Wireless Sensor Networks: Leveraging the Virtual Infrastructure

Stephan Olariu, Ashraf Wada, and Larry Wilson, Old Dominion University
Mohamed Eltoweissy, Virginia Tech

Abstract

Overlaying a virtual infrastructure over a physical network is a time-honored strategy for conquering scale. We design a general-purpose virtual architecture for wireless sensor networks and show that it can be leveraged by a number of different protocols.

Due to advances in nanotechnology, miniaturized, low-cost, low-power devices — referred to as *sensors* — integrating sensing, computing, and wireless communications capabilities have become reality. Ad hoc aggregations of massively deployed sensors create sophisticated sensing, computational, and communication infrastructures called *wireless sensor networks* (WSNs). It is anticipated that in the near future WSNs will pervade society, redefining the way in which we live and work [1, 2].

The main goal of a WSN is to produce meaningful information from raw local data collected by individual sensors. This goal must be achieved while ensuring that the WSN continues to provide, for extended periods, timely and accurate information in the face of security attacks and hardware failures.

Overlaying a virtual infrastructure over a physical network is a time-honored strategy for conquering scale. There are essentially two approaches to this exercise. The first is to design the virtual infrastructure in support of a specific protocol. However, more often than not, the resulting infrastructure is not useful for other purposes. The alternate approach is to design a general-purpose virtual infrastructure with no particular protocol in mind. The challenge, of course, is to design the virtual infrastructure in such a way that it can be leveraged by a *multitude* of different protocols.

To the best of our knowledge, research studies addressing WSNs thus far have taken only the first approach. To wit, in [3] a set of paths is dynamically established as a result of the controlled diffusion of a query from a source node into the network. Relevant data is routed back to the source node and possibly aggregated along these paths. The paths can be viewed as a form of data dissemination and aggregation infrastructure. However, this infrastructure serves the sole purpose of routing and data aggregation, and it is not clear how it can be leveraged for other purposes. A similar example is offered by [4] where sensors use a discovery procedure to dynamically establish secure communications links to their neighbors; collectively these links can be viewed as a secure communications infrastructure. Again, it is not clear that the resulting infrastructure can be leveraged for other purposes.

We view the first main contribution of this article at the conceptual level. Indeed, we introduce a simple and natural general-purpose virtual infrastructure for WSNs consisting of massive deployment of anonymous sensors. The second main contribution of this article is to show that several protocols ranging from routing to data aggregation to security can leverage this virtual infrastructure. A companion paper [5] shows that the virtual infrastructure can be established in a lightweight fashion and is highly dynamic, adapting to changing network conditions.

System Assumptions

Individual sensors are tiny mass produced commodity devices, lacking unique identifiers. Sensors have a nonrenewable power supply, and once deployed must work *unattended*. In order to save energy, sensors are in *sleep* mode most of the time, waking up at random for short intervals under the control of an internal timer. We assume that sensors are equipped with a short-range radio transceiver and have, optionally, laser transmission capabilities [2]. Radio messages sent by a sensor can reach only the sensors in its immediate proximity, a tiny fraction of the overall sensor population. Sensor clocks drift at a bounded rate allowing short-lived and group-based synchronization, where a group is loosely defined as the collection of sensors that *collaborate* to achieve a given task.

Genetic Material

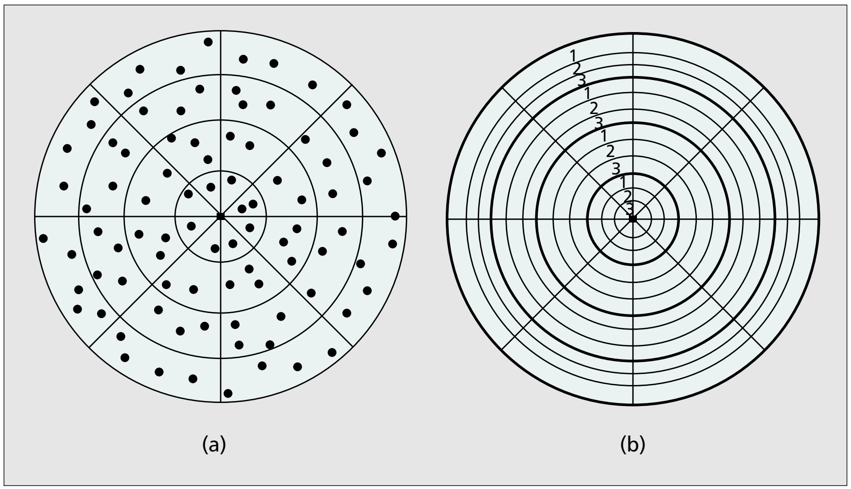
We assume that just prior to deployment (perhaps onboard the aircraft that drops them in the terrain) the sensors are injected with the following *genetic material*:

- A standard public domain pseudo-random number generator.
- A set of tuples $(c_i, k(c_i))$, $(1 \leq i \leq p)$, where c_i is a *color*, and $k(c_i)$ is a *secret key* associated with color c_i.
- A perfect hash function ϕ.
- An initial time: at this point all the clocks are synchronous; later, due to clock drift, synchronization is lost.

The way in which this genetic material is used by individual sensors will be discussed in detail later in the article.

Interfacing with the Outside World

The WSN is connected to the outside world through an external *sink*, whose role may be played by an aircraft over-flying the WSN or a low Earth orbit (LEO) satellite. The sink has a full range of computational capabilities, can send long-range directional broadcasts to all sensors, and has a steady power supply. It also acts as a collector of the information generated by the WSN. Sink-to-sensor and sensor-to-sensor communica-



■ Figure 1. *Illustrating the dynamic coordinate system; b) illustrating color sets.*

tion is by radio, while sensor-to-sink communication is by *laser* and is handled by specialized *reporting sensors*.

A Virtual Infrastructure for WSNs

Since flat architectures do not scale, it is important to endow the WSN with a virtual infrastructure that can be leveraged for efficient protocol design. The proposed virtual infrastructure consists of the dynamic coordinate system, the cluster structure, and the communication structure. To help organize the virtual infrastructure, we assume a centrally-placed *training agent* (TA), equipped with a long-range radio and a steady power supply, that can communicate with both the sink and the sensors in the deployment area.

The Dynamic Coordinate System

Referring to Fig. 1a, the dynamic coordinate system divides the deployment area into *coronas* and *wedges*, defined as follows:

- **Coronas**: Using differential transmission power, the area is ruled into coronas determined by concentric circles of increasing radii centered at the TA. All coronas have the same width, slightly less than the sensor transmission range.
- **Wedges**: These are equiangular wedges centered at the TA obtained by using directional transmission.

The resulting coordinate system is *dynamic* as it can be re-established dynamically in response to changing network conditions. We refer to [5] for the technical details of establishing the dynamic coordinate system.

By using signal strength readings obtained during the establishment of the coordinate system, each sensor selects one of the p colors c_i and remembers the associated secret key k_i. All other tuples in its memory are deleted at this point. The effect of this choice of colors by individual sensor nodes is that coronas are further subdivided into p color sets, as illustrated in Fig. 1b. In each corona the color sets are numbered in the same order, from 1 to p. As a result, the WSN is partitioned into p color *graphs*, where a color graph is the communication graph having the sensors of the same color as vertices with two sensors connected by an edge whenever they are within transmission range of each other. It is worth noting that the width of coronas combined with massive deployment guarantee that the color graphs are connected with high probability.

The Cluster Structure

The dynamic coordinate system suggests a simple *clustering scheme*: a cluster is the locus of all sensors having the same coordinates. It is important to note that clustering is obtained for free once the coordinate system is established. Also, our clustering scheme does not assume synchronization and accommodates sensor anonymity: sensors need not know the identity of the other sensors in their cluster.

The Communication Structure

Depending on the way the information is collected in the WSN, we propose two communication structures. In the *centralized structure* in Fig. 2a sensing data collected in a source cluster is routed, within the same wedge, along a virtual path to the TA where it is uploaded to the sink.

In the *distributed structure* illustrated in Fig. 2b data collected in a source cluster is routed to a destination cluster from which it is uploaded to the sink. We note that the routing illustrated in Fig. 2 involves only sensors in the same color graph.

Task-Based Management

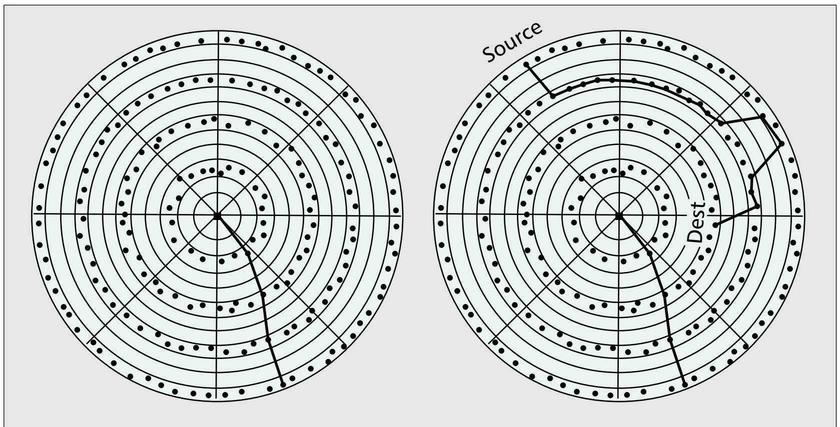
The goal of this section is to describe a task-based management system for WSNs that leverages the virtual infrastructure.

The Model

We take the view that the WSN performs tasks mandated by a remote end user. Our task-based management system involves the following layers:

- **Application layer**: The high-level consumer of information produced by the WSN
- **Middleware**: The interface between the WSN and the application layer

Referring to Fig. 3, the application issues A-tasks, defined in terms of application-level abstractions, to be performed by the WSN. An A-task takes the form of a tuple consisting of a



■ Figure 2. *Centralized communication structure; b) distributed communication structure.*

high-level action, along with a desired level of quality of service (QoS). As an example, the A-task (Fire, p) requires that the occurrence of fire in the area of interest be correctly detected with probability at least p, where p specifies the requested QoS.

The middleware, running at the sink, provides the interface between the application layer and the WSN. It parses the A-tasks from the application layer, considers the current load and capabilities of the WSN including the remaining energy budget, and negotiates a contract with the application layer before committing the network. After a contract has been agreed on, the middleware translates the corresponding A-task into primitive tasks (P-tasks), assigned to individual clusters. The clusters must then perform these P-tasks at the negotiated QoS level and send the aggregated data back to the sink for consolidation. The consolidated information is then passed on to the application layer.

Electing the Task Support Workforce

A P-task is a tuple $T(A, c, S, D, \pi, Q)$, where:
- A describes the action to be performed, say, detecting the presence of smoke.
- c specifies the color graph to be used.
- S specifies the identity of the cluster tasked with data collection (sensing).
- D specifies the identity of the cluster tasked with storing the resulting information.
- π specifies the sequence of clusters along which the data is to be sent from S to D.
- Q specifies the desired QoS.

In addition to the sensors in cluster S, a number of sensors along T are selected to act as routers, relaying the data collected to D. Collectively, these sensors are the *workforce* $W(T)$ associated with T. Referring to Fig. 2a, we note that in a centralized communication model the role of D is played by the TA itself ,and path π is implicit consisting of routers in the same wedge as S. Things are different in the distributed communication model. In this case we distinguish between two scenarios:
- **Explicit specification**: The destination cluster D as well as the path π are explicitly specified along with T. In this case, D and π are determined by the sink and communicated to the WSN. As discussed in [6], π is specified very succinctly relative to S.
- **Implicit specification**: Neither D nor π is specified by the sink. They are determined distributively by the sensors in $W(T)$ as discussed later (see also [6]).

The process by which $W(T)$ is selected follows. During a time interval of length Δ the sink (through the TA) issues a *call for work* containing the parameters of T. The sensors that happen to be awake during interval Δ and satisfy the conditions specified (color, energy level, membership in S, D, or π) stay awake and constitute $W(T)$. It is intuitively clear that by knowing the number of sensors, density of deployment, and expected value of sleep periods, one can fine-tune Δ in such a way that $W(T)$ is commensurate with the desired QoS.

It is extremely important to note that, as discussed next, a by-product of the call for work is that all the sensors in $W(T)$ are synchronized for the duration of the task.

Sensor Synchronization

The goal of this subsection is to discuss the details of the task-based synchronization protocol. We begin our discussion with a generic protocol that will be specialized to suit our needs.

Generic Synchronization —Using the genetic material, each sensor can generate (pointers into) three sequences of random numbers as follows:

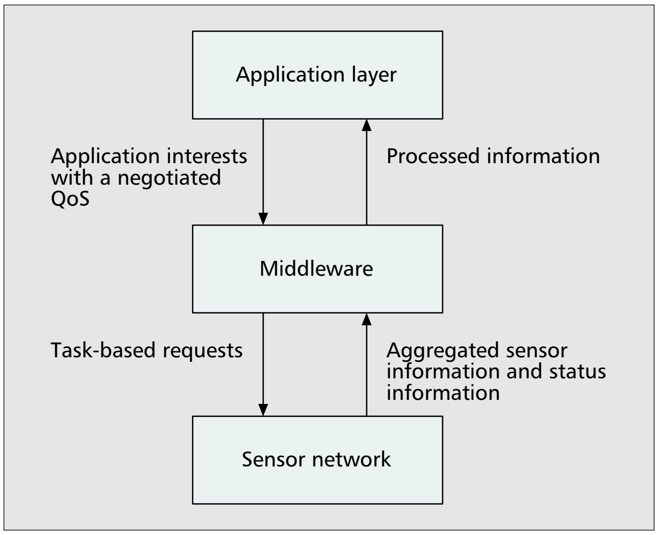

Application layer
Application interests with a negotiated QoS
Processed information
Middleware
Task-based requests
Aggregated sensor information and status information
Sensor network

Figure 3. *Illustrating task-based network management.*

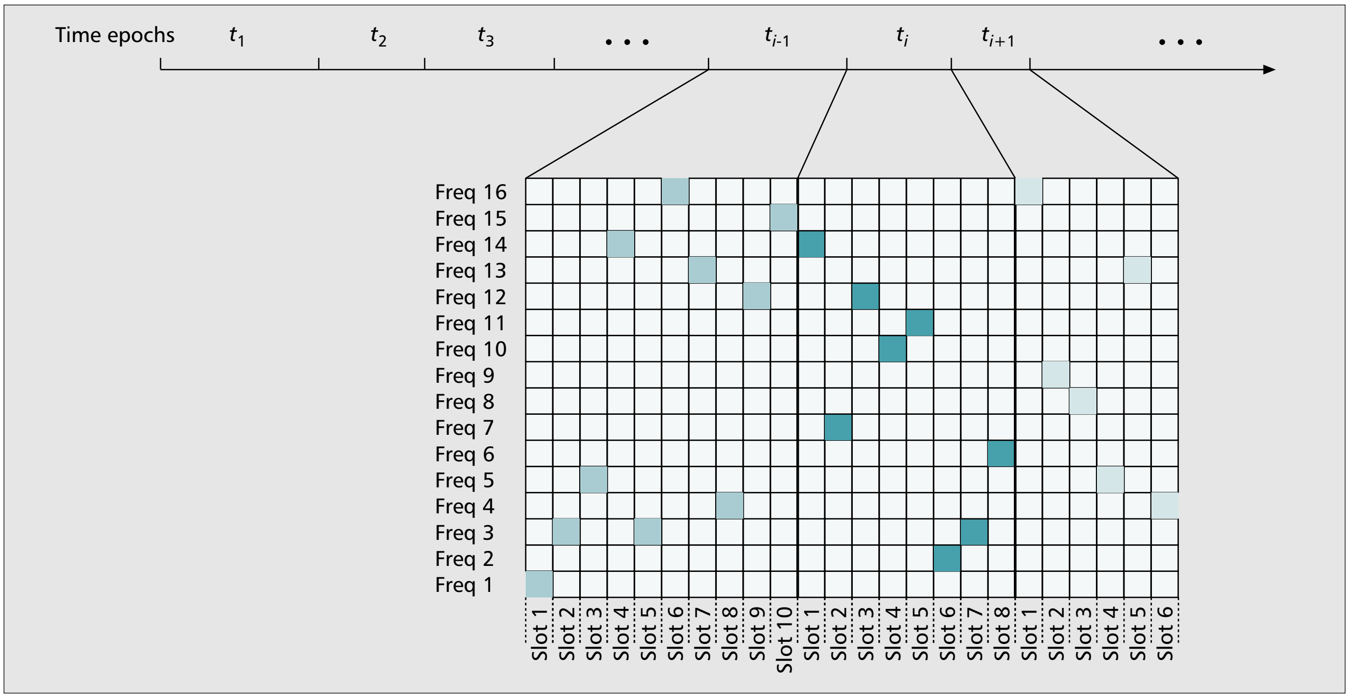
- A sequence $t_1, t_2, \ldots, t_i, \ldots$ of time epoch lengths
- A sequence $n_1, n_2, \ldots, n_i, \ldots$ of frequency sets drawn from a huge universe (e.g., the industrial, scientific, and medical [ISM] band)
- For every i, ($i \geq 1$), a permutation $f_1^i, f_2^i, \ldots$ of frequencies from n_i

The interpretation of these sequences is as follows. Time is ruled into epochs: during the ith time epoch, of length t_i, frequency set n_i is used, subject to the hopping sequence f_1^i, f_2^i, ... Thus, as long as a sensor is synchronous to the sink, it knows the current time epoch, the offset into the epoch, the frequencies, and the hopping pattern for that epoch.

Suppose that the sink dwells t µs on each frequency in the hopping sequence. For every i, ($i \geq 1$), we let l_i stand for t_i/t (assumed to be an integer); thus, epoch t_i involves a hopping sequence of length l_i. Think of epoch t_i as being partitioned into l_i slots, each slot using its own frequency selected by the hopping pattern from the set n_i. We refer the reader to Fig. 4 where some of these ideas are illustrated. For example, time epoch t_{i-1} uses a set of frequencies n_{i-1} = {1,3,4,5,12,13, 14,15,16}. Similarly, t_i uses the set of frequencies n_i = {2,3,6,7,10,11,12,14}, while epoch t_{i+1} uses n_{i+1} = {4,5,8,9,13, 16}. The figure also illustrates the specific frequencies used in each slot.

It is clear that determining the epoch and the offset of the sink in the epoch is sufficient for synchronization. Our synchronization protocol is predicated on the assumption that sensor clock drift is bounded. Specifically, assume that whenever a sensor wakes up and its *local* clock shows epoch t_i, the master clock at the sink is in one of the time epochs t_{i-1}, t_i, or t_{i+1}. Using its genetic information, the sensor knows the last frequencies l_{i-1}, l_i and l_{i+1} on which the sink will dwell in the time epochs t_{i-1}, t_i, and t_{i+1}, respectively. Its strategy, therefore, is to tune in cyclically to these frequencies, spending $\tau/3$ time units on each of them. It is clear that eventually, the sensor meets the sink on one of these frequencies. Assume, without loss of generality, that the sensor meets the sink on frequency λ in some (unknown) slot s of one of the epochs t_{i-1}, t_i, or t_{i+1}. To verify the synchronization, the sensor will attempt to meet the sink in slots $s + 1$, $s + 2$, and $s + 3$ at the start of the next epoch. If a match is found, the sensor declares itself synchronized. Otherwise, the sensor will repeat the above process.

It is important to understand that the synchronization protocol outlined is probabilistic: even if a sensor declares itself synchronized, there is a slight chance that it is not. However,



■ Figure 4. *Illustrating generic synchronization.*

Temp	41–60	61–80	81–100	101–120	121–130	131–140	141–150	151–160
Code	000	001	010	011	100	101	110	111

■ Table 1. *Illustrating temperature ranges and their encoding.*

a mis-synchronization will be discovered quickly, and the sensor will re-attempt to synchronize.

Task-Based Synchronization — The generic synchronization protocol discussed above can be used as a building block for a more sophisticated task-based synchronization protocol. The motivation is that since several color sets are present in a cluster, it is possible for a cluster to perform several tasks in parallel. However, any attempt at synchronization using the generic synchronization protocol will result in all the concurrent tasks using *exactly* the same frequency set and the same hopping sequence, creating frequent collisions and the need for subsequent retransmission.

Suppose we wish to synchronize the workforce $W(T)$ of a task T that uses some color class c, and the generic synchronization protocol would show that the actual time epoch is t_i. The idea is to use the perfect hash function ϕ to compute a *virtual* time epoch t_j with $j = \phi(i, k(c), T)$ to be used by $W(T)$. Therefore, the sensors in $W(T)$ will act as if the real time were t_j, using frequency set n_j and frequency hopping sequence f_1^j, f_2^j, …. Thus, different concurrent tasks will employ different frequency sets and hopping sequences, minimizing the occurrence of collisions.

A Lightweight MAC and Data Aggregation Protocol

Once the sensors in S have collected sensory data, it is necessary to send this data to the repository D. One of the main concerns is that the number of sensors in S is usually large and, unless an expensive TDMA scheme is set up, a large amount of energy will be lost to collisions and subsequent retransmissions. We now show how the virtual infrastructure can be leveraged to produce a very efficient MAC protocol that also allows data aggregation. As it turns out, the resulting data aggregation involves some data loss but does not require the sensors to have unique identities.

Assume that the results of the data collection specific to task T can be partitioned into 2^k, ($k \geq 0$), disjoint groups. Thus, each sensor in S will encode its data in a string of k bits. Since the sensors in S are synchronous, they transmit the data collected bit by bit starting, say, left to right as follows: a value of 0 is not transmitted, while a 1 will be transmitted. The routing sensors associated with T in the next cluster along π pick up the values transmitted. The following disambiguation scheme may be used:
• No bit is received — a 0 is recorded.
• A bit of 1 is received — a 1 is recorded.
• A collision is recorded — a 1 is recorded.

It is clear that as a result of this disambiguation scheme, every receiving sensor stores the logical OR of the values transmitted by the sensors in S. Note also that while there was loss of information in the process of aggregating data, no further loss can occur in traversing π: this is because all routers transmit the same bit string. By enforcing a simple rule on the senders from S, the maximum value can be communicated. For a given bit, if at least one sender has a 1, all senders that have a 0 quit sending all their remaining bits.

For example, consider a sensor network that is tasked to monitor and report the temperature in cluster S. Referring to Table 1, for the application at hand temperatures below 121°F are considered to be noncritical, and if such a temperature is reported, no specific action is to be taken. By contrast, temperatures at or above 121°F are considered to be critical and trigger a further monitoring action. The encoding featured in Table 1 is specifically designed to reflect the relative importance of various temperature ranges. For example, the temperature ranges in the noncritical zone are twice as large as those in the critical zone. Also, notice that the leftmost bit differentiates critical from noncritical temperatures. Thus, if the sink receives a temperature whose leftmost bit is a 1, further action is initiated; if the leftmost bit is 0, no special action is required.

Let us see how our MAC and data aggregation works in this context. Assume that a group of three sensors in S have collected data and are about to transmit it to the sensors in a neighboring cluster along π. The values collected are encoded, respectively, as 010, 011, and 010. Thus, none of the values indicates a critical situation. After transmission and disambiguation, the receiving sensors will store 011, which is the logical OR of the values transmitted. Notice that although the data aggregation process involves loss of information, we do not lose critical information. This is because the logical OR of noncritical temperatures must remain noncritical. Similarly, if the logical OR indicates a critical temperature, one of the aggregated temperatures must have been critical; thus, action must be initiated.

Securing Wireless Sensor Networks

The inherent vulnerability of massively deployed WSNs to a multitude of threats, including physical tampering, exacerbates concerns about privacy and security. In many application domains WSNs constitute a mission-critical system component and require commensurate security protection. Thus, security is a major issue that must be resolved in order for the potential of WSNs to be fully exploited. The task of securing WSNs is complicated by the fact that the sensors are mass-produced anonymous devices with a severely limited energy budget. Security must be provided even though sensors are unattended and vulnerable to a vast array of attacks [4, 7–9].

The main goal of this section is to show that our virtual infrastructure and task-based management can be leveraged to address two other important issues: making sensors tamper-resistant and providing traffic anonymity in WSNs.

We note that our task-based synchronization is, in fact, a very powerful and lightweight encryption device. Indeed, to an outside observer successive epoch lengths, hopping sets, and hopping patterns appear as the product of an unknown random process. Given that techniques are known to discover a hopping sequence by monitoring transmissions, security can only be provided if the design modifies the hopping sequence in less time than is required to discover the sequence. The choice of frequency hopping parameters determines the time required to discover the sequence (the magnitude of the challenge to an adversary).

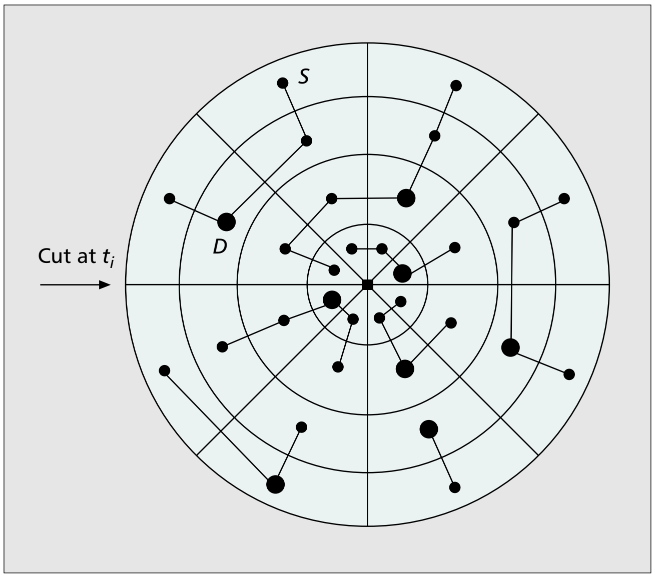
Making Sensors Tamper-Resistant

Since sensors must function unattended, the potential for physical tampering is significant. It is worth noting that while predeployment tamper detection may be worthwhile, post-deployment tamper detection is of little use since, in the vast majority of applications, inspecting individual sensors is impossible or impractical.

The most obvious tamper resistance strategies are hardware-based and involve special hardware circuits within the sensor to protect sensitive data, special coatings, or tamper seals. However, these solutions require extra circuitry that increases the cost and hardware complexity of sensors. Thus, not surprisingly, tamper resistance or tamper protection is not found in present-day sensors [7].

Our solution to the tamper resistance problem does not require additional or more sophisticated hardware.

The *tampering threat model* assumes that the adversary is either trying to force open an individual sensor in situ or attempting to physically remove the sensor from the deployment area. We assume that once removed, the genetic material can be learned and the sensor compromised.



■ Figure 5. *Illustrating hardwired destinations.*

A sensor that detects a tampering attempt will wipe out its memory. Thus, the first threat is easy to handle. We guard against the second threat by relying on the color information of a sensor. The TA periodically sends out a control signal covering the entire deployment area that allows individual sensors to reconfirm their distance to the TA. If the sensor is removed from the area of deployment, it will notice changes in the signal strength received (and thus color) from the TA and will erase its own memory, preventing the tampering agent from gaining access to information secret to the WSN. An alternative solution based on neighborhood signatures is discussed in [5]. It is important to note that if the TA is incapacitated by the adversary, the role of the TA can be taken over by the (external) sink.

Authentication

Sensor authentication is one of the key problems in securing WSNs. We assume that frequency hopping and the tamper-proofing scheme just discussed prevent sensors from being compromised. The security threat for the purpose of authentication is an *external* sensor attempting to masquerade as a legitimate sensor. In particular, it is conceivable, although extremely unlikely, that the intruder was able to guess the set of frequencies and the hopping sequence in the current epoch, and is requesting information from one of its neighbors. However, neighboring sensors exchange color information, which is orthogonal to frequency hopping, adding yet another barrier for the intruder.

Traffic Anonymity in Wireless Sensor Networks

Denial of service attacks that target key nodes in the communication network or compromise communications in some other way could undermine the functionality as well as the performance delivered by the WSN [7, 9]. Particularly vulnerable are the components of the virtual infrastructure. Since knowledge of this virtual infrastructure can be instrumental in successfully compromising network security, maintaining the anonymity of the virtual infrastructure is a primary security concern. Somewhat surprisingly, in spite of its importance, the anonymity problem has not been addressed by prior work on WSNs [10].

We now briefly outline how the virtual infrastructure can be leveraged to provide traffic anonymity in WSNs. The full technical details are beyond the scope of this article and can be found in [6]. Our solutions define schemes for randomizing communications such that the cluster structure and coordinate system used remain undetectable and invisible to an observer of network traffic during both the setup and operation phases of the network. It is worth noting that traffic anonymity in a wired network is achieved by injecting spurious traffic into the network, essentially hiding the actual traffic. The associated cost in terms of energy makes this strategy impractical in WSNs. We propose two solutions, outlined next.

The Centralized Solution — This is by far the simplest and involves relying on traffic randomization specified by an entity external to the WSN. Consider a sequence $T_1, T_2, \ldots, T_n$ of P-tasks to be performed simultaneously by the WSN. While the source cluster S_i of task T_i may be dictated by specific interests of the end user, the choice of the destination cluster D_i and the specific path π_i along which the information is sent to D_i can be chosen externally such that the overall traffic is randomized.

The Distributed Solution

To understand the idea behind the distributed solution to the traffic anonymity problem, imagine that in time epoch t_i we perform a cut-through the coordinate system of Fig. 1a, as illustrated in Fig. 5. Consider a task $T(A, c, S, D, \pi, Q)$ issued during time epoch t_i. The idea is that both D and π are determined by individual sensors (using their genetic material) as a function of the location of S and need not be specified explicitly. In the figure, the large dark circles denote destination clusters, and data is routed along a predetermined path. Since the cut is different from epoch to epoch, the net effect of this approach is to randomize the traffic in the WSN, making it look chaotic to an external observer. We refer the reader to [6] for the technical details.

Concluding Remarks

In this article we propose a virtual infrastructure for a massively deployed collection of anonymous sensors. We show that by leveraging this infrastructure one can design efficient protocols for WSNs. Wireless sensor networks are evolving. Resource-centric, network-centric, and data-centric architectures for WSNs have been proposed. We envision that these architectures will converge toward a standards-based service-centric architecture analogous to the Web services architecture, albeit lightweight. Our current research leverages the proposed virtual infrastructure for constructing such a service-centric architecture for WSNs.

Acknowledgment

This work was supported, in part, by a grant from the Commonwealth of Virginia Technology Research Fund (SE 2001-01) through the Commonwealth Information Security Center.

References

[1] I. F. Akyildiz *et al.*, "A Survey on Sensor Networks," *IEEE Commun. Mag.*, vol. 40, no. 8, Aug. 2002, pp. 102–14.

[2] B. Warneke *et al.*, "SmartDust: Communicating with a Cubic-Millimeter Computer," *IEEE Comp.*, vol. 34, no. 1, 2001, pp. 44–15.

[3] C. Intanagonwiwat *et al.*, "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM Trans. Net.*, vol. 11, no. 1, Feb. 2003.

[4] H. Chan, A. Perrig, and D. Song, "Random Key Pre-distribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy*, Berkeley, CA, May 2003.

[5] A. Wadaa *et al.*, "Training a Wireless Sensor Network," to appear, *Mobile Networks and Apps.*, 2004.

[6] S. Olariu *et al.*, "Traffic Anonymity in Wireless Sensor Networks," in preparation.

[7] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and Approaches for Distributed Sensor Network Security," Tech. rep. 00-010, NAI Labs, 2000.

[8] A. Perrig *et al.*, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, 2002, pp. 521–34.

[9] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," *IEEE Comp.*, vol. 35, no. 10, 2002, pp. 54–62.

[10] S. Olariu *et al.*, "On Providing Anonymity in Wireless Sensor Networks," *Proc. Wksp. Energy-Efficient Wireless Commun. and Net.*, Phoenix, AZ, Apr. 2004.

Biographies

MOHAMED ELTOWEISSY is a (visiting) professor of computer science at Virginia Tech. He is also a professor of computer science at James Madison University. His research interests include information security and privacy, wireless sensor and ad hoc networks, network security, computer-supported cooperative work, and distributed computing. He has published extensively in books, refereed journals, and conference proceedings. He also served on numerous technical committees for conferences, workshops, seminars, and NSF panels. He has an aggressive record of funding (over 10 million). He founded the Commonwealth Information Security Center (CISC) in Virginia and was a founding member of the award-winning Virginia Alliance for Secure Computing and Networking (VA SCAN). He earned a Ph.D. in computer science (R93) from Old Dominion University, and an M.S. (R89) and B.S. (R86 with top rank) in computer science and automatic control from Alexandria University, Egypt. He was nominated by JMU for the Virginia State-Wide Outstanding Faculty Awards in 2003. He is a member of ACM, ACM SIGSAC, and the honor societies of Phi Kappa Phi and Upsilon Pi Epsilon.

STEPHAN OLARIU (olariu@cs.odu.edu) is a professor in the Computer Science Department at Old Dominion University. His research interests include wireless sensor networks, wireless communications and mobile computing, parallel and distributed systems, performance evaluation, and medical image processing.

LARRY WILSON received a B.S. (1963) from Midwestern State University, and an M.S. (1968) and a Ph.D. (1971) from the University of Texas at Austin. He is an associate professor of computer science at ODU with publications in the areas of Mikusinski operator functions, software reliability, parallel processing, mobile computing, and wireless sensor networks.

ASHRAF WADAA [M] received a B.Sc. (honors) degree in computer science and automatic control from Alexandria University, Egypt in 1986. Currently he is a Ph.D. candidate in the Department of Computer Science, Old Dominion University, Norfolk, Virginia. He has co-authored several refereed journal and conference publications. His major research interests include wireless ad hoc and sensor networks, network security, and database systems. He is a member of the honor society of Phi Kappa Phi.