Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A cross-layer fault tolerance management module for wireless sensor networks

Ozlem KARACA†1, Radosveta SOKULLU2
(1Izmir Vocational School, Dokuz Eylul University, Izmir 35160, Turkey)
(2Department of Electrical and Electronics Engineering, Ege University, Izmir 35100, Turkey)
†E-mail: ozlem.karaca@deu.edu.tr

Received Feb. 3, 2012; Revision accepted May 29, 2012; Crosschecked Aug. 3, 2012

Abstract: It is a well-established fact that wireless sensor networks (WSNs) are very power constraint networks, but besides this, they are inherently more fault-prone than any other type of wireless network and their protocol design is very application specific. Major reasons for the faults are the unpredictable wireless communication channel, battery depletion, as well as fragility and mobility of the nodes. Furthermore, as traditional protocol design methods have proved inadequate, the cross-layer design (CLD) approach, which allows for interactions between different layers, providing more flexible and energy-efficient functionality, has emerged as a viable solution for WSNs. In this study we define a fault tolerance management module suitable to the requirements, limitations, and specifics of WSNs, encompassing methods for fault detection, fault prevention, fault management, and recovery. The suggested solution is in line with the CLD approach, which is an important factor in increasing the network performance. Through simulations the functionality of the network is evaluated, based on packet loss, delay, and energy consumption, and is compared with a similar solution not including fault management. The results achieved support the idea that the introduction of a unified approach to fault management improves the network performance as a whole.

Key words: Wireless sensor networks (WSNs), Cross-layer frameworks, Protocol design, Fault management
doi:10.1631/jzus.C1200029 **Document code:** A **CLC number:** TP393

1 Introduction

Wireless sensor networks (WSNs) are inherently more fault prone than any other type of wireless network. There are a number of reasons for this: continuously changing and unpredictable wireless communication channel, battery depletion that leads to changing levels of available power, and fragility and mobility of the nodes, to name only a few major reasons. As errors cannot be prevented, an important issue is to provide methods for minimizing and limiting the spreading of the error in the network. However, the faults in WSNs cannot be approached in the same way as in traditional wired or wireless ad hoc networks because of the following major differences: traditional networks are not limited in power supply and devices in ad hoc networks can be regularly recharged, while WSNs are usually designed and deployed to function without possibilities of recharging or maintenance; existing protocols for reliability like TCP/IP provide point-to-point reliability (Stewart and Metz, 2001; Drabkin *et al*., 2008), while WSNs aim at reliable event detection (Gungor *et al*., 2007), so the traditional point-to-point reliability is not an issue; traditional networks rely on functional medium access control (MAC) protocols to provide efficient mechanisms for accessing the medium and avoiding collision, while in WSNs, MAC protocols have to deal primarily with issues of energy efficiency and scheduling, thus only partially preventing errors. Another very important issue is that the layered protocol structure that traditional networks rely on is often replaced in WSNs by a cross-layer design (CLD) approach, which allows for interactions between different layers and provides more flexible and energy-efficient functionality.

It is obvious that because of these differences, as well as the great importance placed on fault tolerance in the operation cycle of the WSNs, new fault management techniques have to be developed to face these aspects. In this work we define a fault tolerance management module (FTMM) suitable to the requirements, limitations, and specifics of WSNs. It encompasses methods for fault detection, fault prevention, fault management, and recovery. The suggested solution is in line with the CLD approach, which has proved to be an important factor in increasing network performance for WSNs (Shakkottai *et al.*, 2003; Srivastava and Motani, 2005). When severe faults occur in WSNs, MAC and routing protocols must provide new links and routes, transport protocols must adaptively resolve retransmission issues, and application layer protocols must decide what level of loss is tolerable. Thus, multiple levels of redundancy may be needed and a cross-layer approach exploring the interactions among different layers is desirable (Paradis and Han, 2007). The suggested module uses the benefits of CLD and provides a dual action: helps prevent the occurrence of severe faults that disrupt the operation of the WSN and helps decrease the effects of errors which can not be prevented. Comparative simulation results with a an existing CLD based system without fault tolerance operation are presented which clearly prove the improved performance of a network incorporating the FTMM in terms of goodput, packet loss, and prolonged network lifetime as a result of reduced energy consumption.

2 Fault tolerance and fault management in WSNs

2.1 Major issues and definitions

Fault tolerance is a function required in WSNs. It is defined as the ability of the network to continue functioning in the presence of link and/or node failures (Akyildiz *et al.*, 2002). The level of fault tolerance is highly dependent on the specific WSN application.

According to Khan *et al.* (2008), a good solution regarding fault tolerance has to comply with two very important characteristics: it has to be both energy-aware and vulnerability-aware (Khan *et al.*, 2008). Paradis and Han (2007) presented a detailed investigation on the different aspects of fault management in WSNs. According to their taxonomy, methods for dealing with the faults can be summarized into four different groups:

1. Fault prevention: methods that aim to prevent and minimize the occurrence of faults by (1) ensuring full network coverage and connectivity during the design and development stages, (2) providing constant monitoring of the network and the nodes and triggering reactive actions when required, and (3) providing redundancy in nodes and connections.

2. Fault detection: methods that aim to detect signs of abnormal network operation based on monitoring specific networks performance parameters like packet loss and delay.

3. Fault identification: using the information collected from monitoring the network operation, different hypotheses are developed for the possible origin and type of the faults.

4. Fault recovery: methods that minimize the effect of the faults or, if possible, provide recovery.

Faults in WSNs might be due to reasons very different from those in traditional networks. For example, while packet loss or extensive delay is a clear indication of congestion in wired networks, in WSNs we might define at least three different reasons for this: node response failure (due to battery depletion, node failure, or mobility), failed link, or congestion in the network.

The resource limitation and unattended nature of most WSNs make fault tolerance a very important issue for this type of network. As traditional protocols are not applicable, the design of new methods suitable to this type of network becomes an important research issue. Thus, the fault management framework is defined as a generic unified structure that takes into consideration a variety of different applications and diverse sources of faults.

2.2 Related work

There is a sizeable amount of literature dealing with the subject of fault tolerance in WSNs. Different authors have suggested quite diverse methods and provided solutions including one or more of the major functions required for dealing with faults. The analysis of the existing literature presented below is based on some major network parameters used, namely remaining energy, buffer level, packet loss, latency, and network lifetime.

Energy supply is the most restricted resource in a wireless node; thus, the decreased energy levels are an important warning for possible fault occurrence (Zhao *et al*., 2002; Mini *et al*., 2004). Taking this factor into consideration, Zhao *et al*. (2002) proposed eScan, a mechanism designed to monitor the remaining energy levels of the nodes in the network. It is based on in-network aggregation of energy levels for different regions. These scans are updated when a node's state changes, thus providing a current picture of the network energy state at a minimal resource cost. The monitoring process is started by the user. Each node starts a local energy scan where the energy is specified as a value in an interval (min, max) instead of a single value. The receiving node collects information from neighboring nodes along the route, thus providing collective information about the available energy on that specific route. The collected information can help in notifying the user of depleting energy resources or unpredicted abnormal activities in the network. In Zhao *et al*. (2002), the term 'energy map' was defined as the "collective information about the amount of available energy in each part of the network". The authors argued that it is a very important parameter in ensuring prolonged network lifetime and that the whole WSN design should be based on the energy map. Obviously, for WSNs it is not feasible to keep such monitoring continuously or on an individual node basis. The fact that this algorithm is launched by the user is considered a disadvantage since it can reduce its monitoring efficiency. Another possible option could be to allow periodic scheduling or event-triggered operation. In our proposal we have adopted an event-triggered monitoring approach, related to the sleep-awake cycle of the nodes.

In Mini *et al*. (2004), two different methods were suggested to predict the energy levels of the nodes and create an energy network map of the network. The aim is to prolong the network lifetime using this energy map. The authors have shown how to construct an energy map using both probabilistic and statistical prediction-based approaches. Simulation results compared the performance of these approaches with that of a naive one, in which no prediction is used. The results showed that prediction-based approaches outperform the naive in terms of a variety of parameters. In a similar way as described in Zhao *et al*. (2002), the process of monitoring has to be initiated by the user. However, if an instantaneous change occurs in the system, it might be missed. Zhao *et al*. (2002), Mini *et al*. (2004), and Vidhyapriya and Vanathi (2008) stressed the importance of closely following the remaining energy of the nodes as a vital parameter in WSN. Furthermore, it is a good indicator of the possible presence of errors. When the energy of a node is reduced, the node will shortly die, which will then reflect on the network in increased delays and packet losses or even network operation breakdown. Thus, the process of keeping track of the remaining energy level of a node can be viewed as a mechanism of an early warning system.

Sankarasubramaniam *et al*. (2003) dealt with monitoring the congestion in the network. The authors argued that if congestion occurs, the nodes' buffer occupation levels will increase. Then, if the total available buffer space stays above a given limit for a certain period of time, this will be an indication that congestion has occurred. Thus, by monitoring the network and getting indication of possible congestion, which can give rise to errors, decisions and actions can be taken at an early stage and interruption in the network operation will be diminished or avoided. The important point is the correct determination of suitable monitoring intervals and specific buffer levels so that possible adverse effects of increased buffer occupation levels could be appropriately minimized.

In Wan *et al*. (2003), the condition of the node buffer was investigated in a similar way as in Sankarasubramaniam *et al*. (2003), except that the channel state was controlled. Sudden changes in the channel conditions, which are being periodically observed, can be treated as a warning. The changes in the channel and the buffer conditions together are an indicator of congestion, so fault detection can be executed using this information. Observing the channel is performed by listening to the channel; thus, it leads to consuming a certain additional amount of energy. Continuously listening to the channel, especially in WSNs, will be an inefficient method, because energy is the most restricted resource. In this study a particular sampling rate is used for listening to the channel, which allows for getting enough information about the channel while consuming minimal energy.

Ergen and Pravin (2007) used delay as a parameter that indicates the presence of fault. This study claims that if measures are taken to keep the maximum allowed delay limited and under control, optimal lifetime for the whole network can be achieved. The simulation results support this claim.

Packet loss is yet another major indicator for fault detection. In Wan *et al.* (2005), a reliable data transmission protocol for WSN was presented. The authors have aimed at hop-by-hop reliable data transmission rather than end-to-end reliability. The distributed system solution proposes that if a node experiences packet loss, it should attempt to receive the same packet from other one-hop distance neighbors. The destination node reports back to the sender-neighbor when the packet is received. Even though this is a well-thought and distributed system solution, it may increase packet traffic quite significantly in busy WSNs.

Park *et al.* (2004) dealt with the problem of reliable downstream point-to-multipoint (sink-to-sensor) data delivery. Their proposed approach, named GARUDA, consists of several submodules. Of special interest is the recovery process based on a two-stage NACK. The information contained in the header of the NACK is analyzed to minimize the overhead in the retransmission process. The recovery itself is done as localized (defined as core) and then if necessary for the rest of the nodes (defined as non-core), thus reducing the possibility of increasing the congestion while performing retransmissions.

One of the ways to increase reliability in WSNs is to use alternative routes between the source and the sink. What is actually important in most WSN applications is not that a specific single packet has reached the sink but that the information related to a specific event has been properly reported to the sink. Han *et al.* (2005) investigated the packet loss and the probability of error. In the system under consideration, the application layer has to receive information from a certain number of nodes in a specific interval. Based on the average of the acquired results the application takes a decision. If the amount of received data (packets) is too little, the decision might not be correct. So, the authors proposed a solution to keep track of the number of received packets in a given interval and to keep it above a certain predefined level (threshold). The key factor is regulating the number of retransmissions. This work points to a specific relationship between the number of retransmissions and the packet loss, which is also accepted in our approach. When packet loss increases, additional retransmissions are required but if their number is too high it will only make the situation worse by creating additional congestion in the network. Thus, by keeping track of the changes in the number of required retransmissions, a node can get initial information about the presence of congestion conditions in the network, or vice versa, by reducing that number (or preserving a given threshold value), congestion can be alleviated at the price of limited packet loss.

Together with packet loss, traffic load in the network can also be viewed as an indicator of worsening network operation. Increased delay and unexpected reduction in the incoming traffic can be considered direct indicators for the presence of faults. Staddon *et al.* (2002) proposed an algorithm for tracing down dead nodes in networks with a base station. The nodes are classified as ‘dead’ or ‘alive’, regarding their energy levels and, ‘silent’ or ‘traced’, regarding their ability to receive/send packets. If a node seizes to send measurements to the base station it is either because it has died or because its immediate neighbor (over which it is routing the packets to the base stations) has died. The authors argued that the proposed algorithm allows the base station to get information about the topology of the network at a much lower communication overhead cost then other similar algorithms. Additionally, this work provides insights on how to identify and isolate the non-functional nodes (faulty nodes) based on their communication with the neighboring nodes.

Sympathy, as suggested in Ramanathan *et al.* (2005), is a tool proposed for detecting and debugging failures in WSNs. The authors proved that, for a large class of information gathering applications, it is possible to diagnose errors based on a limited set of parameters. One of them is traffic and more specifically the lack of or reduced traffic. A normally working node will continuously generate some kind of traffic, consisting of packets related to sensed events, synchronization, routing updates, etc. The generation of less than the expected traffic in a given area is used as a trigger for the fault detection and localization algorithm. Besides, the information kept in neighbor lists and routing tables is used as a connection measure, while the number of transmitted packets and the number of received packets at the sink are used as measures of the traffic flow. On the other hand, information about the node state is derived from the regular node updates. This work provides a lot of insight on the metrics and methods that can be used for the timely detection of errors in WSNs.

Our work builds on some of the ideas suggested in previous studies, especially those related to the metrics involved in fault detection. However, our proposed fault management scheme incorporates a number of metrics that have so far only been taken into account separately. Furthermore, the system we propose is distributed and built using a CLD methodology, which allows a unified approach to fault detection, identification, and recovery. In the following sections we present the details of the system as well as comparative simulation results.

3 Proposed system's building blocks

The fault management system we propose has a modular structure and uses the taxonomy presented in Paradis and Han (2007). The major modular blocks defined comprise fault prevention, fault detection and identification, and fault recovery procedures (Fig. 1).

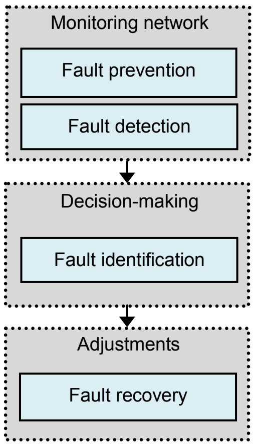

Fig. 1 Structure of the proposed fault management system

It is a distributed system, as each node relies on its own information and information collected from one-hop neighboring nodes regarding the major parameters observed. Furthermore, as all the procedures rely on the cross-layer information exchange, both from lower to upper protocol layers and from upper to lower protocol layers, the system is in the true sense a CLD approach. The system model we consider is detailed below.

The network in consideration consists of nodes that are aware of their coordinates and the coordinates of the sink node. Depending on these coordinates the network is divided into two areas: the faraway area RFA, defined as farther from the sink as related to the current node, and the close by area RCB, defined as the area closer to the sink as related to the current node. Every node can decide whether a specific request packet has come from the faraway area or from a close by area, and by definition, only nodes found closer to the sink will answer.

The network operates in a fully distributed manner. Each node decides by itself whether to participate in the communication process or not, based on the value of the 'participation determination' (PD) parameter. The value of PD is determined by the energy level of the node at that moment, by the buffer occupancy, by the number of packets sent, the signal-to-noise ratio (or SNR, inferred from the received packet), and the response coming from the FTMM module.

A unified cross-layer module, called XLM, was developed in Akyildiz *et al.* (2006), achieving efficient and reliable event communication in WSNs with minimum energy expenditure. This work introduces the idea of using a PD parameter. Based on the PD concept, XLM performs receiver-based contention resolution, local congestion control, and distributed duty cycle operation to realize efficient and reliable communication. The value of the PD parameter is set to 1 if the first four conditions in Eq. (1) are satisfied. Each condition in Eq. (1) constitutes a certain communication functionality.

$$\mathrm{PD}=\begin{cases}1, & \text{if } \begin{cases}\xi_{\mathrm{RTS}}\ge\xi_{\mathrm{RTS}}^{\mathrm{thrv}},\\ P_i^{\mathrm{TX}}\le kP_{\mathrm{TX}}^{\mathrm{thrv}},\\ B\le B_{\max}^{\mathrm{thrv}},\\ E_{\mathrm{rem}}\ge E_{\mathrm{rem}}^{\min\,\mathrm{thrv}},\\ \mathrm{FT}_{\mathrm{action}}\le\mathrm{FT}_{\mathrm{action}}^{\mathrm{thrv}},\end{cases}\\ 0, & \text{otherwise,}\end{cases} \tag{1}$$

where ξ_{RTS} is the current value of the SNR obtained from the received RTS packet, $\xi_{\mathrm{RTS}}^{\mathrm{thrv}}$ is the threshold value of the accepted minimum SNR, P_i^{TX} is the number of packets transmitted by node i and $P_{\mathrm{TX}}^{\mathrm{thrv}}$ is the threshold accepted as a corresponding maximum, B and $B_{\max}^{\mathrm{thrv}}$ are respectively the current value of the buffer and the maximum allowed buffer occupation threshold value per node, E_{rem} and $E_{\mathrm{rem}}^{\mathrm{min\,thrv}}$ are the currently remaining energy level and the accepted threshold value respectively, and $\mathrm{FT}_{\mathrm{action}}$ and $\mathrm{FT}_{\mathrm{action}}^{\mathrm{thrv}}$ are respectively the values coming from the FTMM module and the threshold, which refers to the set of initial system values.

Our work is inspired by XLM and also uses the PD parameter. However, unlike XLM, in our proposed scheme, the definition of the PD parameter first requires a check of the value originating from the fault management module. Also, we introduce several threshold levels, which can be controlled by the result coming from the fault management module.

Each node operates on a duty cycle consisting of a sleep period and an awake (active) period. The value D defines the active portion of the cycle, and

$$0 \le D \le 1. \tag{2}$$

If D=0, then the node does not have any active time, while D=1 shows that a node is constantly awake (i.e., it is transmitting and receiving packets all the time).

Every node follows a 5 s period (C) of frame transmission (Akyildiz *et al.*, 2006; Gupta *et al.*, 2009). The value D defines the length of active state. While in active state, the node i is occupied with receiving packets for period of time Z_i^{RX}, processing packets coming from the upper layers and forwarding packets (transmitting) for time Z_i^{TX}, or listening to the channel for a period Z_i^{LIN}. Z_{PKT} is the average time required for the successful transmission of a packet from one node to the next neighbor. Finally each node is active for DC seconds and is asleep for $(1-D)C$ seconds. Thus, for the operations executed by node i while in active state we can define Eqs. (3) and (4):

$$Z_i^{\mathrm{RX}} = Z_{\mathrm{PKT}} \sum_{j \in N_i^{\mathrm{int}}} P_j^{\mathrm{TX}}, \tag{3}$$

$$Z_i^{\mathrm{TX}} = Z_{\mathrm{PKT}} \rho \left(P_{ii} + \sum_{j \in N_i^{\mathrm{int}}} P_j^{\mathrm{TX}} \right), \tag{4}$$

where ρ is the packet retransmission coefficient and is defined as

$$\rho = \frac{P_i^{\mathrm{TX}} + P_i^{\mathrm{fault}}}{P_i^{\mathrm{TX}}}, \tag{5}$$

with P_i^{TX} being the number of packets transmitted by node i and P_i^{fault} the number of packets that could not be transmitted successfully.

$$P_i^{\mathrm{TX}} = P_{ii} + \sum_{j \in N_i^{\mathrm{int}}} P_j^{\mathrm{TX}}, \tag{6}$$

where P_{ii} denotes the number of packets generated by node i and P_j^{TX} denotes the number of packets relayed from neighbor node j in the area RCB.

The time that the node spends on listening is defined as

$$\begin{aligned} Z_i^{\mathrm{LTN}} &= \mathrm{DutyCycle} \cdot \mathrm{FrameLength} \\ &-Z_{\mathrm{PKT}} \left(P_{ii} \frac{P_i^{\mathrm{TX}} + P_i^{\mathrm{fault}}}{P_i^{\mathrm{TX}}} + \frac{2P_i^{\mathrm{TX}} + P_i^{\mathrm{fault}}}{P_i^{\mathrm{TX}}} \sum_{j \in N_i^{\mathrm{int}}} P_j^{\mathrm{TX}} \right). \end{aligned} \tag{7}$$

The specific algorithm of operation of the FTMM depends on the type of errors that are detected. During the fault prevention and fault detection phases, the FTMM algorithm provides monitoring of the network conditions including the parameters of both the channel and the node. During the fault identification phase, the nature of the faults is determined; thus, during the fault recovery phase specific actions are taken, including SNR adjustment, changing nodes' neighbor list, and modifying the packet generation and/or transmission rate. If necessary, the threshold values of the PD parameter are adjusted, which helps mitigate the effects of the existing faults. This is done based on the value of $\mathrm{FT}_{\mathrm{action}}$. If $\mathrm{FT}_{\mathrm{action}}$ is smaller than or equal to its threshold value, the process is referred to as a 'regular process' and no changes are performed; otherwise, the threshold values are adjusted according to Eq. (8):

$$\begin{cases} k_{t1} = \begin{cases} k_{t0} + k_{\mathrm{level}}, & \mathrm{FT}_{\mathrm{action}} > 0, \\ k_{t0} - k_{\mathrm{level}}, & \mathrm{FT}_{\mathrm{action}} < 0, \end{cases} \\ P_{ii_{t1}} = \begin{cases} P_{ii_{t0}} + \beta, & \mathrm{FT}_{\mathrm{action}} > 0, \\ P_{ii_{t0}} / \alpha, & \mathrm{FT}_{\mathrm{action}} < 0. \end{cases} \end{cases} \tag{8}$$

Here β is defined as the transmission rate throttle factor, and α is one tenth of the initial value of the generated packet rate (Akyildiz *et al.*, 2006). When network conditions return to normal, the FTMM module readjusts the threshold values for better network efficiency. The specific algorithm of the FTMM operation will be explained in more detail in the following subsections.

3.1 Fault prevention

Because the most critical parameter in WSNs is the nodes' energy, the remaining energy level of the nodes provides very important early indication for possible errors. If a node's remaining energy is critically reduced, then the node cannot successfully transmit its own packets or relay its neighbors'. During communication, intermediate nodes in the network relay neighbors' packets. In case of congestion the sending/receiving buffer of a node fills up to a critical level. Therefore, the buffer occupancy is taken as a second indicator that gives information about network conditions. If the buffer of the intermediate node is critically full, then continuing to participate in the communication with its neighbors might increase only packet loss and latency, diminishing the general performance of the network.

Thus, in our proposed solution, a node monitoring its remaining energy and buffer space can decline to participate in or withdraw from the communication process based on the value of the PD parameter. This solution helps reduce the possibility of error occurrence and untimely interruption of network operation. Furthermore, thresholds for these values are introduced, allowing the node to be withdrawn from the communication process or to limit the node's participation in the communication process. Limited participation is defined as the case in which the buffer level of a node is above a critical value and the remaining energy is still below but very close to a critical level. In such a situation the node will reduce the number of transmitted packets, by giving priority to the ones it generates and declining to relay other nodes' packets. By the introduction of the above mentioned thresholds, the withdrawal of a node from the communication process is achieved in a graceful manner without abruptly interrupting the network operation.

3.2 Fault detection

For fault detection it is very important to monitor the state of the network. In the suggested system this is done in an indirect distributed manner. None of the nodes have a complete picture of the whole network. However, based on continuous monitoring of its own node parameters (buffer level, remaining energy as well as that of the direct neighbors, and the retransmission count, i.e., reTX) and the channel parameters (SNR), each node can create quite a useful picture of its close neighborhood. When a node comes out of sleep state it broadcasts a short 'state message' that is used to refresh the information in the so-called "neighbors' lists" (its one-hop neighbors). It also carries the current energy value for that specific node. Reduced buffer levels as well as indications of low SNR are an early warning for the possibility of errors. Furthermore, increased packet loss and latency present a confirmation. Reduced SNR of the received packet is interpreted as an indicator of faults originating due to worsening channel conditions and possibilities for reduced network capacity. Thus, SNR is also monitored carefully. When the information about the channel parameters and node parameters is handled together, it is possible to provide an early warning to congestion or reduced performance in the network (Fig. 2).

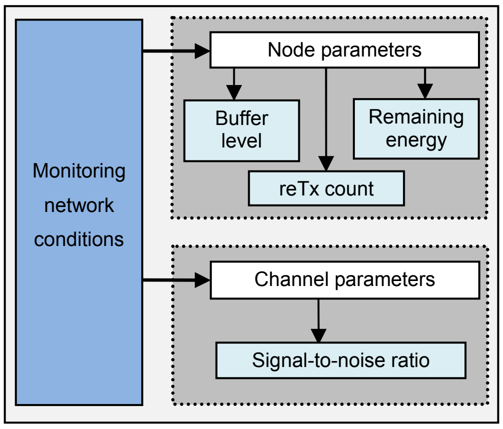

Fig. 2 The proposed fault detection scheme

3.3 Fault identification

The distributed operation of the suggested fault management system, where each node evaluates the situation on its own, supports scalability and allows its application to dense networks with a large number of nodes. During the fault identification phase each node tries to make an independent decision about the source of the fault. Depending on the reason for the fault the following three major cases have been defined: channel fault, node fault, and congestion. Fig. 3 presents the main algorithm that runs on each node.

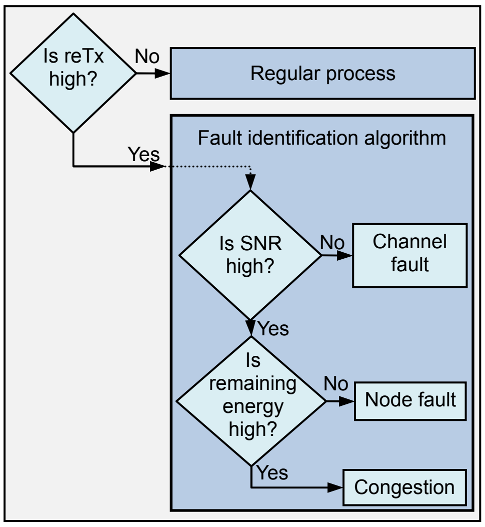

Fig. 3 The proposed fault identification algorithm

Channel faults occur when the SNR of the communication between two nodes worsens. The retransmission number increases (and also the buffer occupancy level increases) and low SNR leads to increased loss of packets and increased latency. Running the algorithm for fault identification given in Fig. 3 and based on the above mentioned information, each node can determine that a channel fault has occurred.

Node fault is defined as the case for which the SNR of the channel is high but there are disturbances in the packet flow between the node and a specific immediate neighbor, usually indicated by increased packet loss and delay.

If the channel quality is within limits, the remaining energy is high but the retransmission rate and the latency are dangerously above expected limits, then faults might be identified as due to congestion.

3.4 Fault recovery

Fault recovery is the last sub-module in our proposed fault management framework. To try recovery procedures, each node should have defined the reason for the fault. Unless the reason of faults is found, the node cannot take precautions, which will eliminate or reduce the impact of the fault. In the logical chain, the fault identification module is a preceding one and as an output provides information as to the possible causes of faults. Based on this input the fault recovery module performs the solutions (Fig. 4).

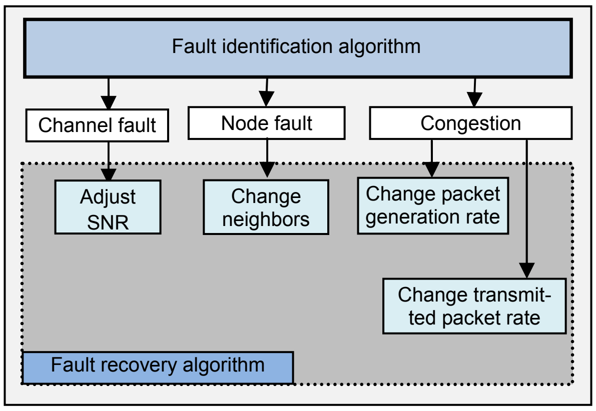
Fault identification algorithm
Channel fault
Node fault
Congestion
Adjust SNR
Change neighbors
Change packet generation rate
Change transmitted packet rate
Fault recovery algorithm

Fig. 4 The proposed fault recovery algorithm

As each node decides on its own whether or not to participate in the communication process based on the values of the PD parameters, it is possible to adjust the SNR threshold to reflect the worsening channel conditions. Adjusting the SNR threshold allows for continuation of the packet flow, thus reducing the packet loss and increasing the number of retransmissions until the channel quality returns to normal.

In the case of identifying a node fault, the fault recovery procedure might prompt a possible change in the immediate neighbors, based on the neighbors' list information.

In the distributed system we propose, it is not possible to fight congestion on a network scale. However, if a number of nodes that have identified congestion in a given area reduce their rate of transmission and retransmission for a certain period of time to a plausible minimum, this would greatly help and in most cases resolve the situation. Decreasing the packet generation rate will result in decreased packet traffic, and thus the network is given a chance to recover to its normal course of operation. However, it is obvious that this recovery mechanism is at the price of increased delay.

4 Fault tolerance management module

The proposed fault management system and specifically the fault tolerance management module (FTMM) have conceptually been developed related to a cross-layer protocol design framework described in detail in Sokullu and Karaca (2009) and Karaca (2010). The communication model involved is distributed and every node makes its own decision about the outcome of fault tolerance management. Due to the decentralized decision and operation mechanism, the size of the network can be increased with very little extra control message traffic. Furthermore, using the CLD approach and considering the functions of all layers of the protocol stack in a unified manner allow for optimization and simplification. The proposed fault management system does not assign different roles or role distribution among the nodes, so additive control packet traffic is kept to a minimum. After defining the major building blocks and procedures in Section 3, in this section we proceed with a more detailed description of the stages of operation of the proposed fault tolerance management module. An overview of the functional structure of the proposed system is given in Fig. 5.

The following stages have been defined: query, decision, and action. The query stage realizes the functions of possible fault prevention and fault identification. During this stage, each node collects information about its own state and the state of the immediate surroundings. Nodes keep track of their remaining energy and buffer levels, and the number of retransmitted packets (reTX count) as an early warning for the presence of errors. The SNR of the incoming packets is used as an indication of the channel state together with information about the remaining energy level of its immediate neighbors. Based on the knowledge of the buffer state and the retransmission (reTx) count each node can estimate the increase in packet loss. Furthermore, time stamping (relayed packets and RTS messages) allows for evaluation of the incurred delay in communicating with the neighboring nodes. So, at the query stage, the node can choose to take fault prevention measures by gracefully withdrawing from further participation in the communication process, and/or can proceed with fault detection based on the acquired information (SNR, packet loss, and delay).

During the decision stage, the cause of the error is investigated, based on the fault identification algorithm described in Section 3.3. The data collected in the query stage is used as an input to the decision stage. In Section 3.3 we define three types of possible errors: errors due to bad channel conditions, errors due to node failures, and errors due to congestion. In accordance with these definitions, the fault identification algorithm is triggered by three major inputs: values of the SNR below a predefined threshold, value of the remaining buffer space below a predefined threshold, and the reTx count.

The reTx count is used in determining whether a transmission error has occurred. The default value for the reTx count in the initial set of node values is set to three. In general, it is expected that with increase in the packet retransmission rate the probability of successfully receiving a packet will also increase, since

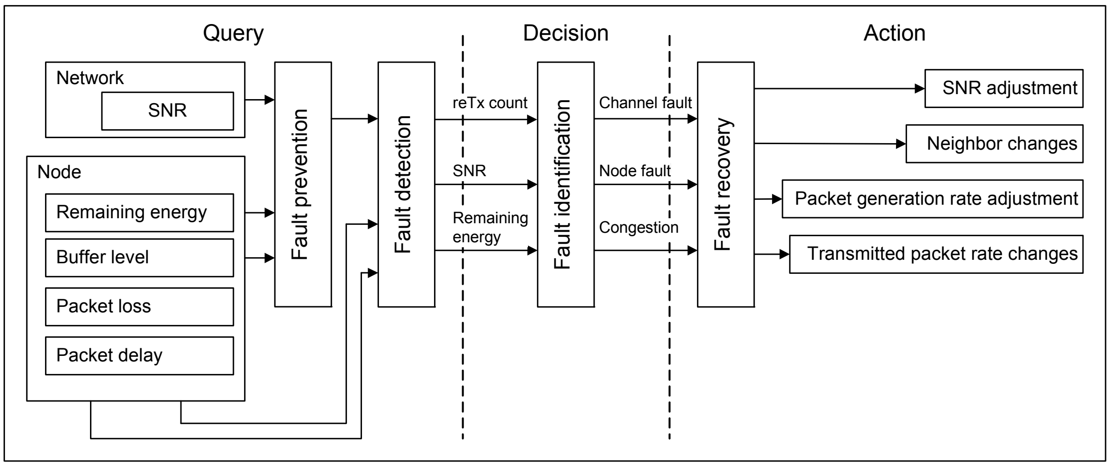

Fig. 5 Functional structure of the proposed fault management system

there is a higher possibility for the packet to be successfully transmitted. According to Bouabdallah *et al.* (2009), however, for a retransmission count greater than three the probability of successfully receiving the packet remains nearly constant. Furthermore, if the unsuccessful transmission is caused by the channel itself, new retransmissions will increase only the channel traffic making the situation worse. Therefore, the possibility of receiving packets at the next node is not affected. For this reason, a reTx count of four is adopted in our system as a trigger to the fault identification procedure.

After deciding which possible error has occurred, the system enters the next stage, called the action stage. During this stage, actions are taken for eliminating or reducing the effect of the errors as described in Section 3.4, limiting the traffic load and thus allowing the network to return to normal operation. At the action stage, the FTMM realizes the following actions: change of transmitting packet rate, adjustment of packet generation rate, change of the neighbor, and change of the SNR level used for packet acceptance at a specific node.

In the communication model we have taken into consideration, each node transmits two types of packets: packets generated by the node itself and packets relayed from other neighbor nodes. So, in case of any indication of increased traffic (congestion), the first action that the fault recovery module takes is reducing the rate of relayed packets, giving priority to its own generated packets. As a result, by responding less to transmit requests from neighbors the node reduces its total transmission rate while keeping the transmit rate for its own packets constant. At first this might seem as a breakup in the transmission of other nodes' packets. However, this does not occur as each node detects a faulty neighbor and acts accordingly as described below. If the signs persist the node will reduce its packet generation level.

If a faulty neighbor node is detected, the node itself will search in its neighbors' list for an exchange.

Finally, at the action stage the node can also act to adjust the SNR as explained in Section 3.4.

The operation of the FTMM described here relies on normal packet exchange between the nodes and the introduction of very short 'state report' control message. These packets are sent when a node wakes up or can be requested using the poll option. Their length is four bytes compared to the 32 bytes for regular data packets. Thus, as a whole the proposed fault management scheme takes advantage of the available information and the CLD introducing a minimal control overhead.

5 Comparative simulation results

Adopting the idea of 'participation determination' suggested in XLM (Akyildiz *et al.*, 2006) and the CLD approach as discussed in Sokullu and Karaca (2009) and Karaca *et al.* (2012), we have extended the concept with a number of additional features to incorporate the fault management system FTMM as explained in Sections 3 and 4. The CLD approach provides great advantages regarding optimization possibilities as previously discussed in Sokullu and Karaca (2009), Karaca (2010), and Karaca *et al.* (2012). As the main contribution of the work presented in this paper is the introduction of an integrated platform for fault management, using the advantages of CLD, for evaluating its effects on the performance of the whole network, a comparison with a similar, CLD based simulation platform, XLM, is provided.

Simulations are carried out with 300 nodes, randomly distributed in a 100 m×100 m field. One node is selected as the sink. Each simulation runs for 60 s. Ten simulations are performed with different node deployments and the average values are plotted.

In this section we present simulation results comparing the performance between XLM and FTMM in terms of packet loss, delay, and power consumption.

Packet loss is investigated based on the goodput and the retransmission losses. The goodput is defined as the ratio of the number of packets successfully received at the sink to the number of packets transmitted by the node that collected the data. The packet loss arising in relation to retransmission is measured by the total number of packets that have been dropped by a given node after the maximum number of retransmission attempts has been achieved and the packet could not be successfully transmitted. The average latency is evaluated as the average of the total delay the packets experience at each node averaged over the number of packets transmitted. The power consumption is evaluated based on the power required to transmit a single packet. Additionally, the average number of hops a packet travels from the source node to the sink is also studied.

These parameters have been investigated as a function of the duty cycle where the duty cycle is defined as the ratio of the time during which the node is active (can receive and transmit packets) to the time during which the node is asleep. In the simulation the duty cycle is varied from 0.1 to 1, where the value of 1 means that the node is active all the time.

Fig. 6 gives the goodput as a function of the duty cycle for the two systems simulated: the XLM and the FTMM.

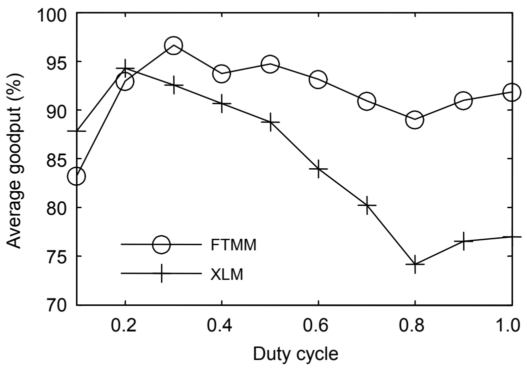

Fig. 6 Goodput versus duty cycle

It can be seen that for the FTMM the goodput ratio is low for a duty cycle of 0.1 and gets better with the increased duty cycle value. In XLM, goodput is high for the duty cycle values between 0.2 and 0.5, but it gets worse and falls quite sharply for longer active time. Here the value of 0.5 means the node is awake for half of the cycle. On the other hand the FTMM goodput is quite stable in the whole range, and stays much higher than the XLM goodput for values of the duty cycle above 0.8, which can be viewed as a result of the introduction of fault management. For duty cycle values up to 0.2, FTMM demonstrates worse performance due to the fact that there are quite a small number of nodes awake and thus some of the proposed measures cannot be carried out successfully.

A packet is retransmitted if the node has failed to send it successfully in the first attempt. It is typical for WSNs to limit the number of allowed retransmissions (Vuran *et al.*, 2005; Akyildiz *et al.*, 2006; Gupta *et al.*, 2009). Retransmission can be tried for up to seven times for each packet, after which the packet is dropped. The total retransmission losses for the network are calculated using the total number of dropped packets. FTMM and XLM have the same retransmission loss levels at the lowest duty cycle rate (Fig. 7), but considering the whole range FTMM has better performance than XLM. These losses show an increasing tendency in the FTMM system as well.

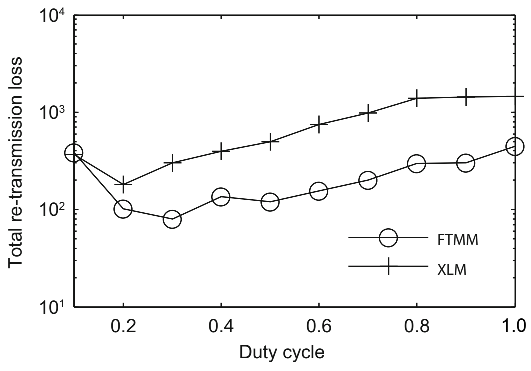

Fig. 7 Retransmission loss versus duty cycle

Another important parameter that has been evaluated in the performance comparison is the average delay experienced in the network. The difference between the time when the sink receives a packet and the time when the packet was sent is defined as transmission delay. Thus, the average delay for all packets is calculated as the total transmission time for all packets divided by the number of packets received by the sink. As shown in Fig. 8, the general trend is reduction in the delay with the increase of the duty cycle; however, for larger values of the duty cycle the average delay is increased.

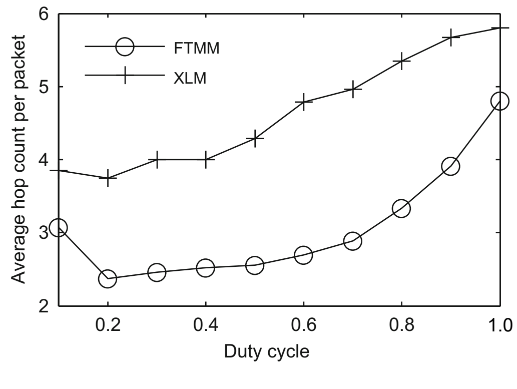

Fig. 8 Hop count versus duty cycle

Similar behavior is observed for both systems. The delay for the FTMM is consistently greater than the one for the XLM system. This can be explained by the fact that when a node wakes up it sends a 'state report' control message. The time required to receive and process these messages is the reason for the increased delay in the FTMM. Furthermore, when nodes sleep for a longer period (low duty cycle) the delay is greater, which is also an expected result. It is also interesting to note that the delay is nearly constant for the range 0.5–0.7 while for a duty cycle above 0.7, for both the FTMM and the XLM the delay is notably increased. This case can be justified taking into consideration factors like increased channel traffic (more retransmissions required) and increased number of neighbors, i.e., increased number of average traversed hops.

Comparing the available energy of the nodes in the beginning of every simulation run with that at the end provides a measure for the average energy used. If it is divided by the total number of packets received by the sink node, then the average energy per successful packet transmission is found. Fig. 8 shows that, overall, FTMM requires less per packet energy than XLM. For low duty cycles there is a very slight difference in the two systems compared.

For the XLM, the retransmission losses increase for a duty cycle greater than 0.3 (Fig. 9). This is supported by the fact that nodes spend a great amount of their energy for transmission. The advantage of FTMM is consuming less energy despite the fact that a 'state report' message is used. This can be explained by the fact that the 'state report' message is sent only when a node wakes up and also that being a control message it is very short. It is much shorter compared to a data packet and by sending it the retransmission of a much longer data packet is avoided.

Another performance metric that has been studied is the number of hops required on the average for a packet to reach from the data collecting node to the sink. The investigation of this metric for the XLM and FTMM provides some further insights into their operation and stresses the advantages of the suggested FTMM (Fig. 10).

As a packet is sent from the generating node, its hop count is increased at every relay node it passes. The number of hops for all packets reaching the sink is defined as the total number of hops transversed. If this is divided by the number of packets that have arrived at the sink, then the average number of hops per packet can be defined. In general, this parameter is increased when the duty cycle is increased and the observation is true for both the XLM and the FTMM. However, notice that generally the FTMM, due to its preventive and recovery actions, manages to deliver the same number of packets with fewer hops. The difference between the two systems is around two hops. The FTMM sends packets that are at a distance of one hop from the sink always with only one hop, while this is not always true for the XLM system. Also, note that in the FTMM the minimum number of hops is observed for a duty cycle of 0.2 to 0.5. The general increasing trend for duty cycles greater than 0.5 in both cases can be explained with the fact that when there are more nodes awake the number of available neighbors, ready to participate in the communication process, is increased.

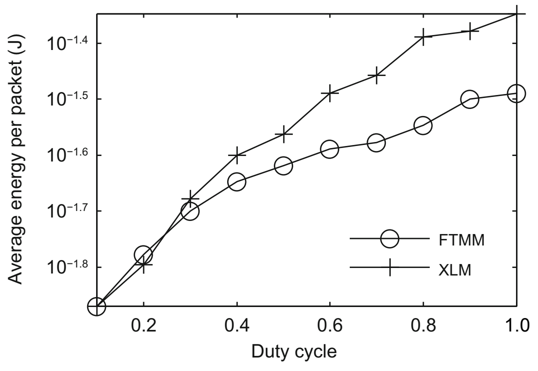

Fig. 9 Energy versus duty cycle

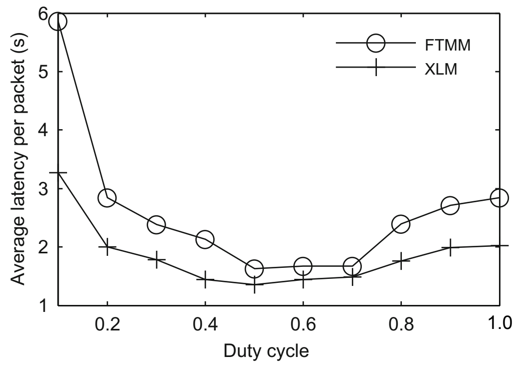

Fig. 10 Latency versus duty cycle

6 Conclusions

In this paper, we present the details of a fault tolerance management module for WSNs, realized using a cross-layer design approach. The suggested

FTMM relies on distributed operation. The performance of the network, based on packet loss, delay, and energy consumption, is compared through simulation with another cross-layer design solution not including fault management, XLM. The achieved simulation results support the idea that the introduction of a unified approach to fault management improves the network performance in terms of packet loss, delay, and power consumption.

When considering the results of the comparison it has been observed that for different performance criteria including goodput, latency, and consumed energy (Figs. 6, 10, and 9, respectively), the duty cycle of 0.3 provides optimum performance for the FTMM. At that point the goodput achieves a maximum value, while the packet loss is a minimum. The average number of hops is at a minimum between 0.2 and 0.5, and delay is minimum in the interval 0.5–0.7, while the minimum consumed energy, observed at a duty cycle of 0.1, is only slightly increased for the above mentioned range (around 0.3). A general conclusion can be made, that for the type of network considered in this work, if nodes are active for about 30% of the transmission cycle, this provides an optimal case considering goodput, packet loss, and energy consumption.

The comparative simulation results between XLM and FTMM prove the superior performance provided by the development of the fault management module. The proposed system uses less energy and ensures transmission of packets with lower packet loss, which means that it can support a prolonged network lifetime. The longer delay compared to XLM is a tradeoff imposed by the introduction of fault prevention, fault detection, fault identification, and some recovery procedures. Furthermore, as this delay is not abnormally high it can be evaluated as a reasonable price to pay for the improved performance.

References

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. A survey on sensor networks. *IEEE Commun. Mag.*, **40**(8):102-114. [doi:10.1109/MCOM.2002.1024422]

Akyildiz, I.F., Vuran, M.C., Akan, O.B., 2006. A Cross-Layer Protocol for Wireless Sensor Networks. 40th Annual Conf. on Information Sciences and Systems, p.1102-1107. [doi:10.1109/CISS.2006.286630]

Bouabdallah, F., Bouabdallah, N., Boutaba, R., 2009. Cross-Layer Design for Energy Conservation in Wireless Sensor Networks. Proc. IEEE Int. Conf. on Communications, p.1-6. [doi:10.1109/ICC.2009.5198872]

Drabkin, V., Friedman, R., Kliot, G., 2008. On the (un)reliability of TCP connections: the return of the end-to-end argument. *IEEE Distr. Syst. Online*, **9**(8):2. [doi:10.1109/MDSO.2008.22]

Ergen, S.C., Pravin, V., 2007. Energy efficient routing with delay guarantee for sensor networks. *Wirel. Networks*, **13**(5):679-690. [doi:10.1007/s11276-006-8149-y]

Gungor, V.C., Vuran, M.C., Akan, O.B., 2007. On the cross-layer interactions between congestion and contention in wireless sensor and actor networks. *Ad Hoc Networks*, **5**(6):897-909. [doi:10.1016/j.adhoc.2007.02.007]

Gupta, S., Vuran, M.C., Gursoy, M.C., 2009. Power Efficiency of Cooperative Communication in Wireless Sensor Networks. 3rd Int. Conf. on Signal Processing and Communication Systems, p.1-10. [doi:10.1109/ICSPCS.2009.5306417]

Han, Q., Lazaridis, I., Mehrotra, S., Venkatasubramanian, N., 2005. Sensor Data Collection with Expected Reliability Guarantees. 3rd IEEE Int. Conf. on Pervasive Computing and Communications Workshops, p.374-378. [doi:10.1109/PERCOMW.2005.73]

Karaca, L.O., 2010. Designing a Fault Tolerant Cross-Layer Framework in Wireless Sensor Networks. PhD Thesis, Ege University, Izmir, Turkey (in Turkish).

Karaca, L.O., Sokullu, R., Prasad, N.R., Prasad, R., 2012. Application oriented multi criteria optimization in WSNs using on AHP. *Wirel. Pers. Commun.*, **65**(3):689-712. [doi:10.1007/s11277-011-0280-0]

Khan, N.M., Ihsan, A., Khalid, Z., Ahmed, G., Ramer, R., Kavokin, A.A., 2008. Quasi Centralized Clustering Approach for an Energy-Efficient and Vulnerability-Aware Routing in Wireless Sensor Networks. Proc. 1st ACM Int. Workshop on Heterogeneous Sensor and Actor Networks, p.67-72. [doi:10.1145/1374699.1374712]

Mini, R., Loureiro, A., Badrinath, B.R., 2004. The distinctive design characteristic of a wireless sensor network: the energy map. *Comput. Commun.*, **27**(10):935-945. [doi:10.1016/j.comcom.2004.01.004]

Paradis, L., Han, Q., 2007. A survey of fault management in wireless sensor networks. *J. Network Syst. Manag.*, **15**(2):171-190. [doi:10.1007/s10922-007-9062-0]

Park, S.J., Vedantham, R., Sivakumar, R., Akyildiz, I.F., 2004. A Scalable Approach for Reliable Downstream Data Delivery in Wireless Sensor Networks. Proc. 5th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing, p.78-89. [doi:10.1145/989459.989470]

Ramanathan, N., Chang, K., Kapur, R., Girod, L., Kohler, E., Estrin, D., 2005. Sympathy for the Sensor Network Debugger. Proc. 3rd Int. Conf. on Embedded Networked Sensor Systems, p.255-267. [doi:10.1145/1098918.1098946]

Sankarasubramaniam, Y., Akan, O.B., Akyildiz, I.F., 2003. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. Proc. 4th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing, p.177-188. [doi:10.

1145/778415.778437]

Shakkottai, S., Rappaport, T.S., Karlsson, P.C., 2003. Cross-layer design for wireless networks. *IEEE Commun. Mag.*, **41**(10):74-80. [doi:10.1109/MCOM.2003.1235598]

Sokullu, R., Karaca, O., 2009. Simple and Efficient Cross-Layer Framework Concept for Wireless Sensor Networks. 12th Int. Symp. on Wireless Personal Multimedia Communications, S05-1.

Srivastava, V., Motani, M., 2005. Cross-layer design: a survey and the road ahead. *IEEE Commun. Mag.*, **43**(12):112-119. [doi:10.1109/MCOM.2005.1561928]

Staddon, J., Balfanz, D., Durfee, G., 2002. Efficient Tracing of Failed Nodes in Sensor Networks. Proc. 1st ACM Int. Workshop on Wireless Sensor Networks and Applications, p.122-130. [doi:10.1145/570738.570756]

Stewart, R., Metz, C., 2001. SCTP: new transport protocol for TCP/IP. *IEEE Internet Comput.*, **5**(6):64-69. [doi:10.1109/4236.968833]

Vidhyapriya, R., Vanathi, P.T., 2008. Reliable energy-efficient routing with novel route update in wireless sensor networks. *J. Zhejiang Univ.-Sci. A*, **9**(8):1099-1110. [doi:10.1631/jzus.A072260]

Vuran, M.C., Gungor, V.C., Akan, O.B., 2005. On the Interdependence of Congestion and Contention in Wireless Sensor Networks. Proc. ICST SenMetrics, p.136-147.

Wan, C.Y., Eisenman, S.B., Campbell, A.T., 2003. Coda: Congestion Detection and Avoidance in Sensor Networks. Proc. 1st Int. Conf. on Embedded Networked Sensor Systems, p.266-279. [doi:10.1145/958491.958523]

Wan, C.Y., Campbell, A.T., Krishnamurthy, L., 2005. Pump slowly fetch quickly (PSFQ): a reliable transport protocol for wireless sensor networks. *IEEE J. Sel. Areas Commun.*, **23**(4):862-872. [doi:10.1109/JSAC.2005.843554]

Zhao, Y.J., Govindan, R., Estrin, D., 2002. Residual Energy Scan for Monitoring Sensor Networks. IEEE Wireless Communications and Networking Conf., p.17-21.