Innovations Syst Softw Eng (2005)
DOI 10.1007/s11334-005-0015-z

REGULAR ARTICLE

Rajesh Mathew · Mohamed Younis · Sameh M. Elsharkawy

Energy-efficient bootstrapping for wireless sensor networks

Received: ■ / Accepted: ■ / Published online: ■

Abstract Wireless sensor networks are poised for increasingly wider uses in many military and civil applications. Such applications has stimulated research in a number of research areas related to energy conservation in such networks. Most such research focuses on energy saving in tasks after the network has been organized. Very little attention has been paid to network bootstrapping as a possible phase where energy can be saved. Bootstrapping is the phase in which the entities in a network are made aware of the presence of all or some of the other entities in the network. This paper describes a bootstrapping protocol for a class of sensor networks consisting of a mix of low-energy sensor nodes and a small number of high-energy entities called gateways. We propose a new approach, namely the slotted sensor bootstrapping (SSB) protocol, which focuses on avoiding collisions in the bootstrapping phase and emphasizes turning off node radio circuits whenever possible to save energy. Our mechanism synchronizes the sensor nodes to the gateway's clock so that time-based communication can be used. The proposed SSB protocol tackles the issue of node coverage in scenarios, when physical device limitations and security precautions prevent some sensor nodes from communicating with the gateways. Additionally, we present an extension of the bootstrapping protocol, which leverages possible gateway mobility.

Keywords Sensor networks · Network bootstrapping · Energy-aware design · Communication protocols

R. Mathew · M. Younis
Dept. of Computer Science and Electrical Eng.,
University of Maryland Baltimore County,
1000 Hilltop Circle, Baltimore, MD 21250, USA
E-mail: rmathe3@umbc.edu, younis@cs.umbc.edu
Tel.: +410-4553968

S.M. Elsharkawy
Dept. of Electrical Eng. and Computer Science,
The Catholic University of America,
620 Michigan Ave. NE, Washington, DC 20064, USA
E-mail: elsharkawy@cua.edu
Tel.: +202-3194620

1 Introduction

There has been growing interest in recent years in the potential of sensor networks in a variety of military and civil applications, such as target tracking and geographical studies. With the aid of sensor devices, such tasks can be accomplished with limited human intervention. Military reconnaissance can be done remotely using scattered low-energy sensing devices in the zone where targets are expected, and thus data can be gathered from hostile environments at low cost and minimal risk. In addition, sensor nodes are gaining popularity in geographical studies in hostile terrain, or in studies where terrain is not necessarily hostile but large amounts of readings are required, while at the same time keeping hardware costs under acceptable limits, thereby necessitating very light devices with limited communication and computation capabilities.

An important characteristic of sensor devices is that their battery capacity is very small, much smaller than conventional wireless devices like laptops and even PDAs, thereby making energy conservation one of the most important issues in sensor network research. Sensor nodes are usually equipped with short-haul radios, and communication accounts for a major portion of energy usage. Therefore, energy-efficient communication protocols are essential for sensor networks in order to enhance their robustness and extend systems' lifetimes [1–6].

Typically, a set of sensors is spread throughout an area of interest in order to detect and possibly track events/targets in this area. The sensing circuitry probes the surrounding environment. After performing signal processing of the observed data, sensors communicate these data to a command center usually through a relay or a data concentrator called a gateway. Gateway nodes are capable of long-haul communication and have richer energy resources compared to sensor nodes. The gateway can perform a host of other tasks such as arbitration of medium access and generating routing tables [7]. Due to scalability requirements and a desire to avoid overloading the gateway, network clustering is recommended through the involvement of multiple gateways, as

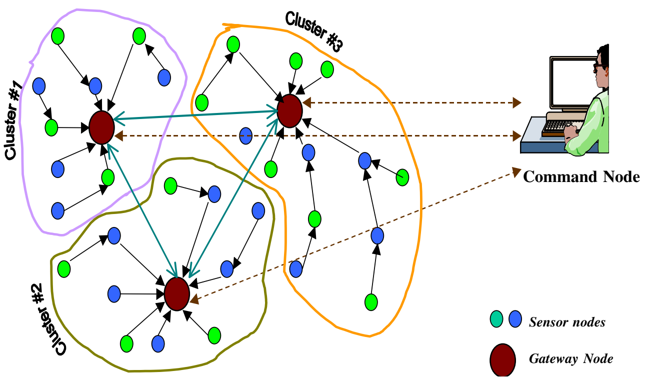

Fig. 1 Multigateway clustered sensor network

shown in Fig. 1. Clusters are formed such that its gateway is located within the communication range of all the sensors in its cluster.

Before a network is clustered, sensors and gateway nodes need to be informed about the presence of their surrounding nodes. Bootstrapping the sensor network refers to the discovery of deployed sensors and establishing single-or multihop communication links between each gateway and sensors that are accessible to it. Bootstrapping in sensor networks can be very challenging because human intervention in setting up and administering the network is not possible for many of the applications that sensor networks are used for. As we elaborate in the sections ahead, inefficient bootstrapping protocols can consume a substantial portion of the very limited sensor energy. In many sensor networks, battery capacities cannot be replenished and replacement of batteries is not feasible. Every unit of energy saved by optimal use increases the lifetime and, consequently, the utility of the network. The complexity of bootstrapping significantly diminishes if energy is not an issue and a number of mechanisms may be used. We show in this paper that bootstrapping can be a significant energy-consuming process and needs to be studied in greater details. We present an energy-efficient bootstrapping protocol that conserves sensor energy and synchronizes sensor clocks, thereby reaping the benefits of synchronization at this early phase of network operation.

1.1 Problem statement

Sensor discovery can typically be performed through repetitive sensor beaconing or through continual probing for the gateway. In repetitive beaconing each sensor broadcasts hello messages to discover its neighbors and flood such information to the gateway. Such an approach involves many, and often excessive, messages, making it an energy burden. In addition, it will not suit setups in which the gateway is deployed a while after the sensors are deployed. Such a scenario is expected in many applications like combat field surveillance and disaster management. Moreover, repetitive sensor beaconing requires an initial intersensor trust, which may allow unauthorized nodes to join the network or even hinder the node discovery process, a scenario that is totally unacceptable in security-sensitive applications. On the other hand, probing for gateway announcements and establishing direct contact with it can be very energy efficient and secure. However, gateway-based protocols for node discovery typically need sensors to keep their receivers ON during the discovery phase waiting for an indication that the gateway has begun operation. In many cases imposing this requirement on the sensors causes a substantial portion of the sensors' battery to be consumed until the network is bootstrapped and the nodes are recognized by the gateways. Furthermore, sensors may deplete the bulk of their batteries if there is a significant time lag between the deployment of sensors and gateways.

When sensors keep their receivers active after deployment, the gateways can send out announcements to indicate their presence and start network bootstrapping. Such a scheme, as depicted in Fig. 2, has one of the gateways transmit information indicating that the nodes are now permitted to reply. The sensor nodes then begin to transmit their replies. As there is no sense of time base at this point in time, the sensors often face collision and have to retransmit the reply message. One possible strategy for countering collision has the sensor node pick a random time after the gateway's announcement to reply and, on detecting a collision at a predetermined time, sleep for a random interval and repeat the entire procedure. We will consider this protocol as the baseline to which we compare the performance of our energy slotted sensor bootstrapping (SSB) efficient protocol.

When sensors are deployed earlier than the gateways, the sensors keep trying to bootstrap repeatedly but succeed only after the gateways are deployed. In some scenarios, a number of sensor nodes may not hear the gateway's announcement when they are deployed later. Hence, if the gateway comes up before the sensors and sends out an announcement that is missed by the sensors, the sensors will be left with their receivers ON, unaware that the gateway has already sent out its announcement. This would lead to battery depletion. Moreover, the effect of collisions on sensor transmissions can be very dramatic. Not only will unorganized sensors' transmissions extend the bootstrapping time and increase energy consumption, but they can also lead to long periods of failed retransmissions during which gateways cannot even exchange messages among themselves or communicate with other sensors. The importance of solving this problem is further underscored when we consider scenarios where the number of nodes may run into the hundreds of thousands.[1]

1.2 Our contribution

In this paper, we present *Slotted Sensor Bootstrapping (SSB)*, an energy-efficient, highly scalable bootstrapping protocol.

[1] Wireless Ad Hoc Networks: Smart Sensor Networks, `http://w3.antd.nist.gov/wahn_ssn.shtml`

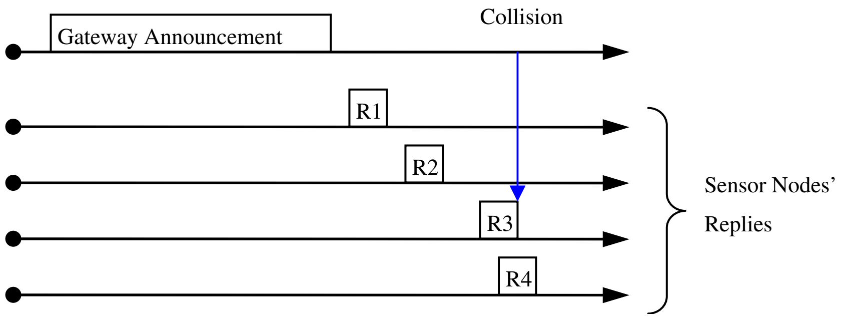

Fig. 2 Collisions in bootstrapping phase

The moot concept of the SSB protocol is that we permit the sensors to intermittently switch their receivers ON and OFF during the bootstrapping phase, thereby accruing considerable savings in battery power. If the gateway is deployed after the sensors, the SSB mechanism becomes even more effective. To achieve this, we introduce time-based synchronization during the bootstrapping phase, thereby reaping the benefits of such synchronization through network operation instead of synchronizing the nodes at a later stage, e.g., after network clustering. This feature helps in making the protocol very efficient in terms of energy consumption. We also leverage such synchronization at various points during the operation of the protocol.

At the end of the bootstrapping phase, each gateway should know the set of reachable sensors and synchronize their clocks with its own clock. Clock synchronization enables the use of a time-multiplexed model of communication, such as TDMA. TDMA-based medium access control has been shown to be very energy efficient for sensor networks [8,9]. At the end of the network bootstrapping, the nodes are already synchronized with the gateways, and hence it is easier to incorporate TDMA-like schemes at this juncture.

We have extended the bootstrapping protocol to benefit from possible gateway mobility so that node coverage is enhanced. Sensor nodes often have physical limitations as far as transmission capabilities are concerned that intrinsically reduce the ratio of nodes that may be discovered as compared to the total number of nodes actually present in the network. If gateways possess motion capabilities, a greater number of nodes can be discovered during bootstrapping. We have developed an algorithm to determine appropriate motion patterns for the gateway to efficiently discover additional sensors that could not be reached initially.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the details of the SSB protocol. In this section, we also present simulation results that demonstrate the efficiency and scalability of the proposed SSB scheme. Section 4 looks at the problem of deciding motion patterns for enhanced node discovery in scenarios where gateways possess motion capabilities. Section 5 concludes the paper and points out future research directions.

2 Related work

Energy constraints distinguish sensor networks from other wireless communication networks. Such constraints have motivated research at different layers of the protocol stack and in other related problems such as clustering, routing, and data aggregation. However, most such research pays little attention to energy efficiency in the bootstrapping phase. In this paper, we make the case that the bootstrapping phase can potentially be an expensive one in terms of energy and, hence, is one that merits a closer look than it currently receives.

Time-based bootstrapping of sensor networks has been studied in [10]. In that paper, the authors use the notion of a "frame" in time, which is divided into many parts serving different functionalities. The schedule within a frame is fixed and includes slots for all types of communication. A recently powered-up sensor will listen to traffic to find neighbors and engage with them in order to join the network. A gradual increase in transmission range strategy, known as incremental shouting, is applied in [11] for sensor discovery. Every node sends announcements with increasing energy to explore the neighborhood. Close sensors, which hear such announcements, would acknowledge the message and indicate their location. The process is repeated by every sensor and stops when every sensor knows a preset number of neighbors and considers them its group. A group can take on sensing tasks and arbitrate the load among the members. The same approach is applied to groups in order to form a network topology. Such approaches are so complicated that they may consume lots of energy by themselves. In addition, there is no clear termination condition, which makes their convergence questionable.

Approaches for sensor discovery and location determination were suggested in [12]. The basic idea is to use beacons to make the presence of a sensor known to its neighbors and use multilateration to estimate the relative location of sensors. Absolute location can be calculated with the availability of reference sensors that are either hand-placed or equipped with GPS. We assume that every sensor knows its location, although in the future we would like to explore such ideas as a complementary procedure to the SSB protocol, using the gateways as the base reference locations.

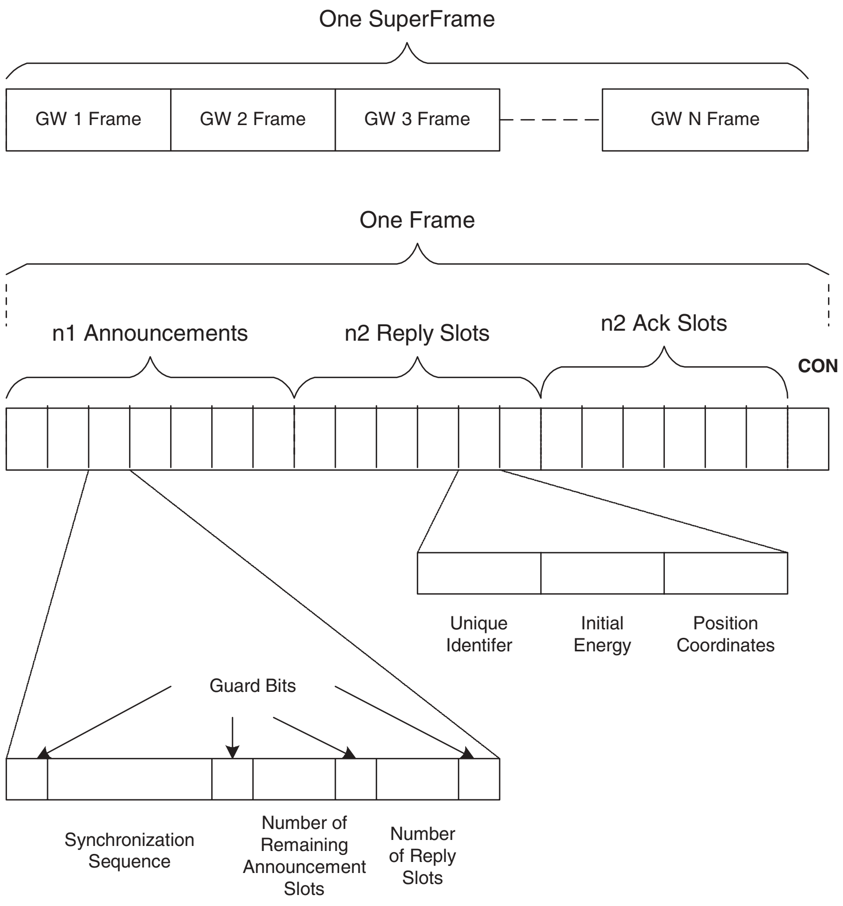

Fig. 3 Frames and superframes

With sensor devices being extremely energy constrained, it is beneficial to save energy at all possible phases of sensor's design and in the network operation, thus justifying the need for low-energy hardware design, energy-aware software, and energy-efficient communication protocols. Low-power electronics, power-down modes, and energy-efficient modulation are examples of hardware-based techniques [1,13,14]. Examples of energy-aware software includes operating systems such as TinyOS [15] and duty-cycle-controlled software execution [16].

Further, great focus is being placed on energy-efficient physical layer protocols. Energy saving through the use of time-based MAC in wireless sensor networks has been explored in [8,9,17,18]. The principle is to schedule activation of the radio receiver so that it can be turned off when a message is not expected. Turning off the receiver has been shown to achieve savings of up to 70% in energy consumption [17]. Approaches for determining when to turn off the receiver vary. While slots are prescheduled in [9,18], the decision for deactivating the receiving circuit is made autonomous in [17] by probing the environment. A reservation-based approach for scheduling medium access is pursued in [8]. Nodes make a request to a base station, which responds with a traffic control message indicating medium access schedule. Nodes not included in the traffic control message can turn off their receivers. Mechanisms like S-MAC [19, 20], a MAC protocol similar to TDMA, also provide solutions to the problem of optimizing the MAC layer. However, all published mechanisms address normal network operation and do not investigate the potential of time synchronization in the bootstrapping process.

Clustering a sensor network is important for increasing the scalability of the network, and many approaches to doing this have been proposed. Clusters can be assigned different frequencies or codes if FDM or CDMA is used, thus benefiting from spatial reuse of bandwidth [21]. The dynamic leader selection mechanism of LEACH [22] has the sensor nodes themselves decide whether they would be cluster heads or not, and other nodes form clusters around these "clusterheads." All these approaches help the sensors save energy.

In addition, energy-aware routing has received lots of attention [1,23]. A number of approaches—Minimum Total Power Transmission Power Routing, Minimum Battery Cost Routing, Min-Max Battery Cost Routing, and Conditional Max-Min Battery Capacity Routing—and their relative advantages and drawbacks are discussed in [24]. A survey can be found in [25].

Apart from algorithms, which advise nodes about routes to be used for transmission, there also exist algorithms that suggest precisely how much or what part of the available

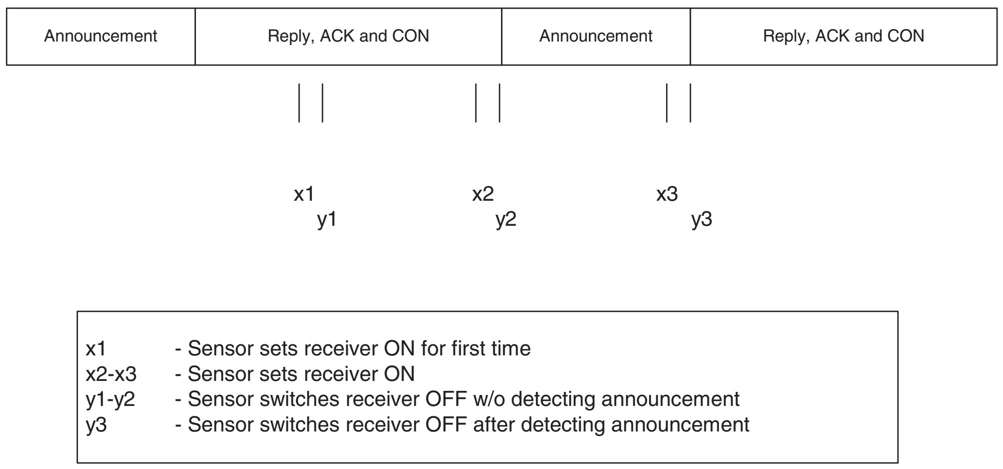
Announcement
Reply, ACK and CON
Announcement
Reply, ACK and CON
x1
y1
x2
y2
x3
y3
x1 - Sensor sets receiver ON for first time
x2-x3 - Sensor sets receiver ON
y1-y2 - Sensor switches receiver OFF w/o detecting announcement
y3 - Sensor switches receiver OFF after detecting announcement

Fig. 4 Node detection announcement within one frame if gateway is already deployed

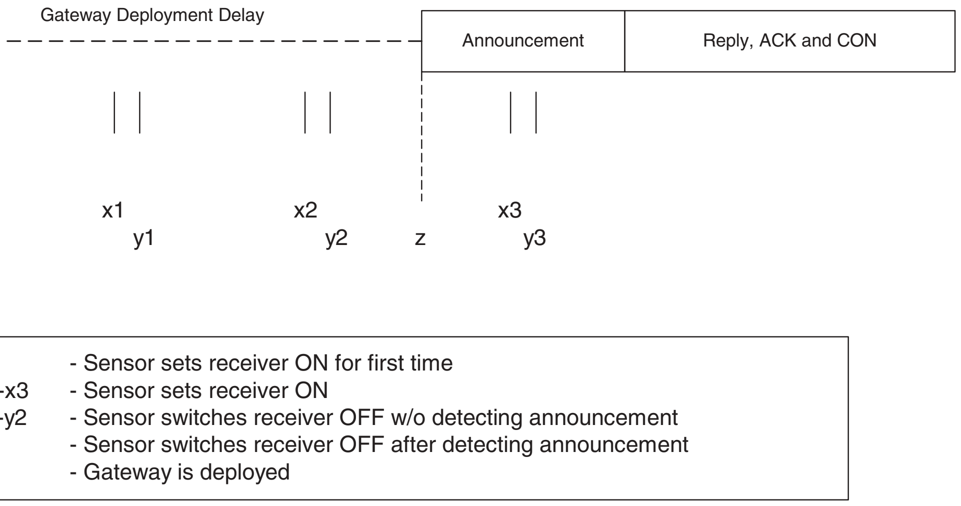
Gateway Deployment Delay
Announcement
Reply, ACK and CON
x1
y1
x2
y2
z
x3
y3
x1 - Sensor sets receiver ON for first time
x2-x3 - Sensor sets receiver ON
y1-y2 - Sensor switches receiver OFF w/o detecting announcement
y3 - Sensor switches receiver OFF after detecting announcement
z - Gateway is deployed

Fig. 5 Sensor's receiver ON time is minimized even if gateway is deployed late

information should be actually transmitted to the gateway. Instead of each node transmitting the raw data to its respective gateway, it is possible for the nodes to locally fuse information using data aggregation techniques and then send these fused data to the gateway. Though this implies increased computational costs in terms of processing for aggregation of data, it also means possibly a large reduction in communication costs, as these procedures may reduce the volumes of data flowing from the sensors to the gateways. A number of such data aggregation techniques are discussed in [26], and the impact of such techniques is studied in [27].

3 Energy-efficient bootstrapping protocol

In this section, we present the SSB bootstrapping protocol. At the outset, we assume that gateways are high-power entities and can run any distributed algorithm, such as those presented in [28], to establish transmission ordering among themselves. Hence, once deployed, gateway nodes will establish communication links among themselves, negotiate ordering for transmitting announcement messages to the sensors, and then proceed to the bootstrapping procedure.

3.1 Bootstrapping phases

This section studies the information that needs to be exchanged between the network's entities, namely the gateways and sensors, to perform the bootstrapping function and presents the motivation for the various phases in the proposed bootstrapping protocol. First, the gateways need to indicate to other nodes that they are operational and must indicate this fact in some unique manner, in the sense that no other data transmission should be mistaken for a gateway announcing its operational presence. Following the gateway's announcement, the nodes should be provided with timing information for their reply. When the nodes reply, we assume that they have a unique identifier, which they simply send to the gateways, possibly encrypted with a key that only the gateway

```
Nodes

Start
1.   Set receiver to ON
2.   While announcement not detected
3.       Listen to [2*n(bits in one announcement slot)] bits
4.       Sleep for T(number of announcement slots-2)
5.   End while
6.   Decide whether to send in reply or not
7.   If sending reply
8.       Pick slot at random and send reply
9.       If collision detected
10.          sleep till next frame
11.          go to 6
12.      end if
13.  Else
14.      Sleep for one frame
15.      go to 6
16.  end if
17.  Come up each frame/super-frame for synchronization.
end

Gateways

start
1.   Decide an ordering amongst themselves
2.   If myId=currentSenderId then
3.       Send announcements
4.       Listen for Replies. For each reply call processNodeReply
5.       Send CON.
6.       If NO_NEW_NODES entry
7.           Set for all gateways including self, send out control message BOOTSTRAPPING_OVER
8.   Else
9.       Set receiver to ON
10.  Based on expected schedule, decide whether incoming packet is from Node or Gateway and process accordingly, by
     calling processGatewayMessage if it is a gateway message or else tabulating node information.
11.  end if
end
```

Fig. 6 Boostrapping protocol algorithm

knows. The gateways in turn tabulate these replies and use this information for clustering and further network operations. The gateway announcement contains a sequence of bits, which the nodes use to synchronize their time bases with those of the gateways. Once the sensors have been time synchronized with the gateways, they need to be periodically resynchronized in order to counter any clock drifts.

We propose that the bootstrapping procedure be divided into the following four phases:

1) Announcement phase: The gateways announce their presence and solicit responses.
2) Reply phase: Sensor nodes reply back to the gateways and report their identifications and operational parameters such as transmission range.
3) Acknowledgement phase: The gateways acknowledge the sensors' replies.
4) CON phase: Gateways send control instructions to the sensor nodes.

These phases are discussed in greater detail in the following subsections. Before we proceed with the details of the various phases, it is important to introduce the concepts of frames and superframes.

Frames A set of one announcement phase, one reply phase, one acknowledgement phase, and one CON phase constitutes a frame. A frame serves as a TDMA slot reserved for a gateway to transmit its announcements.

Superframes A set of *g* frames, each one corresponding to a unique gateway, forms a superframe. Hence, the number of frames in a superframe is equal to the number of gateways in the network. Therefore, the superframe is used as a TDMA-like transmission calendar for the gateways to eliminate or

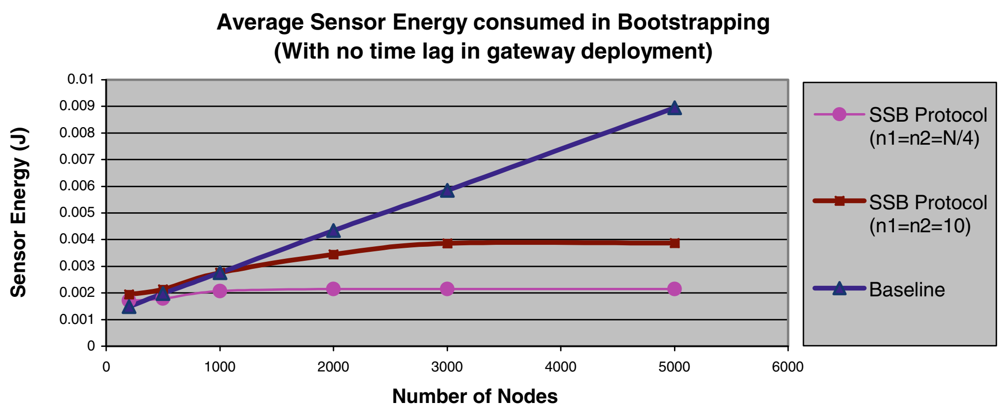

Fig. 7 Average sensor's energy consumed in bootstrapping

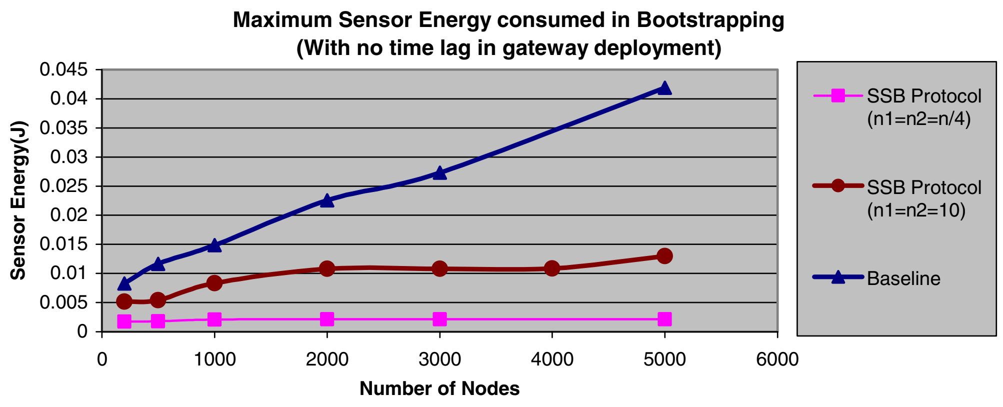

Fig. 8 Maximum energy consumed in bootstrapping

minimize collisions among their announcement messages during the bootstraping phase. A gateway and the sensor nodes sending responses to it can only use the gateway's corresponding frame to send these messages. We set down the additional condition that a set of frames is a superframe only if the first frame in the superframe is the frame corresponding to the gateway, which has been elected leader by the leader election algorithm. This algorithm can also generate a unique priority for each gateway to be used for ordering the gateways' transmission frames. Thus, given an ordered sequence of g gateways, G_1 to G_g, we have g frames constituting a superframe if and only if frame i corresponds to G_i for all i, $1 \leq i \leq g$. Superframes keep repeating until no gateway receives a reply in one superframe; this is considered to be the termination condition for the SSB protocol. The number of superframes required for the bootstrapping depends on the number of sensors and on the quality of sensor-to-gateway communication links. Figure 3 shows the structure of superframes and the different phases in a frame.

Announcement phase In the announcement phase of the protocol, the gateway indicates to the sensor nodes its presence and a time range during which they can reply. All this information is packed into what is termed a "slot." Each announcement phase consists of n_1 such slots. In each slot, the following information has to be included:

1) *Synchronization bits*: Before the gateway supplies any information to the sensors, it will need to transmit a synchronization sequence, a few bits to aid the sensors to synchronize their clock. Clock synchronization enables the sensor to recognize packet boundaries. This is an important characteristic of the SSB protocol and is one of the concepts responsible for energy saving. Algorithms for clock synchronization and the choice for unique bit sequence are explained in [29]. It should be noted that we employ only frame-based clock synchronization. Therefore, sensors and gateways are synchronized to the same frame boundaries and not necessarily the same clock reading. The approach can be extended to include adjustments to the global time as well. We assume that a total of n_3 bits are needed for clock synchronization.
2) *Number of remaining announcement slots*: This conveys to the sensor nodes how many more announcement slots exist before they can start replying. This prevents the sensor nodes from replying before the gateway concludes its announcements, causing collisions and further eating up of energy. For n_1 announcement slots, $\log n_1$) bits will be needed to transmit this information.

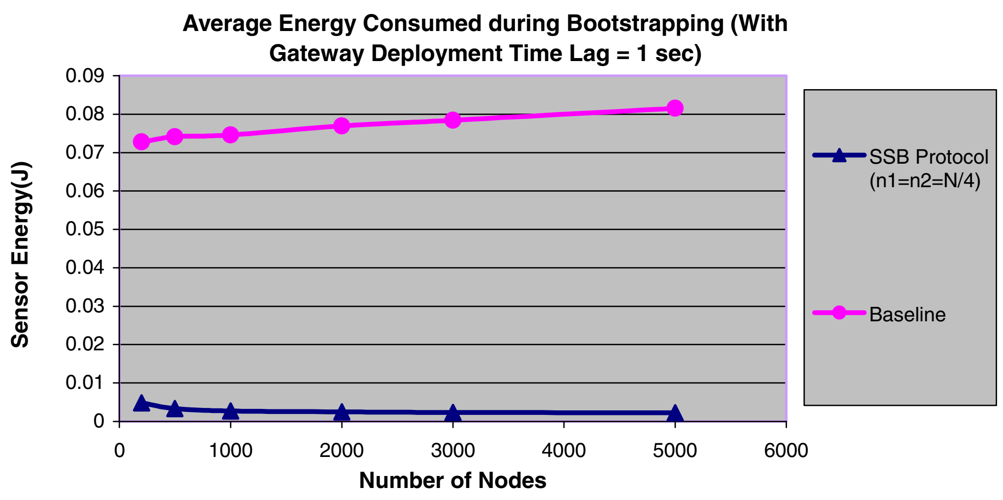

Fig. 9 Effect of deployment time lag on sensor energy

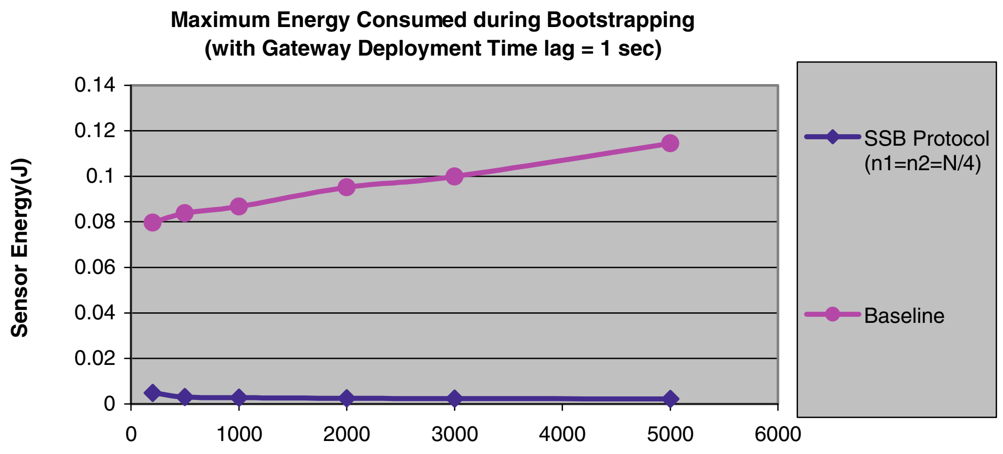

Fig. 10 Maximum consumed energy with deployment time lag

3) *Number of reply slots*: The previous piece of information tells the nodes when they can begin transmitting their replies. However, the sensors also need to be told when they can start transmitting their replies. Incorporating information about the number of reply slots in the protocol also offers flexibility. For instance, when sufficient nodes from one zone have replied, the gateway may cut down dynamically on the number of reply slots. This dynamic adjustment is an application- and scenario-specific extension and can be incorporated if needed. For n_2 reply slots, the number of bits required is $\log n_2$).

Thus, the total number of bits in an announcement "a" will be $a = n_3 + \log(n_1) + \log(n_2)$.

We selected the synchronization bit sequence to be a set of ones. As indicated in Fig. 3, we use guard bits of zeros to prevent contiguous sequences of ones being formed and mistaken for the synchronization sequence transmitted by the gateways. Thus, we ensure that the synchronization sequence is the largest single sequence of ones during the bootstrapping phase. In addition, the synchronization sequence ends with a zero, providing an edge trigger for the sensor nodes to synchronize with the transmitting gateway. We will analyze the length of the synchronization sequence later in this section.

Reply phase The sensor nodes, which can listen to any of the gateways' broadcasts, recognize the announcement upon seeing the synchronization bit sequence and use the edge trigger to synchronize their clock with that of the transmitting gateway. They now have the number of announcement slots that have passed and the number of available reply slots. Hence, the nodes switch their receivers and transmitters to the OFF mode. At the end of the announcement phase, sensor nodes start replying. The reply can include any information the gateway will need to tabulate. Examples of information that the nodes might want to send to the gateway include unique node identifiers, geographical coordinates, remaining energy, and transmission range. The information listed may not always be necessary, depending on the type of application and the clustering and routing mechanisms used in conjunction with the SSB protocol. For instance, the need for geographical coordinates can be eliminated if the network uses multilateration or other location discovery techniques such

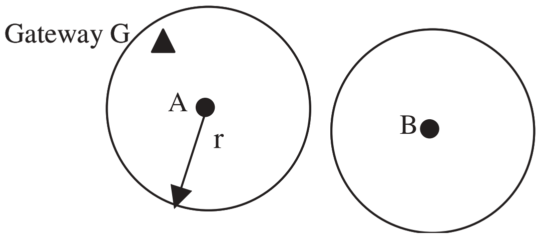

Fig. 11 Node range and gateway reachability

as those described in [12]. Further, some parameters such as range might be requested by the gateway at a later stage and may not be required right at the bootstrapping phase and consequently may be kept out of the reply phase.

We have eliminated the possibility of collisions occurring in the reply phase due to a lack of synchronization between the nodes. However, there is a finite number of reply slots in the reply phase, and the sensors present must choose from these slots. We use a random function, and the nodes pick up slots to reply and send in their replies during that slot. However, it is not possible to ensure that no conflict will occur among the nodes for the same reply slots. Even with a random function it is possible that multiple nodes will select the same reply slot to transmit. We use an exponential backoff scheme. Thus a node first decides whether it is going to reply or not and, only if it decides to, goes ahead picking a slot at random. The exponential backoff scheme keeps reducing the probability of replying each time the node detects a collision. We assume that the nodes possess a carrier sense capability.

When a node decides not to reply in the current frame, it should come up in each frame from that point on and repeat the decision-making process. That is, it should decide whether to reply or not, and if it decides to, it will choose a random slot from the available reply slots. A node that sends in a reply successfully stops transmission until the functional network operation begins and does not attempt to reply any more when any other gateway sends out announcements. All the gateways that hear the node's reply will tabulate its information.

Acknowledgement phase The above phases would by themselves suffice if high-energy efficiency were the only goal and node coverage were not as important, possibly because of the presence of a large number of sensor nodes, a subset of which might be able to satisfy the functional needs of the nodes. However, we add a degree of robustness to our scheme by adding acknowledgements. Acknowledgements can simply take the form of the gateways sending back identifiers to the sensors. This serves as notification to the sensor node that it has been recognized and has become part of the network. We may have varying situations where the sensor node sends out a reply that is not received correctly by the gateway, justifying the presence of an acknowledgement. Examples of such situations include the following:

(1) The node sends a reply, but the gateway is not within its transmission range.

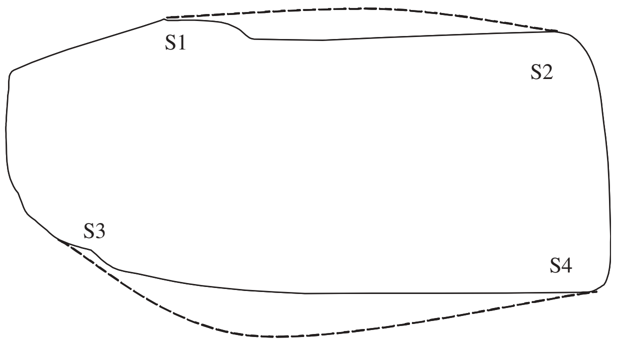

Fig. 12 New and old convex hulls

(2) We face the hidden terminal problem, in which the node under consideration sends a reply and simultaneously another node sends a reply to the gateway. If these two sensor nodes cannot detect each others' transmissions and transmit the entire reply message, collision can occur at the gateway, and consequently neither reply gets tabulated correctly by the gateway.

Acknowledgement is one of the phases where we leverage the concept of time slots. A sensor that has replied in slot m needs to keep its receiver ON only during acknowledgement slot m instead of keeping it ON while all the acknowledgements are transmitted.

Control phase The CON phase is very short and contains control information for any sensor nodes; else it could be a sequence of 0 s. This control information could be anything that the gateways need to convey to the sensor nodes to ensure proper operation. For instance, it could be a directive to start normal operation or a signal that the bootstrapping phase is over and the next few slots would contain the clustering information or even the routing table. Further, this phase could be used to indicate to the nodes that have already replied that the information in the next frame is not related to bootstrapping and is, in fact, data being exchanged between the gateways. Thus, the control phase offers great flexibility and can be used to perform a wide range of functions.

The sensor nodes that successfully reply to any gateway switch their receivers OFF and wake up periodically when the synchronization sequence is transmitted as part of the announcement by each gateway. This ensures that the nodes remain synchronized with the gateways. Also, they must come up at the end of each frame for the CON message.

Analysis of the synchronization sequence The length of the synchronization sequence depends on the normal data being transmitted. The sequence should be long enough to avoid misinterpretation, and hence we must choose a sequence length such that it is greater than the possible length of data that may occur during the bootstrapping phase. Let the number of bits required for the reply message, e.g., for the unique node identifier, position coordinates, remaining energy, and/or any other attributes, be n_4.

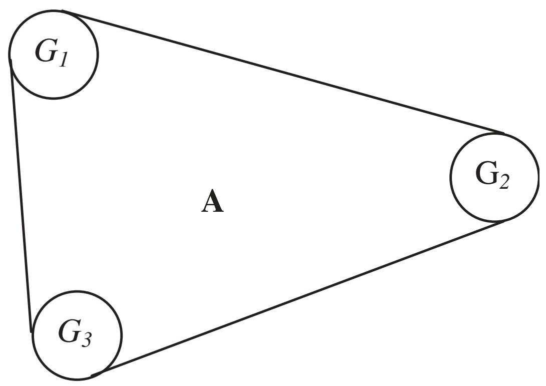

Fig. 13 Area internal to convex hull may be uncovered

Therefore, the number of ones in the synchronization sequence has to be greater than $\log(n_1)$, $\log(n_2)$, and n_4 in order to eliminate the need for bit stuffing [29]. In this context, bit stuffing refers to the mechanism of inserting redundant bits in a transmission to avoid its being mistaken for a sequence of bits, which has a special significance. Handling bit stuffing would either require additional specialized hardware or software capabilities, both of which would introduce additional complexity to the resource-constrained sensor nodes. Hence, we have selected the number of synchronization bits n_3 to be $\max[(\log(n_1), \log(n_2), n_4)]$. As mentioned earlier, the attached guard bits eliminate the possibility that the number of remaining announcement slots and number of reply slots when transmitted sequentially will present a similar possibility.

Management of sensor's receiver So far, this paper has been discussing the bootstrapping steps to be taken assuming that the nodes hear an announcement from the gateways as soon as they are deployed. However, it is possible that the nodes will be deployed and not perceive any announcement either because they wake up during some phase of the protocol other than the announcement phase or even that the gateway has not been deployed. In this context, "hearing an announcement" refers to the node's detection of the unique synchronization sequence. During the announcement phase, the sensor nodes that come up will detect an announcement being made. The baseline bootstrapping protocol discussed in Sect. 1 would require that the sensors keep their receivers ON till they detect an announcement. This could be inefficient for the nodes that do not hear the synchronization sequence on waking up. However, we aim at avoiding this condition, thereby saving energy.

The sensors that come up during the announcement will realize that they are in an announcement phase when they see a sequence of ones (the synchronization pattern) after hearing (a) bits, where a = the number of bits in one announcement slot. To increase the probability that they hear an announcement even if they come up in the midst of an announcement slot, we provide for a window of detection by requiring that the nodes keep their receivers ON for time required to transmit two announcement slots, as shown in Fig. 4. Thus, we require that they stay on for *T(2)*, where *T(n)* henceforth denotes the time required to receive *n* announcement slots. If in this period the nodes hear an announcement, they proceed as described above by sending in replies.

When the sensors do not hear an announcement, the process must be optimized. Instead of having the nodes stay awake till they detect an announcement in such a case, we permit them to switch their receivers OFF for period $T(n_1 - 2)$, where n_1 = the number of announcement slots. Then they come up again and listen for $T(2)$. They repeat this process till they encounter an announcement. By sleeping for $T(n_1 - 2)$, it is ensured that the node will converge to an announcement slot within the time span of a frame, as shown in Fig. 4. This is because if the node sets its receiver OFF just before the announcement phase in a frame begins, it will sleep for $T(n_1 - 2)$ and wake up with $(n_1 - (n_1 - 2)) = 2$ announcements yet to go, thereby ensuring that the complete window of two announcements is heard. Further, if the node sets its receiver OFF after hearing a fraction f of an announcement where $f < 1$, it will wake up with $(n_1 - (n_1 - 2) - f)$ announcements, that is, $(2 - f)$ announcements, yet to go, and f being less than one, we ensure that the sensor node hears one complete announcement.

In Figs. 4 and 5, we show the time instants (x_i and y_j) at which the nodes arise and sleep, respectively. Figure 5 shows that even when the gateway is deployed with a time lag, by sleeping for $T(n_1 - 2)$ the sensor node can switch its receiver OFF for long periods. This contributes to significant energy savings as confirmed by our simulations, discussed later in this section. Figure 6 outlines the bootstrapping protocol.

3.2 Mathematical analysis

This section presents a mathematical model for the energy consumption of the SSB protocol and that of the baseline protocol described in Sect. 1. Applying the deployment-specific values for the model's parameters gives an idea of how the two protocols match up against each other. The terms used in this section are defined as follows:

n_1:	Number of announcement slots
a:	Number of bits in each announcement slot
n_2:	Number of reply slots
s:	Number of bits in the CONTROL phase
N:	Number of sensor nodes in the system
$\log(N)$:	Number of bits in each acknowledgement slot (since it contains the sensor ID). $\log(N)$ is used as a shorthand for $\lceil \mathrm{Log}(N) \rceil$
r:	Number of bits in a reply slot
p:	Number of bits needed to encode one position coordinate

Assuming that the reply consists of a unique node identifier and the node's position coordinates, we have $r = \log(N) + 2 \times \log(p)$. In the baseline approach, the gateway sends out an announcement and all the sensor nodes reply at randomly

```
Outer Motion
      i.   Draw Convex Hull around nodes that have replied so far using any of the methods explained
           in [31]
     ii.   Positions are computed such that they are at distance r from each other on the convex hull
    iii.   Place each Gateway at assigned positions and perform bootstrapping procedure for each
           position
     iv.   Using tabulated replies, draw convex hull again
      v.   Find stretches along convex hull which are "new"
     vi.   If no such new stretches exist END
    vii.   Compute points at distance r from each other on each of the stretches
   viii.   Go to iii
Inner Motion
      i.   Initialize boundary to initial convex hull
     ii.   Use Packing Algorithm to determine where to position gateways to cover maximum area in
           minimum steps within this boundary
    iii.   Divide these points between the gateways and position them at each point
     iv.   Repeat the bootstrapping protocol for each positioning of the gateways
```

Fig. 14 Algorithm for enhanced node discovery

chosen instants. The process continues till all the sensor nodes have replied. Even without factoring in the collisions that frequently occur, the average number of bits that a node would have to receive before successfully replying is ($r \times N/2$). This is because each node would, on average, have to listen to *N/2* replies, each of length *r*, before it got a chance itself to reply.

In the SSB protocol, for each node that tries to become a part of the network, the following two scenarios are possible:

1. The node comes up and immediately perceives an announcement within the first $T(2) = 2a$ bits it receives. The probability of this happening is $\frac{(n_1 \times a)}{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s}$. In this case, the node will have to hear $\frac{3 \times a}{2}$ bits on average. Furthermore, from that point on, it will have to receive an average of $\frac{N}{2 \times n_2} \times s$ bits, since the node has to receive the "s" CON bits $\frac{N}{2 \times n_2}$ times on average from the time that it correctly hears an announcement, assuming that there are no collisions while replying.
2. The second possibility we have is that the sensor node wakes up when either the reply phase or the CON phase is going on. This happens with a probability of $\frac{((n_2 \times \log(n_2))+(n_2 \times r)+s)}{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s}$. In this case, the node has to listen to *(2a)* bits for an average of $\frac{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s}{(n_1-2) \times a}$ $\times \frac{1}{2}$ times before it detects an announcement, that is, an average of $\frac{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s}{(n_1-2) \times a} \times a$ bits, and then analysis proceeds as in case (1) above.

Thus on average a node has to receive the following number of replies:

$$\left\{\frac{(n_1 \times a)}{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s} \times \left[\left(\frac{3 \times a}{2}\right)+\left(\frac{N}{2 \times n_2} \times (s)\right)\right]\right\} + \left\{\frac{((n_2 \times r)+(n_2 \times \log(N))+s)}{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s} \times \left[\frac{(n_1 \times a)+(n_2 \times r)+(n_2 \times \log(N))+s}{(n_1-2) \times a} \times a + \left(\frac{3 \times a}{2}\right)+\left(\frac{N}{2 \times n_2} \times (s)\right)\right]\right\}.$$

Consider a network that spreads over a 1000×1000 m network, where the number of nodes = 2000. Using the simple mechanism, the average number of bits a node has to transmit/receive is 22,000. On the other hand, the SSB protocol performs admirably when the parameters for the same network are plugged in. With the parameters n_1=500, n_2=500, the average number of bits received is 107.64, whereas for n_1=50 and n_2=50, it is 183.33, thus giving a large performance gain.

3.3 Simulation results

We used C++ to build an event-driven simulator for sensor networks to run our experiments. This simulator permits us

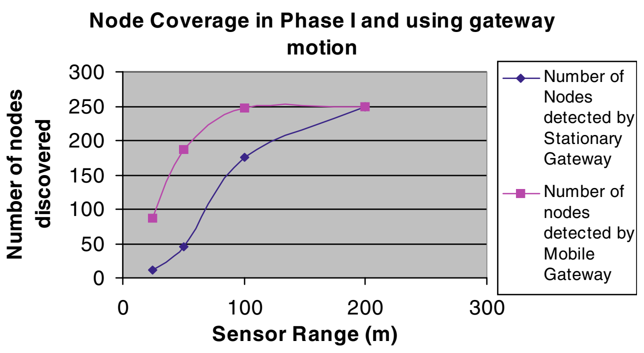

Fig. 15 Number of nodes discovered—enhanced algorithm (maximum: 250 nodes)

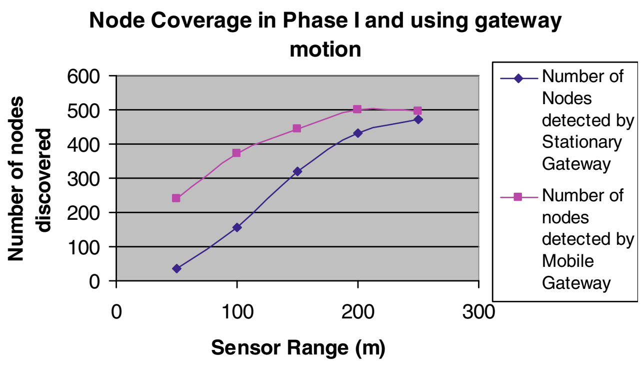

Fig. 16 Number of nodes discovered—enhanced algorithm (maximum: 500 nodes)

to fill in parameters like number of sensors, sensor range, initial energy, dimensions of deployment area, etc. and offers flexibility in terms of sensor distribution patterns. The pattern that we use is a uniform distribution. The sensor nodes are assumed to possess the following communication model characteristics: initial energy = 0.5 *J*, E_{elec}= 50 nJ/bit, E_{amp}=100 pJ/bit/m^2 [14]. The channel transmission energy is assumed to be proportional to the square of the distance (d^2).

We selected a deployment area of size 10 × 10 km. We ran the simulations for the following number of nodes in the network: 200, 500, 1000, 2000, 3000, 4000, and 5000, for both the baseline protocol and the SSB protocol. For the SSB protocol, the values we have chosen in our experiments for the number of reply slots and announcement slots are: (1) $n_1 = n_2 = N/4$ and (2) $n_1 = n_2 = 10$, where N is the number of sensors in the network.

We chose these values to have one set of results with n_1 and n_2 comparable to N and another set with constant small values of n_1 and n_2. In all our experiments, we limited the number of gateways to 3. We present two sets of results, with one case where the gateways and the sensors all come up immediately on deployment and a second case where the gateways come up after some time lag. The performance of the baseline protocol as well as that of the SSB protocol with the two sets of values of reply slots are shown in Figs. 7–10.

Figures 7 and 8 depict the results for the scenario where the nodes and the gateway are all deployed at exactly the same instant. Figures 9 and 10 show results taken for the case where the gateway does not come up at time instant $T = 0$ but takes some time before it starts sending out announcements. We have selected this period to be 1 s. For Figs. 7 and 8, we show the performance result of the SSB protocol with $n_1 = n_2 = N/4$ and $n_1 = n_2 = 10$ compared to the baseline protocol. For Figs. 9 and 10, we chose only to use $n_1 = n_2 = N/4$ as the performance of the SSB protocol is similar over this wide range of values, as seen in Figs. 7 and 8. Figures 7 and 9 show the average sensor energy consumed during the bootstrapping phase, whereas Figs. 8 and 10 show the maximum energy consumed by a sensor. This is a very important parameter, as this may directly affect the time for the first node to die, which consequently affects data routes. As is evident in Figs. 7–10, the SSB protocol performs well for all permutations of the parameters and scales well, as seen from the fact that its performance at 5000 nodes is not very different from that at 200 nodes. This is a big advantage for the SSB protocol, especially considering that the baseline approach uses up as much as 9% of battery capacity in the bootstrapping phase when there are 5000 nodes in the network, without any time lag in the deployment of sensors and gateways. The worth of the SSB protocol is further underscored when we assume a time lag in gateway deployment, as seen in Fig. 10. In this case, the sensors' energy levels may fall by 0.1 J when using the baseline protocol, whereas the SSB protocol still provides appreciable performance preventing the energy from falling more than 0.005 J.

4 Gateway motion for enhanced node discovery

Bootstrapping of the network should detect as many sensor nodes as possible. In the SSB protocol, we have assumed that the gateways can reach all the nodes as they have sufficiently high transmission ranges and energy levels. However, it may happen that the sensor nodes are not able to communicate with the gateways because of large distances separating them or because the existence of obstacles in the transmission path. Such conditions may prevent the nodes with their low power transmitters from reaching the gateways. This issue can be handled to a certain extent if we have gateways with some motion capability. In such a case, we can have the gateways move to locations determined to ensure greater coverage and, consequently, end up with a network with a larger number of discovered nodes. However, it is practically impossible in most scenarios for the gateways to possess beforehand information about the boundaries of the area of deployment. Hence, in this section, we propose an approach that would not require that the gateways have prior knowledge about the extent of deployment. Our approach efficiently determines possible positions for the gateway to move in order to increase the probability of discovering additional sensors.

4.1 Problem statement

At the outset, we consider some possible scenarios in terms of spatial coverage when the gateways do not possess motion

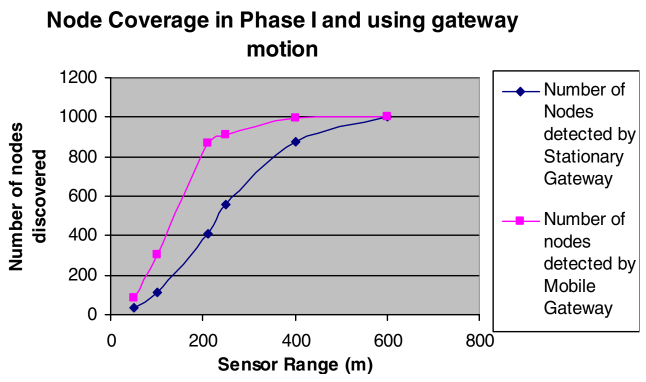

Fig. 17 Number of nodes discovered—enhanced algorithm (maximum: 1000 nodes)

capabilities. In the scenario shown in Fig. 11, sensor node A and sensor node B have a radius of transmission r. However, gateway G in this case lies at a distance $d \leq r$ from sensor A and at a distance $d' > r$ from sensor node B. If the gateway does not possess motion capabilities, but only sensor nodes, which lie in its initial transmission radius r, those nodes will be recognized by the gateways. Thus, if we have g gateways in an area R, having sensor nodes with transmission radius r scattered, each will recognize the nodes within an area πr^2 lying within the circle of radius r centered at the individual gateway.

Thus, the total number of nodes that would be recognized if no such circles created overlap with each other would be proportional to $g(\pi r^2)$, assuming that the sensor nodes were uniformly distributed across the area under consideration. Hence, the ratio of nodes that will be recognized by the gateways to the total number of sensor nodes scattered in the network can be expressed as $g(\pi r^2)/R$. Therefore, if the value of r is very small as compared to the value of R, we will discover a small portion of the nodes in the network, thereby decreasing the network's full functionality and resource levels. This value is also an upper bound assuming a uniform distribution and the actual number of nodes discovered by the gateways could be less than such an upper bound due to the following reasons:

1. This value of node coverage has been derived assuming that the circles of coverage do not overlap or, in other words, that no two gateways are at a distance $d < 2r$ from each other. If two gateways are closer to each other than this distance, the number of nodes discovered will decrease further. This is because some of the nodes that fall within a distance r from one gateway could also be within a distance r from another gateway. Additionally, some gateways' transmission areas might extend outside the deployment area, further decreasing the percentage of the area covered by the gateways.
2. Irrespective of the bootstrapping mechanism used at the outset, it is possible that repeated collisions would prevent some sensor nodes from successfully registering with the gateways. This continues to be an issue regardless of gateway motion but serves to remind us that the calculated upper bound may be very hard to achieve and that we need to incorporate measures to increase node coverage within these constraints imposed by collisions.

We intend to address this issue using mobile gateways to increase sensor node coverage using the SSB protocol and repeating the steps involved, with the variant that the steps are now repeated for different positions of the gateways. Our approach is detailed in the following subsection.

4.2 Gateway motion algorithm

We start with the SSB protocol described in the previous section. Once the termination condition is met, that is, when no replies are received in a superframe by any gateway, the algorithm for enhanced node discovery kicks in. Applying such an algorithm generates a set of locations that the gateways are advised to move to for enhanced node discovery. We next discuss two directions of motion: outer and inner motion of the convex hull of the initially discovered sensors.

4.2.1 Outer motion

The basic idea behind the algorithm is that we draw the periphery of the network as we know it at a given point in time. This can be done by drawing the convex hull around the nodes discovered thus far. A convex hull for a set of points Q is defined as the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior. Techniques for determining the convex hull of a set of points are described in [30]. Once the convex hull is determined, the problem reduces to deciding points along this convex hull at which the gateways should be placed. The algorithm marks out points at a distance r from each other on this convex hull and proceeds to place the gateways at these points from which they will carry out the steps of the SSB protocol. This process is repeated for each of the locations selected on the convex hull. Nodes that reply when the gateways are in these new positions are tabulated and added to the previously recorded entries. Assume that we have k points for positioning and g gateways; each gateway on average would have to be positioned at k/g locations. The convex hull is hence not recalculated till all locations are covered.

Once the k points on the present convex hull are covered, the information for the new replies is also taken into account, the convex hull is reconstructed, and a new set of k' points are determined as the new gateway locations. Thus the convex hull keeps growing outwards and new nodes are continuously tabulated. Whenever it is determined that the convex hull has not changed from one iteration to the next, the process is terminated and the nodes that have replied thus far are assumed to be those existing in the network.

Consider that we have a scenario like that shown in Fig. 12. We have the new convex hull and the old hull differing only in two stretches S1–S2 and S3–S4, with S1–S2 and S3–S4 lying on the new hull. Therefore, we do not position gateways all along the new convex hull but instead optimize on the time required by positioning the gateways only along the new stretches. For this we need to have an intermediate conversion process that would pick out any such similarities between the two convex hulls, discard them, and retain only the new stretches.

While the gateways move to the locations determined by our algorithm, they continue to broadcast the synchronization and control messages. Thus the nodes are kept synchronized. The important characteristic of our protocol that aids in gateway motion for enhanced node discovery is that once the nodes are recognized and acknowledged by the gateways, the nodes do not need to send any information to the gateways. They are only required to keep receiving the synchronization and control messages that the gateways send. Thus, the fact that the gateways that have received replies from the nodes are no longer within the transmission range of the nodes does not hamper our algorithm in any way. Since the gateways have sufficient transmission energy, they can manage to reach all nodes, thus ensuring smooth operation of the protocol.

As mentioned above, the procedure is terminated when the convex hull drawn does not change from one iteration of the algorithm to the next. Assuming that we have a normal or uniform distribution for the sensor nodes, the fact that no gateways have received any replies would imply that the density of sensors has fallen sufficiently to justify terminating the discovery process. If the gateways are not able to hear from additional sensors, this does not necessarily mean that all the nodes have been discovered. However, as most deployments would typically employ a normal or uniform distribution, this termination condition would allow our algorithm to discover most of the deployed sensors.

4.2.2 Inner motion

However, after the above procedure is carried out, it is possible that large portions within the initially constructed convex hull will not be covered. For instance, consider the scenario shown in Fig. 13, in which we have three gateways, G_1, G_2, and G_3. The convex hull constructed around the nodes, which replied successfully to these gateways while in these positions, is used for moving outwards as explained earlier. However, this motion leaves a large area to be covered within this initially constructed convex hull. Therefore, we need to position the gateways within this initial convex hull optimally, in the sense that we should cover as much area as possible in a minimum number of repositioning steps. Given that we are considering circles of radius r as the area that would be covered by each gateway in one position, our problem reduces to fitting in a maximum number of such circles into the convex hull. Hence we suggest that a two-dimensional circle packing algorithm [31] be used for this purpose, with the radius of the circle being r, the range of the sensor nodes. Using a circle packing algorithm would ensure that we cover much of the area under consideration in a minimum number of gateway repositioning steps.

We use a simple packing algorithm by constructing the smallest rectangle possible, which will contain the convex hull as well as an integral number of circles of radius r horizontally and vertically. The centers of these circles are chosen as the points where the gateways will position themselves. These points are treated in a similar manner to those picked on the convex hull, and after positioning themselves on these points, the gateways perform the SSB procedure.

We perform the inner motion first and then proceed to the outer motion so that the gateways will not be in effect retracing their paths in case they perform the outer motion first. This saves time for gateway motion. In both outer and inner node discovery, the computation is done at one gateway. This gateway then indicates to the other gateways their next locations. Similarly, at the end of an iteration of the bootstrapping protocol, the gateways transmit information about which nodes have replied to the gateway elected as leader using the same mechanism. The algorithm for gateway motion is outlined in Fig. 14.

4.3 Experimental results

We extended the simulation setup described in Sect. 3.3 to incorporate our algorithm for enhanced node discovery. We ran the simulations for a variety of scenarios and compared the number of nodes discovered using the SSB protocol to the number of nodes discovered using this enhanced node discovery algorithm. We simulated three deployments, in each of which we had three gateways. The deployments simulated were as follows:

1. 250 sensors scattered within a 250 × 250 m area. The sensor ranges used were 25, 50, 100, and 200 m.
2. 500 sensors scattered within a 500 × 500 m area. Sensor ranges used were 50, 100, 150, 200, and 250 m.
3. 1000 nodes scattered in a 1000 × 1000 m area. Sensor ranges were 50, 100, 210, 250, 400, and 600 m.

Figures 15–17 present the number of nodes discovered by the enhanced algorithm as compared to the number of nodes discovered if we had just the SSB protocol. As the results show, the node coverage increases using the enhanced node discovery mechanism for all the scenarios. The difference between the node coverage with and without gateway motion is less drastic when the sensors' range is relatively large compared to the dimensions of the area of deployment because most of the sensors can reach the gateway directly. However, when the sensors' transmission range is small, the difference in the number of discovered nodes is very high.

5 Conclusions and future work

Although recent research on sensor networks has tackled energy efficiency and awareness of network operation and management, very little attention has been paid to energy consumption during network bootstrapping. In this paper, we have shown that sensors can deplete significant portions of their batteries during bootstrapping if it is not done efficiently. We have described SSB, an energy-efficient bootstrapping protocol for sensor networks. The protocol synchronizes nodes during the discovery process and thus prevents collision in wireless transmission and ensures coordination among the nodes. The presented SSB protocol permits the sensors to switch their receiver OFF for a large portion of the bootstrapping phase and prevents energy wastage when there is a time difference in node deployment.

Simulation results have confirmed the superiority of the performance of the SSB protocol in terms of energy efficiency and scalability. If the gateway is deployed at the same instant as the sensors, our approach consumes only 25% of the energy required by contemporary protocols. When the gateway is deployed after a time lag, the SSB protocol can achieve one to two orders of magnitude in energy saving.

Node coverage is an important parameter, and the utility of a network grows as the number of discovered nodes increases. For scenarios where physical limitations of the sensor device prevent it from communicating with the gateway for bootstrapping purposes, we have proposed a solution that involves gateways with motion capabilities and presented an algorithm to determine motion patterns for the gateways in order to locate new sensor nodes. Our algorithm finds locations for gateways to move to without the knowledge of the size or boundaries of the area of deployment. The effectiveness of this algorithm has also been confirmed by simulation.

Our future plans include extending the presented gateway motion algorithm to better arbitrate new locations among gateways and further optimize their motion patterns. In the current version, the distance of the locations from the gateways' current location is not factored in. Therefore, this entire phase could be further optimized by having more optimal strategies for location assignment, whereby the gateways could cover the same area as they do at present, but in less time, by making better decisions as to which gateway goes to which new location.

References

1. Abidi AA, Pottie GJ, Kaiser WJ (2000) Power-conscious design of wireless circuits and systems. Proc IEEE 88(10):1528–45
2. Akyildiz F et al (2002) Wireless sensor networks: a survey. Comput Netw 38:393–422
3. Estrin D et al (1999) Next century challenges: scalable coordination in sensor networks. In: Proceedings of the 5th annual international conference on mobile computing and networks (MobiCOM '99), Seattle
4. Pottie GJ, Kaiser WJ (2000) Wireless integrated network sensors. Commun ACM 43(5):51–58
5. Sohrabi K et al (2000) Protocols for self-organization of a wireless sensor network. IEEE Pers Commun 7(5):16–27
6. Min R et al (2001) Low power wireless sensor networks. In: Proceedings of the international conference on VLSI design, Bangalore, India
7. Younis M, Youssef M, Arisha K (2002) Energy-aware routing in cluster-based sensor networks. In: Proceedings of the 10th IEEE/ACM international symposium on modeling, analysis and simulation of computer and telecommunication systems (MASCOTS2002), Fort Worth, TX
8. Havinga P, Smit G (2000) Energy-efficient TDMA medium access control protocol scheduling. In: Proceedings of the Asian international mobile computing conference (AMOC 2000)
9. Jolly G, Younis M (2005) An energy efficient, scalable and collision less MAC layer protocol for wireless sensor networks. J Wireless Commun Mobile Comput 5(3):285–304
10. Sohrabi K, Gao J, Ailawadhi V, Pottie G (1999) A self-organizing sensor network. In: Proceedings of the 37th allerton conference on communication, control, and computing, Monticello, IL
11. Subramanian L, Katz R (2000) An architecture for building self-configurable systems. In: Proceedings of the IEEE/ACM workshop on mobile ad hoc networking and computing (MobiHOC 2000), Boston
12. Meguerdichian S et al (2001) Localized algorithms in wireless ad hoc networks: location discovery and sensor exposure. In: Proceedings of the IEEE/ACM workshop on mobile ad hoc networking and computing (MobiHOC 2001)
13. Havinga PJM, Smit GJM (2000) Design techniques for low power systems. J Syst Arch 46(1):1–21
14. Min R, Bhardwaj M, Cho S, Sinha A, Shih E, Wang A, Chandrakasan A (2000) An architecture for a power aware distributed microsensor node. In: IEEE workshop on signal processing systems (SiPS '00)
15. Hill J, Szewczyk R, Woo A, Hollar S, Culler D, Pister K (2000) System architecture directions for networked sensors. In: Proceedings of the 9th international conference on architectural support for programming languages and operating systems, Cambridge, MA
16. Chandrakasan SA (2000) Energy aware software. In: Proceedings of the 13th international conference on VLSI design, Calcutta, India, pp 50–55
17. Singh S, Raghavendra CS (1998) PAMAS: power aware multi-access protocol with signaling for ad hoc networks. ACM Comput Commun Rev 28(3):5–26
18. Arisha K, Youssef M, Younis M (2002) Energy-aware TDMA-based MAC for sensor networks. In: Proceedings of the IEEE workshop on integrated management of power aware communications, computing and networking (IMPACCT 2002), New York
19. Ye W, Heidemann J, Estrin D (2002) An energy efficient MAC protocol for wireless sensor networks. In: Proceedings of the 21st international annual joint conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York
20. Elson J, Romer K (2002) Wireless sensor networks a new regime of time synchronization. In: Proceedings of the 1st workshop on hot topics in networks (HotNets-I), Princeton, NJ
21. Lin CR, Gerla M (1997) Adaptive clustering for mobile wireless networks. IEEE J Select Areas Commun 15(7):1265–1275
22. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd international conference on system sciences (HICSS '00)
23. Youssef M, Younis M, Arisha K (2002) A constrained shortest-path energy aware routing algorithm for wireless sensor networks. In: Proceedings of the IEEE wireless communication and networks conference (WCNC 2002), Orlando, FL
24. Toh C-K (2001) Maximum battery life routing to support ubiquitous mobile computing in wireless adhoc networks. IEEE Commun 39(6):138–147
25. Akkaya K, Younis M (2005) A survey on routing protocols for wireless sensor networks. Elsevier J Ad Hoc Netw 3(3):325–349
26. Heinzelman W, Kulik J, Balakrishnan H (1999) Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of the 5th ACM/IEEE mobicom conference, Seattle
27. Krishnamachari B, Estrin D, Wicker S (2002) The impact of data aggregation in wireless sensor networks. In: International workshop on distributed event-based systems, (DEBS '02), Vienna, Austria
28. Malpani N, Welch JL, Vaidya N (2000) Leader election algorithms for mobile adhoc networks. In: 4th international workshop on discrete algorithms and methods for mobile computing and communications, Boston
29. Stallings W (2000) Data and computer communications 6/e. Prentice Hall, Englewood Cliffs, NJ

30. Cormen T, Leiserson C, Rivest R Introduction to algorithms. Prentice Hall of India

31. Collins CR, Stephenson K (1997) A circle packing algorithm. Manuscript. `http://www.math.utk.edu/~kens/ACPA/ACPA.ps.gz`