The definite version is published in Proc. IEEE Vehicular Technology Conference, Taipei, Taiwan, May 2010.

Contention-Based Neighborhood Estimation

Helmut Adam[1], Evşen Yanmaz[1], Wilfried Elmenreich[1], and Christian Bettstetter[1,2]

[1] University of Klagenfurt, Institute of Networked and Embedded Systems, Mobile Systems Group, Austria

[2] Lakeside Labs GmbH, Austria

***Abstract*—This paper proposes a probabilistic technique that enables a node to estimate the number of its neighbors that fulfill certain criteria. The technique does not require any *a priori* information about the network topology. Based on a query from the estimating node, the neighbors send busy tones in a sequence of time slots. Evaluating the fraction of empty slots, the querying node can infer about its neighborhood. Mathematical analysis and numerical simulations show how the estimation success and accuracy depend on the available time period. Applications of the method can be found, for example, in RFID systems and medium access protocols.**

***Index Terms*—Neighborhood estimation, slotted random access, cooperative networks.**

I. Introduction

Several algorithms and protocols in communication systems assume that a node knows how many adjacent nodes it has. Such neighborhood knowledge can be exploited to optimize link-layer and networking functions, such as medium access, leader election, and cooperative relaying (see, e.g., [1]–[3]). It is actually beneficial in all problems where the node degree is of interest. As a simple example, knowledge about the number of competing nodes in Slotted-ALOHA [4] can be exploited to derive an optimum channel access probability. From a more general perspective, we might not be interested in the *total* number of neighbors but in the number of neighbors with a particular *attribute* (e.g., minimum battery level, storage capability). In all cases, however, only the pure *number* of nodes not their *identities* is needed.

The problem of estimating the number of neighbors has not been addressed sufficiently in the literature. Mostly, the number of neighbors is gathered when addressing other problems, where also the identity of nodes is of interest. Commonly, a coarse neighborhood estimation is performed by overhearing data transmissions, where all neighbors transmit periodically, e.g., some dedicated `Hello` messages. However, such an approach is not well suited for more specialized problems, such as getting the number of common neighbors of a set of nodes in a cooperative network, identifying backhaul nodes for wireless sensor nodes whose energy levels are above a certain threshold, or for time dynamic networks with expeditiously changing neighborhoods. Furthermore, in dense networks, a high network load would result if all nodes exchange data messages. Collisions between message transmissions would trigger several retransmission attempts, which worsens the situation and makes it impossible to derive a tight bound for the maximum time until the estimation can be performed.

The contribution of this work is a novel method for neighborhood estimation. Based on probabilistic trials, it can quickly estimate the actual number of nodes without any *a priori* knowledge of the network. The accuracy is a system design parameter. The algorithm enables us to make an estimation after each trial, which means that it can be also used as an *anytime algorithm* [5], i.e., an algorithm that can be stopped at an arbitrary time instant and still provides meaningful results.

The paper is structured as follows. Section II proposes the neighborhood estimation technique. Section III evaluates this technique. Section IV summarizes related work and highlights the differences to our approach. Finally, Section V concludes and gives an outlook to further research.

II. Neighborhood Estimator

A node is interested in the number of its neighbors, either the total number or the number of neighbors that fulfill certain attributes, such as energy level, connectivity, or common neighbors. Our approach to solve this task is as follows. The querying node broadcasts a `Query` message which optionally contains the requested attributes. Subsequently, a contention frame consisting of s time slots starts, similar as in contention-based medium access protocols. Each node receiving the `Query` message and fulfilling the attributes transmits a busy tone in each of the slots with a given probability p (Bernoulli process). These tones just indicate activity on the channel and do not convey any other information. The querying node observes the channel occupation during the contention period and infers from its observation the number of nodes.

A. Basic Design Issues

The node can in principle count the number of

- empty slots (no node transmits in this slot),
- non-colliding slots (one node transmits in this slot), and
- colliding slots (multiple nodes transmit in this slot).

If n is the actual number of nodes, the probability that there is an empty slot is $P_0 = (1-p)^n$. The probability that there is a non-colliding slot is $P_1 = n\,p\,(1-p)^{n-1}$. Finally, the probability that a collision occurs in a given slot is $P_x = 1 - P_1 - P_0$. By counting the number of empty, non-colliding, and colliding slots during a sufficiently large number of slots, a node can estimate these three probabilities. Based on such a probability, it can estimate n using the inverse functions of the above expressions.

The following questions arise: Which of these slot types are well-suited to estimate the number of nodes? Should the node count all three slot types? Is one slot type more beneficial in practice? To answer these questions, let us further analyze these probabilities. Figure 1 illustrates them for a given access probability p as a function of n.

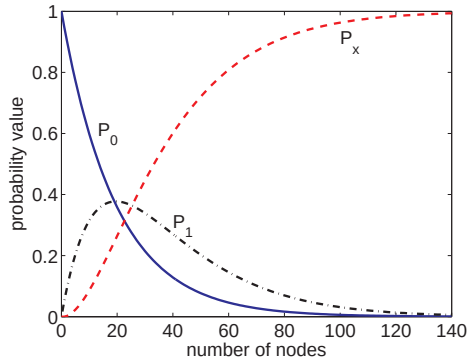

Fig. 1. Slot probabilities for $p = 0.05$ versus n

We observe that P_1 first increases with increasing n, has its maximum, and then decreases again. For given n, there are in general two different values of P_1, meaning that the function is not injective. Thus, counting only the number of non-colliding slots is impractical for estimating n.

In contrast, the probabilities P_0 and P_x are monotonic with respect to n and can thus be inverted to get an unambiguous value of n. Hence, in theory, both slot types enable us to estimate n. In practice, however, in order to detect collisions, the transmission of special codewords might be necessary. This would in turn require a longer transmission time than simple busy tones. Furthermore, due to capturing effects, collisions could be interpreted as non-colliding slots. These facts motivate us to focus exclusively on the number of *empty slots* for estimating the number of neighbors.

B. Neighborhood Estimator Based on Empty Slots

The number of nodes n can be calculated from the empty slot probability P_0 and the slot access probability p using

$$n = \frac{\ln P_0}{\ln (1-p)} \,. \tag{1}$$

To estimate n, it is thus sufficient to estimate P_0 and apply it in (1). To do so, we count the number of empty slots e in a frame with s slots. The relative frequency is

$$\hat{P}_0 = \frac{e}{s} \,. \tag{2}$$

Applying the relative frequency definition of probability,

$$\lim_{s\to\infty} \hat{P}_0 = P_0 \,. \tag{3}$$

Thus, $\hat{P}_0$ is a good estimate for P_0 if s is sufficiently large.

C. Success of the Neighborhood Estimator

After an observation period of s slots, we can obtain one of the following three cases:

1) *No empty slots ($e = 0$).* If no empty slots occurred, an estimate for n cannot be made using (1), because $\ln \hat{P}_0$ with $\hat{P}_0 = 0$. The probability for this event is

$$\mathrm{P}[e = 0] = (1 - P_0)^s = (1 - (1-p)^n)^s \,. \tag{4}$$

The estimator can return a lower bound for n, given by

$$\hat{n}_l = \frac{\ln\left(\frac{1}{s}\right)}{\ln (1-p)} \,, \tag{5}$$

which is the largest $\hat{n}$ that can be estimated with the given number of slots s. To avoid this case, the probability p must be decreased and/or the number of observed slots s should be increased.

2) *All empty slots ($e = s$).* Also if each slot is empty, a good estimate for n cannot be given. The probability for this event is

$$\mathrm{P}[e = s] = P_0^s = (1-p)^{ns} \,. \tag{6}$$

For this case, the estimator can return an upper bound

$$\hat{n}_u = \frac{\ln\left(\frac{s-1}{s}\right)}{\ln (1-p)} \,, \tag{7}$$

which is the smallest $\hat{n}$ that can be estimated with the given s. To avoid this case, the probability p and/or the number of observed slots s should be increased. Note that $e = s$, also if $n = 0$. A way to detect this case is to use a dedicated test slot with access probability $p = 1$.

3) *Some empty slots ($0 < e < s$).* If some of the slots are empty and some are non-empty, the estimator will return

$$\hat{n} = \frac{\ln\left(\frac{e}{s}\right)}{\ln (1-p)} \,. \tag{8}$$

as a successful estimate for the number of nodes n.

As can be seen, the parameters p and s determine the success of the estimator. Clearly, we would like to maximize the likelihood of the third case. We demand an estimation success probability of

$$\mathrm{P}\left[0 < e < s\right] \geq \Omega \,, \tag{9}$$

where the threshold Ω is close to $100\,\%$. This can be achieved if p and s are chosen in a way that

$$\mathrm{P}[e = 0] \leq \frac{1+\Omega}{2} \quad \text{and} \quad \mathrm{P}[e = s] \leq \frac{1+\Omega}{2} \,. \tag{10}$$

D. Accuracy of the Neighborhood Estimator

The relative error $\epsilon = \left|\frac{\hat{n}-n}{n}\right|$ made by the estimator should be below a desired threshold Θ with confidence α, i.e.,

$$\mathrm{P}\left[\epsilon \leq \Theta\right] \geq \alpha \,, \tag{11}$$

where Θ and α are input parameters to the estimator.

For given p and s, the estimator will in general produce different values of $\hat{n}$ for the same n. It is of interest to determine the distribution of $\hat{n}$. The number of empty slots e is binomially distributed with parameters P_0 and s. If s is large and P_0 is neither close to 0 nor to 1, we can approximate the binomial distribution by a normal distribution with mean $\mu = s\,P_0$ and variance $\sigma^2 = s\,P_0\,(1 - P_0)$. As a rule of thumb, this approximation is accurate if the products $s\,P_0$ and $s\,(1 - P_0)$ are larger than 5 [6]. Using the normal distribution and the observation that (8) is a monotonically decreasing function of e, and thus differentiable, we can follow the steps

of [7] (Example 6a.2.1) to show that $\hat{n}$ follows a normal distribution with mean $\mathrm{E}[\hat{n}] = n$ and variance

$$\mathrm{Var}[\hat{n}] = \frac{1}{s} \cdot \frac{1-(1-p)^n}{(1-p)^n \cdot \left(\ln(1-p)\right)^2} \,. \tag{12}$$

For the normal distribution the left hand side of (11) becomes

$$\mathrm{P}\left[(1-\Theta)n \le \hat{n} \le (1+\Theta)n\right] = 2 \cdot \Phi\left(\frac{\Theta\, n}{\sqrt{\mathrm{Var}\left[\hat{n}\right]}}\right) - 1 \,, \tag{13}$$

with $\Phi(\cdot)$ being the cumulative distribution function (cdf) of the standard normal distribution. If we use (13) in (11) and rearrange for Θ we obtain

$$\Psi(s) := \frac{\Phi^{-1}\left(\frac{1+\alpha}{2}\right) \cdot \sqrt{\mathrm{Var}\left[\hat{n}\right]}}{n} \le \Theta \,. \tag{14}$$

For given p, α, and n, we get a bound $\Psi(s)$ on the relative error ϵ, which is a function of s and guarantees $\mathrm{P}\left[\epsilon \le \Psi(s)\right] = \alpha$. Thus, to satisfy $\Psi(s) \le \Theta$, the number of slots s needs to be sufficiently high.

E. Choosing a Suitable Slot Access Probability

Following the discussion above, for given p and s, the estimator will produce meaningful results only in a range of operation $[n_{\min}, n_{\max}]$, where the likelihood of *no empty slots* and *all empty slots* is low. If the actual n is higher or lower, an estimation may fail. Furthermore, there is a tradeoff between the estimation success and quality on the one hand and the required number of slots, i.e., the estimation delay, on the other hand.

The goal is to determine a suitable value for the access probability p that

- works in a given estimation range $[n_{\min}, n_{\max}]$,
- achieves an estimation success probability of Ω, see (9),
- achieves an error threshold Θ with confidence α, see (11),
- minimizes the number of required slots s.

We choose an access probability that minimizes the difference between the bounds $\Psi(s)$ for $n = n_{\min}$ and $n = n_{\max}$:

$$p_r = \underset{p}{\mathrm{argmin}} \left| \frac{\mathrm{Var}[\hat{n}]_{n=n_{\min}}}{n_{\min}^2} - \frac{\mathrm{Var}[\hat{n}]_{n=n_{\max}}}{n_{\max}^2} \right| . \tag{15}$$

From this, we obtain the required number of slots that guarantees all above conditions:

$$\begin{aligned} s = \max\Bigg(& \frac{\left(\Phi^{-1}\left(\frac{1+\alpha}{2}\right)\right)^2 \cdot \left(1-(1-p_r)^{n_{\min}}\right)}{n_{\min}^2 \cdot \Theta^2 \cdot (1-p_r)^{n_{\min}} \cdot \ln(1-p_r)^2}, \\ & \frac{5}{(1-p_r)^{n_{\max}}}, \frac{5}{1-(1-p_r)^{n_{\min}}}, \\ & \frac{\ln\left(1-\frac{1+\Omega}{2}\right)}{\ln\left((1-p_r)^{n_{\min}}\right)}, \frac{\ln\left(1-\frac{1+\Omega}{2}\right)}{\ln\left(1-(1-p_r)^{n_{\max}}\right)} \Bigg) . \end{aligned} \tag{16}$$

Let us explain this expression. The first term represents the necessary number of slots to achieve the requested estimation accuracy for $n_{\min}$. It is obtained by setting (12) into (14) and solving for s. The s-value for $n_{\max}$ is the same due to the way we have chosen p_r. The second and third term of (16) ensure the validity of approximating the distribution of e with a normal distribution. Finally, the last two terms verify that the estimation success probability is at least Ω.

III. Performance Evaluation

Let us now evaluate the proposed neighborhood estimator. If not stated otherwise, we request an error threshold $\Theta = 10\,\%$ with confidence $\alpha = 95\,\%$. Furthermore, we require an estimation success probability of $\Omega = 99\,\%$. A number of nodes n is chosen; each of these nodes accesses a slot with probability p for a duration of s slots; finally, the estimation $\hat{n}$ is made. This random experiment is repeated $5\,000$ times.

A. Evolution of the Estimation Quality over Time

In the first experiment, the estimator assumes a range of operation from $n_{\min} = 100$ to $n_{\max} = 200$. The actual number of nodes is $n = 150$. Figure 2 illustrates the sample average of the estimated number of nodes $\hat{n}$ and the $95\,\%$ confidence interval as a function of s. The dashed line represents the target error threshold Θ. The averaged $\hat{n}$ reaches the real value n quite fast. However, the 95% confidence interval gets narrower only gradually over time. The accuracy is within the requested margin after $s = 637$ slots.

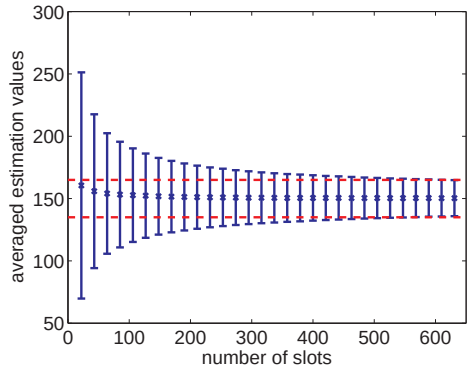

Fig. 2. Sample average of the estimated number of nodes with 95% confidence interval, when $n = 150$. The dashed line is the estimator's target error range. Results are based on $5\,000$ samples for each given slot number.

Figure 3 illustrates the evolution of the bound $\Psi(s)$ for the relative error (solid line). The dashed line is the intended error margin Θ. To verify the concordance of simulation and mathematics, we compute at each slot an estimate $\hat{\Psi}(s)$ for $\Psi(s)$ by using (14) but replacing n by the averaged estimations $\hat{n}$ and replacing $\mathrm{Var}\left[\hat{n}\right]$ by the sample variance of the estimations. The values $\hat{\Psi}(s)$ are shown as crosses.

B. Robustness to Incorrect Estimation Range

The question arises as to which performance will be achieved if the actual number of nodes n is out of the estimation range assumed by the estimator, i.e., either below 100 or above 200 nodes. The estimation process provides a valid estimate (some empty slots) for $n = 10$ nodes, but the error is about $30\,\%$ after 637 slots, and more slots would be needed to achieve a better estimation. We also studied the case for $n = 1\,000$ (not shown in figure). Here, the variance of the

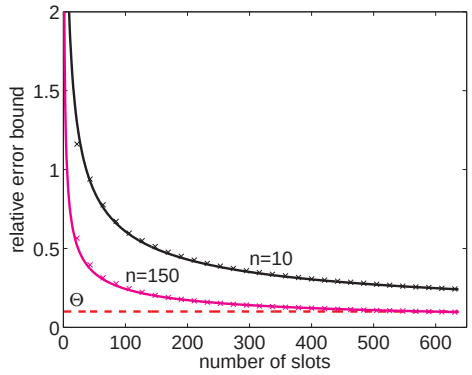

Fig. 3. Bound $\Psi(s)$ of the relative estimation error ϵ.

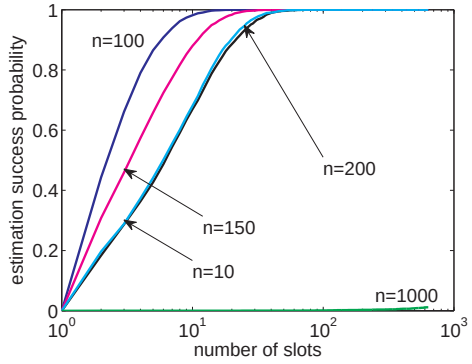

Fig. 4. Estimation success probability $P[0 < e < s]$

estimated values is almost zero for the chosen p and when $s < 650$. The reason for the low variance is that empty slots occur very rarely for the used access probability.

Figure 4 shows the estimation success probability P$[0 < e < s]$ over time for different n values. For all values except $n = 1\,000$, the success probability approaches $100\,\%$ within 100 slots, indicating that meaningful statistics can be collected and the estimation process is successful. The success probability for $n = 1\,000$ nodes is nearly zero, due to the p implied by the given setting. Only at around 600 slots the curve starts increasing, and many more slots are required to obtain valid estimations.

C. Choosing an Appropriate Estimation Range

Table I gives some examples for the number of slots s required to estimate n assuming a certain estimation range $[n_{\min},\ n_{\max}]$. An error threshold Θ is requested with a confidence of $\alpha = 95\,\%$. The estimation success probability is $\Omega = 99\,\%$. The results hold for an arbitrary number of actual nodes n, as long as $n \in [n_{\min}, n_{\max}]$.

The number of required slots depends on the estimation range $[n_{\min}, n_{\max}]$ assumed by the estimator. Clearly, the smaller this interval is, i.e., the more *a priori* information we have about the approximate number of nodes, the faster the estimation can take place. We also observe that s just depends on the fraction $\eta = n_{\max}/n_{\min}$ for a given accuracy. For

TABLE I
REQUIRED NUMBER OF SLOTS TO ESTIMATE $n \in [n_{\min}, n_{\max}]$

$n_{\min}$	$n_{\max}$	Θ	s	time
1	10	10%	1210	10.9 ms
1	10	20%	305	2.2 ms
1	100	10%	6094	54.8 ms
1	1000	10%	45843	412.6 ms
1	1000	20%	45843	412.6 ms
1	10000	10%	585830	5272 ms
10	100	10%	1210	10.9 ms
100	1000	10%	1210	10.9 ms
1000	10000	10%	1210	10.9 ms
100	200	10%	637	5.7 ms

instance, an estimation between 1 and 10 nodes requires the same number of slots as an estimation between 100 and 1 000 nodes. This insight is generalized in Figure 5, which depicts the number of slots s as a function of η.

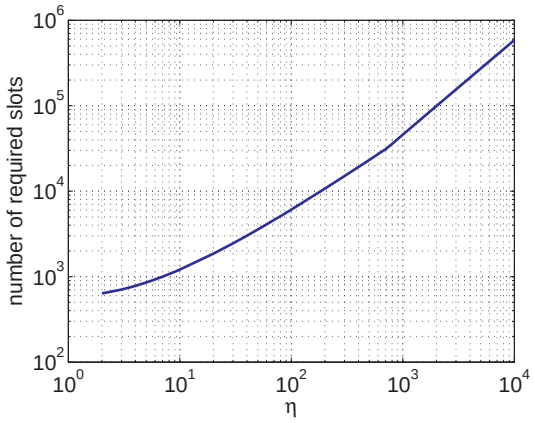

Fig. 5. Number of slots s required to estimate n as function of $\eta = n_{\max}/n_{\min}$. An error threshold $P[\epsilon \leq 10\,\%] \geq 95\,\%$ and an estimation success probability of $99\,\%$ is requested.

This fact enables us to draw some conclusions for the design of a time-efficient estimation process. If a large range of operation is needed, it is inefficient to use a large interval of $[n_{\min}, n_{\max}]$ as input to the estimator. It is better to apply *multiple* rounds of estimation with *smaller* intervals.

For example, an estimation process in the range $[1, 10\,000]$ requires more than 1/2-million slots. If we split up the estimation process into four rounds — the first round with $[1, 10]$, the second with $[10, 100]$, the third with $[100, 1\,000]$, and the fourth from $[1\,000, 10\,000]$) — the number of required slots is reduced to $s = 4 \cdot 1\,210 = 4\,840$. This can be done statically (without feedback from the querying node) where each node automatically changes its precomputed p-value after each 1210 slots to cover the next higher range. If there is some *a priori* knowledge about the number of nodes and a suitable η can be determined, it is not necessary to scan the full range.

We now ask: What is the impact of the required error margin Θ on the number of slots? Comparing row 1 and 2

of Table I, we observe that a larger margin can reduce the required number of slots. Comparing row 4 and 5, we observe that the number of slots does *not necessarily* decrease with increasing Θ. It depends on the overall setting as to which of the terms in (16) determines s.

Finally, let us briefly discuss how much time is needed in practice to perform an estimation. Let us give an example using the standard IEEE 802.11g for wireless communications. The column named 'time' in Table I shows the duration of an estimation attempt assuming a slot period of $9\,\mu$s. In this period, an IEEE 802.11g interface can switch from transmit to receive mode and detect channel activity in its reception range. Although the number of required slots is high, the estimation time is small since no data needs to be transferred during slots beside detectable busy tones.

IV. Related Work

Kodialam *et al.* [8] introduce methods for Radio Frequency Identification (RFID)-readers to estimate the number of tags in their range. Tags choose uniformly randomly a slot in a frame and transmit during this slot a message with a certain probability provided by the reader. The reader counts the number of empty and collided slots and uses these statistics to estimate the number of tags. The reader updates the transmission probability for the tags based on previous results and initiates a new contention frame until a certain accuracy is met.

Howlader *et al.* [9] apply the idea of [8] to underwater communications. Due to capturing effects they use only the frequency of empty slots to conclude on the number of neighbors. They infer from colliding slots and slots with single transmissions to node distributions.

Krohn *et al.* presents in [10] the idea to use jamming signals (transmitting noise) to exchange data in wireless networks. The authors motivate their idea by the fact that the overhead,e.g., sophisticated coding or synchronization preamble, for transmitting single bit information, e.g., yes-no answers, can be of magnitudes higher than the actual information itself.

The method of linear counting, presented by Whang *et al.* in [11], uses a probabilistic algorithm in order to estimate the cardinality of a column, that is the number of unique values in a column of a relation of a database. Likewise as in our approach, they analyze the number of empty slots in the hash table to infer about the cardinality.

Among these works, [8] is most related to our proposed method. However, there is a significant difference since the querying node in our method does not have to give any feedback after each frame. This is possible since our method does not require an update of the transmission probability. Furthermore, by exploiting only the empty slot statistics, there is no need for signaling overhead that would allow to detect collisions. On this account nodes in the estimation process do not need to transmit any information besides simple busy tones, which allows for a faster measuring phase and, thus, a quicker estimation.

V. Conclusions

In this paper, we have introduced an approach for estimating the number of neighbors, optionally, with a particular property, of a given node. The concept is based on a contention-based, probabilistic estimation method. Due to its low overhead, the proposed method is an attractive approach within wireless protocols. Possible applications are MAC protocols, relay selection algorithms, and sensor network coverage estimations.

In the implementation we propose the use of busy tones which allows for very short communication slots and thus for a fast execution of the estimation algorithm. We have shown that there is a tradeoff between the number of used slots and the accuracy of the estimation.

An important lesson learned is that without *a priori* information of the neighborhood or any *feedback* mechanism from the querying node, it is more efficient to use a set of different access probabilities than a single one to estimate a big range of potential neighbors.

In a further step, we plan to split the estimation process in two phases. In the first phase, we plan to cover the whole range with low accuracy and coarsely estimate the number of neighbors. This result is used in a second estimation round with smaller range to increase the accuracy. We believe that this method will reduce the required number of slots significantly while keeping the required feedback small.

Acknowledgments

This work was performed in the project *Cooperative Relaying in Wireless Networks* of the research cluster Lakeside Labs and was partly funded by the European Regional Development Fund, the Carinthian Economic Promotion Fund (KWF), and the state of Austria under grant 20214/15935/23108.

References

[1] S. Kumar, V. S. Raghavan, and J. Deng. Medium access control protocols for ad hoc wireless networks: A survey. *Ad Hoc Networks*, 4(3):326–358, 2006.

[2] K. Nikano and S. Olariu. Uniform leader election protocols for radio networks. *IEEE Trans. Parallel Distrib. Syst.*, 13(5):516–526, May 2002.

[3] H. Adam, W. Elmenreich, C. Bettstetter, and S. M. Senouci. CoRe-MAC: A MAC-protocol for cooperative relaying in wireless networks. In *Proc. IEEE GLOBECOM*, Honolulu, Hawaii, Dec 2009.

[4] L. G. Roberts. ALOHA packet system with and without slots and capture. *ACM SIGCOMM Comput. Commun. Rev.*, 5(2):28–42, 1975.

[5] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

[6] L. M. Leemis and K. S. Trivedi. A comparision of approximate interval estimators for the Bernoulli parameter. *The American Statistician*, 50:63–68, 1996.

[7] C. R. Rao. *Linear Statistical Inference and Its Applications*. John Wiley & Sons, 2 edition, 1973.

[8] M. Kodialam and T. Nandgopal. Fast and reliable estimation schemes in RFID systems. In *Proc. ACM MobiCom*, Los Angeles, CA, Sep. 2006.

[9] S. A. Howlader, M. R. Frater, and M. J. Ryan. Estimating the number of neighbours and their distribution in an underwater communication network (UWCN). In *Proc. Mil. Commun. Inf. Syst. (MiLCIS)*, Canberra, Australia, Nov 2007.

[10] A. Krohn, M. Beigl, and S. Wendhack. SDJS: Efficient statistics in wireless networks. In *Proc. IEEE ICNP*, Berlin, Germany, Oct 2004.

[11] K. Whang, B. T. Vander-Zanden, and H. M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Sys.*, 15, 1990.