International Journal On Engineering Technology and Sciences – IJETS™
ISSN (P): 2349-3968, ISSN (O): 2349-3976
Volume 1 Issue 6, October 2014

REDUCING THE TIME DELAY OF BACKBONE NETWORKS

M. Subashinidevi
P.G. Scholar, Dept. of CSE
A.S.L. Pauls CET
Solavampalayam, Coimbatore
msubashinidevi@gmail.com

S. Castro
Asst. Prof., Dept. of CSE
A.S.L. Pauls CET
Solavampalayam, Coimbatore
suseelcastro@gmail.com

Prof. M. Senthil Kumar
Associate Prof. & Head
Ranganathan Engg. College
REC Kalvi Nagar, Coimbatore
kathir_senthil@yahoo.co.in

Abstract: Various kinds of sleep wake scheduling protocols have been proposed in the literature. Synchronized sleep wake scheduling protocols have been proposed in. In these protocols, sensor nodes periodically or a periodically exchange synchronization information with neighboring nodes. However, such synchronization procedures could incur additional communication overhead and consume a considerable amount of energy. On-demand sleep wake scheduling protocols have been proposed, where nodes turn off most of their circuitry and always turn on a secondary low-powered receiver to listen to "wake-up" calls from neighboring nodes when there is a need for relaying packets. However, this on-demand sleep wake scheduling can significantly increase the cost of sensor motes due to the additional receiver. In this paper, we are interested in asynchronous sleep wake scheduling protocols such as those proposed. In these protocols, each node wakes up independently of neighboring nodes in order to save energy. However, due to the independence of the wake-up processes, additional delays are incurred at each node along the path to the sink because each node needs to wait for its next-hop node to wake up before it can transmit the packet. This delay could be unacceptable for delay-sensitive applications, such as fire detection or a tsunami alarm, which require the event reporting delay to be small.

Index Terms*:* Sleep-wake, Sensor node, Wireless Network s, Energy.

1. INTRODUCTION

Energy conservation is one of the key technical challenges in sensor networks and ad hoc networks. It is necessary to devise communications and networking schemes which make judicious use of the limited energy resources without compromising the network connectivity and the ability to deliver data to the intended destination. In addition, sensor nodes are often subject to further constraints in terms of CPU power, memory space, etc., which call for simple algorithms and schemes whose memory needs are modest.

One of the main mechanisms to save energy is the use of sleep modes at the MAC layer, by which nodes are put to sleep as often as possible. This must be done in such a way that connectivity is preserved since, if too many nodes are sleeping at the same time, the network may end up being disconnected. In the recent literature, several schemes have been proposed which address this problem. For example, SPAN tries to coordinate the sleeping activity of the nodes so that a connecting backbone is always present. GAF identifies groups of nodes which are equivalent from a routing point of view, i.e., in each group, it is sufficient that a single node is awake at any given time. STEM, on the other hand, provides a means to communicate with a node currently asleep, by implementing a rendezvous mechanism based on beacon transmissions. As to the MAC itself, most papers in the literature assume either TDMA-based schemes [4] or multi channel setups in which parallel transmissions can be performed without interference or variants of classic contention-based schemes, usually based on RTS/CTS handshake in order to mitigate the hidden terminal problem.

However, anycast does not necessarily lead to the minimum expected end-to-end delay because a packet can still be relayed through a time-consuming routing path. Therefore, the first challenge for minimizing the expected end-to-end delay is to determine how each node should choose its anycast forwarding policy (e.g., the forwarding set) carefully. The work in proposes heuristic anycast protocols that exploit the geographical distance to the sink node. However, these solutions are heuristic in nature and do not directly minimize the expected end-to-end delay. The algorithms use the hop-count information (i.e., the number of hops for each node to reach the sink) to minimize some state-dependent cost (delay) metric along the possible routing paths. However, these algorithms do not directly apply to asynchronous sleep–wake scheduling, where each node does not know the wake-up schedule of neighboring nodes when it has a packet to forward.

The second challenge stems from the fact that good performance cannot be obtained by studying the anycast forwarding policy in isolation. Rather, it should be jointly controlled with the parameters of sleep–wake scheduling. Note that the latter will directly impact both network lifetime and the packet-delivery delay. Hence, to optimally tradeoff network lifetime and delay, both the wake-up rates and the anycast packet-forwarding policy should be jointly controlled. However, such interactions have not been systematically studied in the literature. In this paper, we address these challenges. We first investigate the delay-minimization problem: given the wake-up rates of the sensor nodes, how to optimally choose the anycast forwarding policy to minimize the expected end-to-end delay from all sensor nodes to the sink. We develop a low-complexity and distributed solution to this problem. We then formulate the lifetime maximization problem: given a constraint on the expected end-to-end delay, how to maximize the network lifetime by jointly controlling the wake-up rates and the anycast packet-forwarding policy. We show how to use the solution to the delay-minimization problem to construct an optimal solution to the lifetime-maximization problem for a specific definition of network lifetime.

Before we present the details of our problem formulation and the solution, we make a note regarding when the anycast protocols and the above optimization algorithms are applied. We can view the lifetime of an event-driven sensor network as consisting of two phases: the configuration phase and the operation phase. When nodes are deployed, the configuration phase begins, during which nodes optimize the control parameters of the anycast forwarding policy and their wake-up rates. It is during this phase that the optimization algorithms discussed in the last paragraph will be executed. In this phase, sensor nodes do not even need to follow asynchronous sleep–wake patterns. After the configuration phase, the operation phase follows. In the operation phase, each node alternates between two subphases, i.e., the sleeping subphase and the event-reporting subphase. In the sleeping subphase, each node simply follows the sleep–wake pattern determined in the configuration phase, waiting for events to occur. Note that since we are interested in asynchronous sleep–wake scheduling protocols, the sensor nodes do not exchange synchronization messages in this sleeping subphase.

Finally, when an event occurs, the information needs to be passed on to the sink as soon as possible, which becomes the event-reporting subphase. It is in this event reporting subphase when the anycast forwarding protocol is actually applied, using the control parameters chosen during the configuration phase. Note that the configuration phase only needs to be executed once because we assume that the fraction of energy consumed due to the transmission of data is negligible. However, if this is not the case, the

transmission energy will play a bigger role in reducing the residual energy at each node in the network. In this case, as long as the fraction of energy consumed due to data transmission is still small (but not negligible), the practical approach would be for the sink to initiate a new configuration phase after a long time has passed.

2. SYSTEM MODEL

We consider a wireless sensor network with N nodes. Let N denote the set of all nodes in the network. Each sensor node is in charge of both detecting events and relaying packets. If a node detects an event, the node packs the event information into a packet, and delivers the packet to a sink s via multihop relaying. We assume in this paper that there is a single sink; however, the analysis can be generalized to the case with multiple sinks.

2.1 Transmitter

When a sleeping node has a packet to send, it enters the active state and monitors both frequencies for *T* seconds. If either frequency is busy, the node backs off and reschedules an attempt at a later time. If, on the other hand, both frequencies are sensed idle during this entire interval, the node transmits a broadcast RTS message which contains the location of the intended destination as well as its own. After sending the RTS, the transmitting node listens in the subsequent slots for CTS messages from potential relays. In each of the CTS slots following the end of the RTS message, the transmitting node acts as follows:

1. If only one CTS message is received, it starts transmission of the data packet whose initial part acts as a CTS confirmation for the node which issued the CTS.

2. If it receives no CTSs, it will send a CONTINUE message and listen again for CTSs, timing out after Np empty CTS slots (which forces the node to abort the handshake and to reschedule it at a later time).

3. If it hears a signal but is unable to detect a meaningful message, it will assume that a CTS collision took place and will send a COLLISION message, which will trigger the start of a collision resolution algorithm (to be described later) and will listen again for CTSs.

After packet transmission, an immediate ACK is expected. If it is correctly received, it completes the transaction and the node can go back to sleep. Notice that if the receiver is an intermediate node toward the destination, in a scheme in which a packet exchange is immediately initiated, the RTS message itself could serve as the ACK. Here, on the other hand, we assume that an explicit ACK is used. If the transmitter does not get an ACK within a given time, it times out and declares the transaction failed. It will then reschedule the same packet for future transmission. After NMaxAtt failed attempts for the same packet, the transmitter will give it up and generate an error message for the higher layers. While listening for CTSs and for the ACK, the node transmits the busy tone to prevent interference from hidden terminals.

Notice that, with the above rules, the protocol does not lead to transmitter deadlock as the sender will never wait indefinitely for CTSs or ACKs. Only in the case of a completed transaction will the transmitter consider the packet as successfully passed to the next hop. A remaining problem with this scheme is that packet duplication may occur. In fact, if the final ACK is lost, the relay node is now in charge of packet delivery, whereas the transmitter will not be aware of this fact and will retry the transmission of the same packet. This ambiguity does not compromise the correctness of the scheme and can be solved by intermediate nodes when an additional copy of the same packet is received and discarded. This requires that

nodes keep the memory of recent transmissions. If this is not possible or desirable, as well as in the case in which multiple copies of the same packet go through disjoint sets of nodes, packet duplication will be detected at the destination, which leads to some inefficiency that, on the other hand, is mitigated by the fact that losing an ACK when the packet was successful is a low probability event and the overall performance impact may be expected to be limited.

2.2 Receiver

Each node will (more or less) periodically wake up and put itself in the listening mode. If nothing happens throughout the listening time, whose duration may be fixed or random, the node goes back to sleep. On the other hand, if the node detects the start of a transmission, it goes into the receiving state. Note that the randomness of the events involved makes the sleep process not exactly periodic. The sleep time will be considered as a constant in the following, for convenience of explanation and of analysis. In reality, more sophisticated schemes could be envisioned in which sleep times could be random or could depend on the battery status.

Upon detecting the start of a message, a listening node starts receiving. At the same time, it activates the busy tone on the busy tone frequency for a duration TRTS. If no valid RTS is received, the node goes back to the listening state, where it stays for the originally scheduled duration. On the other hand, if a valid RTS is received, the node reads the information in it and determines its own priority as a relay.

If, on the other hand, a COLLISION message is received, this means that more than one CTS was generated in the CTS slot. All nodes who did not transmit will drop out while those involved in the collision will start the collision resolution algorithm. Each colliding node will decide with probability 0.5 whether or not to continue. Who decides to continue will send a CTS in the next slot. Three events are possible:

1. Only one node sends, transmitter starts packet transmission, and all others drop;

2. More than one CTSs are sent in the same slot, transmitter sends a COLLISION message, those who did not send drop out, those involved in the collision decide whether or not to continue as before, until the collision is resolved;

3. No CTS is heard, a CONTINUE message is sent by the transmitter, and all nodes who did not select the current slot decide again independently whether to continue as before.

This procedure will terminate in few slots with high probability. In order to force it to be limited, the transmitter can send an ABORT message if the collision is not resolved within $N_{MaxColl}$ CTS slots. Finally, any node which receives a message it does not understand will drop out.

Nodes which heard the RTS correctly will follow the sequence of steps above and they are guaranteed to either become the relay node or to drop out at some point. The event that two nodes think they are the designated relay can be completely avoided if the start of the packet contains the full relay node's address or made very unlikely in a simplified scheme where, at the start of the packet, a random number (included in the CTS as temporary short address) is transmitted. In order to avoid the hidden terminal problem, each node involved in the above procedure will keep the busy tone active until it drops out or, if it is the winner, until the whole data packet has been received.

3. DESIGN OVERVIEW

S-MAC includes approaches to reduce energy consumption from all the sources of energy waste that we have identified, i.e., idle listening, collision, overhearing and control overhead. Before describing the components in S-MAC, we first summarize our

assumptions about the wireless sensor network and its applications. Sensor networks will consist of large numbers of nodes to take advantage of short-range, multihop communications to conserve energy. Most communications will occur between nodes as peers, rather than to a single base station. In-network processing is critical to network lifetime, and implies that data will be processed as whole messages in a store-and-forward fashion. Packet or fragment-level interleaving from multiple sources only increases overall latency. Finally, we expect that applications will have long idle periods and can tolerate latency on the order of network messaging time.

3.1 Periodic Listen and Sleep

As stated above, in many sensor network applications, nodes are idle for long time if no sensing event happens. Given the fact that the data rate is very low during this period, it is not necessary to keep nodes listening all the time. S-MAC reduces the listen time by putting nodes into periodic sleep state.

We call a complete cycle of listen and sleep a frame. The listen interval is normally fixed according to physical-layer and MAC layer parameters, e.g., the radio bandwidth and the contention window size. The duty cycle is defined as the ratio of the listen interval to the frame length. The sleep interval can be changed according to different application requirements, which actually changes the duty cycle. For simplicity, these values are the same for all nodes.

All nodes are free to choose their own listen/sleep schedules. However, to reduce control overhead, we prefer neighboring nodes to synchronize together. That is, they listen at the same time and go to sleep at the same time. It should be noticed that not all neighboring nodes can synchronize together in a multihop network.

3.2 Collision Avoidance

If multiple neighbors want to talk to a node at the same time, they will try to send when the node starts listening. In this case, they need to contend for the medium. Among contention protocols, the 802.11 does a very good job on collision avoidance. S-MAC follows similar procedures, including virtual and physical carrier sense, and the RTS/CTS exchange for the hidden terminal problem. There is a duration field in each transmitted packet that indicates how long the remaining transmission will be. If a node receives a packet destined to another node, it knows how long to keep silent from this field. The node records this value in a variable called the network allocation vector and sets a timer for it. Every time when the timer fires, the node decrements its NAV until it reaches zero. Before initiating a transmission, a node first looks at its NAV. If its value is not zero, the node determines that the medium is busy. This is called virtual carrier sense.

Physical carrier sense is performed at the physical layer by listening to the channel for possible transmissions. Carrier sense time is randomized within a contention window to avoid collisions and starvations. The medium is determined as free if both virtual and physical carrier sense indicates that it is free. All senders perform carrier sense before initiating a transmission. If a node fails to get the medium, it goes to sleep and wakes up when the receiver is free and listening again. Broadcast packets are sent without using RTS/CTS. Unicast packets follow the sequence of RTS/CTS/DATA/ACK between the sender and the receiver. After the successful exchange of RTS and

CTS, the two nodes will use their normal sleep time for data packet transmission. They do not follow their sleep schedules until they finish the transmission. With the low-duty-cycle operation and the contention mechanism during each listen interval, S-MAC effectively addresses the energy waste due to idle listening and collisions. In the next section, we will present details of the periodic sleep coordinated among neighboring nodes. Then we will present two techniques that further reduce the energy waste due to overhearing and control overhead.

4. COORDINATED SLEEPING

Periodic sleeping effectively reduces energy waste on idle listening. In S-MAC, nodes coordinate their sleep schedules rather than randomly sleep on their own. This section details the procedures that all nodes follow to set up and maintain their schedules.

It also presents a technique to reduce latency due to the periodic sleep on each node.

4.1 Choosing and Maintaining Schedules

Before each node starts its periodic listen and sleep, it needs to choose a schedule and exchange it with its neighbors. Each node maintains a schedule table that stores the schedules of all its known neighbors. It follows the steps below to choose its schedule and establish its schedule table.

1) A node first listens for a fixed amount of time, which is at least the synchronization period. If it does not hear a schedule from another node, it immediately chooses its own schedule and starts to follow it. Meanwhile, the node tries to announce the schedule by broadcasting a SYNC packet. Broadcasting a SYNC packet follows the normal contention procedure. The randomized carrier sense time reduces the chance of collisions on SYNC packets.

2) If the node receives a schedule from a neighbor before choosing or announcing its own schedule, it follows that schedule by setting its schedule to be the same. Then the node will try to announce its schedule at its next scheduled listen time.

3) There are two cases if a node receives a different schedule after it chooses and announces its own schedule. If the node has no other neighbors, it will discard its current schedule and follow the new one. If the node already follows a schedule with one or more neighbors, it adopts both schedules by waking up at the listen intervals of the two schedules.

(Fig. 1) Neighboring nodes A and B have different schedules. They synchronize with nodes C and D respectively.

To illustrate this algorithm, consider a network where all nodes can hear each other. The node that starts first will pick up a schedule first, and its broadcast will synchronize all its peers on its schedule. If two or more nodes start first at the same time, they will finish initial listening at the same time, and will choose the same schedule independently. No matter which node sends out its SYNC packet first (wins the contention), it will synchronize the rest of the nodes. However, two nodes may independently assign schedules if they cannot hear each other in a multihop network. In this case, those nodes on the border of two schedules will adopt both. For example, nodes A and B in Fig. 1 will wake up at the listen time of both schedules. In this way, when a border node sends a broadcast packet, it only needs to send it once. The disadvantage is that

these border nodes have less time to sleep and consume more energy than others.

Another option is to let a border node adopt only one schedule the one it receives first. Since it knows that some other neighbors follow another schedule, it can still talk to them. However, for broadcasting, it needs to send twice to the two different schedules. The advantage is that the border nodes have the same simple pattern of periodic listen and sleep as other nodes.

We expect that nodes only rarely see multiple schedules, since each node tries to follow an existing schedule before choosing an independent one. However, a new node may still fail to discover an existing neighbor for a few reasons. The SYNC packet from the neighbor could be corrupted by collisions or interference. The neighbor may have delayed sending a SYNC packet due to the busy medium. If the new node is on the border of two schedules, it may only discover the first one if the two schedules do not overlap. To prevent the case that two neighbors miss each other forever when they follow completely different schedules, S-MAC introduces periodic neighbor discovery, i.e., each node periodically listens for the whole synchronization period.

The frequency with which a node performs neighbor discovery depends on the number of neighbors it has. If a node does not have any neighbor, it performs neighbor discovery more aggressively than in the case that it has many neighbors. Since the energy cost is high during the neighbor discovery, it should not be performed too often. In our current implementation, the synchronization period is 10 s, and a node performs neighbor discovery every 2 min if it has at least one neighbor.

5. MAXIMIZATION OF NETWORK LIFETIME

In the previous section, we solved the delay-minimization problem. In this section, we use the result to develop a solution to the lifetime-maximization problem (P). Hence, we convert Problem (P) to the following equivalent problem that controls $\vec{T} = (T_1, T_2, \cdots, T_N)$, $\mathbf{A} \in \mathcal{A}$, and $\mathbf{B} \in \mathcal{B}$:

$$\begin{aligned} \textbf{(P1)} \quad & \max_{\vec{T} \in (\mathbb{R}^+)^N, \mathbf{A}, \mathbf{B}} \ \min_{i \in \mathcal{N}} T_i \\ & \text{subject to } D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \le \xi^*, \ \forall i \in \mathcal{N} \\ & p_i = 1 - e^{-\frac{t_l}{e_i T_i}}, \quad \forall i \in \mathcal{N} \end{aligned}$$

Where $\mathbb{R}^+ = (0, \infty)$. For any given $\vec{p}$, $(\mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p}))$ is the optimal anycast policy that minimizes the delay from all nodes,

$$\text{i.e., } D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \le D_i(\vec{p}, \mathbf{A}, \mathbf{B}) \text{ for all } (\mathbf{A}, \mathbf{B}).$$

Hence, we can rewrite Problem (P1) as follows:

$$\begin{aligned} \textbf{(P2)} \quad & \max_{\vec{T} \in (\mathbb{R}^+)^N} \ \min_{i \in \mathcal{N}} T_i \\ & \text{subject to } D_i(\vec{p}, \mathbf{A}^*(\vec{p}), \mathbf{B}^*(\vec{p})) \le \xi^*, \ \forall i \in \mathcal{N} \\ & p_i = 1 - e^{-\frac{t_l}{e_i T_i}}, \ \forall i \in \mathcal{N}. \end{aligned}$$

6. CONCLUSION

In this paper, we develop an anycast packet-forwarding scheme to reduce the event-reporting delay and to prolong the lifetime of wireless sensor networks employing asynchronous sleep–wake scheduling. Specifically, we study two optimization problems. First, when the wake-up rates of the sensor nodes are given, we develop an efficient and distributed algorithm to minimize the expected event-reporting delay from all sensor nodes to the sink. Second, using a specific definition of the network lifetime, we study the lifetime-maximization problem to optimally control the sleep–wake scheduling policy and the anycast policy in order to maximize the network lifetime subject to an upper limit on the expected end-to-end delay.

REFERENCES

[1] E. Yetginer, T. Uekae, and I. Tan Kislasi, "Power efficient traffic grooming in optical WDM networks," 2009 IEEE GLOBECOM.

[2] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links," 2010 ACM SIGCOMM Workshop on Green Networking.

[3] A. Coiro, M. Listanti, A. Valenti, and F. Matera, "Reducing power consumption in wavelength routed networks by selective switch off of optical links," IEEE J. Sel. Topics Quantum Electronics, vol. 17, no. 2, Mar./Apr. 2011.

[4] J. Kim, X. Lin, N. B. Shroff, and P. Sinha, "On maximizing the lifetime of delay-sensitive wireless sensor networks with anycast," in Proc. IEEE INFOCOM, Pheonix, AZ, Apr. 2008, pp. 807–815.

[5] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," Comput. Netw., vol. 43, pp. 317–337, Oct. 2003.

[6] W. Ye, H. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," IEEE/ACM Trans. Netw., vol. 12, no. 3, pp. 493–506, Jun. 2004.

[7] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in Proc. SenSys, Nov. 2003, pp. 171–180.

[8] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in Proc. IPDPS, Apr. 2004, pp. 224–231.

[9] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pp. 147–163, 2002.

[10] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A.Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy- efficient wireless sensor networks," in Proc. MobiCom, 2001, pp. 272–287.

AUTHOR DETAILS

M. Subashinidevi was born in Virudhunagar District, Tamil Nadu, India in 1980. She completed her B.Sc., M.Sc., and M.Phil. courses in Computer Science. She has over 5 years of industrial experience. Presently, she is pursuing her M.E. Computer Science and Engineering in A.S.L. Pauls College of Engineering and Technology, affiliated to Anna University, Chennai. Her area of interest includes Data Mining and Wireless Networks.

Prof. M. Senthil Kumar was born in Ramanathapuram District, Tamil Nadu, India in 1982. He obtained his B.Sc., M.Sc. and M.Tech. Degrees in Electronics in the years 2002, 2004 and 2006 respectively. He has more than 8 years of teaching experience. He has presented more than 30 research papers in various national and international conferences. He has also published more than 15 research papers in reputed international journals. He has guided several UG and PG students for their project work. His area of interest is Energy Conservation and Optimization Techniques in Wireless Sensor Networks. Currently, he is with Ranganathan Engineering College, Coimbatore, India, as Associate Professor and Head of the Department of Electronics and Communication Engineering.