Reactive Virtual Position-Based Routing in Wireless Sensor Networks

Abdalkarim Awad*, Andreas Mitschele-Thiel* and Falko Dressler†
*Dept. of Computer Science, Ilmenau University of Technology, Germany
†Institute of Computer Science, University of Innsbruck, Austria

***Abstract*—Virtual position-based routing protocols have many attractive characteristics for wireless sensor networks. Typically, such protocols use a proactive scheme for updating routing tables. Because sensor networks can have very low data rate, sending periodic beacons to update routing tables can be very expensive. Instead, reactive approaches might be more appropriate in such scenarios. MANET-inspired reactive routing protocols do not scale well because of the effort in the order of $O(n)$ for each routing information update. In this paper, we present Reactive Virtual Cord Protocol (RVCP), a data-centric reactive virtual position based routing protocol for use in sensor networks. Route discovery is directed towards the destination and hence there is no need to flood the entire network to discover a route. Our approach is based on Virtual Cord Protocol (VCP), an efficient, virtual relative position based routing protocol that also provides support for data management as known from typical Distributed Hash Table (DHT) services. To minimize the end-to-end delay and energy consumption, we used adaptive techniques for the development of RVCP.**

I. Introduction

New advances in the technology make it possible to produce small size computing devices that can sense the surrounding environment. These devices are called sensor nodes and are equipped with a wireless communication channel. Wireless Sensor Networks (WSNs), which are composed of cooperating of sensor nodes, present a cost-effective, practical, and capable solution to support many application scenarios. Sensor nodes typically resource limited devices, but WSN application domains are diverse and they have to manage a variety of data types, including simple temperature, light intensity, and humidity measures; or more complex data types such as sound, images, or even video. It is foreseen that the size of such network will be very large and hence efficient routing is a crucial issue in these networks [1]. One of the key challenges in WSN routing design is how to guarantee packet delivery, i.e. to ensure that a path to any destination can be found [2], [3], in combination with an optimized network lifetime [4].

Traditionally, routing algorithms can be classified as proactive or reactive. Proactive algorithms, also called global state algorithms, employ classical routing strategies such as distance-vector routing (e.g., Destination-Sequenced Distance Vector (DSDV) [5]) or link-state routing (e.g., Optimized Link State Routing (OLSR) [6]). Those approaches maintain routing information in the network even if these paths are currently not being used. Moreover, they usually rely on flooding techniques for updating in case of topology changes. On the other hand, reactive algorithms have been developed as a response to this observation (e.g., Dynamic Source Routing (DSR) [7], Ad Hoc on Demand Distance Vector (AODV) [8], and Dynamic MANET on Demand (DYMO) [9]). Reactive routing protocols use on-demand route acquisition systems, where a node sends a route request (RREQ) whenever it needs to send a message to a node for which a route not already exists. Reactive routing protocols are generally more scalable, since they generate less network traffic. They are thus suitable for highly dynamic ad hoc networks [10]. However, maintaining routes only while in use leads to a delay for the first packet to be transmitted.

Besides these classic Mobile Ad Hoc Network (MANET) based approaches, geographic or location based routing algorithms have been developed to eliminate some limitations using the positions of all nodes. Geographic routing algorithms can be divided mainly into two types: real location based and virtual location based. For the first type, it is necessary that each node knows its exact physical location. Commonly, it is assumed that each node determines its position using Global Positioning System (GPS) or some other type of positioning service [11]–[13]. For the second type, a virtual location is allocated to each node.

Approaches like Geographic Hash Table (GHT) [14] map the data items to real locations in the network. Thus, GHTs assume that each node know its physical location and the area is pre-defined before deploying the network. As geographic coordinates are sometimes not available, error-prone, and the used greedy routing might end up in dead ends, transformations have been proposed to create a modified virtual topology [15]. Furthermore, virtual coordinate based solutions such as Virtual Ring Routing (VRR) [16] or Virtual Cord Protocol (VCP) [3], [17] rely on a virtual topology created and maintained for efficient routing and data management.

One of the most recent approaches is VCP [3], a DHT-like protocol in which all data items are associated with numbers in a pre-determined range $[S, E]$, i.e. on an one-dimensional cord. All the available nodes capture this range by having a unique ID in the range $[S, E]$. Thus, each node in the network is responsible for a portion of the entire space defined by its relative position to physical neighbors. This way, it is possible to store data on the nodes by mapping data items deterministically in space using a hash function. The corresponding key-value pair is then stored at the node whose position is closest to the key. Greedy routing is performed based only on the position of the physical neighbors. To retrieve data items, nodes have to apply the same hash function to find the

978-1-4577-0636-3 /11/$26.00 ©2011 IEEE

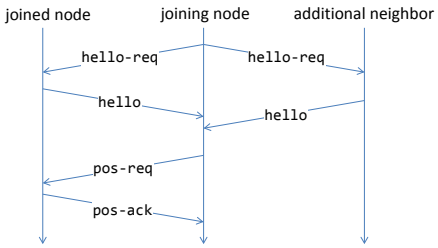


Fig. 1. Join process: message exchange during the join process

key value. They then can route the request to the node whose position is closest to the key. Yet, still this protocol is proactive, thus, consuming a non-negligible amount of energy for topology maintenance, and reducing the network lifetime [4].

We investigated the performance of VCP assuming each node sends a periodic hello message to update network topology. In order to increase the network lifetime, we developed the Reactive Virtual Cord Protocol (RVCP), an extended version of VCP to support reactive node joins and updates of routing tables. In this paper, we describe the findings of our performance evaluation and present the reactive routing scheme. In detailed evaluations, we compared the new RVCP protocol to the standard VCP as well as to typical MANET routing.

II. Reactive Join Process

In this section, we describe the join process in our new Reactive Virtual Cord Protocol (RVCP). As stated previously, RVCP is a DHT-like protocol. All data items are associated with numbers in a pre-determined range $[S, E]$ and the available nodes capture this range. Thus, each node captures a part of the entire range.

When a node joins the RVCP network, it must set three important variables: its position, predecessor, and successor in the virtual cord. Each node determines these values based on the positions of its single-hop neighbors. One `hello` messages from the neighboring nodes is sufficient to set these parameters. At network startup, one node must be pre-programmed as initial node, i.e. it gets the position S. The joining node has to discover the network structure, i.e. all neighboring nodes and their position in the cord. As shown in Figure 1, the joining node sends a request message (`hello-req`) in RVCP. This is answered by the nodes that already joined the network by sending `hello` messages. All these messages are broadcasted to all neighbors. To avoid collisions that can occur during the `hello` exchange, each node should wait a random time (backoff) before starting to send the `hello` message. In such a self-organizing system, it may happen that more than one node in the same physical region tries to join at the same time. Therefore, we used position request messages (`pos-req`) and the respective acknowledgments (`pos-ack`), which guarantee that each node will get a unique position in the cord.

Based on the received `hello` messages, the joining node gets information about its physical neighbors and their adjacent nodes. If the node sending the `hello` is not yet in the physical neighbors table, it will be added. Otherwise, the entry is updated

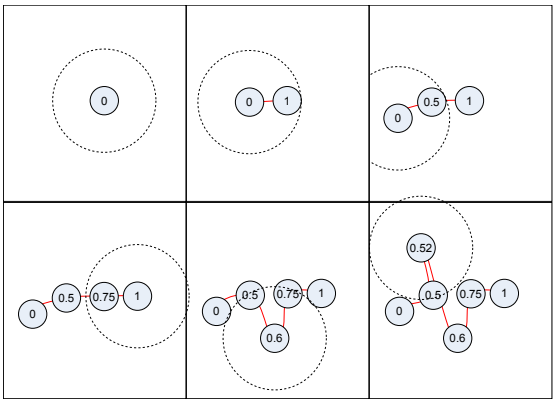


Fig. 2. Join operation in VCP and RVCP

accordingly. Of course, the `hello` message updates also the successor and predecessor information. If the node has not yet joined the network, it internally starts the join process based on the received information – a join delay T_{ps} must have elapsed before re-asking for a relative position.

The cord setup follows the same rules used in the standard VCP protocol [3]: If a node can communicate with an end node, 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 the its old position. The new node becomes predecessor of the old node if it received position S. Otherwise it becomes its successor.

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. Additionally, 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.

Finally, if the new node can communicate with only one node in the network, which is neither at S nor E, then the new node asks that node to create a virtual position. 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 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 setup a virtual position. In previous experiments, we discovered that fewer virtual nodes lead to better routing paths [17].

Figure 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 five steps, nodes are placed in the cord according to the simple rule to create new addresses either at an end or in the middle of the cord. In the sixth step, a virtual node is created to join node 0.52.

III. Reactive Routing Table Update

In RVCP, packets are marked with their destinations' locations. As a result, a forwarding node can make a local choice

when greedily choosing a packet's next hop: the neighbor with an ID closest to the packet's destination. This procedure is repeated until the destination is reached. Thus, for routing in RVCP, each node must know its physical neighbors, extracted from the neighbors' `hello` messages.

The routing tables are maintained as follows: When a node receives a data packet, it verifies the corresponding routing table entries. If the routing table is older than RTE, it is considered stale. RVCP depends on timeouts and not on link-layer notifications or probing which makes it independent of the implementation of the lower layers. If routing information is stale, the node starts a new `hello` exchange. The node sends a `hello-req` and waits for a short time Hd, before sending further data packets. This make it possible for the node to receive a maximum number of its neighbors' `hello` messages. Hd is adaptively adjusted depending on the network conditions. We measure it as the minimum time that elapsed between sending a `hello-req` and receiving a `hello` message over the last $n \times RTE$ seconds. That way we decrease the end-to-end delay, because if the network is free, Hd has a lower value. Furthermore, less hello messages will be sent when we have adaptive Hd.

If the node did not receive new routing information after sending the `hello-req`, then it stores the data packet temporally until routing information becomes available. Each node cal locally determine whether it is responsible for some data item, relying on the successor and predecessor information. A stabilization function runs every T_{stab} that detects existing packets that should be further forwarded. Algorithm 1 depicts the routing process. If the routing information are valid, a greedy algorithm is employed to send packets to the physical neighbor that has a cord position closest to the destination until no more progress is possible and the value lies between the positions of the predecessor and successor.

RVCP inherently relies on a previously established cord and, thus, provides guaranteed delivery. In addition, RVCP supports shortcuts whenever a physical neighbor with a virtual number is available that is closer to the destination. In Section IV, we show that the used routes are close to the shortest path.

Algorithm 1 Reactive Greedy Forwarding Algorithm

Require: Received data packet D for destination position D_p, locally maintained data set $[P_{min}, P_{max}]$

```
if P_min ≤ D_p ≤ P_max then
    StoreData(D_p)
else if Hd ≤ ΔT ≤ RTE then
    if ∃N_i ∈ N : |Position(N_i) − D_p| < |P − D_p| then
        Send(N_i, D)
    else
        StoreData(D_p)
    end if
else if ΔT > RTE then
    Send(hello-req)
end if
```

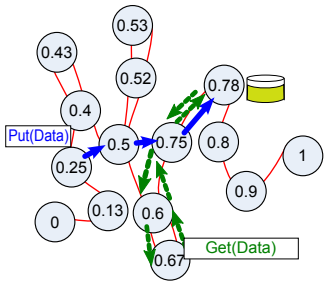


Fig. 3. An example for a inserting / retrieving data using the virtual cord and greedy routing exploiting local neighborhood information

Theorem 1. *Given a static network, VCP's greedy forwarding based on local information guarantees packet delivery to the correct destination.*

Proof: Assume a packet, marked by its originator with key K, gets stuck at a node that is not closest to K, say N, and there exists another node N' that is closest to K. Recall that in VCP each node can communicate at least with its predecessor and successor and the value of the successor is larger than N and the value of the predecessor is smaller than N. Therefore either the successor or the predecessor is closer to N' than N (if not, then $N = N'$). Hence the packet can be greedily forwarded either to the successor or the predecessor until it reaches a node for which neither the successor nor the predecessor is closer to K than the node itself. ■

Consider the example shown in Figure 3. If node 0.25 produces a data item, it first prepares the corresponding hash value – we assume a hash value of 0.781. Node 0.25 will forward the message towards the destination node, i.e. greedily to the physical neighbor closest to the identified hash value. In our case, this is node 0.5. Afterwards, the message will be forwarded to node 0.75, and then to node 0.78 as illustrated in Figure 3. Node 0.78 will finally store the data and will not forward it any further because there is no more possible progress and the value lies between the positions of the predecessor and the successor.

It is important here to say that the update of routing tables occurs only at the nodes on the routing path and not in the entire network. The great advantage of greedy forwarding is its reliance on only knowledge of the forwarding node's immediate neighbors. The amount of state information that needs to be tracked is negligible and dependent on the node density but not on the total number of nodes in the network.

The scalability can by analyzed as follows: The most obvious measure of scalability of a routing protocol is the overhead associated with the maintenance of routing tables. In a sensor network, two measures are important: the size of the routing table and the communication overhead required to keep it up to date. The size is not only refers to the memory size required to store the routing table, but also to how many entries of the table need to be adjusted when nodes join or leave. The communication overhead indicates how much communication is required to update each entry. In RVCP, the routing table

of each node contains only its radio neighbors, hence the size is $O(m)$ where m is the node degree and $m << n$, n being the total number of nodes in the network. Since one `hello` message is sufficient to update each entry in the routing table, the communication overhead to update each entry is again $O(m)$. Thus, RVCP has excellent scalability characteristics.

IV. Performance Evaluation

In this section, we evaluate RVCP for different network conditions and compare it to other protocols. We performed several experiments using OMNeT++, which is an open source discrete event simulator free for academic use. It is component-based C++ simulation environment, it is gaining wide acceptance in the scientific community for building and simulating communication systems. We executed a large set of experiments to explore the impact of various parameter settings, including the network size and the offered traffic load.

A. Analytical comparison with a standard reactive protocol

First, we analytically compare the routing cost of RVCP to the *de-facto* standard in reactive routing, DYMO [9]. Routes in DYMO are discovered on-demand when a node needs to send a packet to a destination currently not in its routing table. A Route Request (RREQ) message is flooded through the network using broadcast messages. If the packet reaches its destination (or a node that cached corresponding path), a Route Reply (RREP) message is sent back containing the requested path information. Each node maintains a local sequence number, which is incremented each time the node sends a RREQ. This allows other nodes to determine the order of discovery messages to avoid stale routing information, to detect duplicate messages, and to ensure loop free routing.

RVCP requires an initialization phase in which each node gets is virtual relative position. For a network of size n, the initialization phase requires $O(n)$ messages. Afterwards, update messages are needed in case the routing information is stale. This requires $O(\sqrt{n})$ and hence the cost C_{RVCP} for sending D items that require update of routing information is:

$$C_{\text{RVCP}} = n + D \times \sqrt{n} \tag{1}$$

On the other side, DYMO requires no initialization phase, but for each routing update it requires $O(n)$ messages and therefore the cost C_{DYMO} for sending D items that require update of routing information is:

$$C_{\text{DYMO}} = 0 + D \times n \tag{2}$$

From this comparison we conclude that standard protocols like DYMO are only efficient for a very low number of messages. If, however, the number of forwarded data items is large, RVCP is clearly outperforming such protocols.

B. Simulation environment

We implemented a simulation model of the RVCP in the OMNeT++ simulation toolkit. We also used the INET framework that provides detailed simulation models of typical network protocols. In our simulation, we used RVCP on the top of the IEEE 802.11 Wireless LAN protocol. The communication range is about $40\,\text{m}$. We investigated our approach for network sizes between 100 and 400 nodes. We also adjusted the plane dimensions to keep the density of nodes per square meter constant. Additionally, we evaluated networks with different node degrees. To explore the effect of the used data rate, we also varied the data rate between 0.05 and $1\,\text{pps}$. In the simulation experiments, we measured the end-to-end delay, mac-layer collisions, and total number of broadcast messages. End-to-end delay represents the latency that a message experiences from source to destination, whereas the number of collisions and broadcasts indicate the power consumption of a sensor node. A summary of our simulation scenarios is shown in Table I. For statistical confidence, we executed each experiment at least 5 times with different random seeds.

TABLE I
Summary of simulation scenarios

Input Parameter	Value
Number of Nodes	100–400
Playground size	$200\,\text{m} \times 200\,\text{m}$ to $600\,\text{m} \times 600\,\text{m}$
Node placement	Grid, Random
Data rate	CBR, $1\,\text{pps}$, $0.2\,\text{pps}$, $0.1\,\text{pps}$, and $0.05\,\text{pps}$
Initialization time	$40\,\text{s}$
Start of data transmission	uniformly distributed in $[100, 120)\,\text{s}$
End of data transmissions	$200\,\text{s}$
Destination node	Upper left node or Random

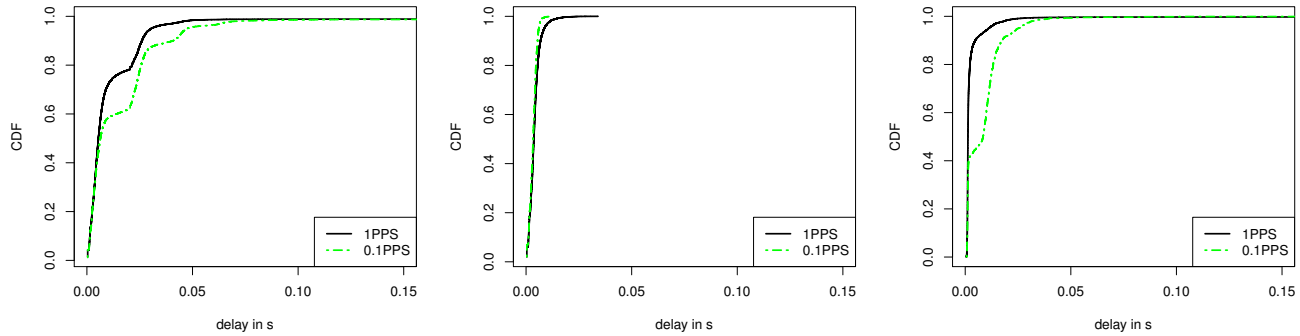
We visualized the results using the empirical cumulative density function (CDF) and box plots. CDF is a useful way to show the distributions of the output. We chose box plots because they are more robust in the presence of outliers than the classical statistics based on the normal distribution. Boxplots indicate the median value and the quartiles. Additionally, we plot the average as small box.

C. Simulation comparison with a standard reactive protocol

We started our simulation experiments by comparing the performance of RVCP to the proactive version VCP [3], [17], and to the reactive protocol DYMO [9], [18].

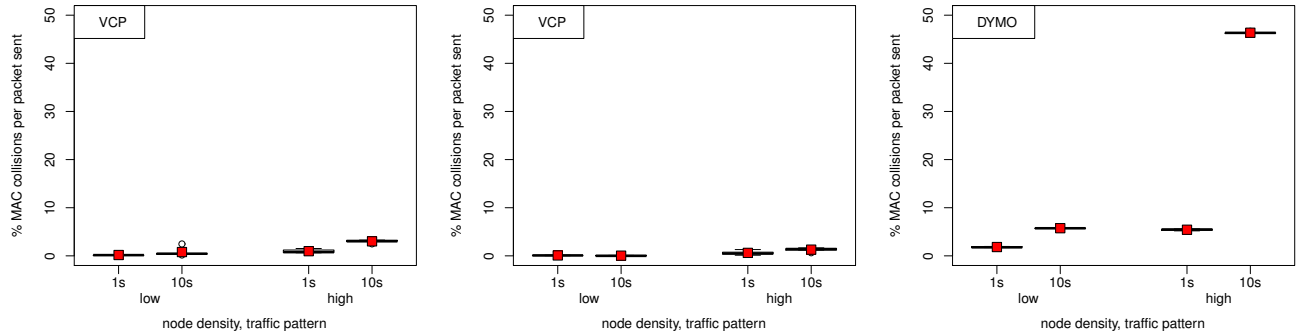
In this comparison, we placed 100 nodes in a rectangular area in form of a grid. We varied the node density and the traffic rate. We configured an average node degree of 16 for the high density scenario, and an average node degree of 4 in the low density scenario. We investigated two traffic rates of $1\,\text{pps}$ and $0.1\,\text{pps}$. Each node in the network sends packets to the same destination, which is equivalent to store the same data type collected from all the sensor nodes at a base station.

Figure 4 shows that the measured latency of RVCP and of DYMO is in the same range. In both protocols, most of the packets (more than $90\,\%$) experience an end-to-end delay of less than $0.05\,\text{s}$. The reason that some packets show a higher delay is the reactive nature of these protocols, which implies updating the routing information before sending data packets. Because of the proactive nature of VCP, it results in much lower end-to-end delays: all data packets have been sent with



(a) RVCP (b) VCP (c) DYMO

Fig. 4. End-to-end delay



(a) RVCP (b) VCP (c) DYMO

Fig. 5. MAC layer collisions

a delay of less than 0.05 s.

Figure 5 shows the ratio of MAC collisions per link-layer packet sent for nodes deployed in a grid for all the three protocols. As can be seen, only in the case of a high node density and a mean interval between two application-layer messages of 10 s, the collision ratio in the DYMO experiments exceeds 40 %. MAC collisions in all other scenarios were reaching only insignificant ratios. VCP keeps the collision ratio in almost all experiments close to 0 %. In the low load scenario (one message every 10 s) with a high density, the collision ratio for RVCP is at about 2 %, caused by the necessary route updates.

D. Performance with different network size

We further evaluated the performance of the proactive and the new reactive variants of the VCP protocol as the number of nodes increases while keeping the traffic load offered by each node constant. We varied the number of nodes from 100 to 400. Each node generated one packet every 10 s to a randomly chosen destination. Figures 6(a) and 6(b) show the results for the latency and number of broadcast messages, respectively.

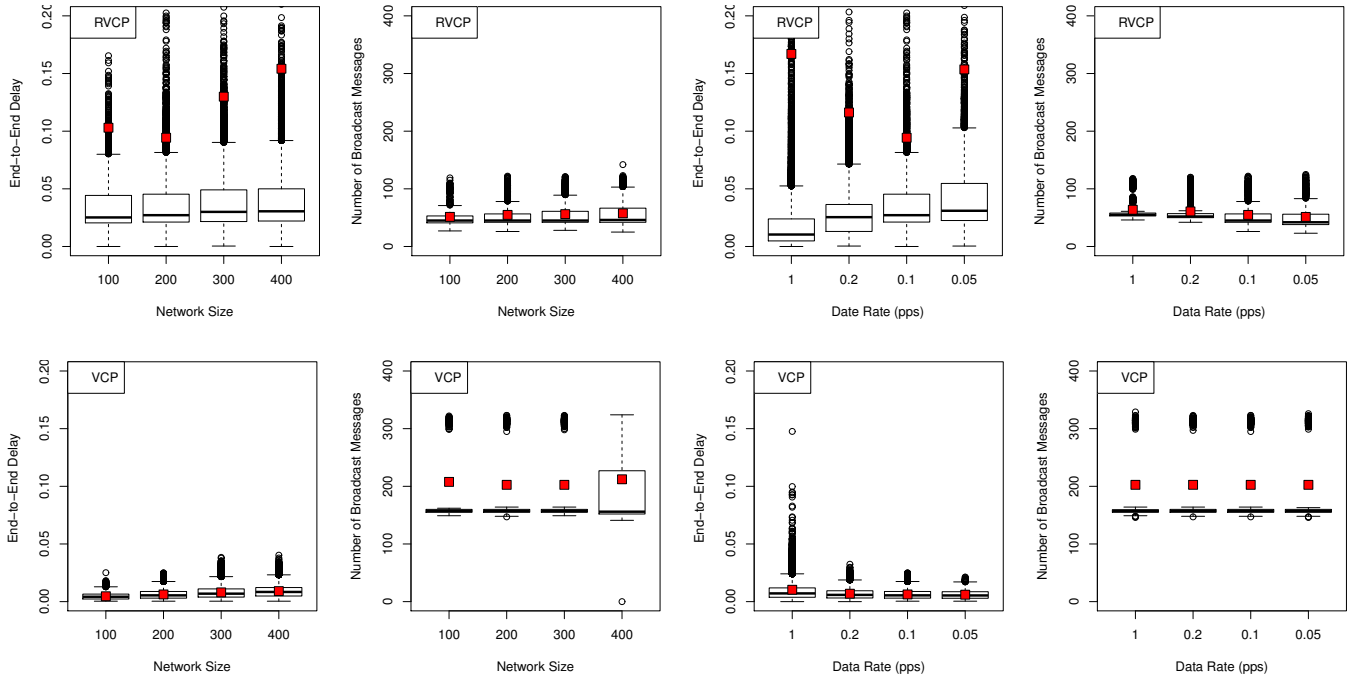
Figure 6(a) shows that VCP achieves lower delays compared to RVCP for all network sizes because it never has to queue packets waiting for route updated to complete. In fact, the end-to-end delay of VCP is proportional to the path length as a result of only the propagation delay. The delay of RVCP is larger due to the need for routing information updates. Please note that we selected a low load scenario to allow routes to timeout. On the other hand, RVCP produces significantly less broadcast messages compared to VCP as shown in Figure 6(b), because it sends `hello` messages on demand. The success ratio was close to 100 % for all network sizes.

E. Performance with different network traffic

In this set of experiments, we compared the performance of the RVCP and VCP routing protocols with increasing traffic load while keeping the size of the network constant at 200 nodes, deployed randomly in a rectangular playground of size 300 m × 300 m. Each node sends packets to random destinations at the different packet rates.

In RVCP, the delay as shown in Figure 6(c) increased inverse proportional with the data rate. This effect can be explained by the route timeouts used by RVCP. In low data rate scenarios, RVCP has to set up a route for almost every packet because the available routes have timed out. On the other hand, the delay for VCP slightly decreases. This effect is caused by the congestion in the network.

Figure 6(d) shows the number of broadcast messages sent by each node. It is clear that the RVCP causes much less broadcast messages and the number of these messages depends on the traffic in the network. On the other hand, VCP sends constant amount of broadcast messages regardless of the normal traffic in the network.



(a) End-to-end delay for different network sizes (b) Number of broadcast messages per node for different network size (c) End-to-end delay for different data rates (d) Number of broadcast messages per node for different data rates

Fig. 6. Comparison between RVCP and VCP for different network sizes and data rates

V. Conclusion

In this paper, we presented Reactive Virtual Cord Protocol (RVCP), a fully reactive data-centric routing protocol that minimizes the number of necessary broadcast messages. The join process as well as the update of the routing table are done on demand. There are several advantages of RVCP compared to other reactive protocols. First, updating routing table does not require flooding the entire network, instead a directed `hello-req` message toward the destination is employed. Secondly, RVCP do not require initial node addresses that are unique in the entire network. Additionally, it provides typical DHT functionality for data management. We evaluated RVCP in the context of wireless sensor networks. The simulation results show that RVCP provides stable performance across wide range of parameter settings.

References

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges," *Elsevier Ad Hoc Networks*, vol. 2, pp. 351–367, October 2004.

[2] M.-J. Tsai, H.-Y. Yang, B.-H. Liu, and W.-Q. Huang, "Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1228–1241, 2009.

[3] A. Awad, R. German, and F. Dressler, "Exploiting Virtual Coordinates for Improved Routing Performance in Sensor Networks," *IEEE Transactions on Mobile Computing*, 2010, available online: 10.1109/TMC.2010.218.

[4] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 1, pp. 1–39, February 2009.

[5] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Computer Communications Review*, pp. 234–244, 1994.

[6] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *IEEE INMIC 2001*, Lahore, Pakistan, December 2001, pp. 62–68.

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

[8] C. E. Perkins and E. M. Royer, "Ad hoc On-Demand Distance Vector Routing," in *2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90–100.

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

[10] R. Bhaskar, J. Herranz, and F. Laguillaumie, "Efficient Authentication for Reactive Routing Protocols," in *IEEE AINA-06*, vol. 2. Vienna, Austria: IEEE, April 2006, pp. 57–61.

[11] S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-Free Positioning in Mobile ad-hoc Networks," in *HICSS 2001*, Big Island, Hawaii, January 2001.

[12] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The anatomy of a context-aware application," *ACM/Springer Wireless Networks*, vol. 8, pp. 187–197, March 2002.

[13] N. B. Priyantha, A. K. Miu, H. Balakrishnan, and S. Teller, "The cricket compass for context-aware mobile applications," in *ACM MobiCom 2001*, Rome, Italy, July 2001, pp. 1–14.

[14] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," in *ACM WSNA 2002*, Atlanta, Georgia, September 2002.

[15] R. Flury, S. V. Pemmaraju, and R. Wattenhofer, "Greedy Routing with Bounded Stretch," in *28th IEEE Conference on Computer Communications (INFOCOM 2009)*. Rio de Janeiro, Brazil: IEEE, April 2009.

[16] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual Ring Routing: Network routing inspired by DHTs," in *SIGCOMM 2006*, Pisa, Italy, September 2006.

[17] A. Awad, C. Sommer, R. German, and F. Dressler, "Virtual Cord Protocol (VCP): A Flexible DHT-like Routing Service for Sensor Networks," in *IEEE MASS 2008*. Atlanta, GA: IEEE, September 2008, pp. 133–142.

[18] C. Sommer, I. Dietrich, and F. Dressler, "A Simulation Model of DYMO for Ad Hoc Routing in OMNeT++," in *ACM/ICST SIMUTools 2008, OMNeT++ Workshop*. Marseille, France: ACM, March 2008.