SDJS: Efficient Statistics in Wireless Networks

Albert Krohn, Michael Beigl, Sabin Wendhack
Telecooperation Office
Universität Karlsruhe
{krohn, michael, wendhack}@teco.edu

Abstract

Synchronous Distributed Jam Signalling (SDJS) is a new transmission scheme targeted to highly mobile and ad hoc wireless systems. It is based on the synchronous, parallel and superimposing emission of jam signal on the physical layer. SDJS is intended to be implemented as a feature on existing standards. It enables those system with the ability to fast estimate statistical parameters. This paper presents SDJS in general and then focuses on the application to estimate the parameter number of devices *in a mobile setting. We studied SDJS and its application through a mathematical model and simulations and proved the idea in a real-world implementation on a mobile network, where we estimated the number of devices in real time (within $5ms$), increasing the estimation speed compared to traditional approaches by factor 1000.*

1. Introduction

Communication in pervasive and wearable computing scenarios has to cover tasks that are normally summarized under the name *ad hoc networks* or *sensor networks*. The typical properties of those settings are that many devices (which are in this paper also referred to as stations or nodes) work together in a spontaneous peer-to-peer manner without a central access point or master device. The participating devices are highly mobile which requires adequate and very flexible connection methods and data exchange schemes. Communication is normally cell-oriented and supported by routing like surveyed in [1]. The high mobility leads as well to fast changing communication settings. The number of devices working together and their positions vary constantly.

During various settings in the Smart-Its [2] project, we designed, simulated, implemented and tested a wireless, RF based sensor and communication platform which is now used in many other projects as well. Our experience with the design of such a network shows that in such settings classical network subjects like reliable data transmission and routing are still issues, but the importance of measuring and controlling the network states and nodes conditions grows. This is due to the fact that nodes are frequently leaving or joining the network but also because environment conditions are constantly changing. Such change in the conditions influences not only the network behaviour for routing and hand-over but also influences the network behaviour in small areas like the neighbouring nodes in single hop distance. Communication parameter like bit or packet error rate, throughput or channel delay vary and influence the network performance also in small scale areas. Furthermore, not only the network technology but also applications need to be aware of these changing conditions. Traditional approaches use control messages to inform nodes about changing conditions. This approach consumes considerable amount of bandwidth and therefore as well energy in frequently changing environments with many network nodes.

One of the most important condition parameter that we found in our settings, such as the Relate project [3], is how many active wireless devices are in the environment at a given point in time. This example is more closely explained in the next section. Other condition parameters that where required are for example the number of nodes with non-empty sending buffers or the number of nodes running low on energy.

In this paper we introduce a new approach for estimation and exchange of condition parameters via RF communication in a very efficient way. The method is modelled, simulated and implemented in a real world setting with 50 wireless devices. The results of the test runs are promising.

1.1. Example Application and Motivation

In Relate we have developed prototypes for relative positioning of tangible user interface. Relate technology consists of sensors - especially ultrasonic sensors - for determining distances and angles between objects and a wire-

less link to organize the measurements and to exchange results. For precise measurements it is required to frequently - in practice several times per second - estimate the number of involved nodes within the environment.

A simple approach is to use control messages and ping all devices within the environment. For our setting with large number of devices this is very inefficient. With e.g. 100 devices, the channel access has to handle 100 concurrent accesses. In [4] the mean delays of packet transmissions in a single hop networks based on IEEE 802.11 are analysed. Simulations and calculation show that the mean time necessary for 100 stations to answer would be around 5 seconds. Although the delay also depends on initial conditions of the MAC protocol (which can positively influence the overall time) this approach leads from the principle to unacceptable termination times for ad-hoc settings due to its method of collecting the information successively.

Instead of using such a serial approach we propose a probabilistic transmission method allowing parallel access to the channel. The principle of this method is introduced in section 2. How to use this method for very fast estimation of the number of devices in a wireless cell is outlined in section 3. Simulations are presented and discussed in section 4. Section 5 shows a complete implementation and practical evaluation of the proposed techniques.

2. *SDJS* : Synchronous Distributed Jam Signalling

The Synchronous Distributed Jam Signalling is a transmission scheme that differs significantly from typical radio communication. Instead of communicating encoded (digital) data it transfers information by sending jam signals on the physical layer in a given order. The SDJS method allows to estimate information based on sensing the received signals and allows parallel (overlaid) sending of information thus largely reducing transmission time compared to other approaches. SDJS is in the first instance not intended for transmission of data nor is it a wireless protocol. SDJS is dedicated to transmission of information signals that are used to estimate conditions and parameters in wireless ad hoc networks.

The principle of SDJS works as follows: Sources of information - e.g. all devices within a network cell - emit jam signals synchronously. Signals are intended to superimpose on the channel. The transmission does not include source coding nor channel coding nor any error correction. Receivers detect the jam signals and estimate information based on statistical methods. SDJS can be integrated as a feature into nearly every existing wireless platform or protocol as it only requires few things from the platform it is implemented on:

1. the ability to send a jam signals (e.g. turn on the carrier

and transmit noise)

2. the ability to sense jam signals from other transceivers (e.g. using the carrier sense mechanism)

3. a precise time synchronization among all participating devices

4. methods for reliable broadcasting of messages

5. methods for channel reservation (e.g. RTS/CTS)

SDJS always start with a broadcasted start signal (a normal broadcasted data packet) that all participants receive. This start signal defines the beginning of the short SDJS transmission scheme and refreshes the time synchronization between the devices. After this start signal, a known number of *slots* follows. In each of theses slots, each station decides to either send a jam signal or try to detect possible jam signals from other stations. Jam signals from different devices can superimpose on the channel. This transmission scheme could be interpreted as an ON/OFF keying with reception during off-times. The minimum length of the slots is given through the radio and protocol parameter of the underlying platform. The devices must be able to either *switch from reception to transmission and transmit a jam signal* or *switch from transmission to reception and detect a jam signal from a remote transmitter* within one slot. Besides, the channel delay can affect the timing scheme, as it shifts the synchronization of the devices if the synchronization is realized just through the reception of the start signal. Taking the physical layer parameters from IEEE 802.11a [5], the theoretical minimal slot time is:

$$
\begin{aligned}
slottime = & \max(t_{rx/tx}, t_{tx/rx}) + 2 \cdot t_{channeldelay} \\
& + t_{carriersense} \\
= & \max(2\mu s, 12\mu s) + 2 \cdot 1\mu s + 4\mu s \\
= & 18\mu s \qquad (1)
\end{aligned}
$$

This means that an 802.11a specified transceiver can perform the necessary action in a slot time of $18\mu s$ equal to 55kBit bandwidth. This theoretical minimum of the slot time must eventually (depending on the transceiver platform) be lengthened to deal with additional synchronization weaknesses and clock drifts.

After the reception of the start signal, each station has to prepare a *transmission vector* carrying the information whether to send or to receive in each of the following SDJS slots. The transmission vector is a sequence of bits representing transmission of jam signal (one) or reception (zero) for the according slots. Then, during the SDJS scheme, each device generates a second vector, the *reception vector* which contains a *one* at positions, where jam signals have been detected and a *zero* where the channel was quiet. At positions where the station itself transmitted a jam signal, the reception vector contains a one as well. The details are explained in an example in the following section.

2.1. General activity flow in SDJS

Figure 1 shows the activity flow of the SDJS scheme. The devices (A,B and C) are participating in a network cell and are SDJS enabled. They perform normal data exchange at times before t_1 which is marked with packets named "data". At t_1, Station B sends out a broadcast packet with the information to start the SDJS scheme. It contains information on the SDJS scheme and reserves the channel until t_6. After the successful emission of this start signal (t_2), all stations prepare a SDJS transmission vector (interpreted from left to right). Device A has 1001000, device B 1000010 and C has 0010100. These transmission vectors can be found in Figure 1 as black and grey cells during the SDJS scheme. The SDJS slot length is $t_4 - t_3$. The SDJS reception vector is represented as zeros and ones in the SDJS black and grey cells. After the preparation of the SDJS transmission vector, each device follows the SDJS scheme. The first steps are now explained in details:
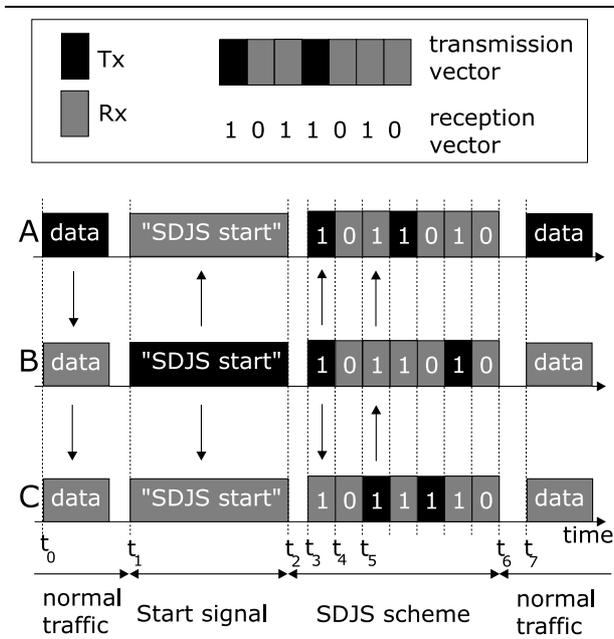

Figure 1. SDJS scheme

t_2 : The SDJS scheme begins, the channel is reserved until t_6 and all participating stations prepare their transmission vector.

t_3 : The first SDJS slot starts. Device A and B turn to Tx and transmit a jam signal, because they have *one* in their transmission vector on the first bit. Device C has *zero* as the first bit in the transmission vector and therefore turns its transceiver to reception and looks for a jam signal. The jam signal can be detected (both A and B signals superimpose on the channel) and Device C places a *one* in the first position of the reception vector. Devices A and B place as well a *one* in the first position of the reception vector, because they transmitted (and therefore *virtually* received their own signals).

t_4 : All stations have a *zero* in their second position of the transmission vector. They all turn to Rx and try to find a jam signal. As there is no device transmitting, nothing can be detected on the channel and all three devices place *zero* in the second position of their reception vector.

t_5 : Now device C has ONE on third position of its transmission vector, therefore emits a jam signal. The other both turn to receive, find the jam signal from station C and place a *one* in the according positions of their reception vector. The SDJS schemes continues like explained until t_6. Then the reception vectors are used for further calculations.

t_6 : The SDJS scheme ends and the access and traffic on the channel is switched back to normal operation.

t_7 : The next normal data packet is exchanged.

The SDJS scheme can be evoked at any time. It begins with a broadcasted message and ends after the reservation time for the SDJS scheme. The SDJS scheme can be used for many different statistical approaches in wireless networks. For this reason, the starting packet needs to contain details on the appended SDJS scheme like the length of the slots, the number of bits in a transmission/reception vector, in which way the devices should select the positions for transmission of jam signals. In the upper example, all station select two positions in a 7 bits long transmission vector. Any other way to choose the number and positions of the jam signals is possible and can make sense in the according context.

2.2. Technical details on SDJS

The SDJS scheme is based on two mechanisms: using jam signals for transmission during a slot time and using a carrier sense mechanism to detect jam signals. Therefore, the timing of a slot has to be selected (see section 2) to guarantee that signals can be found including the consideration of channel delays etc. This simple mechanisms of jam signalling can cause two major problems which are typical low level network issues:

1. The false alarms: A receiving device can detect a jam signal even though there is none due to noise on the channel.
2. Bad detection probability: A present jam signal is not detected because the emitting device is far away or the signal is attenuated due to multi path fading, objects in the line of sight etc.

These two major problems are not explicitly addressed in this paper. Nevertheless, mechanisms are used to keep those issues small in the current implementation: Both problems can be reduced by increasing the number of samples to detect a jam signal. The implementation presented in section 5 uses 192 samples per slot plus some detection algorithms and heuristics to keep the influence of both effects small to the results. Other parameters like automatic gain control and thresholds for detection where selected to be optimal for the setting. The mathematical model in section 3 could be extended for modelling the channel errors by simply multiplying them as probabilities wherever a signal reception is modelled.

3. Estimation of number of devices

The introducing example from the Relate project motivated the requirement to estimate the number of devices in an ad hoc setting of mobile objects equipped with wireless radio. The technical mechanism used was presented in section 2. Now a mathematical model is presented to form the basis for a fast and accurate estimation of the number of devices. For the purpose of this estimation the SDJS mechanism can be used in many ways. *One* possibility (motivated from the target application) is using the SDJS scheme only *once* and then using the reception vector as input to an estimator. The decision to use SDJS in this way was based on the motivation to produce an especially *fast* estimation. For the following section, we focus only on this one model and way to take advantage of SDJS.

3.1. The analytic model

In this model, each device sends out only one jam signal per scheme. The position of this jam signal is chosen randomly with an even distribution. The reception vector then carries at least one *one* (or more, depending on other devices around). The basis for the estimation is the number of ones (sum of digits) over the reception vector. If e.g. ten devices are present and they all chose different positions for their jam signals, all devices would find ten *one*s in their reception vector. But if e.g. two devices chose the same slot, only nine slots would be positive. The possible superposition of jam signals on the channel requires an estimation based on a model for those collisions. If the number of devices exceeds the number of slots significantly it is very likely that all positions in the reception vector are positive and the estimation is useless. The model presented here is targeted on moderate number of devices (around 250 max-

imum) and focussing on a very fast estimation. Other models can be targeted to other typical characteristics.

Some descriptors are defined for the modelling:

k: the actual number of devices present

$\tilde{k}$: the estimated number of devices

A_k: probability variable for the number of jam signals received (number of *one*s in the reception vector), depends on the actual number k

a: the number of received jam signals in one experiment (sum of digits of a reception vector)

s: the number of slots in the SDJS scheme (equal to the number of bits in a transmission/reception vector)

The first mathematical question that occurs is to model the probability to see a jam signals when k devices are actually there. This problem can be modelled through basic combinatorics: k elements are to be placed into s slots. For this model, only the combinations that occupy exactly a slots are of interest.

1. the number of all combinations of k elements to be placed in s slots is s^k
2. the number of combinations to chose a slots is $\binom{s}{a}$
3. taking k elements, the number of combinations to form groups of the size a is the surjective mapping from k to a:
$$\sum_{i=0}^{a}(-1)^i \binom{a}{i}(a-i)^k$$
4. now, the probability to find a combination of the size a is:
$$P(A_k = a|k) = \frac{\binom{s}{a}\sum_{i=0}^{a}(-1)^i \binom{a}{i}(a-i)^k}{s^k} \quad (2)$$

The resulting probability needs some interpretation. First of all, it's impossible to receive more *one*s than devices are actually there ($a \leq k$); only if errors occur which are not modelled here. Then the number of found *one*s is as well limited to the number of slots ($a \leq s$). Figure 2 shows the probability distribution of the received number of jam signals a depending on the actual present number of devices k. This example is based on a vector length of 64 which is equal to the one used in the implementation in section 5. It is obvious that e.g. for $k = 40$ the number of received *one*s (a) has its highest probability below 40 that is to say at $a = 30$. This is due to the collisions. With 40 devices sending a jam signal on a random position, it is very likely that some collisions will always occur and it is less likely that no collisions will occur. The probability for no collisions with 40 devices can hardly be found at $k = 40, a = 40$.

The next step is to model an estimator to find the actual number of devices only from the reception vector.
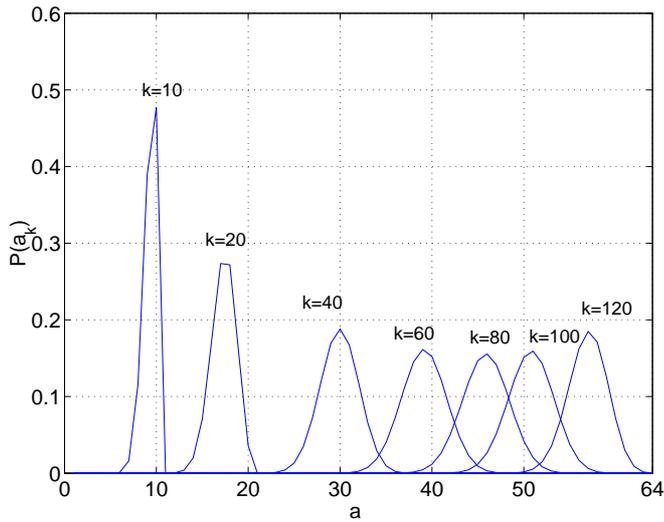
Figure 2. Probability distribution

a	$\tilde{k}$	$P(A_k = a\|k)$
1	1	1,000
5	5	0,855
10	10	0,481
15	17	0,328
20	24	0,245
25	31	0,231
30	39	0,209
35	49	0,173
40	62	0,175
45	79	0,175
46	84	0,176
50	98	0,177
55	127	0,182
60	173	0,257
61	190	0,274
62	219	0,324
63	275	0,409
64	n/a	n/a

Table 1. Some values of the Estimator for $s = 64$

3.2. The estimator

As already mentioned, the estimation will be based on the number of *one*s in the reception vector. As the estimator should only use a single reception vector, it is clear from the beginning that the estimation can only output s different results. The reception vector can have $1..s$ ones but not more. Each of those numbers will result in an estimation of the actual number of devices present. The finished estimator will therefore be a look-up table mapping a to the estimated number of devices $\tilde{k}$. The estimation used here is a maximum likelihood estimation. For a given a from the reception vector of length s, the likelihood estimator must resolve the following statement:

$$\frac{\partial}{\partial k} \cdot \frac{\binom{s}{a} \sum_{i=0}^{a} (-1)^i \binom{a}{i}(a-i)^k}{s^k} = 0 \qquad (3)$$

Doing so, for each possible number a with $0 < a \le s - 1$ there is an estimation result $\tilde{k}$. The highest value ($a = s$) cannot give a valid estimation result because the higher the actual number of devices k is, the more probable it becomes to actually receive all $a = s$ bits in the reception vector. The table 1 shows some results from the estimator. Each possible received number of jam signals a is mapped to the according $\tilde{k}$ with the highest probability to produce a bits in the reception vector (maximum likelihood estimation). Receiving e.g. $a = 40$ ones in the reception vector would result in the estimation that 62 devices are around. If an estimation is implemented based on this table some constrains have to be considered. As the estimation is not a linear function of the found jam signals ($\tilde{k} \ne c \cdot a$), the variance of the estimation

increases with the number of devices. The proposed estimation is based on only a single SDJS scheme. Looking into table 1 it is clear that only certain results $\tilde{k}$ are possible. It is e.g. impossible to estimate the number $k = 80$ precisely with only one SDJS scheme. The value $\tilde{k} = 80$ does not exist in the estimation table (would be around $a = 45, 7$) and can therefore never be a result of the estimation. For precise measurements different statistical uses of SDJS should be considered.

3.3. Alternative models

There are many other models possible based on the SDJS scheme for estimations of parameters - not only the one presented above. We have discussed other approaches that focus e.g. more on the precision of estimations and less on the speed. One works with only 6 SDJS slots, but needs typically more than 100 consecutive SDJS schemes. It is based on an estimation of the probability of the presence of jam signals on those 6 slots. We found a way to include that short period of 6 slots into the physical frame of Aware-Con [6] as an inherent part of the protocol and needed only 1% of the bandwidth for it.

Another possibility is to use more than 64 slots in the proposed method to improve the precision of the estimation for higher numbers of devices. Taking the $64 \mu s$ slot-time from the implementation, even 512 slots would only need $33ms$

which can still be considered as real time for human actions.

4. Simulations

The Simulation were based on a 256-slots SDJS scheme. The estimation table was calculated prior to the actual simulations. Then, different settings for $k = \{50, 100, 200, 400\}$ were simulated. The detection errors were modelled with the symmetric error SER (slot error rate). SER gives the probability that a jam signal is not detected although it is there or that a jam signal is being detected although the channel was quiet. The SER was set to values up to 10^{-2} which is worse than normal simulations for wireless channels which assume BER (bit error rate) of better than 10^{-3}. Additionally, the BER is hard to compare to the SER as the detection of a slot is a different mechanism than receiving data and therefore needs different modelling. A total of over 20000 estimation form the basis of the Figure 3. The x-axis shows the errors (the unit is devices) of the estimations during the simulations. Positive or negative errors are regarded equally. The y-axis shows the accumulated percentage of the occurrences. This means that e.g. on the curve for $k = 100$, 95% of the simulations returned with an estimation error equal or less than eight. Only for $k = 400$, a curve with an error ($SER = 0.01$) was included. It can bee seen, that the influence of bad detection is fairly small. Curves for lower number of devices ($k = \{50, 100, 200\}$) did not change when the SER was tuned from 0 to 10^{-2}. For higher number of devices, the quality of the estimation generally decreases. This is due to basically two reasons:

1. the estimation itself becomes worse, because for higher numbers of a, the estimation $\tilde{k}$ is not as precise as for smaller number. Table 1 shows the non-linear relation between a and $\tilde{k}$

2. a single error (false alarm, bad detection) results in a higher absolute error again due to the estimation model.

5. Implementation and experiment

The proposed SDJS physical layer function and the statistical estimator have been implemented on the *Smart-Its particle computer* [2], [7]. The particle computer is a system of wirelessly connected small sensor nodes used for development of applications in the area of ubiquitous and pervasive computing. The particle computers have a customized wireless protocol called AwareCon [6], which guarantees time synchronization, reliable communication, access control and fulfils the SDJS requirements mentioned in section 2. Nevertheless, broadcasting messages in a reliable way is
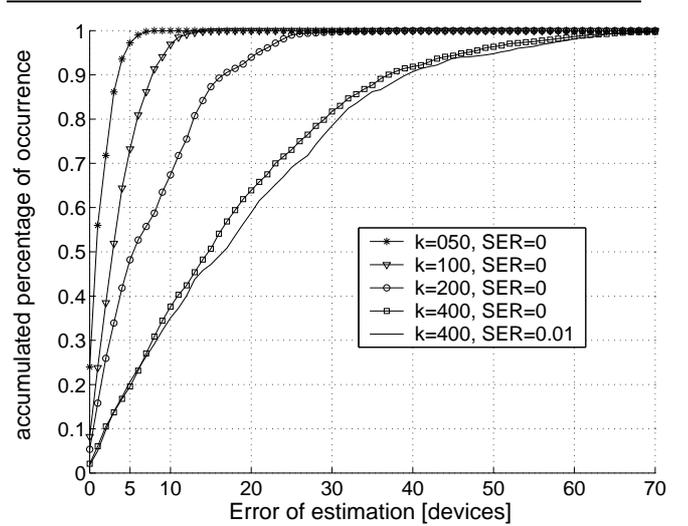

Figure 3. Simulation for s=256

still not solved completely and a lot of research effort is invested to find solutions there for mobile networks. See section 6 for background information. To shield the effect of incomplete broadcasting in the setting, the broadcast mechanisms were supervised manually.

The time for one SDJS-*slot* was $63\mu s$. The AwareCon protocol uses a physical frame of the length of $13ms$. Normally, one physical frame transports one packet of data. For SDJS, the starting signal plus 64 SDJS-slots could be placed in that frame which resulted in a performance time of one SDJS scheme of as well $13ms$. The time actually needed for the access control, start signal, SDJS slots and estimation is only a $5ms$ portion of that $13ms$ frame time. The experiments were carried out in an office environment. The setting included different group sizes from 5 to 50 devices and different spatial arrangements with near and far devices and groups of devices within one room. More estimations were taken for higher numbers that for smaller ones. An overall number of 1820 estimation were gathered during the various settings. The figure 4 gives an example of a very dense setting on a desk in an office. In other settings the devices were more distributed in the room, attached to persons, windows, chairs and all kinds of every day objects in the office environment.

Figure 5 shows the overall results of all the experiments carried out in our office. The z-axis shows the relative occurrence of the pairs *actual number of devices - estimated number of devices*. Starting from an *actual number of devices*, one can read out how often which number of devices was *estimated*. This figure gives a good impression of how good the actual estimation worked. The estimation follows the bisection line of the x-y axis. For higher number of de-

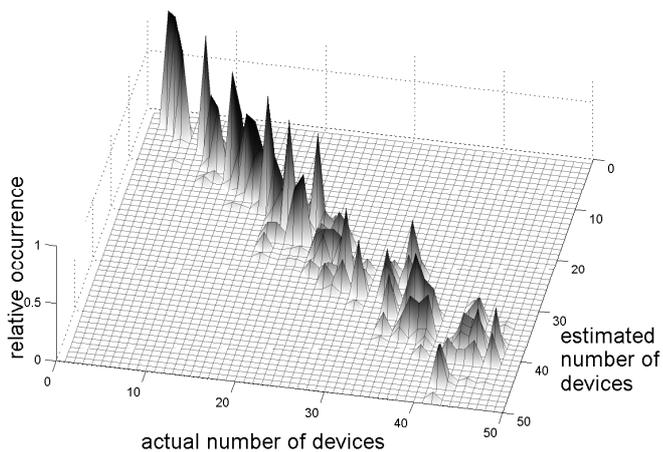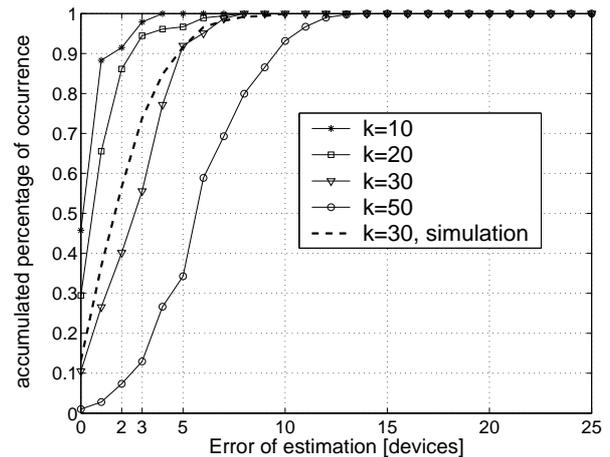**Figure 4. An experiment with over 40 particle computers**

Figure 6. Accumulated error distribution of the experiments

from simulations. The $k = 30$ curve from the implementation is nearly identical to the one from simulations. It only differs by one device. This is a convincing proof that the idea of SDJS can really contribute useful results in the real world.

6. Related work

The closest related work published on the issue to estimate the number of nodes can be found in [8]. The authors present a statistical approach to estimate the number of nodes in a sensor network based on collected data packets that carry source IDs. They provide an efficient estimator and prove their concept in simulations. However, the scheme presented there is targeted more towards large scale sensor networks whereas our work is focussed on highly mobile settings and the speed (real time) of the estimation. In [9], a scheme is presented that uses jam signals that superimpose on the radio channel as well. The goal there is to improve the detection probability of signals with "boosting" them on the channel using overlaid jam signals. It works as well in a distributed way.

A mentioned precondition for SDJS is the ability to broadcast messages in a reliable way. Different research efforts have addressed this network issue. Recently, the problems of using standard and known broadcast mechanism (like from IP) in mobile and ad hoc setting has been discovered as non-trivial. A survey of broadcast methods for mobile networks and their typical behaviour can be found in [10]. In [11], an efficient scheme is presented that is based on the 802.11 standard but uses RTS, CTS, ACK and a new RAK packet in an optimised way to avoid unnecessary contention

Figure 5. Experiment on estimation

vices, the variance obviously increases. This is due to the same reasons given for the simulations.

The errors of the estimation during the experiment are summarized in Figure 6. They show that for e.g. $k = 20$, 95% of the errors are equal or less then 3 objects. 20 Objects is a typical number for applications and setting that we built in the recent years. Therefore the good estimation result is promising and very useful. For $k = 30$ a curve from simulations ($SER = 0.01$) was added to the graph to prove the implementation and compare the actual quality of the estimation in a real-world setting to the "academic"results

periods during the distributed operation in the networks. The work in [12] presents a statistical approach to the delivery of multi/broadcast messages. The protocol proposed guarantees the distribution of a message to a given area only to a certain probability. In [13], the broadcast is organized in an efficient way using piggy-backed acknowledges. All that work struggles with the typical behaviour of nodes in mobile setting like fast changing and non-predictable routes for delivery and limited resources. Like the comparison in [10] shows, the "silver bullet" for reliable multi/broadcast for mobile settings has not been found yet. The deterministic approaches often have insufficient scalability and the probabilistic approaches lack in reliability. For our experimental setting, the broadcasting was based on a probabilistic scheme.

Another important precondition for the use of SDJS is a precise time synchronization to guarantee the synchronous use of the slots during the SDJS scheme. Synchronization is in general as well a complex task in wireless ad-hoc networks. In [14] a tree-based method is proposed that is centralized and assumes that for most applications, the requirements for accuracy are relatively low (which is not the case for SDJS: high synchronization precision is necessary). The section *Related Work* in [14] gives as well a good summary of the state-of-the-art methods for synchronization in wired and wireless networks. In [15], different approaches to synchronization like diffusion, building of clusters and node based algorithms are presented and their simulation is discussed.

In our real world implementation for the experiments carried out, we used the synchronization services of the Aware-Con protocol on the particle computer platform. This synchronization guarantees an accuracy of better than $4\mu s$ between all active nodes in a network cell and better than $1\mu s$ after a resynchronization through a broadcast packet at the beginning of a SDJS scheme.

7. Conclusion and ongoing work

In this paper we presented SDJS, a novel approach towards the estimation of parameters in wireless ad hoc networks. SDJS is a modulation scheme with superimposing and synchronous jam signals combined with a statistical evaluation. One statistical model to estimate the parameter *number of present devices* was proposed, formulated, simulated and proved in a real-world implementation. It estimated the number of devices in the implementation within $5ms$ with an error of ≤ 3 devices for 95% of the experiments with 20 devices running. The simulation returned the same accuracy for estimating 50 devices but using a 4 times longer scheme. This shows that in general any required accuracy can be achieved by adapting the SDJS slot number. Compared to conventional processes like collection of IDs

of surrounding partner devices it can accelerate the estimation by at least factor 1000 for settings greater than 100 devices. The proposed scheme has a trade-off between speed and accuracy. Better estimations need longer SDJS vectors. Improvements or alternative statistical usages of SDJS have been pictured in section 3.3.

The mentioned problems of false alarm and bad detection probabilities are the most important issues to focus on in this work. The detection of jam signals must be stable and robust, correlation techniques are being discussed. The incorporation of the RSSI of the received jam signals can be helpful for weighting of signals in the statistical detection models. Further, the issue of reliable (acknowledged) broadcast in mobile settings is subject to ongoing research effort. SDJS can be used for settings where a many-to-one communication of redundant information needs to be processed efficiently. The parallel and overlaid transmission can be interpreted as a logic *or*-operation of bits on the RF channel. All data collection that uses bit wise or-combination of vectors can be realized efficiently with SDJS. A further improvement can be achieved by recombining the estimations that have been produced locally and therefore increase the statistical confidence significantly.

8. Acknowledgements

References

[1] S. Ramanathan and Martha Steenstrup. A survey of routing techniques for mobile communications networks. In *Mobile Networks and Applications, Special issue: routing in mobile communications networks*, 1996.

[2] Michael Beigl, Tobias Zimmer, Albert Krohn, Christian Decker, and Philip Robinson. Smart-its – communication and sensing technology for ubicomp environments. ISSN 1432-7864 2003/2, University of Karlsruhe, 2003.

[3] *http://www.comp.lancs.ac.uk/∼hwg/relate.html*, accessed: 04/2004.

[4] Marcelo Carvalho and J.J. Garcia-Luna-Aceves. Delay analysis of ieee 802.11 in single-hop networks. In *11th IEEE International Conference on Network Protocols (ICNP'03)*, page 146, Atlanta, USA, November 2003.

[5] LAN/MAN Standards Committee of the IEEE Computer Society. Supplement to ieee standard for information technology telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. part 11: wireless lan medium access control (mac) and physical layer (phy) specifications: high-speed physical layer in the 5 ghz band. *IEEE Std 802.11a-1999*, 1999.

[6] Michael Beigl, Albert Krohn, Tobias Zimmer, Christian Decker, and Philip Robinson. AwareCon: Situation Aware Context Communication. In *Proceedings of Ubicomp 2003*, Seattle, USA, October 2003.

[7] *http://particle.teco.edu*, accessed: 4/2004.

[8] C. Budianu and L. Tong. Estimation of the number of operating sensors in a wireless sensor network. In *Proc. of Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA., November 2003.

[9] Timo Weiss, Joerg Hillenbrand, Albert Krohn, and Friedrich K. Jondral. Efficient signaling of spectral resources in spectrum pooling systems. In *10th Symposium on Communications and Vehicular Technology in the Benelux*, November 2003.

[10] Einar Vollset and Paul Ezhilchelvan. A survey of reliable broadcast protocols for mobile ad-hoc networks. Technical Report CS-TR-792, University of Newcastle upon Tyne, 2003.

[11] Min-Te Sun, Lifei Huang, Anish Arora, and Te-Hwang Lai. Reliable mac layer multicast in ieee 802.11 wireless networks. In *International Conference on Parallel Processing (ICPP'02)*, Vancouver, Canada, August 2002.

[12] Jun Luo, Patrick Th.Eugster, and Jean-Pierre Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *InfoCom 2003*, San Fransisco, USA, April 2003. IEEE.

[13] M. Mock, E. Nett, and S. Schemmer. Efficient reliable real-time group communication for wireless local area networks. In *Third European Dependable Computing Conference EDCC-3*, Prague, Czech Republic, Sep 1999. Springer.

[14] Jana van Greunen and Jan Rabaeay. Lightweight time synchronization for sensor networks. In *Second ACM International Workshop on Wireless Sensor Networks and Applications in conjunction with ACM MobiCom 2003*, San Diego, CA, USA, September 2003.

[15] Qun Li and Daniela Rus. Global clock synchronization in sensor networks. In *InfoCom 2004*, Hong Kong, March 2004. IEEE.

[16] *http://www.smart-surroundings.org*, accessed: 04/2004.