Neighbor Cardinality Estimation with Low-Power Transceivers: Implementation and Experimental Results

Micha Rappaport*, Evşen Yanmaz*, and Christian Bettstetter*+
*Institute of Networked and Embedded Systems (NES), University of Klagenfurt, Austria
+Lakeside Labs GmbH, Klagenfurt, Austria
Email: micha.rappaport@aau.at

***Abstract*—Protocols for neighbor cardinality estimation can be used by a node in an ad hoc or sensor network to obtain a fast approximation of the number of nodes in its radio range. While the performance of such estimators was studied via simulations and mathematical analysis, we show here for the first time a proof-of-concept and performance test with an implementation on low-power wireless sensor devices.**

We illustrate the challenges of implementing the recently proposed Multi-Feedback Estimator (MFE) on Z1 devices. Experimental results show that – with certain computational improvements – MFE can perform faster than simple neighbor counting for scenarios with many nodes or relaxed accuracy requirements.

***Index Terms*—Neighbor estimation protocols, slotted random access, wireless sensor networks.**

I. Introduction

A lot of effort has been given to understanding the impact of node degree on network capacity, connectivity and protocol performance in wireless networks [1]–[4]. Furthermore, the number of neighbors of a node is constantly being used for scheduling and medium access, routing, and topology control among others. As networks grow larger and become more heterogeneous, efficient estimators to determine the number of neighbors (especially, number of neighbors with certain capabilities) are likely to be more necessary.

Commonly, protocols do not count the number (cardinality) of neighbors explicitly but gather this information by overhearing data packets [5]. However, such implicit approaches are not suitable in applications that utilize the demographics of networks of heterogeneous nodes with different capabilities. Similarly, for certain applications an exact number of neighbors might not be necessary, but an estimate of it with certain accuracy is sufficient [6]. Therefore, estimation algorithms have been proposed, especially for Radio Frequency Identification (RFID) systems for tag counting [7]–[12] and recently for large-scale wireless networks [13]. There is however still a lack of testing such estimation algorithms in practical systems. Most strategies require hardware with high computation power or high data rates to achieve the indicated performances in the theoretical analysis.

Our objective is to give a *proof of concept* that neighbor estimation algorithms can be implemented in cheap sensor devices and work in principle. We conduct an experimental analysis of a recently proposed contention-based neighbor estimation scheme: the Multi-Feedback Estimator (MFE) [13]. On one hand, this scheme has been compared with other schemes by simulations and has shown to perform well in wireless networks [13]. On the other hand, the algorithm is computationally intensive. We implement MFE on the sensor node platform Z1 by Zolertia and illustrate the challenges posed by implementation compared to simulations: How can we implement MFE in a time and energy efficient manner? What are the limitations of the hardware platform? The used platforms are sensor nodes with little processing power. Therefore, the estimation delay not only depends on the required communication between nodes, but also the processing time of the computations of the MFE algorithm. In this paper, we *assess by experiments* the performance of the MFE algorithm in terms of estimation time and accuracy. We illustrate the impact of the steps of the algorithm on the delay and provide improvements on the implementation to make the MFE algorithm feasible for low-power transceivers. Observe that MFE belongs to a class of adaptive neighbor estimation algorithms and is used as a representative scheme to illustrate the challenges that would be faced by several other estimation methods. We conduct experiments to test the performance of MFE for different network sizes from 20 nodes to 100 nodes. Furthermore the estimation time as well as the precision requirements are varied to demonstrate the usability of the scheme for different applications. For comparison, we also implement *a simple counting mechanism* to determine the number of neighbors. Thereby we show that as the network size grows, using an estimation algorithm is more suitable than explicitly exchanging messages to count the neighbors. Our findings show that many computational improvements are necessary to observe the full theoretical benefits on simple hardware.

This article is structured as follows. Section II explains how MFE works. Section III gives detailed information about the implementation. Section IV provides the setup for the experiments and analyzes the results. Finally, Section V concludes the article.

II. Multi-Feedback Estimator

This section describes the MFE algorithm [13]. Our motivation for choosing MFE is two-fold: it *performs better* than the other schemes analyzed in [13] and it is very *computationally intensive*. Therefore, we can better observe the relation between theoretical performance and practical limitations of neighbor estimation in low power transceivers.

The algorithm is contention-based, where each node responds to a query node over a number of slots s with a given probability p. The algorithm utilizes the number of empty slots observed by the query node at the end of each estimation round. The goal is to count the number of the query node's neighbors that satisfy certain criteria (e.g., battery level, common neighbor with another node). The desired accuracy of the estimation depends on the application of interest. Estimation can in principle be done in a non-adaptive or adaptive manner; i.e., the estimation parameters can be fixed at the beginning or can be tuned as the estimation continues. MFE belongs to the latter category, where the estimation consists of multiple rounds, at the beginning of which the number of slots and the access probability for the next round is broadcast. At the end of each round, the query node needs to give feedback to the neighboring nodes, based on the estimated accuracy. The estimation delay is heavily dependent on the time required to deliver the feedback messages (depends on the used hardware) and the number of required feedback rounds.

The estimation process consists of an initial phase and a main phase. In the initial phase the goal is to determine an initial access probability p such that at least one empty slot is observed. For completeness of this paper, we reproduce MFE in Algorithm 1 of [13] with simplifications where necessary.

The input parameters to the algorithm are as follows: duration of each round in the initial phase (c), feedback duration in terms of number slots (β), desired accuracy (θ), and confidence (α). The parameters e_i, p_i, and s_i are the number of observed empty slots, the access probability, and the number of slots in the i-th round, respectively.

The number of neighboring nodes is n and the maximum-likelihood estimate (MLE) of it at the end of the i-th round is denoted by $\hat{n}_i$ and it is obtained by solving the following for z [13]:

$$\sum_{j=1}^{i-1} e_j \ln(1-p_j) = \sum_{j=1}^{i-1} \frac{s_j - e_j}{1-(1-p_j)^z} \cdot (1-p_j)^z \cdot \ln(1-p_j) \ . \tag{1}$$

The variance is given by [13]:

$$\mathrm{Var}\left[\hat{n}\right] = \left(\sum_{j=1}^{i-1} \frac{1}{\frac{1}{s_j} \cdot \frac{1-(1-p_j)^{\hat{n}_j}}{(1-p_j)^{\hat{n}_j} \cdot \left(\ln(1-p_j)\right)^2}} \right)^{-1} \ . \tag{2}$$

The accuracy of the estimation is assessed by computing the variance of the estimate until round i and checking whether $\Psi_i = \sqrt{\mathrm{Var}[\hat{n}]} \cdot \Phi^{-1}\left(\frac{1+\alpha}{2}\right) \cdot \hat{n_i}^{-1} \leq \theta$, where $\Phi(\cdot)$ is the cumulative distribution function of normal distribution.

Algorithm 1 Multi-Feedback Estimator (MFE)

Input parameters: c, β, θ, α

1) Query node sends `Neighbor Query` containing c.
2) Initial phase:
   a) Every polled node transmits in slot i with probability 2^{-i}.
   b) After c slots, polled nodes listen for 1 slot:
      - If $\sum_i e_i = 0$: query node stays silent; all nodes continue in the initial phase.
      - If $\sum_i e_i \neq 0$: all nodes proceed with main phase.
3) Main phase:
   a) Query node determines $\hat{n}$ solving (1) and assesses accuracy using (2):
      - Accuracy is met: Query node stays silent to stop estimation.
      - Accuracy is not met: Query node determines $p_i = 1 - \exp\left(-\frac{1.594}{\hat{n}_i}\right)$ and s_i using (3) for next round and broadcasts these values.
   b) Each polled node transmits with probability p_i in each of the s_i slots.
   c) Proceed at 3).

The number of slots s_i to be used at the i-th round is computed using [13]

$$s_i = \begin{cases} 1, & \text{initial phase} \\ s_{w_i}, & \text{main phase } \& \ s_{w_i} \leq s_{b_i} + \beta + 1 \\ \max\left(s_{q_i}, \beta\right) & \text{main phase } \& \ s_{w_i} > s_{b_i} + \beta + 1 \end{cases} \tag{3}$$

with

$$s_{q_i} = \left\lceil \frac{\left(1 - q_i^{\hat{n}_i}\right) \cdot (1+\beta) \cdot \hat{n}_{w_i}^2}{q_i^{\hat{n}_i} \cdot \left(\hat{n}_{w_i}^2 - \hat{n}_i^2\right) + q_i^{\left|\hat{n}_i - \hat{n}_{w_i}\right|} \cdot \hat{n}_i^2 - \hat{n}_{w_i}^2} \right\rceil \ , \tag{4}$$

where $q_i = 1 - p_i$; s_{b_i} is the minimum number of slots necessary such that the estimation accuracy is met in the next round; $\hat{n}_{w_i} \in \{\hat{n}_i \cdot (1-\Psi_i), \hat{n}_i \cdot (1+\Psi_i)\}$ is the estimation of n that would require more slots to finish the estimation process after the current round; s_{w_i} is the corresponding number of required slots to n_{w_i}. The minimum s_{b_i} is computed using the maximum variance of the i-th round, which is given by:

$$\mathrm{Var}_i\left[\hat{n}\right] = \frac{\frac{1}{s_i} \cdot \frac{1-(1-p_i)^{\hat{n}_i}}{(1-p_i)^{\hat{n}_i} \cdot \left(\ln(1-p_i)\right)^2} \cdot \left(\frac{\hat{n}_i \cdot \theta}{\Phi^{-1}\left(\frac{1+\alpha}{2}\right)}\right)^2}{\frac{1}{s_i} \cdot \frac{1-(1-p_i)^{\hat{n}_i}}{(1-p_i)^{\hat{n}_i} \cdot \left(\ln(1-p_i)\right)^2} - \left(\frac{\hat{n}_i \cdot \theta}{\Phi^{-1}\left(\frac{1+\alpha}{2}\right)}\right)^2} \ , \tag{5}$$

Solving $\Psi_i \leq \theta$ with variance given in (5) for s yields s_{b_i}.

Observe that an adaptive algorithm as MFE that requires several messages from the query node naturally implies that the speed of estimation will depend on the hardware used in the implementation. Even if the feedback transmissions are short in time, i.e., the time spent in communication between nodes is small, the mathematical computations to run the estimation algorithm can be a major limitation depending on the hardware. In the next section, we elaborate on the implementation of MFE on low-power transceivers and discuss

the advantages and disadvantages as well as the optimizations needed to improve the performance of the algorithm.

III. Implementation of Multi-Feedback Estimator on Z1 Nodes

The hardware platform used to carry out the experiments is the sensor node Z1 by Zolertia. This sensor node is specifically designed to run on very low power. The MCU is the low-power and low-cost MSP 430 by Texas Instruments. The transceiver is the Texas Instruments CC2420, which runs IEEE 802.15.4 in the 2.4 GHz ISM band. Therefore the sensor nodes are compatible with ZigBee and 6LoWPAN specifications. 802.15.4 specifies the lower two layers, the physical layer and the medium access control layer. The upper layers are specified by the operating system TinyOS, which is an open source software based on nesC. nesC is an extension to the C programming language with a focus on the limited memory on sensor nodes.

Using the low-power off-the-shelf nodes restricts the implementation in many ways. First, the computational power of the MCU is very low. Also, there is no floating point unit to perform complex operations. The calculations mentioned in the previous section need to be optimized during the implementation process. In the following, we show how the estimation performs in sensor networks where the neighbor cardinality is of importance but the power is usually very limited. Clearly, optimizing the calculations is crucial, but it is also a platform-dependent process. Note that on more powerful platforms there is still a lot of room for improvement of the performance.

The experiments utilize homogeneous sensors in terms of hardware and software. Every node can be used to collect information about its neighbors. This node will be referred to as *query* node and its neighbors as *polled* nodes.

The following steps need to be considered for the implementation of neighbor estimation:

A) Communication between the nodes
B) Synchronization between the nodes
C) Calculation of the estimation

A. Communication between the nodes

There are two types of communication between the nodes. Firstly, the estimation parameters need to be broadcast from the query node to the polled nodes (steps 1 and 3a in Algorithm 1). Secondly, in the contention phase, the polled nodes need to respond to the request. The first type of communication needs transmission of data packets. For the response from the polled nodes, it is only necessary that the nodes indicate activity on the channel and they do not need to convey any further information. To this end, we utilize `BUSY` tones with very short duration to minimize the delay for contention.

To start the estimation process, the query node broadcasts the estimation parameters within the `neighbor query message (NQM)`. At the beginning of each round in the main phase, the query node uses the `update message (UM)` to broadcast the updated estimation parameters. The

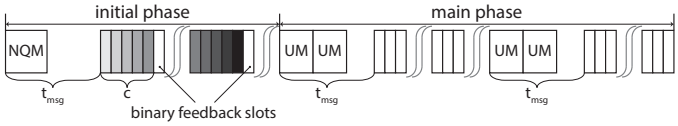
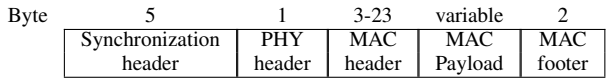
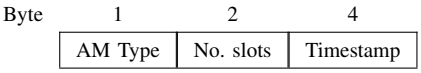


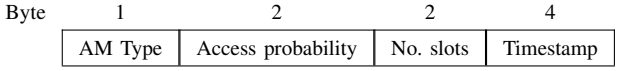
Fig. 1. Slot structure of the estimation process.



(a) Format of a 802.15.4 MAC frame.



(b) MAC Payload of a `NQM`.



(c) MAC Payload of an `UM`.

Fig. 2. Frame formats.

frame structure of these messages can be seen in Fig. 2. The slot structure is depicted in Fig. 1. Between each round of the initial phase and before each `UM` there is an undefined duration that the query node needs to perform calculations.

Fig. 2a shows the standard MAC frame for the IEEE 802.15.4 protocol. Fig. 2b and 2c show the MAC payload for the `NQM` and `UM` respectively. Each MAC protocol data unit (MPDU) carries one byte, the AM Type, for message identification in TinyOS. Furthermore, they carry the number of slots for the next contention phase as well as the access probability for each slot in the contention phase (`UM` only). In the initial phase, the access probability is always set to 2^{-i} where i is the current slot count (see step 2 in Algorithm 1). Finally they also carry the timestamp at which the next contention phase will begin.

The `BUSY` tones are transmitted by each node with a certain probability in the timeslots after receiving an `NQM` or a `UM` from the query node (see steps 2a and 3b in Algorithm 1 and Fig. 1). Transmitting a `BUSY` tone on the CC2420 transceiver is possible in two ways: Either by sending out one or multiple consecutive data packets or by transmitting a continuous signal. TinyOS is build with a packet-based radio stack that allows only transmission of data packets. To use data packets as `BUSY` tones, it is necessary to disable the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) in the MAC layer to allow collisions on the medium. The downside of this method is a delay and more strict synchronization requirements. The transceiver CC2420 though offers the possibility of switching into a test mode. Thereby it is possible to transmit a continuous signal as `BUSY` tone for arbitrary durations. This signal can be switched on and off quickly allowing short time slots. On one hand, this means that the time needed for contention decreases. On the other hand, the nodes must be switched between the regular packet

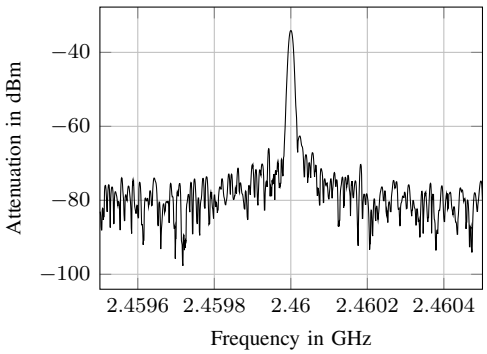


Fig. 3. Spectrum of a BUSY tone.

mode and the test mode. This destabilizes the nodes since the test mode is only for development purposes and not for productive use [14]. For example the polled nodes sometimes drop the first packet that they receive after switching from the test mode to the packet mode.

For these experiments, we chose to run the nodes in the test mode. This leads to a better performance than using data packets as BUSY tones. To avoid packet loss due to switching between test and regular packet mode, the feedback message UM is transmitted twice. We created a new component in the PHY layer of the TinyOS radio stack. This component is then used to emit or detect a continuous unmodulated carrier that represents a BUSY tone. Hence, MFE cannot be implemented in a single layer of the radio stack but rather on multiple layers. An example for the spectrum of this BUSY tone can be seen in Fig. 3.

B. Synchronization between the nodes

To synchronize the nodes, the most intuitive way is to use packet-level time synchronization. The query node needs to broadcast an NQM at the beginning of the estimation. This message can be used to spread synchronization information amongst the nodes. TinyOS already provides a tool for that through the CC2420TimeSyncMessageC component. With this component, it is possible that the transmitter includes a timestamp into a message which will then be transformed to the local clock of the receiver. Hence, the NQM and the UM include the timestamp at which the next contention phase will start. Thereby the time slots of the contention phases are aligned very precisely varying only by a few micro seconds.

C. Calculation of the estimation

The calculations consist of multiple steps:

- Estimate the number of nodes by solving (1)
- Calculate the accuracy Ψ_i
- Calculate the access parameters p_i and s_i for the next contention phase

These calculations are not trivial to implement on the Z1 nodes. Most calculations require floating point operations, but the Z1 nodes do not provide a floating point unit. Therefore, all floating point calculations must be computed in software without any hardware acceleration and hence are relatively slow. Another possibility would be to store precomputed values in a lookup table. Due to the complexity of the calculations, more precisely the number of input parameters, this is impossible. Storing all these values would lead to such large lookup tables that could not be stored on the nodes memory.

Hence several optimizations regarding the implementation of the calculations are performed. First of all, constant values like $\sqrt{2}$ or $\Phi(\alpha)$ have been precomputed. All exponentiations are using integer exponents, so they have been replaced by repetitive multiplications instead. Fixed point operations are used instead of floating point operations whenever possible. The estimation itself (see (1)) cannot be solved analytically and the nodes do not provide the required power to solve it numerically. Hence the two sums need to be calculated each round for the current estimate z. If the two sides of the estimation do not match, z is increased or decreased accordingly to find the correct estimate. To speed up this calculation, the left hand side is accumulated in each round since it is independent of z. For the initial rounds the estimation is simplified. It is assumed that with each busy slot the estimate doubles since the access probability bisects. Finally, many calculations use the natural logarithm of the same probability value repeatedly. These logarithms are stored once they have been computed for the first time.

The parameters, setup, and performance of this implementation is discussed in the next section.

IV. Performance Analysis

The experimental setup is as follows: All nodes are placed such that they are within communication range of each other and all of them are stationary (see Fig. 4). The query node is connected to a PC with a serial connection to collect the results of the experiments. The nodes are using a 5 MHz wide channel in the 2.4 GHz band. The experiments are carried out in an office environment with interference from other technologies like WLAN or Bluetooth. The number of nodes in the network is varied from 20 to 100. The message delay is fixed to $t_{\text{msg}} = 30$ ms and the timeslot length is 1 ms. The input parameters as described in Section II are: $c = 5$, $\beta = 30$, $\theta = \{5\%, 10\%, 25\%, 50\%\}$ and $\alpha = 95\%$. For each experiment setup the results have been averaged over 100 runs.

First, we investigate the accuracy of our implementation. Fig. 5 shows the number of slots the estimation takes given different network sizes and estimation accuracies θ. The values are obtained from both experiments and numerical analysis using the theory described in Section II. These results are independent of the hardware used. Depending on the used platform, the slot duration in seconds can be substituted to determine the actual estimation delay. The estimations in our experiments take on average longer than in theory. The 5% quantile is in many data points close to the average value of the theory. The experimental results exhibit a high variance because of both the wireless channel and the limitations due to the switching between test and regular mode. It is likely that some neighbors do not receive all NQM or UM and hence do not participate in the overall or part of the estimation.

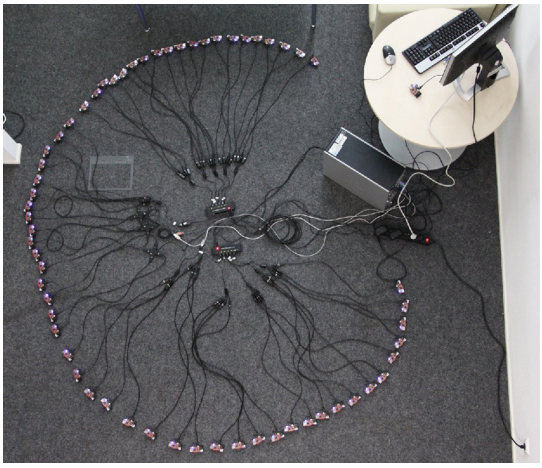


Fig. 4. Experimental setup.

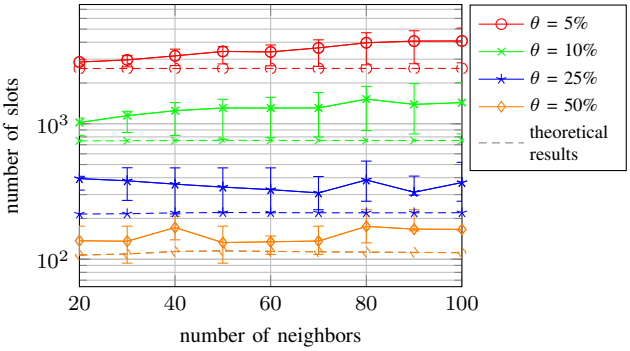


Fig. 5. Average estimation delay in slots for different accuracies. Theory with average values, experimental results with average and 5% / 95% quantiles.

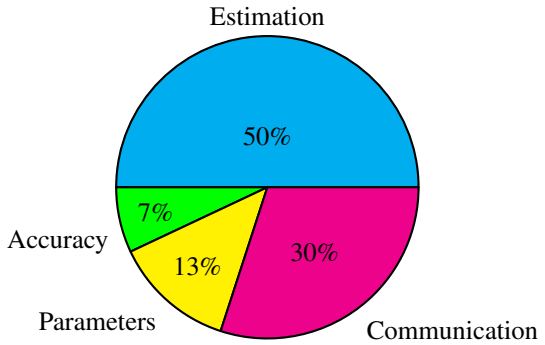


Fig. 6. Distribution of the average times needed for each step of the estimation process.
Estimation: Time needed to calculate the estimated number of neighbors.
Accuracy: Time needed to calculate the estimated accuracy.
Parameters: Time needed to calculate the parameters for the next round.
Communication: Time needed for communication among nodes.

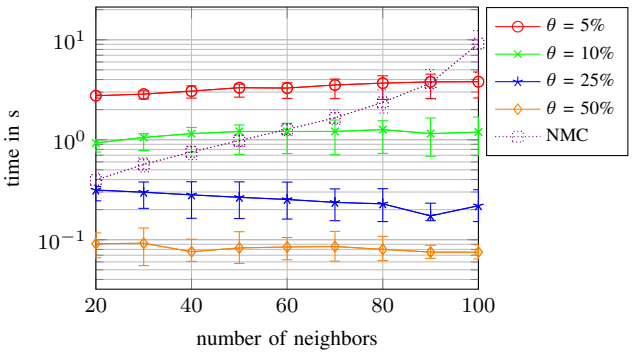


Fig. 7. Average estimation delay considering only communication for different accuracies with 5% / 95% quantiles.

When carrying out the estimation, not only the time that is needed to communicate with the neighbors, but also the time that is required for the calculations needs to be considered. As mentioned earlier, most calculations should take place in the floating point domain. But since the Z1 sensor nodes do not have a floating point unit, the calculations are carried out in software rather than in hardware which takes considerably longer. Fig. 6 shows the distribution of the time that is needed for the estimation process averaged over all experiments.

Observe that the actual interaction between the nodes takes only a fraction of the time, but the majority is spent doing the calculations. The fraction of the communication part increases for increasing error tolerances, whereas it decreases for stricter accuracy requirements. Similarly, when the network size increases, the communication part becomes less significant as it is for small networks. This shows that significant improvements can be achieved when sensor nodes with higher computational power or with a floating point unit are used.

Next, our goal is to check whether estimation in wireless sensor networks is suitable or whether explicit counting is a better approach to find the number of neighbors of a node. To this end, we compare the performance of MFE with a simple non-colliding message counting (NMC) scheme. In NMC, the query node broadcasts a `query` and all neighbors reply with a `hello` which is again acknowledged by the query node. If a neighbor does not receive an `acknowledgement (ACK)`, it keeps sending its `hello` until it receives an `ACK`. Fig. 7 shows the time that is needed for the estimation given different network sizes and estimation accuracies where the delay due to calculations is neglected. The run time of NMC increases exponentially with the number of nodes. For small networks, NMC performs better, but for larger networks it is outperformed by MFE. In small networks up to 20 nodes, MFE can only compete with NMC for rough estimations, where an error above 25% is permitted. The estimation delay of MFE increases only slowly with with the number of nodes making it especially useful for large networks.

Fig. 8 shows the total delay for the estimation process for different accuracies including delays due to communication and calculations. The total estimation delay of MFE is higher than that of NMC for $\theta = 5\%$. But the delay increases slower than that of NMC, hence for larger number of nodes, the use of an estimation algorithm is still expected to perform better. If the desired accuracy is not as high (10% or 25%), then MFE outperforms NMC even with the high computation delay for networks containing 100 or more nodes. For even more relaxed

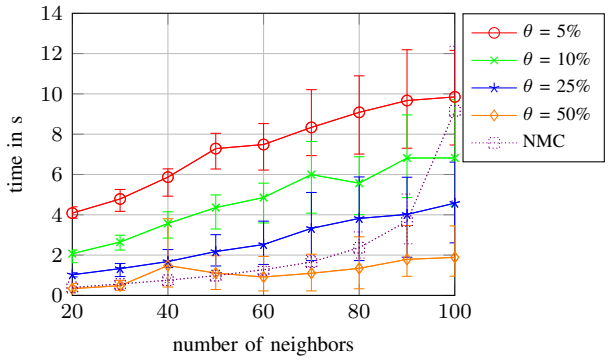


Fig. 8. Average estimation delay including calculations for different accuracies with 5% / 95% quantiles.

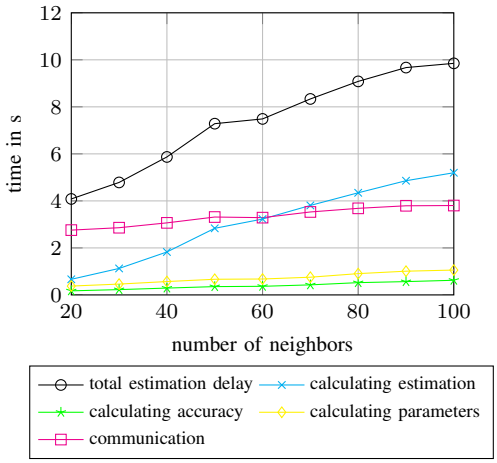


Fig. 9. Average estimation delay for each estimation step, θ = 5%.

accuracy requirements above 50% MFE always outperforms NMC. Fig. 9 shows the breakdown of the delay components for $\theta = 5\%$. Observe that depending on the network size the delay due to communication increases slowly, whereas delay due to the estimation calculation increases significantly.

V. Conclusions

In this paper, we investigated the feasibility of state-of-the art neighbor estimation protocols on practical systems. Specifically, we focused on low-power communication devices and illustrated the challenges that need to be addressed for efficient implementation of the estimation algorithms. We showed that computational effort dominates the estimation delay compared to delay due to interactions between nodes for certain scenarios. We provided improvements on the implementation such that limited power and memory available on sensor nodes are taken into account. Our results show that especially for larger network sizes, even though computationally intensive, neighbor estimation algorithms are more suitable than explicit counting.

Acknowledgments

This work was partly funded by ERDF, KWF and BABEG under grant 20214/15935/23108 within the Lakeside Labs project RELAY. The authors would like to thank Helmut Adam and Wasif Masood for valuable discussions.

References

[1] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, pp. 388–404, Mar. 2000.

[2] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Comput. Surv.*, vol. 37, pp. 164–194, Jun. 2005.

[3] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," *IEEE/ACM Trans. Netw.*, vol. 14, pp. 479–491, Jun. 2006.

[4] Y. Sasson, D. Cavin, and A. Schiper, "Probabilistic broadcast for flooding in wireless mobile ad hoc networks," in *Proc. IEEE Wireless Commun. and Net. (WCNC)*, New Orleans, US-LA, Mar. 2003, pp. 1124–1130.

[5] A. Cerpa and D. Estrin, "ASCENT: Adaptive self-configuring sensor networks topologies," *IEEE Trans. Mobile Computing*, vol. 3, pp. 272–285, Jul.-Aug. 2004.

[6] H. Adam, E. Yanmaz, and C. Bettstetter, "Medium access with adaptive relay selection in cooperative wireless networks," *IEEE Trans. Mobile Computing*, vol. PP, no. 99, 2013.

[7] H. Han, B. Sheng, C. C. Tan, Q. Li, W. Mao, and S. Lu, "Counting RFID tags efficiently and anonymously," in *Proc. IEEE Intern. Conf. Comput. Commun. (INFOCOM)*, San Diego, US-CA, Mar. 2010.

[8] C. Qian, H. Ngan, Y. Liu, and L. Ni, "Cardinality estimation for large-scale RFID systems," *IEEE Trans. Parallel and Distributed Syst.*, vol. 22, pp. 1441–1454, Sep. 2011.

[9] M. Kodialam and T. Nandgopal, "Fast and reliable estimation schemes in RFID systems," in *Proc. ACM Intern. Conf. on Mobile Computing and Networking (MobiCom)*, Los Angeles, US-CA, Sep. 2006.

[10] M. Kodialam, T. Nandagopal, and W. C. Lau, "Anonymous tracking using RFID tags," in *Proc. IEEE Intern. Conf. Comput. Commun. (INFOCOM)*, Anchorage, US-AK, May 2007, pp. 1217–1225.

[11] H. Adam, E. Yanmaz, C. Bettstetter, and M. Kodialam, "Erratum to anonymous tracking using RFID tags [INFOCOM 2007]," Nov. 2011.

[12] B. Sheng, Q. Li, and W. Mao, "Efficient continuous scanning in RFID systems," in *Proc. IEEE Intern. Conf. Comput. Commun. (INFOCOM)*, San Diego, US-CA, Mar. 2010.

[13] H. Adam, E. Yanmaz, and C. Bettstetter, "Contention-based estimation of neighbor cardinality," *IEEE Trans. Mobile Computing*, vol. 12, pp. 542–555, Mar. 2013.

[14] *2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. C)*, Texas Instruments Inc., Jul. 2013.