A multipath prefix routing for wireless sensor networks

Moufida Maimour, Zahia Bidai

Noname manuscript No.
(will be inserted by the editor)

A Multipath Prefix Routing for Wireless Sensor Networks

Moufida Maimour · Zahia Bidai

Received: date / Accepted: date

Abstract When a spanning tree is built on top of a wireless network and an appropriate labelling scheme is applied, the complexity of the routing mechanism in terms of memory usage and control messages can be drastically reduced using compact routing. In this paper, we propose MPR (Multipath Prefix Routing), a multipath routing protocol for wireless sensor networks. MPR is a hybrid (both reactive and proactive) protocol that operates on an already built spanning tree rooted at the sink (the collect station). Besides the tree path, additional paths are built based on an appropriate labelling scheme and neighbourhood relationships. For practical implementation, we propose and evaluate two different labelling schemes. We mainly show that in a perfect W-ary tree, the additional number of bits required to encode labels is at most $(H-1)$ where H is the height of the tree. Finally, we apply MPR to the ZigBee standard and evaluate its performance using simulations. MPR has a small state routing while control messages overhead is maintained low compared to traditional multipath routing protocols.

Keywords Tree Routing · Prefix Routing · Multipath Routing · Labelling Schemes · Wireless Sensor Networks · ZigBee

M. Maimour
Université de Lorraine, CNRS, CRAN, UMR 7039
Campus Sciences, BP 70239, Vandœuvre-lès-Nancy Cedex, 54506, France
E-mail: Moufida.Maimour@univ-lorraine.fr

Z. Bidai
Laboratory of Research in Industrial Computing and Networking, LRIIR
Oran University, BP 1524 El M'naouer, Algeria
E-mail: bidai.zahia@univ-oran.dz

1 Introduction

Since wireless sensor networks (WSN) are infrastructureless, sensor nodes have to cooperate as in ad hoc networks to perform data routing. In established networks such as the internet, this task is achieved through the use of relatively small routing tables thanks to the IP hierarchical addressing scheme. In WSNs, such a routing model is unfeasible due to their dynamic nature and big number of involved nodes rather randomly deployed. The complete routing table at each node would have a size of $O(N)$ where N is the number of nodes in the network. This makes such table-based routing protocols unscalable to large networks since sensors have very limited memory space. Reactive routing protocols such as AODV [12] provide a partial solution to this problem by building routing tables on-demand. However, they consume much communication bandwidth due to the flooding mechanism of their control messages. Schemes such as OLSR [13] maintain routes proactively in order to reduce message latency but have similar memory requirements as AODV. DSR [20] does not require the maintenance of any routing table but still be inefficient in terms of bandwidth since it also makes use of flooding. Geographic routing appears to be more memory efficient since it only needs $O(1)$ state per node [21]. However additional hardware or localisation algorithms which are resource-hungry are required. Moreover, many other issues related to real-world environments still an open research question in geographic routing such as the presence of holes and obstacles.

Compact routing [25] is another alternative technique for routing where a reduced routing table can be used provided that nodes are appropriately labelled. Interval routing is one of the most important compact routing that first introduced by Santoro & Khatib [35]

for tree networks and subsequently extended by Leeuwen & Tan [24] to other network topologies. Interval routing requires tables of size $O(d)$ where d is the degree of a node. Each link (channel or neighbour) is associated with an interval of integers. The link is selected if the destination address is within the interval. More recently, hierarchical routing that relies on maintaining a hierarchy in the network topology as an inevitable solution to the problem of scalability in WSN. A hierarchical topology can be achieved through building a spanning tree or by applying a clustering algorithm [29,5]. The two techniques may be combined by building a spanning tree using only the cluster-heads in a cluster-based topology [7].

When endowing an already built hierarchy with an appropriate labelling scheme as those used in compact routing, an efficient data gathering can be performed. In fact, it has been proven that using a Depth-First Scheme (DFS) to label nodes in the tree (resulting in an interval routing) allows an optimum scheme [24], that is, it allows to find the shortest path (in terms of number of traversed links) to the tree root. However, interval tree routing often provides fragile routing paths and is hardly applicable in dynamic networks. A single hop failure may result in recomputing all the assigned labels in the tree. To overcome the poor adaptability of the classical interval routing to dynamic changes in the network topology, one can make use of a different compact routing scheme called *prefix routing* [6]. Prefix routing is more robust to topology changes since precedence information is explicitly available. The tree root is labelled by the empty string. Each node in the tree is labelled by a string such that a parent label is a substring of each of its children labels. A node is selected as the next hop if its label has the longest matching prefix of the destination node label.

In this paper, we are concerned with the construction of multiple paths from one sensor to the collect station or the sink. Multipath routing allows fault tolerance when additional paths are maintained to serve as backup on primary path failure. When paths are used simultaneously, it allows more bandwidth and more balanced traffic in the network. Consequently, high data rate applications such as multimedia wireless sensor networks (MWSN) can be handled more efficiently while increasing the network lifetime. We propose MPR (Multipath Prefix Routing) protocol that builds and maintains multiple disjoint paths from one sensor to the sink. MPR relies on an already built spanning tree rooted at the sink and assumes that a prefix labelling scheme is available along with this tree.

MPR is both proactive and reactive multipath routing protocol. It is proactive since at least one path can be inferred from the tree structure. It is reactive since if required, it will build additional paths based on a very light discovery process. From the one hand, heavy control traffic inherent to reactive routing is reduced and on the other hand, overhead due to routing tables exchange is avoided as it is the case in proactive routing. Moreover, the use of prefix tree routing allows reducing the size of routing tables in the sensor nodes in addition to help in ensuring paths disjointness.

This paper is organised as follows. The related work is summarised in Section 2. In Section 3, our network model is described along with some definitions and assumptions. Section 4 gives the details of our multipath routing protocol. Two labelling schemes for practical implementation are proposed and evaluated in Section 5. Section 6 applies our proposal to the specific case of the ZigBee Standard and Section 7 reports on the obtained results based on simulations preformed to get more insight on the behaviour of our proposed multipath routing in the context of the ZigBee technology. Section 8 concludes the paper and gives some future directions.

2 Related Work

Routing is the process by which a node decides on which link a message has to be sent to reach its destination. In order to determine on which link a packet has to be sent, a routing protocol makes use of the information available in a routing table (or state) maintained at router nodes. Due to their limited bandwidth and memory, routing in WSN introduces new challenges compared to traditional networks. Thus, routing protocols with small state and reduced control traffic are required. With respect to these criteria, traditional routing protocols such as AODV [12], OLSR [13] and DSR [20] do not scale well to large networks. As a result, researchers focused on geographic, hierarchical and more recently compact routing techniques.

Geographic or location-based routing protocols make use of location information to limit the route discovery flooding to a geographic area around the destination [42] or to guide packet forwarding through a simple greedy forwarding [21]. In this latter, each node forwards the packet to a neighbour closer to the destination than itself, until ultimately the packet reaches the destination. Geographic routing needs only $O(1)$ state per node however additional hardware or localisation algorithms which are resource-hungry are required. Issues related to real-world environments still an open research question in geographic routing such as the presence of holes and obstacles. More recently, authors of [36] use some implicit geometric properties of a WSN to

reduce the average routing table size by storing selective routing paths. Based on an approximate distance oracle, they propose a distributed routing algorithm and show that the obtained path stretch is close to optimal.

As opposed to flat routing, hierarchical routing emerged as an inevitable solution to the problem of scalability in WSN. Flat routing protocols are quite effective in relatively small networks. However, they scale very bad to large and dense networks since, typically, all nodes are alive and generate more processing and bandwidth usage. On the other hand, hierarchical routing protocols have shown to be more scalable and energy-aware in the context of WSNs. In hierarchical-based routing, nodes play different roles in the network and typically are organised into clusters [29,5].

Compact routing [25] is another alternative technique to reduced routing state. Interval routing is one of the most important compact routing that first introduced by Santoro & Khatib [35] for tree networks and subsequently extended by Leeuwen & Tan [24] to other network topologies. Interval routing requires tables of size $O(d)$ where d is the degree of a node. However, interval tree routing often provides fragile routing paths and is hardly applicable in dynamic networks. A single hop failure may result in recomputing all the assigned labels in the tree. This motivated prefix routing schemes [6].

S4 protocol (small state and small stretch) proposed in [30] adopts compact routing for WSNs. It builds on the work of Thorup and Zwick [38] with some changes to meet the requirement of a practical implementation. It ensures a maximal routing stretch of 3 while using $O(\sqrt{N})$ node state. S4 selects $O(\sqrt{N})$ nodes as beacons. Each node forms a virtual local cluster consisting of nodes whose distances to the present node are within their distances to the closest beacons. A node maintains the shortest-path routes to all nodes within its cluster as well as the shortest-path routes to all the beacon nodes. Thus, to reach a destination node outside its cluster, a source node first routes toward the beacon closest to the destination node. While S4 does well at bounding the maximal routing stretch, it requires $O(\sqrt{N})$ node state and maintenance traffic, which may be significant for very large networks and for some constrained sensor node platforms.

Based on proposed hierarchical routing protocols, authors of [19] developed a framework called HR (Hierarchical Routing) that captures the common characteristics of these protocols. HR uses a kind of prefix routing on top of a multi-level clustered topology where the hierarchy is built in a bottom-up approach (from leaves to the root). They evaluate the implementation of their framework in TOSSIM and on a 60-node test-bed. They demonstrate that from the practical perspective HR can offer low routing stretch despite only logarithmic routing state.

As apposed to S4 and HR, in our work we aim to build multiple disjoint paths to the sink rather than one path to other nodes in the network. Multipath routing protocols enable a source node to discover several paths towards the destination. Traditionally, multipath routing was targeted to failure tolerance. Additional paths are maintained to serve as backup on primary path failure [15,17]. The multiple discovered path can also be used to concurrently transmit data to allow better reliability [44,14], more bandwidth [28], reduced end-to-end delay [18], load balancing [23,27] or higher network lifetime [31]. As for MPR, [26] proposed to build multiple paths based on a spanning tree rooted at the sink. It differs from MPR in the fact that paths discovery is proactive and sink-initiated. The first phase consists in building a spanning tree while discovering alternate paths. The second phase allows the discovery of additional paths through the propagation of RALT messages initiated at each node where an alternate disjoint path is found during the first phase.

With respect to the ZigBee standard, there is no much research on routing dedicated to the ZigBee network layer. The focus was only on AODV and tree routing defined in the ZigBee standard. The aim of most previous work was to improve and enhance the tree routing for ZigBee cluster-tree and mesh networks. In fact, tree paths could be longer because the data packets follow the hierarchical tree topology to the destination even if the destination is located nearly. Many works proposed to make shortcuts to find paths with less hop count than the tree path. This is mainly done through the use of neighbours tables as in [22] and [37]. Enhanced Hierarchical Routing Protocol (EHRP) proposed in [16] uses network addresses in addition to neighbours information to take shortcuts without incurring extra overhead. Authors of [39] proposed a new Self-Learning Routing (SLR) mechanism which is more efficient than EHRP. SLR inherits the low overhead of tree routing and the path efficiency of mesh routing since it does not send route discovery packets. It routes packets solely based on network addresses and overhearing. Enhanced Tree Routing [34] intended to achieve certain balance between performance and cost by improving the hop-counts of the ZigBee Tree Routing with minor additional complexity. In addition to neighbours table, authors utilises the structured address assignment scheme of the ZigBee standard.

In our previous work [9,11], we proposed to utilise shortcuts to build multiple paths to the sink in the ZigBee context. In this work, we present a general formu-

lation of a multipath routing protocol based on prefix-routing. We, additionally, introduce a light discovery phase to avoid problems related to the on-the-fly routing process adopted in [9,11]. Finally, in this work, we propose and assess the overhead that can be introduced by two different labelling schemes.

3 Network Model

3.1 Assumptions and Definitions

We model our sensor network as a graph $G(V, E)$ composed of a set of vertices (sensor nodes) V and a set of edges (links) E. We assume that G is an undirected graph. That is edges (u, v) and (v, u) are the same and implies that u and v are in the radio transmission range of each other[1]. Without loss of generality, we further assume that G is connected, i.e., there is a path between every pair of nodes in this graph. Let $T(V, E_T)$ be the spanning tree of the connected undirected graph G rooted at a specific vertex $r \in V$ referred to as the *tree root* and corresponds to the sensor network *sink*. In order to build the spanning tree, a *building-up Method* can be adopted : one edge from G is selected at a time such that no cycles are created. This is the case of Berkeley's TinyOS sensor platform [26] where a flooding-based beaconing protocol is used to build and maintain a spanning tree rooted at the sink. Besides, the ZigBee network layer [4] is able to build and maintain a tree structure in addition to star and mesh topologies.

We use $P(u \rightarrow v)$ to denote a path from node u to node v and we can write $P = (u, w_1, w_2, \ldots, w_{n-1}, w_n, v)$ to explicit P's nodes. When considering the particular destination r (the root of the tree), we simply use $P(u)$ to designate the path from node u to the tree root. We also use $P_T(u)$ to designate the path from a node u to the tree root using only links in the spanning tree T. This path is unique[2] and called interchangeably, the *child-parent* or the *tree* path of node u. Moreover, we define a concatenation operator "$\rightarrow$" to designate the operation that allows to concatenate two paths P and P' so that $P \rightarrow P'$ designates the path where the last vertex of P is connected to the first vertex of P'.

The parent of a node v in T is noted $\pi(v)$ and a *descendant* u of a node v is a node such that v is in the tree path (using only tree links) from u to the root of T. We say also that v is an *ancestor* of u. Two children of the same node are called *siblings*. We refer to as a *subtree* rooted at node $t \in T$ the tree consisting of this node t and all of its descendants. The subtree corresponding to the root node r is the entire tree T. We define the *depth* of a node as the length (number of hops) of its tree path to the root. The tree root has a depth 0 and a non-root node has a non-zero depth which equals its parent's depth plus one. The height of the tree H is defined as the maximum depth among all the tree nodes. We also use the term *degree* of a node to designate the total number of its adjacent links or neighbours and use $dg(u)$ to denote the degree of node u in G.

In WSN, it is very difficult to build a global addressing scheme. Some structured addressing schemes were proposed in the literature [4,32,8]. In this work, we make use of the spanning tree with a labelling scheme as the one used in prefix routing that allows reducing the size of routing tables [6,40]. Each node is labelled by a string from an alphabet $\Sigma = \{s_i, i \in [1, W], W \geq 2\}$ where W is the tree width which is the maximum number of children a parent can have in T. The tree is then said to be a *W-ary tree*. Moreover, if the root and each internal node in the tree has either 0 or W children, the tree is said to be a *full W-ary*. A *perfect W-ary tree* is a full W-ary tree in which all leaf nodes are at the same depth.

The root is labelled by the empty string λ which is by definition the prefix of all strings built from Σ. For a parent with label L, the k^{th} child is labelled by $L \| s_k$ where $\|$ is a concatenation operator. In what follows, the assigned label to a node u located at depth d_u in T is noted $l(u)$. We define $prefix_k(u)$ to be the first k symbols of the string $l(u)$. For short, we use $prefix$ instead of $prefix_1$. We also define $prefix(u, v)$ as the substring that corresponds to the first similar symbols in u and v. If $u = v$ then $prefix(u, v) = l(u) = l(v)$, otherwise $w = prefix(u, v)$ if $\exists w \in \{s_i\}^*, \exists u', v' \in \{s_i\}^+ : u = wu' \wedge v = wv' \wedge prefix(u') \neq prefix(v')$. w can be empty then we write $prefix(u, v) = \lambda$. Table 1 summarises main notations used throughout this paper.

3.2 One-Path Prefix Routing

Prefix routing is more resilient to dynamic changes in the network. A node can join the tree and get a label from Σ that is not already used by one of the children of its parent. The prefix routing still operate correctly as opposed to interval routing. In the latter, labels have to be recomputed to handle such changes in the tree topology. In prefix routing (summarised by algorithm 1), when a node c receives a message originated from s

[1] This assumption is a necessary condition to be able to use reverse paths.

[2] paths in trees are unique. If we assume that another path exists, this would lead to a cycle in T which is in contradiction to tree definition.

Notation	Description
$G(V, E)$	network graph.
$T(V, E_T)$	the spanning tree of G rooted at sensor network sink r.
$P(u \to v)$	path from u to v.
$P(u)$	path from node u to the tree root.
$P_T(u)$	path from node u to the tree root using only links in the spanning tree T.
$P \to P'$	the path that results from concatenating P and P' in this order.
$\pi(v)$	v's parent.
$dg(u)$	u's degree in G.
d_u	u's depth in T.
$l(u)$	u's label.
$prefix_k(u)$	the first k symbols of the string $l(u)$; $prefix(u) = prefix_1(u)$.
$prefix(u, v)$	the first similar symbols in u and v.
H	tree height, $H \geq 2$.
W	tree width, $W \geq 2$.
$\mathcal{T}_u$	The MPR subtree rooted at u where $\pi(u) = r$.

Table 1 Notations

to be routed to node d ($d \neq c$), it makes the following routing decision :

If d is downstream then forward to child x with the longest prefix(d,x) else go upstream until prefix(s,d), the first common ancestor of nodes s and d is reached.

Note that prefix routing is a kind of source routing where the path to the tree root from node u is given by its label $l(u)$. Moreover shortcuts are possible as proposed in [40] to improve its optimality in terms of number of traversed nodes when sensor-to-sensor communication is needed. It is worth mentioning that prefix routing between the sink and any sensor does not require these shortcuts.

Algorithm 1: Prefix Routing Algorithm (PR)

Input: This node's id (c) , destination id(d)
Output: The next hop node (x)

```
if d == c then
    return
end
if c is an ancestor of d then
    x is the child with the longest prefix(d,x)
else
    x = π(c)
end
```

4 Multipath Prefix Routing Protocol (MPR)

In MPR, we are concerned with building multiple disjoint paths from one node in the sensor network to the sink. MPR relies on an already built spanning tree $T(V, E_T)$ rooted at the sink $r \in V$ and assumes that a labelling scheme as the one introduced in Section 3.1 is available. The tree is assumed to have both width and height of at least 2. In fact the construction of more than one path is of limited interest since the tree is reduced to a star topology when $H = 1$ and a linear topology when $W = 1$.

At any time, a sensor is assumed to maintain an up-to-date neighbours table. This latter can be built upon initialisation of the network and updated periodically to adapt to network topology changes. Without loss of generality and for seek of clarity, we consider the case of one source s with depth d_s in T willing to transmit data via multiple paths to the sink (r) which is not directly reachable from s ; Otherwise, MPR considers the obvious path where s transmits directly to r.

MPR adopts a hybrid approach to build and maintain its paths. From the one hand, it is a proactive routing protocol since at least one path can be inferred from the tree structure. One path that can be used without any discovery process is $P_T(s)$, the tree path from s to r. On the other hand, MPR is a reactive routing protocol since, if required, it builds additional paths based on a very light discovery process based on the tree structure, its labelling and neighbours tables. Prior to giving details about the paths discovery mechanism, we give the following definitions and lemmas :

Definition 1 (Paths disjointness) Two paths $P(a)$ and $P(a')$ from two different nodes a and a' respectively to the tree root r are said to be disjoint if their only common node is r. If the two paths have the same source $s = a = a'$ then there are two common nodes s and r.

Definition 2 (MPRS) An MPR subtree (MPRS) of T is a subtree rooted at one of the child nodes of the sink r. An MPRS rooted at u is noted $\mathcal{T}_u$ and is said to be *busy* if its root (u) belongs to an already established path.

Lemma 1 *An MPRS of T rooted at u is uniquely identified by one symbol from Σ which is the u's label. We have $\forall v \in \mathcal{T}_u \quad : \quad prefix(v) = prefix(u) = l(u)$*

Proof (see appendix A.1)

Definition 3 (MPR-neighbour) Neighbours u and v are said to be *MPR-neighbours* if and only if each of them belongs to a different MPRS. We say also that u is an *MPR-neighbour* of node v and vice versa.

Lemma 2 *Assume two neighbour nodes u and v. If $prefix(u) \neq prefix(v)$ or equivalently $prefix(u, v) = \lambda$ then nodes u and v are* MPR-neighbours.

Proof (see appendix A.2)

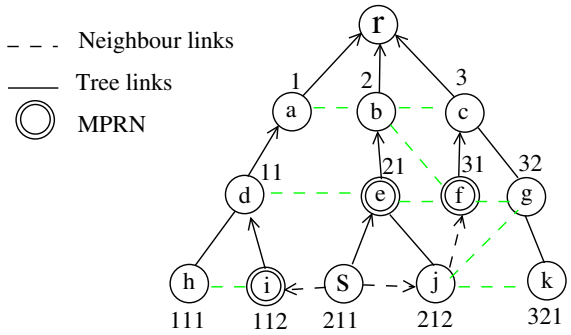


Fig. 1 Illustrative MPR Example.

Definition 4 (MPRN) The MPR-node of a given path (from s to r) is the first node in this path that belongs to a *non busy* MPRS. The MPRN of the tree path from a source s is its own parent $\pi(s)$.

Basically, a built path $\mathcal{P}$ in MPR is the concatenation of a discovered path $P(s \rightarrow m)$ and a tree path $P_T(m)$ where m is the MPRN of this path. The idea behind MPR is to discover the first portion of the path then reuse an already established tree path. The discovery phase is detailed in Section 4.2. At this stage, it is worth noting that in addition to the tree path $P_T(s)$ and thanks to the MPRN notion, MPR is able to use proactively the path $(s) \rightarrow P_T(w)$ if there is a node w that is an MPR-neighbour of s.

Figure 1 illustrates MPR on a simple topology where solid and dashed lines represent respectively the tree links (E_T) and the neighbouring links. The nodes are labelled using strings from the alphabet $\Sigma = \{1, 2, 3\}$. Paths used by the source s to forward data to the sink r are represented using arrowed lines. In this example, two paths can be used immediately, the tree path $\mathcal{P}_1 = P_T(s)$ and the path $\mathcal{P}_2 = (s) \rightarrow P_T(i)$ with i as the MPRN. In fact, i is an MPR-neighbour of s since they belong to two different MPRSs and the i's MPRS is not busy. The third path $\mathcal{P}_3 = (s, j) \rightarrow P_T(f)$ requires a discovery phase to build its first portion (s, j) since j belongs to the same MPRS as s. Based on definition 4, nodes e, i and f are respectively the MPRN of paths $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$.

4.1 Paths Disjointness

As stated before and thanks to the proactive property of MPR, the source is able to immediately start transmitting data on at least one path. Whenever the application requires more paths, a light discovery process can be initiated by the source using explore messages.

The labelling scheme is of great importance in achieving a light discovery process in MPR. Labelling contributes in ensuring that two paths are disjoint without a complete discovery as it is the case of reactive routing protocols where request messages have to be forwarded from a node to another until achieving the final destination. The following theorem sets the disjointness criterion between two tree paths :

Theorem 1 *Two tree paths $P_T(a)$ and $P_T(a')$ where $a \neq a'$ are* node-disjoint *if and only if their respective nodes (except the root) belong to a different MPRS.*

Proof (see appendix A.3)

Corollary 1 *Two tree paths $P_T(a)$ and $P_T(a')$ are node disjoint if and only if $prefix(a) \neq prefix(a')$ or equivalently $prefix(a, a') = \lambda$.*

Proof The proof of this corollary is straightforward using Lemma 1 and Theorem 1.

Corollary 2 *Each MPRS contains only one disjoint path to the root.*

Proof (see appendix A.4)

4.2 Paths Discovery and Data Transmission

In order to build the first portion of a path in MPR, a lightweight discovery process is initiated by the source using explore messages. An explore message (*ExploreMsg*) is forwarded from a *candidate* node to another based on specific rules using unicast. A *candidate* node is a node that may lead to the root via a disjoint path. In Section 4.4 will be discussed the case where multiple candidate nodes are available. In order to ensure paths disjointness, an *ExploreMsg* carries a list $\mathcal{ML}$ that contains the prefix of the currently used MPRSs. Two cases are possible :

- The explore message reaches an MPRN which means that a disjoint path exists. This path is the one followed by the *ExploreMsg* augmented by the tree path of the MPRN to the root. The MPRN sends a Response message (*ResponseMsg*) with the prefix of the MPRS it belongs to back to the source using the *ExploreMsg* reverse path. Upon the reception of the response message, the source updates the list $\mathcal{ML}$ accordingly and if required by the application begins forwarding data packets via the neighbour on which the discovery was initiated. A node determines whether it is an MPRN for the path followed by an *ExploreMsg* simply by checking the absence of its prefix in the $\mathcal{ML}$ carried by the *ExploreMsg*. If $prefix(c) \notin \mathcal{ML}$ then c is an MPRN.

– The explore message arrives at a node from which there is no forwarding candidates. This node sends an *ErrorMsg* on the explore message reverse path toward the source. Upon the reception of the *ErrorMsg*, this node neighbour marks it as a *non-candidate* node and tries to discover a new path by sending an *ExploreMsg* to a candidate node if available; otherwise it forwards the *ErrorMsg* back to the source. Likewise, the source marks the neighbour from which it receives an *ErrorMsg* as a *non-candidate*. A neighbour node is marked as a *non-candidate* for a given duration and at least for this discovery phase.

In both cases, if another candidate is available then the source can initiate another discovery phase mainly if the required number of paths is not achieved. In order to avoid infinite wait, the source triggers a new path discovery by issuing an *ExploreMsg* toward another candidate if no *ResponseMsg* is received within a determined duration. Note that the discovery process can be stopped as soon as the maximum possible number of disjoint paths is reached. Indeed, the number of node-disjoint paths is dictated by the network connectivity. It is limited by the minimum number of neighbours among the source and sink nodes. We can at most build $min(dg(r), dg(s))$ paths. The source stops sending explore messages as soon as the number of built paths achieves $dg(s)$ the value of its own degree in G. One can implement a mechanism that allows the source to know the degree of the sink so it can stop earlier the discovery process if $dg(r) < dg(s)$. Note also that marking nodes as *non-candidate* allows limiting the number of sent explore messages.

Algorithm 2 summarises the main operations related to the paths discovery and maintenance (see Section 4.3). The source records the available paths using a list of forwarding nodes and their corresponding MPRN nodes (only using their prefixes). An intermediate node records in its memory the identity of the next node to which send a received data packet.

In Figure 1, the third path $\mathcal{P}_3$ is discovered as follows. An *ExploreMsg* is sent to node j since it is the only available candidate (node i is already in use by path $\mathcal{P}_2$). Upon the reception of the *ExploreMsg*, j has to choose among candidate nodes f, g and k. Here, we assume that we take the decision to rather choose an MPR-neighbour which allows to find immediately a disjoint path if the corresponding MPRS is not busy. If two MPR-neighbours are available then the one with the smallest depth is chosen. In this latter case, shortest paths are privileged. Applying these two rules leads to choose node f as the next recipient of the *ExploreMsg*. Other rules can be applied in order to get paths with different properties and/or based on other metrics than the number of hops. For instance the path $(s, j, k) \rightarrow P_T(g)$ for which node g is the MPRN allows to get a longer path which nevertheless presents less interference, with the tree path $\mathcal{P}_1$, than the path $(s, j) \rightarrow P_T(f)$ with f as MPRN. The maximum number of paths is achieved since $dg(s) = 3$. The source stops sending explore messages and the resulting forwarding nodes list is $\{e, i, j\}$ and $\mathcal{ML} = \{2, 1, 3\}$.

4.3 Paths Maintenance

MPR functioning builds on a tree structure in addition to up-to-date neighbours tables which both can be affected by network dynamics mainly due to node or link failure. Neighbours tables can be maintained using periodic exchange of hello messages. In order to maintain the tree structure, orphan nodes that occur after a parent node is dead for instance, have to be able to join a new parent in the vicinity. Association and disassociation procedures as those provided by the IEEE 802.15.4 MAC sub-layer [3] can be adopted to build and maintain the tree structure. The prefix labelling is updated accordingly such that a node that joins a new parent gets the label of its parent plus its sibling rank. The prefix routing still operate correctly as opposed to interval routing where labels have to be recomputed to handle such changes in the tree topology.

Hello messages allow also to detect whether the next node in a path is still alive for routing. Moreover, link or node failure in a path can be detected in the absence of an ACK for a data packet after a certain number of retries if an ACK-based transmission is used. When an intermediate node detects a link failure, it issues a failure message (*FailureMsg*) back to the source. On the reception of this message, the source and the intermediate nodes stop sending data packets. *FailureMsg* may include in its header the sequence number of the last sent data packet so the source can retransmit it if full reliability is required by the application. The source may also redistribute the failed path traffic on the other paths. Meanwhile, the node that detects the failure issues an *ExploreMsg* if any candidate node is available in the vicinity ; otherwise it triggers an *ErrorMsg* back to the source.

4.4 Multiple Candidate Nodes

When a node is willing to transmit an explore message, multiple candidates may be illegible to be the next node in the path. The choice between multiple candidates can be done randomly or obey to some rules dictated by

Algorithm 2: - MPR Paths Discovery and Maintenance

Data: This node's id (c), the list of MPRSs already in use maintained by the source ($\mathcal{ML}$)

***ExploreMsg* packet processing (intermediate node)**

Mark the last crossed node as the *previous node*
if $prefix(c) \notin ExploreMsg.\mathcal{ML}$ **then** `// I am an MPRN`
  Send a *ResponseMsg* packet on the *ExploreMsg* reverse path with *ResponseMsg*.MPRS $= prefix(c)$
else if *no candidate node* **then**
  Send an *ErrorMsg* on the *ExploreMsg* reverse path
else `// there is a candidate node`
  Forward the *ExploreMsg* packet to a candidate node
end

***ResponseMsg* packet processing**

if *I am the source* **then**
  Add the last crossed node to the list of the forwarding nodes and add *ResponseMsg*.MPRS to $\mathcal{ML}$
  Start, if required, forwarding data packets via the corresponding neighbour
  Send, if required, another *ExploreMsg* if another candidate is available
else
  `// intermediate node`
  Mark the last crossed node as the *next node* in the currently built path
  Forward on the *ExploreMsg* reverse path
end

***ErrorMsg* packet processing**

if *I am the the first crossed node* **then**
  Mark the last crossed node as a *non-candidate* node
end
if *no candidate node* **then**
  Send an *ErrorMsg* on the *ExploreMsg* reverse path
else
  `// there is a candidate node`
  Forward the *ExploreMsg* packet to a candidate node
end

***FailureMsg* packet processing**

Stop sending data packets
if *I am the source* **then**
  Start data transmission on a backup path if any
else
  `// intermediate node`
  Forward *FailureMsg* toward the source on the reverse path
end

the application requirements. These rules can make use of the relationship type between the current node and its potential candidates. We define the following kinds of relationship between neighbours and show how the prefix labelling allows a given node to determine such relationships :

- **Parent-child relationship.** Node u is node v's parent iff : $\exists s_i \in \Sigma : l(v) = l(u)||s_i$ or equivalently $prefix(u, v) = l(u)$.
- **MPR-neighbours.** Nodes u and v are said to be MPR-neighbours iff (as already stated in lemma 2) : $prefix(u) \neq prefix(v)$ or $prefix(u, v) = \lambda$.
- **Siblings.** Nodes u and v are siblings iff : $d_u = d_v = d \wedge prefix_{d-1}(u) = prefix_{d-1}(v)$ or equivalently $d_u = d_v = d \wedge |prefix(u, v)| = d - 1$. We have $prefix(u, v) = l(\pi(u)) = l(\pi(v))$
- **Cousin relationship.** Two nodes are cousin if they belong to the same MPRS but are not siblings and none is in the parent-child path of the other i.e. : $prefix(u) = prefix(v) \wedge (prefix_2(u) \neq prefix_2(v) \vee (d_u \neq d_v \wedge prefix(u, v) \neq l(u) \wedge prefix(u, v) \neq l(v)))$ We can distinguish further a close cousin from a distant cousin. Two cousins u and v are distant iff $prefix_2(u) \neq prefix_2(v)$ and close otherwise.

To speedup the discovery process and potentially increase the number of discovered paths, we can introduce other relationship types :

- **Bordering neighbour.** A bordering neighbour is a neighbour which is connected to another MPRS. Bordering neighbour of v is a neighbour u that has an MPR-neighbour that is not in the MPRS of both u and v. This kind of relationship requires two-hop neighbourhood information but allows speeding up the discovery process. This is the case for instance of node j in Figure 1 which is a bordering neighbour of s since it has an MPR-neighbour f.
- **Sink neighbour.** This is the case of a neighbour that is itself neighbour of the sink.

Finally, one can adopt other rules to forward explore messages :

- If two or more candidates have the same relationship then choose the one with the least depth which allows going faster upstream ;
- we can choose to avoid forwarding an explore message to a non child descendant and similarly a non parent ancestor ;
- one can choose to adopt a more conservative approach where it never forwards an explore message to a node with higher depth to avoid going downstream and forming loops.

4.5 Neighbours Table

Building the neighbours table is performed using periodic hello messages that contain in addition to the MAC layer address, the label of their originator. In order to maintain the memory requirements as low as possible, we propose to include the relationship type rather than the whole neighbour label in the neighbours table. That is, each entry includes the following fields :

- Neighbour's *MAC layer address*, which allows physical communication with the neighbour ;
- Neighbour *relationship type*, a field computed using the labels of this node and its neighbours. It includes all or a part of the relationship types defined in Section 4.4.
- Neighbour *state*, which gives information on whether a neighbour is the next or the previous node in a path (if a path is established) and if it is a candidate node or not otherwise. A neighbour that is already used by another path can be considered as a non candidate.

In addition to the MAC layer address, we only introduce 5 extra bits to implement our protocol. 2 bits are sufficient to encode the state and at most 3 bits are required to encode the different relationship types. Note that the absence of the whole neighbour label in the chosen entry structure does not allow some optimisations such as routing on neighbour's depth. One can choose to store the complete label of neighbours if optimal path discovery is privileged against memory utilisation.

5 MPR Prefix Labelling Analysis

In order to implement MPR prefix labelling, we propose to use either a fixed-length or a variable-length labelling schemes. In the former, all nodes' labels are of the same length whereas in the latter, a node's label length depends on its depth in the tree. In both schemes, an alphabet Σ composed of $W+1$ symbols is required for a W-ary tree where each node has at most W children ($W \geq 2$). In a fixed-length labelling scheme (FLS), one more symbol is required to complete the label so all labels are composed of H symbols. In a variable-length labelling scheme (VLS), the additional symbol indicates the end of the label. Let $\Sigma = \{0, 1, 2, \ldots, W\}$ and $\Sigma^* = \Sigma - \{0\}$.

Definition 5 (FLS) In fixed-length labelling scheme (FLS), a node $u \in V$ located at depth $d > 0$ of the spanning tree T is labelled with $l(u)$:

$$l(u) = s_1||s_2||\ldots s_d||0\ldots 0 \quad (1)$$

where $s_i \in \Sigma^*$ gives u's sibling rank. $(H-d)$ 0s are added to obtain an H-symbol label. The root gets the label composed of H null symbols.

Definition 6 (VLS) In variable-length labelling scheme (VLS), a node $u \in V$ located at depth $d > 0$ of the spanning tree T is labelled with $l(u)$:

$$l(u) = \begin{cases} s_1||s_2||\ldots s_d||0 & \text{if } d < H \\ s_1||s_2||\ldots s_{H-1}||s_H & \text{if } d = H \end{cases} \quad (2)$$

The root gets the label composed of one null symbol 0.

In order to evaluate the previously defined labelling schemes, we give in the following, the definition of a flat addressing scheme :

Definition 7 (FS) In a flat addressing scheme (FS), a node $u \in V$ is assigned a unique address from $[0\,..\,\lceil log|V|\rceil]$ where $|V|$ is the number of nodes in the tree T.

Theorem 2 *In a perfect W-ary tree ($W \geq 2$), both FLS and VLS have a maximum label size of $H\lceil log(W+1)\rceil$ with at most $H-1$ additional bits with respect to the FS addressing scheme.*

Proof (see appendix A.5)

Corollary 3 *In a perfect W-ary tree ($W \geq 2$), an FLS or a VLS label size is at most twice the size of an FS address.*

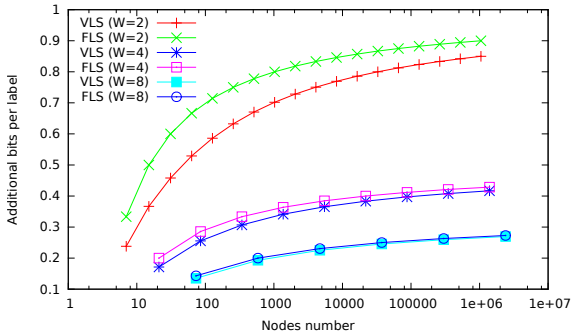
Proof (see appendix A.6)

Let $E[B_{FLS}]$ and $E[B_{VLS}]$ be the mean number of bits required by FLS and VLS respectively. In a perfect W-ary Tree, we have :

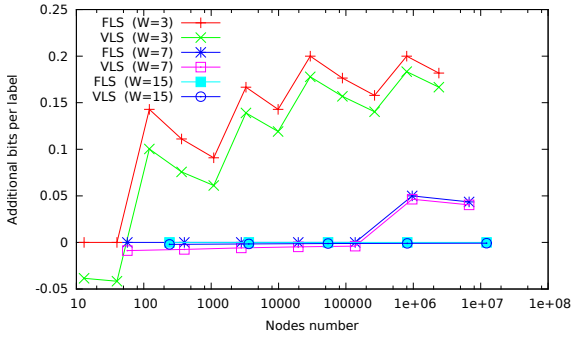
$$E[B_{FLS}] = H.w = H.\lceil log(W+1)\rceil \quad (3)$$

$$E[B_{VLS}] = w.\frac{\sum_{d=0}^{H}(d+1)W^d}{|V|} \quad (4)$$

Figure 2 plots, as the number of nodes increases in a perfect W-ary Tree, the ratio of additional bits induced in average by the labelling schemes VLS and FLS with respect to FS. This ratio for a labelling scheme $LS \in \{VLS, FLS\}$, is $(E[B_{LS}] - \lceil log|V|\rceil)/\lceil log|V|\rceil$. For a given number w of available bits to encode a symbol in Σ, we consider for FLS and VLS the worst case where $W = 2^{w-1}$ (Figure 2(a)) and the best case where $W = 2^w - 1$ (Figure 2(b)). In order to generate our data, we varied the tree height until a minimum number of nodes is achieved (here one million).



(a) Worst cases



(b) Best cases

Fig. 2 Ratio of additional number of bits in VLS and FLS with respect to FS in a perfect W-ary Tree.

We can see that when the network size (and equivalently the tree height) increases, the ratio of additional bits increases without exceeding 1 which confirms corollary 3. We note that when the tree width (W) increases, the ratio of additional bits decreases and that VLS and FLS become equivalent as it is the case when $W = 8$. They almost require the same number of bits as FS addressing scheme. When the tree width is small, say 2 to 4, VLS performs better than FLS in terms of additional bits.

When the tree width is well chosen (Figure 2(b)), VLS and FLS induce a very limited extra bits with respect to FS. In some cases, VLS may present a smaller number of bits than FS as for $W = 7$ and a number of nodes less than $137,257$. Choosing between VLS and FLS is dictated by the tree parameters. VLS appears to be more appropriate for long thin trees (big H and small W) while FLS must be adopted when the tree width exceeds a given threshold (say 7) since it is additionally easier to be handled mainly in packets headers.

In what follows, we consider the case of a full W-ary tree ($W \geq 2$) with R ($R > 1$) [3] nodes (routers) at each level having W children and the remaining $(W - R)$ (end devices) having no children. The number of nodes at level $d > 0$ is $W.R^{d-1}$ which gives an overall number of nodes :

$$|V| = 1 + W \sum_{d=1}^{H} R^{d-1} = \frac{WR^H - W - 1 + R}{R - 1} \quad (5)$$

The number of bits required in FS is $\lceil log(|V|) \rceil$:

$$E[B_{FS}] = \lceil log(\frac{WR^H - W - 1 + R}{R - 1}) \rceil \quad (6)$$

The mean size of a label in FLS and VLS are given by :

$$E[B_{FLS}] = H.w = H.\lceil log(W + 1) \rceil \quad (7)$$

$$E[B_{VLS}] = w.\frac{1 + W.H.R^{H-1} + \sum_{d=1}^{H-1} W.R^{d-1}(d + 1)}{|V|} \quad (8)$$

Figures 3 and 4 show the ratio of additional bits induced by FLS and VLS with respect to FS in a full W-ary tree as the number of nodes increases. When the tree width is set then the worst case for FLS and VLS consists in having only two routers per level i.e. $R = 2$. Figure 3 plots the ratio of additional bits when $w = 3$ allowing a tree width in the range $[4, 7]$. The number of routers per level is set to $R = 2$ (worst case) and $R = 4$. In Figure 4, w is set to 4 allowing 8 to 15 children. We always observe that the number of additional bits increases with the number of nodes. Note that FLS and VLS labels size may be 3 times an FS address when $R = 2$. Nevertheless, in the context of WSN an ad hoc network, any node is likely to be a router. Thus, the number of routers per level is more likely to be equal to the tree width which leads to a perfect tree rather than a full tree.

6 MPR Applied to ZigBee

The ZigBee Standard [4] provides the network layer specifications to be implemented on top of the PHY and MAC layers standardised by the IEEE 802.15.4 [3]. The two standards are tightly coupled to provide the consumer standardisation for low-power and low-rate wireless communication devices. The IEEE 802.15.4 MAC layer defines two types of nodes : Reduced Function Devices (RFDs) that can act only as end devices and Fully Functional Devices (FFDs) able to operate as a coordinator or a router.

[3] The case $R = 1$ is not considered since it does not allow building multiple disjoint paths : at least the last link is common.

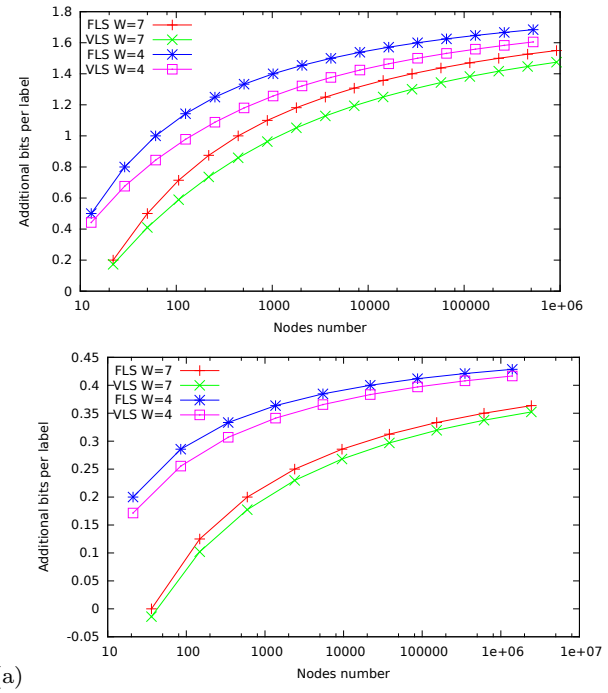


Fig. 3 Ratio of additional number of bits in VLS and FLS with respect to FS in a full W-ary Tree : $w = 3$ and (a) $R = 2$,(b) $R = 4$.

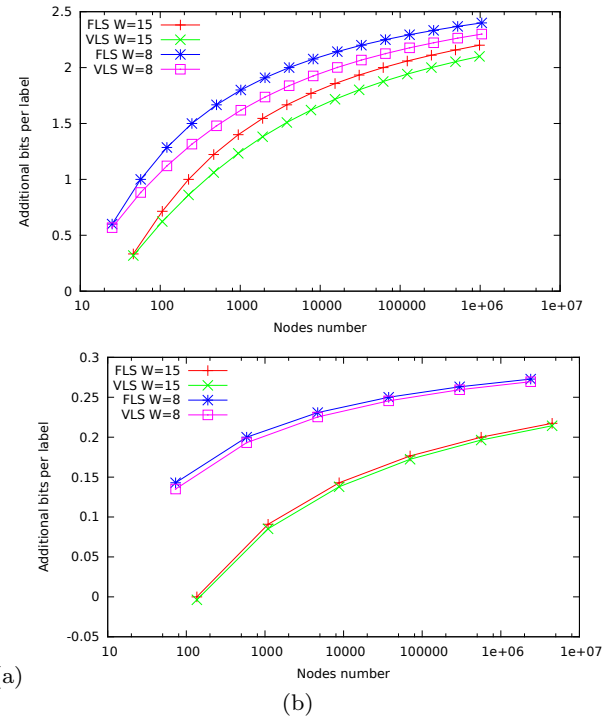


Fig. 4 Ratio of additional number of bits in VLS and FLS with respect to FS in a full W-ary Tree : $w = 4$ and (a) $R = 2$,(b) $R = 8$.

The ZigBee network layer extends the basic star topology of an IEEE 802.15.4 PAN to a tree topology where the root, called ZigBee coordinator (ZC), and all internal nodes called ZigBee routers (ZRs) are FFDs. RFDs can only be leaf nodes and are called ZigBee End Devices (ZEDs). The ZC is responsible for initiating the formation of the tree topology. Parent-child relationships are established when a ZR or a ZED joins the network.

The ZigBee addressing scheme and its tree routing are overviewed before showing how MPR can be applied to ZigBee networks.

6.1 ZigBee Addressing Scheme

The ZigBee network layer specifies a distributed algorithm for address assignment. The ZC fixes three parameters (L_m, C_m, R_m) related to the tree topology : L_m is the maximum depth of the tree, C_m and R_m are respectively the maximum number of children and child routers a parent can have in the tree. Note that $C_m \geq R_m$, the ZC has depth 0 and that devices at depth L_m can only be ZEDs. Based on its depth d, a parent node distributes to each of its child routers an address sub-block of size $Cskip(d)$, $d = 0, 1, ..., (L_m - 1)$ given by :

$$Cskip(d) = \begin{cases} 1 + C_m.(L_m - d - 1) & \text{if } R_m = 1 \\ \dfrac{1 + C_m - R_m - C_m.R_m^{(L_m - d - 1)}}{1 - R_m} & \text{otherwise} \end{cases} \quad (9)$$

The k^{th} router device gets the address :

$$A_k = A_p + (k - 1)\ Cskip(d) + 1\ (1 \leq k \leq R_m) \quad (10)$$

and the n^{th} end devices gets the address :

$$A_n = A_p + R_m\ Cskip(d) + n\ (1 \leq n \leq C_m - R_m) \quad (11)$$

6.2 ZigBee Tree Routing

When the tree topology is adopted, the ZigBee standard provides tree routing (ZTR) which is a kind of interval routing since a given interval of addresses is located downstream one node as a result of the ZigBee addressing scheme presented in the previous section. Routing paths are directly inferred from network addresses based on the parent-child relationships. When a packet is received by a ZR with address A_c and depth d, it decides on whether the next hop node for the destination address A_d is up or down the tree by applying

the routing rules of Algorithm 3. ZTR is simple to implement and is lighter in terms of memory and processing requirements than an AODV-like routing. Thus it is more suitable for the ZigBee limited resources devices.

Algorithm 3: - ZigBee Tree Routing Algorithm (ZTR)

Input: This router depth (d), this router address (A_c), this router's parent address (A_p), destination address (A_d).
Output: The next hop node address (A_x)

if $A_d \in]A_c\ ,\ A_c + Cskip(d-1)[$ **then** `// the destination is downstream`

  if $A_d > A_c + R_m.Cskip(d)$ **then** `// the destination is a child and the packet is directly sent to this child`

    $A_x = A_d$
  else
    `// The packet is forwarded down to an ancestor of the destination node`
    $A_x = A_c + 1 + \lfloor \frac{A_d - A_c - 1}{Cskip(d)} \rfloor .Cskip(d)$
  end
else
  `// Destination is upstream and the packet is forwarded to this node parent`
  $A_x = A_p$
end

In ZTR, communication is limited to parent-child links in the tree. As a consequence, routing paths could be longer because packets follow the hierarchical tree topology to the destination even if the destination is located nearly. Shortcuts are possible to improve interval routing stretch [41,16,22,33,39,34].

ZTR often provides fragile routing paths and is hardly applicable in dynamic networks. A single hop failure producing an orphan node may result in recomputing all the assigned labels in the tree. Rather, in a prefix routing schemes, the orphan nodes gets simply the concatenation of its new parent's label and its new sibling rank. In the following section, we show how a prefix routing label can be assigned to a ZigBee node so the ZTR can be replaced by a prefix tree routing that is more robust to topology changes. Furthermore, prefix routing labels allows the implementation of our multipath routing protocol (MPR) in ZigBee networks.

6.3 ZigBee Multipath Prefix Routing (ZMPR)

The cluster-tree topology is assumed to be rooted at the sink that plays the role of the coordinator. As for MPR, ZMPR considers building multiple paths from one node to the sink. Multiple paths allows either the use of an alternative path if the tree path is broken or their simultaneous use to increase the available bandwidth in the presence of a high data rate reporting. ZMPR is applied on a steady state network where all the devices are well associated to their parent devices. ZMPR discovery and maintenance phases are similar to those of MPR. Prefix labels assignment can be performed in two ways :

- Make advantage of the association procedure specified in the ZigBee standard. The *Association Response Command* can be used to carry the prefix label of the parent node to the joining (child) node.
- Compute the prefix labels from the already ZigBee assigned addresses. For a given node u at depth $d_u \in [1, L_m]$, each element s_k $(1 \leq k \leq d_u)$ of its label can be computed using the following formula proposed by [34] :

$$s_k = \begin{cases} \lfloor \dfrac{A - k - \sum_{i=1}^{k-1} Cskip(i-1)(s_i^c - 1)}{Cskip(k-1)} \rfloor + 1 & \text{if ZR} \\ A - k + 1 + R_M(1 - Cskip(k-1)) & \\ \qquad - \sum_{i=1}^{k-1} (s_i - 1) Cskip(i-1) & \text{if ZD} \end{cases} \tag{12}$$

Where A is the ZigBee network Address of node u and $Cskip(d)$ is given by equation (9).

In the first method, an additional field has to be added in the *Association Response Command*. Its size depends on the largest possible label in the tree. In the second method, some computations are necessary and consumes resources even if they can be optimised. The nature of the network and the application requirements dictate the use of one method rather than an other.

In ZigBee, a neighbours table is maintained so to implement MPR, we need only to extend this table to meet our needs. A ZigBee neighbours table entry already includes the 64-bit MAC address of the neighbour and a relationship type. This latter field can be encoded on one byte and the ZigBee standard specifies only 4 relationship types, namely : parent, child, sibling and none. We propose to use 4 bits to encode other relationship types as those introduced in Section 4.4. The 4 remaining bits can be used to encode the state field. In this way no extra fields are introduced to the ZigBee neighbours table.

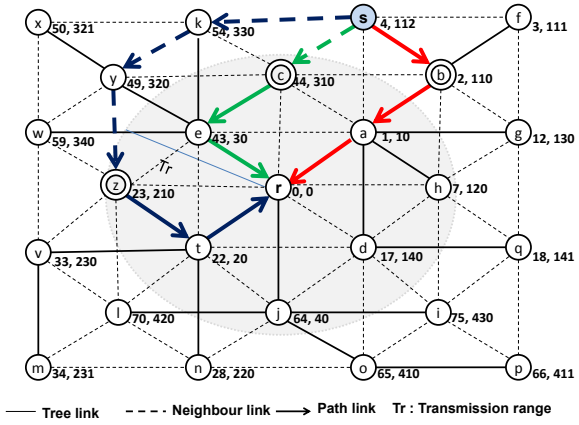


Fig. 5 Rhombic lattice topology.

6.4 Illustrative Example

Figure 5 shows a network topology with parameters $(L_m, C_m, R_m) = (3, 4, 4)$ where all the sensor nodes are evenly distributed on a square-shaped area. The sink is the ZC located at the centre. The transmission range (Tr) chosen in this topology allows to cover at least three neighbours and at most eight neighbours. The $Cskip$ values are respectively 21, 5 and 1 at depths 0, 1 and 2 (Equation (9)). For a given node, we give, separated by a comma, the ZigBee address (interval label that results from equations (10)-(11)) and the prefix label using VLS.

In this example, the source node s is at depth 3 and has ZigBee address 4 and nodes a, c, b, f and k as neighbours. The prefix label of s is 112 (node s is the second child of b which is the first child of a which is the first child of the root r (PAN)). As shown in Figure 5, three node-disjoint paths are established from s to r. The first one is the classical path based on parent-child links and is the shortest one $\mathcal{P}_1 = P_T(s) = (s, b, a, r)$. To construct the second path, node s takes c as its next-hop node since c is an MPR-neighbour of s since $prefix(c) \neq prefix(s)$. Thus $\mathcal{P}_2 = (s) \rightarrow P_T(c)$. The establishment of the third path requires a discovery phase through node k that is the last remaining neighbour as the next-hop. k sends the *ExploreMsg* to y rather than x since y is located at a lower depth. y in turn sends the *ExploreMsg* to z that is an MPR-neighbour that belongs to a non busy MPRS. The resulting node-disjoint path is $\mathcal{P}_3 = (s, k, y) \rightarrow P_T(z)$.

6.5 Prefix Labelling Analysis in ZigBee

The ZigBee standard specifies that the tree height (L_m) ranges from 2 to 7, that the tree width (C_m) and the number of child routers does not exceed 32. Moreover, the ZigBee network address is encoded using $n = 16$ bits which further limits the maximum value that the L_m parameter can take depending on the values given to C_m and R_m. Given n bits to encode addresses, the maximum number of nodes a tree with parameters L_m, C_m and R_m can not exceed 2^n, that is (Equation (5)) :

$$\frac{C_m R^{L_m} - C_m - 1 + R_m}{R_m - 1} \leq 2^n$$

which results in :

$$L_m \leq log_R \left(\frac{2^n(R_m - 1) + C_m + 1 - R_m}{C_m}\right)$$

Knowing that ZigBee sets a maximum to $L_m = 7$ then the maximum depth we can have is :

$$min(7, \lfloor log_{R_m} \frac{2^n(R_m - 1) + C_m + 1 - R_m}{C_m} \rfloor) \quad (13)$$

In a perfect C_m-ary tree where $C_m = R_m$, the maximum depth is given by :

$$min(7, \lfloor log_{C_m}(2^n(C_m - 1) + 1) \rfloor - 1) \quad (14)$$

Taking into consideration the above limitations and using formulas (6) and (8), we generate for all possible ZigBee tree parameters, the maximum and mean label size induced by FS and VLS. We do not consider FLS since VLS performs better as already shown in Section 5. Figures 6(a) and 6(b) plot respectively the mean and the maximum number of bits required to encode labels as the tree width varies. In average and with respect to FS, both ZigBee (16 bits) and VLS introduce extra bits to encode their addresses. However, we can see that in average VLS induces less extra bits for many values of W. Maximum values shown in Figure 6(b) almost correspond to small values of R_m. The maximum size of a VLS label is at most 42 bits.

As stated before, the smallest VLS labels can be achieved when the tree is perfect i.e. where the root and each internal node has W children. Figure 7(a) and 7(b) plot respectively the mean and the maximum number of bits required to encode labels as the tree width varies in the case of a perfect C_m-ary tree ($C_m = R_m$). We can see that in average, VLS requires less than 16 bits and tends to achieve FS labels size. Even the maximum label size of VLS, as shown in Figure 7(b), does not introduce more than 5 bits as compared to a 16-bit ZigBee address.

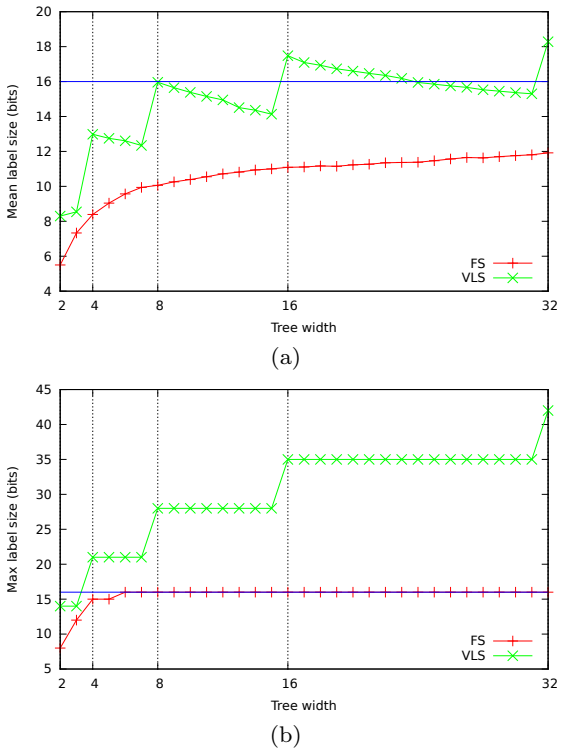


Fig. 6 Mean and maximum label size in a ZigBee network ($Rm = 2..C_m$).

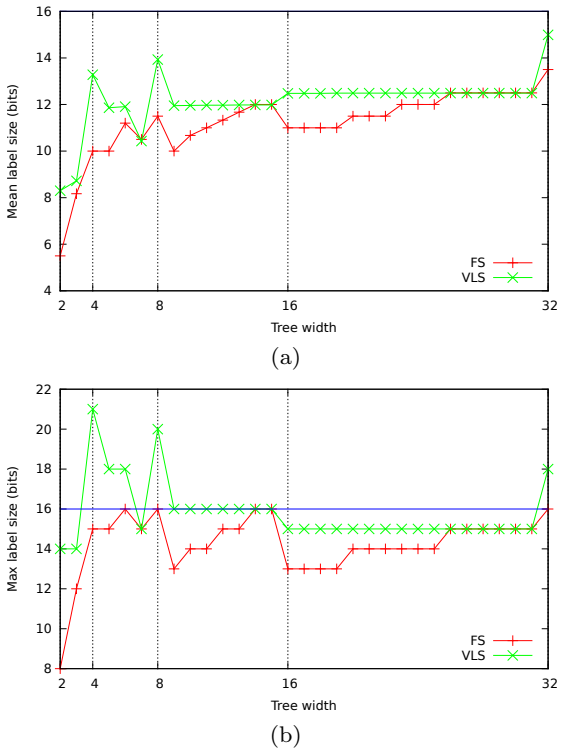


Fig. 7 Mean and maximum label size in a ZigBee network ($R_m = C_m$).

7 Simulation Results

We performed ns2 [2] simulations to get more insight on the behaviour of our proposed multipath routing in the context of the ZigBee technology. We adopted the IEEE 802.15.4 implementation provided by Zheng [43] and the two ray ground propagation model provided in ns2. In the ns2 implementation of our multipath routing protocol called ZMPR (ZigBee version of MPR), we considered six relationship types, namely : child, parent, sink neighbour, MPR-neighbour, sibling and cousin. When transmitting explore messages, we chose to do not consider a child node as a candidate in order to converge rapidly and avoid loops. The explore message is first sent to a sink neighbour if any then in this order to an MPR-neighbour, a cousin node and lastly a sibling node. Since ZMPR follows a hybrid approach, we compare it with the classical proactive ZigBee tree routing (ZTR) and a reactive node disjoint multipath routing based on AODV called MAODV [10].

We simulated a sensor network similar to the one of Figure 5 consisting of static sensor nodes ranging from 25 to 189 deployed in a square sensor field. The sink is located at the centre of the area and sensor nodes are deployed in a rhombic lattice with 10 meters separating two adjacent nodes located at the same row or column of the grid. The transmission range is set to 11 m and 15 m so nodes can have at most 8 and 12 neighbours respectively. For both ZTR and ZMPR, the ZigBee tree parameters (L_m, C_m, R_m) are set to $(7, 4, 4)$ so a perfect 4-ary tree with a maximum depth of 7 is built. The source, randomly chosen among nodes in the network, sends to the sink data packets using a Poisson traffic generator.

We adopted the ns2 energy model and assigned to each node the same initial energy level of 0.5 J. Energy consumption is estimated based on power consumption of MICAz sensor node [1] where transmission, reception and idle time power consumption are respectively : $42mW$, $59.1mW$ and $60\mu W$. Each simulation lasts at least 120 seconds and repeated 20 times. Simulation results are averaged to produce mean values for different metrics to assess the performances of the simulated protocols. Main simulation parameters are summarised in table 2.

We begin by evaluating the control messages overhead of ZMPR compared to a table-based multipath routing protocol MAODV. We do not consider initial tree construction overhead since it does not apply for every path Establishment and that most of WSN testbeds build such a tree at their initial setup. However, control messages related to tree maintenance due to network dynamics are counted. Figure 8 shows the total number

Parameters	Values
(L_m, C_m, R_m)	$(7, 4, 4)$
Number of nodes	25, 49, 100, 189
Distance between adjacent nodes	10 m
Transmission range (Tr)	11 m or 15 m
Packet payload	80 bytes
Transmission rate	1280 bps, 32 Kbps
Initial energy	0.5 J
Transmission power	42 mW
Reception power	59.1 mW
Idle power	60 μW

Table 2 Simulation Parameters

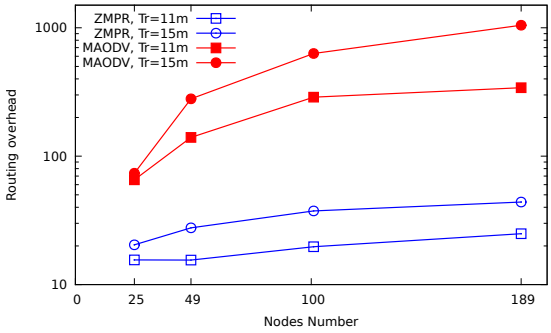


Fig. 8 Routing overhead in ZMPR compared to MAODV.

of exchanged control messages required to build multiple paths as a function of network size. We observe that in both protocols, the routing overhead increases with the number of nodes in the network. In fact, when the number of nodes increases, the source randomly chosen is likely to be located at a higher depth and consequently the number of control messages increases. However, unlike MAODV that makes broadcasts to build its paths, ZMPR finds node-disjoint paths with a very limited overhead as depicted by Figure 8.

Figure 9 plots the average number of node-disjoint paths built by ZMPR and MAODV as a function of network size. We can observe that the number of paths depends on the network connectivity rather than the number of nodes in the network. When the transmission range is higher (15 m instead of 11 m), the degree of both the source and the sink are more likely to increase. As a result, the number of built paths increases as well. We note that ZMPR is able to build more paths than MAODV. This is due to the fact that the former uses unicast explore/response messages while the latter is based on RREQ/RREP broadcast which augments collisions and so the ratio of successful received control RREQ/RREP messages. Even if ZMPR is able to build more paths, in what follows, we limit their number to two and compare it to ZTR.

Figure 10 shows the average length (hop count) of paths built by ZMPR compared to ZTR as the number

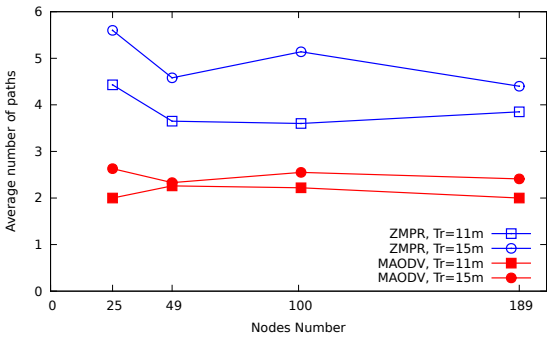


Fig. 9 Number of built paths in ZMPR compared to MAODV.

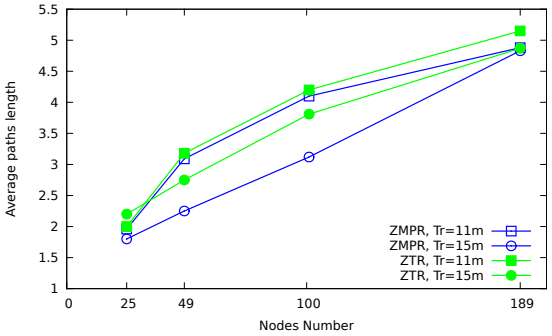


Fig. 10 Length of paths in ZMPR compared to ZTR.

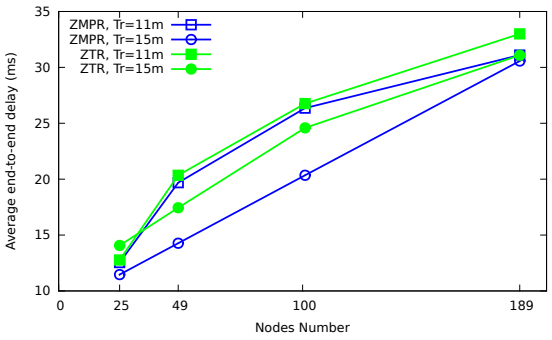


Fig. 11 Routing delay in ZMPR compared to ZTR.

of nodes varies. We clearly observe that when the network size increases the average length of paths increases too. However, we can see that the average path length in ZMPR is almost shorter than the tree path (ZTR) mainly when the network density is higher ($Tr = 15m$). This is mainly due to the fact that the tree path comprises only tree links while other paths can be shortened through the use of neighbour relationships a node can have. This results in reduced delays in data routing to the sink as depicted by Figure 11.

From an energy perspective, unlike single path tree routing, multipath routing allows spreading energy utilisation across a larger number of nodes. This feature leads to a higher network lifetime. Figure 12 shows

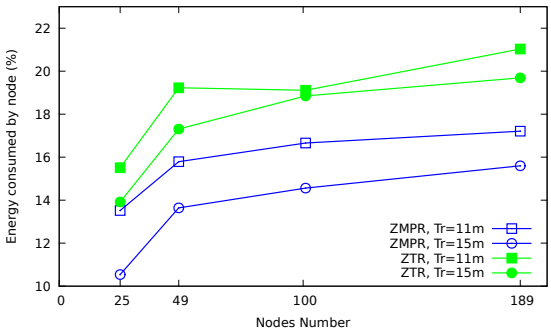


Fig. 12 Average consumed energy by routing nodes in ZMPR and ZTR.

the percentage of average consumed energy per routing node in the network under different network sizes. Energy consumption increases with the network size since the number of involved nodes in routing increases. Independently of the transmission range value, we can observe that ZMPR consumes less energy than ZTR.

In order to evaluate the benefit of multipath routing in terms of throughput, we rise the transmission rate to 50 packets per second giving a useful transmission rate of 32 Kbps and repeated our simulations with 189 nodes and transmission ranges of 11 and 15 m. Figure 13 plots the network throughput variation during 120 seconds of simulation. We can see that multipath routing outperforms single path routing especially when the transmission range is set to 11 m. In this case, the mean throughput achieved by ZMPR and ZTR is 31622 bps and 26708 bps respectively. ZMPR allows 18% improvement with respect to ZTR. When the transmission range increases, the improvement is only about 10% since ZMPR is more affected by interference when the transmission range is higher (15 m).

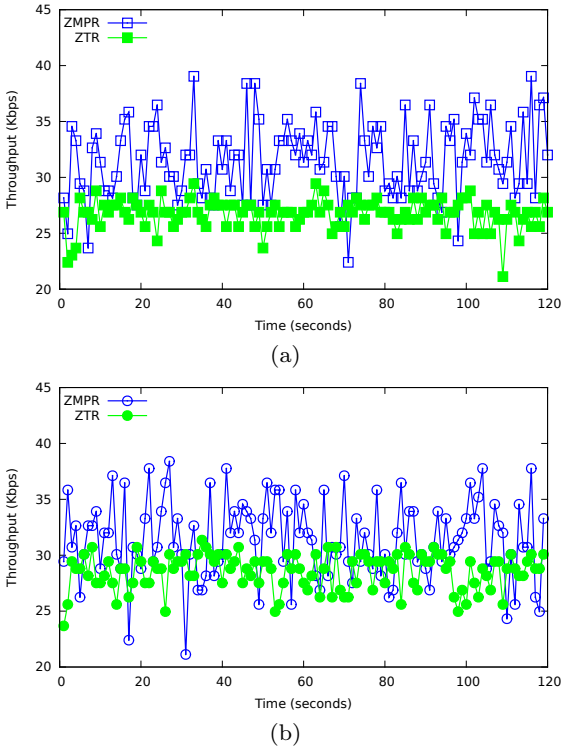


Fig. 13 Overall throughput achieved in ZMPR compared to ZTR (a) $Tr = 11$ m, (b) $Tr = 15$ m.

8 Conclusion

Multipath routing when well designed allows the emergence of new WSN application with higher requirements in terms of quality of service and robustness. In this paper, we proposed a new multipath routing protocol (MPR) that utilises an already built hierarchy rooted at the sink to build additional paths other than the tree path. This is achieved through the use of prefix routing in addition to neighbours information maintained at each node of the network. MPR is a hybrid routing protocol where at least one path is proactively discovered and others can be built using a light discovery process.

For practical implementation, we proposed two labelling schemes and evaluated their respective overhead in terms of additional required bits. We showed that in a perfect W-ary tree, this overhead is limited to at most $(H-1)$ bits where H is the height of the tree. An adequate definition of the maximum number of children in the tree allows to further reduce the number of bits required to encode nodes' labels.

We proposed ZMPR, the MPR adaptation to the ZigBee standard with very limited changes to the standard. Indeed, we made use of its association procedure, its neighbours table and potentially its interval addressing scheme. ZMPR is implemented in ns2 simulator and simulation results showed mainly its limited overhead in terms of control messages compared to a table-based multipath protocol as well as energy consumption distribution compared to a tree path protocol.

As a future work, we plan to make more simulations to assess different discovery strategies in addition to the robustness of MPR against frequent topology changes. The use of a different metric than the number of hops to select a candidate has to be explored. We are mainly considering the use of an interference-aware metric to minimise the inter-path interference in MPR. Furthermore, we expect to carry out a formal specification to verify MPR correctness. Afterwards, an implementation of MPR can be undertaken for experiments on a real WSN test-bed.

A Appendix

A.1 Proof of Lemma 1

Based on the definition of an MPRS, an MPRS root is located at depth 1. Since v is a descendant of u and assuming that $l(u) = s_1$ then based on the prefix labelling scheme defined in Section 3.1, we can write $l(v) = s_1 s_2 \dots s_{d_v}$. It follows that $\forall v \in \mathcal{T}_u \quad : \; prefix(v) = prefix(u) = l(u) = s_1$. The MPRS identifier is unique since its root is unique.

A.2 Proof of Lemma 2

Assume that nodes u and v are not MPR-neighbours, that is, they belong to the same MPRS. This means that (based on lemma 1) $prefix(v) = prefix(u)$. Using contraposition : if $prefix(u) \neq prefix(v)$ then u and v are *MPR-neighbours*.

A.3 Proof of Theorem 1

Let a and a' be at depth d and d' respectively. We can write :

$$P_T(a) = (a_d, a_{d-1}, \dots, a_2, a_1, r)$$

$$P_T(a') = (a'_{d'}, a'_{d'-1}, \dots, a'_2, a'_1, r)$$

Since $P_T(a)$ and $P_T(a')$ are tree paths then $\forall i,\ prefix(a_i) = l(a_1)$ and $\forall j,\ prefix(a'_j) = l(a'_1)$.
It is obvious that if all nodes (except r) that compose $P_T(a)$ and $P_T(a')$ belong to two different MPRSs then based on definition 1, it follows that $P_T(a)$ and $P_T(a')$ are node-disjoint. Assume now that $P_T(a)$ and $P_T(a')$ are node-disjoint. Based on definition 1, their unique common node is $r : \forall\ i, j\ l(a_i) \neq l(a'_j))$ and in particular $l(a_1) \neq l(a'_1)$ then $prefix(a_1) \neq prefix(a'_1)$ which means that nodes a_i $(i = 1..d)$ and a_j $(j = 1..d')$ belong to two different MPRS rooted at a_1 and a'_1 respectively (lemma 1).

A.4 Proof of Corollary 2

Let $P(s)$ and $P'(s)$ be two paths from s to the root r. Assume that node $v_i \in P(s)$ and node $w_j \in P'(s)$ belong to the MPRS $\mathcal{T}_u$. It follows that $P(s) = (s, \dots, v_i, \dots, u, r)$ and $P'(s) = (s, \dots, w_j, \dots, u, r)$, hence $P(s)$ and $P'(s)$ have at least u as a common node in addition to s and r which means that $P(s)$ and $P'(s)$ are not node-disjoint.

A.5 Proof of Theorem 2

In a W-ary tree, $W+1$ symbols are required then the number of bits required to encode a symbol in Σ is $w = \lceil log(W + 1) \rceil$. Thus, the maximum label size in both FLS and VLS is $H.w = H.\lceil log(W + 1) \rceil$. In order to compute the maximum number of additional bits by the proposed labelling schemes, we consider the worst possible case where the tree have a width of 2^{w-1}. In a perfect W-ary tree, we have :

$$|V| = \frac{W^{H+1} - 1}{W - 1}$$

To uniquely identify every node in this tree using FS, we need at least $\lceil log|V| \rceil$ bits. if $W = 2^{w-1}$, we can easily show that $|V| \in]W^H, 2 \times W^H[$, that is :

$$|V| \in]2^{H(w-1)}, 2^{H(w-1)+1}[$$

Which results in $\lceil log|V| \rceil = H(w-1) + 1$ as the minimum number of bits to address all nodes in the tree. Since, in the worst case, the number of required bits in FLS and VLS is $H.w$, it comes that the number of additional bits is at most $H - 1$.

A.6 Proof of Corollary 3

Since $W \geq 2$, the number of bits per symbol w is at least 2. We can write :

$$Hw \geq 2H > 2(H-1) \Rightarrow Hw < 2(Hw - (H-1))$$

Thus the corollary is proved.

References

1. Crossbow MICAz product datasheet. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf
2. Network simulator 2. http://www.isi.edu/nsnam/ns
3. IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (LR-WPANs). IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006) pp. 1–314 (2011)
4. Zigbee alliance. zigbee document 02130r9: Network specification (July 2004)
5. Afsar, M.M., Tayarani-N, M.H.: Clustering in sensor networks: A literature survey. Journal of Network and Computer Applications **46**, 198–226 (2014)
6. Bakker, E.M., van Leeuwen, J., Tan, R.B.: Prefix routing schemes in dynamic networks. Computer Networks and ISDN Systems **26**(4), 403 – 421 (1993)
7. Banerjee, S., Khuller, S.: A clustering scheme for hierarchical control in multi-hop wireless networks. In: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2, pp. 1028–1037 vol.2 (2001)
8. Bhatti, G., Yue, G.: A structured addressing scheme for wireless multi-hop networks (2007). Mitsubishi Electric Research Labs, Cambridge, MA
9. Bidai, Z., Haffaf, H., Maimour, M.: Node disjoint multipath routing for zigbee cluster-tree wireless sensor networks. In: International Conference on Multimedia Computing and Systems (ICMCS), pp. 1–6 (2011)
10. Bidai, Z., Kechar, B., Haffaf, H.: Node disjoint multipath routing protocol supporting quality of service in wireless sensor networks. In: in Proceedings of International Conference on Web Information Technologies (ICWIT'08), pp. 281–286. Sidi Bel Abbes, Algeria (29-30 June 2008)
11. Bidai, Z., Maimour, M., Haffaf, H.: Multipath extension of the zigbee tree routing in cluster-tree wireless sensor networks. International Journal of Mobile Computing and Multimedia Communications **4**(2), 30–48 (2012)
12. C. Perkins E. Royer, S.D.: Ad hoc on demand distance vector (aodv) routing (1999)
13. Clausen, T., Jacquet, P.: Optimized link state routing protocol (olsr). IETF RFC 3626 (2003)

14. Deb, B., Bhatnagar, S., Nath, B.: Reinform: Reliable information forwarding using multiple paths in sensor networks. In: Proceedings of 28th Annual IEEE International Conference on Local Computer Networks, p. 406415. Bonn, Germany (2003)
15. Ganesan, D., Govindan, R., Shenker, S., Estrin, D.: Highly-resilient, energy-efficient multipath routing in wireless sensor networks. ACM SIGMOBILE Mobile Computing and Communications Review **5(4)**, 11–25 (2001)
16. Ha, J., Park, H., Choi, S., Kwon, W.: EHRP: Enhanced hierarchical routing protocol for zigbee mesh networks. In: IEEE Communications Letters, vol. 11, pp. 1028–1030 (2007)
17. Hou, X., Tipper, D., Kabara, J.: Label-based multipath routing (lmr) in wireless sensor routing. In: Proceedings of the 6th International Symposium on Advanced Radio Technologies (ISART 04). Boulder, CO (2004)
18. Huang, X., Fang, Y.: End-to-end delay differentiation by prioritized multipath routing in wireless sensor networks. In: Military Communications Conference (MILCOM), pp. 1277–1283. Atlantic City (2005)
19. Iwanicki, K., Steen, M.V.: On hierarchical routing in wireless sensor networks. In: IPSN, p. 133144 (2007)
20. Johnson, D.B., Maltz, D.A., Broch, J.: DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, chap. 5, pp. 139–172. Addison-Wesley (2001)
21. Karp, B., Kung, H.T.: Gpsr: Greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom '00, pp. 243–254. ACM, New York, NY, USA (2000)
22. Kim, T., Kim, D., Park, N., Yoo, S.E., Lopez, T.: Shortcut tree routing in zigbee networks. In: Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on (2007)
23. Lee, S., Gerla, M.: Split multipath routing with maximally disjoint paths in ad hoc networks. In: IEEE ICC, vol. 10, pp. 3201–3205 (2001)
24. van Leeuwen, J., Tan, R.B.: Routing with compact routing tables. Tech. Rep. RUU-CS-83-16, Department of Computer Science, University of Utrecht, Utrecht (1983)
25. van Leeuwen, J., Tan, R.B.: Compact routing methods: A survey. In: DEPT. OF COMPUTER SCIENCE, UTRECHT UNIVERSITY, pp. 99–109. University Press (1995)
26. Lou, W.: An efficient n-to-1 multipath routing protocol in wireless sensor networks. In: IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, pp. 672–679. IEEE, Washington, DC (2005)
27. Lu, Y.M., Wong, V.W.: An energy-efficient multipath routing protocol for wireless sensor networks. Wiley International Journal of Communication Systems, special issue on Energy-efficient Networks Protocols and Algorithms for Wireless Sensor Networks (2006)
28. Maimour, M.: Maximally radio-disjoint multipath routing for wireless multimedia sensor networks. In: Proceedings of the 4th ACM workshop on Wireless multimedia networking and performance modeling colocated with MSWIM'08. Vancouver, Canada (2008)
29. Maimour, M., Zeghilet, H., Lepage, F.: Sustainable Wireless Sensor Networks, chap. Cluster-based Routing Protocols for Energy-Efciency in Wireless Sensor Networks, pp. 167–188. Intech (2010)
30. MAO, Y., WANG, F., QIU, L., LAM, S.S., SMITH, J.M.: S4: Small state and small stretch routing protocol for large wireless sensor networks. In: USENIX NSDI. Cambridge, MA, USA (2007)
31. Ming Lu, Y., W. S. Wong, V.: An energy-efficient multipath routing protocol for wireless sensor networks: Research articles. Int. J. Commun. Syst. **20**(7), 747–766 (2007)
32. Mohsin, M., Prakash, R.: IP address assignment in a mobile ad hoc network. In: Proceedings of Milicom, vol. 2, p. 856861 (2002)
33. Nefzi, B., Song, Y.Q.: Performance Analysis and improvement of ZigBee routing protocol. In: 7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT'2007. Toulouse France (2007)
34. Qiu, W., Skafidas, E., Hao, P.: Enhanced tree routing for wireless sensor networks. Ad hoc networks **7**(3), 638–650 (2009)
35. Santoro, N., Khatib, R.: Labeling and implicit routing in networks. The Computer Journal **28**(1), 5–8 (1985)
36. Sarkar, R., Zhu, X., Gao, J.: Distributed and compact routing using spatial distributions in wireless sensor networks. ACM Transactions on Sensor Networks (TOSN) **9**(3), 32 (2013)
37. Subasri, M., Ramesh, R.: Neighbor table based shortcut tree routing in zigbee wireless networks. Int. J. Adv. Eng **1**(3), 367–372 (2015)
38. Thorup, M., Zwick, U.: Compact routing schemes. In: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures, pp. 1–10. ACM (2001)
39. Tsai, C.H., Pan, M.S., Lu, Y.C., Tseng, Y.C.: Self-learning routing for zigbee wireless mesh networks. In: IEEE Asia-Pacific Wireless Communications Symposium (APWCS) (2009)
40. Wu, J., Sheng, L.: Deadlock-free routing in irregular networks using prefix routing. In: IN PROCEEDINGS OF PARALLEL AND DISTRIBUTED COMPUTING SYSTEMS, pp. 424–430 (1999)
41. Wu, J., Sheng, L.: Deadlock-free routing in irregular networks using prefix routing. In: ISCA 12th International Conference on Parallel and Distributed Computing Systems, pp. 424–430 (1999)
42. Y. Yu, D.E., Govindan, R.: Geographical and energy-aware routing: a recursive data dissemination protocol for wireless sensor networks. Tech. Rep. UCLA-CSD TR-01-0023, UCLA Computer Science Department (2001)
43. Zheng, M.: NS2 simulator for IEEE 802.15.4 (2004). http://ees2cy.engr.ccny.cuny.edu/zheng/pub/
44. Zhong, F.Y.G.: Gradient broadcast: A robust data delivery protocol for large scale sensor networks. ACM Wireless Networks (WINET **11**, 285–298 (2005)