Efficient Mobile Data Collection with Mobile Collect

Navid Hassanzadeh*, Olaf Landsiedel†, Frederik Hermans‡, Olof Rensfelt‡, Thiemo Voigt*
*Swedish Institute of Computer Science, Sweden, {navid, thiemo}@sics.se
†KTH, Stockholm, Sweden, oland@kth.se
‡Uppsala University, Sweden {frederik.hermans, olofr}it.uu.se

***Abstract*—The main task of most deployed wireless sensor networks is data collection. While a number of solutions have been designed for static networks, there are currently no widely used data collection algorithms for mobile sensor networks. In this paper, we concentrate on scenarios where many nodes, both data sources and sinks, move along a certain track in one direction, a scenario that is common in sports events. Rather than designing a new protocol from scratch, we extend an existing data collection protocol with lightweight mechanisms to make it efficient for mobility. Our extensive simulations and results in a testbed that includes mobile robots demonstrate that our solution is able to achieve high packet delivery rates at low energy consumption. For our target scenario, our solution more than doubles packet delivery rates when the network is sparse. Our solution also works well in scenarios with a higher degree of mobility where nodes move according to a more demanding random waypoint model.**

I. Introduction

Most of the existing sensor network applications and protocols are designed to collect data from a number of sensors distributed around a certain area. These sensor networks are often static in the sense that nodes do not move. There are also networks that incorporate some mobile nodes. In many cases, there is one mobile node that acts as a base station and travels closer to the data sources in order to save energy [1]. There are also scenarios where nodes are attached to mobile objects, in many cases animals, for examples zebras [2] or rats [3]. These networks often use delay-tolerant networking approaches since there is no need for real-time data and since the network is usually sparse and encounters are rare events.

In this paper we target another scenario namely data collection for sensor networks where all nodes are mobile, both data sources and sinks. In particular, we are interested in events where all nodes move along one track in the same direction. While there are many sports events with such movement patterns one could also imagine a big hiking, nordic walking or a roller blading event [4]. Even though many people participating in such events carry mobile phones, for a third party service provider that wants to offer services based on real time data collected at the event, it is much cheaper and more practical to equip the majority of the participants with cheap sensor nodes rather than mobile phones and give a subset of the nodes Internet access via e.g. GSM. This way, we have a scenario where many mobile nodes including some base stations move along the track in the same direction. In this paper, we target reliable and energy-efficient data collection in such networks.

Since in our scenario the degree of mobility [5] is quite low, we opt for modifying a data collection protocol for static sensor networks rather than designing a protocol from scratch. Towards this end, we modify the Contiki Collect protocol [6], a protocol similar to the Collection Tree Protocol (CTP) [7] for Tiny OS, to make it more suitable for mobile scenarios. We enhance Contiki Collect with mechanisms to detect and repair loops since these occur more often in mobile than in static scenarios. Furthermore, we also enable nodes to more quickly find new parents as in mobile scenarios nodes often move out of range. Moreover, we provide an implementation for the Contiki operating system [8].

We perform experiments both in simulation and experiments on real hardware on a sensor node testbed that includes mobile robots. In the experiment we use two different MAC layers: Contiki's default MAC layer ContikiMAC [9] and an implementation of A-MAC [10] for Contiki. Our results show that the resulting protocol that we call Mobile Collect has low overhead and is able to achieve a high packet delivery rate at low cost in the target scenario where all nodes move along a track in the same direction. When the network is sparse, the packet delivery rate more than doubles with Mobile Collect compared to the Contiki Collect protocol. We also perform experiments using a more demanding random waypoint mobility model with a higher degree of mobility [5]. Our experiments show that Mobile Collect performs very well in such a scenario when it is run on top of A-MAC: Mobile Collect is able to sustain a high delivery rate of around 70% at a low energy consumption of 10 mJ per received packet even in scenarios where nodes move quite fast with speeds between 2 and 8 m/s.

The main contributions of this paper are:

- We design and implement Mobile Collect, a data collection protocol for scenarios where both sinks and sources move along a track.
- We evaluate Mobile Collect both in simulation and on real hardware in a testbed that includes mobile robots demonstrating that Mobile Collect achieves a high delivery rate at a low energy cost per received packet.
- Our results also show that even in scenarios with a higher degree of mobility such as a random waypoint model, data collection is possible without using a MAC layer

that is designed for mobile sensor networks. Instead, the lightweight mechanisms we use to enhance an existing data collection protocol for static networks are sufficient to achieve good performance in many scenarios.

The paper proceeds by discussing related work in the next section. Section III explains our major design choices. We present simulation results in Section IV and results in a testbed with real hardware in Section V before we conclude the paper.

II. Related Work

We divide the related work in three main areas: approaches that are based on delay-tolerant networking (DTN) and approaches that consider mobility of the data sink as well as MAC layers designed for mobility.

A. Delay-tolerant networking approaches

Mo and Fall early suggested to use delay-tolerant approaches in wireless sensor networks [11]. Many applications have been developed following this approach, in particular in the area of wildlife monitoring. ZebraNet is one of these wild life monitoring applications where the sensor nodes are attached to zebras [2]. The data is distributed among the zebras' nodes until a mobile sink node (usually attached to a vehicle) comes in the vicinity of the zebras to collect their data for off-line data analysis. Also Ratpack uses a similar delay-tolerant networking approach for data collection [3]. Similar to our approach, in these applications all nodes are mobile. In contrast to our approach that opts for low delay, they collect the data for off-line analysis.

Similar to delay-tolerant networking are the concepts of data muling and opportunistic networking. A number of researchers have employed these concepts to collect data with mobile muling entities [12], [13], [14].In these approaches some of the sensor nodes are not mobile but the forwarding nodes are. In our approach, also the sensor nodes are mobile and we aim towards data collection with low delays.

B. Mobile sinks

There are a number of protocols that consider sink mobility. Probably the most well-known protocol is the Whirlpool Routing Protocol (WARP) [15]. WARP is an extension of CTP. When a sink moves the existing distance vector tree searches an old location to find a possible neighbor node with connection to the sink node. Then it quickly switches to this neighbor which has a path to the sink node. Another class of sink mobility works are those that consider mobile sinks as data collectors and where the research challenge is to compute optimal paths for the sink through the sensor network to minimize the energy consumption of the network as a whole. While most of the approaches have been evaluated in simulation only, Mudigonda et al. have experimentally compared several approaches [1]. In contrast to these approaches, we consider scenarios where not only the sink but all nodes are mobile.

C. MAC Layers for Mobility

There are quite a few MAC layers specifically designed for mobile scenarios. These include MMAC [16], MS-MAC [17] and AM-MAC [18]. All of these MAC protocols are evaluated by simulation only whereas we also provide implementations on real hardware. Furthermore, our results also indicate that in many scenarios a dedicated MAC layer for mobility is not necessary. Note that for example, the simulations in MMAC were performed with an average speed of 0.1 m/s [16] whereas in our experiments nodes move much faster.

III. Designing Mobile Collect

Mobile Collect introduces adaptive routing mechanisms to extend collection tree protocols to the mobile domain. Our target scenario (see Section I), has a number of key properties: (1) Nodes are rarely disconnected if the network is dense enough. Hence, while the topology itself changes frequently, a node commonly has a couple of other nodes in communication range. (2) Nodes stay in range over time periods of tens of seconds to minutes. In our evaluation we show that these two observations enable us to achieve high reliability and energy efficiency without the need to develop a new protocol or integrate mechanisms from Delay Tolerant Networking (DTN).

We base our work on Contiki Collect, a data collection protocol for the Contiki OS similar to the Collection Tree Protocol (CTP) [7]. As CTP, Contiki Collect uses ETX as a route selection metric. In contrast to CTP that uses the 4-bit link estimator to estimate the ETX, Contiki Collect uses explicit unicast probe messages which simplifies the discovery and utilization of new paths [6]. Hence, we assume that Contiki Collect is more efficient than CTP in dynamic topologies even though none of the protocols is primarily designed for mobile scenarios.

In this section we first identify the key challenges in mobile scenarios that reduce protocol performance. Next, we introduce our adaptive extensions.

A. Challenges in Mobile Scenarios

Based on experimental traces from both simulation and deployments, we identified the following key challenges in mobile settings.

1) Timeout Handling: After sending a packet, Contiki Collects waits for a predefined time (retransmission timeout) to receive an acknowledgment. If this timeout triggers, it retransmits the packet for a predefined number of tries. The protocol associates the packet loss with decreased link quality. Hence, on a timeout it increases the routing metric (ETX) to punish the timed-out route. However, in mobile settings, the majority of packet losses do not come from decreased link quality, but from nodes moving out of range. Hence, instead of retransmitting packets for a number of times, we argue that a timeout should trigger a node switching to a new route.

2) Routing Loops: Contiki Collect makes use of the same approach as CTP to detect loops: Packets should only traverse the routing tree along a decreasing routing gradient. Hence, a forwarder shall have a lower routing metric than previous

nodes. If a forwarder receives a packet from a child node that has a lower routing metric than itself, the forwarding node will send a notification to the child node. The child node then either updates its routing metric or selects another parent. In our experiments we noted that this technique is not sufficiently agile for mobile scenarios.

B. Introducing Mobile Collect

After identifying the key challenges for Contiki Collect, we introduce our extensions to ensure high reliability and energy efficiency in mobile settings: (1) Parent switch on timeout and (2) avoiding routing loops.

1) Parent Switch on Timeout: In mobile scenarios, we assume that a timeout indicates that the target node has disappeared from the communication range of the sending node. Thus, instead of punishing the timed-out route by slightly increasing its routing metric ETX, we increase the ETX to the maximum value which enforces a parent switch. Next, the source node applied two strategies to repair the routing topology:

- **Local Repair:** If the node has other potential parents in its routing table, it will try to connect to one offering the best routing progress and broadcast its new routing metric to its child nodes (see Figure 1a). Child nodes may reconnect to other parents, if these offer better routing progress when compared to the new routing metric of the current parent.
- **Global Repair:** If the local repair fails, the node will by setting its routing metric to the maximum value, signal its child nodes that it is not available anymore for forwarding (see Figure 1b). Based on their routing table entries (or after discovering new neighbors) nodes will connect to new parents and repair the routing topology (see Figure 1c).

Overall, these techniques are also applied in both CTP and default Contiki Collect. However, Mobile Collect employs it significantly more aggressive as it switches parents after a single timeout. While this agility allows Mobile Collect to cope with high degrees of topology dynamics, it also introduces an increased risk of loops, which we discuss next.

2) Avoiding Routing Loops: The dynamic topology caused by the mobility of nodes in our application scenarios and the agile parent change in Mobile Collect, increase the risk of routing loops (see example in Figure 2). In this section we discuss our extensions to avoid loops. If a packet is trapped in a routing loop, it repeatably traverses it until its time to live (TTL) expires and it is dropped. Hence, routing loops strongly decrease reliability while increasing network load and energy consumption.

Mobile Collect extends Contiki Collect by enabling a node to track the parents of all its neighbors. Thus, nodes in Mobile Collect announce the IDs of their parents in their routing beacons. This allows us to implement two mechanisms to prevent loops: (1) sibling suppressions and (2) triangle suppression.

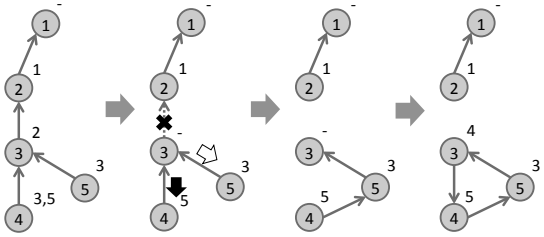
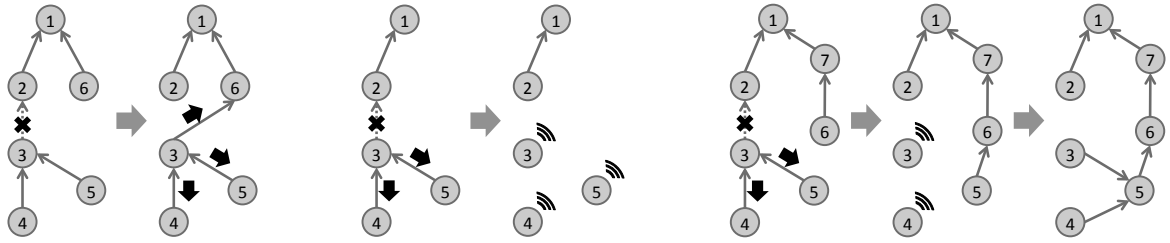


Fig. 2: Loop Example: an update message of the routing metric is not received by all neighbors (white arrow marks packet loss, numbers next to a node indicates its routing table entries). As Node 3 looses its connection to the sink, it signals to its sibling to disconnect. As node 5 did not receive this update, it still announces that it can provide a route to the sink. Hence, node 4 connects to it and consequently node 3 connects to 4. Mobile Collect prevents these loops. Note that, a parent change triggers a node to broadcast its new routing metric.

- **Sibling Suppression:** Upon loosing its parent, a node also disconnects from all other nodes that share this parent. Thus, it blocks connecting to these until they have updated their routing metric after finding a new parent. Employing this to the example depicted in Figure 2, sibling suppression would prevent node 4 from choosing node 5 as new parent, as both had node 3 as parent.
- **Triangle Suppression:** Additionally, we employ traditional triangle suppression to avoid routing triangles. Knowing the parents of all its neighbors, a node avoids closing a routing triangle. Hence, it protects node 3 from choosing node 4 as parent. This approach is similar to loop handling strategies used by traditional distance vector routing protocols such as RIP [19].

Although it is desirable to prevent a loop creation rather than detecting and repairing it, sometimes loop repairing is inevitable. To repair a loop, we adhere to the mechanism used in Contiki Collect: A forwarding node should always have a lower routing metric than its predecessor. Otherwise, this indicates a loop in the routing topology. In case a routing loop is detected, a forwarding node sends a notification packet to its predecessor to notify it about the forwarders routing metric. However, this approach is not agile enough in mobile scenarios. To avoid persistent loops, Mobile Collect extends this by setting the routing metric of the forwarder to infinity, forcing the predecessor to find a new parent.

Motivated by our initial analysis of shortcomings in Contiki Collect in mobile settings, we show that by merely extending two mechanisms in Contiki Collect we integrate mobility (see Figure 3). First, agile parent switching allows us to quickly adapt to topology changes. Second, our mechanisms for loop avoidance ensure that our dynamic topologies remain loop-free. Next, we show in our evaluation that these two lightweight extensions enable reliable data collection in mobile settings at low energy consumption.



(a) Local topology repair by choosing a new parent.

(b) Local repair fails, the network disconnects.

(c) Nodes reconnect after discovering new neighbors.

Fig. 1: Basic mobility mechanism in Mobile Collect: Parent switch triggered by packet loss. Please note, when a node in Contiki Collect has no parent, it engages in frequent beaconing to detect new neighbors. Also, in Mobile Collect a parent change triggers a node to broadcast its new routing metric.

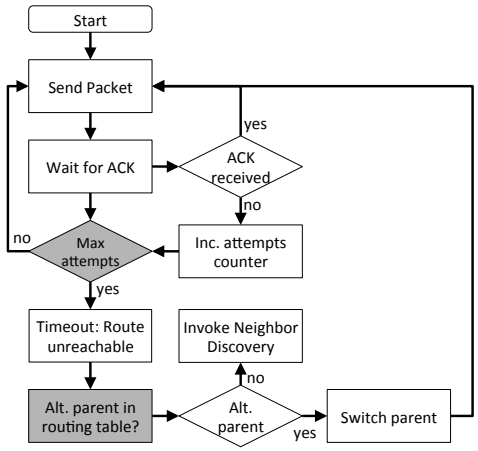
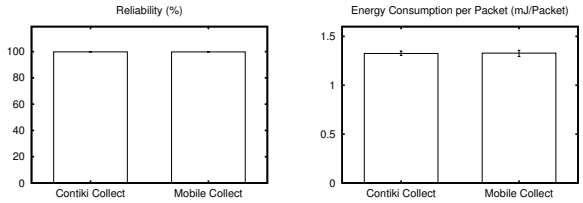


Fig. 3: System diagram of Mobile Collect for sending a packet: To enable reliable communication in mobile settings, we make only a small number of extensions to Contiki Collect, these are marked in gray.



(a) Mobile Collect is almost as reliable as Contiki Collect

(b) Mobile Collect's energy consumption is similar to Contiki Collect's

Fig. 4: In a static scenario Mobile Collect performs similar to Contiki Collect which indicates low overhead.

IV. Simulations

In this section we present simulation results that demonstrate the efficiency of Mobile Collect.

A. Simulation Setup

For our simulations we use the Contiki simulator COOJA that simulates networks of Contiki nodes [20]. COOJA executes deployable Contiki code, i.e., we run an implementation of Mobile Collect that is also executable on real hardware. We use BonnMotion [21] to generate the mobility scenarios for our experiments. If not mentioned otherwise, we simulate 50 nodes out of which three are sink nodes. We simulate with two different nodes speeds: A slower speed with node speeds between 2 and 8 m/s and a faster one with node speeds between 5 and 15 m/s which corresponds to speeds ranging from quick walking to fast roller blading. When not noted otherwise, Mobile Contiki uses ContikiMAC as the MAC layer, as it is the default MAC layer in Contiki. Contiki Collect always uses ContikiMAC since it is optimized for ContikiMAC.

Additionally, we ported A-MAC to Contiki and use it as MAC layer in Mobile Collect, as it promises an increased performance in dense networks [10]. To integrate Mobile Collect into A-MAC, we extended the probes of A-MAC by one field: We announce the routing metric of a node in is MAC-layer probes. As a result, we reduce the overhead of beacons and metric updates, that are now part of the frequently sent beacons.

B. Performance in Static Scenarios

In this experiment, we compare the efficiency of Mobile Collect to Contiki Collect in a static scenario where nodes do not move. The goal with this experiment is to evaluate the overhead of Mobile Collect. The simulation scenario consists of 100 nodes one of which is the sink node. We run three rounds of simulations. Each run lasts about one simulated hour during which nodes send around 5000 packets.

We depict the results in Figure 4. The results show that Mobile Collect has roughly similar performance as Contiki Collect. The energy consumption per packet of Mobile Collect is slightly higher than Contiki Collect's. This is expected since Mobile Collect interprets packet loss and the subsequent timeout as a loss of route and generates unnecessary beacon packets which increases power consumption. As the topology in this experiment is quite dense this happens frequently in our scenario. In summary, the results show that the overhead of Mobile Collect is low.

C. Performance under the Highway Mobility Model

In the experiment in this section we investigate the performance of Mobile Collect for our target scenario, i.e., nodes moving together along a certain track but with different speeds as in the Roller Blading scenario [4]. For the mobility

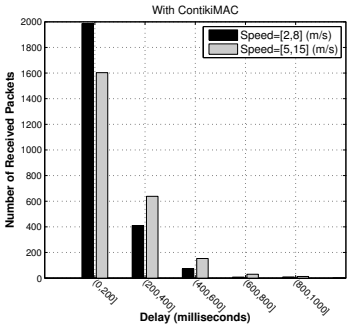


Fig. 6: The delays with Mobile Collect are small.

generation, we extended BonnMotion with a unidirectional highway model [22]. For our experiments we set the number of lanes in the highway model to five.

The scenario includes 50 nodes with three of them being sink nodes. We use ContikiMAC as the MAC protocol. In the simulations we vary the transmission range of the nodes. Based on the transmission range and output of BonnMotion we calculate the density as the average node degree.

Figure 5 shows our results. The figure shows improved performance for Mobile Collect compared to Contiki Collect. The largest improvements are for low densities. For a node degree of 1.7, the packet reception rate of Mobile Collect is three times higher than that of Contiki Collect. Figure 5b shows that also the energy consumption of Mobile Collect per received packet is three times lower at this density. At higher densities the difference between Mobile and Contiki Collect decreases since high densities resemble static scenarios in particular for the lower node speed. As expected, both data collection protocols perform better at lower node speed. We also perform experiments with A-MAC. As the results are similar to the ones presented here we do not show them.

In contrast to approaches based on delay-tolerant networking, our scenarios require data collection with short delays. Figure 6 shows that most of the delays are lower than 400 ms. As expected, the delays are lower when the sensor nodes move slower since more routes are lost when nodes move faster and lost routes obviously increase the delay. Not shown in the figure is the maximum delay of all packets that is less than 3 seconds when the average speed of nodes is between 2 and 8 m/s and around 8.2 seconds for nodes that move with an average speed between 5 and 15 m/s.

In summary, the results show that our design decisions discussed in Section III improve the performance of Mobile Collect in the target scenario and is able to provide data collection with short delays.

D. Performance under the Random Waypoint Model

Although our main target scenarios are those where sensor nodes move along a certain track in the same direction (which corresponds to the unidirectional highway model), in this experiment we also evaluate the performance of Mobile Collect under random movement. Random movement is much more challenging for data collection than moving along a track and hence we want to explore the limitations of our approach. In this experiment we use a random waypoint model [23] with a pause time of zero as a mobility model and each node chooses its random speed from a [2,8] m/s interval. As we expect that in these scenarios the MAC layer has an impact on performance, we evaluate Mobile Collect both on ContikiMAC and A-MAC in this experiment.

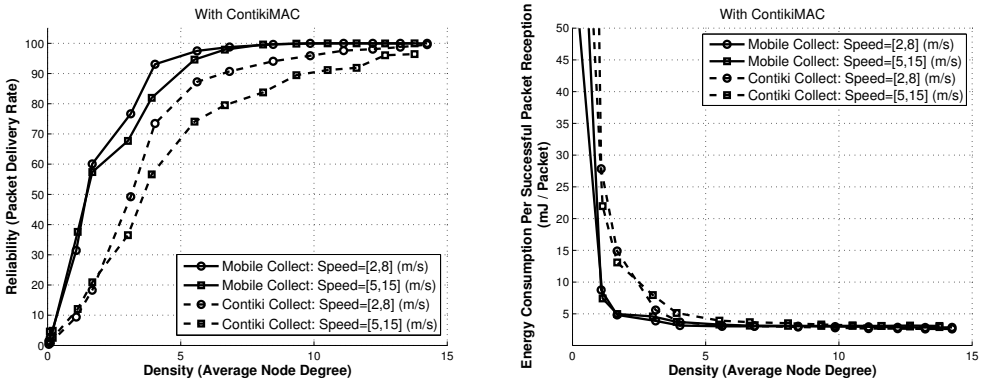
The results in Figure 7 show that Contiki Collect's packet delivery rate increases as the density increases, similar to what happens in the highway scenario, see Figure 5. Contiki Collect's packet reception rate is, however, always under 30% even when the node density is high. These results show that Contiki Collect cannot cope with the high degree of mobility in this scenario.

The figure also reveals a behaviour that depends very much on the MAC layer. The packet delivery rates of Mobile Collect on both A-MAC and ContikiMAC show similar trends. They increase steadily up to a node degree of around 3-4. Then they decrease up to a node degree of about 7 to 8 where Mobile Collect's reliability increases on top of A-MAC but plains out on top of ContikiMAC. At very low densities, the network is very sparse but has a very low diameter, i.e., most of the packets that are delivered are delivered in one hop as our log files reveal. The average number of hops a packet travels before it reaches the sink increases from around one to 3.5 as the density increases to 14. More hops have a negative impact on reliability but this is partly compensated by the higher density which makes it easier for a child to find parents for the next hop.

The behaviour in Figure 8a explains why the packet delivery rate for Mobile Collect on top of A-MAC increases for higher node degrees. This figure shows that at high densities the number of duplicate packets decreases with A-MAC. There are two main causes for duplicate packets. First, an acknowledgement for a packet is not received even though the receiver has sent it. Second, a sender receives the same packet again that it previously sent. In that case, the sender drops the packet. The major reason for seeing more duplicate packets with ContikiMAC is that the time it takes to finish a packet exchange from the receiver's probe to the reception of the acknowledgement at the sender is much shorter for A-MAC that uses hardware acknowledgements compared to ContikiMAC that employs a CCA check before sending the acknowledgement. Since ContikiMAC needs more time to complete this data exchange, the receiver and sender have often moved out of range which is the major reason for the higher number of duplicate packets.

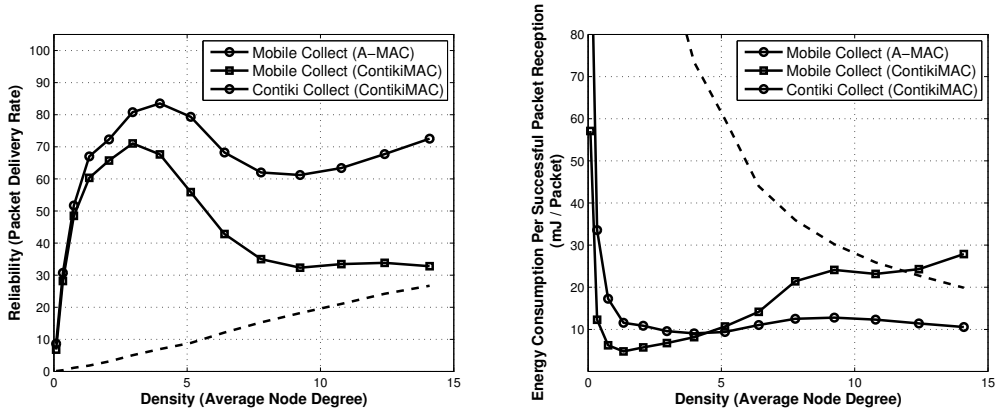
Another effect that comes into play is the number of lost routes. The number of lost routes decreases when the density increases for Mobile Collect (see Figure 8b). This explains that the negative trend in reliability between densities of 4 to 8 does not continue when the density increases. Note that even though A-MAC can much quicker connect to parents than ContikiMAC, the absolute number of lost routes is higher for A-MAC since the overall number of routes is much higher.

In Figure 7b we depict the energy consumption per packet for the different protocols. The base cost of probing, i.e.,



(a) Reliability (b) Energy per packet

Fig. 5: Reliability and energy performance in Mobile and Contiki Collect with Contiki-MAC in highway mobility model. Mobile Collect improves reliability and decreases energy consumption per received packet.



(a) Reliability (b) Energy per packet

Fig. 7: Reliability and energy performance of Mobile and Contiki Collect under the random waypoint mobility model. Mobile Collect on top of A-MAC performs very well.

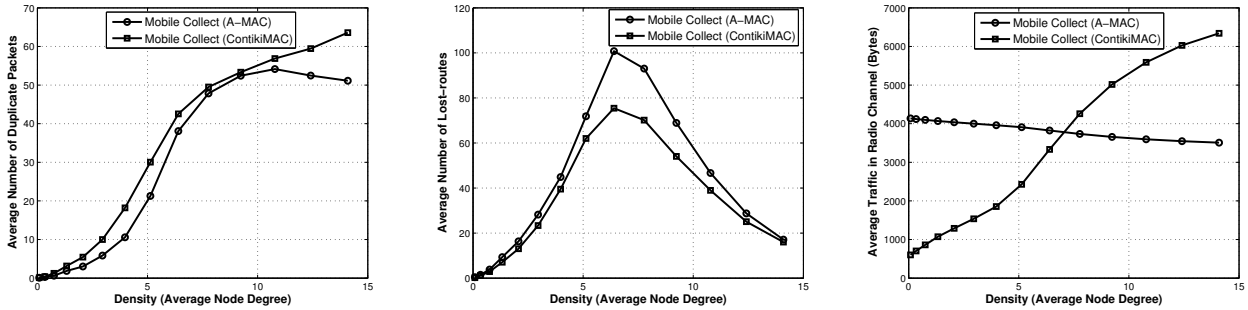
probing when there is no traffic, used in receiver-initiated MAC protocols such as A-MAC is known to be higher than the cost of short wake-ups performed by sender-initiated MAC protocols such as ContikiMAC [10]. This is also shown in Figure 8c that depicts the radio traffic in the air. The figure shows that the traffic in the air is much higher in very sparse networks which translates to the higher energy consumption in Figure 7b for node degrees up to three. At higher densities, A-MAC needs less beaconing which makes the traffic in the radio channel decrease. ContikiMAC on the other hand has problems setting up a tree which makes it send more beacons. The beacons are broadcasted which is expensive in sender-initiated protocols. Therefore, Mobile Collect on top of A-MAC is more energy-efficient than on ContikiMAC for higher densities.

In our scenarios with a lot of traffic, the energy consumption per packet decreases when the reliability increases as for a similar energy consumption more packets can be received by the sink. Therefore, the energy consumption per received packet is higher for Contiki Collect than for Mobile Collect. Note, however, that packet losses can also contribute to low energy consumption if packets are dropped at an early stage and do not need to be forwarded by other sensor nodes. Therefore, the energy consumption is not completely inversely proportional to the reliability.

E. Comparison against DYMO

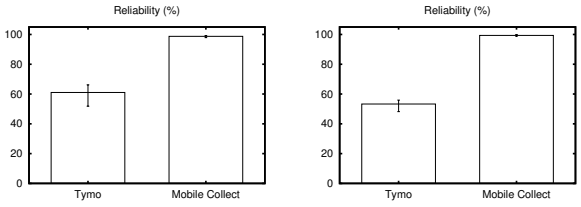
In this section we compare the performance of Mobile Collect against DYMO [24]. DYMO is a dynamic MANET routing protocol similar to AODV. We note that our comparison is not entirely fair since MANET protocols target slightly different application scenarios and require a two-way handshake before data can be delivered. DYMO, however, is one of the few protocols that are suitable for our scenario and that has an implementation for sensor nodes available.

We use TYMO, an implementation of DYMO in TinyOS, as base for this evaluation. TYMO also supports duty cycling at the MAC layer. Unfortunately, our initial experiments showed that even in static scenarios where nodes do not move the packet loss rate of TYMO with duty cycling is as low as 50%. Therefore, we disabled duty cycling for TYMO. In this experiment we simulate 50 nodes, one of them a sink node. All nodes move according our highway mobility model as presented above.



(a) Number of duplicate packets (b) Number of lost routes (c) Traffic on air

Fig. 8: Microbenchmarks that explain the performance of Mobile Collect



(a) Node speed between 2 and 8 m/s (b) Node speed 5 to 15 m/s

Fig. 9: Mobile Collect achieves higher packet delivery rates than TYMO

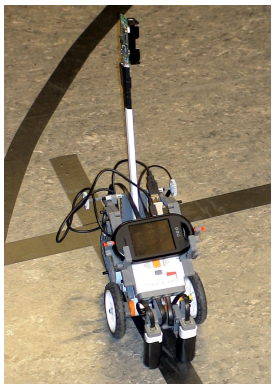


Fig. 10: Robots carrying sensor nodes are used for experiments in the testbed.

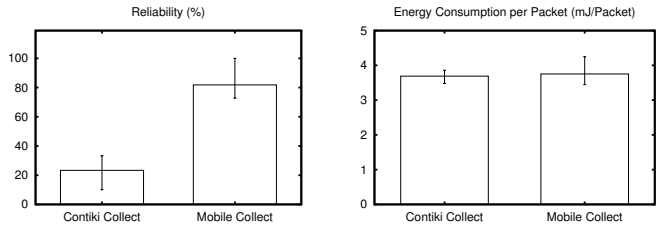
The results in Figure 9 show that Mobile Collect achieves a packet delivery rate close to 100% while TYMO delivers only around 60% of the packets even at the lower speed. At the higher speed, TYMO's delivery rate decreases to roughly 50%.

V. Experiments in Mobile Testbed

In order to demonstrate that our algorithms also work on real hardware we perform experiments in a testbed that includes mobile nodes.

A. Experimental Setup

We conduct experiments using Sensei-UU, a sensor network testbed that supports repeatable experiments involving mobile sensor nodes [25]. In Sensei-UU, stationary sensor nodes are complemented by mobile sensor nodes that are carried by robots (see Fig. 10). The robots use markers on the floor



(a) Packet Delivery Rate for Robot's Packets (b) Energy consumption per received packet

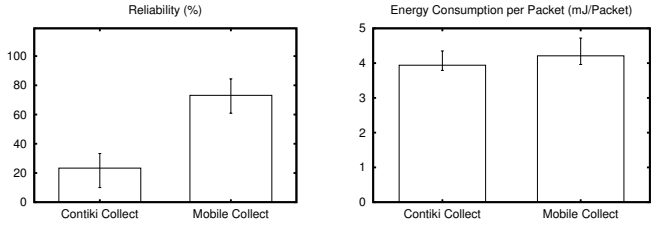
Fig. 11: Mobile Collect delivers more packet from the mobile robot at the same energy cost as Contiki Collect.

for navigation and localization. The robots' movements are controlled via a mobility script provided by the experimenter.

We set up the testbed with Tmote Sky sensor nodes in a corridor in the Ångström building of Uppsala university. Two clusters of four stationary sensor nodes are deployed at each end of the corridor. We set transmission power to -3 dBm, so that nodes from different clusters cannot communicate. Depending on the experiment scenario, one or two mobile sensor nodes travel along a straight track of 32 m length. At each end of the track the mobile sensor nodes are within communication range of the closest cluster only.

B. Experiments with one Mobile Robot

In the first experiment one robot moves from one cluster to the other cluster. This robot acts as a data source. The purpose of this experiment is to show that with Mobile Collect we can quickly adapt routes when moving out of and in range of the static part of the network. The results in Figure 11 show that almost all packets sent by the robot arrive at one of the sinks with Mobile Collect while with Contiki Collect the number of lost packets is quite high. The overall energy cost is similar as shown in Figure 11b. This figure shows the overall network energy consumption per received packet, i.e., the sum of all individual's nodes energy consumption divided by the overall number of received packets. Note that the energy consumption per received packet is similar to the energy consumption per received packet under the highway mobility shown in Figure 5b which shows that our simulation results are accurate.



(a) Packet delivery rate for robots' rackets

(b) Energy consumption per received packet

Fig. 12: Mobile Collect delivers more packet from the mobile robots at only slightly higher energy cost than Contiki Collect.

Our logs also show that while the robot is in the first cluster, all packets are received independent of the algorithm used. As the robot moves towards the second cluster, Mobile Collect is able to buffer packets and then quickly route packets to the sink in the second cluster while Contiki Connect keeps its old routes and increases the ETX rather than using new routes.

C. Experiments with two Mobile Robots

In the next experiment we use two mobile robots that move from the first to the second cluster. Both robots are data sources, not sinks. One robot moves two meters ahead of the other one. Figure 12 demonstrates the results. Also in this scenario, Mobile Collect is able to deliver much more data from the robots than Contiki Collect even though the energy consumption per received packet is slightly higher. As expected, when the leading robot comes close to second cluster it updates its routes and delivers packets to the new sink. Very soon also the second robot routes packets to the sink via the first robot. As the send buffer of the first robot is almost full this leads to a few packet drops and hence lower reliability for Mobile Collect as in the previous experiment.

VI. Conclusions

In this paper we have demonstrated that a few lightweight extensions to a standard collection protocol makes the protocol suitable also for scenarios where all nodes, both sink and sources, are mobile. We have evaluated our mechanisms both through simulation and experiments on real hardware and demonstrated high packet delivery rates at low energy consumption.

Acknowledgments

This work has been partly funded by SSF, VR, the SICS Center for Networked Systems (CNS), the Uppsala VINN Excellence Center for Wireless Sensor Networks WISENET, and partly supported the FP7 NoE CONET. CNS is funded by VINNOVA, SSF, KKS, ABB, Ericsson, Saab SDS, TeliaSonera, T2Data, Vendolocus and Peerialism. WISENET is funded by VINNOVA, Uppsala University, Banverket, CRL Sweden, FOI, Imego, JonDeTech, Pricer, SenseAir, SICS, TNT-Elektronik, TermoSense, VTT, Upwis, ÅAC Microtec, and WiseNet Holding. The authors thank Gunnar Karlsson (KTH) and Peter Stenlund (Vendolocus) for their input.

References

[1] M. Mudigonda, T. Kanipakam, A. Dutko, M. Bathula, N. Sridhar, S. Seetharaman, and J. Hallstrom. A mobility management framework for optimizing the trajectory of a mobile base-station. In *European Conference on Wireless Sensor Networks*, February 2011.

[2] T. Liu, C. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: Experiences with Impala and ZebraNet. In *ACM MobiSys*, June 2004.

[3] J.A.B. Link, K. Wehrle, O. Osechas, and J. Thiele. Ratpack: Communication in a sparse dynamic network. In *ACM SIGCOMM*, SEATTLE, WA, USA, August 2008.

[4] P. Tournoux, J. Leguay, F. Benbadis, J. Whitbeck, V. Conan, and M. Dias de Amorim. Density-aware routing in highly dynamic dtns: The rollernet case. *IEEE Transactions on Mobile Computing*, 10(12):1755–1768, December 2011.

[5] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wireless Networks*, 10(5):555–567, 2004.

[6] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, M. Durvy, JP Vasseur, A. Terzis, A. Dunkels, and D. Culler. Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In *ACM SenSys*, Seattle, USA, November 2011.

[7] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Berkeley, CA, USA, November 2009.

[8] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *EmNets*, Tampa, USA, November 2004.

[9] A. Dunkels. The contikimac radio duty cycling protocol. TR 13, SICS Technical Report, December 2011.

[10] P. Dutta, S. Dawson-Haggerty, Y. Chen, C. J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *ACM SenSys*, November 2010.

[11] M. Ho and K. Fall. Poster: Delay tolerant networking for sensor networks. In *Proc. of IEEE Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[12] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. *Distributed Computing in Sensor Systems*, pages 244–257, 2005.

[13] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *ACM SenSys*, November 2006.

[14] B. Pásztor, M. Musolesi, and C. Mascolo. Opportunistic mobile sensor data collection with scar. In *IEEE MASS*, October 2007.

[15] J. W. Lee, B. Kusy, T. Azim, B. Shihada, and P. Levis. Whirlpool routing for mobility. In *ACM MobiHoc*, 2010.

[16] M. Ali, T. Suleman, and Z.A. Uzmi. MMAC: A mobility-adaptive, collision-free mac protocol for wireless sensor networks. In *IPCC*, Phoenix, USA, April 2005.

[17] D. Zhang, Q. Li, X. Zhang, and X. Wang. De-ass: An adaptive mac algorithm based on mobility evaluation for wireless sensor networks. In *WiCOM*, September 2010.

[18] S.C. Choi, J.W. Lee, and Y. Kim. An adaptive mobility-supporting mac protocol for mobile sensor networks. In *Vehicular Technology Conference, VTC2008-Spring*, Marina Bay, Singapore, May 2008.

[19] C. L. Hedrick. RFC 1058: Routing information protocol, June 1988.

[20] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *SenseApp*, November 2006.

[21] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *SIMUTools*, March 2010.

[22] H. Arbabi and M.C. Weigle. Highway mobility and vehicular ad-hoc networks in ns-3. In *Proceedings of the 2010 Winter Simulation Conference (WSC)*, December 2010.

[23] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502, 2002.

[24] I. Chakeres and C. Perkins. Dynamic manet on-demand (dymo) routing. Ietf-draft 21, Internet Engineering Task Force, January 2011.

[25] O. Rensfelt and F. Hermans and P. Gunningberg and L. Larzon and E. Björnemo. Repeatable experiments with mobile nodes in a relocatable wsn testbed. *The Computer Journal*, 54(12), 2011.