Computing, March 2015, Vol. 97, Issue 3, pp 205-236

LINKORD: Link Ordering-Based Data Gathering Protocol for Wireless Sensor Networks

Marjan Radi[1,*], Behnam Dezfouli[1], K. A. Bakar[1], S. A. Razak[1], Malrey Lee[2]

[1] *Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia*
[2] *School of Electronics and Information Engineering, ChonBuk National University, Korea*

Abstract

With respect to the multi-hop communication pattern of wireless sensor networks, all the nodes should establish multi-hop paths towards a common data gathering point to provide a data gathering service for the underlying applications. Although data gathering protocols provide a simple service, these protocols suffer from poor performance in practice due to the power constraints of low-power sensor nodes and unreliability of wireless links. Existing data gathering protocols rely on the ETX metric to find high-throughput paths through assuming there is an infinite number of transmission attempts at the link layer for delivering a single packet over every link. However, in practice the link layer provides a bounded number of transmissions per packet over individual links. Therefore, employing existing data gathering protocols in these situations may result in the construction of the paths that require more than maximum number of provided link layer transmissions for delivering a single packet over each link. In this regard, we propose a path cost function which considers the limitation on the number of provided link layer transmissions and relative position of the links along the paths according to their data transmission probability. Furthermore, we introduce a data gathering protocol which uses the proposed path cost function to construct high-throughput paths. Moreover, this protocol employs a newly designed congestion control mechanism during the data transmission process to provide energy-efficient and high-throughput data delivery. The simulation results show that, the proposed protocol improves data delivery ratio by 70% and network goodput by 80%, while it reduces the consumed energy for data delivery by 50% compared to the default data gathering protocol of TinyOS.

Keywords: Wireless Sensor Networks, Data Gathering, Link Ordering, Link Quality, Energy-Efficiency

1. Introduction

Since wireless sensor networks offer new opportunities to observe the physical environments and interact with inaccessible areas (e.g., jungles, earthquake disaster area, battlefield target area, or inside a nuclear reactor), these networks are being used in a wide range of applications [1, 2]. The main observable traffic pattern in wireless sensor networks is many-to-one, in which data flows from many sensor nodes towards a single base station [3–6]. Due to the limited radio range of the sensor nodes, intermediate nodes require to perform data relaying to forward the collected data from source nodes towards the sink node [7–9]. Therefore data gathering service is one of the key building blocks of different wireless sensor networks protocols to support multi-hop data transmission pattern [10]. The main aim of data gathering protocols is to construct the network routing topology, which allows the collected data from source nodes to be delivered to the network data gathering point [10, 11].

Due to the importance of providing efficient data gathering capability in wireless sensor networks, several data gathering protocols have been developed over the past decade. The utilized path cost function is the main factor that differentiates these protocols [12]. Experimental studies on the performance of the existing data gathering protocols show that, while the existing data gathering protocols provide a basic service for many applications, they suffer from poor performance in practice (in terms of reliable and energy-efficient data delivery) due to the high dynamics of low-power wireless links [13–15]. According to these studies, the required number of link layer transmissions (including retransmissions) for successful packet delivery from a given source node to the sink over unreliable links influences the network-wide contentions, throughput and network energy consumption [16, 17]. Therefore, reducing the required number of link layer transmissions is critical to provide efficient data gathering in wireless sensor networks [18, 19]. In this context, the Expected Transmission count metric (ETX), is widely utilized by the existing data gathering protocols to provide efficient and reliable data gathering in wireless sensor networks [12]. The ETX value of a given link is defined as $\frac{1}{p \times q}$, where p and q are

*Corresponding author:
Email address: radi@ieee.org (Marjan Radi)

DOI: http://link.springer.com/article/10.1007/s00607-014-0414-9

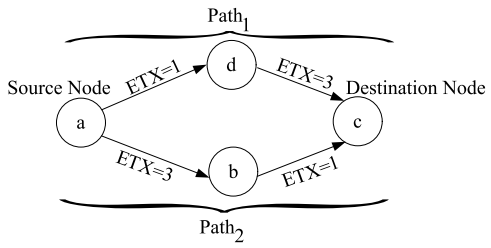

Figure 1: The influence of link positions on the data transmission cost of paths.

probabilities that a packet is successfully transmitted and acknowledged respectively [20]. Based on the definition of this metric, it assumes the link layer provides an infinite number of transmissions at each link to deliver every transmitted data packet to the destination [12, 21]. Since by this assumption the link layer never drops a packet, the ETX value of each path only depends on the ETX of its constructed links, while the location of the links on the path is not important. Therefore, this unrealistic assumption allows a commutative path cost computation, which simplifies the calculation of the expected number of transmissions over each path for successful packet delivery. However, these protocols may construct paths including links which cause a higher number of transmissions than the maximum number of provided link layer transmissions. Since in practice the link layer provides a finite number of transmission attempts at each link, existing data gathering protocols expose low performance in the real-world implementations. The main reason is that, packet drops due to the limited number of transmissions at the link layer on the links near the destination are very costly, as the packets have traversed several links before they are dropped. Figure 1 presents this issue through a simple example. Based on this figure the ETX summation of the links along both paths are equal to 4. Assume the link layer provides at most 2 transmission attempts at each link. Consequently, the probability that a packet passes the link with an ETX value equal to 3 is $\frac{1}{3} + (1 - \frac{1}{3}) \times \frac{1}{3} = 0.55$. Therefore, the probability of packet drop due to the limited number of link layer transmissions on the last link of $path_1$ is higher than $path_2$. Since packet drops near the destination are very costly, the data transmission cost of $path_1$ is higher than $path_2$. However, simple summation of the link ETX values along the paths does not capture this issue. Expected number of Transmissions On a Path (ETOP) is a recently proposed path cost function as a part of Dynamic Source Routing (DSR) protocol for wireless mesh networks, which estimates the data transmission cost of individual paths with respect to the limited number of offered link layer transmissions at each link [22]. Since, through this cost function each source node should determine the entire path towards the destination node by a Dijkstra algorithm, it cannot be used in wireless sensor networks. The reason is that, in wireless sensor networks data gathering tree construction process should be initiated by the sink node and data transmission cost of every path from a given sensor node towards the sink should be estimated based on the calculated cost in the reverse direction (from sink towards the sensor nodes) [23]. However, designing a distributed data gathering protocols which considers the order of the links along the paths from sensor nodes towards the sink is a non-trivial task.

From the discussion given above, it is obvious that existing data gathering protocols cannot provide an efficient data gathering service when the link layer provides a finite number of transmission attempts at individual links. The reason is that packet drops due to the bounded number of link layer transmissions on the links close to the sink node impose a significant packet delivery cost to the network. Therefore, in order to construct a minimum cost data gathering tree in term of the required number of packet transmissions for every single packet delivery, it is required to develop data gathering protocols which consider the following issues: (1) the maximum number of offered link layer transmission attempts for delivering a single packet over each link, (2) relative position of the links along the paths, and (3) packet delivery probability of individual links. In this regard the main contributions of this paper are:

Firstly, we propose a path cost function which considers the limitation on the number of provided link layer transmission attempts at individual links, relative position of the links along the paths and packet delivery probability of every link in order to construct the paths which minimize the required number of transmissions for a successful packet delivery. The proposed path cost function, called Successful or Failed packet Transmission Cost (SFTC), enables every node to estimate data transmission cost through its neighboring nodes with respect to the relative position of the links from that particular node towards the sink based on the calculated cost in the reverse direction (from sink towards the sensor nodes).

Secondly, we propose a LINK ORDering based data gathering protocol (LINKORD) which is composed of two phases: 1) Data gathering tree construction phase, and 2) Data transmission phase. The aim of this protocol is to provide a high performance data gathering in terms of data delivery ratio, network goodput, and energy consumption. During the data gathering tree construction phase every node identifies cost of data transmission towards the sink node through its neighboring nodes using the proposed SFTC path cost function. Therefore, at the end of this phase each node has identified several paths through its neighboring nodes towards the sink in the constructed network data gathering tree. The data transmission phase takes care of data forwarding from source nodes towards the sink during network operation. Furthermore, this phase also handles network congestion and path failures during the data transmission process. To this end, a Congestion Control with Parent Change capability

(CCPC) approach is developed which enables every child node to change its congested parent node in order to reduce the network congestion level.

Thirdly, we implement the proposed data gathering protocol and evaluate its performance compared to the TinyOS's Collection Tree Protocol (CTP) under two different scenarios. Comprehensive simulation studies confirm that LINKORD improves data delivery ratio by 70%, end-to-end network goodput by 80%, and energy consumption by 50% compared to the TinyOS's CTP.

The rest of this paper is organized as follows: Section **??** provides an overview on the existing data gathering protocols in wireless sensor networks. We introduce the proposed SFTC path cost function and LINKORD in Section 3 and Section 4 respectively. Performance evaluation and comparison of LINKORD against TinyOS's CTP is performed in Section 5. We conclude in Section 6.

2. Related Work

Due to the importance of sensor data gathering, there has been a huge amount of research over the past decade to provide efficient data gathering in wireless sensor networks [12, 21, 23]. The major difference of these protocols is in their employed cost functions for parent selection during the network data gathering tree construction process. Based on the empirical studies on the characterization of wireless links, data transmission quality of wireless links fluctuates highly over time due to the effects of wireless propagation such as multipath fading, shadowing, and path loss [24–26]. These studies suggest that employing link quality-based routing metrics can provide stable network performance by selecting high-quality links which reduce the number of packet losses and retransmissions [16, 27–29]. Therefore, the majority of existing data gathering protocols utilize link-quality based routing metrics to construct a stable tree routing structure which includes high-quality paths.

MintRoute was the first proposed data gathering protocol for wireless sensor networks. This protocol uses a distance vector routing approach to advertise global routing state [13]. Furthermore, it uses the ETX [20] as the routing cost metric to compute the data transmission cost of available paths. To this aim, MintRoute utilizes routing tables to preserve the neighborhood information at individual nodes along with their respective data transmission cost towards the sink node. In MintRoute, the entire nodes periodically broadcast routing messages to share their achieved routing information towards the sink node and update the preserved routing information at individual nodes.

MultihopLQI is a variation of *MintRoute* which uses a hardware-based link quality estimation metric for computing the data transmission cost of different paths [30]. Unlike MintRoute, MultihopLQI uses the link quality indicator (LQI) metric as its routing cost metric which is provided by 802.15.4 RF transceivers [27]. This protocol avoids keeping routing tables through maintaining the state of the best parent at a given time. Although LQI can provide immediate approximations regarding to the bit error rate of the successfully received packets, this information can be only provided by the CC2420 radio chip [31]. This limitation causes the MultiHopLQI protocol to be hardware dependent. Moreover, as this approach estimates the quality of links based on the successfully received packets, it cannot differentiate between packet losses due to the poor link quality or those due to the collisions [32, 33]. These inaccurate link quality estimations result in frequent network tree reconstruction and unstable routing topology.

The Collection Tree Protocol (CTP) is the default data gathering protocol of TinyOS [21]. The main aim of this protocol is to benefit from integration of the data and control plans in order to provide quick response against routing failures during the data transmission process and repair the broken paths with a minimum interruption in the data gathering performance. As with MintRoute, CTP also uses a distance vector routing algorithm with ETX cost metric to establish the paths that minimize the expected number of transmissions for data delivery. In addition, this protocol uses the network active traffic to probe the topology and detect routing problems, while it also employs the Trickle algorithm [34] to control the beaconing exchange rate of the network nodes. In this protocol the routing framework consists of three main components: Routing Engine (RE) that determines the best next-hop neighboring node towards the sink node as well as updating the routing tables; the Forwarding Engine (FE), that takes care of forwarding data packets to the determined nodes by the RE; and the Link Estimator (LE) that periodically determines the data transmission quality of available links using the four-bit link quality estimation approach [35]. Furthermore, TinyOS' CTP is capable of detecting network congestion during the data transmission process through setting the *congestion flag* of the data packets [12]. In this context, whenever the packet buffer of a node is at least half full, it sets the congestion flag of its outgoing packets to 1 in order to notify the child nodes about its congestion status. If a node identifies its parent is congested, it stops its packet transmission until the congestion status of its parent changes or it finds an alternative path towards the sink. In this protocol a node will change its current congested parent to another parent whose multi-hop ETX value is lower than the other available neighbors.

Hyper is a data gathering protocol which enables mobile users to access the sensor network data during their environmental evaluations or maintenance work. This protocol uses a distance vector routing strategy for relaying data packets to the mobile sinks [36]. To support a low-latency access to the sensor data by mobile users, Hyper provides fast neighborhood assessment and multiple data gathering trees for simultaneous utilization by several users. Since

Hyper uses ETX as the routing cost metric to construct reliable routing trees, it can initialize the tree construction process whenever the neighborhood connectivity is evaluated. Consequently, each node periodically exchanges beacon messages with its neighboring nodes to obtain the required link quality information for tree construction. Since the root of each constructed tree is a mobile sink, whenever a mobile sink arrives to a new location it should initiates the link estimation process for evaluating its connections towards its new neighbors by broadcasting a series of beacon messages. When each node evaluates its neighborhood connectivity, sink node initiates the routing tree construction process by flooding a *tree update* message into the network. Since mobile users are practically delay sensitive and the neighborhood connectivity assessment should be performed in a short period by the network nodes, this protocol cannot provide an efficient data gathering service in large-scale dense wireless sensor networks. This is due to the fact that fast neighborhood assessment in high-density networks intensifies contentions for channel access and causes a huge number of packet losses which in turn reduces the accuracy of the collected neighborhood information by individual nodes.

CentRoute is another data gathering protocol which combines a source routing approach with a centralized route computation mechanism (in a microserver) to rescue the nodes from making routing decisions [37]. As with several other routing protocols, CentRoute uses ETX cost metric for parent selection. The main idea behind designing this routing framework is that the resource limitations of sensor nodes influence the correctness and accuracy of the decision which is made at each mote based on its limited local information. Therefore, this protocol aims to employ microservers to make routing decisions based on the combination of the achieved information from multiple motes. In order to construct a tree topology which is rooted at a microserver, each node periodically broadcasts a *join request* message. If a node that is already joined to the tree receives this message, it forwards the message to a microserver via its parent. Then upon reception of this message at a microserver, it generates a *join reply* message and forwards this message towards the sender node using source routing.

Dozer is another data gathering framework which aims to fulfill the performance demands of periodic data gathering in monitoring applications [38]. This protocol integrates a tree routing protocol with a TDMA-based MAC layer to support energy-efficient data delivery by reducing idle listening and overhearing periods. To this aim, Dozer constructs a network routing tree in which the entire communications between all the nodes and their parents towards the sink node are coordinated by a TDMA-based MAC protocol. In order to construct this data gathering framework, whenever a node joins to the network routing tree during the data gathering tree construction process, it broadcasts beacon messages at the start of its time schedule to assign time slots for connection-requests from disconnected nodes and enables them to join to the network tree. Furthermore, each node preserves a list of prospective parents for fast recovery from route failures during the data transmission process.

3. Successful or Failed Packet Transmission Cost Function Design

This section dedicated to describe the proposed path cost function in this paper. In this regard, the considered network and packet transmission models for calculating packet transmission cost of multi-hop paths are given in the first part of this section. Finally, the last part presents the design of the proposed SFTC cost function in detail.

3.1. Network Model and Notations

A wireless sensor network can be considered as a directed graph *G(N, E, P)*, where N is the set of sensor nodes, E is the set of links between nodes, and P is the set of packet delivery probabilities over the links in set E. Let N_i be the set of neighboring nodes of node n_i. Therefore, set E should be represented as $E=\{e_{x,y}|x \in N \; and \; y \in N_x\}$ and each link $e_{x,y} \in E$ has a packet delivery probability $0 < p_{x,y} \leq 1$ with a single transmission effort.

The packet transmission process is modelled as follows. At the start of the data transmission phase, a given source node n_0 starts to transmit its data packets towards the sink node over a path with n hops through node n_1, n_2, ..., n_n. Therefore, node n_0 will pass its packet to the link layer which is responsible for transmitting this packet to node n_1. The transmitted packet by node n_0 will be received by node n_1 with probability $p_{0,1}$ after 1 transmission attempt. If node n_1 does not receive the packet transmitted by node n_0, this packet will be transmitted again by the link layer of node n_0. Link layer of node n_0 repeats this process until the maximum number of link layer transmissions (r transmissions) is reached or the packet is successfully received by node n_1. If after the rth retry the packet dose not reach to the receiver node n_1, the sender node will drop the packet. While, if node n_1 receives the transmitted packet successfully, it will transmit the packet to node n_2. This process will be repeated between all the nodes on the selected path, until node n_n receives the transmitted packet from node n_0 or the packet drops at an intermediate node along the path. Let h_s be the number of successive hops that a packet passed successfully from a source node before it is dropped. Thus, $h_s=i$ means that the transmitted packet from node n_i failed to reach to node n_{i+1}. Therefore, if $t_{x,y}$ signifies the required number of link layer transmissions for a single packet delivery over link $e_{x,y}$, $h_s=i$ denotes that $t_{i,i+1}$ was more than r. Since the link layer offers a bounded number of transmission attempts at individual links, some packets may be dropped during the data transmission process somewhere along the selected paths. By considering this possibility the relative position of the links highly influences the data delivery cost

of different paths. Therefore, the data transmission cost of every path cannot be calculated through a simple summation of the data transmission quality of its constructed links which has been employed in the existing data gathering protocols [12, 21].

Consider the case where the link layer provides an infinite number of transmissions for packet delivery over individual links. By this assumption, every node can calculate the required number of packet transmissions for successful packet delivery to its neighboring nodes as:

$$E[e_{x,y}] = \sum_{i=1}^{\infty} i(1-p_{x,y})^{i-1} p_{x,y} = \frac{1}{p_{x,y}} \quad (1)$$

where $E[e_{x,y}]$ is the expected number of required transmissions for successful packet transmission over link $e_{x,y}$, and $p_{x,y}$ is the probability of successful packet transmission from node n_x towards node n_y. With respect to the Equation 1, the expected number of required packet transmissions for successful packet delivery over a path $(n_0, n_1, ..., n_n)$ can be calculated as:

$$E[e_{0,n}] = \sum_{i=0}^{n-1} \frac{1}{p_{i,i+1}} \quad (2)$$

Now assume that link layer performs at most r transmission attempts over each link for a single packet delivery. In this case the expected number of packet transmission attempts for transmitting a single packet over link $e_{i,i+1}$ is:

$$E[e_{i,i+1}] = (\sum_{k=1}^{r} k(1-p_{i,i+1})^{k-1} p_{i,i+1}) + r(1-p_{i,i+1})^{r} \quad (3)$$

where $(1-p_{i,i+1})^{k-1} p_{i,i+1}$ is the probability that the transmitted packet from node n_i have been successfully delivered to node n_{i+1} at *kth* transmission attempt. Therefore, the expected number of transmissions for successful packet delivery over link $e_{i,i+1}$ after performing k transmission attempts is calculated through multiplying this term by k. Moreover, term $r(1-p_{i,i+1})^{r}$ denotes the expected number of transmission attempts performed by node n_i while node n_{i+1} could not receive the transmitted packet. With respect to Equation 3, the expected number of packet transmissions over a path $(n_0, n_1, ..., n_n)$ in terms of packet delivery probability of the links and the maximum number of link layer transmissions over each link can

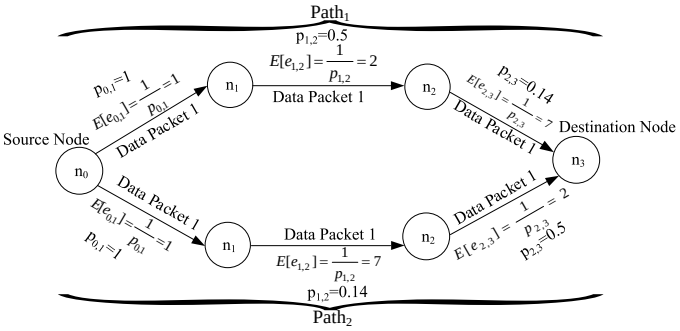

Figure 2: A sample scenario to evaluate data transmission cost over two different paths from a source node towards a given destination node.

be expressed as:

$$E[e_{0,n}] = P_{path}(E_s[e_{0,n}]) + (1-P_{path})(E_f[e_{0,n}])$$

where

$$P_{path} = \prod_{i=0}^{n-1} (1-(1-p_{i,i+1})^{r})$$

$$E_s[e_{0,n}] = \sum_{i=0}^{n-1} E[e_{i,i+1}]$$

$$E_f[e_{0,n}] = \sum_{i=0}^{n-1} \left[(1-p_{i,i+1})^{r} (\prod_{j=0}^{i-1} (1-(1-p_{j,j+1})^{r})) (\sum_{l=0}^{i} E[e_{l,l+1}]) \right] \quad (4)$$

where $\prod_{j=0}^{-1}$=1, P_{path} signifies the probability of successful reception of the transmitted packet by source node n_0 at the sink node (i.e., node n_n), *1-*P_{path} is the probability of packet drop over a path from source node n_0 towards the sink node, $E_s[e_{0,n}]$ is the expected number of packet transmissions for packet delivery over path $n_0, ..., n_n$, and $E_f[e_{0,n}]$ is the expected number of packet transmissions over path $n_0, ..., n_n$, while the packet fails to reach the sink node. Furthermore, term $(1-(1-p_{i,i+1})^{r})$ indicates the probability of successfully packet delivery over link $e_{i,i+1}$ and term $(1-p_{i,i+1})^{r}$ represents the probability that a packet fails to pass the link $e_{i,i+1}$ after performing r transmission attempts.

Figure 2 illustrates two different paths from node n_0 towards node n_3 to clarify the differences between data transmission cost of multi-hop paths when the link layer provides a limited or an infinite number of transmission attempts. In the case that the link layer provides an infinite number of packet transmissions for a successful packet delivery, the data transmission cost of both paths is equal to 10 (according to Equation 2). Consider the scenario where the link layer provides at most 2 transmission attempts at each link (i.e., r=2). First assume node n_0 wants to transmit a packet to node n_3 through path$_1$. The probability that the transmitted data packet from node n_0 passes link $e_{0,1}$, link $e_{1,2}$ and link $e_{2,3}$ with at most 2 transmission attempts is 1, 0.75, and 0.26 respectively. Therefore, the probability of packet drop after performing r transmission

attempts at each link is equal to 0, 0.25 and 0.73 respectively. Based on Equation 4, data transmission cost of $path_1$ in the case where the link layer provides at most 2 transmission attempts is equal to 5. While, if node n_0 selects $path_2$ to transmit a data packet towards node n_3, the expected number of packet transmissions over this path is equal to 4.19. The higher data transmission cost of $path_1$ compared to $path_2$ is due to the fact, that in the first path, the low-quality links with high probability of packet drop after performing 2 transmission attempts are located near the destination node. With respect to this example in the cases where the link layer offers a bounded number of transmission attempts, packet delivery cost of a multi-hop path is highly related to the maximum number of link layer transmissions and relative position of the links along the path. However, assuming an infinite number of link layer transmissions over individual links simplifies the calculation of the required number of packet transmissions for successful packet delivery from a source node towards the destination node. The reason is that, by this assumption the link layer never drops a packet over the links along a path. Therefore, the expected number of packet transmissions over a path only depends on the packet delivery probability of the links along the path, while their arrangement is not important.

3.2. Successful or Failed Packet Transmission Cost of Multi-Hop Paths

In data gathering protocols the sink node initiates the construction process of the network data gathering tree through broadcasting a routing packet to the network. To this aim, the utilized path cost function in these protocols should be able to estimate the data transmission cost from each node toward the sink based on the calculated cost in the reverse direction (from sink towards the sensor nodes). In this context, the SFTC is designed in such a way that it can estimate cost of data delivery from every node to the sink node through the transmitted packet from the sink node which traverses the backward links towards the nodes. However, as the proposed routing cost function considers the relative position of the links along the paths, estimating the transmission cost from every node towards the sink based on the calculated cost in the reverse direction is non-trivial.

According to the given network model and variables in Section 3.1, SFTC of a path (n_0, n_1,...,n_n) with n hops from sink node n_n towards source node n_0 can be represented as follows:

$$SFTC = (\sum_{i=n}^{1} E[e_{i-1,i}]\Psi(e_{i,n})) \tag{5}$$

where $\Psi(e_{i,n})$ is a the weighting function that scales the required number of transmission attempts at each link in order to reflect the influence of link positions and their respective data transmission quality on the data transmission cost of a path. The insight behind using this weighting function is that, it scales the expected number of link layer transmissions on a given link $e_{i-1,i}$ based on the ratio of the required number of packet transmissions for successful data delivery over the traversed links from sink node n_n till node n_i to the maximum number of offered link layer transmission attempts. In order to have a successful packet delivery over a given link through performing at most r transmission attempts, the ratio of the required number of packet transmissions over that link for successful packet delivery to the r value should be less or equal to 1. If this ratio is higher than 1, there is a probability of packet drop after performing r transmission attempts. Therefore, to calculate the data transmission cost of different paths based on the maximum number of provided link layer transmission attempts at individual links (i.e., r), weighting function $\Psi(e_{i,n})$ can take the following values:

$$\Psi(e_{i,n}) = \begin{cases} 1 & \text{if } n = i \\ \prod_{j=n}^{i+1} \Psi(e_{j,j-1}) & \text{if } n \neq i \end{cases}$$

where

$$\Psi(e_{j,j-1}) = \begin{cases} 1 & \text{if } \frac{(1/p_{j-1,j})}{r} \leq 1 \\ \frac{(1/p_{j-1,j})}{r} & \text{if } \frac{(1/p_{j-1,j})}{r} > 1 \end{cases} \tag{6}$$

Furthermore, the expected number of packet transmissions over link $e_{i-1,i}$ in terms of packet delivery probability over the links and maximum number of link layer transmission attempts in the cases where the link layer provides at most r transmission attempts has been shown through Equation 3. By substituting Equation 3 in Equation 5, the estimated cost for a single packet transmission over a path (n_0, n_1,...,n_n) with n hops in term of number of packet transmissions can be calculated as:

$$SFTC = (\sum_{i=n}^{1}((\sum_{k=1}^{r} k(1-p_{i-1,i})^{k-1}p_{i-1,i}) + r(1-p_{i-1,i})^{r})\Psi(e_{i,n})) \tag{7}$$

In order to clarify the differences between $\sum_{i=0}^{n-1} E[e_{i,i+1}]$ and SFTC in the situations where the link layer provides a bounded number of transmission attempts at individual links, Table 1 shows the data transmission cost of different paths through these path cost functions. As can be seen from this table, $\sum_{i=0}^{n-1} E[e_{i,i+1}]$ provides similar data transmission cost for the paths with equal number of hops. For instance, path (1,1,0.25) and path (0.25,1,1) have an equal $\sum_{i=0}^{n-1} E[e_{i,i+1}]$ value regardless of the number of link layer transmissions (i.e., 1 or 3 transmission attempts) and position of the links along the paths. While employing the SFTC path cost function to calculate the data transmission cost of different paths with equal $\sum_{i=0}^{n-1} E[e_{i,i+1}]$ values results in dissimilar values. For instance, through SFTC, path (0.25,1,1) has lower data transmission cost compared to path (1,1,0.25). This is due to the fact, that SFTC prefers the paths with high-quality links near the destination. In fact, if the low-quality links are located near the destination,

Table 1: Comparison of the data transmission cost of multi-hop paths with assuming a limited and an infinite number of transmission attempts at individual links.

Paths	r	SFTC	$\sum_{i=0}^{n-1} E[e_{i,i+1}]$
Path1=(1,1,0.25)	1	9	6
Path2=(0.25,1,1)	1	3	6
Path3=(0.25,1,1,1)	1	4	7
Path4=(1,1,0.25)	3	9.66	6
Path5=(0.25,1,1)	3	9	6
Path6=(0.25,1,1,1)	3	10	7

high probability of packet loss over these links results in the waste of network resources to forward data packets over the previous links along a path before those packets are dropped. Therefore, although path (0.25,1,1,1) has more hops and higher $\sum_{i=0}^{n-1} E[e_{i,i+1}]$ value compared to path (1,1,0.25), but in the cases where the link layer provides 1 transmission attempt SFTC assigns a lower cost to the first path in comparison with the second one. However, if the link layer provides 3 transmission attempts, SFTC prefers path (1,1,0.25) compared to the path (0.25,1,1,1). The reason is that, in this situation the required number of transmission attempts over a link with a p value equal to 0.25 is close to the number of provided link layer transmission attempts. Therefore, in this case SFTC gives chance to the packets to reach the destination with lower network resource utilization by selecting path (1,1,0.25) with 3 hops instead of selecting path (0.25,1,1,1) with 4 hops.

4. Link Ordering-Based Data Gathering Protocol

After network initialization, every node should identify at least one path towards the network data gathering point (i.e., sink node) [39]. Therefore, LINKORD aims to construct the minimum cost network data gathering tree which is rooted at the sink node and provides efficient data gathering during network operation through two different phases. The first phase establishes minimum cost paths from every sensor node towards the sink through the proposed SFTC path cost function. After construction of the network data gathering tree, the second phase is responsible for transmitting the collected data by the source nodes towards the sink. This section provides the detailed operation of each phase.

4.1. Data Gathering Tree Construction Phase

In the proposed LINKORD protocol every nodes uses the SFTC path cost function to identify the data transmission cost towards the sink node through its neighboring nodes. Therefore, each node should be aware about the data transmission quality of its links towards its neighboring nodes. We assume a link quality estimation process is performed at the network initialization phase before the data gathering tree construction process [20, 40]. So, during the data gathering tree construction process all the nodes can utilize their achieved link quality information to calculate the data transmission cost of different paths towards the sink node.

At the start of data gathering tree construction process through LINKORD protocol, all the nodes initialize their data transmission cost towards the sink node to an invalid value (i.e., -1) to show that they have not identified any path towards the sink so far. In LINKORD, the sink node initializes the data gathering tree construction process by broadcasting a routing packet towards its neighboring nodes with *Minimum path cost*, and $\Psi(e)$ fields which are initialized to 0 and 1 respectively (Lines 9-14 of Algorithm 1). The main format of this routing packet is illustrated in Figure 3. During the path construction process, whenever node n_i receives a routing packet from node n_j, it first searches its neighborhood table to find the preserved link quality information towards node n_j. Node n_i utilizes the preserved neighborhood information regarding neighbor n_j and included information in the received routing packet to calculate the data transmission cost towards the sink node through node n_j (Line 22 of Algorithm 1). When node n_i calculates the data transmission cast towards the sink node through node n_j, it preserves the routing information through this node in its routing table (i.e., path cost, and $\Psi(e)$ value towards the sink node). Furthermore, if node n_i has not identified any path towards the sink so far, (i.e., its preserved $minCost_{i,sink}$ variable is equal to -1), it records node n_j (i.e., the node which has sent the routing packet) as its parent node. In this case, node n_i also updates the *Minimum path cost*, and $\Psi(e)$ fields of the received routing packet according to the lines 18-31 of Algorithm 1 and rebroadcasts this packet again. While, if node n_i has a parent node with minimum path cost towards the sink, it checks whether node n_j offers a lower cost path compared to its selected parent node before it decides to rebroadcast the received routing packet. Node n_i will change its parent to node n_j, if it provides a lower cost path towards the sink node compared to the currently selected path. Since node n_i has changed its parent and it has found a new path with lower cost compared to its previously announced path, it should update the *Minimum path cost towards sink*, and $\Psi(e)$ fields of the received routing packet based on the newly calculated values and rebroadcasts this packet again (lines 33-46 of Algorithm 1). While, if the data transmission cost of the identified path through node n_j is higher than the minimum data transmission cost which has been identified by node n_i so far, node n_i only preserves the related routing cost information of node n_j (lines 47-49 of Algorithm 1). This process will be repeated between all the nodes during the data gathering tree construction process, until all the nodes identify the data transmission cost towards the sink node through their entire neighboring nodes.

4.2. Data Transmission Phase

When the network data gathering tree is established, network nodes can start to forward their collected data

Algorithm 1 Data gathering tree construction algorithm.

Notations:
RoutingPkt: Transmitted routing packet by sink for constructing the network data gathering tree
$RecRtPacket_{j,i}$: Received routing packet by node n_i which has been transmitted by node n_j
$RecRtPacket_{j,i} \rightarrow minCost$: Included value in the *Minimum path cost towards sink* field of the received routing packet by node n_i which has been transmitted by node n_j
$RecRtPacket_{j,i} \rightarrow \Psi(e_{j,sink})$: Included value in the $\Psi(e)$ field of the received routing packet by node n_i which has been transmitted by node n_j
$minCost_{i,sink}$: The minimum data transmission cost towards sink node which have been identified so far by nodes n_i
$Parent_i$: Current parent of node n_i towards the sink node
LastSFTC: The last calculated *SFTC* value
if (it is sink node) **then**
    create a *RoutingPkt*
    $RoutingPkt \rightarrow minCost$=0
    $RoutingPkt \rightarrow \Psi(e)$=1
    broadcast the *RoutingPkt*
end if
if (a *RoutingPkt* is received by node n_i from node n_j) **then**
    search the neighborhood table to find the corresponding entry for node n_j
    fetch $p_{i,j}$
    if ($minCost_{i,sink}$==-1) **then**
        for (k=1;1;r) **do**
            $E[e_{i,j}] = k(1-p_{i,j})^{k-1}p_{i,j} + r(1-p_{i,j})^r$
        end for
        $SFTC = RecRtPacket_{j,i} \rightarrow minCost + (E[e_{i,j}] \times RecRtPacket_{ji} \rightarrow \Psi(e_{j,sink}))$
        $RecRtPacket_{j,i} \rightarrow minCost = SFTC$
        if ($\frac{1/p_{i,j}}{r} \leq 1$) **then**
            $RecRtPacket_{j,i} \rightarrow \Psi(e_{i,sink}) = RecRtPacket_{j,i} \rightarrow \Psi(e_{j,sink}) \times 1$
        else
            $RecRtPacket_{j,i} \rightarrow \Psi(e_{i,sink}) = RecRtPacket_{j,i} \rightarrow \Psi(e_{j,sink}) \times (\frac{1/p_{i,j}}{r})$
        end if
        $Parent_i = node\ n_j$
        $LastSFTC = SFTC$
        broadcast the *RoutingPkt* with the updated fields
    else
        for (k=1;1;r) **do**
            $E[e_{i,j}] = k(1-p_{i,j})^{k-1}p_{i,j} + r(1-p_{i,j})^r$
        end for
        $SFTC = RecRtPacket_{j,i} \rightarrow minCost + (E[e_{i,j}] \times RecRtPacket_{j,i} \rightarrow \Psi(e_{j,sink}))$
        if ($SFTC < LastSFTC$) **then**
            $RecRtPacket_{j,i} \rightarrow minCost = SFTC$
            if ($\frac{1/p_{i,j}}{r} \leq 1$) **then**
                $RecRtPacket_{j,i} \rightarrow \Psi(e_{i,sink}) = RecRtPacket_{j,i} \rightarrow \Psi(e_{j,sink}) \times 1$
            else
                $RecRtPacket_{j,i} \rightarrow \Psi(e_{i,sink}) = RecRtPacket_{j,i} \rightarrow \Psi(e_{j,sink}) \times (\frac{1/p_{i,j}}{r})$
            end if
            $Parent_i = node\ n_j$
            $LastSFTC = SFTC$
            broadcast the *RoutingPkt* with the updated fields
        else
            preserve the calculated *SFTC* value and $\Psi(e_{j,sink})$ for neighbor n_j in the routing table
        end if
    end if
end if

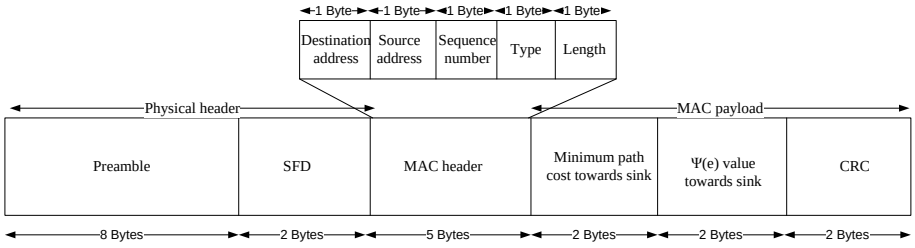

Figure 3: Routing packet format.

towards the sink node through the identified paths. In this context, whenever a node collects some information from environment or receives a data packet from one of its child nodes, it forwards these data packets towards the sink node through its selected parent node during the data gathering tree construction process. Consequently, upon reception of a data packet by a given node it first checks whether the received packet has been received previously. If it realizes that the received packet is a duplicate packet, it will discard the packet to avoid waste of network resources. While if the received packet is not duplicated and the receiver node has a free space in its packet buffer, it stores the received packet into the packet queue to forward the packet towards the sink node. Otherwise it drops the packet because of buffer overflow.

Due to the high dynamics of low-power wireless communications, limited resources of sensor nodes, and shared wireless communication medium, many-to-one data gathering pattern in wireless sensor networks can easily cause network congestion and a huge number of packet drops as the result of packet buffer overflow [41, 42]. Therefore, to reduce the effects of network congestion on the data gathering performance, the CCPC mechanism is proposed in conjunction with the developed data gathering protocol. The proposed technique exploits the incoming and outgoing rates of individual nodes to measure the congestion degree of nodes based on their data packet reception, data packet generation and packet transmission rates. To this aim every node n_i calculates the incoming rate from its individual child nodes (e.g., node n_j) as follows:

$$R_j = \frac{1}{T_j} \tag{8}$$

where T_j is the packet reception interval from node n_j.

Furthermore, every node calculates its packet forwarding rate by inverting the time period from when a packet arrives at its MAC layer until the last bit of that packet is transmitted successfully.

In order to eliminate the overhead of transmitting control packets for congestion control purpose, CCPC approach adds some control bits to the data packets during the data transmission process in order to provide valuable information for congestion control and route maintenance purposes. As demonstrated in Figure 4, the required information to perform the proposed CCPC mechanism is added to each data packet as a CCPC control frame. As with routing packet, in this packet 2 bytes are used to include the data transmission cost from every sender node towards the sink for path maintenance purpose. The *Incoming rate* and *Outgoing rate* fields share the congestion degree of every node among its neighboring nodes, which allows the child nodes of a congested node to change their current parent to the best potential parent that may reduce the network congestion level. *Selected child* field determines the identity of the child node that should change its current congested parent node. Finally, the *PC* bit shows the parent change eligibility of the sender node, and *CF* bit identifies that the sender of the packet is congested.

As can be seen from Figure 5, network nodes can be in different states during the data transmission phase in order to perform the proposed CCPC approach. Initially a node n_i is in the *no-Congestion* state. During this stage, whenever node n_i receives a newly generated data packet from the application layer or a non-duplicated data packet from one of its child nodes, it checks the filled percentage of its buffer space. If at the packet reception time the filled space in its packet buffer is equal or higher than the user specified threshold value (i.e., TQ_c), it moves to the *Congestion-Detected* state which means that it deals with congestion.

In the *Congestion-Detected* state, node n_i notifies its neighbors that its buffer will be overflowed in the near future by setting the *CF* bit of its outgoing data packets to 1. Moreover, node n_i waits for a period to find a child node which is eligible to change its current parent (i.e., node n_i) through setting a timer as:

$$T_{cs} = \rho \times \frac{1}{R_{avg}}$$
$$\text{where} \tag{9}$$
$$R_{avg} = \frac{R_i^c}{\rho}$$

where R_{avg} is the average packet reception rate at node n_i from its child nodes, R_i^c is the packet reception rate at node n_i from all of its child nodes, and ρ is the number of active child nodes of a given node.

When node n_j overhears a transmitted packet from its current parent (i.e., node n_i) with the *CF* bit equal to 1, it identifies that the packet buffer of its current parent is being overflowed. Therefore, it starts to find another parent node among its identified potential parent nodes to change its current parent. To this aim, node n_j retrieves

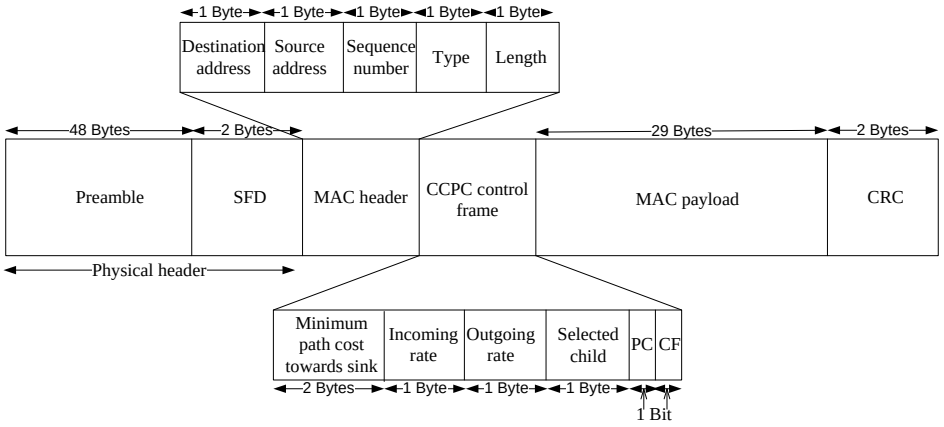

Figure 4: Data packet format.

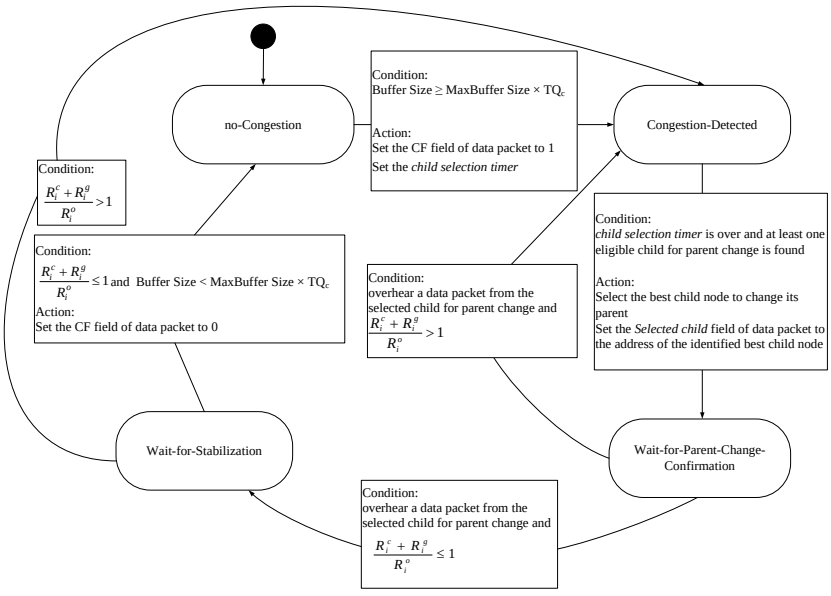

Figure 5: The state diagram of CCPC.

the preserved incoming and outgoing rates for its potential neighboring nodes from its neigborhood table to find the best parent node n_k as follows:

$$\Delta R = R_k^o - (R_k^c + R_k^g + R_j^o) \quad (10)$$

where ΔR indicates the difference between packet reception rate and packet transmission rate of a node, R_k^o is the packet transmission rate of node n_k, R_k^c is the packet reception rate of node n_k from its child nodes, and R_k^g is the packet generation rate at the application layer of node n_k. Notice that, a potential parent node n_k for node n_j is a node which holds the inequality $CostToSink_k \leq MinCostVal + \beta \times MinCostVal$. In this inequality, $CostToSink_k$ is the data transmission cost from node n_k towards the sink node, $MinCostVal$ is the minimum data transmission cost that node n_i have been calculated so far, $0 \leq \beta \leq 1$ is a threshold value to select a neighboring node towards the sink as a potential parent.

If node n_j finds a potential parent node to change its current parent, it sets the PC bit of its outgoing data packets to 1. When node n_i which is in the *Congestion-Detected* state receives a data packet from a child node n_j with PC bit equal to 1, it updates the related information regarding the parent change eligibility of this child node in its neighborhood table. If the waiting time for child selection (i.e., T_{cs}) is over, node n_i searches its neighborhood table to find a child node with maximum data transmission rate which is eligible to change node n_i to another parent node. Afterwards, node n_i sets the *Selected child* field of its outgoing data packets to the address of the selected child node and changes its current state to the *Wait-for-Parent-Change-Confirmation*.

If node n_j overhears a data packet from its current parent (i.e., node n_i) with the *Selected child* field equal to its address (i.e., node n_j), it should change its current parent from node n_i to its identified best potential parent n_k.

Whenever node n_i which is in the *Wait-for-Parent-Change-Confirmation* state overhears a transmitted data packet from its selected child (i.e., node n_j) for parent change process, it realizes that child node n_j has changed its current parent to another node. Therefore, node n_i updates the preserved packet reception rate information in its neighborhood table regarding to this child node to 0. Furthermore, it checks its current congestion status as:

$$CD = \frac{R_i^c + R_i^g}{R_i^o} \quad (11)$$

If $R_i^c + R_i^g$ exceeds the R_i^o, there still some backlogged packets in node n_i's packet buffer. Thus, node n_i should change it state to the *Congestion-Detected* state again to balance its packet reception and transmission rates. While, if packet arrival rate of node n_i's buffer is smaller or equal to its outgoing rate, n_i realizes that congestion is abated and it can move to the *Wait-for-Stabilization* state.

During the *Wait-for-Stabilization* state, whenever node n_i receives a data packet, it checks the ratio of its packet incoming rate plus its packet generation rate to its packet outgoing rate. If this ratio is lower than 1, node n_i recognizes that the congestion is alleviated. So it can move to the *no-Congestion* state when the filled space in its packet buffer is lower than the user specified threshold value. Otherwise, it should enter the *Congestion-Detected* state again to equalize its packet reception and transmission rates.

LINKORD also utilizes the network traffic to update the preserved link quality information at individual nodes and detects routing problems during the data transmission process. In this context, every node updates the preserved information regarding the data transmission quality of its links towards its potential parents nodes based on number of acknowledged, unacknowledged and overheard data packets from them. Moreover, every node updates the preserved neighborhood information in its neighboring table during the data transmission process using the *Minimum path cost towards sink* field of the transmitted packets. In addition, whenever a node receives a data packet, it compares the included path cost in the packet with its own path cost towards the sink to identify if there exists any routing inconstancy. If the data transmission cost of the receiver node is higher than the included cost in the data packet, the receiver node starts to update the routing state of its neighboring nodes.

5. Performance Evaluation

This section analyzes and compares performance of LINKORD against default CTP of TinyOS [12, 21]. First we describe the considered simulation framework, simulation scenarios and performance parameters for performance evaluation. Then we analyze and discuss the simulation results.

5.1. Simulation Setup

We have performed our performance evaluations using the OMNeT ++ framework. In order to provide an accurate wireless channel model and improve the accuracy of the simulation results, we have developed a physical layer module that considers path loss, multipath effect, transmission power variations, noise floor variations and the capture effect based on the presented models in [43–46]. The radio parameters are chosen based on the Mica2 mote specifications. Furthermore, we have implemented B-MAC [47] as the underlying MAC protocol in our simulation software. In all of the figures, each result point shows the median of 20 simulation runs, while the error bars represent the upper and lower quartiles. Table 2 represents the default simulation parameters of this paper in detail.

Since LINKORD lies in the similar subset of the design space as the default CTP of TinyOS [12, 21], this protocol is selected as the benchmark for performance evaluations under two different simulation scenarios as follows:

i. **First Simulation Scenario**: This scenario aims to evaluate the efficiency of LINKORD and TinyOS's

Table 2: Simulation parameters.

Radio	
Average noise power [dBm]	-106
Noise figure [dB]	13
Switch to TX/RX [μs]	250
Radio sampling [μs]	350
Evaluate radio sample [μs]	100
Noise bandwidth [Hz]	30000
Modulation	NC-FSK
Encoding	Manchester
Baud rate [Bauds per second]	38400
Transmission power [dBm]	0
Standard deviation of transmission power heterogeneity	1.2
Standard deviation of noise floor heterogeneity	0.9
Number of settling bits	49
Radio speed after encoding [bits per second]	19200
Reference distance [m]	1
PL (d_0) [dB]	55
Environment	
Ambient temperature [C°]	27
Path loss exponent (outdoor)	4.7
Multipath channel variations (outdoor)	3.2
B-MAC	
Initial backoff [slots]	32
Congestion backoff [slots]	16
Sampling interval [ms]	20
Other parameters	
Network topology	Random
Number of nodes	400
Area size	40mX40m, 60mX60m
Packet buffer size [packets]	12
TQ_c	0.5
β	0.1

CTP to discover high performance paths from sensor nodes towards the sink. Since the position of the links along the paths with small number of hops has little or no impact on the network data gathering performance, the location of source and sink nodes should be adjusted in a way, that results in construction of long distance paths. Therefore, source nodes should be at the farthest distance from the sink node to maximize the length of established paths from source nodes towards the sink. To achieve this goal, in this scenario only 20 nodes with the largest distance from sink node are considered as the source nodes and each one generates 20 data packets destined to the sink node. This setting provides a framework to evaluate the effectiveness of considering link positions during the data gathering tree construction process to achieve an efficient data transmission over long paths in wireless sensor networks. Furthermore, to study the influence of number of offered link layer transmission attempts per packet delivery, different performance parameters are evaluated under situations where the link layer provides 1 and 3 transmission attempts at individual links.

ii. **Second Simulation Scenario**: The second simulation scenario is intended to study the proficiency of different protocols to reduce the effects of network congestion on the data gathering performance in the applications where all the nodes generate data at a same rate. Therefore, in all the experiments under this scenario, the congestion control capability of TinyOS's CTP is enabled [12], and the proposed CCPC approach is used during the data transmission phase of LINKORD. Furthermore, every sensor node generates 30 data packets destined to the sink node. As with the first simulation scenario, all the performance parameters are evaluated under situations where the link layer provides 1 and 3 transmission attempts at individual links to study the effects of the number of provided transmission attempts per packet at the link layer on the network data gathering

performance. In order to be sure that different protocols are compared in the situation where the network gets congested, various performance metrics are evaluated against packet generation interval of the sensor nodes ranges from 1 to 64 seconds. Since the link level throughput in the developed simulation framework is approximately 28 packets per second and the considered network topology has 399 sensor nodes, so the network should become congested when each node generates a data packet every 14 seconds.

5.2. Performance Parameters

We have evaluated and compared the performance of the LINKORD and TinyOS's CTP using following parameters:

i. **Data delivery ratio**: This metric measures the ratio of successfully received data packets by the sink node to the total number of transmitted data packets by the source nodes. Therefore, it indicates the ability of different protocols to enhance the data transmission reliability.

ii. **End-to-end network goodput**: This metric is measured as the ratio of the total number of successfully received data bits by the sink node to the data transmission duration. In other words, this metric shows how different data gathering protocols influence the rate of successful packet delivery to the sink node.

iii. **Average Consumed Energy for Packet Transmissions**: This metric reveals the average consumed energy by individual sensor nodes for transmitting data packets towards the sink node which is presented as the percentage of total battery capacity of a sensor node.

iv. **Packet delivery overhead**: This metric indicates the overhead cost of running different data gathering protocols to provide data gathering service in wireless sensor networks through measuring the ratio of all the transmitted packets during the data transmission process to the number of successfully received data packets at the sink node.

5.3. Performance Evaluation Under First Simulation Scenario

This section compares the performance of LINKORD with default data gathering protocol of TinyOS in order to validate the effects of link positions along the paths on the network data gathering performance in the cases where the link layer provides a bounded number of transmissions.

5.3.1. Data Delivery Ratio

Figure 6 shows the ratio of the received data packets at the sink node to the transmitted data packets by the source nodes through employing LINKORD and TinyOS's CTP. As can be seen from this figure, in the cases where the link

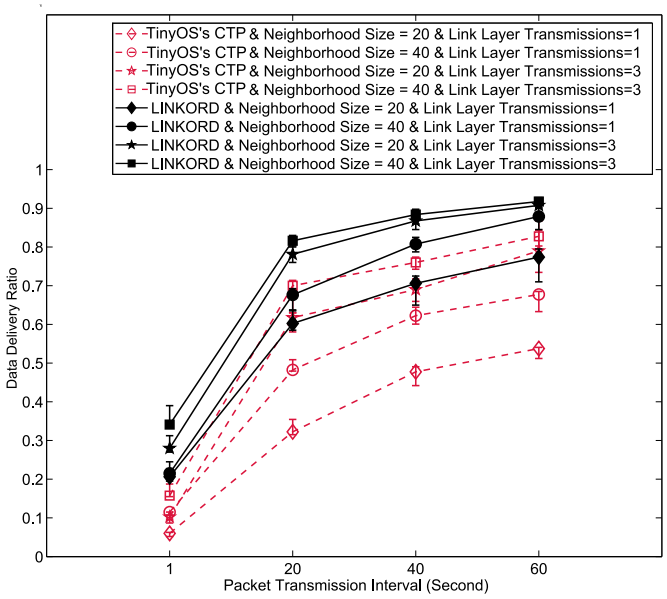

Figure 6: Data delivery ratios achieved by LINKORD and TinyOS's CTP versus network traffic load under the first simulation scenario.

layer provides 1 and 3 transmission attempts at each link LINKORD improves data delivery ratio about 70% and 33% compared to the TinyOS's CTP respectively. The reason behind this trend is that, LINKORD constructs more reliable paths compared to the TinyOS's CTP which calculates the data transmission cost of different paths through a simple summation of the ETX values of the links along the paths. According to the LINKORD design, this protocol calculates the data transmission cost of each path based on the required number of transmission attempts for successful packet delivery over individual links of that path and the maximum number of offered link layer transmissions. Therefore, LINKORD assigns higher data transmission cost to the paths with links where the expected number of transmission attempts for successful packet delivery is higher than the maximum number of offered link layer transmissions. However, as TinyOS's CTP assumes there is an infinite number of transmission attempts at individual links, in the cases where the link layer provides a limited number of transmissions, it selects paths that are more unreliable compared to the selected paths by the LINKORD. Consequently, as the number of offered link layer transmission attempts increases the achieved data delivery ratio through both approaches is getting close to each other. In both approaches by increasing the neighborhood density of nodes from 20 to 40 nodes, the data deliver ratio elevates about 33%. The reason for this incremental trend is that, elevating the neighborhood density decreases the number of packet drops due to the hidden node terminal problem. Furthermore, raising the network density results in the paths with a lower number of hops. Consequently, data transmission over shorter paths reduces the channel access contentions and wireless interference between network nodes which in turn improves the data delivery ratio.

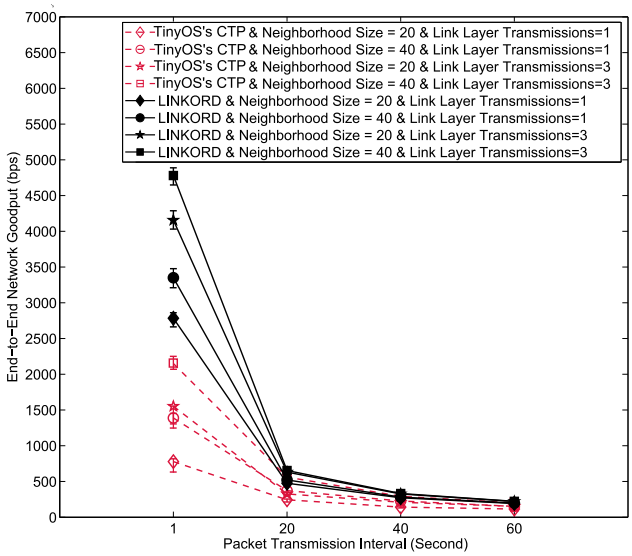

Figure 7: The end-to-end network goodput achieved by LINKORD and TinyOS's CTP versus network traffic load under the first simulation scenario.

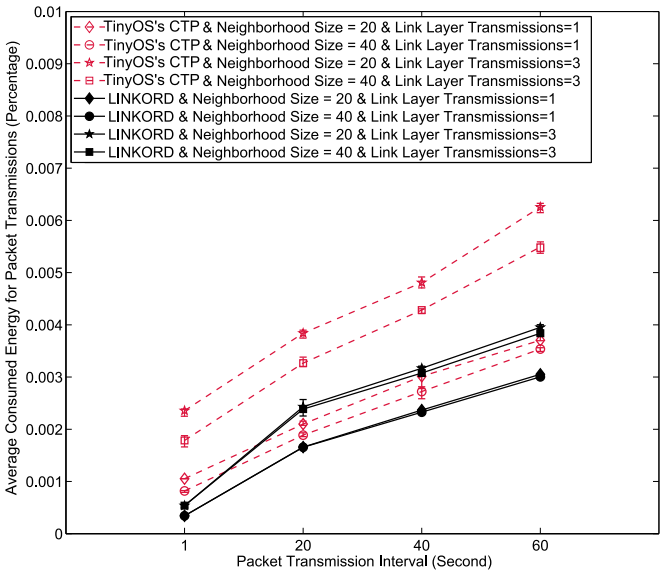

Figure 8: Percentage of average consumed energy for packet transmission towards the sink node by LINKORD and TinyOS's CTP versus network traffic load under the first simulation scenario.

5.3.2. End-to-End Network Goodput

Figure 7 depicts the measured end-to-end goodput for LINKORD and CTP of TinyOS against packet generation rate of the source nodes. As it can be observed, in insensitive traffic loads LINKORD improves the end-to-end network goodput about 80% and 70% compared to the TinyOS's CTP in the situations where the link layer provides 1 and 3 transmission attempts. By reducing the network traffic load, still LINKORD provides higher end-to-end network goodput relative to the TinyOS's CTP. This behavior is mainly due the fact that LINKORD considers the relative position of the links along the paths with respect to their packet delivery probability and the number of offered link layer transmission attempts. Since the network traffic is convergecast, the traffic load of the nodes near the sink node is always higher than the other nodes. Therefore, as LINKORD prefers the paths with high-quality links near the sink node and low-quality links near the source nodes, it can help to reduce the traffic load of the nodes near the sink node and improves the end-to-end network goodput. Another observation that can be drawn from this figure is that, as the neighborhood size of nodes raises from 20 to 40 the provided end-to-end network goodput by both approaches elevates. This observation can be described as follows: Firstly, as discussed in the previous section, increasing the neighborhood size of individual nodes reduces the number of packet drops due to the hidden node terminal problem which in turn improves the network goodput. Secondly, increasing the neighborhood density results in the paths with a lower number of hops from every sensor node towards the sink. Therefore, data transmission over paths with lower number of hops reduces the channel contention level among the network nodes which in turn elevates the network goodput.

5.3.3. Average Consumed Energy for Packet Transmissions

Figure 8 demonstrates the percentage of average consumed energy for packet delivery to the sink node against network traffic load through using LINKORD and TinyOS's CTP. As expected, LINKORD reduces the percentage of average consumed energy for transmitting data packets towards the sink node about 50% and 30% compared to the TinyOS's CTP in the cases where the link layer provides 1 and 3 transmission attempts respectively. This is mainly due to the utilized path cost function in LINKORD. In fact, LINKORD assigns higher data transmission cost to the paths with low-quality links near the sink node by using the SFTC path cost function. Thus, it significantly reduces the number of packet drops due to the limited number of link layer transmissions on the links near the sink node. As a consequence, LINKORD highly reduces the network energy waste due to transmitting data packets over a large number of hops which will be dropped somewhere close to the sink node.

5.3.4. Packet Delivery Overhead

In order to study the overhead caused by different protocols to transmit data packets towards the sink node, this section analyzes the packet delivery overhead of different protocols which is defined as the total number of transmitted packets during the data transmission phase to the number of received packets at the sink node. Figure 9 presents the packet delivery overhead caused by employing different protocols as a fraction of the network traffic load. As expected LINKORD reduces the packet delivery overhead by 41% and 30% compared to the TinyOS's CTP when the link layer provides 1 and 3 transmission attempts at each link. This is a direct result of considering the relative position of the links along the paths with respect to their

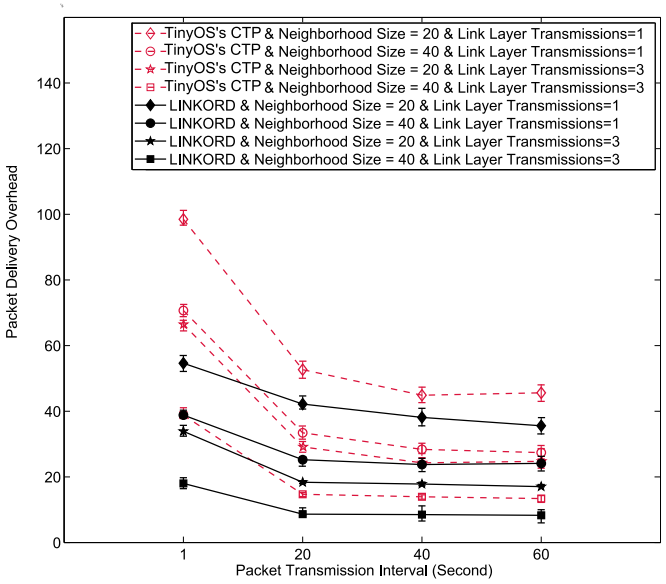

Figure 9: Packet delivery overhead of LINKORD and TinyOS's CTP various network traffic load under the first simulation scenario.

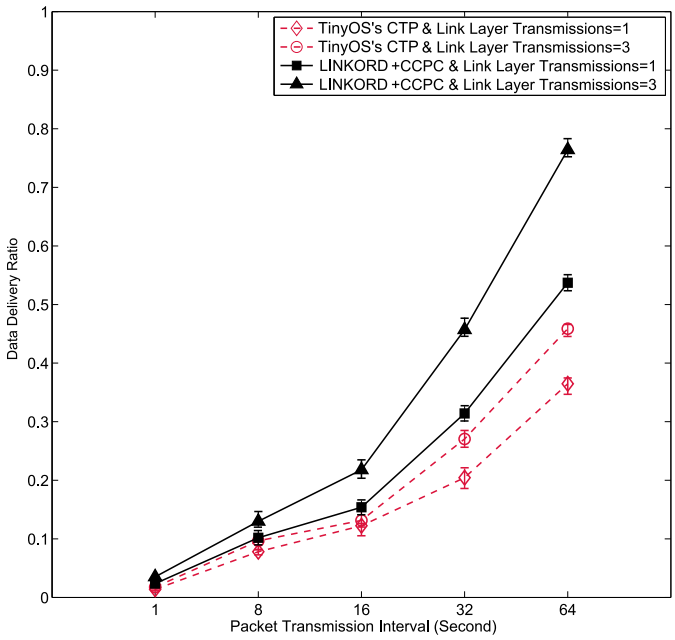

Figure 10: Data delivery ratios achieved by LINKORD, and TinyOS's CTP versus network traffic load under the second simulation scenario.

packet delivery probability and maximum number of provided link layer transmissions. In fact, LINKORD reduces packet delivery overhead through selecting paths which incur a lower number of transmissions while they provide higher data delivery ratio. Furthermore, increasing the neighborhood size from 20 to 40, reduces the packet delivery overhead of both protocols. This decreasing trend can be explained as: Raising the neighborhood size of nodes results in the paths with lower number of hops which in turn reduces the total number of packet transmissions for a single packet delivery over each path.

5.4. Performance Evaluation Under Second Simulation Scenario

This section studies the network data gathering performance through employing LINKORD, and TinyOS's CTP in the cases where all the nodes generate data at a same rate and both protocols utilize congestion control mechanism during the data transmission process.

5.4.1. Data Delivery Ratio

Figure 10 presents the achieved data delivery ratio through different protocols as a function of packet generation rate of sensor nodes under the second simulation scenario. This figure reveals that LINKORD provides up to 60% and 50% higher data delivery ratio compared to the TinyOS's CTP in the cases where the link layer provides 1 and 3 transmission attempts at each link respectively. This performance improvement is mainly due to two reasons. The first reason is that, LINKORD considers the position of the links along the paths based on their packet delivery probability and the number of offered link layer transmissions at each link. Thus, these results confirm the inefficiency of calculating data transmission cost of multi-hop paths through summation of their link ETX values in the cases where the link layer provides a limited number of transmission attempts at every link. The second reason is that by employing the CCPC approach during the data transmission phase, whenever a given child node realizes its current parent may face packet drop in the near future due to buffer overflow, it starts to forward its packets towards the sink node through another eligible parent node. In CCPC approach, every node which conceives it should change its current parent, considers the packet reception and transmission rate of its potential parents to select the best potential parent node instead of making a blind parent selection (e.g., in TinyOS's CTP a child selects a new parent based on its offered data transmission cost). Through the CCPC approach, each node first perceives how much would be the ratio of packet input rate to the output rate of a potential parent node, if it selects that node as its parent. In other words, through the CCPC approach each child node can choose a parent which has less probability of being congested compared to the other potential parents when this child node switches to that newly selected parent. In TinyOS's CTP, every node which realizes its current parent is congested, blindly changes its parent to another one which provides lower multi-hop ETX towards the sink node compared to other neighbors. Therefore, since in TinyOS's CTP nodes do not consider the incoming and outgoing rates of their neighbors to change their congested parent node, the achieved data delivery ratio through this protocol is significantly lower than LINKORD.

5.4.2. End-to-End Network Goodput

Figure 11 shows the achieved end-to-end goodput by LINKORD compared to the TinyOS's CTP under the sec-

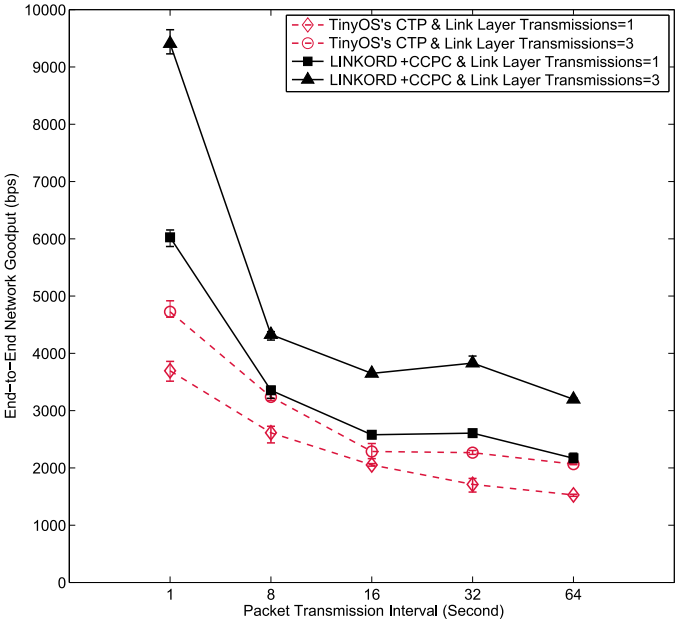

Figure 11: The end-to-end network goodput achieved by LINKORD and TinyOS's CTP versus network traffic load under the second simulation scenario

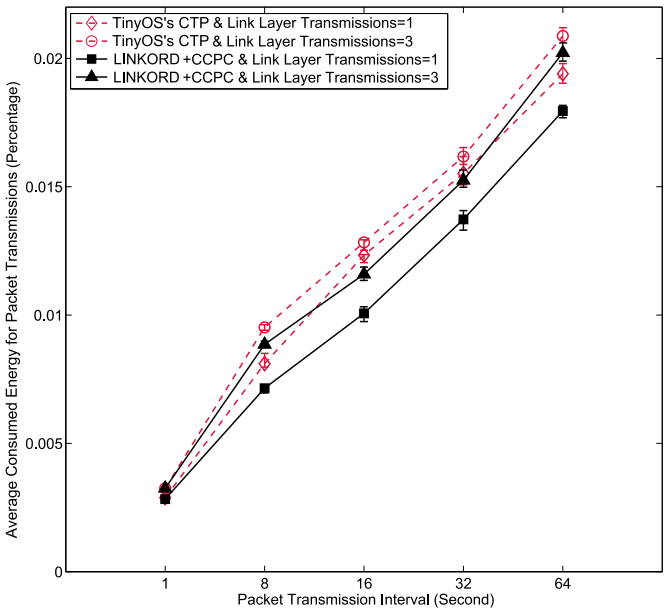

Figure 12: Percentage of average consumed energy for packet transmission towards the sink node by LINKORD and TinyOS's CTP versus network traffic load under the second simulation scenario.

ond simulation scenario. As with the achieved results under the first simulation scenario, the goodput of both protocols increases as the network traffic load intensifies. Again in this scenario, LINKORD improves the end-to-end network goodput compared to the TinyOS's CTP. The first reason behind this behavior is that, TinyOS's CTP assumes the link layer offers an infinite number of transmissions per packet and it does not consider the position of the links along the paths. Therefore, when the link layer provides a bounded number of transmissions, the selected paths through TinyOS's CTP may include links which require a higher number of transmissions for successful packet delivery than the maximum number of offered link layer transmissions. While, LINKORD arranges the links along the paths based on their packet delivery probability and maximum number of achievable transmissions at the link layer. Moreover, these results confirm that under high traffic loads where the probability of congestion is very high, balancing the incoming and outgoing traffic rates of network nodes results in efficient network bandwidth utilization. Since the proposed CCPC approach adjusts the data reception rate of very node based on its service rate, it allows the nodes to efficiently use the network resources for delivering the collected data from environment to the sink node.

5.4.3. Average Consumed Energy for Packet Transmissions

Figure 12 plots the percentage of average consumed energy for packet transmission towards the sink node through different protocols under the second simulation scenario. As can be seen from this figure LINKORD reduces the consumed energy for packet transmission up to 30% and 20% compared to the TinyOS's CTP when the link layer provides 1 and 3 transmission attempts at every link respectively. This performance improvement is a direct consequence of using SFTC path cost function during the data gathering tree construction phase of LINKORD. Another reason is that CCPC enables every child node of a congested parent to change its current parent to another node which can handle the outgoing rate of this child node without any congestion. In fact, this approach helps each congested node to reduce the number of packet drops due to the buffer overflow through adjusting its packet incoming rate according to its packet outgoing rate. Consequently, by reducing the probability of packet drops due to the buffer overflow, the consumed energy for packet transmission is reduced.

5.4.4. Packet Delivery Overhead

Figure 13 depicts the ratio of the total number of transmitted packets during the data transmission phase to the number of received packets at the sink node as a function of network traffic load for different protocols. This figure shows that under this new scenario the delivery cost of LINKORD is still 43% and 30% lower than TinyOS's CTP when the link layer provides 1 and 3 transmission attempts at each link respectively. This is because of the utilized route selection mechanism in LINKORD which allows the nodes to select paths that cause a lower number of transmissions and higher data delivery ratio compared to the TinyOS's CTP. Furthermore, as the required information to perform CCPC approach is added to the data packets as a CCPC control frame, employing this approach in conjunction with LINKORD has not increased the data delivery overhead compared to TinyOS's CTP. Moreover, as CCPC approach enables every node to adjust its packet incoming rate based on its packet outgoing rate, it signifi-

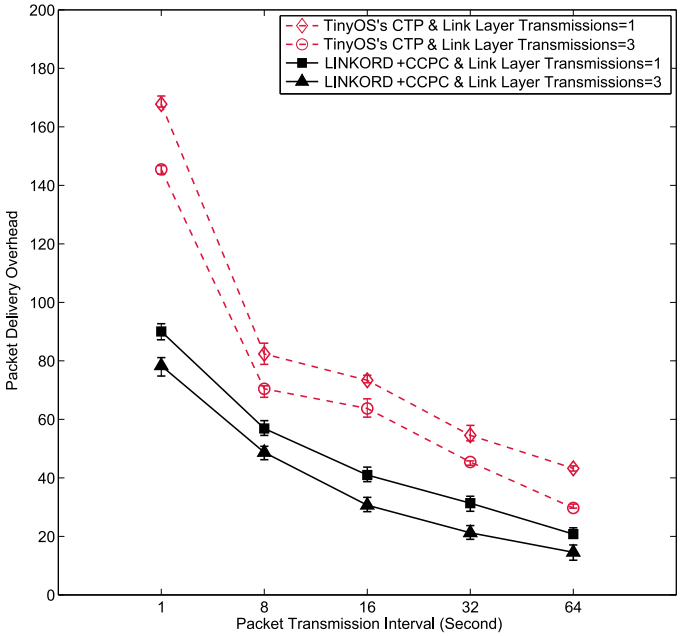

Figure 13: Packet delivery overhead of LINKORD and TinyOS's CTP versus network traffic load under the second simulation scenario.

cantly reduces the total number of transmission attempts for a successful packet delivery.

6. Conclusion

This paper proposed a data gathering protocol called LINKORD to provide efficient data gathering with respect to the bounded number of link layer transmissions and packet delivery probability of network links. In the situations where the link layer provides a limited number of transmissions attempts at each link, the reliability and cost of data delivery is not only related to the number of hops along the paths or their respective data transmission quality. In fact, since packet drops near the sink node due to the bounded number of link layer transmissions is very costly, the relative position of the links along each path plays an important role in computing the data transmission cost of individual paths. Therefore, the main aim of LINKORD is to calculate the data transmission cost of every path based on the relative position of the links along that path. In this context, this paper also proposes a SFTC path cost function which is a non-commutative function of the packet delivery probability of the links. The proposed path cost function is utilized by the network nodes during the data collection tree construction process to identify low-cost paths towards the sink. Furthermore, a CCPC approach is developed in conjunction with LINKORD in order to provide efficient data delivery during the data transmission phase. This approach allows every node to adjust its packet incoming rate based on its packet outgoing rate whenever it identifies its packet buffer will be overflowed in the near future, through notifying the child nodes to change their congested parent.

The simulation results show the higher performance of LINKORD compared to the TinyOS's CTP in terms of data delivery ratio, end-to-end network goodput, energy consumption and packet delivery overhead. The achieved results reveal that, by taking into consideration the relative order of the links along the paths and controlling the incoming rate of the nodes during the data transmission phase, data gathering performance improves significantly when the link layer provides a bounded number of transmission attempts per packet delivery.

References

[1] J. Yick, B. Mukherjee, D. Ghosal, Wireless Sensor Network Survey, Computer Networks 52 (2008) 2292–2330.

[2] C. F. García-hernández, P. H. Ibargüengoytia-gonzález, J. García-hernández, J. A. Pérez-díaz, Wireless Sensor Networks and Applications: A Survey, International Journal of Computer Science and Network Security 7 (2007) 264–273.

[3] E. E. P. K. Gilbert, K. Baskaran, E. B. Elijah Blessing, Research Issues in Wireless Sensor Network Applications: A Survey, International Journal of Information and Electronics Engineering 2 (2012) 702–706.

[4] T. Arampatzis, J. Lygeros, S. Manesis, A Survey of Applications of Wireless Sensors and Wireless Sensor Networks, in: Proceedings of the 2005 IEEE International Symposium on Control and Automation Intelligent Control, IEEE, Limassol, Cyprus, 2005, pp. 719–724.

[5] D. England, B. Veeravalli, A Robust Spanning Tree Topology for Data Collection and Dissemination in Distributed Environments, IEEE Transactions on Parallel and Distributed Systems 18 (2007) 608–620.

[6] B. Dezfouli, M. Radi, S. A. Razak, K. Whitehouse, K. A. Bakar, T. Hwee-pink, Improving Broadcast Reliability for Neighbor Discovery , Link Estimation and Collection Tree Construction in Wireless Sensor Networks, Computer Networks 62 (2014) 101–121.

[7] M. Radi, B. Dezfouli, K. A. Bakar, M. Lee, Multipath Routing in Wireless Sensor Networks: Survey and Research Challenges, Sensors 12 (2012) 650–685.

[8] M. Radi, B. Dezfouli, K. A. Bakar, S. A. Razak, M. A. Nematbakhsh, Interference-Aware Multipath Routing Protocol for QoS Improvement in Event-Driven Wireless Sensor Networks, Tsinghua Science & Technology 16 (2011) 475–490.

[9] M. Radi, B. Dezfouli, K. A. Bakar, S. A. Razak, T. Hwee-Pink, IM2PR: interference-minimized multipath routing protocol for wireless sensor networks, Wireless Networks (2014).

[10] S. Moeller, A. Sridharan, B. Krishnamachari, Routing Without Routes: The Backpressure Collection Protocol, in: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10), Stockholm, Sweden, pp. 279–290.

[11] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, A. Woo, The Collection Tree Protocol (CTP), Technical Report, TEP 123, TinyOS Network Working Group, 2006.

[12] U. Colesanti, S. Santini, The Collection Tree Protocol for the Castalia Wireless Sensor Networks Simulator, Technical Report, No 729, Department of Computer Science, ETH Zurich, Switzerland, 2011.

[13] A. Woo, T. Tong, D. Culler, Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, ACM, Los Angeles, CA, USA, 2003, pp. 14–27.

[14] K. Srinivasan, P. Dutta, A. Tavakoli, An Empirical Study of Low Power Wireless, ACM Transactions on Sensor Networks 6 (2010) 1–49.

[15] S. Das, H. Pucha, K. Papagiannaki, Studying Wireless Routing Link Metric Dynamics, in: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07), pp. 327–332.
[16] N. Baccour, M. B. Jamaa, A Comparative Simulation Study of Link Quality Estimators in Wireless Sensor Networks, in: Proceedings of 17th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '09), London, pp. 301–310.
[17] A. Vlavianos, L. K. Law, I. Broustis, S. V. Krishnamurthy, M. Faloutsos, L. Kong, Assessing Link Quality in IEEE 802.11 Wireless Networks: Which is the Right Metric?, in: Proceedings of the 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '08), Cannes, French Riviera, France, pp. 1–6.
[18] K.-H. Kim, K. G. Shin, On Accurate Measurement of Link Quality in Multi-Hop Wireless Mesh Networks, in: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom '06), Los Angeles, CA, USA, pp. 38–49.
[19] R. Draves, B. Zill, J. Padhye, Comparison of Routing Metrics for Static Multi-Hop Wireless Networks, ACM SIGCOMM Computer Communication Review 34 (2004) 133–144.
[20] D. S. J. D. Couto, D. Aguayo, J. Bicket, R. Morris, A High-Throughput Path Metric for Multi-Hop Wireless Routing, in: ACM Mobicom Conference, ACM, San Diego, CA, USA, 2003, pp. 134–146.
[21] O. Gnawali, R. Fonseca, K. Jamieson, Collection Tree Protocol, in: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09), Berkeley, California, pp. 1–14.
[22] G. Jakllari, S. Eidenbenz, Link Positions Matter : A Noncommutative Routing Metric for Wireless Mesh Networks, IEEE Transactions on Mobile Computing 11 (2012) 61–72.
[23] F. Wang, J. Liu, Networked Wireless Sensor Data Collection: Issues, Challenges, and Approaches, IEEE Communications Surveys and Tutorials 13 (2011) 673 – 687.
[24] A. Cerpa, J. L. Wong, M. Potkonjak, D. Estrin, Temporal Properties of Low Power Wireless Links: Modeling and Implications on Multi-Hop Routing, in: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05), Urbana-Champaign, IL, USA, pp. 414–425.
[25] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, Technical Report, UCLA/CSD-TR 02-0013, Computer Science Department, UCLA, 2002.
[26] A. Meier, T. Rein, J. Beutel, L. Thiele, Coping with Unreliable Channels: Efficient Link Estimation for Low-Power Wireless Sensor Networks, in: Proceedings of the 5th International Conference on Networked Sensing Systems, Kanazawa, pp. 19–26.
[27] N. Baccour, A. Kouba, L. Mottola, M. A. Zuniga, H. Youssef, C. A. Boano, M. Alves, Radio Link Quality Estimation in Wireless Sensor Networks : A Survey, ACM Transactions on Sensor Networks 8 (2012) 183–217.
[28] S. Lin, G. Zhou, K. Whitehouse, Y. Wu, Towards Stable Network Performance in Wireless Sensor Networks, in: Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS '09), Washington, DC, USA, pp. 227 – 237.
[29] V. C. Borges, M. Curado, E. Monteiro, Cross-Layer Routing Metrics for Mesh Networks: Current Status and Research Directions, Computer Communications 34 (2011) 681–703.
[30] TinyOS Network working group, The MultihopLQI Protocol, 2009.
[31] K. Srinivasan, P. Levis, RSSI is Under Appreciated, in: Proceedings of the 3th ACM Workshop on Embedded Networked Sensors (EmNets '06), Boston, MA, USA, pp. 1–5.
[32] D. Puccinelli, M. Haenggi, Duchy: Double Cost Field Hybrid Link Estimation for Low-Power Wireless Sensor Networks, in: Proceedings of the 5th Workshop on Embedded Networked Sensors (HotEmNets'08), Charlottesville, Virginia, USA, pp. 1–5.
[33] C. A. Boano, M. A. Zuniga, T. Voigt, A. Willig, K. Romer, The Triangle Metric: Fast Link Quality Estimation for Mobile Wireless Sensor Networks, in: Proceedings of 19th International Conference on Computer Communications and Networks (ICCCN'10), Zurich, Switzerland, pp. 1–7.
[34] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle : A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks, in: Proceddings of the First Symposium on Networked System Design and Implementation (NSDI '04), San Francisco, CA.
[35] O. Gnawali, K. Jamieson, P. Levis, R. Fonseca, Four-Bit Wireless Link Estimation, in: Proceedings of the 6th Workshop on Hot Topics in Networks (HotNets '06), Atlanta, GA, USA, pp. 1–7.
[36] T. Schoellhammer, B. Greenstein, Hyper: A Routing Protocol to Support Mobile Users of Sensor Networks, Tech Report, Center for Embedded Network Sensing (CENS) (2006).
[37] J. Heidemann, D. Estrin, Centralized Routing for Resource-Constrained Wireless Sensor Networks, Technical Report August, UCLA, Los Angeles, CA, USA, 2007.
[38] N. Burri, P. V. Rickenbach, Dozer: Ultra-Low Power Data Gathering in Sensor Networks, in: Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07), Cambridge, Massachusetts, USA, pp. 450–459.
[39] M. Radi, B. Dezfouli, K. A. Bakar, S. A. Razak, M. Lee, Network Initialization in Low-Power Wireless Networks: A Comprehensive Study, The Computer Journal In Press (2013) 1–24.
[40] M. Radi, B. Dezfouli, K. A. Bakar, S. A. Razak, Integration and Analysis of Neighbor Discovery and Link Quality Estimation in Wireless Sensor Networks, The Scientific World Journal 2014 (2014) 1–23.
[41] W.-w. Fang, J.-m. Chen, L. Shu, T.-s. Chu, D.-p. Qian, Congestion Avoidance, Detection and Alleviation in Wireless Sensor Networks, Journal of Zhejiang University Science C 11 (2009) 63–73.
[42] V. Deshpande, P. Sarode, S. Sarode, Root Cause Analysis of Congestion in Wireless Sensor Network, International Journal of Computer Applications 1 (2010) 31–34.
[43] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, D. Culler, Exploiting the Capture Effect for Collision Detection and Recovery, in: Proceedings of The 2nd IEEE Workshop on Embedded Networked Sensors, Sydney, Australia, pp. 45–52.
[44] M. Z. n. Zamalloa, B. Krishnamachari, An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links, ACM Transactions on Sensor Networks 3 (2007) 165–199.
[45] G. Zhou, T. He, S. Krishnamurthy, Models and Solutions for Radio Irregularity in Wireless Sensor Networks, ACM Transactions on Sensor Networks 2 (2006) 221–262.
[46] B. Dezfouli, M. Radi, S. A. Razak, K. A. Bakar, T. Hwee-pink, Modeling Low-Power Wireless Communications, Computer networks and its Applications (2014) 1–31.
[47] J. Polastre, J. Hill, D. Culler, Versatile Low Power Media Access for Wireless Sensor Networks, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04), Maryland, USA, pp. 95–107.