# Napping Backbones: Energy Efficient Topology Control for Wireless Sensor Networks

Rituparna Ghosh and Stefano Basagni
Department of Electrical and Computer Engineering
Northeastern University
E-mail: {rghosh,basagni}@ece.neu.edu

***Abstract*—In this study we have investigated the effectiveness of building "napping backbones" for data dissemination in wireless sensor networks. The NAPBACK protocol builds connected backbones whose nodes are endowed with a sleep/awake schedule that induces considerable energy savings, and hence prolongs the network lifetime. Via simulations on networks with up to 250 nodes we have observed increases on network lifetime up to almost 70% with respect to previous topology control protocols (S-DMAC). Increased latency is the price to pay for the improvements on lifetime, which currently makes NAPBACK a viable solution for delay-insensitive WSN applications. Multifold are the research directions opened by this initial study. We are planning to design different methods for defining the schedules of the backbone nodes. Final aims include the minimization of the latency, as well as throughput maximization. Sleep/awake scheduling methods should also be independent of nodes synchronization, and could be based on deterministic strategies, rather than the simple randomized technique used here.**

***Index Terms*—Wireless sensor networks, energy efficient node schedules.**

## I. Introduction

This paper concerns the definition of a mechanism for the control of the topology of a wireless sensor network (WSN), i.e., a multi-hop wireless network whose nodes are typically deployed in large numbers and are energy constrained. The nodes have sensing capabilities and the sensed data is routed to data collection points called *sinks* via intermediate sensors.

The kind of topology control we are interested in is the one that uses those features of the sensor nodes for which their radio interface can be "put to sleep," i.e., turned off. This is done for conserving a node's energy, since radio functions (transmitting, receiving and just being idle) are by far the most energy consuming activities of a sensor node. At the same time, turning off nodes has immediate consequences on the topology of the WSN. Topology control concerns the definition of a sleep/awake schedule for the sensor nodes in such a way that the topology formed by the nodes that are awake can still deliver data to the sink, while the number of nodes that are asleep is maximized to conserve the highest amount of energy. The overall aim is that of prolonging the network lifetime.

Our approach to topology control consists in selecting a subset among the sensor nodes that forms a connected backbone. Sensed data are delivered to the sink only via the backbone nodes. The basic idea is to keep the nodes in the backbone awake, and to put every other node to sleep. In [1] we have shown that this approach, termed Sensor-DMAC (S-DMAC), yields to increases in network lifetime with respect to other backbone formation and topology control mechanisms such as DMAC [2] and GAF [3]. However, nodes in S-DMAC backbones, being always awake and relying data for every other node, tend to deplete their energy quite quickly, which limits the improvements on network lifetime. Therefore, we propose a new awake/asleep mechanism for the backbone nodes. Nodes in the backbone take "naps" (sleeping periods that are shorter than those of non-backbone nodes) in such a way that, while conserving energy, they can still provide efficient data delivery to the sink. Through thorough ns2-based simulation experiments, we show that our solution, termed *NAPBACK*, is effective in increasing the network lifetime up to almost 70% with respect to DMAC and S-DMAC.

In this paper we describe the NAPBACK protocol (Section II) and show the results of our performance comparison among NAPBACK, S-DMAC and DMAC (Section III). Detailed NAPBACK procedures as well as an in-depth section on related work will appear in the full-blown paper.

## II. The NAPBACK Protocol

The NAPBACK protocol is executed at each sensor node periodically, in a completely distributed and localized way, which meets the requirements of wireless sensor networking. The protocol starts by building a backbone among the network nodes according to the backbone formation protocol DMAC [2]. Nodes are divided into clusterheads (CHs), gateways (GWs) and ordinary nodes (ONs) so that every ordinary node has a CH as its immediate neighbor. CHs and GWs form a connected backbone (backbone nodes). This same approach is followed by the protocol S-DMAC [1], where the backbone nodes stay awake while the (radio interfaces of the) ONs are kept to sleep. When an ON senses an event, it wakes up and send the corresponding data packet to one of the neighboring CHs. Then it goes back to sleep. The data reaches the sink through the backbone nodes.

NAPBACK starts from an S-DMAC backbone, and compute a sleep/awake schedule for the backbone nodes. The nodes are synchronized on a frame base. Sleep/awake schedules are calculated at the backbone nodes at the beginning of each frame. This allows the schedules to adapt to possible changes in the network topology and in data traffic.

In the following, *Nap-count* will be used to indicate the number of times that a CH decides to wakeup in the current frame. CHs are completely free to pick up their own schedule, independently of every other node's schedule. Once a node is up, it stays up for a predefined amount of time called *Non-nap-period*. We consider the backbone divided into levels $0, \ldots, \ell \leq n$. A node (sink included) belongs to level $k$ if its hop distance from the sink is $k$. The ***upstream node*** of a level-$k$ node $v$ is a neighboring backbone node that belongs to level $k-1$. An ON has only one upstream node, namely, its CH. For a level $k$ backbone node all its neighboring backbone nodes that are in level $k-1$ are its own upstream nodes.

Once the S-DMAC backbone is formed, a node starts the computation of its napping schedule. This process starts from the backbone nodes, and it is then distributed to the ONs. Once the schedules are up at the nodes, routing of the sensed data from the sensors to the sink is performed according to the sleep/awake schedule of the backbone nodes. In the following we describe the two major phases of schedule calculation and data forwarding.

## A. Schedule calculation

Time is divided into frames of length NAP-FRAME. All nodes wake up at the beginning of each frame. The napping schedules are calculated and exchanged among the nodes during the SCHED-XCHANGE-PERIOD at the beginning of each frame. During this time no data traffic is forwarded and any sensed data is buffered by the nodes.

The CHs, independently, choose the Nap-count number of times they will wake up in the current frame. The schedule is then broadcast to all their neighbors. The ONs simply note down the schedule of their CH.

Since the CHs do not depend on any neighbors for deciding their schedules, it is left to the GWs to ensure that their sleep/awake schedule guarantees the forwarding of the data packets to the sink. Adapting to the schedule of upstream and downstream CHs could lead to a high Nap-count for GWs. However, we observe that by having a GW adapting to the schedules of its upstream nodes and to that of those downstream CHs that choose it as their upstream node is sufficient to guarantee some common awake time for data forwarding in the current frame. Hence, the GWs calculate their schedules in the following way: A GW $v$ that belongs to level $k$ wakes up for all the CHs at level $k+1$ that are dependent on it (i.e., when all the CHs who have $v$ as their upstream node are awake). $v$ also wakes up when all the $k-1$-level backbone nodes (both CHs and GWs) it is dependent on are awake. The resulting schedule ensures that in any given frame nodes will find at least a common Non-nap-period with nodes they depend on and with nodes that depend on them. Once the GWs of level $k$ have computed their schedules, they broadcast them to all their level-$k+1$ neighbors, enabling them to forward their data at the right time.

## B. Data forwarding

A node sensing an event at a time when the backbone node that is serving it is not awake buffers it. Otherwise, it forwards the corresponding packet to the upstream nodes. In particular, when an ON has data to send it checks whether its CH is up, and if so it transmits the packet. Otherwise, it buffers the packet and keep sleeping till it knows its CH is up. A backbone node (CH and GW) $v$ wakes up according to its own computed schedule. Upon waking up, it checks which of its upstream nodes are awake, based on their advertised schedules. Then, $v$ unicasts the packet to all the upstream nodes that are awake. In our implementation, these successive unicasts are sent with a jitter in order to reduce the probability of collisions with multiple dependent nodes trying to send data packets to the same destination at the same time. When $v$ receives an ACK from at least one of the nodes it sent the packet to, it removes the packet from the buffer, and transmit the following packet (if any) if time allows. After Non-nap-period, $v$ goes back to sleep.

## III. Simulation Results

Our ns2 implementation is based on the CMU wireless extension, i.e., on the IEEE 802.11 MAC with DCF [4]. Our simulations refer to scenarios where $n$ wireless nodes with maximum transmission radius of 30m are randomly and uniformly placed in a geographic square area of side $L$. We make the assumption that two nodes are neighbors if and only if their Euclidean distance is $\leq$ 30m. Each node has an initial energy of 2J. The power consumed while transmitting, receiving, being idle and in sleep mode are 14.88mW, 12.5mW, 12.36mW and .016mW, respectively. The energy model used corresponds to the specifications of the TR 1000 radio receiver from RF Monolithics. The number of nodes $n$ has been assigned the values 100, 150, 200 and 250, while $L$ has been set to 200m. This allows us to test the protocols on increasingly dense networks. Simulations are obtained for different data rates of .05pkt/s, .07pkt/s and .1pkt/s. The packet size is set to 64 bytes. The Non-nap-period is .4s. The frame length varies from 10s to 30s. All results are obtained averaging over 100 (connected) topologies. The sensor nodes wake up between 0 and 40s and the total simulation time is 5000s.

Metrics of interest for our study are the following (all averages).

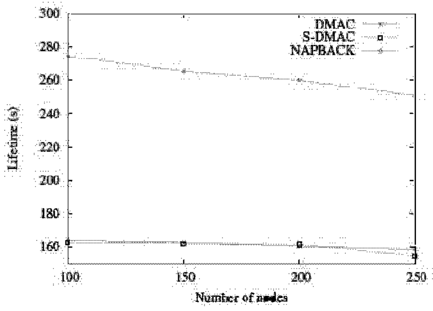

Fig. 1. Network lifetime

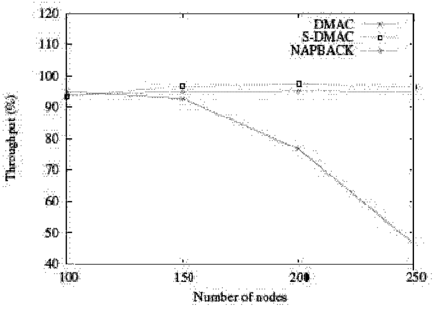

Fig. 3. Data throughput

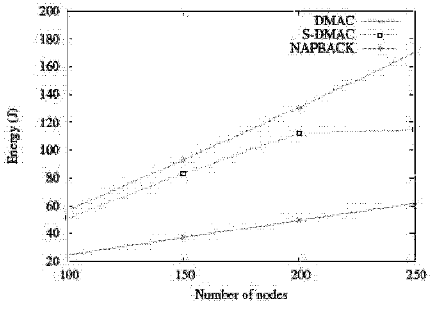

Fig. 2. Residual energy left in the network

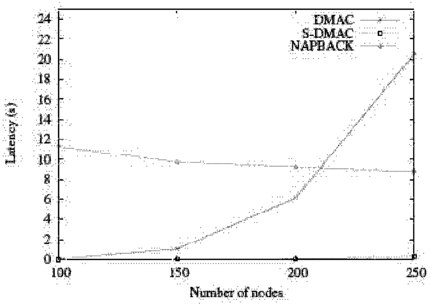

Fig. 4. Latency of packet arrival at the sink

- Network lifetime: The time needed for the 1st node to "die" because of energy depletion. A node is declared dead when its energy drops to 0.
- Latency: The time it takes for delivering a packet from its source to the sink.
- Throughput: Ratio between the number of packets received at the sink to the number of packets generated at the nodes.
- Total energy: Residual energy left in the network at the end of the network lifetime.

### A. *Evaluation of DMAC, S-DMAC and NAPBACK.*

All averages refer to a data rate of 1pkt/20s and a frame length of 20s. Nap-count here is set to 5.

Figure 1 compares the network lifetime of the three protocols. We hardly notice any difference in results between S-DMAC and DMAC. This is because in both cases the idle energy dominates and the backbone nodes being awake keep consuming energy (independently of the network load). The first node to die in both the cases is a backbone node. As expected, we see a remarkable improvement in case of NAPBACK. In particular, especially for lower network densities (which implies a smaller number of collisions, and hence re-transmissions), we observe improvements of up to 68.9% in the network lifetime.

Figure 2 shows the total residual energy left in the entire network. S-DMAC and NAPBACK show very little difference. This is because greater lifetime is achieved in NAPBACK, forcing the network to be operational for a longer time. As expected, DMAC being an energy draining protocol performs poorly compared to the other two (nobody goes to sleep in DMAC).

Figure 3 shows the average throughput achieved by the three protocols. While S-DMAC and NAPBACK maintain a comparable throughput, a steady degradation of performance is incurred in DMAC with increasing network density. With higher density, overhead in terms of beaconing and backbone maintenance increases, which severely affects the routing of the data packets to the sink in DMAC.

Figure 4 shows the average latency of data packet arrival at the sink. S-DMAC performs the best with a latency of 0.3secs in the worst case, while DMAC has comparable latency in sparse networks (e.g., networks with 100 nodes). The performance of DMAC deteriorates with increasing density and the latency reaches 20secs in the worst case. This is due to backbone maintenance overhead, that becomes dominating in case of high density networks (networks with 200 nodes and up), hampering the routing of data packets. NAPBACK latency lays in the middle. It averages around 12s in sparse networks (100 nodes) and around 10s in denser topologies (250 nodes). The best results are obtained for S-DMAC since the backbone nodes are always awake and sensed data can be immediately routed to the sink.

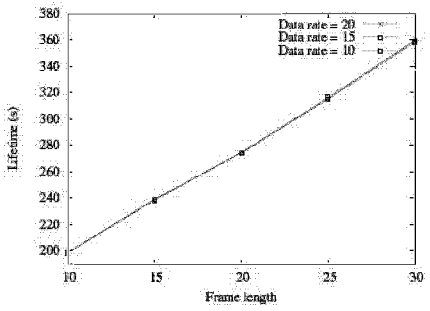

Fig. 5. Effect of frame length on network lifetime

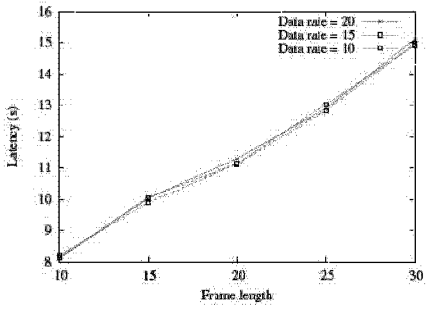

Fig. 6. Effect of frame length on average latency

## B. NAPBACK: Effect of frame length

Figure 5 and Figure 6 show the effect of frame length on the network lifetime and latency, respectively. The figures refer to the case of networks with 100 nodes, with data rates of 1pkt/(10, 15, 20)s, Nap-count set to 5 and Non-nap-period equal to .4s. The shorter the frame length, the smaller are lifetime and latency. For the same Nap-count, shorter frame length indicates greater periodicity with which schedules are calculated. This means that nodes wake up more frequently thus decreasing the network lifetime and average latency. We observed that the data rate has very little effect on the values obtained (so little that is not captured in the curves). This is due to the fact that the improvements are largely dependent on how much we can put the nodes too sleep, and not on the data rates as considered here.

Figure 7 shows the effect of frame length on the average throughput. For low data rates (e.g., 1 packet every 20s) the throughput deteriorates gradually with increasing frame length. For larger frame length, nodes have less common Non-nap-periods (i.e., periods when they and their neighbors are awake) with their upstream neighbors. For higher data rates, nodes experience collisions while routing the data packets through the upstream nodes when frames are shorter. For shorter frames and the same Nap-count, nodes now wake up more and there is a higher likelihood of collisions, leading to lower throughput. In all the cases, however, there

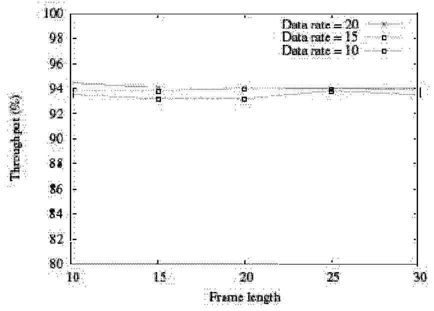

Fig. 7. Effect of frame length on average data delivery ratio at the sink. (data rate=1pkt/(10, 15, 20sec), nodes=100, Nap-count=5)

is not much of a difference in the throughput values and the obtained results lie between 93.3 and 94.5%.

We have also investigated the effect of Nap-count on the performance of NAPBACK when the frame length is set to 20s. We have observed a steady decrease in network lifetime and latency with increasing Nap-count (values range from 3 to 10). This is because nodes wake up more frequently within the same fixed frame length. As observed before, varying data rates have very little effect. We have also investigated the impact of different values of Nap-count on the throughput. For all three considered data rates, the values lie between 93 and 94%. The fluctuations in the obtained results is the least for moderate data rate of 1 packet every 15secs.

## IV. Conclusions

In this study we have investigated the effectiveness of building "napping backbones" for data dissemination in wireless sensor networks. The NAPBACK protocol builds connected backbones whose nodes are endowed with a sleep/awake schedule that induces considerable energy savings, and hence prolongs the network lifetime. Via simulations on networks with up to 250 nodes we have observed increases on network lifetime up to almost 70% with respect to previous topology control protocols (S-DMAC). Increased latency is the price to pay for the improvements on lifetime, which currently makes NAPBACK a viable solution for delay-insensitive WSN applications.

## References

[1] S. Basagni, A. Carosi, and C. Petrioli. Sensor-DMAC: Dynamic topology control for wireless sensor network. In *Proceedings of the 60th IEEE Vehicular Technology Conference, VTC 2004 Fall*, Los Angles, CA, September 26–29 2004.

[2] S. Basagni. Distributed clustering for ad hoc networks. In A. Y. Zomaya, D. F. Hsu, O. Ibarra, S. Origuchi, D. Nassimi, and M. Palis, editors, *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, pages 310–315, Perth/Fremantle, Australia, June 23–25 1999. IEEE Computer Society.

[3] Y. Hu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th ACM Annual International Conference on Mobile Computing and Networking*, pages 70–84, Rome, Italy, July 16–21 2001.

[4] The VINT Project. *The ns Manual*. http://www.isi.edu/nsnam/ns/, 2002.