Exploiting Virtual Coordinates for Improved Routing Performance in Sensor Networks

Abdalkarim Awad, Reinhard German, and Falko Dressler, *Senior Member*, *IEEE*

Abstract—We present the Virtual Cord Protocol (VCP), which exploits virtual coordinates to provide efficient and failure tolerant routing and data management in sensor networks. VCP maintains a virtual cord interconnecting all the nodes in the network and which, operating similar to a Distributed Hash Table (DHT), provides means for inserting data fragments into sensor nodes and retrieving them. Furthermore, it supports service discovery using indirections. VCP uses two mechanisms for finding paths to nodes and associated data items: First, it relies on the virtual cord that always provides a path toward the destination. Second, locally available neighborhood information is exploited for greedy routing. Our simulation results show that VCP is able to find paths close to the shortest path (achieving a stretch ratio of less than 125 percent) with very low overhead. We also extended VCP with data replication mechanisms to improve failure handling. The routing performance of VCP, which clearly outperforms other ad hoc routing protocols such as Dynamic MANET On Demand (DYMO), is similar to other virtual addressing schemes, e.g., Virtual Ring Routing (VRR). However, we improved VCP to handle frequent node failures in an optimized way. The presented results outline the capabilities of VCP to handle such cases more efficiently compared to other protocols. We also compared the capabilities to reliably store and retrieve data in the network to Geographic Hash Tables (GHTs). VCP, in the worst case, performs similar to GHTs, but outperforms this protocol in most cases, especially when complex routing is involved.

Index Terms—Virtual coordinates, ad hoc routing, data management, sensor networks.

1 Introduction

The use of virtual coordinates is currently being investigated for efficient routing in Wireless Sensor Networks (WSNs) [1], [2], [3], [4], [5]. In general, WSNs provide an interesting research domain because they represent a class of massively distributed systems in which nodes are required to work in a cooperative and self-organized fashion to overcome scalability problems [6], [7], [8]. Additionally, WSNs are facing strong resource limitations: many sensor nodes with strong CPU, energy, and bandwidth restrictions need to be operated to build stable and operational networks. This includes the need to solve problems with high dynamics introduced by joining and leaving nodes.

In recent years, WSNs have changed from purely academic research testbeds into real-world applications. Nevertheless, many of the original research issues still apply [6]. Among others, routing has attracted many research projects. Over the last decades, a wide variety of routing protocols has been developed in the domain of sensor networks [9]. However, most practical approaches rely on standard Mobile Ad Hoc Network (MANET) protocols such as Ad Hoc on Demand Distance Vector (AODV) [10], Dynamic MANET On Demand (DYMO) [11], or Dynamic Source Routing (DSR) [12]. On the other hand, WSNs show specific capabilities that demand completely different routing approaches. One of the major requirements in the domain of sensor networks is the need for network-centric operation [13], [14]. This property relies on the main working principles of WSNs, i.e., data are collected in a distributed way and need to be analyzed as close as possible to the data source. This working behavior saves communication and energy resources in sensor networks to a large extent. Combined with database technology such as data stream query processing, further optimization can be achieved [15].

In particular, the following two operations (see Fig. 1) need to be supported in an efficient way. First, data need to be identified. This includes a mapping between an identifier and node addresses where the data should be retrieved from ("pull") or where the data should be transferred to ("push"). Service discovery could be supported in the same way by associating services with identifiers and nodes. Second, packets have to be routed from and to the identified node. Furthermore, data preprocessing might be supported.

These observations motivate the combination of data storage and communication capabilities. Interestingly, similar requirements can be observed in a different application domain, e.g., in peer-to-peer networks used in the Internet. Here, Distributed Hash Tables (DHTs) are successfully used to distribute data over a large number of peers and to find optimal paths toward these data [16]. The main idea is simple: Data items are associated with numbers and each node in the network is responsible for a range of these numbers. Therefore, it is easy to find the node at which a data item is stored. Usually, DHTs are built on the application layer and rely on an underlying routing protocol that provides connectivity between the nodes. Systems like Chord [17] or Pastry [18] have been implemented, in which

- *A. Awad is with the Faculty of Computer Science and Automation, Ilmenau University of Technology, PO Box 10 0565, 98684 Ilmenau, Germany. E-mail: abdalkarim.awad@tu-ilmenau.de.*
- *R. German is with the Department of Computer Science, University of Erlangen, Martensstr. 3, Erlangen 91058, Germany. E-mail: german@informatik.uni-erlangen.de.*
- *F. Dressler is with the Institute of Computer Science, University of Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria. E-mail: falko.dressler@uibk.ac.at.*

Manuscript received 26 June 2009; revised 15 Dec. 2009; accepted 12 Aug. 2010; published online 15 Nov. 2010.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2009-06-0254. Digital Object Identifier no. 10.1109/TMC.2010.218.

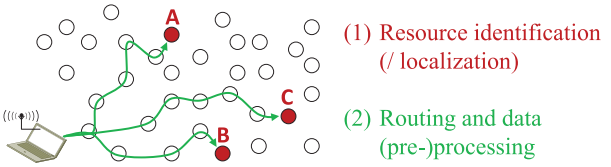

Fig. 1. Scenario description: data and services need to be identified and (optimal) paths for information exchange need to be provided.

the nodes communicate taking advantage of the already existing routing protocols on the Internet.

Implementing DHTs in WSNs as an overlay and relying on typical MANET routing protocols has the drawback that these routing protocols already need to maintain globally valid topology information of the entire network. This cannot scale well because additional overhead is needed to maintain the overlay. In addition, the DHTs have not been designed to take advantage of geographic dependencies, e.g., physically neighboring nodes. On the other hand, routing protocols that use geographic location like Greedy Perimeter Stateless Routing (GPSR) [19] and Geographic Routing Without Location Information (GRWLI) [20] can scale well. Unfortunately, obtaining the location is not only costly and susceptible to localization errors, but sometimes even not feasible. Thus, greedy forwarding cannot guarantee reachability of all destinations because of possible dead ends [21]. Transformation of geographic coordinates can be considered as a first step toward the use of virtual coordinates only [22]. Also, hierarchical approaches using a mixture of geographical coordinates and a virtual overlay have been considered [23]. Comprehensive surveys of such virtual coordinate-based solutions can be found in [4], [5].

In this paper, we present an extended version of our Virtual Cord Protocol (VCP), which explicitly exploits the advantages of using virtual addressing schemes in WSNs [3], [24]. Motivated by studies on dynamic address allocation techniques that do not scale well in sensor networks [25], we address two objectives in this paper: efficient routing toward clearly identified data items and fault tolerance w.r.t. frequent node failures. VCP follows similar concepts as are used in DHTs to provide $O(1)$ complexity for storing and retrieving data items in the network. We extended the protocol to support data replication on the virtual cord in order to improve the ability to cope with frequent node failures in the network. VCP inherently provides service discovery features [26] using DHT-based indirections [27].

In the first part of the paper, we analyze the routing performance of virtual address-based protocols. In particular, we compare our protocol to one of the most promising approaches of virtual routing, namely, Virtual Ring Routing (VRR) [1]. As a reference measurement, we compare both to DYMO [11], which is the defacto standard for MANETs in the scope of the IETF and the successor of AODV, which is usually used for comparing routing protocols to in the fields of WSNs and MANETs. From the results, we see that both virtual address-based protocols clearly outperform the most recent MANET approach.

In the second part of the paper, we study the protocol behavior in the case of frequent node failures. Such a scenario is not unusual in WSNs because nodes may fail, e.g., due to energy outages, or just become "invisible" due to changing conditions of the radio channel. For this evaluation, we explicitly modeled a scenario that is similar to the one described in the Geographic Hash Table (GHT) paper [28] in order to be able to compare the results with this approach as well. For this scenario, we directly compare simulation results for VCP and VRR, and rely on published measurements for GHTs.

The results demonstrate that the routing performance of VCP is similar to VRR in the optimal case, i.e., considering no node failures. However, in the case of node failures, VCP demonstrates the strengths of its efficient cord management. VCP achieves much better tolerance to node failures compared to VRR due to its low maintenance overhead.

The rest of this paper is organized as follows: In Section 2, we outline relevant related work. In Section 3, an overview of the working principles of our VCP protocol is presented. This section also outlines the concepts of data replication in VCP. After a detailed performance analysis of the VCP routing in comparison to DYMO and VRR in Section 5, we investigate the capabilities of VCP to handle frequent node failures in Section 6. Selected performance measures of the data replication mechanism are presented in Section 7. Lastly, Section 8 concludes the paper.

2 Related Work

DHT-based approaches for data management in WSNs can be classified into three main categories: real location-based, virtual location-based, and location independent. GHTs [28], [29] hash keys into geographic locations, so the data items are stored on the sensor node geographically close to the hash of its key. Typically, stored data are replicated locally to ensure persistence when nodes fail. Like normal DHTs, GHTs are built as overlays and rely on underlay routing protocols. For underlay routing, GPSR [19] is used, which exploits the physical location of nodes. Thus, it is assumed that all nodes in the network know their exact location. Since the greedy forwarding routing mode fails in areas in the network without any deployed nodes, GPSR switches to another mode for routing, face routing, which uses a planar subgraph without crossing edges. In comparison, VCP used a virtual cord for efficient routing. It features a predefined hashing range allowing applications to clearly associate data items to places in the virtual cord.

The problem of dead ends and the possibly inefficient face routing to overcome this problem have been investigated in a number of approaches. For example, the Visibility-Graph-based Routing Protocol (VIGOR) uses a hierarchical approach relying on typical MANET protocols in the overlay [30]. This approach has been further extended to completely switch to virtual coordinates on the overlay by performing some coordinate transformation [23]. Similarly, a transformation of geographical coordinates into virtual positions that prevent concave holes in the network allows optimized greedy routing relying on GHT-like services [4], [22].

One of the first virtual coordinate-based protocols was GRWLI [20]. Instead of using real node locations, it constructs an n-dimensional virtual coordinate system, which is based on finding the perimeter nodes and their locations. Then, a relaxation algorithm is used to find the virtual location of all nodes. However, the drawback of having many dimensions resulting from a large n is that forming virtual coordinates

requires a long time to converge [31]. Subsequently, it consumes more communication power. Again, the problem of possible dead ends exists, so the greedy forwarding algorithm cannot guarantee a path to the final destination.

VRR [1] is a routing protocol inspired by overlay DHTs. It uses a unique key to identify nodes. This key is a location independent integer value. VRR organizes the nodes into a virtual ring in the order of increasing identifiers. For routing purposes, each node maintains a set of cardinality r of virtual neighbors that are nearest to their node identifier in the virtual ring. Each node also maintains a physical neighbor set with the identifiers of nodes it can communicate with directly. A proactively maintained routing table identifies the next hop toward each virtual neighbor. The forwarding algorithm used by VRR is quite simple. VRR picks the node with the identifier closest to the destination from the routing table and forwards the message toward that node. The problem of such protocols is that the adjacent nodes in the virtual ring can be far away in the real network. As a result, forwarding to the nearest node can result in a very long path. Moreover, the scalability is an issue because, as the network gets larger, the protocol needs to maintain routing tables of increasing size.

GSpring [32] tries to improve the performance of greedy forwarding. First, each node assigns itself an initial coordinate. Subsequently, nodes adjust their coordinates by simulating a system of springs and repulsion forces. Now, greedy routing is performed with only about 15 percent overhead compared to using real addresses.

In the hop id routing scheme [33], each node maintains a hop id, which is a multidimensional coordinate based on the distance to some landmark nodes. In general, landmarks can be randomly selected in the network. However, to obtain better performance and to reduce the effect of dead ends, the authors present several methods for landmark selection. To construct and maintain the hop id system, one node first floods the entire network to build a shortest path tree. Then, landmarks are selected. Finally, each node periodically adjusts its hop id and broadcasts it using a hello message. When greedy forwarding fails to deal with dead ends, a landmark guided detour is designed and is used with an expanding ring search algorithm to route out of dead ends.

The special case of unidirectional links has been investigated in [2]. The developed virtual coordinate assignment protocol (ABVCap_Uni) supports routing in sensor networks with unidirectional links. Exploiting the availability of unique network IDs of all nodes, the protocol tries to place nodes with unidirectional links into rings and to treat a ring as an extended node. Routing is performed on virtual addresses assigned to real nodes and extended nodes.

The concept of data-centric routing is also exploited in GEM [34], which is a data-centric routing scheme based on a virtual polar coordinate. GEM builds a Virtual Polar Coordinate System (VPCS) that can be used by Virtual Polar Coordinate Routing (VPCR) for routing data and events in which names are hashed to a level and an angle. The main problem in GEM is that when nodes or links fail, a large number of nodes in the systems need to rebuild routing information. Similarly, PathDCS is a data-centric storage scheme that requires a tree construction for routing [35]. The trees are rooted at landmarks, which are used to direct packets in the routing process. Hence, the landmarks are prone to be overloaded. Moreover, in PathDCS, not all the nodes in the network can be storage nodes and, therefore, the load balancing of necessary storage is not optimal. As such, both solutions, GEM and PathDCS, extend the concept of GHT by adding virtual coordinates.

Landmarks are used also in HVGR, which uses concepts from Voronoi graphs to construct and maintain a virtual hierarchy that spans all sensor nodes [36]. The algorithm assumes some number of first level (or highest level) landmarks which divide the network into different first level subregions, whereby each level is divided into subregions. Basically, the landmarks can be selected randomly; however, this can lead to bad performance results. Better results can be obtained by using a flooding-based selection algorithm. It is clear that number of landmarks and the location of them have big effect on the performance of the approach. Landmarks can affect the success rate, the stretch ratio, overhead, and the quality of load balancing.

TABLE 1
Initial Parameters for the Join Process

Parameter, default value	Description
Start $S = 0.0$	lowest position on the cord
End $E = 1.0$	highest position on the cord
Position $P = \mathrm{N/A}$	current position in the cord
HelloPeriod $T_h = 1\,\mathrm{s}$	time interval between `hello` messages
SetPosDelay $T_{ps} = 1\,\mathrm{s}$	time interval before re-requesting a new position
SetVPosDelay $T_{vps} = 1\,\mathrm{s}$	time interval before requesting a virtual position
BlockDelay $T_b = 1\,\mathrm{s}$	blocking period to prevent assigning the same position to more than one node
Interval $I = 0.1$	interval between the two end positions $[S, E]$ and successor or predecessor position
VirtInterval $I_v = 0.9$	interval between node position and virtual node position

3 Virtual Cord Protocol

The idea behind the Virtual Cord Protocol is to combine data lookup with routing techniques in an efficient way. VCP accomplishes this by placing all nodes on a virtual cord, which is also used to associate data items with. A hash function is used to create values in a predefined range $[S, E]$ and each node in the network maintains a part of the entire range. The routing mechanism relies on two concepts: First, the virtual cord can be used as a path to each destination in the network. Additionally, locally available neighborhood information is exploited for greedy routing toward the destination. In the following, the operation of VCP is introduced and extensions for failure management are presented.

3.1 Joining Operation

A number of initial variables are initialized in the startup phase as listed in Table 1. One node must be preprogrammed as initial node, i.e., it gets the position S. We employ `hello` messages to discover the network structure, i.e., all physical neighbors and their position in the cord. In the current implementation of VCP, the `hello` messages are transmitted by means of broadcasting, i.e., each node broadcasts a `hello` message every T_h. The joining operation could also be

executed using an on-demand mechanism, which has advantages in static networks or those with a high node density. Based on the periodically transmitted `hello` messages, the joining node gets information about its physical neighbors as well as their successors and predecessors. Algorithm 1 illustrates the handling of `hello` messages. If the node has not yet joined the network, Algorithm 2 is used to get a relative position in the cord. The artificial join delay T_{ps} is used to prevent conflicts between multiple nodes that simultaneously ask for relative positions.

Algorithm 1. Handling of `hello` messages
Require: Locally stored state of all neighbors in set N
Ensure: Maintain neighbor set N and set virtual address
1: Receive neighbor information from node N_i
2: **if** $N_i \notin N$ **then**
3: $N \leftarrow N \cup \{N_i\}$
4: **else**
5: Update N_i
6: **end if**
7: **if** P is unset AND T_{ps} has expired **then**
8: Reset T_{ps}
9: Try joining the cord (see Algorithm 2)
10: **end if**

Algorithm 2. Joining of the cord
Require: Neighbor information stored in set N
1: **if** $\exists N_i, N_j \in N : \mathrm{Succ}(N_i) == N_j$ **then**
2: $P \leftarrow (\mathrm{Pos}(N_i) + \mathrm{Pos}(N_j))/2$
3: Coordinate position P with N_i
4: Coordinate position P with N_j
5: **else if** $\exists N_i \in N : \mathrm{Pos}(N_i) == S$ **then**
6: $P \leftarrow S$
7: **if** $\mathrm{Succ}(N_i)$ is unset **then**
8: $P_{N_i} \leftarrow E$
9: **else**
10: $P_{N_i} \leftarrow (\mathrm{Succ}(N_i) + S)/2$
11: **end if**
12: Coordinate positions P and P_{N_i} with N_i
13: **else if** $\exists N_i \in N : \mathrm{Pos}(N_i) == E$ **then**
14: $P \leftarrow E$
15: $P_{N_i} \leftarrow (\mathrm{Pred}(N_i) + E)/2$
16: Coordinate positions P and P_{N_i} with N_i
17: **else if** $\exists N_i \in N$ **then**
18: **if** Timer T_{vps} not yet started **then**
19: Start timer T_{vps}
20: **else if** T_{vps} has expired **then**
21: Create virtual node at N_i
22: Coordinate new position P with N_i
23: **end if**
24: **end if**

Each node joining the network has to receive at least one `hello` message from a node that already joined the cord in order to get a relative position in this cord. If a node can communicate with only an end node (lines 5-16 in Algorithm 2), i.e., a node that has either position S or E, the new node takes over this end value as its virtual cord position. The old node gets a new position between the end value and its successor or predecessor depending on its old position. If a node can communicate with two adjacent nodes in the cord, the new node gets a position between the values of the two adjacent nodes (lines 1-4), i.e., the new node becomes successor of the old node with the lower position value and predecessor to the node that has the higher position value. An acknowledgment procedure is used for reliable position exchange among the involved nodes (annotated as *coordinate position* in Algorithm 2).

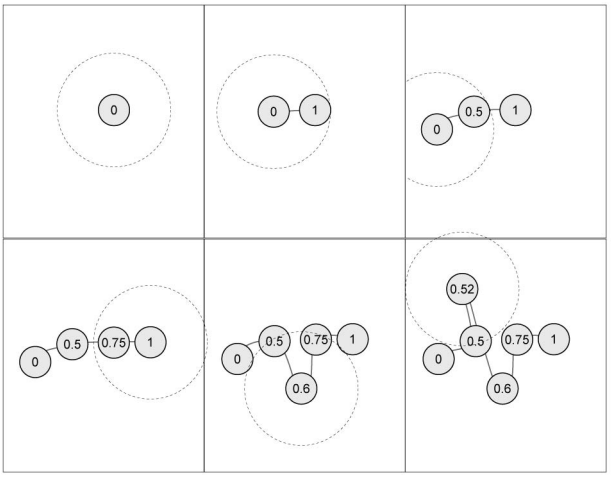

Fig. 2. Basic join operation in VCP, six nodes are joining the network according to the rules described in Algorithm 2.

Finally, if the new node can communicate with only one node in the network, which is neither at S nor E, then this node is asked to create a virtual node (lines 17-23). This virtual node gets a position between the position of the real node and its successor or predecessor. The new joining node can now get a position in between the real and the virtual position of the node in the cord. Notice that the node has to wait some time T_{vps} before asking for a virtual node. This timeout is used to encourage the node to find multiple neighbors, i.e., to get a proper position in the cord without the need to set up a virtual position. Simulation results have shown that fewer virtual nodes lead to better routing paths.

Fig. 2 shows the joining process for a six-node network. The outer circle indicates the communication range of the newly joining node. In the first four steps, nodes are placed in the cord either at an end or in the middle of the cord. In the fifth step, the node joining the network finds two adjacent nodes (0.5 and 0.75). So, it becomes the successor of the first node (0.5) and predecessor of the other node (0.75). The new address is a number between 0.5 and 0.75.

However, when the sixth node joins the network, nodes 0.5 and 0.75 are no longer adjacent. Thus, node 0.5 is asked to create a virtual node as shown in Fig. 3. The main idea is simple: the cord must connect all the nodes in the network in an ordered way. Thus, if no two nodes adjacent in the cord are in the direct communication range of the new node, a virtual node is created to balance the cord again.

It is clear that the joining of a new node only affects a small number of nodes in the vicinity and it is independent of the total number of nodes in the network. In fact, the insertion of a new node only affects $O(m)$ nodes, where m is the number of local neighbors. The final result of the join process is a virtual cord that interconnects all the nodes in the network. This cord does not need to fulfill any specific requirements. In particular, it does not need to be efficient in any sense. However, the cord spans the numbering range

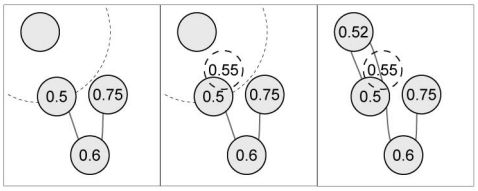

Fig. 3. Operation of creating virtual node. For a new node that only finds one neighbor, the neighbor (node 0.5) creates a virtual node 0.55. Now, the new node can join as 0.52.

$[S, E]$, which can be exploited by a hash function for data storage or service discovery. Furthermore, this cord supports efficient routing in the network. This is detailed in the next section.

3.2 Routing

Routing in VCP is done using the virtual cord. Additionally, local neighborhood information is exploited for greedy routing. The greedy forwarding works as follows: a node with relative position P forwards a packet to its neighbor N_i that has the closest virtual position to the destination D_p. The forwarding is terminated if no more progress is possible, i.e., the local coordinate P is closest D_p. Based on the established cord, VCP routing will always lead to a path to the destination—it is not possible to run into a dead end. Additionally, VCP allows to take shortcuts whenever a physical neighbor with a virtual number is available that is closer to the destination. Simulation studies indicate that the path stretch is almost optimal (about 125 percent in the worst case). This process is outlined in detail in Algorithm 3.

Algorithm 3. Greedy forwarding algorithm
Require: Received data packet D for destination position D_p, locally maintained data set $[P_{min}, P_{max}]$
1: **if** $P_{min} \leq D_p \leq P_{max}$ **then**
2: Store data
3: **else**
4: Select $N_i \in N$ such that $\forall N_j \in N, i \neq j : |\text{Pos}(N_i) - D_p| < |\text{Pos}(N_j) - D_p|$
5: Send D to N_i
6: **end if**

Fig. 4 depicts the network after adding 15 nodes. In our example, node 0.25 has a message to transmit to node 0.78. Thus, it will forward the message toward the destination node via node 0.5, which has the closest position to the value among the physical neighbors. Afterward, node 0.5 will send it to node 0.75, and finally node 0.75 will forward it to node 0.78.

3.3 Failure Management

The presented cord management is working very well as long as all nodes stay available after forming the cord. Greedy forwarding can guarantee the reachability of the destination only if there is no failure. However, in case of node failures, greedy forwarding might fail and the cord becomes unstable. To overcome the problem of finding a path toward the destination in case of node failures, we propose a new scheme to find an alternative path.

The `hello` messages are used to identify failed nodes. In particular, we store the time stamp of the last `hello` message in the routing table, i.e., the physical neighbor

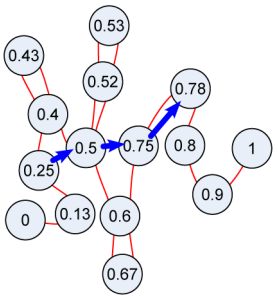

Fig. 4. An example routing path using greedy routing along the virtual cord and exploiting local neighborhood information.

table, in addition to successor and predecessor positions. If a node did not receive a `hello` message from a neighbor for $n \times T_h$, where T_h is the `hello` message period, this neighbor is marked as a dead node. From the information available in the routing table, each node can locally check whether the correct destination of a packet is this dead neighbor itself or one of this neighbor's physical neighbors.

During packet routing, there are two cases in which greedy forwarding cannot reach the correct destination because of a dead end in the cord. In the first case, the failing node could be the final destination. In this case, the packet can be either dropped or stored within the neighbor of the failed node. If the operation was to retrieve data items, the connection is counted as not successful. We added data replication techniques to counteract this case (see Section 7).

In the second case, the failing node could be the next hop toward the destination but not the final destination itself. In this case, we have to find an alternative path. The procedure is as follows: The neighbor of the failing node locally creates a so-called *no path interval* (NP-I). This interval corresponds to the range of IDs that the dead node was responsible for. Then, the node sends a *no path* (NP) packet, which includes NP-I to another active node in its neighborhood. This node is selected according to its position in the cord that should be as close as possible to NP-I. In order to prevent routing loops, this information needs to be stored on all nodes involved in this process. However, the stored NP-I data are expected to be expired after T_{np}. From now on, each node either transmits the data using greedy forwarding toward the destination if there is a neighboring node closer to the destination available, or it continues to send NP packets. Using the stored NP-I data, this information will never be sent twice. If an NP packet reaches a node, which already has NP-I in its table, it has to send a *no path back* (NPB) packet as an indicator of a detected loop. The procedure of treating routing packets is shown in more detail in Algorithm 4. The interval $[P_{min}, P_{max}]$ is maintained by evaluating the distance between the current node and the neighbors on the cord. In particular, this interval is used to identify the final destination for each packet. NP-I is maintained by checking the locally stored NP information, updating the timeouts and expiring old entries.

Algorithm 4. Handle routing failure packets
Require: Received data packet D for destination position D_p, locally maintained data set $[P_{min}, P_{max}]$
1: **if** $P_{min} \leq D_p \leq P_{max}$ **then**
2: Store data

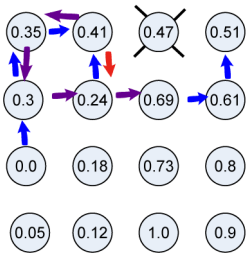

Fig. 5. Routing in the case of a node failure: node 0.41 generates an NP packet that is forwarded until node 0.69 continues to forward the original data packet.

3: **else if** $D \in$ NP-I **then**
4: Send NPB back
5: **else if** $\exists N_i \in N : |\text{Position}(N_i) - D_p| < |P - D_p|$ **then**
6: Send D to N_i
7: **else**
8: Maintain NP-I
9: Send NP back
10: **end if**

In very dynamic scenarios with frequent node failures, this rather strong loop prevention does not guarantee 100 percent success rate. We consider this a minor issue as the overall success rate in our simulations is very promising. Permanent node failures have to be handled by replication (Section 3.4).

An example for packet forwarding in case of a node failure is illustrated in Fig. 5. In this example, node 0.0 produces a data packet destined to address 0.51. According to the greedy routing principles of VCP, the data packet will be forwarded by nodes 0.30 and 0.35 until it reaches node 0.41. At this node, a dead end is detected because the previously existing node 0.47 has failed. Thus, node 0.41 will create a *no path interval* and, accordingly, send an NP packet back the path to node 0.35. Similarly, the NP packet is forwarded until it reaches node 0.24. This one, according to the programmed rules, tries node 0.41 again, however, 0.41 detects a loop and sends an NPB packet to node 0.24. In turn, node 0.24 tries to find another path by sending an NP packet to node 0.69. Finally, this node can resume greedy forwarding toward destination node 0.51.

3.4 Replication Strategies

In order to improve the reliability of VCP in case of possible node failures, we integrated data replication techniques. Replication requires two conceptual steps. First, replicas need to be created and maintained (updated). Second, in the case of failures, available replicas need to be identified. We used two different strategies to support the failure handling of VCP.

First, we investigated concepts to manage replicas in the physical neighborhood of a node. The advantage of this concept is the simplified creation and maintenance of replicas. The broadcast characteristic of the radio communication can be exploited to communicate replication messages to all physical neighbors in a single step. This concept is depicted in Fig. 6a. In the case of failures, the virtual cord can be used to find a node close to the original location, i.e., by routing a request to the vicinity of the failed node. Then, a ring search can be performed, again, using broadcast messages, to identify a replica. The simplified creation and maintenance of replicas is accompanied by a slightly more expensive and time consuming ring search for available replicas.

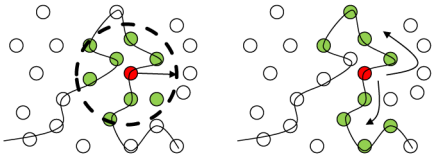

Fig. 6. Replication strategies: on physical neighbors (left) and along the cord (right).

In a second approach, we studied the possibility to exploit the available cord structure of VCP to maintain replicas along the cord, i.e., on direct neighbors in both directions. In this approach, the number of replicas can be controlled by storing duplicates on the n nearest nodes in each direction. The creation and maintenance of replicas needs at least $2n$ messages. On the other hand, the location of replicas is straightforward and, thus, in the case of node failures, no extra effort needs to be spent to identify such replicas.

4 Simulation Model and Parameters

For analyzing the performance of the virtual address-based routing protocols VCP and VRR, we implemented these models in OMNeT++ [37], a discrete event simulator free for academic use. Additionally, we used the *INET framework* that provides detailed simulation models of typical MAC and physical layer protocols.

In our simulation, we built our protocol on the top of the IEEE 802.11 protocol. The key objective was to rely on a contention-based MAC protocol that suits best for distributed and self-organizing routing protocols like VCP and the evaluated competitors. We also have an implementation for IEEE 802.15.4 available [38], which is frequently used in the context of sensor networks. However, as only the non-beacon enabled mode could be used because of the missing coordinator functionality, the performance of IEEE 802.15.4 is expected to be comparable to IEEE 802.11, both operating as contention-based MAC protocols. Furthermore, the comparison to other protocol evaluations in the literature becomes easier as most of them use IEEE 802.11 for the simulation setup as well. For all communications, the complete network stack is simulated and wireless modules are configured to closely resemble IEEE 802.11b network cards transmitting at 2 Mbit/s with RTS/CTS disabled. We also executed simulation experiments for other data rates, in particular for 250 kbit/s and 10 Mbit/s, to resemble typical sensor network and MANET characteristics, respectively. The results only change linearly for throughput and delay. Therefore, we decided not to include the according graphs. For the simulation of radio wave propagation, a plain free-space model is employed, with the transmission ranges of all nodes adjusted to a fixed value of 50 m.

Furthermore, we used our recently implemented model of the DYMO routing protocol, which is now publicly available [39]. DYMO is the successor of AODV, the latter is the protocol most frequently used to evaluate sensor and ad hoc routing protocols with. Certainly, an ad hoc routing protocol has different objectives compared to VCP and other virtual coordinate-based solutions; however, this comparison provides kind of ground truth measures to

TABLE 2
INET Framework Module Parameters

Parameter	Value
mac.bitrate	2 Mbit/s
mac.broadcastBackoff	31 slots
mac.maxQueueSize	14 Pckts
mac.rtsCts	false
snrEval.bitrate	2 Mbit/s
snrEval.headerLength	192 bit
snrEval.snrThresholdLevel	4 dB
snrEval.thermalNoise	−110 dB
snrEval.sensitivity	−85 dB
snrEval.pathLossAlpha	2.5
snrEval.carrierFrequency	2.4 GHz
snrEval.transmitterPower	1 mW
channelcontrol.carrierFrequency	2.4 GHz
channelcontrol.pMax	2 mW
channelcontrol.sat	−85 dBm
channelcontrol.alpha	2.5

TABLE 3
Simulation Parameters

Input Parameter	Value
Number of Nodes	100
Playground size	180 m × 180 m or 400 m × 400 m
Node placement	Grid or random
Data rate	CBR, 1 pps or 0.1 pps
Initialization time	400 s
Start of data transmission	uniformly distributed in [400, 418] s or [400, 580] s
End of data transmission	490 s or 1300 s
Destination node	Upper left node
Fraction of failing nodes	0 %, 20 %, 40 %, 60 %, 80 %, or 100 %
On time	uniformly distributed in [0, 120] s
Off time	uniformly distributed in [0, 60] s
Start of node failures	400 s
End of node failures	1300 s

evaluate the routing performance of any ad hoc routing solution. All simulation parameters used to configure the mentioned modules are summarized in Table 2.

In the baseline scenario, 100 nodes are deployed either in the form of a grid or randomly on a rectangular area. A single node is dedicated as the sink node and placed on the upper left corner of the playground. An example of a 25-node network is depicted in Fig. 7. The figure includes the virtual relative positions and the virtual cord connecting all the nodes. We allowed for an initial transient period of 400 s in which VCP and VRR initialize their address information and routing tables. Afterward, each node starts transmitting at a time in the range [400, 418] s for a data rate of 1 pps (packets per second) and in the range [400, 580] s for a data rate of 0.1 pps. The experiment is terminated at 490 s and 1300 s, respectively. This experiment is then extended to a general peer-to-peer communication setup, where destination nodes are randomly chosen instead of using a single sink node.

For the second set of experiments, we analyzed the failure tolerance of the routing protocol by periodically switching a fraction of the nodes on and off (except for the sink node). Both the on and off intervals are uniformly distributed in a predefined range. After a node is reactivated, it needs to rejoin the network and reestablish its routing information. The configuration of this failure scenario corresponds to the one used to analyze GHTs [28]. For statistical evidence, for each experiment, we performed 10 runs. All the simulation parameters are summarized in Table 3.

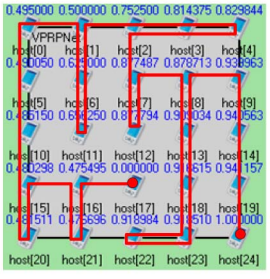

Fig. 7. Screenshot of the simulation setup. Nodes are deployed either in a grid or randomly on a rectangular area.

For our evaluation, we selected four basis metrics. First, we analyzed the success ratio, which describes the capability of the protocol to ensure correct data delivery. Then, the MAC collisions were analyzed to get an impression of the overall network load. Furthermore, the path length is evaluated and, finally, the end-to-end delay. All results discussed in the following sections are shown as boxplots. For each data set, a box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Additional whiskers extend from the edges of the box toward the minimum and maximum of the data set. Data points outside the range of box and whiskers are considered outliers and drawn separately. Additionally, the mean value is depicted in the form of a small filled square.

5 Routing Performance of Virtual Coordinate-Based Protocols

In a first set of simulations, we evaluated the routing performance of VCP. We looked into different aspects that might influence the protocol and performed a comprehensive comparison to DYMO, which is the current standard of the IETF for ad hoc routing, and with VRR, a competitive approach relying on virtual coordinates for routing in sensor networks.

5.1 Quality of Routing Paths

In order to inspect the quality of the routing paths, we examined the stretch ratio, i.e., the ratio between the length of the path traversed by VCP and the shortest path. For different network sizes, we measured the path length from all nodes to the upper left node.

As a base measure, we analytically evaluated the average path length. The average length of routing paths l_{avg} depends on the number of nodes n in the network. We only consider nodes deployed in a grid network in a rectangular area. The maximum length of routing paths l_{max} in the grid can be calculated recursively as shown in (1). The closed form is given in (2). Because the grid is symmetrical, the average length of routing paths l_{avg} can be calculated according to (3). Thus, in case of 25 nodes, the average path length $l_{avg} = 4$ and the maximum path length $l_{max} = 8$ (between end corners).

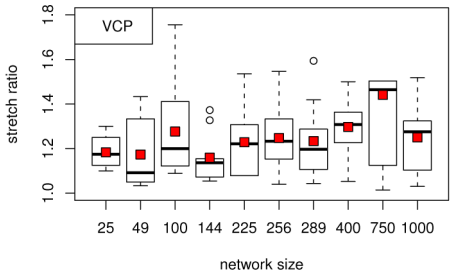

Fig. 8. Stretch ratio for different network sizes.

$$l_{max}(1) = 0,$$
$$l_{max}(n) = l_{max}(\sqrt{n} - 1) + 2, \quad (1)$$
$$l_{max} = 2\sqrt{n} - 2, \quad (2)$$
$$l_{avg} = \frac{l_{max} + l_{min}}{2}, \quad = \sqrt{n} - 1. \quad (3)$$

Fig. 8 shows the measured stretch ratio in the simulations as we varied the network size from 25 to 1,000 nodes. The stretch ratio slightly increases with the network size; however, this increase is reasonable and the median stays below 25 percent. This low stretch level outlines the optimal path selection of VCP. For comparison, the stretch ratio of VRR increases to over 40 percent already for network sizes larger than 200 nodes (data not shown). This is because successors and predecessors, which are included in the routing table, can be far away from each other and, therefore, messages may be routed over many unnecessary nodes. This indicates that in VCP, messages are traveling near optimal paths.

5.2 Influence of the Network Size

We performed a series of experiments to explore the effect of network size on the performance of VCP. Each node in the network sends packets to the same destination, which simulates storing the same data item collected from all the sensor nodes in the network. We varied the network size from 25 to 225 nodes.

Fig. 9a shows the simulation results. The path length increases with the network size in a logarithmic manner. However, there are a few nodes that used a path length larger than the shortest path to reach the destination. Taking a look on the mean and the median of the path lengths, they are almost equal to $l_{max}/2$, which is an indication of good path selection. Also, more than 75 percent of the nodes have a path length smaller than l_{max}. The end-to-end delay is proportional to the path length because it is not necessary to queue packets. Furthermore, the success ratio was 100 percent for all large network sizes.

Instead of evaluating a single sink environment, we also simulated a peer-to-peer communication scenario. It is to be expected that VCP performs especially well and that there is no performance degradation compared to the single source scenario. As can be seen in Fig. 10, both the path length and the end-to-end delay behave as expected for increasing network sizes. For the experiments considering node failures (Section 6), we stay with this peer-to-peer setup.

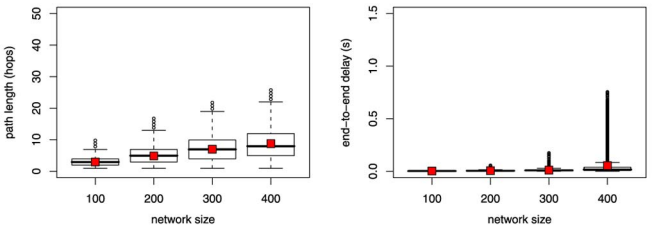

Fig. 10. Peer-to-peer scenario: number of hops and end-to-end delay for varying network size.

5.3 Influence of the Traffic Load

To study the behavior of our protocol under varying traffic load, we kept the number of nodes in the network constant at 100 nodes and each node in the network sent 100 packets to the same destination. For the first experiments, the time

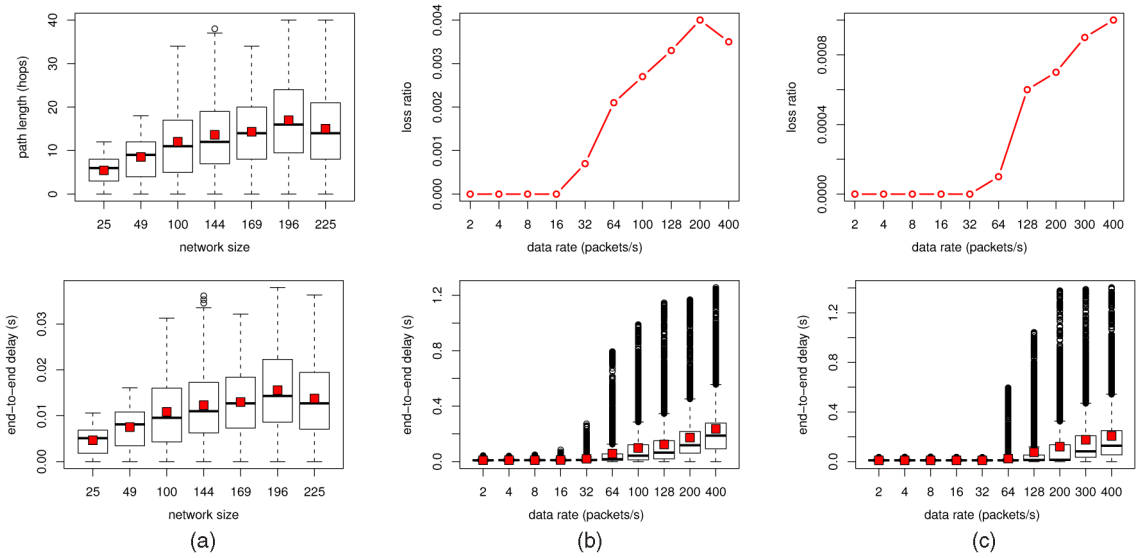

Fig. 9. First performance evaluation results. Most figures are drawn as boxplots indicating the median value and the quartiles. The small box shows the mean value. (a) Number of hops (top) and end-to-end delay (bottom) for varying network size. (b) Loss ratio (top) and end-to-end delay for bursty traffic behavior. (c) Loss ratio (top) and end-to-end delay for constant traffic rate.

TABLE 4
Influence of the Traffic Load for Random Deployment

Measure	1 pps	2 pps
success rate	100 %	99.95 %
average delay	0.0068 s	0.0174 s
max delay	0.065 s	0.234 s
average hop count	5.65	5.79
max hop count	16	16

between successive packets was randomly selected in the interval [0, 0.5] s, which is equivalent to sending at least 2 pps. Afterward, we decreased this time interval down to [0, 0.005] s, which is equivalent to sending at least 400 pps. As shown in Fig. 9b, packets can be delayed at the MAC layer due to congestion. As a result, the end-to-end delay increases with increasing traffic load. Nevertheless, the delay is still within an acceptable range. When sending packets with rates below 16 pps, the effect of congestion is negligible. However, the delay reaches a peak of 1.2 s when sending with a rate of 400 pps (compared to only 0.03 s in the other case). The effect of increasing traffic was not so big on the packet delivery rate. As can be seen from Fig. 9b, the loss ratio was below 0.4 percent; therefore, the success rate was still above 99.6 percent.

We repeated the same experiment using a constant packet rate. We started sending packets every 0.5 s and decreased this duration down to 0.005 s. As shown in Fig. 9c, the results are a little bit better. There was no impact of increasing traffic load until 32 pps and the mean delay was lower than before. To explore the performance of VCP in random deployments, we performed several simulations using a network consisting of 200 nodes deployed randomly in a 600 m $\times$120 m plane, which is similar to scenarios described in [1] for VRR. The packet rate was set to 1 or 2 pps, which is equivalent to 200 or 400 CBR flows, respectively. In less than 20 s, all the nodes had joined the network. The average number of control messages sent by each node (excluding `hello` messages) was 10.54. Each node started sending a 100 byte packet to a random destination at a random time in the interval [50, 230] s. All nodes stopped sending at time 950 s. As shown in Table 4, the results are very promising. For example, the success rate is almost 100 percent, even for high traffic load. In comparison, the success rate for VRR dropped to 60 percent if either the network size was increased to more than 200 nodes or if the traffic load was doubled.

5.4 Comparison to DYMO and VRR

We also evaluated the routing performance of VCP in comparison to VRR, a competitive approach relying on virtual coordinates for routing, and to DYMO, which is the most recent standard of ad hoc routing protocols developed by the IETF MANET working group.

In a first experiment, we evaluated the end-to-end delay as observed by the application. Furthermore, we analyzed the protocol behavior for different network densities and traffic rates. Finally, we evaluated the influence of the network topology, i.e., grid or random. The results for the grid scenario are depicted in Fig. 11 (for random deployment, the results are similar). For better comparability, we normalized the latency to the path length, i.e., to a per-hop delay. As can be seen, the per-hop delay for VCP and VRR is quite similar. Both the mean and the median are at about 1 ms. Some outliers can be observed up to about 10 ms. Both protocols are very robust w.r.t. the network density and the traffic load. The MANET routing protocol DYMO performed slightly worse for higher traffic load (depicted as 1 s traffic pattern). For lower traffic rates, the observed delay increased largely. This effect can be explained with the route timeouts used by DYMO in our experiment. In the 10 s example, DYMO has to set up a route for almost each packet because the available routes have timed out. Thus, each time an additional route setup delay adds to the transmission delay.

For the analysis of routing protocols designed for wireless networks, a main measure to observe is the number of MAC layer collisions. This metric helps to evaluate the overall load in the network. Fig. 12 shows the results for the grid scenario (for random deployment, the number of MAC collisions shows a higher variance, but a similar trend). In particular, the number of MAC collisions is almost zero for the virtual address-based routing protocols (for the high density scenario, VRR shows about 5 percent collisions per data packet sent, which is negligible). However, the number of collisions is quite high for DYMO for high node densities [39].

6 Protocol Behavior in the Presence of Node Failures

In the second set of experiments, we focused on the protocol behavior in presence of frequent node failures. As a node failure, we consider in general any event that prevents communication to a particular node at a given time, e.g., complete energy outages, node replacement, or interrupted

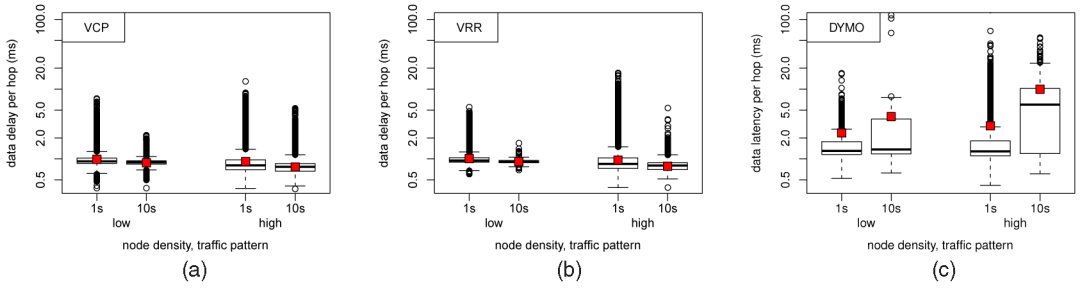

Fig. 11. Delay performance of VCP, VRR, and DYMO in the grid scenario: depicted is the latency as observed by the application normalized to the path length; all figures are plotted using a log scale y-axis. (a) Per-hop delay of VCP. (b) Per-hop delay of VRR. (c) Per-hop delay of DYMO.

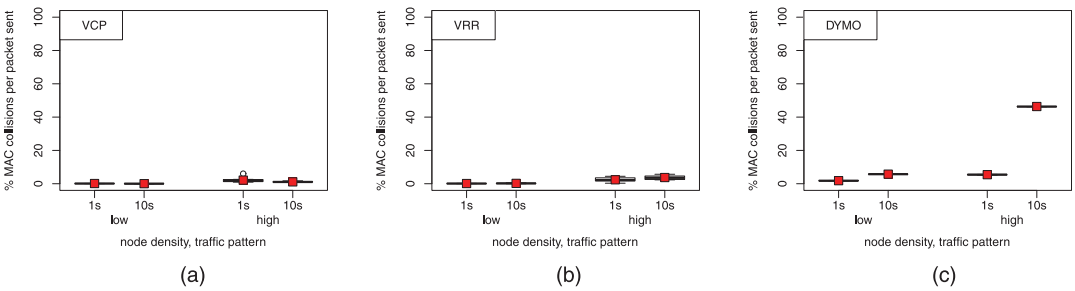

Fig. 12. MAC layer collisions per date packet sent for VCP, VRR, and DYMO in the grid scenario. (a) Collisions of VCP. (b) Collisions of VRR. (c) Collisions of DYMO.

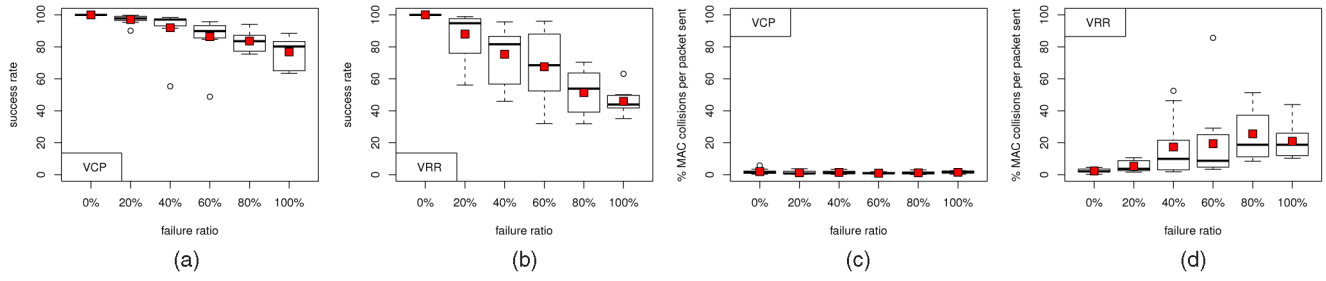

Fig. 13. Failure performance: depicted are the success rate and number of MAC collisions for the grid scenario. (a) Success rate of VCP. (b) Success rate of VRR. (c) Collisions of VCP. (d) Collisions of VRR.

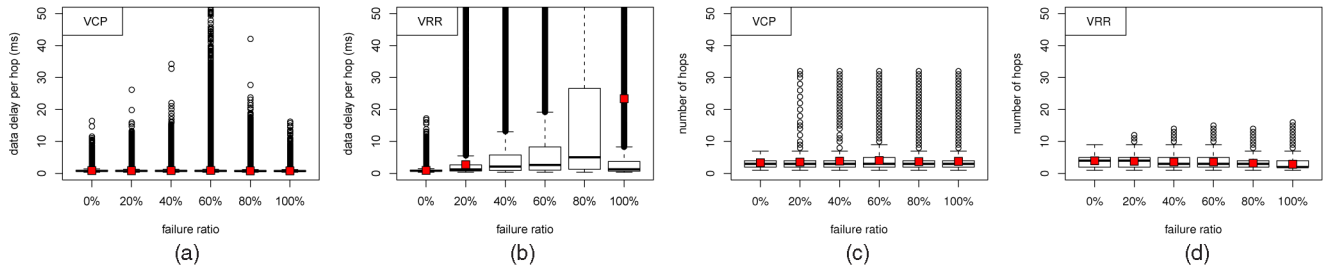

Fig. 14. Failure performance: depicted are the per-hop delay and the path lengths for the grid scenario. (a) Delay of VCP. (b) Delay of VRR. (c) Path length of VCP. (d) Path length of VRR.

communications due to changes in the radio propagation. For these experiments, we only consider VCP and VRR because the network load (and therefore the collision probability) increases too much for MANET protocols such as DYMO [39].

The general setup is the same as used for evaluating the routing performance. However, we further introduced node failures as described in Section 4. We follow the simulation setup described in [28]. In particular, node failures are modeled as a uniformly distributed on/off process. We increase the number of nodes toggling their state from 0 to 100 percent. Four metrics were chosen for the performance comparison. In all the figures, we show results for the grid scenario. For random deployment, the variance was sightly higher but the trend of the results was exactly the same.

First, we investigated the success rate, i.e., the number of transmissions that were completed successfully. Figs. 13a and 13b depict the measured success rate for VCP and VRR, respectively. As can be seen, the ratio of successful transmissions degrades with the number of failing nodes. However, VCP still maintains a success rate of about 70 to 80 percent. In contrast, the success rate degrades much faster for VRR (down to 50 percent). A look at the network load reveals some effects that explain the reduced success rate of VRR compared to VCP. Figs. 13c and 13d show the number of MAC layer collisions. As can be seen, there are almost no collisions for VCP, which outlines the capability of this protocol to work even in situations with a high number of node failures.

When VRR enters the transmission phase after its initial join phase, it simply forwards packets to the node that has the ID closest to the packet ID. It needs to be noted that "closest" ID means the ID that is closest on the virtual ring. When the network size increases and many nodes fail periodically, the forwarding tables become incomplete and only represent a local view of the whole network. VRR has two different strategies to handle such failure situations: exact repair and local *vset-path* repair. The idea is to bypass the failed node. However, this technique only works for a limited number of node failures.

The effect can also be observed when looking at the latency performance. Fig. 14a shows that the median per-hop delay of VCP is not affected by the failing nodes, whereas, as shown in Fig. 14b, the delay of VRR increases with the failure ratio.

We further studied the path length, which outlines the capability of the routing protocol to find shortest paths even in the case of many node failures. Figs. 14c and 14d depict the simulation results. While the average path length is

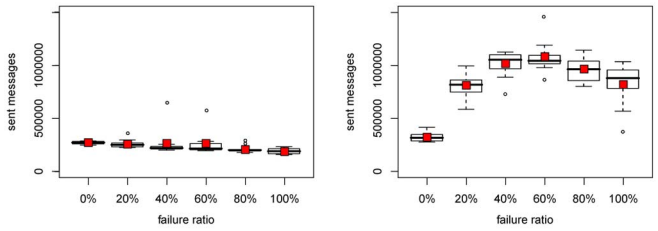

Fig. 15. Total number of messages for routing in VCP (left) and VRR (right).

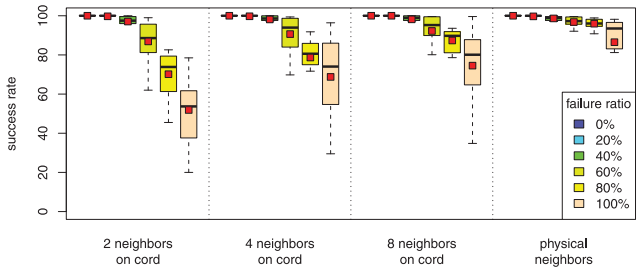

Fig. 16. Success rate with replication (grid scenario) for different replication strategies.

slightly shorter for VCP, some single outliers correspond to special cases in which the virtual cord needs to be used for routing instead of the optimal greedy routing between physical neighbors.

As a final measure, the number of messages transmitted is shown in Fig. 15. As can be seen, VCP is using roughly the same number of messages in case of no failing nodes. However, with an increasing number of node failures, VRR needs to transmit an increasing number of messages for ring maintenance. In contrast, the number of messages per node stays constant for VCP, that means that the total number of messages sent in the network is even decreasing (due to nodes being in off state). This number can roughly be translated to the energy performance of the protocol, because energy consumption is mainly a function of communication activities in WSNs.

7 Performance of Data Replication

In some final experiments, we evaluated the performance impact of data replication. As analyzed, for example, in the context of GHTs, a clear improvement can be expected in case of frequent node failures [29]. We specifically designed the simulation setup to make the results comparable to those published for GHT [28], [29]. In contrast to the previously described experiments, we performed continuous queries instead of pushing content to a dedicated node. Thus, we expect a much lower success ratio and increased communication delays. In particular, we placed 100 nodes on an area of 160 m $\times$ 160 m. After an initial setup time, two queries are being transmitted per second for a period of 300 s. We performed the experiments both for random and for grid deployment. The results were almost the same; therefore, only data for grid deployment are shown. Furthermore, we experimented with different node densities. It turned out that the most limiting factor is the connectivity in the network. If nodes are deployed too sparsely, isolated nodes or even islands appear, which may falsify the measurement results. For the presented simulation results, this number was zero or negligible in all the runs.

In order to evaluate the replication performance, we first analyzed the resulting query success rate. We show a comparison of the quality of all the implemented replication schemes in Fig. 16. As can be seen, the replication on the cord is less effective compared to the replication on physical neighbors. The worst case success rate can be seen in the scenario in which 100 percent of all the nodes continuously flip between on and off state. However, compared to the performance results reported for GHT [28], VCP performs very well—the results are compared numerically in Table 5. Exploiting the simple broadcast-based replication among nodes in the physical neighborhood, even better results can be achieved.

Such improvement in the success rate can be expected to be expensive in terms of some other metrics. Thus, we investigated other communication metrics such as the path length and the delay. Both kept almost constant, the path length at two to four hops (data not shown), and the delay increased slightly in the scenario with frequent node failures due to the additional effort for discovering alternate replicas (Fig. 7). All the graphs in Fig. 17 refer to measurements of data replication within the physical neighborhood (see Section 3.4). As previously discussed, the replication on the cord led to less effective improvements but the observed trend was similar.

Another interesting metric is the amount of messages needed to keep the replicas up to date. Fig. 7 shows the number of refresh messages, which is continuously increasing with the number of node failures. However, when comparing those to the total number of transmitted messages in Fig. 7, the additional effort seems to be adequate as it only represents about 20 percent of the total communication load. This is still a lot, however, if these data are of high potential interest, the effort can easily be motivated.

Finally, we investigated the storage requirements. In the best case, the number of stored messages would be insensitive to the number of node failures. As can be seen in Fig. 7, this is the case. On average, about five additional messages have to be stored on each node. The variance is also in a tolerable range. Only a few nodes have to store more than 15 messages, but those qualify as outliers in the statistical evaluation.

TABLE 5
Comparison of the Success Ratio and the Necessary Refresh Messages for VCP versus GHT

Failure ratio	GHT success rate	GHT refresh messages	VCP success rate	VCP refresh messages
0	100 %	1.6	100 %	0.00
20	99.7 %	1.5	99.2 %	0.02
40	98.6 %	1.6	95.2 %	0.03
60	97.3 %	1.8	95.3 %	0.05
80	94.2 %	1.8	95.2 %	0.08
100	83.3 %	1.6	92.1 %	0.08

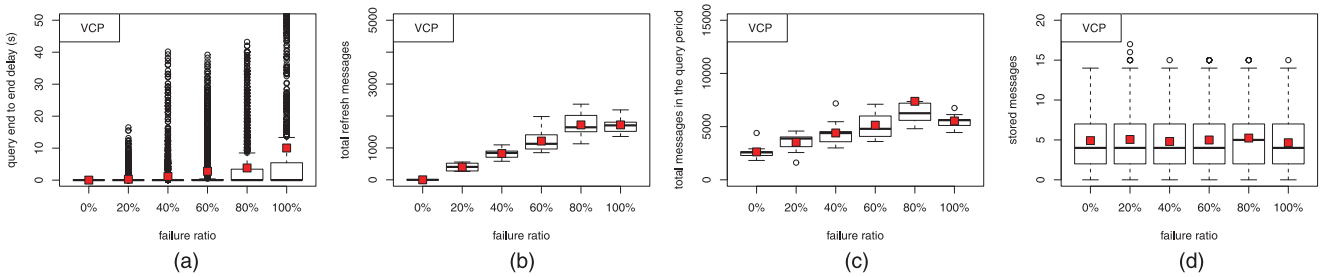

Fig. 17. Replication performance; depicted are the per-hop delay, transmitted and stored messages for the grid scenario. (a) Delay. (b) Refresh messages. (c) Total number of messages. (d) Storage requirements.

8 Conclusion and Future Work

Virtual Cord Protocol is a routing protocol designed for efficient data management in sensor networks. VCP exploits the benefits of using virtual coordinates for efficient routing. Using application-specific hash functions, data items can be associated with particular nodes. Furthermore, the protocol supports data replication in order to improve the failure performance.

In an extensive simulation-based performance evaluation, we assessed the impact of the network density, the traffic load, and potential node failures. We compared the routing performance of virtual address-based protocols, in particular VCP and VRR, with a typical ad hoc routing protocol (DYMO). The results clearly demonstrate the advantages of virtual coordinate-based approaches compared to classical ad hoc routing protocols in the optimal case, i.e., without any node failures. Both VCP and VRR, which show a comparable performance, outperform DYMO. However, in case of node failures, VCP demonstrates its strengths of efficient cord management. It shows a much better tolerance to node failures compared to VRR due to its low maintenance overhead. Based on these results and taking into account the capabilities of virtual address-based protocols to manage data using an application-specific hash function, we conclude that virtual coordinate-based approaches are better suited for WSNs, especially if these networks are dynamic w.r.t. node failures.

Finally, we evaluated the performance of the implemented data replication scheme. The resulting failure performance underlines the high quality of VCP for data management and routing in sensor networks; compared to GHTs, VCP achieves at least the same success rate with a much smaller number of refresh messages. Nevertheless, the additional effort for replication is not negligible but on an acceptable level (roughly 25 percent communication overhead).

Future work will include studies of the suitability of different hash functions for content replication in sensor networks. Recently, we also implemented VCP on real sensor nodes in our lab to verify its applicability on resource limited sensor nodes [27].

Acknowledgments

This work was partially supported by DAAD grant "Peer-to-peer techniques for sensor networks" under grant number 331 4 04 001. The manuscript is based on earlier work on VCP that was presented at IEEE MASS 2008 (basic protocol) and IEEE/IFIP WONS 2009 (comparison to VRR).

References

[1] M. Caesar, M. Castro, E.B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual Ring Routing: Network Routing Inspired by DHTs," *Proc. ACM SIGCOMM*, Sept. 2006.

[2] C.-H. Lin, B.-H. Liu, H.-Y. Yang, C.-Y. Kao, and M.-J. Tasi, "Virtual-Coordinate-Based Delivery-Guaranteed Routing Protocol in Wireless Sensor Networks with Unidirectional Links," *Proc. IEEE INFOCOM*, Apr. 2008.

[3] A. Awad, C. Sommer, R. German, and F. Dressler, "Virtual Cord Protocol (VCP): A Flexible DHT-like Routing Service for Sensor Networks," *Proc. Fifth IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS '08)*, pp. 133-142, Sept. 2008.

[4] M.-J. Tsai, F.-R. Wang, H.-Y. Yang, and Y.-P. Cheng, "VirtualFace: An Algorithm to Guarantee Packet Delivery of Virtual-Coordinate-Based Routing Protocols in Wireless Sensor Networks," *Proc. IEEE INFOCOM*, Apr. 2009.

[5] T. Watteyne, D. Simplot-Ryl, I. Augé-Blum, and M. Dohler, "On Using Virtual Coordinates for Routing in the Context of Wireless Sensor Networks," *Proc. 18th IEEE Int'l Symp. Personal, Indoor and Mobile Radio Comm. (PIMRC '07)*, pp. 1-5, Sept. 2007.

[6] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, pp. 393-422, 2002.

[7] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless Sensor Network Survey," *Computer Networks*, vol. 52, no. 12, pp. 2292-2330, Aug. 2008.

[8] F. Dressler, *Self-Organization in Sensor and Actor Networks*. John Wiley & Sons, Dec. 2007.

[9] K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325-349, 2005.

[10] C.E. Perkins and E.M. Royer, "Ad Hoc On-Demand Distance Vector Routing," *Proc. Second IEEE Workshop Mobile Computing Systems and Applications*, pp. 90-100, Feb. 1999.

[11] I. Chakeres and C. Perkins, "Dynamic MANET On-Demand (DYMO) Routing," Internet Draft (Work in Progress), draft-ietf-manet-dymo-10.txt, July 2007.

[12] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, T. Imielinski and H.F. Korth, eds., vol. 353, pp. 152-181, Kluwer Academic, 1996.

[13] B. Krishnamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," *Proc. Int'l Workshop Distributed Event Based System (DEBS '02)*, July 2002.

[14] R. Govindan, "Data-Centric Routing and Storage in Sensor Networks," *Wireless Sensor Networks*, C.S. Raghavendra, K.M. Sivalingam, and T. Znati, eds., pp. 185-205, Springer, 2004.

[15] J. Gehrke and S. Madden, "Query Processing in Sensor Networks," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 46-55, Jan.-Mar. 2004.

[16] *Peer-to-Peer Systems and Applications*, R. Steinmetz and K. Wehrle, eds. Springer, 2005.

[17] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp. 17-32, Feb. 2003.

[18] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware)*, pp. 329-350, Nov. 2001.

[19] B. Karp and H.T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. ACM MobiCom,* pp. 243-254, 2000.
[20] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," *Proc. ACM MobiCom,* Sept. 2003.
[21] M. Mauve, J. Widmer, and H. Hartenstein, "A Survey on Position-Based Routing in Mobile Ad-Hoc Networks," *IEEE Network,* vol. 15, no. 6, pp. 30-39, Nov./Dec. 2001.
[22] R. Flury, S.V. Pemmaraju, and R. Wattenhofer, "Greedy Routing with Bounded Stretch," *Proc. IEEE INFOCOM,* Apr. 2009.
[23] G. Tan, M. Bertier, and A.-M. Kermarrec, "Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing," *Proc. IEEE INFOCOM,* Apr. 2009.
[24] A. Awad, L.R. Shi, R. German, and F. Dressler, "Advantages of Virtual Addressing for Efficient and Failure Tolerant Routing in Sensor Networks," *Proc. Sixth IEEE/IFIP Conf. Wireless On Demand Network Systems and Services (WONS '09),* pp. 111-118, Feb. 2009.
[25] F. Dressler and F. Chen, "Dynamic Address Allocation for Self-Organized Management and Control in Sensor Networks," *Int'l J. Mobile Network Design and Innovation,* vol. 2, no. 2, pp. 116-124, 2007.
[26] C.N. Ververidis and G.C. Polyzos, "Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques," *IEEE Comm. Surveys and Tutorials,* vol. 10, no. 3, pp. 30-45, 2008.
[27] A. Awad, R. German, and F. Dressler, "Efficient Routing and Service Discovery in Sensor Networks Using Virtual Cord Routing," *Proc. Seventh ACM Int'l Conf. Mobile Systems, Applications, and Services (MobiSys '09),* June 2009.
[28] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table," *ACM/Springer Mobile Networks and Applications,* Special Issue on Wireless Sensor Networks, vol. 8, no. 4, pp. 427-442, Aug. 2003.
[29] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," *Proc. First ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA '02),* Sept. 2002.
[30] G. Tan, M. Bertier, and A.-M. Kermarrec, "Visibility-Graph-Based Shortest-Path Geographic Routing in Sensor Networks," *Proc. IEEE INFOCOM,* Apr. 2009.
[31] K. Liu and N. Abu-Ghazaleh, "Aligned Virtual Coordinates for Greedy Routing in WSNs," *Proc. Third IEEE Int'l Conf. Mobile Ad Hoc and Sensor Systems (MASS '06),* pp. 377-386, Oct. 2006.
[32] B. Leong, B. Liskov, and R. Morris, "Greedy Virtual Coordinates for Geographic Routing," *Proc. 15th IEEE Int'l Conf. Network Protocols (ICNP '07),* pp. 71-80, Oct. 2007.
[33] Y. Zhao, Y. Chen, B. Li, and Q. Zhang, "Hop ID: A Virtual Coordinate-Based Routing for Sparse Mobile Ad Hoc Networks," *IEEE Trans. Mobile Computing,* vol. 6, no. 9, pp. 1075-1089, Sept. 2007.
[34] J. Newsome and D.X. Song, "GEM: Graph Embedding for Routing and Data-Centric Storage in Sensor Networks without Geographic Information," *Proc. First ACM Conf. Embedded Networked Sensor Systems (SenSys '03),* pp. 76-88, Nov. 2003.
[35] C.T. Ee, S. Ratnasamy, and S. Shenker, "Practical Data-Centric Storage," *Proc. Third Symp. Networked Systems Design and Implementation (NSDI '06),* pp. 325-338, May 2006.
[36] Ö.D. Incel and B. Krishnamachari, "Enhancing the Data Collection Rate of Tree-Based Aggregation in Wireless Sensor Networks," *Proc. Fifth IEEE Comm. Soc. Conf. Sensor and Ad Hoc Comm. and Networks (SECON '08),* pp. 569-577, June 2008.
[37] A. Varga, "The OMNeT++ Discrete Event Simulation System," *Proc. European Simulation Multiconf. (ESM '01),* June 2001.
[38] F. Chen, N. Wang, R. German, and F. Dressler, "Simulation Study of IEEE 802.15.4 LR-WPAN for Industrial Applications," *Wireless Comm. and Mobile Computing,* vol. 10, no. 5, pp. 609-621, May 2010.
[39] C. Sommer, I. Dietrich, and F. Dressler, "Simulation of Ad Hoc Routing Protocols Using OMNeT++: A Case Study for the DYMO Protocol," *ACM/Springer Mobile Networks and Applications,* Special Issue on Simulation Techniques and Tools for Mobile Networking, doi:10.1007/s11036-009-0174-5, 2009.

Abdalkarim Awad received the BS degree in electrical engineering and the MSc degree in scientific computing from Birzeit University in 1999 and 2003, respectively. In 2006, he joined the Networking Group in the Department of Computer Science, University of Erlangen, Germany, supported by a DAAD scholarship. In 2009, he received the PhD degree from the School of Engineering, University of Erlangen, Germany. Now, he works as a research associate in the Integrated Communication Systems Group with the Faculty of Computer Science and Automation, Ilmenau University of Technology. His current research interests are focused on self-organizing methodologies and their evaluation techniques. He uses these methodologies to solve problems in computer networks and communication systems. His research interests include routing and data management in sensor networks, peer-to-peer systems, and control of distributed systems.

Reinhard German received the diploma in 1991, the PhD degree in 1994, and the Habilitation degree in 2000 from the Computer Science Department, Technical University of Berlin. Then, he joined the Department of Computer Science at the University of Erlangen-Nuremberg, first as an associate professor (system simulation), and since 2004, as a full professor (computer networks and communication systems), where he is currently the head of the department. His research interests include model-based and measurement-based performance analysis, modeling and simulation paradigms and tools, numerical analysis of Markovian and non-Markovian models, vehicular communications, and autonomous sensor/actuator networks.

Falko Dressler received the MSc and PhD degrees from the Department of Computer Science, University of Erlangen, in 1998 and 2003, respectively. In 2003, he joined the Computer Networks and Internet Group at the Wilhelm-Schickard-Institute for Computer Science, University of Tuebingen. Between 2004 and 2011, he has been an assistant professor at the Computer Networks and Communication Systems Chair in the Department of Computer Science, University of Erlangen, coordinating the Autonomic Networking Group. He is now a full professor of computer science heading the Computer and Communication Systems Group at the Institute of Computer Science, University of Innsbruck. He teaches on self-organizing sensor and actor networks, network security, and communication systems. Dr. Dressler is an editor for journals such as Elsevier's *Ad Hoc Networks*, ACM/Springer's *Wireless Networks* (WINET), and Elsevier's *Nano Communication Networks*. He was a guest editor of special issues on self-organization, autonomic networking, and bio-inspired computing and communication for the IEEE's *Journal on Selected Areas in Communications* (JSAC), Elsevier's *Ad Hoc Networks*, and others. Dr. Dressler was the general chair of IEEE/ACM BIONETICS 2007 and IEEE/IFIP WONS 2011. Besides chairing a number of workshops associated with high-level conferences, he regularly acts in the TPC of leading networking conferences such as IEEE INFOCOM, IEEE ICC, IEEE Globecom, IEEE WCNC, and IEEE MASS. Among others, he wrote the textbook *Self-Organization in Sensor and Actor Networks* (Wiley, 2007). Dr. Dressler is an IEEE Distinguished Lecturer in the fields of inter-vehicular communication, self-organization, and bio-inspired networking. He is a senior member of the IEEE (COMSOC, CS, VTS) as well as a senior member of the ACM (SIGMOBILE), and member of GI (KuVS). He actively participates in the IETF standardization. His research activities are focused on adaptive wireless networking and self-organization methods addressing issues in wireless ad hoc and sensor networks, inter-vehicular communication systems, bio-inspired networking, and adaptive network security techniques.