Triangle Routing in Wireless Sensor Networks with Unidirectional Links Revisited - a Look at Different Scenarios

Reinhardt Karnapke and Jörg Nolte
Distributed Systems/Operating Systems Group
Brandenburg University of Technology Cottbus - Senftenberg
Cottbus, Germany
Email: {Karnapke, Jon}@informatik.tu-cottbus.de

***Abstract*—Experiments with wireless sensor networks have shown that links are often asymmetric or unidirectional. This represents a serious problem for many routing protocols, which often depend on bidirectional links. Routing protocols that can use unidirectional links often induce a high overhead. To overcome this problem we introduced Unidirectional Link Triangle Routing, a routing protocol, which uses neighborhood information, gathered actively or passively, to route around unidirectional links. In this paper, we describe Unidirectional Link Triangle Routing in further detail and present additional evaluation results from different application scenarios.**

Keywords - Wireless Sensor Networks, Routing, Unidirectional Links

I. Introduction

This paper is an extended presentation of Unidirectional Link Triangle Routing (ULTR) [1], a routing protocol designed specifically to enable the usage of unidirectional links, which was first published at Sensorcomm 2013.

A unidirectional link from a node A to a node B exists, when node B can receive messages from node A, but node A cannot receive messages from node B. Even though this definition seems straightforward, there are different definitions of asymmetric and unidirectional links in literature. This is partially due to the fact that links change over time. Definitions are often based on different packet reception rates, and a difference of, e.g., more than 90% signifies a unidirectional link. In the same definition, a difference of less than 10% signifies a bidirectional link while all other links are called asymmetric. However, this is just one of the many different definitions for unidirectional, asymmetric, and bidirectional links. In others, only two classes (bidirectional and asymmetric) exist or different percent values are used.

No matter, which definition is given, there is a large number of unidirectional links present in current sensor networks as many experiments have shown (e.g., [2], [3], [4], [5]).

Existing protocols for wireless sensor networks or Mobile Ad-Hoc Networks still lack the ability to handle unidirectional links in an efficient manner. Many of those protocols deal with unidirectional links by removing their negative impact on the routing tables, while some of them use unidirectional links explicitly. Making these unidirectional links usable introduces overhead, which has to be weighted against the gain.

TABLE I
Routing tables in ULTR

Destination	Next Hop	Link Status	Forwarder
D	A	bidirectional	none
E	B	incoming	C
F	G	outgoing	none

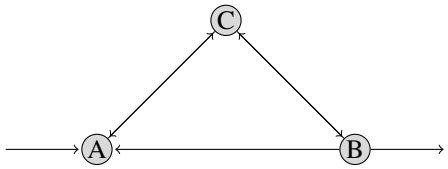
The authors of [6] have evaluated some of the existing protocols and concluded that the gain in connectivity is not worth the cost. While this might have been true for their scenario and the protocols they evaluated, it is possible to have scenarios where network separation occurs when unidirectional links are eliminated. It is also possible to have protocols that induce less overhead than the ones they considered.

Even though ULTR requires neighborhood information, the overhead induced can be kept small, either by gathering information passively, or by using information that is already provided from other communication layers. ULTR has already been published in, this paper takes a closer look at implementation details and offers additional insights into the usability of ULTR in different scenarios.

This paper is structured as follows: Section II describes ULTR in detail before a closer look at the cooperation between MAC and routing is taken in Section III. Section IV describes the general methodology of the evaluation, followed by Section V with details on the simulation setup and Section VI that describes hard- and software used in the real world experiments. The three application scenarios (sense-and-send, single pairing and multiple pairings) along with the evaluation results for simulations and real experiments are presented in Sections VII to IX. We finish with a conclusion in Section X.

II. Unidirectional Link Triangle Routing

In ULTR, neighborhood information is needed. To make a neighborhood table entry on a node A usable for ULTR, it must at least contain the ID of the neighbor (e.g., B), the status of the link to that neighbor (bidirectional, unidirectional-incoming or unidirectional-outgoing) and, if the link is unidirectional-incoming, the identity of another neighboring node (e.g., C), which can be used to forward data to the node in question (node B). Table I shows an example for all three kinds of links.



ID, Status, Forwarder
B, incoming, C
C, bidirectional, none

A

ID, Status, Forwarder
A, outgoing, none
C, bidirectional, none

B

ID, Status, Forwarder
A, bidirectional, none
B, bidirectional, none

C

Fig. 1. Neighbor table entries in ULTR

When a node wants to transmit a message to another node that is not included in its neighbor table or its routing table, it starts a route discovery by transmitting a route request (RREQ) message. This message is flooded through the network and creates routing entries for the source on all nodes it passes. The entries include only the next hop and the distance, resulting in a distance-vector protocol like, e.g., AODV [7].

However, the handling is different once the destination has been reached and transmits the route reply. When a node receives a message that is not flooded, i.e., a route reply (RREP) or DATA message, it checks its routing table to find out, which of its neighbors is the intended next hop just like in AODV. Unlike AODV, there is another step after that one. Once the node knows the neighbor that has been chosen to forward the message, it checks its neighbor table to see if the link to that node is *currently* a unidirectional-incoming one. If it is, and a detour of one hop is possible, the node forwards the packet first to the detour node, which, in turn, retransmits the message to the intended node. Otherwise, the message is silently discarded. Please note that broken links can be treated just like unidirectional-incoming ones.

Figure 1 shows a small part of a network and the corresponding neighborhood table entries used in this protocol: The nodes A, B and C from the example above are connected bidirectionally, with the exception of the link between nodes A and B, which is unidirectional, enabling only transmissions from B to A. The neighborhood table of node A consists of two entries, a bidirectional one for node C and a unidirectional-incoming one from node B, with node C denoted as designated forwarder. The neighborhood table of node B contains node A, which would not be possible without a two-hop neighborhood discovery protocol, as node B does not receive any messages from node A. The link is marked as unidirectional-outgoing, and thus does not need any forwarder. The second entry features node C with a bidirectional link, needing no forwarder either. Finally, the neighborhood table of node C contains nodes A and B, both marked as connected through bidirectional links and not needing any forwarders.

Due to the fact that the unidirectional link and the detour that is taken on the way back form a triangle, this protocol is called Unidirectional Link Triangle Routing.

ULTR is similar to the link layer tunneling mechanism proposed by the unidirectional link working group of the IETF [8], but does not require multiple interfaces on the nodes to communicate. Also, depending on the used neighborhood discovery protocol, it may even be able to work with triangles, which include more than one unidirectional link, which the link layer tunneling mechanism cannot handle. Moreover, ULTR works completely on the routing layer, the link layer is not involved. This is an advantage when timeouts are used, because the extra hop and thus longer delay are not hidden from the routing layer.

A. Neighborhood Discovery

The neighborhood discovery protocol needed for ULTR can be quite simple and needs only be started on a node once it receives the first message from a neighbor, i.e., when the first route request message is flooded into the network. Once it has been started, the neighborhood discovery protocol regularly transmits a message containing the IDs of all nodes from, which this node has received messages recently and the status of its links to and from them. When a node receives such a hello message, it checks whether its ID is contained therein. If it is not, the receiving node knows that it is on the receiving side of a unidirectional link.

In other protocols, where unidirectional links are used, a lot of overhead would now be necessary to inform the upstream node (the sender of the hello message) of the unidirectional link. In this protocol, the upstream node does not need to know about its existence. The receiving node only marks the link as unidirectional-incoming in its neighbor table.

When a node A receives a hello message via the bidirectional link from node C in, which the upstream node of the unidirectional link is listed and the link to that node (from C to B) is marked as bidirectional, node A enters the sender of the hello message (node C) as a forwarding neighbor into the corresponding neighbor table entry (for node B). Please note that this would also be possible if there was a unidirectional link from C to B, but the proactive detection of this special case would probably introduce a large overhead and solve only one special case: If there is a unidirectional link from C to B and no other neighbor of A has a bidirectional link to B.

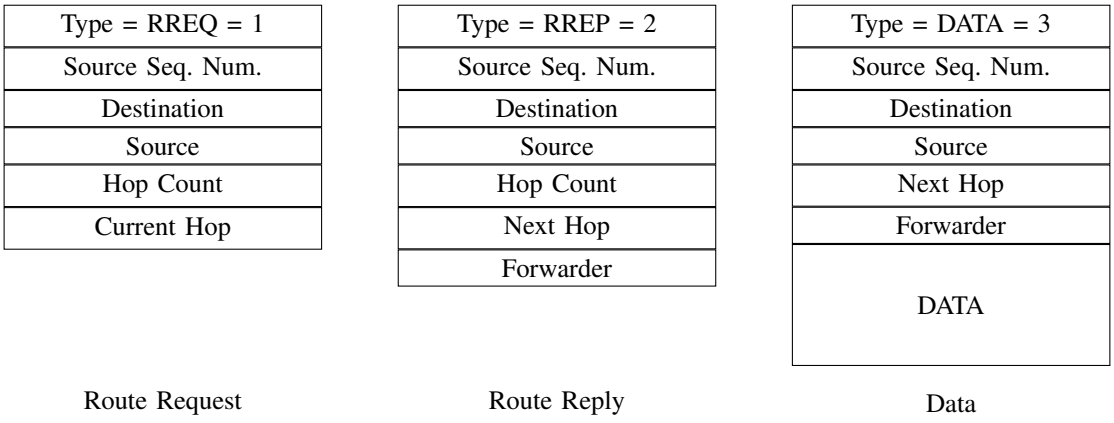


Fig. 2. Message types used in ULTR

When a message (RREP or DATA) is sent the reversed way, it needs to be forwarded along a one-hop-detour. This message can be used to inform the upstream node of the link, which is then entered into the upstream node's neighborhood table as unidirectional-outgoing. Please note that for the routing alone this information would not be necessary, indeed it would be easy to hide the fact that the message has taken a detour. But for the sake of timers that can be used for retries on MAC- or routing layer, it helps to know that the delay could be twice as high. In this case, the information about this special link can be acquired "for free" and could be used to solve the problem described above. The information about the unidirectional-outgoing link can also be used by the MAC layer not only for retries, but also to determine the right two-hop neighborhood of a node, which is a mandatory information for TDMA protocols.

B. Message Types

ULTR uses the standard three message types used by most reactive routing protocols: Route Request, Route Reply and DATA. Figure 2 shows an example for each of them.

A RREQ message contains the identity and sequence number of the source that are used for duplicate detection, followed by the identity of the destination. The hop count is incremented by one on each hop as usual, and the identity of the last hop is used to build the backward route.

A Route Reply message contains sequence number and identity of the source for duplicate detection as well as the identity of the destination. For forwarding purposes the next hop and, if necessary, the forwarding node are included.

The DATA packet contains the sequence number and identity of its source as well as the identity of its destination and, of course, the application data. As the size of the data may vary, the identities of the next hop and, if suitable, the forwarding node needs to be inserted before the data for alignment reasons.

C. Variations

ULTR relies on a neighborhood discovery protocol, which supplies information about incoming and outgoing unidirectional links. If neither the application nor the MAC protocol needs a neighborhood discovery protocol, a variation of ULTR with passive link detection may be used. But passive link detection means that sometimes a node does not know about links to its neighbors, even though they are available. Therefore, a second mode of operation is introduced: if a node does not have a link to the next hop in its neighbor table, it forwards the message nonetheless, with an additional flag telling its neighbors that any of them that do have an active link to the next-but-one hop (i.e., the siblings of the next hop) should also forward the message.

When this variation is used, some modifications of the message types are necessary (see Figure 3). Information about the last hop would have to be included in RREQ messages, in addition to the current hop. Both node IDs are stored in the routing table. A node decides, which entry to use depending on the overheard status of the link. If the next hop is assumed to be connected by a bidirectional link, the normal next hop is used. Otherwise, the message is set to alternate mode and the next-but-one hop is used. The last hop is also used for implicit link detection: If a node overhears the transmission of a message in, which it is denoted as last hop, it knows that the link between itself and the current hop denoted in the message is currently bidirectional.

A RREP message contains three node IDs instead of only two: The last hop ID and current hop ID are used to build the backward route for normal and for alternate mode just as they are used in the RREQ. The next hop ID is used for forwarding. However, the RREP also contains a flag denoting the mode of transmission, which can take on the values "normal" and "alternate". It is evaluated upon message reception to decide if a node shall forward the message or not. In normal mode it only forwards the message when it is denoted as next hop in the message, in alternate mode it also forwards the message if it has the next-but-one hop in its neighbor table.

The DATA message features the same three node IDs that are present in the RREP message. For routing purposes alone, the last hop ID would not be needed, but it is nevertheless included for link status detection. The mode flag is also present again, to enable the usage of alternate mode if the status of the next link is unknown or known to be unidirectional-incoming.

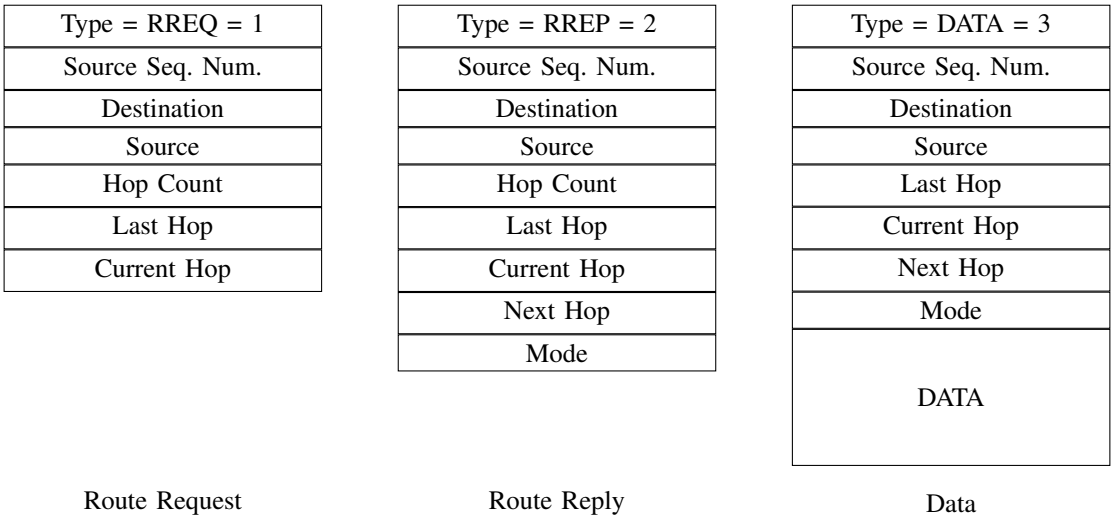


Fig. 3. Message types in ULTR without neighborhood discovery

D. Advantages and Disadvantages

ULTR is a complex protocol. The complexity is the price for the reduced number of data packet transmissions, as no flooding of DATA packets, not even a limited one, is used. Periodic updates of the neighborhood table ensure that the link status information it holds is always up to date, which also enables implicit local repair. Altogether this should lead to a higher delivery ratio. On the downside, the usage of hello messages also leads to more protocol overhead, as these messages can be quite large in dense networks. Therefore, the typical tradeoff between actuality and network load has to be made when setting the hello period, which makes configuring the protocol harder. On the other hand, a new option for cooperation between MAC and routing arises.

Like all routing protocols that use unidirectional links, ULTR also needs a MAC that can transmit over unidirectional links. The information about the existence of the unidirectional links probably needs to be collected to a certain extent anyway, depending on the MAC protocol used. So either this can be retrieved from the MAC without additional cost, or the MAC protocol can query the routing layer for it using an appropriate interface. More information about the cooperation options between MAC and routing is provided in Section III.

III. Cooperation with the MAC-Layer

ULTR was designed specifically to utilize unidirectional links. This makes it imperative to use a MAC layer that can also transmit over unidirectional links. Any protocol that uses the standard ”request to send” - ”clear to send” mechanism is completely unsuitable, as no clear to send message will ever be received over an outgoing unidirectional link. Moreover, nodes with an outgoing unidirectional link will never know that they could be disturbing the communication between two other nodes.

There are some improvements that allow contention based protocols to work with unidirectional links, e.g., ECTS-MAC [9], [10]. Some of the MAC protocols that utilize unidirectional links route their link layer acknowledgments back to the upstream nodes. For this, the neighborhood table used by ULTR could be reused.

Plan based MAC protocols need to know the two-hop neighborhood of each node to identify the collision domain. Within this domain, the varying parameter (e.g., frequency (FDMA), code (CDMA) or slot (TDMA)) needs to be unique for each node. Therefore, a neighborhood discovery protocol is needed, which finds these two-hop neighbors. The protocol(s) used for ULTR could easily be enhanced to deliver this information. Otherwise, if the MAC protocol already has its own neighborhood discovery protocol, it only needs to make the gathered information available to the routing protocol.

The usage of such a neighborhood discovery protocol would also implicitly solve the ”special case” of a unidirectional link triangle with more than one unidirectional link, enabling ULTR to make use of such links as well.

This usage of a single neighborhood discovery protocol for both MAC and routing reduces communication overhead and memory consumption by far. It also ensures that both layers work on the same data. If they would use different algorithms, different storage sizes or replacement strategies, lots of problems could result, as described, e.g., in Murphy Loves Potatoes [11].

IV. Evaluation Methodology

For the evaluation, we chose a comparative approach: We evaluated the performance of ULTR and compared the results to those achieved by other typical routing protocols from the world of sensor networks or MANETs. The protocol most used in sensor network deployments was chosen as first competitor: A tree routing based approach with retransmissions, which is quite common in sense-and-send applications where all nodes transmit their data to the sink regularly (e.g., [4], [11], [12], [13]). As this may seem to be an unfair comparison, two protocols from the MANET area were also chosen as

competitors: DSR in the version that uses unidirectional links [14] and AODVBR [15]. AODVBR does not use unidirectional links, but has an interesting way of detecting them and salvaging the data message that caused the detection. As fourth reference protocol, `Flooding` is included. While it is known that `Flooding` induces a lot of overhead, it can still deliver valuable insights. In the simulations, `Flooding` is used to determine the upper limit of messages that could reach the destination. In the real world experiments carried out for this work, the network load it generates is used to understand the performance of the MAC protocol supplied by the hardware in use.

The distance, measured in hops, is taken as weight function (minimum hop routing), but other weights, e.g., residual energy, could also be used with the same result, as all protocols would work on the same values. Routes with a lower weight replace older ones with a higher value in the routing tables.

The authors of [16] propose a combined evaluation method that uses experiments with real hardware, emulation and simulation techniques in order to speed up the deployment of new protocols. The combination of all three methods enables the developer to identify, which problems occur and shows him/her where further investigation is necessary. The routing protocols AODV, DSR and OLSR were used to evaluate the proposed approach to protocol monitoring. The authors found that latency and timing are crucial to the performance of reactive protocols like AODV and DSR, because of buffering times. The queue-ups that can result from this buffering were apparent in the experiments, but not in the emulations. In conclusion of this paper it can be said that all three methods of evaluation have their own gain for a protocol developer, if they are used correctly. For simulations, the choice of the underlying communication model is crucial. The emulation can be fed with real world connectivity data, and can be used to evaluate the implications of the network stack used on the real devices. Experiments are needed to generate this connectivity data. It is important that for all three methods exactly the same implementation of the protocol is used, and that this implementation is the one that can be used directly on the hardware, which is used in the real experiments.

Following this approach and the advice from Stojmenovic [17], the same implementation was used for both simulations and real world experiments in this evaluation.

In the next section, the methods used in the simulations are described. They enable the evaluation of the algorithms and their ability to handle unidirectional links under controlled circumstances (Section V). The general principle of the real world experiments, including the chosen locations, is described in Section VI.

The actual evaluation of the routing protocols is presented in Sections VII to IX, sorted by the application scenario in use.

V. Simulations

All simulations were performed using the discrete event simulator OMNeT++ [18] with the MiXiM-extension [19]. We modified MiXiM to enable the simulation of unidirectional links [20]. The simulated networks consisted of four different sizes of grids: 100 nodes (10x10), 400 nodes (20x20), 900 nodes (30x30) and 1,600 nodes (40x40). A grid alignment was chosen to represent applications that need area coverage, where each node is equipped with sensors that have a range of one distance unit. But, as will be seen below, the exact placement of the nodes is not important, because connectivity is determined using a connectivity matrix (Section V-A). The different numbers of nodes represent network sizes ranging from small to huge networks, and thus increase the number of hops needed to communicate from one end of the network to the other. This determines the route length, which has a tremendous impact on the performance of all routing protocols.

All simulations are restricted to the usage of a ”perfect behavior” MAC. While it is of course true that the choice of medium access control protocol can have a strong influence on the performance of the routing layer, the goal of the simulations is the evaluation of the ability of the routing protocols to work in the presence of unidirectional links, not of their interaction with the MAC layer. Also, many of the effects of a MAC layer, e.g., the available neighbors for each node, would be the same for all evaluated routing protocols. The effects could only differ between protocols, when they are depending on the generated network load, as different protocols transmit different types of messages with different sizes and in different frequencies. But all of these are highly dependent on the application, and it is not possible to evaluate all possible application scenarios.

As simulation results are never 100% accurate, real world experiments have been conducted, too. Details about the methods of evaluation used for the real world experiments are shown in Section VI. This section follows Stoijmenovic's advice [17], and uses a simple model in order to keep side influences small and results interpretable.

A. Connectivity between Nodes

To simulate a certain connectivity between nodes, thousands of connectivity matrices were generated before running the simulations. The same generated matrices were used for all protocols. The large number of matrices is necessary to simulate the constantly changing nature of wireless links. As the largest networks, consisting of 1,600 nodes, needed to be simulated for the longest time, they also needed the highest number of connectivity matrices: For a single simulation 17,761 connectivity matrices were needed.

In each of these matrices, a (directed) link from node A to node B exists with a probability of α/d^6 where d is the distance between node A and node B. The inverse link, from node B to node A exists with the same probability. Therefore, the link is bidirectional with a probability of $(\alpha/d^6)\times(\alpha/d^6)$, unidirectional (in any one direction) with $\alpha/d^6\times(1-(\alpha/d^6))$ and non existing with $(1-(\alpha/d^6))^2$. The quotient (d^6) reflects the dampening induced by the distance between nodes while α represents the probability that a link between geographically adjacent nodes exists.

Nodes were arranged on a regular grid to reflect application scenarios that need area coverage, e.g., vehicle tracking. As all

nodes were arranged on a grid, nodes that are directly above, below, right or left of a node are called direct neighbors and their distance was defined as 1. α was varied between 0.9, 0.95 and 1, and for each value of α ten sets of matrices with different seeds for the random number generator were generated, leading to 30 sets of matrices per network size, and a total of 996,120 connectivity matrices containing between 10,000 and 2,560,000 entries.

Please note that due to the fact that the matrices were generated randomly, there is no guarantee that there always was a path from sender to destination. Therefore, no upper limit can be calculated, but `Flooding` is used as reference protocol: The number of application messages delivered by `Flooding` is taken as 100% and the delivery ratio of all other protocols calculated accordingly.

B. Application Settings

In each simulation, each node wanted to transmit a total of 110 messages to one or more destinations, depending on the scenario. After the initialization phase of the network, one message was transmitted every 100 milliseconds. To ensure that route discovery was finished, the logging remained inactive until all nodes had started the transmission of their fifth message. The connectivity matrices were changed every second. Please note that the absolute values of the time units are not important for the simulation, only their relation (1:10). They could also have been set to 6 seconds and one minute yielding the same results.

C. Protocol Performance

In the simulations, logging only began once each node had started the transmission of its fifth message. Therefore, the theoretical optimum of delivered messages could be calculated, if connectivity could be guaranteed. But the connectivity matrices were generated randomly, therefore network separation could be possible. `Flooding` delivered close to the theoretical optimum, and is used as maximum for the simulations. For all simulations, the delivery ratio of a protocol is defined as the number of messages delivered by the protocol divided by the number of messages delivered by `Flooding`.

VI. Real World Experiments

To evaluate the influence of medium access control and the properties of real hardware, all protocols were evaluated on 36 eZ430-Chronos [21] sensor nodes.

All protocols use the same sensor nodes on the same locations, meaning that node 0 used to evaluate `Flooding` is the same piece of hardware on the same location as node 0 used in the experiments evaluating ULTR and so on. Depending on the application scenario, the experiments were conducted on some or all of the locations described below. Each protocol was evaluated using a freshly charged set of batteries.

A. Application and Logging

In the real experiments, each node wanted to transmit a message every minute. The experiments ran for one hour each, therefore 60 messages were transmitted by the application on each node. In all experiments, 36 nodes were placed in a square of six times six. Each node recorded the number of application messages it received, and all nodes recorded the number, type and size of all messages they transmitted or forwarded.

Like in the simulations, it was once again possible that nodes were disconnected from the network and suffered from network separation. Also, sometimes nodes failed due to hardware problems. Therefore, the type of messages transmitted by a node was evaluated, too. When a node only transmitted route request messages and not a single data message, it did obviously not find any route to the sink.

TABLE II
Total size of the deployed systems for different routing protocols in Byte

protocol	text	data	bss	dec
AODVBR	14,590	0	2,260	16,850
DSR	17,760	0	3,586	21,346
`Flooding`	12,444	0	1,644	14,088
`Tree Routing`	13,234	0	1,990	15,224
ULTR	14,550	0	2,066	16,616
System without routing	11,918	0	1,418	13,336
Basic System	8612	0	994	9,606

B. Protocol Performance

In the real world experiments, logging began at once. Therefore, the theoretical optimum of delivered messages could be calculated, if connectivity could be guaranteed, which is never the case in real world deployments. In contrast to the simulations, `Flooding` could not be used as reference protocol because it did not always deliver the highest number of application messages. Therefore, the delivery ratio is defined as the number of application messages delivered to their destination divided by the number of application messages transmitted.

C. Program Size

The size of the programs deployed on the eZ430-Chronos is shown in Table II. Please note that the values were measured for scenario 1 (sense-and-send, Section VII), but differ only marginally for the other scenarios as the main components (system and routing protocol) are always the same. Only the application differs from scenario to scenario, but its influence on the program size is marginal.

It can be seen on the table that DSR has by far the largest memory footprint, concerning both flash (”text”) and RAM (”bss”). The lowest footprint can be seen on `Flooding`. It needs only about 500 Bytes flash and 200 Bytes RAM more compared to the system without routing, most of, which is needed for the duplicate suppression.

The basic system, including only the operating system Reflex [22], [23] without any scenario specific parts (no routing protocol, no application) is also shown for comparison. It needs 8612 Bytes of flash and 994 Bytes of RAM. Most of the RAM consumption is due to the 10 network buffers with 64 Bytes each.

(a) affixed to poles (b) placed on the lawn (c) on a stone pavement

Fig. 4. A modified eZ430-Chronos sensor node

DSR did not fit on the micro controllers with the settings used in the simulations, therefore some of them (e.g., the number of messages that can be stored) had to be reduced to make it fit. As DSR has the largest memory footprint, all other protocols had no problem fitting on the micro controller when using the same settings.

D. Experiment Locations

Four different locations were used for the real world experiments:

- On a Desk
- Affixed to Poles
- Placed directly onto a lawn
- Placed directly onto stones

a) Desk Experiments: This deployment is a single hop layout, where each node is able to receive messages from each other node. The nodes lay directly next to each other. An old set of batteries was used without re-charging them, because range did not really matter in these experiments. They were used to validate the correct operation of the protocols.

b) Poles: For the pole experiments, small poles were deployed on the lawn in front of the main building of our university, with about one meter distance between each of them. Then, the sensor nodes were affixed to them using cable straps, at a height of about 20 cm (Figure 4(a)). The pole placement was usually used at 8am.

c) Lawn: After the pole experiments were finished and evaluated, the nodes were reset and placed on the ground directly next to the poles as shown on Figure 4(b). The resets were done by disconnecting the batteries and reconnecting them directly afterwards. The same set of batteries as before was used on each node without charging. When using all four locations, the lawn experiments were started at about 10 AM.

d) Stones: After the lawn experiments, the nodes were disconnected, and poles as well as nodes and batteries collected. The experiments on the stones were always started at about 1 PM, using the same set of 72 AA batteries used in the morning without re-charging, but the pairing of batteries and nodes might have changed, i.e., the batteries that were connected to node 4 in the pole and lawn experiments might be connected, e.g., to node 27 in the stone experiments. These experiments were conducted on the stone pavement on our campus (Figure 4(c)).

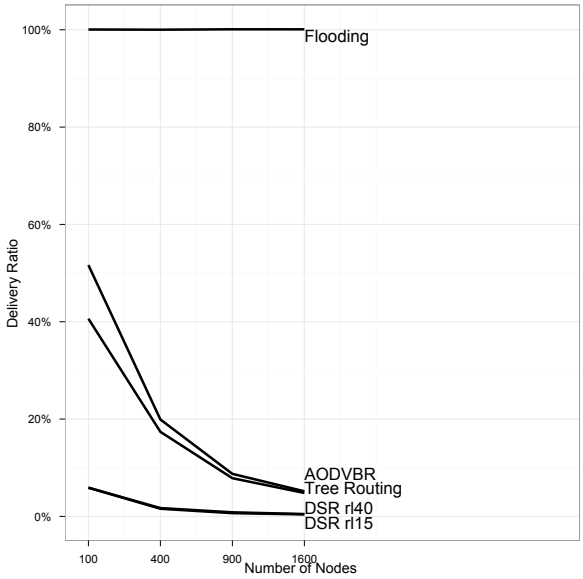


Fig. 5. Delivery ratio of AODVBR, `Tree Routing` and two DSR versions, first scenario

VII. Application Scenario 1: Sense and Send

The application implemented for scenario 1 represents a sense-and-send behavior that is often found in sensor networks: All nodes within the network wanted to transmit all their messages to the same destination.

A. Simulation Results

The destination (sink) was fixed within a single simulation, but multiple simulations with different destinations were evaluated. For the network containing 100 nodes, all nodes in the upper left quadrant were chosen (25 destinations), for the network with 400 nodes this quadrant contained 100 nodes. Evaluating only one quadrant was chosen because of the symmetry of the network, and because of run time limits (a single simulation of `Flooding` in a network consisting of 1,600 nodes took about 27 hours to complete). For the networks containing 900 and 1,600 nodes a whole quadrant would have meant too many simulations, therefore only the 20 most interesting nodes (the corners and the middle of each quadrant) were chosen (20 destinations).

As 30 different connectivity change lists were used for each destination in each network size, 4,950 simulations with run times between 5 minutes and more than a day were necessary for each protocol.

1) Related Work Protocols: The number of data messages received at the sink for the reference protocols is shown in Figure 5.

It can be seen that none of the other protocols gets anywhere near the performance of `Flooding`, with DSR performing worst. Even in the smallest network consisting of 10 times 10 nodes, both DSR versions (max route length 15 or 40) deliver only about 10% of the messages.

The other two protocols perform better in the small network, but show a steep decline in delivery ratio for the larger

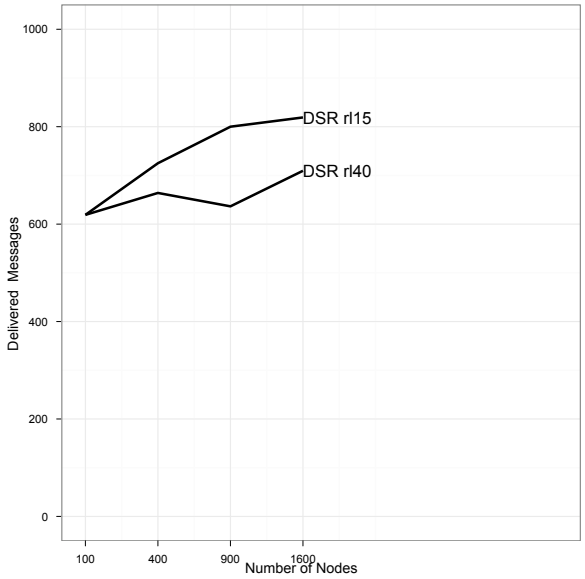


Fig. 6. Total number of Data messages at the sink, two DSR versions, first scenario

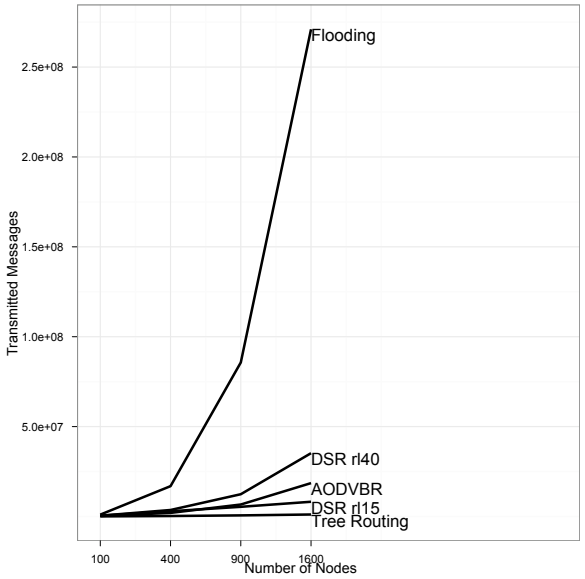


Fig. 7. Number of transmitted messages, `Flooding`, AODVBR, `Tree Routing` and two DSR versions, first scenario

networks. This decline is due to the fact that even though the number of nodes in the network and, therefore, the maximum possible number of delivered messages increases drastically, the total number of delivered messages increases only marginally.

The absolute number of messages received at the sink for the two versions of DSR is shown in Figure 6. It can be seen that DSR delivers a nearly constant number of messages, independent of the network size. While the number of nodes and thus the number of application messages handed to the routing protocol is multiplied by 16, the number of application messages that arrive at the sink increases only marginally. This is due to the fact that DSR suffers heavily from link changes and longer routes change more often. Another interesting fact about DSR is that the version with route length limited to 15 delivers more messages than the one, which allowed route lengths up to 40 hops for all larger networks. The reason for this seemingly strange behavior can be seen when investigating nodes that are about 15 to 17 hops from the sink. Please remember that the hop distance changes as links change. Therefore, nodes might have a distance of more than 15 during their first route discovery, and less during a later one. When only short routes are allowed and no route is found, the messages are stored until a later route discovery finds a route containing 15 hops at max. Then, all stored messages are transmitted at once. These messages have a higher chance of being delivered, as the route information is current and the path is shorter. When using the 40 hop limit, these nodes choose the first, long path that is found. But longer paths have a higher probability of message loss, leading to fewer messages being delivered in total.

The total number of messages transmitted by each of the protocols chosen for comparison is shown in Figure 7. `Flooding` naturally transmitted the most messages by far.

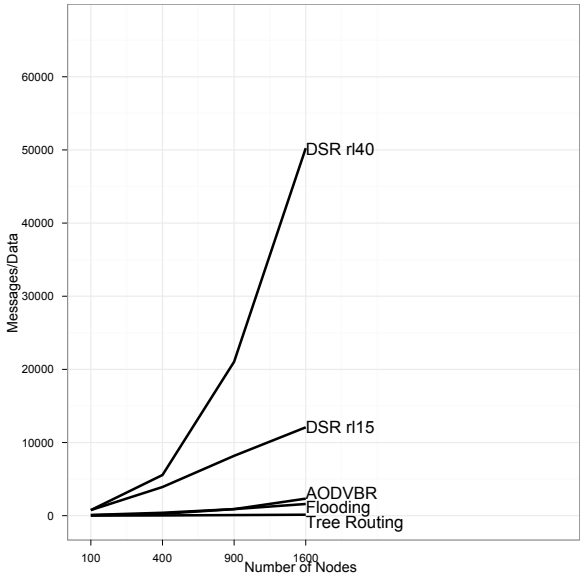


Fig. 8. Number of messages transmitted to deliver a single application message, `Flooding`, AODVBR, `Tree Routing` and two DSR versions, first scenario

Also, it can be seen that `Tree Routing` transmitted very few messages, and DSR with route length 40 transmitted much more messages than the version with route length 15.

The number of messages transmitted in order to bring a single application message to the sink is shown in Figure 8. Even though DSR with route length 40 delivered nearly the same amount of data as DSR with route length 15, the high number of transmitted messages makes it the most costly related work protocol by far. Interestingly, even though it transmits a large number of messages, the high number of delivered messages make `Flooding` the second best. Only

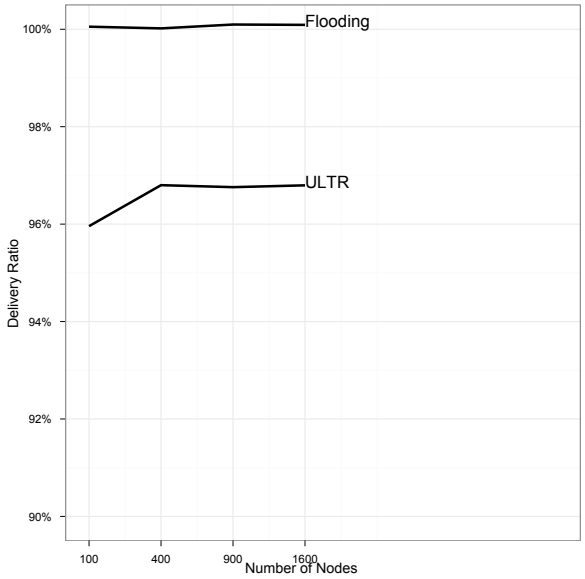


Fig. 9. Delivery ratio of ULTR and `Flooding`, scale starts at 90%, first scenario

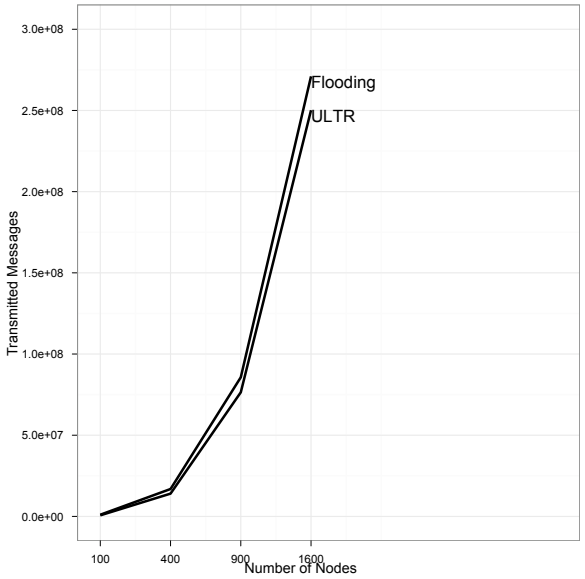


Fig. 10. Number of messages transmitted by ULTR and `Flooding`, first scenario

`Tree Routing` performs better. This is due to the fact that `Tree Routing` has very low costs for delivery failures. Nodes close to the sink are often able to deliver their messages. Nodes that are farther away transmit their messages, and try two retransmissions if the message is not forwarded by the next hop. But, contrary to the other protocols, no route error messages are generated and no new route discovery is initiated when the retransmissions are unsuccessful.

2) ULTR: The delivery ratio of ULTR is compared to `Flooding` in Figure 9. Note that the scale starts at 90%.

What catches the eye right away on the figure is that the delivery ratio is very high and increases with network size. ULTR seems to have reached its maximum at 97% already in networks consisting of 20 times 20 nodes, but to be sure more simulations with larger networks would be necessary. These were not done for this paper for two reasons: First, the simulation run time would be very high. A single simulation of `Flooding` in the 40 times 40 network took more than a day, and 600 of them were necessary. In 50 times 50 networks the value would be much higher. Second, the largest network that was simulated, 40 times 40, already contains 1,600 nodes and it is unlikely that such large sensor networks will be deployed for a real application in the near future. If larger networks are deployed, it is likely that a logical partitioning of the network would be realized on application level, and multiple sinks would be used.

The number of messages transmitted by ULTR and `Flooding` is compared in Figure 10. The figure shows that ULTR needs a lot of message transmissions to compensate for the missing neighborhood discovery protocols: As the protocol was designed with the assumption that either a neighborhood discovery protocol or the used MAC layer would supply link information, it suffered from the absence of accurate information. The passive overhearing that was implemented instead

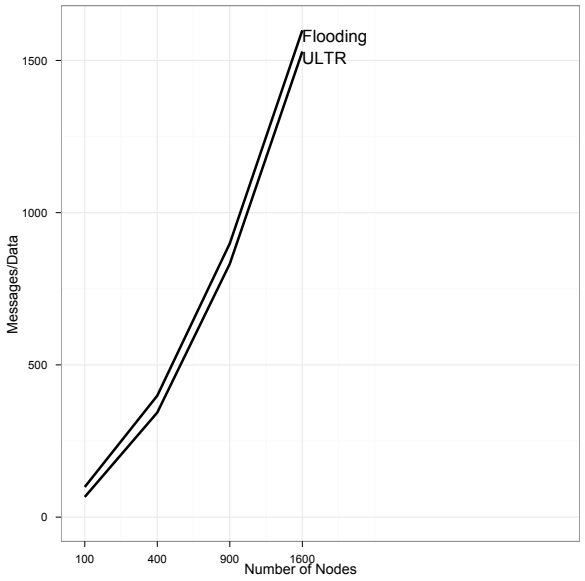


Fig. 11. Number of messages transmitted to deliver a single application message, ULTR and `Flooding`, `first scenario`

can only detect bidirectional and unidirectional incoming links, which makes the explicit usage of unidirectional links all but impossible. Therefore, ULTR tries to find bidirectional links, or, if these are not available, switches to alternate mode for one hop, which increases the network load very much when it is initiated too often. Another problem is timing: Passive detection of links only works when messages are transmitted, but links change more often than messages are transmitted. Therefore, ULTR often worked on outdated information.

The costs that the delivery of a single data message caused on average are shown in Figure 11. ULTR produces almost

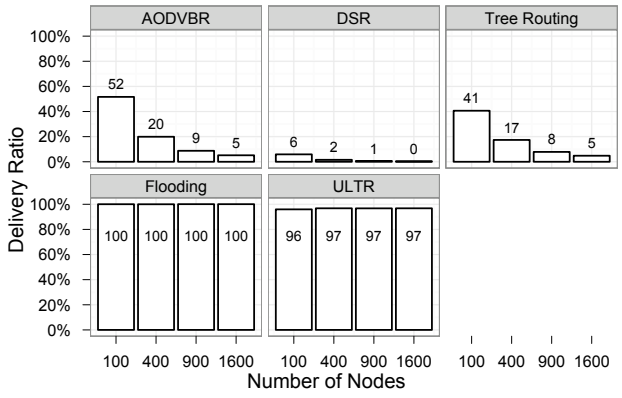


Fig. 12. Delivery ratio of all protocols for different network sizes, first scenario

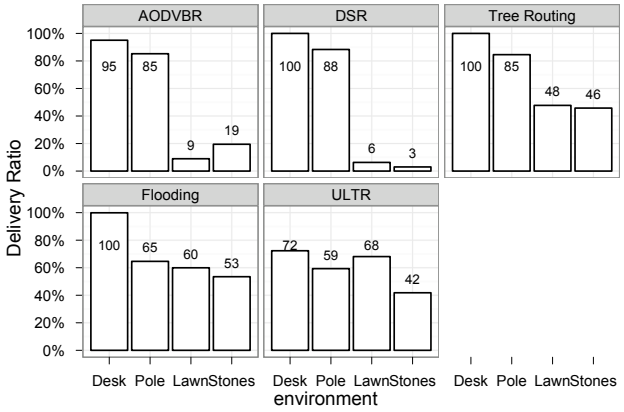


Fig. 13. Delivery ratio of each protocol achieved in the real experiments, first scenario

the same costs as `Flooding` with more than 1,500 messages in the network consisting of 1,600 nodes.

3) Comparison between all Protocols: Concluding the evaluation of these simulations it can be said that ULTR has achieved a much better delivery ratio in application scenario 1 than the protocols used for comparison. Only `Flooding` delivered more messages, which is why it was used as reference, and the delivery ratio of a protocol defined as the number of messages delivered by that protocol divided by the number of messages delivered by `Flooding` (Figure 12).

Even though the simulations did not feature MAC layer elements, it can already be seen that the protocols chosen from related work are not able to work in an environment with many unidirectional links and often changing links in general. On the other hand, the results clearly show that the developed protocol, ULTR, has achieved its design goals, namely resistance against often changing links and usage of unidirectional links. Only `Flooding` delivered better results in the simulations, and it is known that `Flooding` runs into huge MAC layer problems when it is used on real hardware.

B. Real World Experiment Results

For the real world experiments of scenario 1, all four different locations described in Section VI-D were used. Each protocol was evaluated on each location, with node 0 in the lower left corner as destination (sink).

The delivery ratio of each protocol is shown in Figure 13, sorted by protocol and location. For most protocols, the number of delivered messages for the desk and pole locations is roughly the same, as these two locations differed only marginally. The desk location is one hop, while the pole location contained between one and two hops on average. The figure also shows that `Flooding` delivers a nearly constant number of messages for the pole, lawn and stone environments.

The other three reference protocols, AODVBR, DSR and `Tree Routing` show a steep decline in delivered messages for the lawn and stone pavement placements. ULTR always delivered more messages to their destination, except

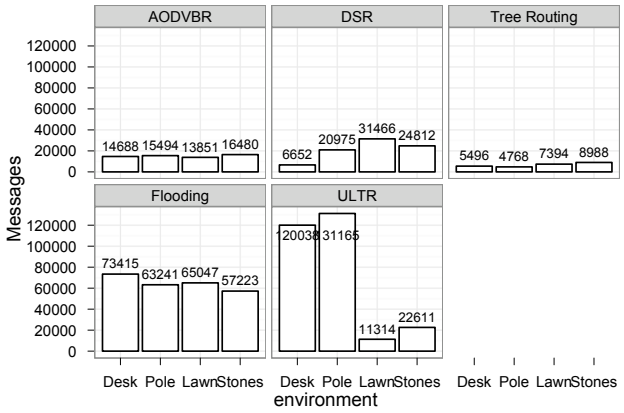


Fig. 14. Total number of messages transmitted by each protocol, real experiments, first scenario

for the stones placement. The reason for the bad results from ULTR lies in its dependency on accurate link information. As described in Section II, ULTR tries to route messages around unidirectional links explicitly. But in order to build this triangle, neighborhood information is needed. The current implementation of ULTR tries to obtain this information passively, by overhearing forwarded messages. For a rapidly changing environment this approach is bound to fail. It would be interesting to see, how a protocol implementation that uses a neighborhood discovery protocol or neighborhood information provided by the MAC-layer would perform.

The number of protocol and data messages transmitted by each protocol can be seen in Figure 14. Once again, `Flooding` remains fairly stable throughout the locations. While all other protocols transmitted more messages in the last two locations (lawn, stone pavement), the number of messages transmitted by ULTR declines. This is once more due to the absence of accurate link information. ULTR was designed with the assumption that link information would be available either from a neighborhood discovery protocol or from the MAC layer. Using only overheard messages instead does not work

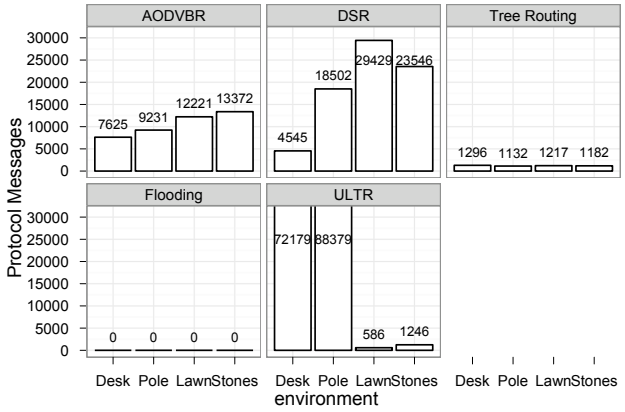


Fig. 15. Number of protocol messages transmitted by each protocol, first scenario

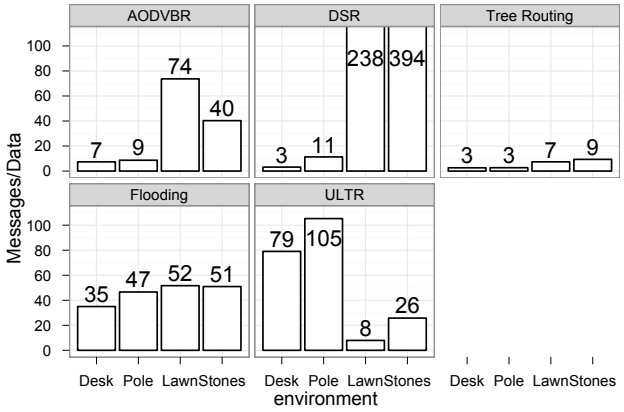


Fig. 16. Total number of messages transmitted by each protocol divided by the number of delivered data messages, first scenario

in the first two locations: When all nodes can transmit directly to the sink and the sink never answers, all links are assumed to be unidirectional and alternate mode is induced for every message. Therefore, every message is flooded. As ULTR uses a route request - route reply mechanism to find routes, the number of transmitted messages in a single hop scenario is even higher than that of `Flooding`, which only transmits data messages.

An even more interesting fact that can be seen in the figure is that the passive neighborhood discovery mechanism starts to work in the multihop environments. When paths are more than 1-2 hops in length, forwarded messages are received more often and the nature of the links can be observed. Therefore, even though it might seem strange, ULTR needs to transmit fewer messages in networks with a larger diameter.

The number of protocol messages, i.e., non-data messages, transmitted by each protocol can be seen in Figure 15. As already seen above, ULTR suffered badly from the missing neighborhood discovery protocol. ULTR nearly always switched to alternate mode. The huge number of protocol messages transmitted by ULTR consisted mainly of route request messages. In fact, a route discovery took place for nearly each data message generated by the application in ULTR.

Another measurement of the cost paid to deliver an application message is shown in Figure 16. The figure shows the total number of transmitted messages divided by the number of application messages that reached the sink. Unsurprisingly, `Flooding` once more shows a relatively constant performance and DSR and AODVBR show too high values. Interestingly, `Tree Routing` seems to have performed quite well. If only this figure was taken into account when choosing a protocol, `Tree Routing` would be preferred. However, the numbers presented here have to be put into perspective. The result of `Tree Routing` is achieved because it uses nearly no protocol messages, and two retransmissions are its only reaction to message loss. No route error messages are generated and no new route discovery is started. Therefore, the cost of a lost application message is much lower than in the other protocols. DSR represents the other end of the spectrum: When a message loss is detected there, a route error message is created and transmitted to the originator. When the route error is received, the route is deleted and a new route discovery is initiated, which leads to a flooding of the whole network. If conditions are really bad, it may even lead to flooding the network twice. Except for the problems experienced by ULTR in the 1-2 hop locations, it performs fairly well. But the results also show that the number of nodes used in the real world experiments was actually a little low - as the simulations have shown (see Section VII-A), the big differences between protocols can be seen better in larger networks. However, using a few hundred nodes in the real world experiments was not possible as there were not that many nodes available.

C. Comparison between Simulations and Experiments

The real world experiments were conducted with 36 nodes, while the simulations featured either 100, 400, 900 or 1,600 nodes. To show that the tendencies seen in the simulations represent those that would be achieved with a large scale sensor network, a network consisting of 36 nodes was also simulated.

Figure 17 shows the median of the delivery ratio of all evaluated protocols for the two multihop experiments (lawn, stones) and the 36 nodes simulation. Naturally, the results of `Flooding` in the simulation are much better than those achieved in the real world experiments, as `Flooding` suffers heavily from the broadcast storm problem in the real experiments. The used CSMA MAC layer simply cannot handle the huge number of messages. The simulation results and those of the two experiment settings are quite similar for the protocols surveyed in this paper. From the protocols used for comparison, only AODVBR shows a large difference between simulation and real world results.

When looking at the results, which those two protocols, AODVBR and ULTR, achieved in the real world experiments, it can be seen that they have a strong variation in delivery ratio between the lawn and stone experiments. This high variation seems to imply that both protocols are especially vulnerable

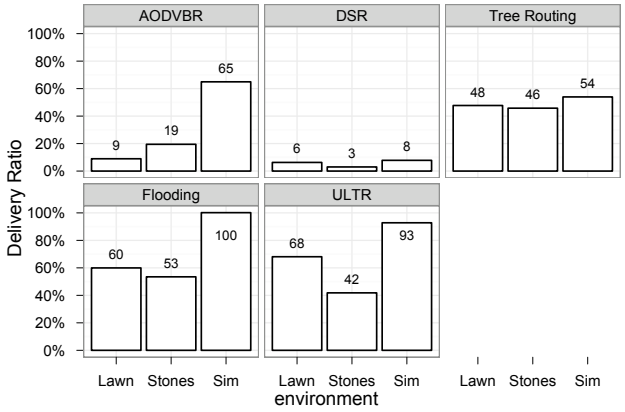


Fig. 17. Delivery ratio of each protocol; experiments vs. simulations

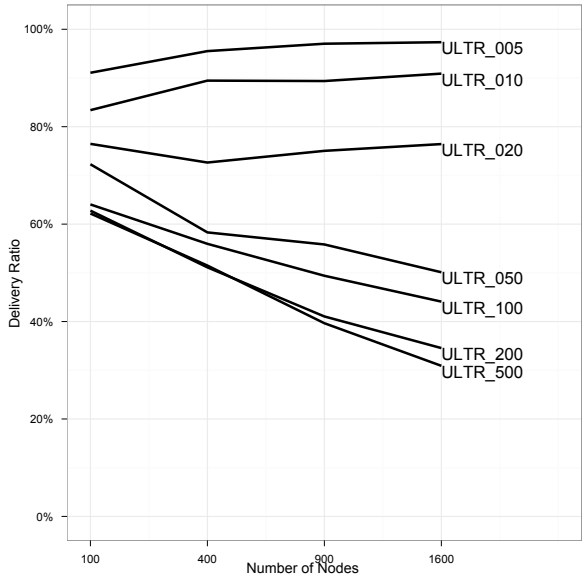


Fig. 18. Delivery ratio achieved in the simulations for different timeouts

to one or more properties of the real experiments, which do not have so much influence on the other protocols.

Both AODVBR and ULTR try to use an explicit detour around unidirectional links, using link information detected during route reply transmission (AODVBR) or during transmission of DATA messages (ULTR). As the connectivity measurements we conducted [2] have shown, link changes occurred even more often than expected, making conditions for AODVBR and ULTR harder in the real world than in the simulations. None of the other protocols suffered as much as these two. For DSR, an additional increase in frequency of changes made no difference, as it could not even tolerate the one simulated. For `Tree Routing`, the small network diameter and the 2 retransmissions on each hop were enough to deliver about 50% of application messages. The link changes would not have influenced `Flooding`, but `Flooding` produced a very high network load, which the MAC layer could not handle. Still, it can also be seen that the deployed sensor network was not large enough for them to show their full potential.

In summary it can be said that the used simulation approach has some limitations, as it does not include the medium access control protocol used in the real experiments. However, the results show that the usage of connectivity matrices and the way they were generated is close to reality, and can be used to evaluate the influence of unidirectional links and frequent link changes on the routing protocols. This is exactly what the simulations were intended for as the used MAC layer and other side effects of the used hardware might (and hopefully will) change for future deployments. When the exact properties of the hardware that will be used in a deployment are known beforehand, these could be included in the simulations. Some of the less favorable communication properties of the eZ430-Chronos (e.g., the inability of the CCA to receive messages during backoff) were only discovered during the connectivity measurements [2].

Another advantage of the simulation model is the fact that the connectivity data gathered during the connectivity measurement experiments can easily be included. The data that could be gathered this way was not presented in this work, because the number of data sets from the connectivity experiments currently available is too small.

D. Importance of Timeouts

The implemented version of ULTR with passive link detection is heavily dependent on the timeouts that are used for the links. If it is set too low, the links are deleted before they may be used, even though they might still exist, resulting in a local broadcast on every hop. If it is set too high, links are assumed to exist, but have broken a long time ago.

The implementation of ULTR uses a timer that fires every 100 ms, and has a parameter called `linkTimeout` that defines how many times that timer must fire before a link is removed from the neighbor table. The results presented above were achieved with a `linkTimeout` of 5, and resulted in a lot of message transmissions but also fairly high delivery ratio. To quantify the impact of the `linkTimeout`, the performance of ULTR was measured with different values of `linkTimeout`: 5, 10, 20, 50, 100, 200 and 500.

Figure 18 shows the delivery ratio achieved by ULTR with the seven different timeouts. It seems that the delivery ratio is constantly decreasing with increasing timeout lengths. This is not surprising, since a link that has been removed from the neighbor table results in a local broadcast. All nodes that receive this message and know the intended next-but-one hop retransmit the message, adding a lot of redundancy. Therefore, removing a link too early does not result in message loss, but in unnecessary network load. However, if the link is deleted too late, i.e., a link is assumed to exist where it has already broken, the message gets lost. Therefore, when considering only the delivery ratio, using a small timeout seems favorable.

However, when the network load is considered, the choice seems to be quite the opposite. Figure 19 shows the cost of delivering a single application message measured in transmitted messages. When using the smallest timeout of 5, about

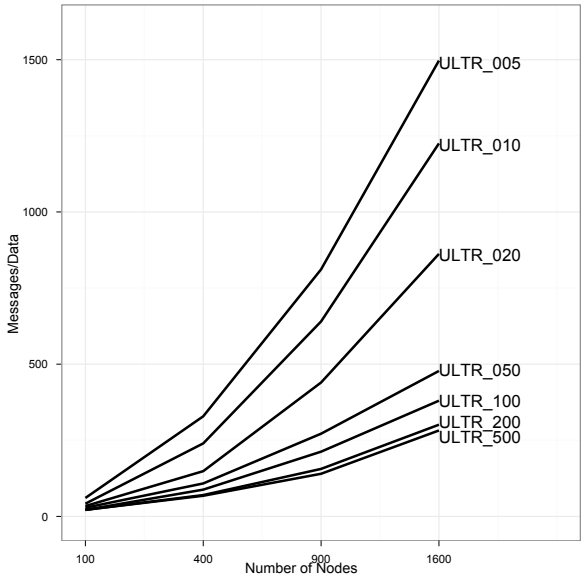


Fig. 19. Number of messages transmitted to deliver a single application message in the simulations for different timeouts

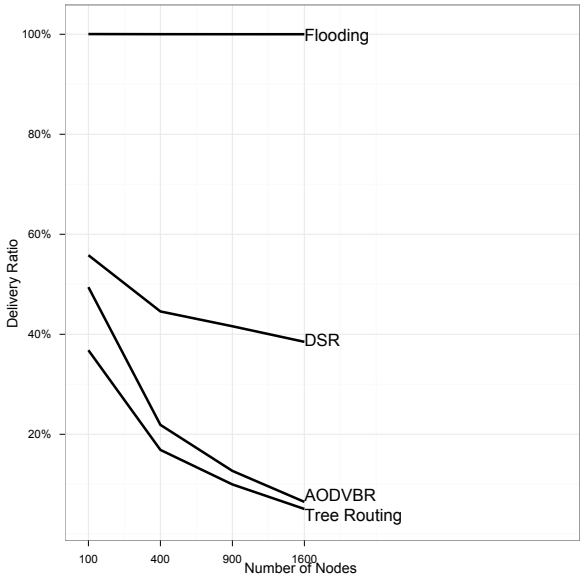


Fig. 20. Delivery ratio of AODVBR, DSR, `Flooding` and `Tree Routing`, second scenario

1,500 messages are transmitted for each application message delivered in the network consisting of 1,600 nodes, which is quite close to the cost of flooding the message. Therefore, the decision, which timeout should be used is a tradeoff between delivery ratio and network load. However, there are limits to the choice: Increasing the timeout above 200 does not change delivery ratio or efficiency much. Also, as the delivery ratio is most often more important than the network load, it is unlikely that a timeout of more than 50 would be used, because higher timeout values lead to a delivery ratio of less than 50%. Still, even this is much more than what the related work protocols achieved, making ULTR a fine choice for the evaluated network types.

VIII. Application Scenario 2: Single Pairing

In this scenario, all settings, including the number of messages a node wants to transmit, are the same as in the sense-and-send scenario. However, instead of a single sink as destination for all messages from all nodes, each node has a randomly chosen partner node it wants to communicate with. This pairing of nodes was generated before the simulations and experiments, and differs only between different network sizes: If, e.g., node 15 is the partner of node 21 for the network consisting of 36 nodes, this pairing remains fixed for all protocols as well as for simulations and real world experiments.

This pairing of nodes represents a communication pattern for MANETs and was chosen because two of the protocols used for comparison (AODVBR and DSR) are MANET protocols.

A. Simulation results

In the simulations for the single pairing scenario, the same connectivity change lists were used that have already been used in the sense-and-send scenario. However, as the destination was not a single fixed one for all nodes, the simulations were not varied according to the destination. Instead, the generated pairings were used as stated above.

Flooding was once again used to measure the upper limit for delivered messages and the delivery ratio was defined as the number of messages delivered by a protocol divided by the number of message delivered by `Flooding`.

1) Related Work Protocols: The delivery ratio of AODVBR, DSR, `Flooding` and `Tree Routing` is shown in Figure 20. For all protocols except `Flooding` the delivery ratio declines with increasing number of nodes. It can be seen that AODVBR and `Tree Routing` suffer the most from the increased route length in the larger networks, as the decline of their delivery ratio is steep. For AODVBR, building the initial route is the crucial part. When a route has been successfully established, the fish bone structure can be used to salvage data packets. But since building the initial route requires a bidirectional path and the probability of a complete path being bidirectional decreases with route length, AODVBR only works in small networks. For `Tree Routing`, building the initial route is no problem. However, due to the dynamic nature of links between nodes, the initial path is obsolete soon and the two retransmissions used as reaction to message loss are not sufficient in larger networks.

The delivery ratio of DSR also declines due to its source routing nature. However, finding the initial route is not a problem, as DSR uses one flooding for each direction. The main problem of DSR is its route maintenance mechanism. When DSR detects a route break it tries to inform the originator of the message that caused the detection of the break. Following this, a new route discovery with all its costs takes place.

The number of transmitted messages for each protocol is shown in Figure 21. Here, the impact of the route maintenance

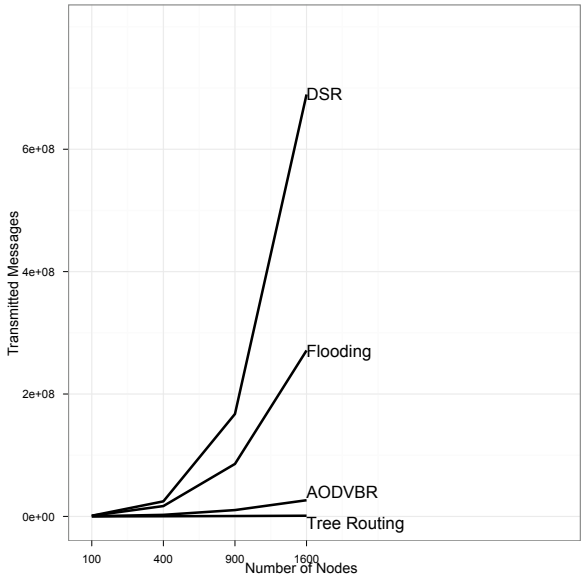


Fig. 21. Number of transmitted messages, AODVBR, DSR, `Flooding` and `Tree Routing`, second scenario

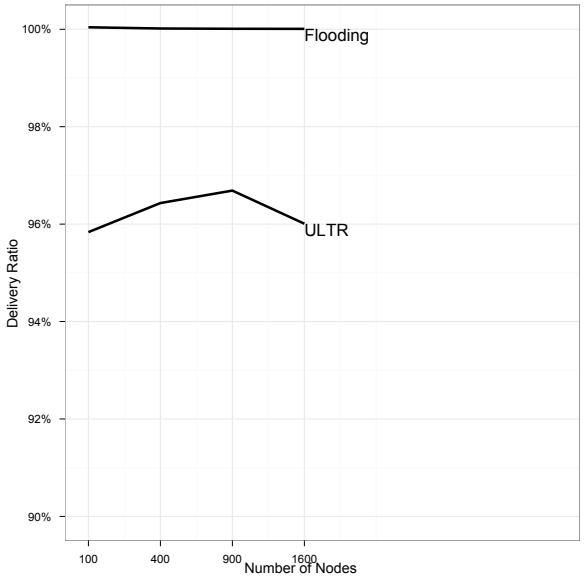


Fig. 23. Delivery ratio of `Flooding` and ULTR, scale starts at 90%, second scenario

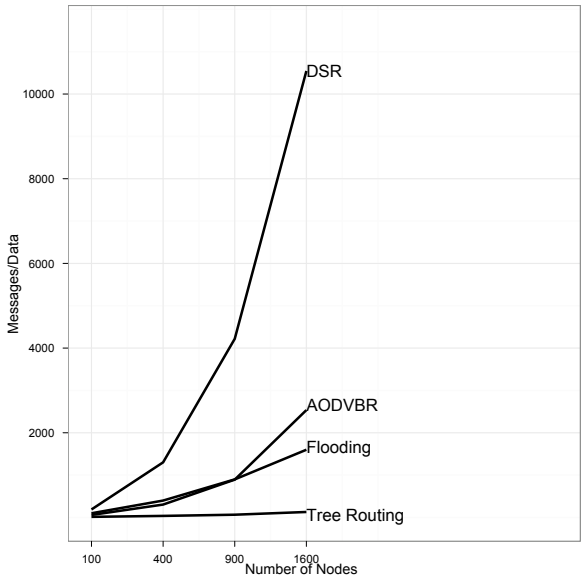


Fig. 22. Number of messages transmitted to deliver a single application message, AODVBR, DSR, `Flooding` and `Tree Routing`, second scenario

mechanism of DSR can be seen: It transmits more than twice as many messages as `Flooding` as it tries to repair broken routes. `Tree Routing` presents the other extreme, it transmits nearly no messages at all, while AODVBR needs slightly more messages.

When the network load is considered (Figure 22), the impact of the low number of messages transmitted by `Tree Routing` can be seen even better: The number of messages transmitted to deliver a single application message would suggest that `Tree Routing` is an excellent choice. However, this fact needs to be correlated with the delivery ratio in most cases, and the delivery ratio of `Tree Routing` is the lowest of all protocols. This is once again due to the length of routes. `Tree Routing` delivers a nearly constant number of data messages to the destination (roundabout 8,000) for the networks with 400, 900, and 1,600 nodes, even though the total number of application messages that is handed to the routing protocol increases proportionally to the number of nodes in the network.

2) ULTR: The delivery ratios of `Flooding` and ULTR are compared in Figure 23. Note that the scale starts at 90%. ULTR delivers well above 95% of application messages for all network sizes. In fact, the performance of ULTR seems largely independent of the network size with a slight increase from 96% to 97% for the network with 900 nodes.

Figure 24 shows the total number of messages transmitted by `Flooding` and ULTR. When these results are compared to those of the sense-and-send scenario (Figure 10), it can be seen that the number of messages transmitted by ULTR has decreased.

Figure 25 shows the number of messages transmitted by `Flooding` and ULTR in order to deliver a single application message. As the delivery ratio of both protocols is nearly equal but ULTR needs much less transmitted messages, the ratio of ULTR is much better than that of `Flooding`.

3) Comparison between all Protocols: The delivery ratio achieved by each of the simulated protocols in the single pairing scenario is shown in Figure 26. It can be seen that ULTR performs better than those chosen from related work. Moreover, the delivery ratio stays roughly the same with increasing network size. For AODVBR, DSR and `Tree Routing` the delivery ratio decreased with network size. But most interestingly, the delivery ratio of DSR improved drastically when compared to the sense-and-send scenario (see Figure 12). Also, the decline in delivery ratio with increasing

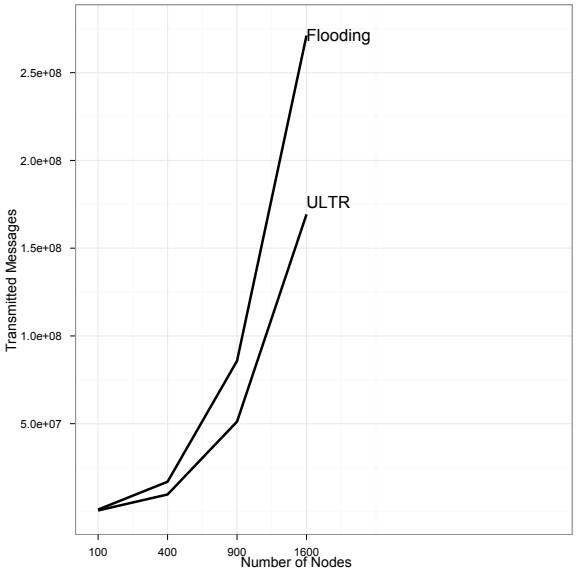


Fig. 24. Number of transmitted messages, `Flooding` and ULTR, second scenario

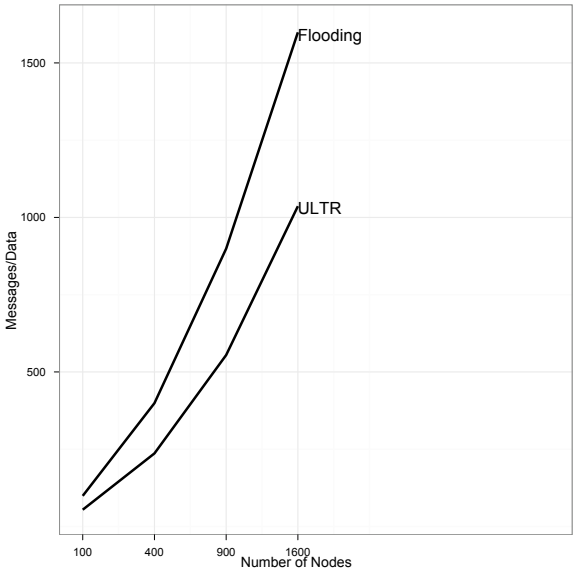


Fig. 25. Number of messages transmitted to deliver a single application message, `Flooding` and ULTR, second scenario

number of nodes is visible, but it is not as steep as for AODVBR and `Tree Routing`.

Concluding the evaluation of these simulations it can be said that DSR gained most from the change of application scenario. This was expected, as DSR was designed for MANET scenarios, not for sense-and-send scenarios in wireless sensor networks. However, the delivery ratio of ULTR is still higher than that of the related work protocols, for all network sizes.

B. Real World Experiment results

In the experiments for the single paring scenario, only two locations were used: The desk and the stone pavement. No

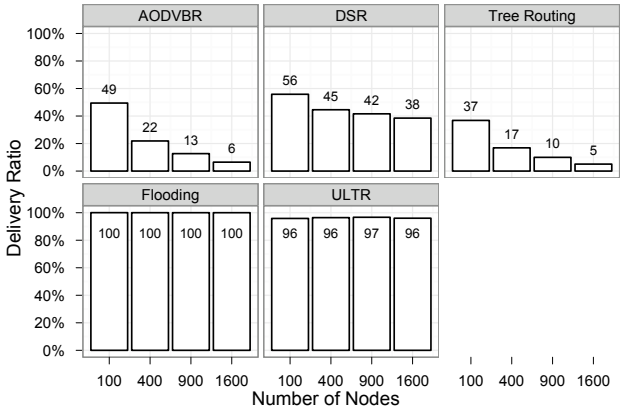


Fig. 26. Delivery ratio of all protocols for different network sizes, second scenario

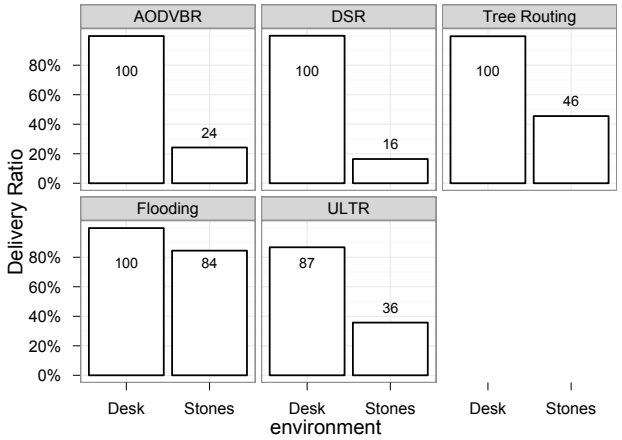


Fig. 27. Delivery ratio of each protocol achieved in the real experiments, second scenario

experiments were made on the poles, because of the similarity between pole and desk scenario. On the desk, all nodes can communicate directly while on the poles the logical distance between nodes was only 1-2 hops even in the sense-and-send scenario where the destination was on the corner of the deployed grid. The pairings used in this scenario reduce the average route length and would result in even more single hop routes for the pole scenario, making the experiments redundant. The lawn placement has been neglected due to its similarity with the stone pavement placement.

Figure 27 shows the delivery ratios of all protocols that were achieved in the real world experiments on the desk and stone pavement. With the exception of ULTR, all protocols delivered 100% of messages in the desk scenario. This behavior has also been seen in the sense-and-send scenario (see Figure 13) and can be explained by the absence of up-to-date link information. ULTR normally depends on the MAC layer or the application to deliver neighborhood information. As none was available, neither from MAC nor from the application, the current implementation relies on passive gathering of neighborhood

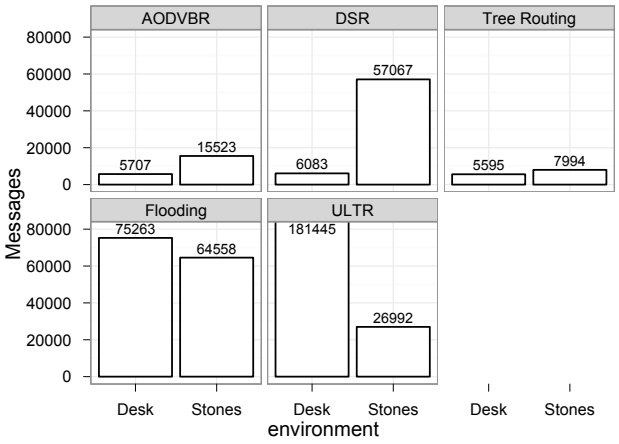


Fig. 28. Total number of messages transmitted by each protocol, second scenario

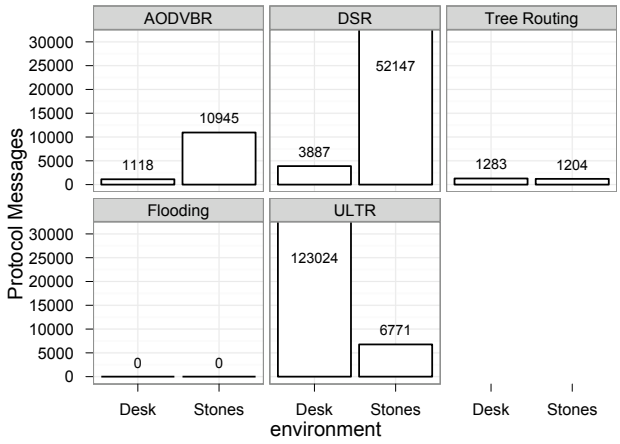


Fig. 29. Number of protocol messages transmitted by each protocol, second scenario

information by overhearing the forwarding of messages. But in a single hop environment not enough forwarded messages are overheard.

In the stone pavement experiments, even `Flooding` did not deliver all messages, which gives an insight into the MAC-layer problematic experienced more or less by all protocols. `Tree Routing` has a good delivery ratio in this scenario as it does not produce too much network load and the average path length was fairly small, making its two retransmissions a good reaction to message loss. ULTR suffers from inaccurate information in its neighbor tables, and often uses its fallback mechanism. DSR is continuously trying to repair routes, and thereby increases the network load very much, which can be seen in the next figure.

The total number of messages transmitted by each protocol is shown in Figure 28. For the experiments on the desk it can be noted that ULTR transmits more messages than `Flooding`, which can also be explained by the fallback mechanism in use: When ULTR starts route discovery, the network is flooded with a route request message. The destination receives this message and answers with a route reply but does not know if the link to the previous hop is unidirectional or bidirectional. Therefore, it uses the fallback mechanism, meaning that each node that knows the next hop forwards the message, which results in a second flooding of the network. Now that the route has been built, the data message can be transmitted. This process is repeated every time that the link timeout removes a link to the destination from a nodes neighbor table. In the stone pavement placement, the passive neighborhood discovery works much better, leading to fewer messages transmitted by ULTR. Here, DSR transmits more than 57,000 messages and thus nearly as many as `Flooding`. ULTR transmits roundabout 27,000 messages while AODVBR and `Tree Routing` transmit about 15,000 and 8,000 messages respectively. These numbers already hint at the fact that `Tree Routing` profits quite a lot from the application setting and the small network diameter.

A more detailed look at the number of messages transmitted by each protocol is given in Figure 29, where only the protocol packets are counted. Naturally, `Flooding` has the least number of protocol messages as it does not use any, and all transmitted packets are data messages. On the desk, AODVBR and `Tree Routing` transmit 1,118 and 1,283 messages respectively. As 36 nodes were present in the network, a flooding of one route request or tree building message by each node would result in 1,296 (36×36) transmissions. Therefore, these three protocols transmitted the expected number of messages. DSR and ULTR flood the network multiple times for each route discovery, resulting in an awfully high number of route request and route reply messages. For DSR this is due to the specification for the operation in the presence of unidirectional links. For ULTR it is once more due to the absence of accurate neighborhood information.

On the stone pavement, the number of protocol messages rises enormously for DSR, as a lot of link breaks lead to the creation of route error messages and subsequent new floodings of the network in order to find a new route. The lowest number of protocol messages (apart from `Flooding`) is transmitted by `Tree Routing`, which only transmits its tree building messages at the start of the experiment. When this figure is compared to the previous one, it can be seen that `Tree Routing` transmitted about 6,800 data messages (7,994 total messages - 1,204 protocol messages), meaning that most of the time the two retransmissions took place.

The number of messages transmitted to deliver a single application message is shown in Figure 30. As there were 36 nodes in the network, `Flooding` transmitted 36 messages for each data message delivered to the destination. AODVBR, DSR and `Tree Routing` transmitted exactly 3 messages for each data message received. The high number of data messages transmitted due to the inaccurate neighborhood information leads to a performance even worse than `Flooding` for ULTR.

On the stone pavement, `Tree Routing` performed best. When the delivery ratio (Figure 27) is also taken into account it can be said that for this application scenario, network size

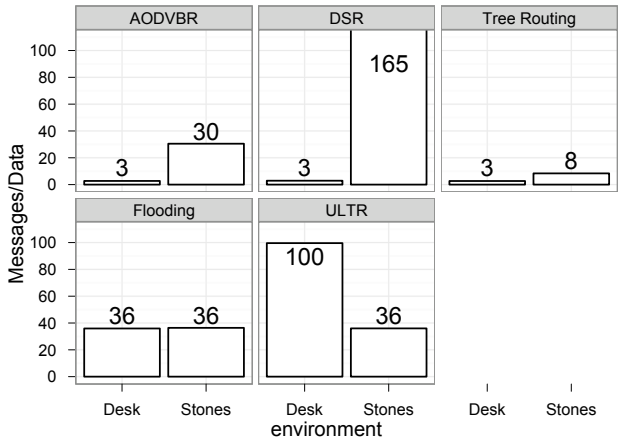


Fig. 30. Total number of messages transmitted by each protocol divided by the number of delivered data messages, second scenario

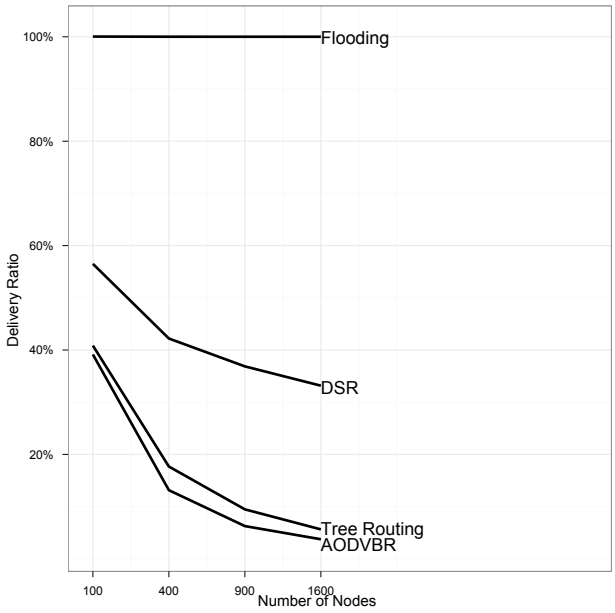


Fig. 31. Delivery ratio of AODVBR, DSR, `Flooding` and `Tree Routing`, third scenario

and placement, the choice of routing protocol should be made between `Tree Routing`, `Flooding` and ULTR. `Tree Routing` produced the least network load per application message delivered and should be chosen if some message losses could be tolerated but the network load is the most important factor. If number of delivered messages is most important, `Flooding` should be used. ULTR represents a good choice in between.

IX. Application Scenario 3: Multiple Pairings

The third application scenario, multiple pairings, once again uses the same settings as the two previous ones, only the application was changed. Instead of all nodes transmitting to a single sink or one communication partner for each node, there are multiple partners now. Each node has one communication partner at the start of the simulations/experiments and transmits the first five messages to this node. Once five messages have been transmitted, the communication partner is changed. This is repeated every time five messages have been transmitted, until the total number of messages specified (110 for simulations, 60 for experiments) has been reached. The pairings of nodes were once again generated randomly before the start, and the same pairings were used for all protocols.

This represents a MANET scenario where all nodes only want to exchange a few messages with a chosen partner before communicating with a different node. The fact that each pairing is only used for five messages results in a reduction of the importance of route maintenance. It is much more likely that a route is stable for five minutes than for a whole simulation/experiment, resulting in less route errors. Instead, route discovery rises in importance, as it is carried out after every five application messages.

A. Simulation results

The simulations once again used the connectivity change lists that were generated before the start, to keep network connectivity equal for all protocols. As in the single pairing scenario, the pairings define a different destination for each node, making the additional simulation parameter `destination` used in the sense-and-send scenario unnecessary.

The delivery ratio remains defined as the number of application messages delivered by a protocol divided by the number of messages delivered by `Flooding` in the simulations.

1) Related Work Protocols: The delivery ratio of the related work protocols is shown in Figure 31. For AODVBR and `Tree Routing`, the number of nodes and, therefore, the route length is much more important than the communication pattern of the application: The changes between single pairing and multiple pairings are marginal (compare Figure 20). The performance of `Tree Routing` increased by one percent for the largest network while that of AODVBR decreased by two percent. A bigger difference can be seen for the smaller networks, where AODVBR has lost 10% of its performance compared to the single pairing scenario in the network consisting of 100 nodes. This decrease in delivery ratio is due to the fact that building the initial route is one of the weaknesses in AODVBR. When searching for a route, the path has to be bidirectional to enable the route reply to use the same path as the route request. Once this path has been established, the fish bone structure that has been built with the route replies can be used to salvage data messages when links break. In the multiple pairings scenario, each node needs to search routes to 22 different nodes instead of only one.

The number of messages transmitted by the related work protocols is shown in Figure 32. With twice the number of transmitted messages as `Flooding`, DSR once more transmitted the most messages by far. AODVBR and `Tree Routing` transmitted far less messages, with `Tree Routing` producing the least number. When the results are compared to those of the single pairing scenario, no substantial differences can be discerned (compare Figure 21).

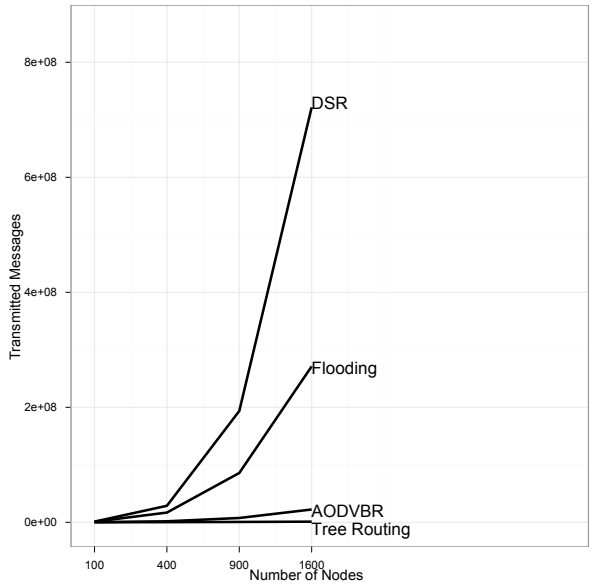


Fig. 32. Number of transmitted messages, AODVBR, DSR, `Flooding` and `Tree Routing`, third scenario

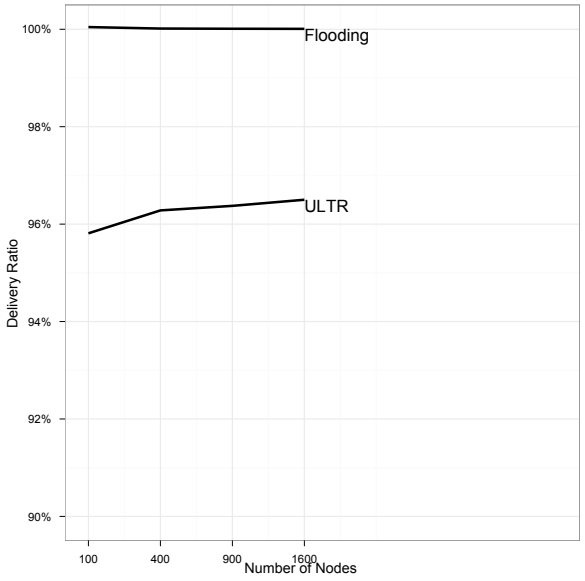


Fig. 34. Delivery ratio of `Flooding` and ULTR, scale starts at 90%, third scenario

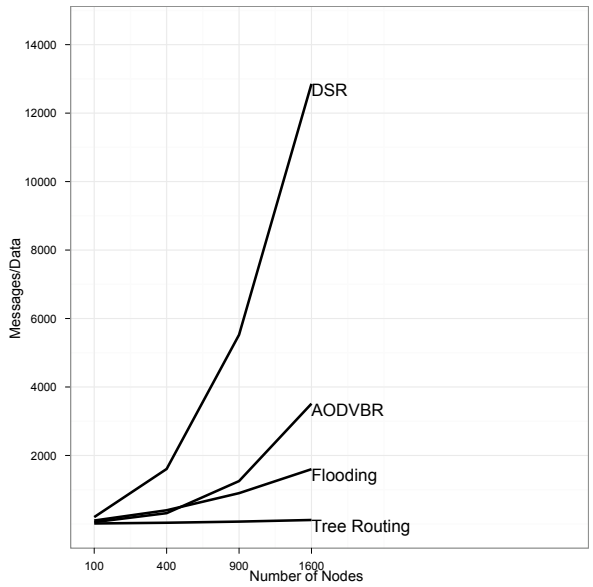


Fig. 33. Number of messages transmitted to deliver a single application message, AODVBR, DSR, `Flooding` and `Tree Routing`, third scenario

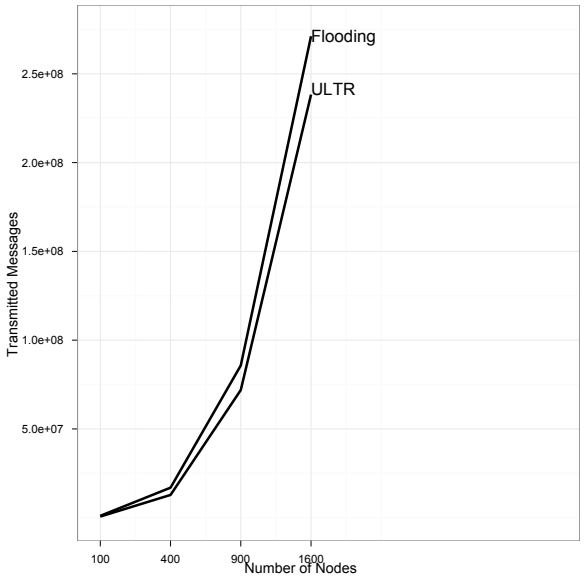


Fig. 35. Number of transmitted messages, `Flooding` and ULTR, third scenario

The cost of delivering a single data message measured in transmitted messages is shown in Figure 33. Only `Tree Routing` transmitted less messages per application message delivered than `Flooding`. This is once more due to the fact that the cost of delivery failure is small in `Tree Routing`. However, when the delivery ratio is also taken into account, `Tree Routing` looses parts of its advantage over AODVBR.

2) ULTR: The delivery ratio achieved by ULTR is compared to that of `Flooding` in Figure 34, note that the scale starts at 90%. It can be seen that ULTR delivers more than 95% of application messages, regardless of network size. It delivers between 95% and 97%, with only a low variation between network sizes.

The high number of delivered messages comes at the price of an increased number of transmitted messages, as Figure 35 confirms. Here, it can be seen that the number of messages transmitted by ULTR has risen when compared to the single pairing scenario (Figure 24). While the number is still lower than that of `Flooding`, it has gotten closer.

This high number of transmitted messages is the reason why the performance of ULTR decreases in the multiple pairings scenario. Figure 36 shows the performance of ULTR and `Flooding`, measured in messages transmitted per application

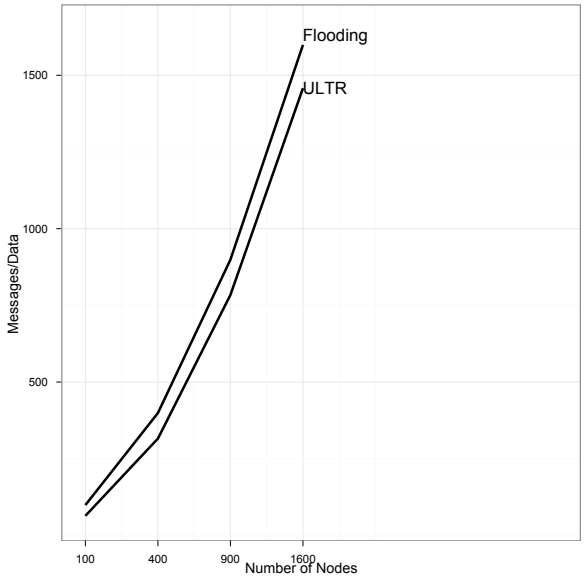


Fig. 36. Number of messages transmitted to deliver a single application message, Flooding and ULTR, third scenario

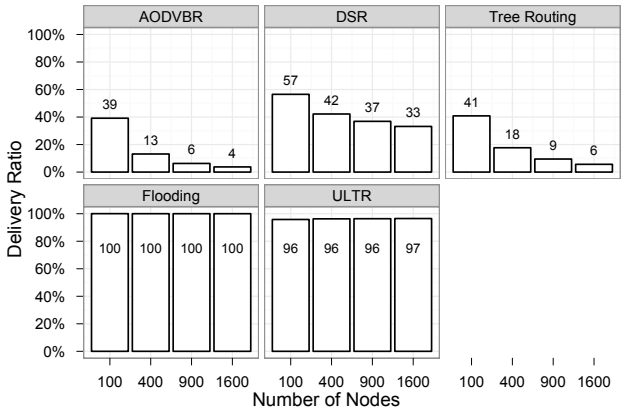


Fig. 37. Delivery ratio of all protocols for different network sizes, third scenario

message delivered. The figure shows that the performance of ULTR not much better that of Flooding in this scenario.

3) Comparison between all Protocols: The delivery ratio of all protocols is compared in Figure 37. The related work protocols, AODVBR, DSR and Tree Routing all show a steep decline in delivery ratio. Interestingly, the decline of delivery ratio is not as steep for DSR as it is for AODVBR and Tree Routing. This is due to the fact that DSR has a better route discovery mechanism. While flooding the whole network twice in order to establish a route produces a lot of network load, it also means that a route will be found in most cases. Only if network separation occurred, no route would be found. How long a route found this way can be used depends on link stability, however. But since it only needs to be used for five messages before a different destination is selected, there is a good chance some of the five messages can

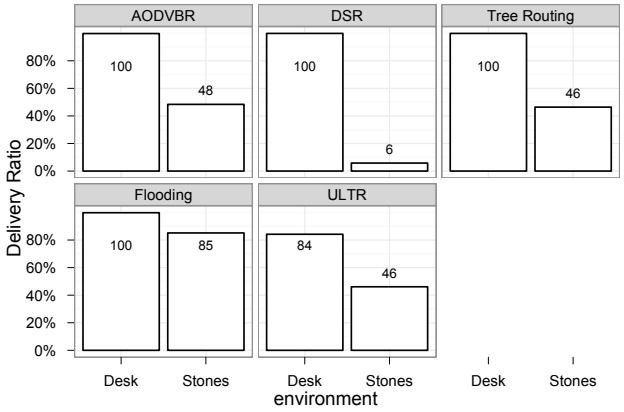


Fig. 38. Delivery ratio of each protocol achieved in the real experiments, third scenario

be transmitted successfully. This can be seen in the network with 1,600 nodes, where DSR was able to deliver one third of application messages, meaning that between one and two messages were delivered to each destination on average. ULTR increases its performance with increased number of nodes.

B. Real World Experiment results

The experiments for the multiple pairings scenario featured the same settings and locations as the experiments for the single pairing scenario (Section VIII-B): The desk placement was used as single hop, and the stone pavement as multihop environment. The pole placement would have been redundant to the desk placement while the lawn placement would have been similar to the stone pavement environment.

The delivery ratio achieved by all protocols in the multiple pairing scenario is shown in Figure 38. In the desk experiments, all protocols reached 100% delivery ratio except for ULTR. This is due to the passive neighborhood discovery: Only when the forwarding of a message is overheard by a node that has already forwarded that message and is listed as last hop, the neighborhood discovery assumes bidirectional links. Otherwise, links are assumed to be unidirectional. This leads to a lot of mistakes, as nodes do not need to forward messages in a single hop environment, meaning that all links in the network are assumed to be unidirectional. Therefore, the backup mechanism is always used unnecessarily, resulting in a high network load, which in turn leads to more collisions and message losses.

On the stone pavement, Flooding delivers most application messages, followed by AODVBR, ULTR and Tree Routing, which deliver about half of the messages transmitted. Only DSR is far worse, with a delivery ratio of only 6%.

The total number of transmitted messages is shown for all protocols in Figure 39. ULTR once more has the highest number of transmitted messages for the single hop environment due to the problems with the neighborhood detection. On the stone pavement, the passive neighborhood detection works better, and the number of transmitted messages is

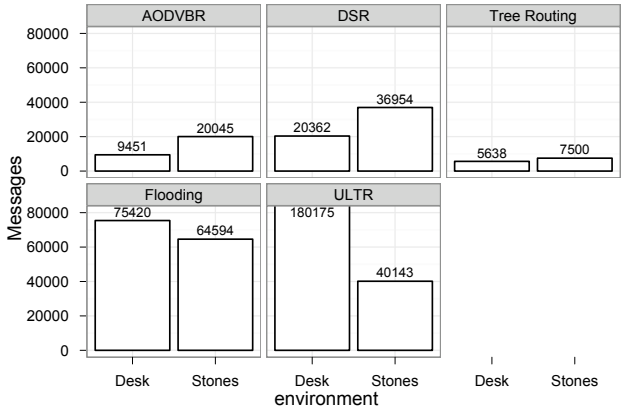


Fig. 39. Total number of messages transmitted by each protocol, third scenario

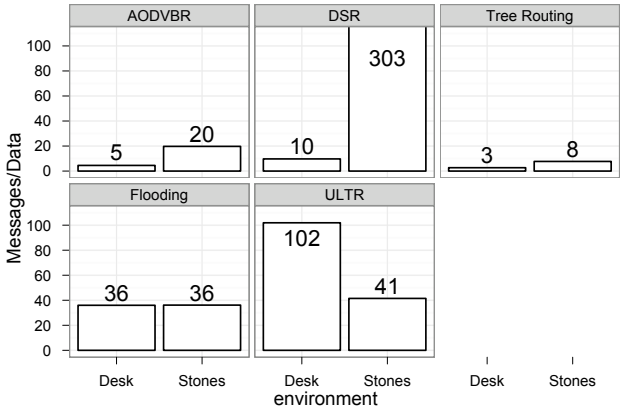


Fig. 41. Total number of messages transmitted by each protocol divided by the number of delivered data messages, third scenario

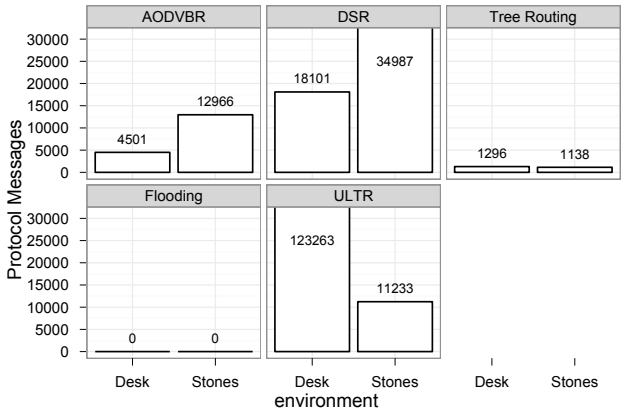


Fig. 40. Number of protocol messages transmitted by each protocol, third scenario

reduced. There, `Flooding` transmits the greatest number of messages while `Tree Routing` transmits the smallest. Still, when considering that only 2,160 application messages were generated it can be seen that `Tree Routing` often used its two retransmissions.

The number of protocol messages transmitted by each protocol can be seen in Figure 40. `Flooding` naturally did not transmit any protocol messages, while ULTR transmitted the most protocol messages in the desk scenario.

On the stone pavement, `Tree Routing` needed the least number of protocol messages, apart from `Flooding`. DSR transmitted the most protocol messages, followed by AODVBR and ULTR.

The number of transmitted messages divided by the number of delivered application messages is used to measure the performance of all protocols in Figure 41. For the desk placement, `Tree Routing` shows the best performance, directly followed by AODVBR.

When the sensor nodes were placed on the stone pavement, `Tree Routing` needed the least number of transmissions to deliver a single application message, which is once again due to the low cost of delivery failure. When only the cost of an application message delivery is considered, `Tree Routing` performs best. However, `Flooding` delivered nearly twice as many messages but needs to transmit three times more messages per delivered application message to reach this increase in delivery ratio. If the delivery ratio is most important, `Flooding` would be chosen for such small networks and this application scenario. If the network load is more important, `Tree Routing` should be chosen.

X. Conclusion

Unidirectional links present a challenge for routing protocols and there are a number different ways of dealing with them.

In this paper we presented ULTR, a routing protocol that uses unidirectional links explicitly, meaning that it needs to know about their existence. This knowledge can either be achieved through a neighborhood discovery protocol, or implicitly by overhearing transmissions.

The version of ULTR described and evaluated in this paper follows the second approach. We compared the delivery ratio and the cost associated with it for ULTR in the version with passive neighborhood discovery and four protocols from related work in three scenarios and showed their advantages and limitations. The evaluation featured simulations as well as experiments with real sensor nodes.

References

[1] R. Karnapke and J. Nolte, “Unidirectional link triangle routing for wireless sensor networks,” in *Proceedings of the seventh International Conference on Sensor Technologies and Applications*, 2013, pp. 7–14.

[2] S. Lohs, R. Karnapke, and J. Nolte, “Link stability in a wireless sensor network - an experimental study,” in *3rd International Conference on Sensor Systems and Software*, 2012, pp. 146 – 161.

[3] L. Sang, A. Arora, and H. Zhang, “On exploiting asymmetric wireless links via one-way estimation,” in *MobiHoc ’07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM Press, 2007, pp. 11–21.

[4] V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt, “The heathland experiment: Results and experiences,” in *Proceedings of the REALWSN’05 Workshop on Real-World Wireless Sensor Networks.*, Jun 2005. [Online]. Available: citeseer.ist.psu.edu/732032.html

[5] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2003, pp. 1–13.

[6] M. K. Marina and S. R. Das, "Routing performance in the presence of unidirectional links in multihop wireless networks," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, ser. MobiHoc '02. New York, NY, USA: ACM, 2002, pp. 12–23. [Online]. Available: http://doi.acm.org/10.1145/513800.513803

[7] C. E. Perkins and E. M. Royer, ""ad hoc on-demand distance vector routing."," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA*, FEB 1999, pp. 90–100.

[8] IETF, "Unidirectional link routing (udlr)." [Online]. Available: http://datatracker.ietf.org/wg/udlr/charter/ last acessed May 2014

[9] S. Mank, R. Karnapke, and J. Nolte, "Mac protocols for wireless sensor networks: Tackling the problem of unidirectional links," in *International Journal on Advances in Networks and Services, vol 2 no 4*, 2009, pp. 218 – 229.

[10] ——, "Mlmac-ul and ects-mac - two mac protocols for wireless sensor networks with unidirectional links," in *Third International Conference on Sensor Technologies and Applications*, Athens, Greece, 2009.

[11] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *Proc. 14th Intl. Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, Apr. 2006.

[12] T. L. Dinh, W. Hu, P. Sikka, P. Corke, L. Overs, and S. Brosnan, "Design and deployment of a remote robust sensor network: Experiences from an outdoor water quality monitoring network," in *LCN '07: Proceedings of the 32nd IEEE Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 799–806.

[13] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh, "Vigilnet: An integrated sensor network system for energy-efficient surveillance," *ACM Trans. Sen. Netw.*, vol. 2, no. 1, pp. 1–38, 2006.

[14] D. Johnson, D. Maltz, and J. Broch, *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*. Addison-Wesley, 2001, ch. 5, pp. 139–172.

[15] S.-J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2000)*, Chicago, IL, Sepember 2000. [Online]. Available: citeseer.ist.psu.edu/lee00aodvbr.html

[16] E. Nordström, P. Gunningberg, C. Rohner, and O. Wibling, "Evaluating wireless multi-hop networks using a combination of simulation, emulation, and real world experiments," in *MobiEval '07: Proceedings of the 1st international workshop on System evaluation for mobile platforms*. New York, NY, USA: ACM, 2007, pp. 29–34.

[17] I. Stojmenovic, "Simulations in wireless sensor and ad hoc networks: matching and advancing models, metrics, and solutions [topics in ad hoc and sensor networks]," *IEEE Communications Magazine*, vol. 46, no. 12, pp. 102–107, 2008.

[18] A. Varga, "The omnet++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, Jun. 2001.

[19] A. Koepke, M. Swigulski, K. Wessel, D. Willkomm, P. Klein Haneveld, T. Parker, O. Visser, H. Lichte, and S. Valentin, "Simulating wireless and mobile networks in OMNeT++: The MiXiM vision," in *1st Int. Workshop on OMNeT++*, mar 2008. [Online]. Available: http://www.st.ewi.tudelft.nl/ koen/papers/mixim.pdf

[20] R. Karnapke, S. Lohs, A. Lagemann, and J. Nolte, "Simulation of unidirectional links in wireless sensor networks," in *7th International ICST Conference on Simulation Tools and Techniques, Lisbon, Portugal*, 2014.

[21] "Texas instruments ez430-chronos." [Online]. Available: http://focus.ti.com/docs/toolsw/folders/print/ez430-chronos.html?DCMP=Chronos&HQS=Other+OT+chronos last acessed May 2014

[22] K. Walther, "Ein ereignisbasiertes betriebssystemkonzept für tief eingebettete steuersysteme," Ph.D. dissertation, BTU Cottbus, 2009. [Online]. Available: http://opus.kobv.de/btu/volltexte/2009/772/

[23] A. Lagemann and J. Nolte, "Integration of event-driven embedded operating systems into omnet++ – a case study with reflex," in *2nd International Workshop on OMNeT++*, Rome, Italy, March 2009.