Communication Paradigms for Sensor Networks

Dragoş Niculescu, NEC Laboratories America

ABSTRACT

When compared with now classical MANETs, sensor networks have different characteristics, and present different design and engineering challenges. One of the main aspects of sensor networks is that the solutions tend to be very application-specific. For this reason, a layered view like the one used in OSI imposes a large penalty, and implementations more geared toward the particular are desirable. This survey presents the three main paradigms for communication in ad hoc networks and discusses their applicability for routing, querying, and discovery. We conclude that the node-centric approach, although the oldest and best understood, is not the most appropriate for large-size low-energy application-specific sensor networks.

INTRODUCTION

Sensor networks are in many ways similar to the much studied mobile ad hoc networks (MANETs), but at the same time have some important differences. The similarities are the ad hoc nature of topology, shared communication medium, and problems of connectivity. The differences are that usually sensor networks involve little spatial mobility, are more resource constrained, and therefore pose new scalability problems in two directions. These are with respect to the size of the network and the size of each node. Sizes of sensor networks are likely to grow as they will become more pervasive. An ON World study [1] predicts that in 2010 more than 465 million radio frequency (RF) modules for sensors will ship, compared to 3 million in 2003. Thirty-five percent of these modules will be used for industrial applications, and 28 percent for home automation and control.

While the scalability of large networks is a known problem, the reduced size of nodes is less often acknowledged. Human interaction with these small entities is not possible on a per node basis, as simple necessities such as configuring or battery replacement become very costly. The small size also implies scarce resources: memory, CPU, and battery. One extremity of the resource spectrum is RFID technology. It can be considered a variation of the sensor network if we see RFID readers as sensors of tagged objects. They "sense" discrete values from a known set, rather than analog values from some monitored phenomena. There are passive RFID tags operating without a battery, using only energy from the reader. A small amount of data may be stored on certain tags, but they are not expected to have any communication or computation capability. These and other sensor networks are expected to help with management of large collections of objects (assets, packages), but size and other resource constraints make the management of the network itself a problem.

Even if physical mobility is less of an issue in most sensor deployments, sensors have small batteries or draw energy from the environment, meaning that they have to operate on a reduced duty cycle. This means that the sensor operates only a fraction of the time, but has to be provisioned properly in order to maintain connectivity and relay capacity. This on-off behavior of nodes is a significant factor that must be taken into account for all communication and synchronization aspects of the sensor network.

Most of the research aspects in sensor networks are in one way or another linked to communication. That is the central issue of sensor networks because it is more expensive energy-wise than computation. It takes on the order of 10 μJ to send 100 bits for 100 m [2], but it takes only 0.06 *n*J to execute a 32-bit instruction [3] — more than 100,000 times less. While we cannot eliminate this discrepancy,[1] because the main task of a sensor node besides sensing is to relay information from other nodes, we can try to trade off communication for computation wherever possible. Compression and aggregation of data are clearly candidates for this task, but other ideas from active networking could also be incorporated. The main justification for this view is that sensor networks are very application-specific, so integration in a layered hierarchy may not be the best way to approach this application diversity. In fact, a lot of current research focuses on cross-layer optimizations for sensor networks, implicitly acknowledging the shortcomings of the layered approach.

The data delivery scenario complexity in a sensor network ranges from simple source-destination communication to a multiple source-multiple destination communication mesh with specified aggregation tasks, data rates, and guarantees. Data dissemination and aggregation services (*routing*, *querying,* and *discovery*) have to be

[1] *One way to reduce the high cost of communication is to use optical instead of RF links. While optical communication is several orders of magnitude cheaper energy-wise than RF [4], it is also very directional, which makes deployment of large populations of sensors nontrivial.*

0163-6804/05/$20.00 © 2005 IEEE

adapted to this difference in complexity, rather than have them pay the penalty of generality for jobs that are application-specific anyway.

Routing is the most basic aspect of data delivery, and a large body of literature is available for MANETs addressing trade-offs with respect to mobility, throughput, and protocol scalability. In sensor networks, however, traffic is unlikely to be similar to Internet traffic that is driven by a human interactive component. This difference consists in both patterns and requirements. Sensed data is specific to the sensed phenomena, and might be compressed, aggregated, delayed, or express delivered, depending on urgency. Because of these differences, routing protocols that are specific to sensor networks need to be designed.

Querying is a usually a pull-based retrieval logically similar to that used in databases. In fact, the stream of data generated by the sensors can be seen as a table that can support spatial and temporal queries. Querying is tightly coupled with routing in providing a unified data delivery scheme. Similarly, it has to be adapted to the conditions of the sensor network, but still maintain the familiar interface of classical databases. The querying category is in fact larger than simple pull procedures. It may include push strategies triggered by subscriptions and events.

Discovery is an important service that directly addresses the autonomous operation of the network. A node needs to discover by itself a sink for the data it generates, without human configuration. A network administrator needs to discover the topology of the network to verify the coverage. Depending on the size and density of the network and the energy budget, the discovery might involve a certain resolution of the network. All these problems are specific to sensor networks and have to be addressed in a manner that considers energy efficiency, resolution, and the on-off nature of the nodes.

These three aspects of dissemination and aggregation (routing, querying and discovery) are examined in this survey from the point of view of three communication paradigms: node-centric, data-centric, and position-centric. The next three sections present these paradigms, and the solutions they provide for data dissemination and aggregation. We comparatively discuss the trade-offs of the three paradigms, how they deal with design challenges such as scalability and robustness, and conclude with a summary.

NODE-CENTRIC

The "traditional" way to look at a network is to have nodes labeled with some names and implement routing based on those names. The current Internet is using this node-centric approach, because it is very intuitive, with good reason: it was used by mail and phone systems before. The node-centric approach can easily incorporate hierarchical addressing, but that may not be a usable advantage in sensor networks that are often organized in a flat logical layout.

One of the original advantages for which IP was praised was the ability to make routing invisible to upper layers, and degrade gracefully in the face of node failures by rerouting. This was accomplished, though, by using a single name to identify nodes and endpoint communication entities. In sensor networks, communication endpoints should not be identified by node names, mainly because of the on-off nature of the network (reduced duty cycle). The layered architecture, although not conceptually linked to node-centric addressing, is the de facto standard in current networks. Sensor networks, however, are application-specific, and clearly separated layers pay an unnecessary generality penalty.

A recent review of unicast routing protocols for ad hoc networks [5] classifies them based on the manner in which they react to route invalidation. Proactive algorithms maintain routes all the time to all destinations. Reactive algorithms discover routes only when they are needed or become invalid. The third category is that of hybrid approaches, which have reactive and proactive components. The trade-off between the approaches is mainly with respect to degree of mobility that determines the lifetime of the path. A proactive approach is better in a fixed or low-mobility network, whereas a reactive one works well when paths break often. In the context of sensor networks, mobility has a different face: low duty cycles and random sleep periods for nodes. It is likely that paths will very often be invalidated in either approach, even if positions of nodes never change.

Because of the layer separation, discovery and querying are left for the application layer, which has to rely on lower-level primitives such as unicast, broadcast, and multicast. Discovery is actually a primitive used by reactive routing algorithms to find paths to a destination. This is usually achieved by network-wide broadcasting, implemented as controlled flooding. In dense RF networks this leads to a phenomenon known as a *broadcast storm* [6] in which many nodes sharing the communication medium rush to transmit the same data in an attempt to ensure full coverage of the network. This effect has high energy costs for nodes that may be on a very limited budget (e.g., a solar panel generates 1 mJ/mm^2/day using indoor light [4]).

To reduce the cost of discovery, researchers have investigated methods that leverage the specifics of the resource to be discovered: fish eye state, location servers, and small world properties, to mention just a few. Fish eye state methods make use of the principle that only information close to a node is detailed and often updated, whereas faraway information has lower resolution or freshness. "Small world" properties are used in other work to reduce the length of discovery paths in large graphs (the theory of the small world is that any two people in the world know each other through six acquaintances on average). Another method to reduce discovery cost includes the use of location servers. For example, to discover position information, a quad tree partitioning scheme is employed in [7], an article briefly discussed later. Other methods use Bloom filters to store compressed membership information so that each node's state is only logarithmic in the size of the network. If the resource to be discovered is the address of another node, mobility can be used to facilitate updating of the location servers. In general,

Discovery is actually a primitive used by reactive routing algorithms to find paths to a destination. This is usually achieved by network-wide broadcasting, implemented as controlled flooding.

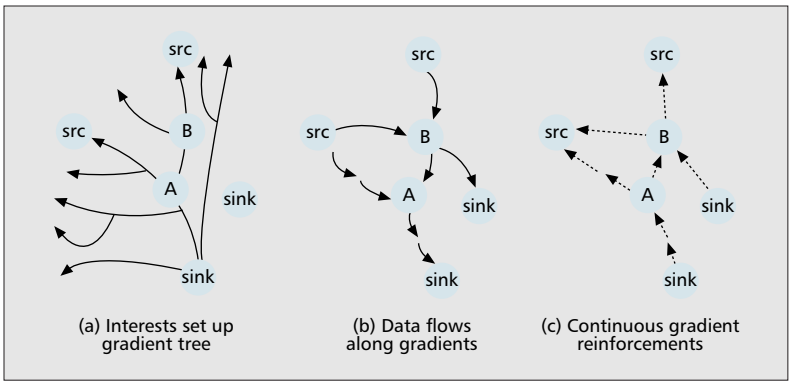

Figure 1. *Directed diffusion.*

however, the problem of discovery in a node-centric network is dealt with by the upper layers, including the application itself. It is worth noting that discovery and updating are similar to the well studied problem of mobility management in cellular systems, which imposes a trade-off between the cost of propagating mobility updates and the cost of paging.

Node-centric addressing handles the problem of multicast well, which was extensively researched for current networks. In sensor networks, multicast groups with memberships, joins and leaves are likely to be overkill to maintain, especially for situations when the group is short-lived (e.g., for distribution of a query or a short notification). Also, a more frequent occurrence is actually the opposite problem, gathercast, or many-to-one communication, when several sources of data stream to the same sink. This problem is directly addressed by the next paradigm.

DATA-CENTRIC

The data-centric approach was found trying to have a network answer the queries of the type "Give me data that satisfies a certain condition." Before examining the similarity to a database query, let us see how this query gets answered in a node-centric network. The identities of the nodes generating data that fits the conditions (e.g., range of temperature) is not known, so the querier must use network-wide discoveries to find these identities. If several sinks are interested in similar data, separate paths must be discovered and maintained independently, while data aggregation can only be performed at destinations.

The most important aspect of the data-centric paradigm is that it is the *content* of the sensor-generated data that drives most implementations of the upper layers: discovery, routing, and querying. Aggregation is performed in the network, and packets are routed based on their content, while the identity of nodes is never involved in the forwarding process.

DIFFUSION

Directed diffusion [8] is a scheme that combines discovery, querying, and routing into one procedure. The easiest analogy is to a publish subscribe system in which sinks declare their interest in some data, interest that eventually reaches the sources of data. The first stage is broadcasting the interest from the sinks to the entire network. This effectively sets up a reverse tree rooted at the sink — the routing entries are called *gradients* here. If several sinks are interested in the same data, diffusion may reuse existing gradients or create new ones. The resulting structure is a collection of overlapping trees (directed acyclic graphs) that can drive data from multiple sources to multiple destinations. In Fig. 1a, only the broadcast from one source is indicated. The routing table at intermediate nodes in the tree contains only next-hop information and the actual gradient. Sources that generate data matching the gradient will forward data along the graph (Fig. 1b), and the sink reinforces the best paths, while the others time out and are removed from the nodes. The last two stages, forwarding and reinforcing, are active for the entire period of data gathering.

When several sinks are interested in the same data, forwarding nodes perform packet duplication because for the given interest they have several next hops in their tables. This publish subscribe method in fact achieves on-demand routing support for many-to-many communication.

Diffusion scales well for large networks with few types of popular queries. The number of gradients kept in nodes depends on the number of queries and density, not on the number of sources or sinks. In Fig. 1b, node A has only one gradient specifying the outgoing neighbor for this query, regardless of the incoming neighbor. Node B has to duplicate incoming matching data to two outgoing neighbors toward the sinks. The difference from the node-centric approach is that the table here does not scale with the number of communication endpoints, but with the complexity of queried data. This makes an important decision factor when choosing it as a communication paradigm in a large network.

SENSOR DATABASES

Another intuitive way to abstract the sensor network is that of a database. The COUGAR project [9] considers that sensors represent the schema, while the tuples are the readings at any given time. The sensor network is assumed to operate on slotted synchronized time, so each sensor has a reading or a null value for a given time. Interface between the query and the network is ensured with a slightly modified SQL which also allows for references to sensors that are spatially close or have other types of relationships between readings. The queries COUGAR is trying to address are long running, need to correlate and aggregate data from different sensors, and may include geographical restriction clauses. For example, to answer the query "Generate a notification whenever two sensors within 5 meters of each other measure simultaneously an abnormal temperature," the SQL is

```
SELECT R1.s.detectAlarmTemp(100),
R2.s.detectAlarmTemp(100)
FROM R R1, R R2
WHERE $SQRT($SQR(R1.loc.x -R2.loc.x)+
$SQR(R1.loc.y -R2.loc.y))<5
      AND R1.s > R2.s AND $every();
```

In COUGAR, query processing is performed in a database front-end, but some basic functions are performed by sensors. The relations in the database are either partitioned across the set of devices or stored in the front end, and the query execution plan has to face asynchronous operation and multiple output responses.

TinyDB is a newer project [10] that also supports an SQL-like interface, but each sensor node has its own query processor. As in COUGAR, sensor data is a single table with one column per sensor type, and tuples are appended periodically. It extends SQL with additional features such as in-network materialization points, joins, grouped aggregations, and triggers. The key energy saving feature is the use of semantic routing trees aimed to reduce the amount of communication involed in answering a query. The functioning of TinyDB is organized into these stages:

- Use flooding to build a spanning tree rooted at the user.
- Synchronize the network, and define epochs and intervals.
- Broadcast a query along the tree.
- During each epoch:
  - –Leaves produce a row of data, and apply the filter/query.
  - –Partial state is passed up during the scheduled interval.
  - –Parents aggregate partial states from all children, apply filter/quer, and pass up new partial state.

These database approaches have the advantage of a simple high-level interface to the sensor network, but really address only one type of problem, that of answering continuous tuple-based queries. In fact, the underlying implementation of the network might still be node-centric, but the only interface available to the user is data-centric.

When large amounts of data are to be relayed over the network, the natural question arises of whether compression can be used. For example, when watching the activity of a sensor, the question is if the output can be predicted, in which case it is not news anymore and does not need to be transmitted. The data receiver may achieve substantial savings by instructing the sensor to actually send data only when it does not satisfy a certain prediction. The PREMON project [11] addresses the issue of how to perform in-network compression and is summarized here as an example of a novel data-centric view of the sensor network. A snapshot of readings from all the sensors in the network at the same time is akin to an image. Monitoring the entire network then becomes watching a "sensor movie." The similarity continues for the compression solution, because depending on the deployment scenario, it is possible that there are both spatial and temporal correlation between sensor readings. MPEG compression already addresses these types of dependencies, but for sensor networks the compression must be performed in a distributed manner, balancing the cost of transmission with that of processing, all under the restriction of limited node storage.

POSITION-CENTRIC

If we see sensor networks as a way to instrument the physical world, the reported data almost always has to be associated with a position (e.g., temperature map or motion detection). While global coordinates and compatibility are desirable, the Global Positioning System (GPS) may not always be used because of line-of-sight conditions, form factor and power requirements, or cost. The problem of positioning nodes in the field has been addressed by many research communities: vision, networking, robotics, and signal processing. Many of these solutions do not adapt directly to the size and power constraints of the sensor, but careful design of the sensor may enable hardware features that allow for effective positioning.

Because sensor networks' main goal is to monitor physical space, their operation is intrinsically linked to location. In many cases it makes more sense to address an area of sensors by their location rather than by their IP addresses. The position-centric approach uses positions of nodes as a primary means to address and route packets. In its simplest form, called Cartesian forwarding, if a source knows the position of the destination, it forwards packets to the neighbor closest to the destination. This method was actually mentioned in the 1970s, in the context of what were then called packet radio networks, precursors of today's ad hoc networks. Several position-based algorithms for routing and discovery are surveyed in more detail in [12].

The position-centric way of addressing comes with a number of advantages and disadvantages. One good thing about it is that there is no need for routing tables in the network, since every node can decide how to forward packets based only on the destination of the packet and some locally gathered information about its immediate neighbors. Another positive aspect is independence from mobility: as long as intermediate nodes with known positions exist between source and destination, routing is performed without the penalty of route discoveries and updates.

The disadvantage is that the source must know the position of the destination. However, this is an implicit requirement in many applications, like sensor networks that relay all data to a unique known collection of static sinks, or when the requester of the data includes its position with the request.

We now examine two important aspects of position-centric forwarding: applications that can be implemented under this paradigm, and resilience in front of obstacles and nonuniform density networks.

ROUTING ON TRAJECTORIES

While most position-centric implementations use straight lines, there is reason to use non-straight lines for routing. Trajectory-based forwarding (TBF) [13] is a generalization that allows forwarding data along arbitrary curves in order to solve problems like routing, broadcasting, multicasting, and discovery. It is a source-based method in that the source declares the path as an $X(t),Y(t)$ parametric encoding included in every data packet. Intermediate nodes, knowing

Because sensor networks' main goal is to monitor physical space, their operation is intrinsically linked to location. In many cases it makes more sense to address an area of sensors by their location rather than by their IP addresses.

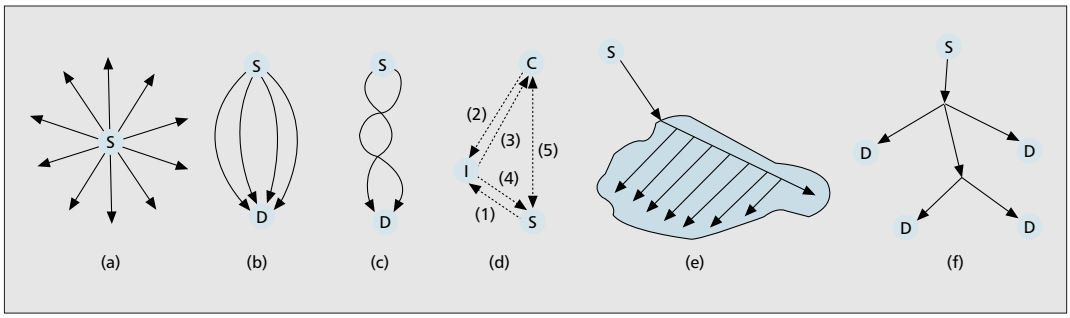

Figure 2. *Applications of TBF.*

their own position and the equation of the curve the packet is supposed to follow, perform greedy forwarding decisions that do not depend on endpoints or names of intermediate nodes.

A sampling of applications that can take advantage of nodes' positions is shown in Fig. 2. A simple flooding replacement, shown in Fig. 2a sends radial trajectories from the point of origin, and relies on the broadcast nature of the medium to deliver the packet to as many nodes as possible. A better approach is shown in Fig. 2e, where parallel rays ensure better coverage when the spacing between the rays is dimensioned around one wireless radio range. TBF can be used to easily achieve diversity of the paths, useful for increasing capacity, load balancing, or increased resilience, as shown in Fig. 2b and 2c. As noted in previous paragraphs, discovery is an important part of data dissemination, and usually an expensive one. Using trajectories, a five-step procedure (Fig. 2d) that avoids flooding the entire network with requests works as follows: servers *S* advertise their location along arbitrary lines (1), and clients *C* also query along arbitrary lines (2); some lines will always intersect, and the intersection nodes *I* are able to notify the clients (3) and the servers (4) about their respective locations for direct TBF-based communication (5). In order to guarantee intersection of client and server directions, it is necessary to send lines in three or four directions from both server and client. Simple trajectories may be composed to achieve arbitrary distribution paths such as the ones needed for multicast (Fig. 2f), and regular trees describing repetitive structures can be specified in a compact low-overhead form. For example, to achieve the plane filling structure in Fig. 3, a simple recursive relation is used: after advancing one unit in some direction, the trajectory splits 60° left and right, recursively propagating that same trajectory code that eventually generates the beehive structure. It is worth noting that with a larger unit, this procedure may be used to discover the topology of the network at a lower resolution, without actually discovering all the nodes of the network. For management purposes, this procedure is much cheaper energy-wise than flooding, but still provides the shape and connectivity conditions of the network.

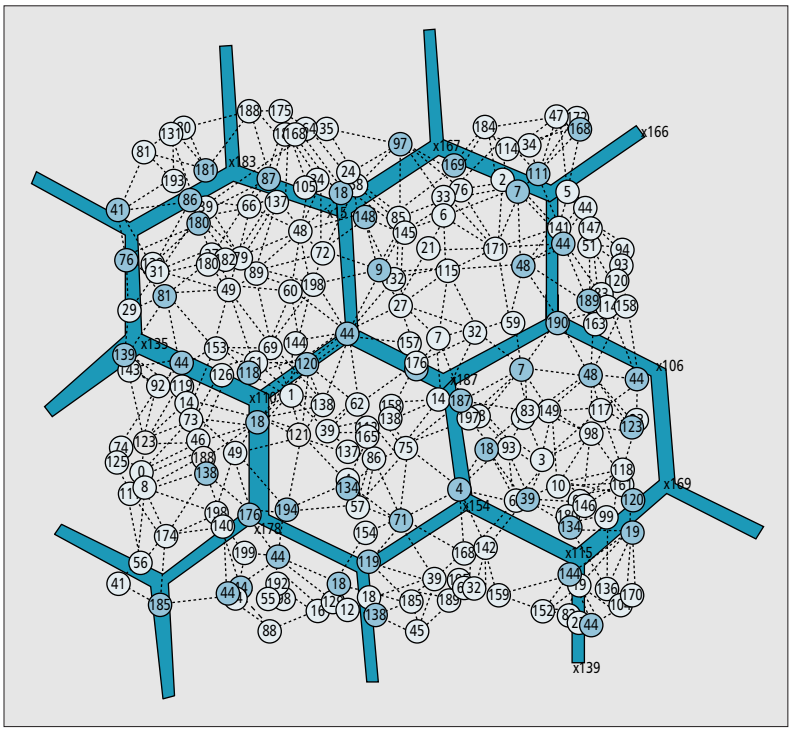

Figure 3. *Broadcasting approximation using a self-overlapping trajectory.*

Routing Around Obstacles

The early solutions for position-centric routing use simple greedy strategies to forward packets. In cases when obstacles interrupt the trajectory, or the density is simply not sufficient to support forwarding close to the trajectory, packets are dropped. Even in simple cases, a node may have no neighbor closer to the destination due to a local feature of the topology, although after a slight detour, a reasonable path may be found.

The FACE algorithm [14] uses a planarization of the graph in order to guarantee delivery of position-centric forwarding. There are several localized methods that can compute a planarization of the graph using one-hop or two-hop information only. The intuition behind planarization is that the plane is seen as a map, each region assigned to a country. A source-destination (*SD* line in Fig. 4) crosses a number of countries, and it is possible to guarantee delivery by making sure that all the regions are traversed in order. While in most cases it is not necessary to actually traverse the region, as greedy forwarding does a reasonable job, sometimes holes in connectivity or obstacles require that the packet follow the lefthand rule to navigate around the region in order to find the exit point. The outside boundary of the network is also a face, so the lefthand rule works in that case as well. This last point emphasizes a potential dis-

advantage, of requiring relatively dense networks in order to achieve reasonable approximations for the desired trajectories (i.e., without large detours).

Support for Node-Centric Applications

In order to use legacy software (node-centric) on top of a position-centric network, a location service is necessary that finds the position of a given address. Grid location service (GLS) [7] is a decentralized service that runs on the mobiles themselves, requiring no fixed infrastructure. Each node A, being a potential destination, establishes a set of location servers, each serving a progressively larger region, depending on the distance to A. When A moves, it updates its location with all its servers. To answer a location query from node B, the communication involves the least square containing both A and B, which in extreme cases may be the entire map, even if A and B are close to each other. Location server management and location updating resemble geographic forwarding as they operate in similar ways. While geographic forwarding pushes packets closer to the destination in physical space, GLS moves packets progressively closer to the destination in ID space.

Bridging between different paradigms is important not only because most existing software is for node-centric networks, but also because it offers another design option to mix and match features from different communication and addressing approaches.

Discussion

It is interesting to compare the surveyed communication paradigms (Table 1) from the point of view of interchangeability and compatibility with legacy networking code. It is clear that the node-centric/layered approach, which was designed with this goal in mind, is best in this respect. When comparing the data-centric and position-centric approaches, it is the position-centric that provides more separable functionalities, similar to a layered approach. The broadcast storm problem has a number of solutions, the best involving position-based improvements at the medium access control (MAC) layer. It is possible to use positions at the routing layer, for discovery, unicast, and multicast. Upper layers can also benefit from node positions, for application-level discovery, increased reliability through path diversity, and topology discovery at various resolutions. Data-centric approaches, on the other hand, tend to provide a top-to-bottom solution, as is the case with directed diffusion. In fact, directed diffusion solves only one problem, but solves it right.

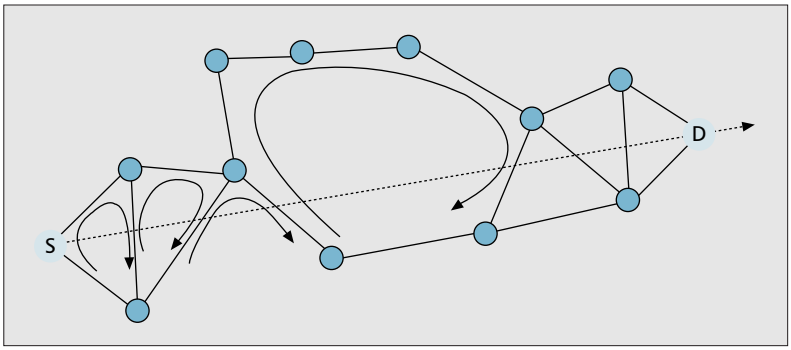

Figure 4. *The FACE algorithm routes around obstacles using the lefthand rule.*

A new IEEE standard, 802.15.4 [15], is aimed at low-power low-distance communication devices that may allow years of battery life. The standard allows for both hierarchical and flat peer-to-peer topologies, and provisions for one-hop reliability and real-time guarantees. At the lower layers, there may be a choice between RF and optical communication, but it is still unclear what the logical and address organization of future sensor networks will be. It can be flat with identical nodes, or hierarchical with cluster heads that are more powerful in terms of storage, computation, and communication. Node-centric communication is suited to hierarchical addressing, but the other two approaches are currently debated mostly in flat addressing scenarios.

Another comparison criterion is mobility, which is a major problem in node-centric networks because paths and connectivity are made of node names. Solutions here are either awk-

	Node-centric	Data-centric	Position-centric
Compatibility	✔	✘	≈
Layered	✔	✘	✔
Additional requirements	✔	✔	✘ (Node positions)
Routing			
Mobility	✘	✘	✔
Scalability	✘	✔	✔
Multicast/gathercast	✘	✔	✔
Disconnection	✘	✔	✔
Discovery	✘ (Flooding)}	✔ (Built in)	≈ (Some support)
Querying	✘ (Upper layers)	✔ (Built in)	✘ (Upper layers)

Table 1. *Comparison of the three paradigms.*

Data collection has an inherent spatial aspect because the main application is monitoring of the physical space. It also has a data centric temporal aspect when the network has to answer or monitor specific queries.

ward (triangle routing in mobile Internet) or wasteful (rediscovery of paths in ad hoc node-centric networks). Here position-centric approaches have the advantage because they do not require particular nodes to be involved in forwarding, but use whichever ones provide connectivity. This is also the case with sensor-specific on-off-type mobility. Mobile directed diffusion is still a research subject, although the original algorithm was not designed to support mobility.

Considering the application-specific nature of sensor network design, a direction worth exploring is that of active sensor networking. From a security point of view, running arbitrary code received over the network might be less desirable, but the advantages are that the communication can really be tailored in all aspects ranging from the link layer to the application for the task at hand. A position-centric approach like TBF makes use of evaluation of expressions to encode trajectories, and can benefit from an active approach. Compression and aggregation tasks require in-network computation that cannot be predicted at design time, so providing installable code as a basic functionality helps in installing any of the mentioned paradigms, or even switching between them. Some of the projects exploring the possibility of installing arbitrary code on sensors are SensorWare [16] and Maté [17]. Their use of TCL scripts and bytecode allows installation of complex distributed algorithms that can access all the communication and sensing capabilities of each node.

Finally, if sensor networks are to be deployed in large sizes, scalability with respect to the number of nodes becomes a deciding factor in choosing a communication paradigm. Node-centric suffers not only with respect to the size of routing tables and frequency of updates, both of which depend on the total number of nodes, but also with respect to the address space. A data-centric approach such as diffusion scales routing tables with the number of queries to be supported, also providing better scalability in large networks.

Data collection has an inherent spatial aspect because the main application is monitoring the physical space. It also has a data-centric temporal aspect when the network has to answer or monitor specific queries. Besides the inherent advantage of being more application-specific, position- and data-centric paradigms also provide scalability advantages for routing, querying, and discovery. An important difference is that they do not link endpoint communication entities to node names, as the node-centric approach implicitly does. Considering all the above factors, it is likely that position-centric, data-centric, or maybe a combination of them is the best bet for future sensor networks.

SUMMARY

The quality of ad hoc networks operating without infrastructure in a completely decentralized manner becomes a disadvantage when trying to design energy-efficient protocols. Communication, which is the most energy-costly aspect of the network, can be organized in three fundamentally different ways: node-centric, data-centric, and position-centric. Node-centric communication is the most popular and well understood paradigm, being currently used in the Internet. The other two surveyed in this article, data-centric and position-centric, are more scalable, better adaptable to applications, and conceptually more appropriate in many cases, and therefore may successfully challenge the node-centric way of looking at the sensor networks.

REFERENCES

[1] C. Chi and M. Hatler, "Wireless Sensor Networks: Mass Market Opportunities," *ON World Inc.*, Feb. 2004.
[2] IEEE Std. 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," 1999.
[3] ARM Ltd., ARM7TDMI core processor product overview (0.13 mm), 2004, http://www.arm.com/products/CPUs/ARM7TDMI.html
[4] B. Warneke *et al.*, "Smart Dust: Communicating with a Cubic-Millimeter Computer," *IEEE Comp.*, vol. 34, no. 1, Jan. 2001, pp. 45-51.
[5] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A Review of Routing Protocols for Mobile Ad Hoc Networks," *Ad Hoc Networks*, vol. 2, no. 1, Jan. 2004.
[6] Y.-C. Tseng *et al.*, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *Wireless Networks*, vol. 8, no. 2/3, 2002, pp. 153-67.
[7] J. Li, J. Jannotti *et al.*, "A Scalable Location Service for Geographic Ad Hoc Routing," *ACM MobiCom*, Boston, MA, Aug. 2000, pp. 120-30.
[8] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks," *ACM MobiCom*, Boston, MA, Aug. 2000.
[9] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards Sensor Database Systems," *Proc. 2nd Int'l. Conf. Mobile Data Mgmt.*, Springer-Verlag, 2001, pp. 3-14.
[10] S. R. Madden *et al.*, "The Design of an Acquisitional Query Processor for Sensor Networks," *SIGMOD*, San Diego, CA, June 2003.
[11] S. Goel and T. Imielinski, "Prediction-based Monitoring in Sensor Networks: Taking Lessons from MPEG," *ACM Comp. Commun. Rev.*, vol. 31, no. 5, Oct. 2001.
[12] M. Mauve, J. Widmer, and H. Hartenstein, "A Survey on Position-Based Routing in Mobile Ad Hoc Networks," *IEEE Network*, Nov./Dec. 2001, pp. 30-39.
[13] D. Niculescu and B. Nath, "Trajectory Based Forwarding and Its Applications," *ACM MobiCom*, Sept. 2003.
[14] P. Bose *et al.*, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," *3rd Int'l. Wksp. Discrete Algorithms and Methods for Mobile Comp. and Commun.*, Seattle, WA, Aug. 1999.
[15] IEEE Std. 802.15.4, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," 2003.
[16] A. Boulis, C. C. Han, and M. B. Srivastava, "Design and Implementation of a Framework for Programmable and Efficient Sensor Networks," *ACM MobiSys*, San Francisco, CA, May 2003.
[17] D. Culler and P. Levis, "Maté: A Tiny Virtual Machine for Sensor Networks," *Int'l. Conf. Architectural Support for Prog. Languages and Op. Sys. (ASPLOS X)*, San Jose, CA, 2002.

BIOGRAPHY

DRAGOŞ NICULESCU (dragos@nec-labs.com) received a Ph.D. in computer science from Rutgers University, and is currently a researcher at NEC Laboratories, Princeton, New Jersey. His current research includes issues in sensor systems, mobile computing and networking, voice over IP, and QoS.