource control method is more effective when persistent high load demands exist [3], [4]. Although traffic control method is much simpler and less costly in comparison with resource control method the fact that the number of packets are reduced exactly at the moment where the monitored event is taking place, it renders insufficient for a number of applications. In such a case resource control method must apply.

The challenge in this case is twofold. At first, a simple algorithm with minor computations must be developed while at second it should be able to count a number of performance parameters before taking the decision for the alternate path. Choosing an alternate path in such dense topologies when congestion arises, is a challenging task since various parameters, which are critical for the network's mission need to be handled. These parameters include: (a) the node's (in a microscopic way) and the network's (in a macroscopic way) remaining energy, (b) congestion, in terms of buffer occupancy and interference, (c) the time to transmit data from the source to the sink, and (d) the packet loss rate of the network. Several research papers have addressed some of these problems using single path routing, either for energy efficiency [5] [6] or for congestion control [7] [8]. However, single path routing algorithms suffer from the problem of individual node power exhaustion, since they tend to use the shortest path to forward their data. This, eventually, leads to the discovery and creation of a new path, which is a power-consuming procedure. In addition, the only way to control congestion using single path routing, is by reducing the transmission rate of source nodes, an unacceptable condition for many applications [3]. We summarize our contribution below: In this paper we present a novel and simple Dynamic Alternative Path Selection Scheme (DAlPaS) which can efficiently choose an alternate routing path in case of congestion, taking into consideration a number of critical performance parameters such as the remaining power of nodes, the available buffer space, the medium interference, as well as the nodes' distance from sink. This work takes in consideration the results presented in [4] and [9] and the target is to keep it as simple as possible in order to be feasibly implemented in wireless sensor nodes. The novelty of this algorithm lies on the dynamic number of parameters that examines in relation with the minimal overhead. In addition, we believe that the simplicity of the algorithm, makes it ideal for low-processing-capable terminals.

978-1-4577-0024-8/11/$26.00

II. Related Work

Resource control as method for congestion control and avoidance as well as for reliable data transmission in WSNs has not attracted a lot of attention. Some notable efforts are presented below.

TARA (Topology Aware Resource Adaptation) [3] protocol focuses on the adaptation of network's extra recourses in case of congestion, alleviating intersection hot spots. TARA copes with buffer occupancy as well as channel loading. In TARA, congestion alleviation is performed with the assistance of two important nodes. These are the distributor and the merger nodes. Between them a "detour path" is established starting at the distributor and ending at the merger. The distributor distributes the traffic coming from the hot spot between the original path and the detour path, while the merger merges the two flows. Thus, in case of congestion and creation of hot- spot, traffic is deflected from the hot spot through the distributor node along the detour and reaches the merge node, where the flows are merged. As long as congestion has been alleviated the network stops using the detour path. For quick adaptation the distributor node keeps in its memory which neighbor is on the original path.

CADA [10] Fang et al proposed CADA. CADA stands for Congestion Avoidance Detection and Alleviation in WSNs. In this algorithm node's congestion level is measured by an aggregation of buffer occupancy and channel utilization. It actually counts the growing rate of buffer's occupancy and when exceeds a certain limit, node is consider congested. On the other hand if packet delivery ratio decreases drastically while the local channel loading reaches the maximum achievable channel utilization, it infers that there is channel congestion. For congestion mitigation CADA employs both resource control and rate control depending on the case. If congestion takes place in an intersection hotspot then resource control applies, while if congestion takes place in a convergence hotspot traffic control applies. Simulation results prove that CADA present better results concerning throughput, energy consumption, end to end delay and average per hop delay in comparison with TARA [3] and "no congestion control" algorithm.

GRAdient Broadcast (GRAB) [11] addresses the problem of robust packets forwarding from source to sink in WSNs using multipath routing. GRAB is divided in three main parts. The first one is source selection. To limit the number of forwarded packets a single node is selected in each area of stimuli, as the source that will forward the event packets. Source is selected as the node with the stronger signal and is called the Center of Stimuli (CoS). Following, the sink builds and maintains a cost field. The cost at a node is the minimum cost needed to forward a packet from this node to the sink. When a node is ready to forward a packet, it broadcasts this packet, having its cost included in the packet. Nodes, whose cost is equal or higher than the forwarding node's cost, drop the packet. The rest broadcast the packet in the same way like the source. This procedure is repeated until the packet reaches the sink. Finally, to control the number of possible selected paths, the source assigns a credit to each packet it transmits. The sum of credit and source costs is the total budget that can be used to send a packet to the sink along a path. In this manner, the network mesh is wider near the source and becomes narrower near the sink.

Directed Diffusion [12] is a data centric protocol, because all communication is for named data. All nodes in a directed diffusion-based network are application-aware. This enables diffusion to achieve energy savings by selecting empirically good paths (small delay) by caching and processing data in-network (e.g., data aggregation). Directed diffusion consists of four (4) basic elements: interests, data messages, gradients, and re-inforcements. An interest message is a query from a sink node to the network, which indicates what the application wants. It carries a description of a sensing task that is supported by a sensor network. Data in sensor networks is the collected or processed information of an event (e.g. physical phenomenon), it is named (addressed) using attribute-value pairs and a sensing task is diffused throughout the sensor network as an interest for named data. This dissemination sets up gradients within the network designed to "draw" events (i.e., data matching the interest). A gradient is a direction state created in each node that receives an interest. This direction is set toward the neighboring node from which the interest was received. Events start flowing towards the sinks of interests along multiple gradient paths. To improve performance and reliability, the empirically "good paths" (e.g small delay) are reinforced by the sink and their data rate increases. On the other hand, unreliable paths (e.g high delay) are negatively reinforced and pruned off.

HTAP [13] is a scalable and distributed framework for minimizing congestion and assuring reliable data transmissions in event based networks. As such, it does not employ rate limiting actions, but tries to maintain a high level of packet rate while minimizing packet losses. It is based on the creation of alternative paths from the source to sink, using the plethora of a network's unused nodes, in order to safely transmit the observed data. The creation of alternative paths involves several nodes which are not in the initial shortest path from the source to the sink. The use of these nodes leads to a balanced energy consumption, avoiding the creation of "holes" in the network and prolonging network lifetime.

III. Dynamic Alternative Path Selection Scheme Description

DAlPaS is a congestion control and avoidance algorithm that attempts to choose an alternate path in case of congestion taking into account a number of basic performance parameters. Complementary to Energy Aware Protocols [14][15] that find the lowest energy route or energy sufficient paths to forward data and base their path alternation decision on these conditions, DAlPaS also takes into consideration the node's congestion situation (both in terms of buffer occupancy and channel interference). On the other hand, while congestion control and reliable data transmission protocols like [3] [13][11] base their "alternate path" decision on a congestion theshold or the path's

cost, DAlPaS also counts the node's remaining power. DAlPaS is completely dynamic and distributed algorithm. Besides the "Setup Phase", where the network is initially discovered, all subsequent decisions are based only on the condition of the node in the current data forwarding "epoch". The details of the algorithm are presented below:

A. Setup Phase

At the beginning of the Setup Phase the Sink broadcasts a "Hello" message marked as Level 0. Nodes that are in the radio range of the sink receive this packet, mark it as Level 1 and set themselves as Level 1 nodes. Level 1 nodes re-broadcast this "Hello" message transmitting in full power. Upon receiving a "Hello" message for the first time, a node adds one to its level and broadcasts it again. A node may receive more than one "Hello" messages (from different neighbors). In such a case it rebroadcasts the message only if it has changed (lowered) its current level information. In case there are many nodes at the immediately lower level, it keeps all in its neighbor table, along with all other connectivity information it overhears. With this procedure, nodes discover each other, build and initialize their neighbor tables, and at the same time record their minimum hop distance to the sink.

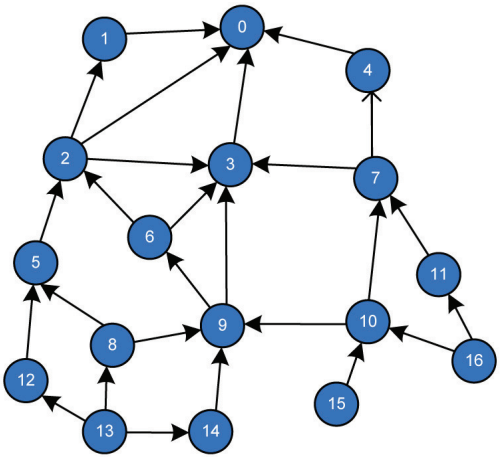

Fig. 1. Initial Network Connectivity

To explain this concept better let us consider the network in Fig. 1. In this scenario the sink (node 0) broadcasts a "Hello" message as Level 0 and nodes 1, 2, 3 and 4 receive this message and set themselves as Level 1 nodes. Then, these nodes broadcast the message downstream. Nodes 5, 6, 7 and 9 receive the "Hello" message and mark themselves as Level 2 nodes. The procedure iterates until the whole network is discovered. Fig. 1 actually illustrates the understanding of each node about all possible upstream paths from itself towards the sink through nodes at the same or lower levels.

When node 8 broadcasts its "Hello" message (as Level 3), node 9 will also receive it. Node 9 will compare the level of this message with the level it already possesses (Level 2) and will ignore it. If for any reason node 9 receives the message from node 3 after the message from node 8, it will update its level from Level 3 to Level 2 and will re-broadcast the updated level. Fig. 2 shows the connectivity resulting from using only the lower level nodes.

TABLE I
EXAMPLE OF NEIGHBOR TABLE FOR NODE 2

Node ID	Buffer Occupancy	Remaining Power	Number of Hops	Flag
0	0	1J	0	True
1	0.90	0.95J	1	True
3	0.92	0.96J	1	True
6	0.85	0.96J	2	True
5	0.90	0.93J	2	True

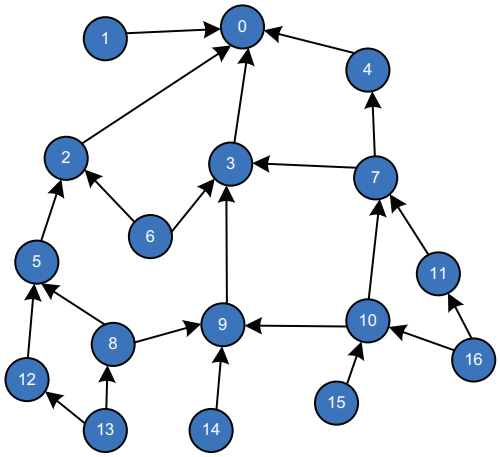

Fig. 2. Placement of Nodes in Levels

An example of neighbor table is presented in Table I. The neighbor table maintains records for the ID of its neighbors, their buffer occupancy, their remaining power, their number of hops to sink, as well as a field called "Flag", which indicates the node's availability at the current moment. The Flag mechanism is explained later in this section. It is clear that the neighbor table holds information for all neighboring nodes and not only for the nodes that are one level closer to the sink. Using this "custom" flooding mechanism all possible routes, which can be used to forward packets upstream, are discovered.

B. DAlPaS Mechanisms

After the setup phase where all possible routes (paths) from source to sink have been discovered, nodes connected to the source begin to forward data. Initially, nodes forward their data packets through the node that provides the shortest route to the sink. Each data packet header contains the sequence number of the packet, the sending node ID and well as the destination (receiving) node ID. When a data packet is received by a node the packet must be acknowledged. If the node has successfully received the packet, it broadcasts an ACK packet which its headers contains a set of fields as described in Table II. All nodes receiving this ACK packet, modify their neighbor table with the updated values.

DAlPaS scheme employ two stages. A soft and hard stage.

Soft Stage: Soft stage is introduced as a proactive way to face (or avoid) transient congestion situation as well as to balance and spread the traffic between as much nodes as possible. Each node is entered in DAlPaS soft stage's alert condition when it receives packets from more than one flows. In such case each node that it faces this situation is a candidate congested node (in case that its receiving rate exceed the transmitting rate). In order to prevent this situation DAlPaS algorithm attempts at first to keep each node receiving data from one flow. To achieve this, the node sets the "Next Packet Sequence Number" field in the ACK packet header to "False" for the specific node ID that would like to "inhibit" its transmission. When this node receives this ACK packet it is informed that it should check for next suitable path starting from a another node at the same level as before. The sending node is able to understand that the reason of this inhibition is another flow, since the Flag field in the ACK packet remains "True". Note, that in soft stage, receiving nodes just advice the sending nodes to find another path, so they keep receiving and forwarding their data until forwarding nodes decide to stop sending packets. In soft stage, receiving nodes always try to keep the flows with the bigger sending data rate. By employing this tactic besides the obvious benefit of buffer based congestion avoidance, the network utilizes its resources uniformly and routing holes are avoided.

Hard Stage: In case of high traffic load or in case where the performance requirements of the application especially in terms of delay are not satisfied through a new routing path, it is possible that an "inhibited" node to decide to keep sending packets to the same node although it is aware that it is possible to cause congestion. In this case if node exceeds the "performance threshold" is entering in hard stage.

Hard stage is a situation where the network **forces** the flows to change routing paths since "performance threshold" is exceeded. Responsible to monitor and apply performance thresholds is "Flag Decision" algorithm which is described below.

Flag Decision Algorithm: Flag decision algorithm runs when a node enters in hard stage. In this stage a node becomes temporarily or permanently unable to accept any more packets from any flows. A node may become unable to receive data for the following reasons:

- **Buffer Occupancy is reaching its upper limit**: If a node is receiving packets at a higher rate than it can transmit, it will soon have its buffer overflowed. For example, in Fig.3 this can happen if nodes 10 and 11 send data to node 7 and their aggregation data rate is more than the data rate that node 7 is forwarding these data. In this case, the buffer of node 7 soon overflows.
  In a scenario like this, when the receiving node figures out that its buffer is about to overflow, it sets its Flag field to in the next ACK packet that it broadcasts to "False". All nodes in the area that receive this packet, alter immediately the flag entry for this node in their "Neighbor Table" to "False" and look in their table for the next hop that is available to forward their data. When the "failed" node has recovered and is able again to receive packets, it broadcasts a "Hello" packet stating this event and neighbor nodes alter the flag to "True" in their neighbor table.
- **Low Remaining Power**: The "Flag Decision Algorithm" also applies in the case when a node is getting power exhausted. Each node is programmed to set its flag to "False" every time its Remaining Power falls below a certain percentage of the total. When this happens, whether during a data session or when a node runs in idle, the node alters its Flag to "False" and informs its neighbors for this event through an ACK or a "Hello" packet. The nodes that receive this message apply the same procedure as for the buffer occupancy case respectively. The remaining power threshold depends on the application. In time critical applications it is best to be kept low to make sure that we use the shortest available path for the longest time. In periodic applications it may be set to a higher level in order to maintain a more energy-uniform utilization of the network.
- **Higher level node unavailability**: Another case in which the "Flag Decision Algorithm" applies is when, although a node is available with respect to buffer and power there is no other node available at a level higher that itself to transmit data to. In this case, it is forced to advertise a "False" flag. If the nodes at the higher level are not available due to Power extinction, buffer occupancy or because they may have been physically removed from the network, this functionality protects the network from forwarding packets to network "black holes".

Alternative Path Creation:Since the algorithm is dynamic, the number of hops to the sink for a node is possible to change when the state of nodes at a level closer to sink changes. The choice of the next node to forward data, after avoiding the congested node, depends firstly on its availability (Flag) and the number of hops to the sink. Using this tactic each node can, in an easy and simple way, find the next node with minimum computation. It just sorts the number of available nodes in ascending order, with respect to their number of hops to the sink, and forwards the packets through the first node in the list. If the first node becomes unavailable in any way (soft or hard) the sender immediately chooses the next node in the list. In case that more than one nodes are in the same level (same number of hops to sink), the table is sorted based on the remaining power (above or below some specific thresholds). Finally in case that more than one nodes are above these thresholds they are sorted by their remaining buffer occupancy. In the extreme case where even this value is the same for more than one nodes, the algorithm chooses the node with the smaller node ID to forward the packet. Using this method the algorithm gives priority to the maintenance of performance metrics like the mean time for the transmission of packets from source to sink, as well as to the network's uniform energy utilization thus avoiding the creation of "holes".

For example let us consider figures 3 and 4. In this example

TABLE II
ACK PACKET HEADER

node ID	Next Packet Sequence Number	Buffer Occupancy	Remaining Power	Number of Hops	Flag

node 14 forwards packets to the sink through nodes 9 and 3. If for any of the reasons explained above node 3 fails, node 9 will have to search in its neighbor table to find a node to replace node 3 since it cannot reach directly the sink. We note (from Fig. 2) that node 9 does not have any other "Level 1" nodes to connect to. Therefore, it must search for "Level 2" nodes. In this case it finds node 6, from which it will forward the rest of the data. In this case since node 6 is "Level 2" node, node 9, immediately becomes "Level 3" node. Concurrently all nodes connected to it (8, 10, 14) will update their tables.

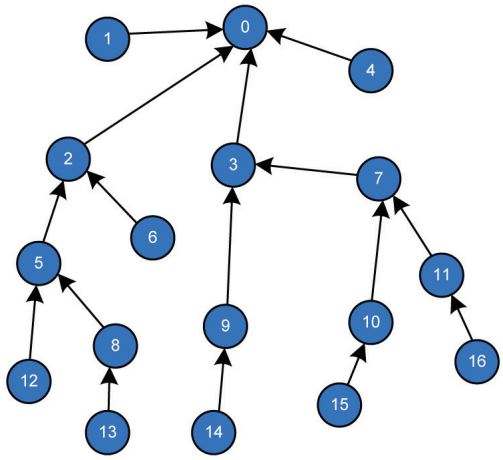

Fig. 3. Shortest Paths from Source to Sink before Node Failure

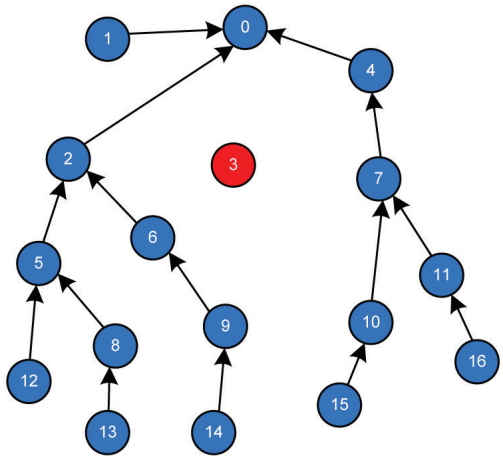

Fig. 4. Paths from Source to Sink after Node Failure

IV. EVALUATION

To evaluate the performance of DAlPaS a series of simulation have been performed using Prowler [16] simulator, a probabilistic wireless network simulator. Prowler provides a radio fading model with packet collisions, static and dynamic asymmetric links, and a CSMA MAC layer. DAlPaS is compared to TARA [3]. TARA is a state of the art congestion control algorithm that also uses "resource increment" method to mitigate congestion. TARA is briefly explained in section II of this paper. The main difference between TARA and DAlPaS is the fact that TARA is using a capacity analysis model through a graph- theoretic approach and attempts each time to form a new topology that has just enough capacity to handle the traffic in case of congestion. TARA is focused on intersection hot spots in permanent congestion situations. On the other hand DAlPaS, through soft and hard stage attempts to control both transient or permanent congestion situations, while it is always using the same topology that has been created in setup phase.

A. Simulation Environment

To perform our simulations we have used the radio propagation model provided by Prowler. The transmission model is given by:

$$P_{rec,ideal}(d) \leftarrow P_{transmit}\frac{1}{1+d^{\gamma}} \quad (1)$$

where, $2 \leq \gamma \leq 4$ and

$$P_{rec}(i,j) \leftarrow P_{rec,ideal}(d_{i,j})(1+a(i,j))(1+\beta(t)) \quad (2)$$

where $P_{transmit}$ is the signal strength at the transmitter and $P_{rec,ideal}(d)$ is the ideal received signal strength at distance d, a and β are random variables with normal distributions $N(0,\sigma_a)$ and $N(0,\sigma_\beta)$, respectively. A node j can receive packets from node i if $P_{rec}(i,j) > \Delta$ where Δ is the threshold.

B. Simulator Setup

In our simulations we set $\sigma_a = 0.5$, $\sigma_\beta = 0.03$ and $p_{error} = 0.05$. The reception threshold is set to be $\Delta = 0.1$.

The rest parameters we employ represent Mica-Z node and the most important of them are presented in next table.

TABLE III
SIMULATION PARAMETERS

Max Data Rate (kbps)	250
Transmission Power (dbm)	2
Receive Threshold (dbm)	-74
Transmission Current (mA)	17.4
Receive Current (mA)	19.7
Fragment Size (bit)	1024
Buffer Size(Bytes)	512K
MAC layer	CSMA/CA

Concerning DAlPaS we set the buffer occupancy threshold to 90% of total and the remaining power threshold to 8% total, thus assisting the node to forward all packets on fly avoiding their drops. 500 nodes were uniformly deployed in a 100m x 100m square area.

C. Performance Metrics

The following metrics are employed:

- The first metric we employ is total energy consumption. This parameter is an indication of the energy efficiency of the algorithms. In this metric we actually sum the remaining power of all nodes after the end of simulation. Total energy consumption is represented in (Eq. 3).

$$E_{total} = \sum_{i=1}^{N} (e_{i,init} - e_{i,res}) \tag{3}$$

where: N is the number of Nodes, $e_{i,init}$ and e_{i-res} are the initial and residual energy levels of node i.

- The second metric we employ is the percentage of successfully received packets (Eq. 4). This metric is particularly important for critical/emergency applications, where every packet has to be received from the sink.

$$ReceivedPktsRatio(\%) = \frac{SuccessfullyReceivedPkts}{TotalPktsSent} \tag{4}$$

- The third metric we employ is Average Hop- by- Hop delay a metric used to declare the time that is spent by a packet in order to be transmitted from one hop in another. This metric in case of congestion is an indication of the congestion control algorithm overhead.

D. Simulation Results

The results are presented in figures 5, 6, 7.

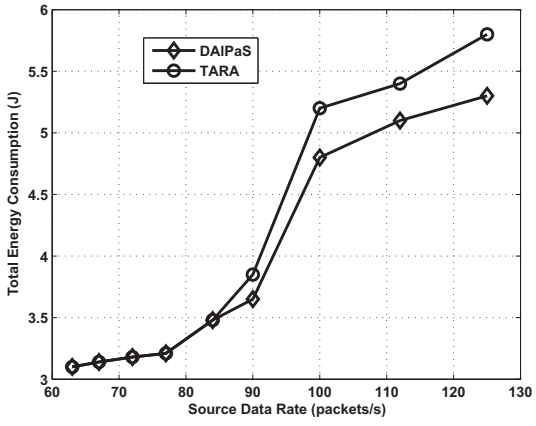

Fig. 5. Total Energy Consumption

It is observed that DAlPaS presents better performance in comparison with TARA in this network configuration. The fact that DalPaS is able to effectively and efficiently utilizes resources by preventing transient congestion conditions, by employing "soft stage" provides the ability of using the whole network resources in case of permanent congestion conditions.

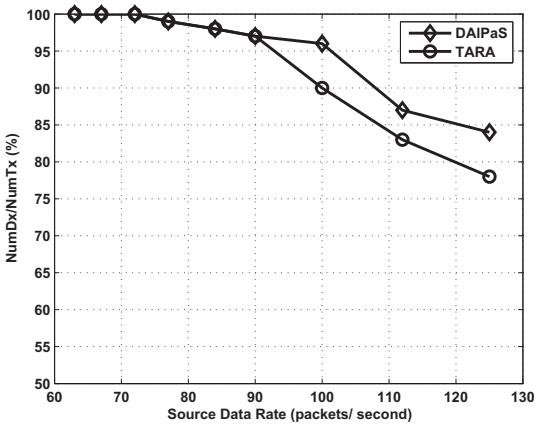

Fig. 6. Percentage of Successfully Received Packets

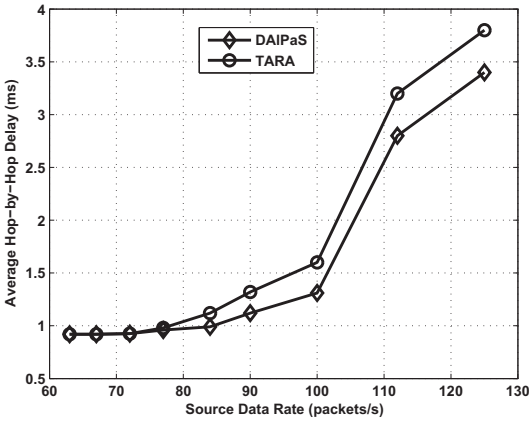

Fig. 7. Average Hop- by- Hop Delay

On the other hand in case of permanent congestion condition where "hard stage" is mostly used provides the network with alternative paths that avoid hotspots or resources insufficient areas. This leads to reduced total energy consumption. Also the simplicity of DAlPaS is depicted in Fig. 7 where it presents less overhead than TARA. Finally DAlPaS seems to present also good performance concerning its efficiency to successfully deliver data packets to sink even in case of heavy load.

V. Conclusions and Future Work

In this paper we propose a novel Congestion Control and avoidance algorithm called DAlPaS (Dynamic Alternative Path Selection) which is able to uniformly utilize network resources (sensor's node power), while, at the same time maintain robust and reliable data delivery. The strength of the proposed algorithm is that it does the work correctly, with simplicity, and with improved performance. The DAlPaS advantages are based on the "soft stage" phase where each node attempts to avoid possible transient congestion situation since it attempts to serve just one flow and "Flag Decision Algorithm" with which

the node makes itself available, or not, to the network with minor computation ("hard stage"). In addition, minor computations are also performed by the forwarding nodes. Those nodes just search in the neighbor list to find available nodes closer to the sink. This makes DAlPas a uniquely suitable algorithm for distributed low-power deployment. Simulation results prove that DAlPaS compares favorably to an established and state of the art algorithms like TARA. We believe that this initial work can be extended to provide additional insights into the possible tradeoffs between Congestion and Propagation Delay and between Energy Consumption and Hop Count.

ACKNOWLEDGMENTS.

This work has been conducted under the European Union Project GINSENG funded under the FP7 Program (FP7/2007-2013) grant agreement no 224282.

REFERENCES

[1] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, December 2004.

[2] S. Toumpis and L. Tassiulas, "Optimal deployment of large wireless sensor networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2935–2953, 2006.

[3] J. Kang, Y. Zhang, and B. Nath, "Tara: Topology-aware resource adaptation to alleviate congestion in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 919–931, 2007.

[4] C. Sergiou and V. Vassiliou, "Alternative path creation vs data rate reduction for congestion mitigation in wireless sensor networks," in *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (Poster Session)*. New York, NY, USA: ACM, 2010, pp. 394–395.

[5] C.-K. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks," *IEEE Communication Magazine*, pp. 138–147., June 2001.

[6] M. K. Marina and S. R. Das, "Ad hoc on-demand multipath distance vector routing," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 3, pp. 92–93, 2002.

[7] Y. Sankarasubramaniam, Akan, and I. F. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in wireless sensor networks," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2003, pp. 177–188. [Online]. Available: http://dx.doi.org/10.1145/778415.778437

[8] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: congestion detection and avoidance in sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 266–279. [Online]. Available: http://portal.acm.org/citation.cfm?id=958523

[9] C. Sergiou and V. Vassiliou, "Energy hole prevention in wireless sensor networks," in *IPSN '10: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (Poster Session)*. New York, NY, USA: ACM, 2010, pp. 398–399.

[10] W.-w. Fang, J.-m. Chen, L. Shu, T.-s. Chu, and D.-p. Qian, "Congestion avoidance, detection and alleviation in wireless sensor networks," *Journal of Zhejiang University - Science C*, vol. 11, pp. 63–73, 2010, 10.1631/jzus.C0910204. [Online]. Available: http://dx.doi.org/10.1631/jzus.C0910204

[11] F. Ye, G. Zhong, S. Lu, and L. Zhang, "Gradient broadcast: a robust data delivery protocol for large scale sensor networks," *Wireless Networks*, vol. 11, no. 3, pp. 285–298, 2005.

[12] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[13] C. Sergiou, V. Vassiliou, and A. Pitsillides, "Reliable data transmission in event-based sensor networks during overload situation," in *WICON '07: Proceedings of the 3rd international conference on Wireless internet*, Austin, Texas, October 2007, pp. 1–8.

[14] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 4, pp. 11–25, 2001.

[15] R. P. Mann, K. R. Namuduri, and R. Pendse, "Energy-aware routing protocol for ad hoc wireless sensor networks," *EURASIP J. Wirel. Commun. Netw.*, vol. 2005, no. 5, pp. 635–644, 2005.

[16] Prowler: Probabilistic wireless network simulator. [Online]. Available: http://www.isis.vanderbilt.edu/Projects/nest/prowler/