Dealing with Faults in Wireless Sensor Networks

Lilia Paradis and Qi Han[*]

Department of Mathematical and Computer Sciences

Colorado School of Mines, Golden, CO 80401

{lparadis,qhan}@mines.edu

February 1, 2007

[*]Correspondence Author: Qi Han. Mailing address: Department of Mathematical and Computer Sciences Colorado School of Mines, Golden, CO 80401. Tel: 303-273-3849. Fax: 303-273-3875. Email: qhan@mines.edu.

Abstract

Wireless sensor networks are resource-constrained self-organizing systems that are often deployed in inaccessible and inhospitable environments in order to collect data about some outside world phenomenon. For most sensor network applications, point-to-point reliability is not the main objective; instead, reliable event-of-interest delivery to the server needs to be guaranteed (possibly with a certain probability). The nature of communication in sensor networks is unpredictable and failure-prone, even more so than in regular wireless ad hoc networks. Therefore, it is essential to provide fault tolerant techniques for distributed sensor applications. Many recent studies in this area take drastically different approaches to addressing the fault tolerance issue in routing, transport and/or application layers. In this paper, we summarize and compare existing fault tolerant techniques to support sensor applications. We also discuss several interesting open research directions.

keywords: sensor networks, fault management, fault detection, fault diagnosis, fault tolerance

1 Introduction

Continuing advances in computational power and radio components, as well as reduction in the cost of processing and memory elements have led to the proliferation of micro-sensor nodes (e.g., Mica motes from Crossbow, Tmote Sky from Moteiv, the MKII nodes from UCLA, SunSpot from Sun, etc.) that integrate computation, communication, and sensing capabilities into a single device. Wireless sensor networks are self-organized networks that typically consist of a large number of such sensing devices with severely limited processing, storage and communication capabilities and finite energy supply. Sensor networks are now rapidly permeating a variety of applications domains such as avionics, environmental monitoring, structural sensing, tele-medicine, space exploration, and command and control. With multihop wireless communication, sensor nodes have made it possible to build reactive systems that have the ability to monitor and react to physical events/phenomena. In addition to resource constraints, sensor networks are also failure-prone [1, 2, 3, 4]. Therefore, fault tolerance is as critical as other performance metrics such as energy efficiency, latency and accuracy in supporting distributed sensor applications.

Sources of Faults: Data delivery in sensor networks is inherently faulty and unpredictable [1]. Failures in wireless sensor networks can occur for various reasons. First, sensor nodes are fragile, and they may fail due to depletion of batteries or destruction by an external event. In addition, nodes may capture and communicate incorrect readings because of environmental influence on their sensing components. Second, as in any ad hoc wireless networks, links are failure-prone [2], causing network partitions and dynamic changes in network topology. Links may fail when permanently or temporarily blocked by an external object or environmental condition. Packets may be corrupted due to the erroneous nature of communication. In addition, when nodes are embedded or carried by mobile objects, nodes can be taken out of the range of communication. Third, congestion may lead to packet loss. Congestion may occur due to a large number of nodes' simultaneous transition from a power-saving state to an active transmission state in response to an event-of-interest [5].

Furthermore, all of the above fault scenarios are worsened by the multihop communication nature of sensor networks. It often takes several hops to deliver data from a node to the sink; therefore, failure of a single node or link may lead to missing reports from the entire region of the sensor network. Additionally, congestion that starts in one local area can propagate all the way to the sink and affect data delivery from other regions of the network.

The Need for Fault Tolerant Protocols and Design Challenges: Sensor networks share common failure issues (such as link failures and congestion) with traditional distributed wired and wireless networks, as well as introduce new fault sources (such as node failures). Fault tolerant techniques for distributed systems include tools that have become industry standard such as SNMP and TCP/IP, as well as more specialized and/or more efficient methods researched in [6, 7, 8, 9]. In [6] reliability is achieved through clustering SNMP agents and replication of crashed agents by peer cluster. In [7] authors apply k-nearest neighbor algorithm from data mining to identify poison message failure and to calculate the probability distribution of poison message. [8] focuses on designing a simple network management tool and provides requirements and implementation for a Web interface enterprise network monitoring using Multi-Router Traffic Graphing. Policy-driven approach to fault management technique is studied in [9], where authors describe a rule-based application that is able to monitor a distributed network, identify problems and take corrective actions. Another novel approach to fault management is introduced in [10] where the authors also use web technologies to manage resources. Proposed web-based architecture uses XML notation to specify dependencies between managed resources and provides query facilities to retrieve dependency information between managed resources. Filtered output can be used for further fault diagnosis and recovery.

The faults in sensor networks cannot be approached in the same way as in traditional wired or wireless networks due to the following reasons: (1) traditional network protocols are generally not concerned with energy consumption, since wired networks are constantly powered and wireless ad hoc devices can get recharged regularly; (2) traditional network protocols aim to achieve point-to-point reliability, whereas wireless sensor networks are concerned with reliable event detection; (3) in sensor networks, node failures occur much more frequently than in wired, where servers, routers and client machines are assumed to operate normally most of the time; this implies that closer monitoring of node health without incurring significant overhead is needed; (4) traditional wireless network protocols rely on functional MAC layer protocols that avoid packet collisions, hidden terminal problem and channel errors by using physical carrier sense (RTS/CTS) and virtual carrier sense (monitoring the channel); in wireless sensor networks, MAC layer protocols [11] have to meet other challenges (such as coordinating a node's sleeping and wake times), and can only mitigate the packet collision problem, not completely solve it. These observations indicate that new fault tolerant protocols are necessary for sensor applications to operate successfully and that these protocols should ensure reliable data delivery while minimizing energy consumption.

Taxonomy of Fault Tolerant Techniques: Recent research has developed several techniques that deal with different types of faults at different layers of the network stack. To assist in understanding the assumptions, focus, and intuitions behind the design and development of these techniques, we borrow the taxonomy of different fault tolerant techniques used in traditional distributed systems [12]:

- fault prevention: this is to avoid or prevent faults;
- fault detection: this is to use different metrics to collect symptoms of possible faults;
- fault isolation: this is to correlate different types of fault indications (alarms) received from the network, and proposes various fault hypotheses;
- fault identification: this is to test each of the proposed hypotheses in order to precisely localize and identify faults;
- fault recovery: this is to treat faults, i.e., reverse their adverse effects.

Note that there do exist some techniques that address a combination of all these aspects. In fact, these techniques operate at different layers of the network protocol stack. Most fault avoidance techniques operate in the network layer, adding redundancy in routing paths; a majority of fault detection and recovery techniques operate at the transport layer; and a few fault recovery techniques perform at the application layer, concealing faults during off-line data processing. In this paper, we use the above taxonomy to systematically summarize and compare the fault tolerant techniques that are potentially useful.

The rest of the paper is organized as follows. Section 2 discusses fault prevention by discussing a few network layer techniques for avoiding failures. Section 3 discusses fault detection. Section 4 describes root causing and tracing failures for fault identification and isolation. Section 5 discusses fault recovery techniques for data collection and data dissemination in sensor networks. Section 6 introduces management frameworks for wireless sensor networks. Section 7 highlights open research questions and concludes the paper with suggested future directions.

2 Fault Prevention

Fault prevention techniques aim to prevent faults from happening by (1) ensuring full network coverage and connectivity at the design and deployment stages, (2) constantly monitoring network status and triggering reactive actions if deemed necessary, or (3) enforcing redundancy in the data delivery path, hoping that at least one of the paths will survive and fulfill the task of data delivery.

2.1 Sensor Network Deployment

Sensor deployment plays an important role in ensuring network connectivity and sensing coverage, and consequently, network resilience to faults. Network coverage refers to how well the sensor network covers the area of the phenomena being monitored. Connectivity refers to the ability of active nodes to stay connected. Designing and deploying a sensor network with considerations about connectivity in mind will provide fault tolerance without the need for fault detection or recovery functionality.

The relationship between connectivity and coverage is quantified by previous work [13]. The authors define k-coverage as any location in the network being monitored by at least k nodes. The network is said to be k-connected if the network remains connected even if any k-1 nodes fail. Two protocols to provide coverage and connectivity guarantees are designed. Coverage Configuration Protocol (CPP) allows the dynamic configuration of the network for various degrees of coverage, assuming that the unnecessary nodes can power down to save energy. It is shown that in the case of the communication range being at least double the sensing range, if the network is at least 1-covered, it is also connected. In the case of the communication range being smaller than two times the sensing range, SPAN, a connectivity maintenance protocol, is used in addition to CPP. Sensor network coverage in the worst and the best case for a given network has also been evaluated [14].

Authors in [15] explore the problem of the number of one-hop neighbors per node necessary for the network connectivity as a function of the network size. The study concludes that the number of nodes has to grow as $O(logn)$ with the network size n. Specifically, the number of neighbors per node can be expressed as $clogn$, where asymptotic disconnectivity results for $c = 0.074$ and asymptotic connectivity is achieved for $c = 5.1774$ with the critical value of c being close to 1.

Sensor networks are often deployed in remote and hazardous areas with sometimes extreme conditions. Depending on the nature of monitored environment and obstacles, it is not always possible to plan deployment *a-priori*. To provide coverage and connectivity in a given area with the least number of sensors possible, two sampling based deployment approaches [16] are considered: (1) concurrent deployment, i.e. the number and location of the nodes are decided prior to deployment itself; and (2) incremental deployment, i.e. the feedback about current coverage and connectivity after a sensor is placed is used to decide the location of the next node. For concurrent deployment, it is shown that $\frac{d}{\varepsilon}log\frac{d}{\varepsilon}$ sensor nodes will provide coverage for a unit area with high probability where ε is the sensing radius and d is VC-dimension. In the case of incremental deployment, nodes are placed based on random sampling; however, random sampling without replacement is used — i.e. the areas already covered cannot be randomly picked again. The area covered by sensing range but not covered by the communication range of any sensor is added to the sampling domain.

2.2 Sensor Network Monitoring

Communication in wireless sensor networks is affected by many factors such as environment, network topology, and transmission power. Hence, packet delivery performance may vary dramatically. Researchers have tried to get a quantitative understanding of communication patterns by performing a set of experiments in an office building, a natural habitat, and an open parking lot [1]. Performance of packet delivery at both physical and MAC layers was measured.

- At the physical layer, controlled variables were signal strength, coding scheme, and distance from the transmitter. It was found that up to a certain distance from the transmitter (about 20 meters indoors using high transmission power), packet reception rate is consistently high (between 90% and 100%); after a certain distance (about 30 meters indoors using high power transmission), the packet reception rate is consistently 0%. However, for any distance in between (from 20 to 30 meters indoors using high power), packet reception rate is highly unpredictable. This range is referred to as a "gray area". The size and the starting point of this "gray area" depends on (1) the environment (e.g., it is similar for indoors and outdoors, but the "gray area" is very large in the natural habitat), (2) the transmission power (lower transmission power actually performed better due to reduced likelihood of interference), and (3) the physical layer encoding scheme.

- At the MAC layer, controlled variables were density of deployment and work load. Two metrics were used: packet loss rate and packet delivery efficiency (the ratio of distinct packets). The results indicated that 35% of the links at the low traffic load (less than one packet per second) and 50% of the links at the high traffic load (more than one packet per second) had 50% or more packet loss rate. The efficiency was found to be 50% and 20% for the low traffic load and the high traffic load, respectively. Another important observation was the asymmetry of the links: even indoors more than 10% of the links had a packet loss difference of more than 50% for packets traveling in different directions on the same link.

In addition, empirical studies [4] found that links with very high or very low reception rates tend to be highly symmetrical, while links with intermediate reception rates appear to be much more asymmetric.

Monitoring network health in wireless sensor networks, as in any traditional networks, provides a fundamental support for efficient network management. The captured network status can be used by network administrators to detect or even predict abnormal behaviors and take remedial actions. Generally, we can divide all the monitoring techniques into active and passive monitoring. Active monitoring typically injects probes into the network, and network-internal performance can then be inferred from the measurement parameters. In addition to the techniques purely based on probing, other active monitoring relies on event reports from the managed nodes. Probing packets or event reports can overload the network; therefore, careful calibration of probing frequency and selection of parameters are often needed for active monitoring. In contrast, passive monitoring observes the traffic already present and then infers network condition. We next discuss how node conditions, link status and network congestion level are monitored in sensor networks without incurring significant overhead.

Monitoring Node Status: Since energy is the most scarce resource for sensor nodes, residual energy level provides a good indication of possible node failures. eScan [17] is an active monitoring technique that monitors remaining energy levels using localized algorithms for in-network aggregation of local representations of energy levels. The algorithm starts with each node creating a local scan of its residual energy level expressed as a range of (min, max) instead of a single value. When a user requests a global view of residual energy, individual local scans are transmitted back towards the sink. Reports from neighboring nodes are aggregated en route to the sink if they have similar approximations of residual energy. As a result, the user receives an energy level map of the sensor network. eScan requires the user to issue the request first, which limits the ability of this monitoring tool. In fact, the nodes can be programmed as event-driven and notify the user when energy level drops suddenly for any unpredicted and undetected reasons. In addition, there is an eScan update after each major sensing event, which suggests that each major sensing event would be twice as expensive for transmission of event data and the eScan update.

Alternatively, a prediction based approach has been proposed for generating an energy map [18]. Each node sends the parameters of the dissipation model to the sink. The sink uses this information to locally update the energy level at each node. Nodes will send an update to the sink only if the difference between their actual energy level and the predicted value exceeds a certain threshold. Not having to constantly update energy levels yields great energy savings, however, it comes at the price of lower precision and higher computational overhead on the nodes.

Monitoring Link Quality: Link quality can be measured by the percentage of undamaged packets received. Tracking the quality of channels at the link layer may enable higher level protocols to adapt to changes in link quality by changing routing structures. One technique designed for link quality monitoring is based on snooping [2], by passively listening to the channel and inferring the loss and success rates via tracking of link sequence numbers. This method does not require any extra messages to be exchanged. However, it does involve overhead in listening to the channel. With recent advances in low-power listening technologies, it is possible to keep the cost of snooping low.

Monitoring Congestion Level: Congestion can be one of the causes for packet loss. A straightforward policy is to evaluate the growth rate of the buffer length [19]. If the sum of the current buffer level and the increment in buffer length during the last time period is higher than the buffer capacity, congestion is detected. Alternatively, CODA [20] uses a combination of the present and past channel loading conditions, and the current buffer occupancy to infer accurate detection of congestion at each receiver with low cost. Listening to the channel at all times may incur high energy costs; hence, in order to minimize overhead, CODA only activates local channel monitoring when buffer levels suggest that congestion may be present.

Discussion: Sensor network monitoring should not be limited to just one metric such as residual energy level, link quality, or congestion level. Other metrics such as buffer occupancy level, topology changes, etc., are equally important and should also be tracked. We cannot rely on an administrator to discover and repair failures; instead, sensor network monitoring should be more adaptive and self-configurable. In other words, sensor networks should be able to respond to a certain observed degradation of the network conditions on a local level. Furthermore, the control data packets used for network monitoring should not add substantial additional overhead, so it may be desirable to piggyback as much control data as possible on top of application requested data.

An issue related to network monitoring is "response implosion" [21] – when a large number of nodes respond to a monitoring request simultaneously and thus create bottlenecks in the area of the sink. Three policies are suggested to address this problem: (1) sampling for densely populated sensor networks, where the sink sends a probability p with the diagnostics query, and each node decides whether to report or not with probability p; (2) self-orchestrated operation that schedules responses from all the nodes for sparse sensor networks; and (3) diffused computation where readings are aggregated as the responses move towards the sink. Out of these three suggested methods, diffused computation outperforms the other two in terms of the number of responses received and the overhead incurred.

2.3 Multipath Routing

Multipath routing has been used in traditional wired networks to provide load balancing and route redundancy. Both of these notions are applicable to sensor networks: load balancing leads to a balance in energy consumption among sensor nodes, hence avoiding power depletion of a particular set of nodes; route redundancy increases the chances of messages to reach the destination, thus improving reliability of data delivery. We next discuss three different approaches to utilizing multipath routing.

Meshed Multipath Routing such as Gradient Broadcast (GRAB) technique [22] creates a forwarding mesh from the source to the sink based on the "cost" of delivering the packet at each node. Nodes farther away from the sink have the highest cost of delivering the message. Packets only propagate along the path of least cost towards the sink. To improve the robustness of the protocol, the resulting mesh is widened based on a credit system. The amount of credit assigned by the source node to the packet determines the width of mesh. GRAB works well in dense networks.

Node-Disjoint Multipath [23] relies on a number of alternate paths that do not share any nodes (other than the source and the destination nodes) with the primary path or other alternate paths. This mechanism ensures that failures in any or all nodes on the primary path do not affect alternate paths. Creating multiple disjoint paths would be easier if the global topology is known.

Braided Multipath [23] is a relaxation of node-disjointedness. It uses braided (or partially disjoint) paths. For each node on the primary path, an alternate path not including that node is found. Such alternate paths are not much more expensive than the primary path in terms of latency and overhead. This setup guarantees recovery when only one or a few nodes on the primary path fail. When all or most of the nodes on the primary path fail, new path discovery is required, introducing significant additional overhead.

Comparison: Multipath routing techniques discussed above utilize density of node deployment for reliable data delivery in different ways. Mesh forwarding techniques such as GRAB provide higher reliability than disjoint paths because they use multiple interleaving alternate paths. Consider a scenario where two disjoint paths have failures at different hops. Disjoint multipath would not be able to recover from this fault without constructing a brand new path. Forwarding mesh would reliably deliver data if there was at least one complete forwarding path between source and destination. Forwarding mesh imposes higher overhead because the message is broadcast by more nodes irrespective of whether there are failures or not. On the other hand, under that scheme only one node generates a report about the event so fewer redundant packets are generated.

3 Fault Detection

Since sensor network conditions undergo constant changes, network monitoring alone may be insufficient. Even with fault prevention mechanisms, failures will still occur, so fault detection techniques need to be in place to detect potential faults. Fault detection in sensor networks largely depends on the type of applications and the type of failures.

Similar to wired networks, sensor networks can use packet loss as an indication of faults. In data dissemination protocols which deliver large segments of data to the entire (or part of the) network, the destination nodes are responsible for detecting the missing packet or the window of missing packets, and communicating the feedback to the source using NACK messaging such as in PSFQ [24] and GARUDA [25]. The potential disadvantage of NACK messaging is that the packets need to be cached indefinitely at the intermediate nodes in case the recovery is requested by the downstream nodes. To address this, PSFQ uses on-demand reporting: the source or any intermediate node sets the report bit to 1 in the outgoing packet when it needs to know the latest status of the downstream nodes. The nodes at the farthest hops initiate the report by sending their ID and current packet sequence number upstream. Each subsequent upstream node piggybacks their ID and sequence pair. Upon receiving the report, the source or intermediate node can dispose of the unnecessary packets in cache. In data collection protocols, due to redundancy in sensor nodes and hence the huge amount of reported sensing values, individual packet loss is rarely detected. Instead, a cumulative metric such as packet delivery rate or fault rate is considered [19, 26]. If a certain threshold is exceeded, communication is considered faulty and appropriate recovery actions are taken as discussed in Section 5.

In addition to packet loss, other metrics such as interruption, delay or lack of regular network traffic are also considered as symptoms of faults [27, 28]. Alternatively, buffer occupancy level and channel loading conditions [20, 19] are used for fault detection (specifically, congestion).

Sensor nodes may also permanently fail. Tools such as "ping" or "traceroute" use ICMP messages to check whether a node is alive or not in wired networks. This approach can also be applied to evaluate the health of sensor nodes. In addition, since sensor nodes are energy-constrained and energy depletion often causes node death, remaining energy level can also be used as a warning of node failure [18, 17].

4 Fault Isolation and Identification

With detected alarms, fault isolation and identification processes will diagnose and determine the real causes. When the sink does not hear from a particular part of the routing tree, it is unknown whether it is due to failure of a key routing node, or failure of all nodes in a region. A fault tracing protocol has been proposed [27] to differentiate between these two cases. This is achieved in two steps. Each individual node first piggybacks its neighbor nodes' IDs to the sink along with its own readings so that the sink can have a complete network topology. Failed nodes can then be traced by using a divide-and-conquer strategy based on adaptive route update messages. The sink broadcasts a route update to determine whether the silent nodes are dead. This approach does not perform well in a large scale sensor network: if there are constant failures, the sink would be frequently broadcasting routing updates, which would cause significant overhead. It is desirable that nodes can make some local decisions about fault severity.

Sympathy [28] considers three possible sources of failures for a node: self, path and sink. Sympathy monitors regular network traffic which is assumed to be frequently generated by each healthy node: sensor readings, synchronization beacons, routing updates, etc. Sympathy treats absence of monitored traffic as an indication of faults. It uses metrics traffic generated at the nodes to localize the failure. These metrics include connectivity metrics (e.g., routing table, neighbor list), flow metrics (e.g., packets transmitted and received per node and per sink), and node metrics (e.g., uptime). The measurements expire if they are not updated for a certain period of time. Sympathy determines whether the cause of failure is in node health, bad connectivity/connection, or at the sink by using an empirical decision tree.

5 Fault Recovery

Having discussed various techniques for fault avoidance, detection, and identification, we next discuss how faults can be treated. In general, faults can be (1) discovered and recovered within the sensor network; or (2) concealed at the sink after collecting and analyzing the readings. Our following discussion will focus on the former with a brief discussion on fault masking.

Faults can be recovered independent of applications. For instance, CODA [20] uses two mechanisms to mitigate congestion. When congestion first occurs, a hop-by-hop backpressure message is propagated to the sources. Nodes that receive backpressure signals can adjust their sending rates or drop packets based on the local congestion policy. In case of persistent congestion, the sink suppresses ACKs for a certain period of time. Lack of ACKs is used as an indication for the nodes to reduce their reporting rates. This type of application-independent recovery does not differentiate between important (e.g., a new report) and unimportant packets (e.g., redundant reports, control packets). On the other hand, application-aware fault tolerant protocols try to achieve application specified metrics (e.g., the percentage of distinct packets delivered), which requires the nodes to analyze packets and take different actions based on packet types. We next discuss existing fault tolerant protocols for sensor data collection (i.e., upstream delivery from sensor nodes to the sink) and sensor data dissemination (i.e., downstream delivery from the sink to sensor nodes).

5.1 Reliable Data Collection

Depending on the sensor application needs, either raw or aggregated data need to be collected. Ensuring reliability in collection of raw or aggregated data is provided in different ways.

Collection of Raw Data: A lot of applications do not utilize node computational capabilities; instead, they collect original sensor reports at the sink for current event detection or archiving for future analysis. Therefore, large amount of data is sent to the sink and usually individual packet loss cannot be tracked. However, if enough readings are received to detect an event, the communication is considered reliable.

The specific reliability metric used is slightly different for various reliable protocols. In ESRT [19], desired reliability R is defined as the actual number of reports received sufficient to reliably detect the event. Observed reliability r has to be within the tolerance range $R \pm \varepsilon$. ESRT adjusts the sensor reporting rate based on observed reliability and congestion status to maintain the network in the optimal operating range: sufficient reliability and no congestion.

Alternatively, reliability can be defined as the percentage of the distinct packets received as in PERG [26]. The protocol aims to guarantee reliability in the expected sense. PERG increases or decreases the number of data re-transmissions based on discrepancy between observed and desired reliability. Different from ESRT, PERG does not take any precautions against congestion at all, so PERG exacerbates the situation if congestion does occur. Both protocols do not use feedback from any intermediate nodes; instead, they rely on end-to-end feedback from the server.

Collection of Aggregated Data: When aggregated data needs to be collected, in-network aggregation has been accepted as a standard way to reduce communication overhead by pushing part of the aggregation to some intermediate sensor nodes. TAG [29] uses a routing tree for in-network aggregation. Two techniques are used by TAG to improve tolerance to loss. First, the intermediate nodes cache the most recent data from their children and reuse it when the children do not report the current results. Second, aggregated values are transmitted to multiple parents, reducing the effects of independent single failures. Different from TAG, SKETCH [30] uses a DAG instead of a tree for data delivery. Given that most nodes have multiple parents in a DAG, an individual link or node failure has limited effects. A robust technique for computing duplicate sensitive aggregates was proposed by combining multi-path routing and duplicate insensitive sketches.

5.2 Reliable Data Dissemination

Downstream data transmission is used for re-tasking sensors or reprogramming sensor nodes; this implies that downstream delivery reliability has to be 100%. To ensure robust delivery to each sensor node in a region, the following approaches have been proposed.

Hop-by-Hop Reliability: PSFQ [24](Pump Slowly Fetch Quickly) ensures reliable delivery of data at each hop. PSFQ consists of two basic operations: message relay (pump) and error recovery (fetch). During "pump" operation, packets are slowly paced from the source towards the sink. After receiving a packet, each node stores it in its cache and waits for a random small period of time before forwarding the packet to its neighbors by broadcasting. Adding this waiting period prevents packet redundancy and collision. If loss is detected, a fetch operation is triggered. A node that experiences packet loss tries to recover the missing packets from neighboring nodes. It broadcasts a NACK message to its neighbors that contains the IDs of missing packets. The first node to respond retransmits the missing data.

Minimum Dominating Set: GARUDA [25] ensures reliable data delivery using a core, an approximation of a minimum dominating set of nodes. The first packet of a message determines the hop count of the node. Each node with a hop count that is a multiple of three becomes a core node. This ensures that each node in the network is one hop away from the core. The core is constructed on a per-message basis. GARUDA uses out-of-sequence forwarding; i.e., packets of higher sequence than expected are not suppressed, but forwarded onward. In addition, availability bitmaps of available packets (A-maps) are exchanged between core nodes. This ensures that core nodes only request a missing packet from an upstream node when it is available, preventing an unnecessary chain of NACK's. Lost packets are recovered first on core nodes and then on the rest of the network. This mechanism is problematic if the first packet's reliability cannot be guaranteed. GARUDA addresses this by using a Wait-for-First-Packet pulse, which is a series of short pulses with a higher amplitude and a smaller period than regular traffic. Shortly after the pulse is sent out, the sink broadcasts the first packet. Nodes stop pulsing when they receive the first packet. The pulse serves as an implied NACK for those nodes that still have not received the first packet.

Combining Source and Local Recovery: ReACT [31] is a reliable multicast data dissemination protocol designed for multi-hop wireless ad hoc networks. The essence of ReACT is to distinguish between localized and global fault sources and to apply local or source recovery, respectively. In ReACT, nodes monitor their queues to determine congestion status. A special flag is set by intermediate nodes to notify the downstream nodes when congestion is present. Each node maintains a neighbor table with congestion information, which will be used to determine whether a packet drop is caused locally or by network congestion. Either local or source recovery is then triggered accordingly. In local recovery, only one non-congested nearby node is needed. In source recovery, nodes send a NACK to the source with missing packet numbers. The source then multicasts the missing packet to all nodes that miss it. The source also reduces the transmission rate if the congestion is persistent. ReACT can be modified to take into account energy constraints and the high failure rate of wireless sensor networks, which are a subset of multi-hop wireless ad hoc networks.

Discussion: To summarize fault recovery protocols discussed in this section, we compiled Table 1 to highlight main features and limitations of each solution. One important difference is the reliability notion used. When dealing with re-tasking or reprogramming nodes, 100% data delivery is necessary; while for event detection, 100% sensor readings delivery may be wasteful of energy. ESRT and PERG try to achieve only a certain degree of reliability, while PSFQ provides 100% reliable delivery. GARUDA defines four different notions of reliability: delivery to all nodes in the entire network, delivery to all nodes in a subregion, delivery to a minimal number of sensors to cover the entire field, and delivery to a probabilistic subset of sensors. Consequently, different methods are used to ensure desired reliability. ESRT and PERG adjust reporting/retransmission rate for all nodes in the entire network. PSFQ and GARUDA use more localized and more fine-grained methods for reliable delivery, such as hop-by-hop recovery or 2 stage recovery (i.e, first core node, then non-core nodes). Finally, it is important to note that ESRT is the only one of the studied methods that provides congestion resolution.

Table1. Comparison of Fault Recovery Techniques

The fault tolerant data collection and data dissemination techniques discussed above function as transport layer protocols; and hence, operate with the assistance of the sensor nodes from where the data originates. An alternative is to completely relieve the nodes from the burden of recovery from faults, and perform all the necessary treatment at the sink. Typically, a model of sensor data is needed and missing data can be interpolated using spatial and temporal correlations among sensor readings [32, 33, 34, 35, 36].

6 Fault Management Frameworks

Fault management frameworks address faults as part of a larger network management structure. Such solutions approach fault management at a higher level; e.g. by designing the management infrastructure and information model. These frameworks can be complemented by the specific fault detection and recovery techniques discussed previously. A number of such frameworks have been introduced for either ad hoc networks or wireless sensor networks.

Guerilla [37] describes a distributed network management architecture with SNMP agents residing on every node. Along a similar line, a probe-based architecture and an information model (both node-scale and network-scale) [38] are proposed for managing ad hoc networks. Nodes calculate local individual network metrics, such as the MAC layer contention derived from the number of RTS packets sent and CTS packets received, and the node routing participation derived from the number of packets that originate at the node and the number of routed packets. These metrics can be used to build the entire network model in order to evaluate the network behavior and take actions if needed. In the probe-based architecture, a set of probes spread out across the network (as opposed to each individual node) capture packets in their range, calculate the above metrics and send the metrics to a network manager. The network manager then builds the global network view combining the views of different network areas received from the probes.

While Simple Network Management Protocol (SNMP) has been one of the dominating management protocol used in wired networks, studies exist on the design of management protocols for ad hoc networks. For instance, Ad Hoc Network Management Protocol (ANMP) [39] uses hierarchical clustering to reduce the number of messages exchanged between the manager and the agents.

Although underlying principles of managing ad hoc networks can also be applied to sensor networks, there are a number of management systems that have been designed and developed specifically for wireless sensor networks. These systems include Digest [40], Sympathy [41, 28], NOSY [42], SNMS [43], AgletBus [44, 45], and MANNA [46]. More specifically, Digest [40] is an architecture to monitor wireless sensor networks with different levels of detail, and it focuses on the design of continuous computing summaries of network properties. Sympathy [41, 28] is a tool for debugging and detecting failures in sensor networks. NOSY [42] is a centralized network monitoring system that keeps track of the progress of code dissemination, adjusts the sensor reporting frequency, pulls information from an individual sensor if needed, and reboots a node if no messages are received for an extended period of time. SNMS [43] is a middleware layer that provides a set of core management services, such as enumerating sensor nodes, remote power management, monitoring physical parameters of sensor nodes, etc. AgletBus [44, 45] is a management centric middleware that provides a consistent and transparent framework for both inter- and intra-nodal coordination and management. Similar to SNMS, AgletBus also includes a set of services such as leader election, event forwarding and power management service.

MANNA [46] is a policy-based network management system for wireless sensor networks. Depending on the network topology and characteristics (homogeneous vs. heterogeneous), MANNA assigns different roles (network managers or agents) to various sensor nodes. These nodes exchange request or response messages with each other for management purposes. MANNA forms a basis for fault management [47], one of several network management services supported by this architecture. Fault management in MANNA mainly relies on the coverage area maintenance service and the failure detection service. Faults are detected in two phases in MANNA. In the installation phase, nodes report their location and energy level to the manager via the agents. The network manager builds coverage and energy models based on the initial information. During the operational phase, nodes update their location or energy whenever there is a change in their state. The network manager periodically performs network auditing by retrieving a node state. If a node which has enough remaining energy according to the energy model does not respond to the auditing, a fault is detected.

7 Open Research Issues

The future vision of wireless sensor networks is to embed numerous tiny sensor nodes in unattended places or systems to monitor and interact with physical world phenomena. These nodes coordinate among themselves to create a network that performs higher level tasks irrespective of all types of interruptions.

The protocols studied in this paper differ from each other in terms of the types of faults they address, the way they address those faults, and the notion of reliability. Ideally, a fault tolerant protocol works for both upstream and downstream data delivery across different underlying topologies. Although the performance of these protocols reviewed in this paper is promising in terms of robustness and energy efficiency, further research is needed to address the scalability and network dynamics in designing fault tolerant protocols.

- Scalability: the number of sensor nodes deployed in the sensing area may be on the order of hundreds or thousands, or more. Techniques developed or tested in smaller networks may simply not scale well, because of the overhead involved.

- Network dynamics: sensor nodes as well as the observed phenomena can be mobile, which introduces another dimension of complexity and also another possible source of failures. This illustrates the need for the network self-configuration and reconfiguration, which is essential to support sensor applications in a dynamic and energy constrained environment.

We next suggest several open research issues in designing fault tolerant protocols in sensor networks.

A cross layer approach to fault tolerance: When there exist severe faults in sensor networks, MAC and routing protocols must accommodate formation of new links and routes to the destination, transport protocols must adaptively decide how to retransmit, and application layer protocols must determine which part of the missing data is critical and what level of loss is tolerable. Therefore, multiple levels of redundancy may be needed and a cross-layer approach exploring the interactions among different layers is desirable.

Recovery from composite faults including congestion: As can be seen from our previous discussions, none of the protocols can recover from all different types of faults. Especially, only ESRT [19] addresses congestion in an efficient manner. We believe that there is a need for a more robust transport layer solution that can recover from node failures, link failures and network congestion. Ideally, the method combines the winning features of the studied protocols in an elegant manner, including quick node-by-node recovery from PSFQ [24], simple congestion detection and handling from ESRT [19], and various reliability notions [25].

Composition of fault tolerance and timeliness requirements: All the protocols discussed in this paper only consider reliability of data delivery as a performance metric. In fact, timeliness is also critical for many sensor applications, such as disaster response, fire controlling, earthquake response, military surveillance, and intrusion detection. The introduction of timeliness complicates the problem furthermore. Additional issues to consider to consider are: (1) the tradeoffs between reliability, timeliness, and energy efficiency; and (2) the applications' preferences if reliability and timeliness needs cannot be satisfied simultaneously.

Data semantics in the presence of faults: The presence of faults in sensor networks introduces uncertainty into standard operations such as answering queries. We should not aim to extract some data from the network in a purely best-effort manner, but rather, to produce results with a clearly defined formal meaning. For instance, it is possible that only a subset of the sensor readings satisfies the application query. Therefore, the network only reports part of the readings filtered by the query. However, the sink does not know whether the remaining reports were not received because of the network faults or because the results were filtered by the query. If we can provide a metric that indicates the completeness of the returned answer, the sink would be better informed. Therefore, it is essential to develop informative quality metrics for sensor applications.

8 Conclusions

This paper reviewed current techniques for dealing with faults in wireless sensor networks. We started with a taxonomy of the sources of faults in wireless sensor networks. We then discussed the algorithms that prevent, detect, identify, isolate, and treat faults. Next, several management infrastructures for sensor networks and ad hoc networks were summarized. Based on our understanding, we also highlighted several possible future research directions. The resource limitation and unattended feature of sensor networks renders the network very faulty, and this survey aims to help us further understand the challenges in designing fault tolerant protocols for distributed sensor applications.

Most fault management techniques in sensor networks have been integrated with application requirements, differently from fault management in traditional networks. The primary reason for this is that sensor networks are resource constrained and direct application of traditional fault management techniques would incur significant overhead. Design of a generic fault management technique for sensor networks must take into account a wide variety of sensor applications with diverse needs, different sources of faults, and various network configurations. In addition, scalability, mobility, and timeliness may have to be considered.

References

[1] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of ACM SenSys*, 2003.

[2] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of ACM SenSys*, 2003.

[3] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of ACM MobiSys*, pages 125–138, 2004.

[4] A. Cerpa, J. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of IEEE IPSN*, April 2005.

[5] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless micro-sensor network models. *Mobile Computing and Communications Review*, 6(2), 2002.

[6] A. L. Dos Santos, Jr. E. P. Duarte, and G. M. Keeni. Reliable distributed network management by replication. *J. Netw. Syst. Manage.*, 12(2):191–213, June 2004.

[7] X. Du. Identifying control and management plane poison message failure by k-nearest neighbor method. *J. Netw. Syst. Manage.*, 14(2):243–259, June 2006.

[8] J. W. Hong, S. Park, Y. Kang, and J. Park. Enterprise network traffic monitoring, analysis, and reporting using web technology. *J. Netw. Syst. Manage.*, 9(1):89–111, March 2001.

[9] H. L. Lutfiyya, M. A. Bauer, A. D. Marshall, and D. K. Stokes. Fault management in distributed systems: A policy-driven approach. *J. Netw. Syst. Manage.*, 8(4):499–525, December 2000.

[10] C. Ensel and A. Keller. An approach for managing service dependencies with xml and the resource description framework. *J. Netw. Syst. Manage.*, 10(2):147–170, 2002.

[11] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of InfoCom*, 2002.

[12] A. S. Tanenbaum and M.V. Steen. *Distributed Systems: Principles and Paradigms.* Prentice Hall, 2002.

[13] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. 2003.

[14] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings of IEEE Infocom*, 2001.

[15] F. Xue and P. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10(10):169–181.

[16] V. Isler, K. Daniilidis, and S. Kannan. Sampling based sensor-network deployment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[17] Y. Zhao, R. Govindan, and D. Estrin. Residual energy scan for monitoring sensor networks. In *Proceedings of IEEE WCNC*, 2002.

[18] R. Mini, A. Loureiro, and B. Nath. The distinctive design characteristic of a wireless sensor network: the energy map. *Elsevier Computer Communications*, 27(10), 2004.

[19] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. Esrt: Event-to-sink reliable transport in wireless sensor networks. In *Proceedings of ACM MobiHoc*, 2003.

[20] C.Y. Wan, S.B. Eisenman, and A.T. Campbell. Coda: Congestion detection and avoidance in sensor networks. In *Proceedings of SenSys*, 2003.

[21] C. Jaikaeo, C. Srisathapornphat, and C. Shen. Diagnosis of sensor networks. In *Proceedings of IEEE ICC*, 2001.

[22] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *ACM WINET*, 11(2), 2003.

[23] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM Mobile Computing and Communications Review*, 1(2), October 2002.

[24] C.Y. Wan, A.T. Campbell, and L. Krishnamurthy. Pump slowly fetch quickly(psfq): A reliable transport protocol for wireless sensor networks. *IEEE JSAC*, 23(2), 2005.

[25] S. J. Park, Ra. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *ACM MobiHoc Conference*, 2004.

[26] Q. Han, I. Lazaridis, S. Mehrotra, and N. Venkatasubramanian. Sensor data collection with expected reliability guarantees. In *Proceedings of IEEE PerSeNS*, 2005.

[27] J. Staddon, D. Balfanz, and G. Durfee. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 122–130, 2002.

[28] N. Ramanathan, K. Chang, L. Girod, R. Kapur, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of SenSys*, pages 255–267, 2005.

[29] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *USENIX OSDI*, 2002.

[30] J. Considine, F. Li, G. Kollios, and J. Brers. Approximate aggregation techniques for sensor databases. In *Proceedings of IEEE ICDE*, 2004.

[31] V. Rajendran, K. Obraczka, Y. Yi, S. Lee, K. Tang, and M. Gerla. Combining source- and localized recovery to achieve reliable multicast in multi-hop ad hoc networks. In *IEEE Networking*, 2004.

[32] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell. Fault tolerance techniques for wireless ad hoc sensor networks. *Proceedings of IEEE*, 2:1491–1496, June 2002.

[33] T. Clouqueur, K. K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. *IEEE Transactions on Computers*, 53(3):320–333, March 2004.

[34] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of IPSN*, 2004.

[35] G. Hartl and B. Li. Loss inference in wireless sensor networks based on data aggregation. In *Proceedings of IEEE IPSN*, 2004.

[36] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model driven data acquisition in sensor networks. In *VLDB Conference*, 2004.

[37] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo. An adaptive management architecture for ad hoc networks. *IEEE Communication Magazine*, 41(2), February 2003.

[38] R. Badonnel, R. State, and O. Festor. Management of mobile ad hoc networks: information model and probe-based architecture. *International Journal of Network Management*, 15(5):335–347, 2005.

[39] W. Chen, N. Jain, and S. Singh. Anmp: Ad hoc network management protocol. *IEEE JSAC*, 17(8), August 1999.

[40] Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of SNPA*, 2003.

[41] N. Ramanathan, E. Kohler, and D. Estrin. Towards a debugging system for sensor networks. *International Journal of Nerwork Management*, 15, 2005.

[42] D. Starobinski. Network observation system (nosy). http://nislab.bu.edu/nislab/projects/wsn_testbed/nosy.ht

[43] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *Proceedings of EWSN*, 2005.

[44] J. Lim, D. Kiskis, and K. Shin. Aglet: Modular coordination and management framework. EECS, University of Michigan, Ann Arbor.

[45] J. Lim, D. Kiskis, and K. Shin. System support for management of networked low-power sensors. In *Proceedings of IEEE/IFIP NOMS*, 2006.

[46] L.B . Ruiz, J.M. Nogueira, and A.A.F. Loureiro. Manna: A management architecture for wireless sensor networks. *IEEE Commmunications Magazine*, 41(41):116–125, 2003.

[47] L.B. Ruiz, I.G.Siqueira, L.B. e Oliveira, H.C. Wong, J.M.S. Nogueira, and A.A.F. Loureiro. Fault management in event-driven wireless sensor networks. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2004.

Lilia Paradis is currently a graduate student in the Department of Mathematical and Computer Sciences, Colorado School of Mines. She is also part of the Toilers Ad Hoc Networking research group. She is interested in distributed communication protocols for wireless sensor networks.

Qi Han received the PhD degree in computer science from the University of California, Irvine in 2005. She is currently an assistant professor in the Department of Mathematical and Computer Sciences, Colorado School of Mines. Her research interests include distributed systems, middleware, mobile and pervasive computing, systems support for sensor applications, and dynamic data management. She is specifically interested in developing adaptive middleware techniques for next generation distributed systems. She is a member of the IEEE and the ACM.

Parameter	ESRT	PERG	PSFQ	GARUDA
Data flow	upstream	upstream	up- and downstream	downstream
Goal	app. specified reliability	prob. guarantee	100% reliability	4 notions*
Topology	direct communication	tree	chain or tree	sink in the middle
Method	reporting rate	retransmission rate	hop-by-hop recovery	min dominating set
Congestion	prevention and recovery	not addressed	prevention	prevention
Assumption	direct communication	no congestion	no congestion	reliable 1st message

*100% reliability, probabilistic reliability, subregion, minimum network coverage

Table 1: Comparison of Fault Recovery Techniques