SDJS: The Duck Hunter Problem in Wireless Sensor Networks

Albert Krohn, Tobias Zimmer, Michael Beigl, Christian Decker, Till Riedel

Telecooperation Office
Vincenz-Priessnitz-Str.3
76131 Karlsruhe
{krohn, zimmer, michael, cdecker, riedel}@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 top of existing standards. It enables those system with the ability to fast estimate statistical parameters. This paper presents how to use SDJS to estimate the parameter *number of devices* in a mobile wireless ad hoc network. This new approach can increase the speed of the estimation by a factor of *1000*. The whole process is also *independent* of the number of nodes involved.

Keywords: wireless sensor network, distributed estimation

1 Introduction

In pervasive and wearable computing, devices often work together wirelessly and spontaneous. The devices are highly mobile which requires adequate and very flexible connection methods and data exchange. The high mobility leads as well to fast changing communication parameters such as the number of devices working together, their position etc. For a network protocol it is important to update those parameters fast and accurately to achieve optimal performance. With traditional approaches that use control messages to inquire or inform nodes about changing conditions, sensor values etc. a considerable amount of bandwidth and therefore energy is used; especially when inquiring information from a large group of nodes. This makes such methods inappropriate for highly varying settings. The method we propose instead allows to collect information in a time of a single packet. This new approach enables networked nodes to estimate and exchange condition parameters via RF communication in a very efficient way. As an example, we will now explain how to retrieve the number of active nodes in a setting with the help of SDJS [1]; but SDJS can be used for many other applications such as sensor fusion or RFID scenarios [2].

2 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 parameters based on sensing the received signals and to parallely send information and thus largely reducing transmission time compared to traditional approaches.

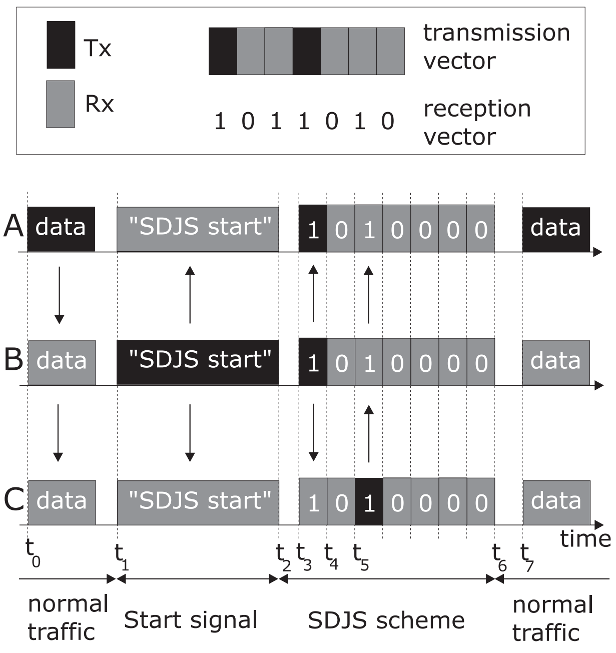

Fig. 1. SDJS scheme

SDJS always starts with a broadcasted start signal (a normal broadcasted data packet) that all participants receive. After this start signal, a known number of SDJS-*slots* follow. For our example, each station selects a random slot to 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.

For the SDJS scheme, two binary vectors are important. First a *transmission vector* carrying a *binary one* at time positions where the node will send a jam signal or a *zero* where the device will listen and not send. The second vector is the *reception vector* which carries the result after a SDJS scheme. It carries a *one* on time positions where the node transmitted or received a jam signal and a *zero* on slot positions where it did not send nor find any signal from remote stations.

Figure 1 shows the activity flow of a SDJS scheme. To illustrate SDJS, we depict the example of a 7 slots long SDJS scheme (transmission and reception vector are 7 bits long) with one positions randomly set. 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 1000000, device B 1000000 and C has 0010000. 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. During the transmissions of jam signal in the SDJS slots, the transmission vector are binary "OR"ed. During slot $t_3 - t_4$, both stations A and B send a jam signal which C can detect. It therefore places a *one* on the first position of its reception vector. After the SDJS process, all stations should carry the same reception vector 1010000.

3 The Duck Hunter Problem

This known problem from standard math literature can – with some extension in the derivation – find an new application in SDJS. We explain the corresponding term in []-brackets :
a given number of duck hunters k [number of devices] are waiting for a flock of ducks [number of SDJS slots s] to appear. They all are experienced hunters and therefore always kill a duck when they aim at it. They all have a just one bullet. Suddenly, the ducks appear and all hunters immediately aim at a random duck and shoot [transmit a jam signal]. They don't have the time to discuss who of them will aim at which duck. Therefore, a number of ducks will be killed [received number of jam signals s]; some of them even by two or more hunters [collisions of jam signals]. Others will be lucky and survive. The duck hunter problem discusses the probability that a certain number of ducks die.
In SDJS, the question is slightly different: SDJS asks how many hunters have been there. We can derive the closed form equation through a basic problem of partitions theory:

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 partitions of the size a:

$$\sum_{i=0}^{a}(-1)^i\binom{a}{i}(a-i)^k$$

[3, Theorem 2.1.7]

4. now, the probability to find a combination of the size a is:

$$P_{a|k}(a|k) = \frac{\binom{s}{a}\sum_{i=0}^{a}(-1)^i\binom{a}{i}(a-i)^k}{s^k} \tag{1}$$

Using a maximum likelihood (ML) estimator, we get the estimation $\tilde{k}$ for the number of devices through

$$\tilde{k}_{ML} = \arg\max_{k}\ P_{a|k}(a|k) \tag{2}$$

With the help of a-priori knowledge confidence intervals can also be given [2]. There is a trade-off between speed and accuracy of SDJS. Increasing the number of slots of one SDJS scheme reduces the collisions but at the same time increases the accuracy and vice versa. For a given application, the required accuracy can be achieved by adjusting the number of SDJS slots accordingly – still resulting in a *constant* and predictable process-time independent of the actual participating number of nodes.

References

1. Albert Krohn, Michael Beigl, and Sabin Wendhack. SDJS: Efficient statistics for wireless networks. In *Proceedings of the 12th IEEE International Conference on Network Protocols*, Berlin, Germany, 2004.
2. Albert Krohn, Tobias Zimmer, Michael Beigl, and Christian Decker. Collaborative sensing in a retail store using synchronous distributed jam signalling. In *3rd International Conference on Pervasive Computing*, Munich, Germany, 2005.
3. H. Joseph Straight. *Combinatrics: an Invitation.* Cole Publishing Company, Belmont, USA, 1993.