DCP: A New Data Collection Protocol for Bluetooth-Based Sensor Networks

Matthias Handy, Frank Grassert, Dirk Timmermann
Institute of Applied Microelectronics and Computer Science, University of Rostock
matthias.handy@etechnik.uni-rostock.de

Abstract

Two economic factors are essential for the success of wireless sensor networks as new key technology: low-cost hardware and strong prototype applications. Although not perfect, Bluetooth can be a driving technology in this domain. We present a new Data Collection Protocol (DCP) for wireless sensor networks. DCP is tailored to Bluetooth-based sensor nodes and therefore enables sensor network applications based on inexpensive hardware. DCP is scalable, robust, and not limited to piconet or scatternet structures.

1. Introduction

The collaboration of countless tiny sensor nodes in a network promises an enormous potential of novel applications. Advances in miniaturization and integration of electronic and mechanical components will enable sensor nodes with a size of a few cubic millimeters in the near future. At the same time, an ongoing price decline will allow the deployment of sensor networks covering thousands of nodes and, as a consequence, replace conventional wired sensors in many areas.

A possible application of wireless sensor networks is *flood prevention*. A thawing period in adjacent mountains and heavy rainfall at the same time can cause rising water levels of nearby rivers. Hence, natural or artificial dikes have to be reinforced to keep flood waters at bay. Typically, sandbags are piled along hundreds of meters or kilometers along a river or lake. These dikes of sandbags have to be monitored permanently during the critical phase. Any site of leakage has to be detected as soon as possible in order to reinforce it with additional sandbags. Humans continuously walking along the dikes usually monitor the site. However, this kind of monitoring does not provide sufficient protection against breaking of dikes. A wireless sensor network could help to detect leaks earlier and thereby prevent floods. Each sandbag can be equipped with at least one sensor node. The sensor data of each node has to be transmitted to a central collection site, called *base station*. A base station is only interested in sensor information that differs from a nominal value (for example, the base station is not necessarily interested in sensor information of dry sandbags).

The base station itself has to take care of receiving all relevant information from the network and to collect sensor information from nodes. This issue is often referred to as *data collection*. Our flood prevention scenario belongs to the group of data collection problems. However, other sensor network applications can not be classified as data collection problems. For example, tracking of an object, e.g. a vehicle is not a data collection problem since most of the communication occurs among nodes, comparing their sensor samples in order to anticipate the motion of the object.

In this paper, we introduce and discuss a novel data collection protocol (DCP) for Bluetooth-based sensor networks that can be adopted for several sensor network applications such as flood prevention. The applicability of Bluetooth in wireless sensor networks is subject of recent research activities. In [1], Leopold et al. discuss various advantages and drawbacks of Bluetooth concerning its applicability in sensor networks. In their conclusion, the authors describe Bluetooth-based sensor networks as applicable for a certain type of applications where data transfers appear infrequently but at high rates. The interference reducing FHSS-technique and a fully implemented MAC-layer are the qualifying properties of Bluetooth. Bluetooth drawbacks cover the issue that in order to transfer data between two Bluetooth nodes a connection has to be established in advance. In addition, the size of Bluetooth-piconets is limited (max. eight active members, one master and up to seven slaves), a fact that can complicate the assembly of large sensor networks. Overlapping piconets, so-called scatternets, can solve this problem. The following chapters will yield more advantages and drawbacks of Bluetooth concerning its applicability in wireless sensor networks.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of Bluetooth and discusses relevant concepts and strategies in detail. Our novel data collection protocol is described in Section 3. Section 4 presents simulation results. We compare DCP with existing protocols in the sensor network domain in Section 5, followed by concluding remarks in Section 6.

2. Preliminaries

DCP utilizes various Bluetooth-specific mechanisms and concepts. For a better understanding of our protocol, this section gives an overview of Bluetooth and clarifies some relevant aspects in more detail. Notice that all statements about Bluetooth refer to release 1.1 of the Bluetooth specification [2].

Bluetooth is a short-range radio standard operating in the ISM-band at 2.4 GHz. It was originally intended as cable replacement for peripherals of personal computers and uses a spread-spectrum-radio. Moreover, a communication channel between two devices changes its frequency every 625 µs. Bluetooth devices can form a so-called piconet. Each piconet consists of one master and up to seven active slaves. The master coordinates piconet communication. Even two connected Bluetooth devices form a piconet, requiring a role assignment. One device becomes master, the other device acts as slave. Overlapping piconets form a so-called scatternet. Scatternets imply that some Bluetooth devices are member of more than one piconet and have been a theoretical model for a long time, not supported by Bluetooth device vendors. Recently, Bluetooth devices are available with limited scatternet support.

Bluetooth technology differentiates, among others, between *inquiry* substate and *page* substate. The *inquiry* substate is used by a unit that wants to discover new devices. During inquiry, the searching device periodically broadcasts ID-packets on changing frequencies (two packets in each 625 µs TX-slot). Each ID-packet contains a 68-bit *Inquiry Access Code* (IAC), either as a general IAC (GIAC) or as one of 63 dedicated IAC (DIAC). A Bluetooth device using the GIAC will receive responses from all other Bluetooth devices in the vicinity (assuming that their inquiry scan mode is enabled). Each DIAC is common for a dedicated group of Bluetooth devices. DIACs are used to discover Bluetooth devices with a common characteristic.

A Bluetooth device switches to *page* substate in order to create a connection to another device. Again, the paging device (the master) periodically transmits ID-packets on changing frequencies containing a *Device Access Code* (DAC). The DAC contains the lower address part (LAP) of the target device. A master can only connect slaves with an enabled page scan mode.

3. Data Collection Protocol

3.1. General remarks

A simple application of our protocol is a network consisting of one base station and several sensor nodes. Not all nodes have to be in communication range of the base station. Therefore, distant nodes have to use routing nodes to reach the base station. Role assignment, e.g. the selection of nodes as routing nodes, has to be repeated periodically to evenly distribute energy consumption among nodes.

If no events occur, it is sufficient for the base station to receive sensor information in a regular manner. However, if an event occurs, it is essential to transmit information about spatial and temporal characteristics of the event to the base station as fast as possible.

Our protocol can be divided into two phases. During set-up phase, the network is formed. Base station and sensor nodes explore their vicinity and search for neighboring nodes. At the end of the set-up phase each sensor node knows at least one *Packet Forward Address* (PFA). If a sensor node is not able to reach base station directly, it transmits its data to a PFA instead. The PFA then is responsible for data forwarding. When the set-up phase has finished, the network is formed and steady-state phase is started. During steady-state, sensor information is transmitted to the base station. After a certain time, the protocol enters set-up phase again and reorganizes the network. The reorganization interval depends on various network parameters, such as type and frequency of topology changes or the energy level of nodes.

Our protocol distinguishes sensor nodes according to their role in the network. Each node holds one of the following two roles. *Cluster members* gather sensor data and forward the data to a *cluster head*. Cluster heads collect sensor data from cluster members and forward it to the base station. The whole sensor networks then consists of a set of clusters; each cluster consists of a cluster head and at least one cluster member. Before cluster members join a cluster head, they are also referred to as *simple nodes.* As described above, a Bluetooth node can either be a master or a slave. A cluster head of our protocol is not necessarily a master and a cluster member not necessarily a slave. Our protocol is flexible by not stipulating master and slave roles.

3.2. Set-up phase

The set-up phase of our protocol, i.e. the formation of the network, is divided into seven steps. Depending on the size of the network, step 5 to step 7 have to be repeated. At the beginning of the set-up phase, all nodes have their inquiry scan mode and page scan mode enabled. Sensor nodes are categorized depending on their hop distance to the base station. For example, nodes within transmission range of the base station are 1-hop-distant nodes. Nodes within transmission range of 1-hop-distant nodes and not within transmission range of the base station are 2-hop-distant nodes, etc.

Initial value: $k = 1$ (hop count)

(1) Nodes decide themselves whether to become a cluster head or not by means of a cluster head selection algorithm. (Cluster head selection strategies are discussed in subsection 3.4.) Let p be the assumed cluster head probability. Then, $p*N$ nodes become a cluster head and $(1-p)*N$ nodes become a cluster member.

(2) The base station initiates an inquiry to discover all nodes in the vicinity.

(3) The base station successively creates connections to all sensor nodes having responded to the inquiry, i.e. *k*-hop-distant cluster heads and cluster members. Each of the connected nodes stores the base station's device address as packet forward address, whereas the base station maintains a table of discovered node addresses. Thereafter, all connections to the base station are closed.

(4) Each *k*-hop-distant sensor node then changes its visibility mode, so that it can not be discovered by inquiry scans (inquiry scan disable). After the completion of set-up phase, the inquiry scan mode is enabled again.

(5) Subsequently, each *k*-hop-distant cluster head initiates an inquiry. The changed visibility of *k*-hop-distant cluster heads prevents them from mutual discovery. The inquiry can yield the following responses:

    a. Cluster heads not discovered by (*k*-1)-hop-distant cluster heads (the 0-hop-distant cluster head is the base station)

    b. Cluster members in range

    If no cluster heads were discovered, proceed to step (7).

(6) Each *k*-hop-distant cluster head then successively creates connections to all (*k*+1)-hop-distant cluster heads that responded to the inquiry. Each (*k*+1)-hop-distant cluster head stores the device address of at least one *k*-hop-distant cluster head as packet forward address and subsequently disables its inquiry scan mode. Each *k*-hop-distant cluster head maintains a table of dependent (*k*+1)-hop-distant cluster heads.

(7) Each *k*-hop-distant cluster head forms a cluster with all discovered simple nodes as cluster members. Cluster members store the device address of their cluster head as packet forward address. The cluster head maintains a table of all cluster members. Notice that connections between cluster head and cluster members are only established on demand.

If step (6) was skipped the set-up phase is finished. Otherwise, increment *k* by 1 and go back to step (5).

3.3. Steady-state phase

When the set-up phase is finished and each node has received a packet forward address, the steady-state phase begins. In this phase, each cluster member periodically transmits its sensor data to its cluster head. The cluster head then forwards the collected data to the base station and uses other cluster heads as routing nodes if needed. The following steps are performed during steady-state phase.

(1) Each cluster member periodically transmits its sensor data to its cluster head.

(2) Cluster heads preprocess sensor data by means of a data fusion or compression algorithm. Hence, cluster heads have already eliminated redundant data and have furthermore reduced the amount of data being forwarded to the base station.

(3) Each cluster head transmits its aggregated sensor data to its PFA.

(4) If a cluster head receives aggregated sensor data from another cluster head, it forwards the data to its own PFA.

This description of the steady-state phase is generic. Subsection 3.7 contains details about the configuration of the steady-state phase.

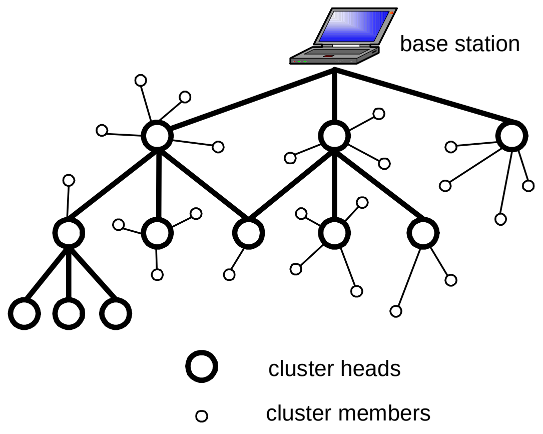

Fig. 1. Structure of a DCP sensor network

3.4. Cluster formation

The selection of cluster heads is not a trivial problem since cluster heads have to comply with several requirements. As an example, cluster heads have to be equipped with sufficient energy reserve in order to perform energy depleting routing and data fusion mechanisms. Moreover, cluster heads should be situated in the center of a cluster in order to minimize intra-cluster energy consumption. From an abstract point of view, each cluster is a group of sensor nodes with a certain assignment of roles. For such a group of nodes, Liu et al. introduce in [3] the term *collaboration group* (or *group* for short). Referring to Liu et al., each group is a set of entities, e.g. sensor nodes, that encapsulates two properties: *scope* and *structure*. The scope of a group defines its members, e.g. all sensor nodes within a certain range. A group's structure defines the roles each member plays in the group. Formally, a group is a 4-tuple

$$G = (A, L, p, R) \quad (1)$$

where A is the set of entities, L is the set of roles, $p : A \rightarrow L$ is a function that assigns each entity a role, $R \subseteq L \times L$ are the connectivity relations between roles. In the case of our protocol, two roles can be distinguished:

$$L = \{cluster\ head,\ cluster\ member\} \quad (2)$$

Additionally, one connectivity relation exists:

$$R = \{(cluster\ member,\ cluster\ head)\} \quad (3)$$

This relation describes, that each cluster member transmits its sensor data to a cluster head. The set of sensor nodes belonging to a cluster is the set of entities forming group *A*. Consequently, the whole sensor network consists of a set of groups (clusters). For a complete determination of the 4-tuple, each entity (sensor node) has to be assigned a role, i.e. *p* has to be defined. Two strategies are possible:

- Roles are assigned before the network is partitioned into groups.
- First, the network is partitioned into groups, and then roles are assigned.

The first strategy is suitable for sensor nodes without „global view" of the network, i.e. sensor nodes that only know their neighboring nodes. With this strategy, each sensor node has to decide autonomously if it becomes a cluster head or not. Non-cluster head nodes then decide which cluster head they join. The second strategy has the advantage of a network partitioning independent of a role assignment. If the role assignment gets lost, network partitioning persists.

Existing clustering protocols for wireless sensor networks mostly use the first strategy: roles are assigned before the network is grouped. As an example, in [4], Heinzelman et al. describe a communication protocol for wireless sensor networks, called LEACH (Low-Energy Adaptive Clustering Hierarchy), where each sensor node determines autonomously a random number between 0 and 1 and compares it to a predefined threshold value. If the random number is less than the threshold the node becomes a cluster head, otherwise not. This pure stochastic selection of cluster heads can be improved by incorporating the energy level of each sensor node into the selection algorithm as described in [5]. In the case of our Data Collection Protocol, cluster heads are stochastically determined similar to the algorithms described in [4] and [5]. Each sensor node determines a random number and compares it to a predefined threshold.

When the role assignment is finished, simple nodes have to join a cluster head and become cluster members in order to complete group forming. However, which cluster head does a simple node choose if more than one is in range? A simple strategy would be to choose the cluster head with the shortest distance. Thus, communication's energy consumption could be minimized. (This assertion implies that the Euclidean distance between two sensor nodes is proportional to the energy consumed by a data transfer between these both nodes. For RF communication, effects such as reflection, interference or an obstructed environment can sophisticate this relation.)

The energy-optimal assignment of simple nodes to cluster heads corresponds to the geometrical construction of a Voronoi diagram. A Voronoi diagram divides a plane into regions. Each region contains sensor nodes closer to a given cluster head than to any other cluster head. The distributed construction of an exact Voronoi diagram is very complex and requires a lot of communication among sensor nodes [6]. Algorithms for an approximate construction of Voronoi diagrams that require less computation and communication are proposed in [6] and [7].

However, there is another way to construct Voronoi diagrams in wireless sensor networks without a global view of the network. Assume each sensor node to be able to measure the strength of a received signal and each cluster head transmits with the same output power. Moreover, assume the strength of a received signal to be inversely proportional to the distance between two nodes. Consequently, each simple node has to join the cluster head with the maximum signal strength. Our Data Collection Protocol benefits from this idea as described in the following subsection.

3.5. Behavior of the base station

As described in Subsection 3.2, at the beginning of the set-up phase the base station attempts to discover sensor nodes in the vicinity. Two strategies for the base station's behavior are possible:

- The base station can act in „pure mode", connecting only cluster heads in the vicinity.
- The base station can act as cluster head, connecting both discovered cluster heads and simple nodes. The base station forms a cluster with discovered simple nodes.

Table 1. Basic Time for Connection Establishment, Data Transfer, And Disconnection Of Selected Bluetooth USB-Modules

Bluetooth-USB-Module (chip set)	Allnet 1572 (Broadcom)	Epox BT-DG02 (CSR)	Aiptek BT-USB (Transilica)
T_g (s)	4.40	4.46	4.47
ready after 6s	99 %	100 %	97%

We use the second strategy for DCP. Thus, cluster heads situated near the base station are relieved.

When the base station's inquiry is finished, it connects to all discovered cluster heads (step 3). For a synchronous protocol cycle, 1-hop-distant cluster heads should defer their own inquiry until step 3 and 4 are completed. Otherwise, a cluster head that received its PFA from the base station would immediately disable its inquiry scan mode and start an inquiry although step 3 and 4 are still running.

We can avoid this behavior by providing each 1-hop-distant cluster head with a delay time. After a cluster head received its PFA from the base station, it waits for this delay time before it starts the inquiry. Assume the base station to connect to cluster heads successively. The delay time T_c of cluster head c then is calculated by

$$T_c = T_g \cdot r \qquad (4)$$

where T_g is the *basic time*, needed for connection establishment, data transfer (PFA) and r is number of 1-hop-distant cluster heads not yet connected by the base station. According to our experiments, the basic time T_g is not larger than 5 seconds for commercial Bluetooth USB modules (Table 1).

3.6. Multipath routing

Consider the case where a sensor node receives multiple PFAs since more than one cluster head connects with this node. For our protocol, such a behavior is advantageous and, therefore, explicitly welcomed. It increases the robustness and efficiency of our protocol. On the one hand, a multipath-routing scheme can be applied. If a cluster head fails depending nodes choose another PFA. On the other hand, sensor nodes with multiple PFAs can choose the one with the shortest distance in order to reduce energy consumption.

If a connection is established between two Bluetooth devices, each of them can measure a RSSI-value of the connection (RSSI=Received Signal Strength Indicator). RSSI measures the signal strength of a received signal [2]. The shorter the distance between two Bluetooth devices the larger the RSSI-value. Multiple PFAs allow a sensor node to choose the one with the largest RSSI value. Hence, the sensor node can adapt its output-power to the shorter distance and thus save energy. Notice that many commercially available Bluetooth devices adapt output power automatically and not controllable by an application. However, a RSSI value is useless if we don't know the corresponding output power. Consequently, the output power of Bluetooth-based sensor nodes must be controllable by the sensor node application.

Our protocol does not maintain connections during steady-state phase. When a data transfer is finished the nodes disconnect immediately. The advantages of this behavior are:

- Clusters are not limited to piconet size. A cluster can consist of more than seven cluster members.
- Assuming a low data transfer rate, energy can be saved if nodes disconnect immediately after data transfers.
- If all connections in our sensor network were maintained during steady-state phase a large scatternet consisting of many piconets would be formed. Although Bluetooth uses frequency hopping, too many piconets would interfere each other [8].

However, delay times for our protocol are higher than for a connected piconet. Alternatively, cluster heads or network regions with temporary high data rates could autonomously decide to maintain a piconet or local scatternet.

3.7. Configuring the steady state phase

Cluster heads receive sensor information from their cluster members. A cluster covers a small region of the sensor network. Thus, different cluster members might likely provide their cluster head with equal or similar sensor measurements. In order to avoid redundancy, cluster heads should compress or fuse received data. As an example, consider a sensor network deployed to monitor temperature in a certain region. If all members of a cluster provide the cluster head with an identical temperature value, the cluster head can fuse the data. Instead of transmitting the same temperature value repeatedly, the cluster head forwards only one „cluster temperature“.

Furthermore, sensor data can be refrained by cluster heads for several acquisition periods. This procedure reduces the overhead in terms of time and energy consumption. Shih et al. showed in [9] that at small packet sizes, energy consumption of an RF-transceiver is dominated by the startup transient and not by the active transmit time. Similar to energy consumption, the time needed to transmit a single bit depends on packet size. Obviously, for packet sizes less than 500 kb, average time per bit is much larger than for bigger packets. Thus, a temporary and spatial aggregation of sensor information is strongly recommended. However, in the case of a critical event in the monitored region, sensor information has to be forwarded immediately. Fig. 2 illustrates the effect of start-up time.

Alternatively, our protocol can be used as underlying layer of a sensor query service. With this service, the base station itself can query single sensor nodes or clusters. Madden et al. describe an interesting approach in [10] called TAG (Tiny AGgregation Service), where the base station sends out SQL-like queries into the network and then waits for the requested data. TAG can be implemented on top of our Data Collection Protocol, allowing a flexible sensor data acquisition.

4. Simulation results

Based on detailed simulations, a first impression of our protocol's performance can be gained. For the simulations, we use a self-developed simulation tool that models protocol behavior depending on various system parameters. In Fig. 3 to Fig. 6, several parameters are determined for various topologies. In the legends of the figures six different cases are specified. The parameters in squared brackets are node density and node's communication range in meters. Node density describes the number of nodes/m². For node density 0.01, network dimensions are 100m*100m. 100 Nodes are randomly distributed over this area. The base station is located in the center of the network at position (50m, 50m). For node density 0.04, 100 sensor nodes are distributed over an area of 50m*50m. The base station is located at position (25m, 25m). For all presented experiments, cluster head probability is varied between 0 and 1 in steps of 0.1. Each cluster head probability is examined in 100 simulation runs whereupon average values and standard deviation are calculated.

Fig. 3 depicts the maximum number of hops to base station, i.e. the longest path of the network. Notice that the longest path is shorter for large communication ranges, which is not surprising. Moreover, the maximum number of hops is not strictly monotonic increasing if cluster head probability is incremented. Except for the combination [0.01; 10], a maximum can be found for every curve. As an example, the input parameter combination [0.04; 15] shows a maximum at cluster head probability 0.2. A further increment of cluster head probability does not result in longer paths.

Fig. 4 illustrates the average number of hops to base station, i.e. the length of the average path. Again this figure reveals for each combination of input parameters a maximum. Thus, for each combination of input parameters, an upper bound for cluster head probability can be found. If we increment cluster head probability further, the average and longest paths will not increase.

Fig. 5 depicts the percentage of unconnected nodes after set-up phase. Notice that input parameter combinations [0.01; 10] and [0.01; 15] are not usable, since even for high cluster head probabilities, the percentage of unconnected nodes is extremely high. The input parameter combinations [0.01; 20] and [0.04; 10] are two-edged cases. A satisfying percentage of unconnected nodes is only accomplished with a large cluster head probability. However, for cluster head probabilities larger than 0.3, clusters are too small to operate efficiently. As an example, a cluster head probability of 0.5 results in clusters with one cluster head and one cluster member at 100% connectivity.

Fig. 6 illustrates the number of nodes that directly depend on a cluster head. This includes both cluster members and directly dependent cluster heads that forward data to the examined cluster head. To summarize, for an efficient operation of our Data Collection Protocol, input parameters have to be chosen wisely. Cluster head probability should be used as instrument for fine tuning only since a variation of this parameter in regions larger than 0.3 leads to inefficient mini-clusters.

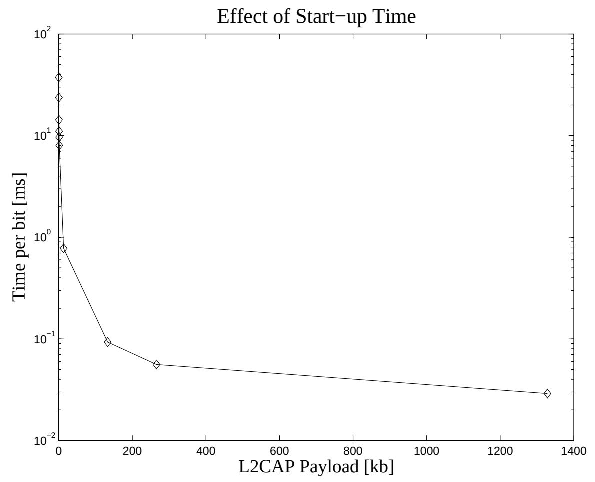

Fig. 2. Effect of startup-time for Bluetooth transceivers (Baseband Packet Type: DM1)

5. Related work

A wireless sensor network is a special case of an ad hoc network. Every protocol for sensor networks has therefore to be compared to existing protocols from the ad hoc domain. In the case of the network layer, reactive and proactive protocols can be distinguished [11]. Our Data Collection Protocol works proactively since routes are discovered during set-up phase. However, ad hoc routing protocols are not suitable for our intended applications of sensor networks because they are optimized for multi-directional data transfers. DCP is mainly designed for unidirectional data transfers. In our intended sensor network applications, data transfers are mostly directed to the base station. For ad hoc routing protocols, communication

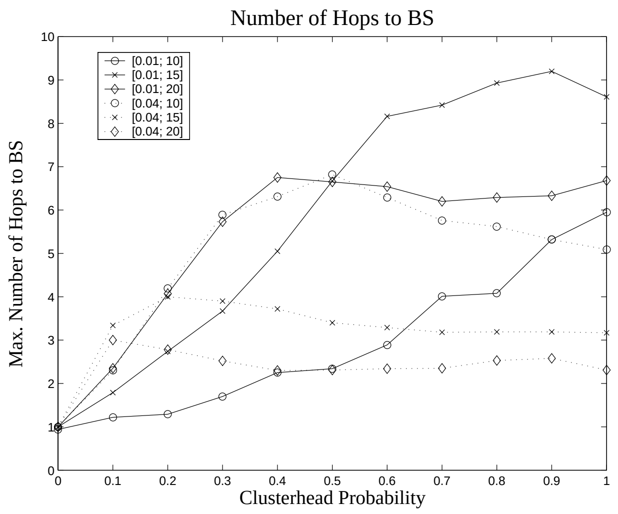

Fig. 3. Maximum number of hops to base station. In squared brackets: [node density; node's communication range].

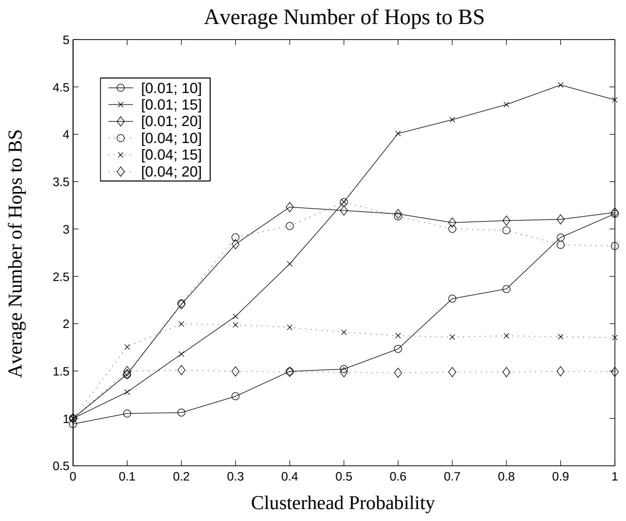

Fig. 4. Average number of hops to base station. In squared brackets [node density; node's communication range].

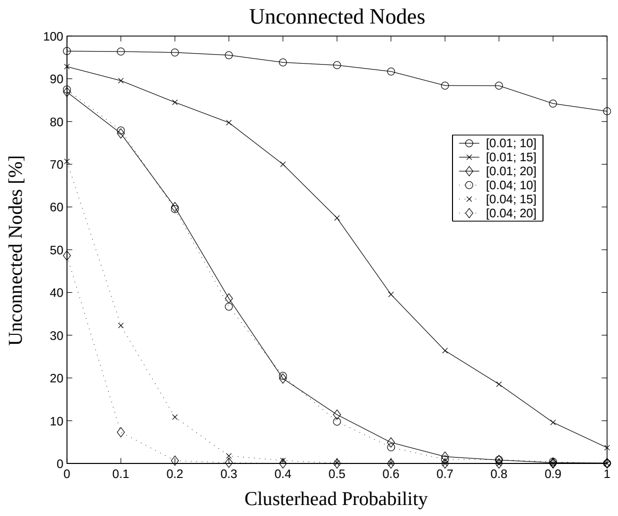

Fig. 5. Percentage of unconnected nodes in a network. In squared brackets [node density; node's communication range].

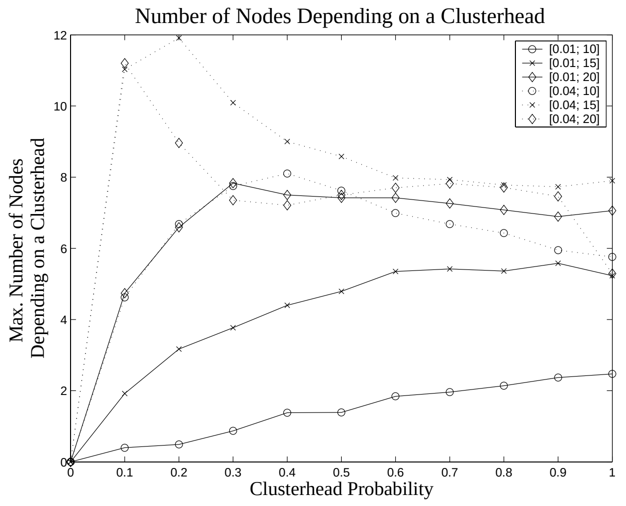

Fig. 6. Number of nodes depending on a cluster head. In squared brackets [node density; node's communication range]

among nodes is more important. Therefore, ad hoc routing protocols do not require a base station. Additionally, ad hoc routing protocols are optimized for networks with highly mobile nodes; routes have to be updated more frequently.

There is one characteristic of ad hoc protocols that disqualifies them from an adoption for Bluetooth-based sensor networks. Ad hoc routing protocols mostly require a broadcast mechanism. However, Bluetooth's broadcast support is limited. A general broadcast to all nodes in communication range is not possible. Only *Page* and *Inquiry* are similar to a broadcast, though transmitted ID-packets may not contain payload data. The master of a Bluetooth piconet can initiate a piconet broadcast. This requires a connection between master and all slaves.

Although our protocol does not depend on scatternets, formation protocols for scatternets might be an alternative. A comprehensive comparison of scatternet formation protocols is given in [12]. As described above, scatternets are overlapping piconets where some of the nodes are members of more than one piconet. Algorithms for scatternet formation are optimized for a high connectivity and maintain connections during operation. This results in shorter delay times but also in higher energy consumption. Many scatternet formation protocols require a *device discovery* procedure before scatternets can be formed. During device discovery, each node performs an inquiry to discover all other nodes in communication range. This knowledge should be symmetric, i.e. if node *u* discovers node *v* the device discovery procedure has to ensure that node *v* discovers node *u*, too. Scatternet formation protocols ensure this by periodically switching each node's state from *inquiry* to *inquiry scan* and back. Hence, device discovery can take a long time and has to be repeated when the topology changes.

Scatternet formation protocols are mostly organized in a decentralized manner, whereas BlueTrees creates a tree-like infrastructure similar to our protocol [13]. In order to limit the number of slaves in a piconet to seven, BlueTrees has to perform an algorithm called „branch reorganization“. In contrast to scatternet formation protocols, our DCP allows clusters containing more than seven members. Furthermore, a disconnection after data transfer is for our intended applications (sporadic data transfers) more energy efficient.

Existing communication protocols for wireless sensor networks might also be suitable for Bluetooth-based sensor nodes. One representative of this group is LEACH (Low-Energy Adaptive Clustering Hierarchy) [4]. LEACH cannot be adopted for Bluetooth-based sensor nodes since it requires a broadcast mechanism and a hybrid MAC layer (a combination of CSMA, TDMA, and CDMA).

6. Conclusion

In conclusion, we introduced DCP, a scalable and robust protocol for energy-efficient data collection in large Bluetooth-based sensor networks. DCP's main advantage is independence from Bluetooth piconet and scatternet limitations. This yields a more flexible and scalable network infrastructure. A periodically repeated role assignment within the network leads to a uniform distribution of energy consumption. Additionally, energy consumption is reduced by a technique that only maintains connections between sensor nodes if data transfers occur. With these characteristics, DCP enables a wide range of sensor network applications with inexpensive hardware.

7. Acknowledgments

This work is supported in part by Gottlieb Daimler-and Karl Benz-Foundation, Ladenburg. We would like to thank Rico Möckel who developed the simulation environment and carried out the simulations.

8. References

[1] M. Leopold, M. Dydensborg and P. Bonnet. Bluetooth and sensor networks: a reality check, *Proceedings of SenSys 2003*, November 2003.

[2] Specification of the bluetooth system, Version 1.1, February 2001.

[3] J. Liu, et al. State-centric programming for sensor-actuator network systems, *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 50-62, October-December 2003.

[4] W. Heinzelman, A. Chandrakasan and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks, *Proc. of the 33rd International Conference on System Sciences (HICSS '00)*, January 2000.

[5] M. Handy, M. Haase and D. Timmermann. Low-energy adaptive clustering hierarchy with deterministic cluster-head selection, *Proc. of IEEE Int. Conference on Mobile and Wireless Communications Networks*, Stockholm, September 2002.

[6] X. Li, P. Wan and O. Frieder. Coverage in wireless ad hoc sensor networks, *IEEE Transactions on Computers*, vol. 52, no. 6, pp. 753-763, June 2003.

[7] Z. Butler and D. Rus. Event-based motion control for mobile sensor networks, *IEEE Pervasive Computing*, vol. 2, no. 4 pp. 34-42, October-December 2003.

[8] J. Bray. How many bluetooth piconets fit in a room?, URL: http://www.informit.com, Prentice Hall PTR, May 2001.

[9] E. Shih, et al. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks, *Proc. of ACM SIGMOBILE*, Rome, July 2001.

[10] S. Madden, M. Franklin, J. Hellerstein and Wei Hong. TAG: a tiny aggregation service for ad-hoc sensor networks, *Proc. of the 5th Symposium on Operating Systems Design and Implementation*, December 2002.

[11] C. Perkins, Ad hoc networking, Addison-Wesley, December 2000.

[12] S. Basagni, R. Bruno and C. Petrioli. A performance comparison of scatternet formation protocols for networks of bluetooth devices, *Proc. of the 1st IEEE International Conference on Pervasive Computing and Communications*, Dallas, March 2003.

[13] G. Záruba, S. Basagni and I. Chlamtac. BlueTrees – scatternet formation to enable bluetooth-based personal area networks. In *Proc. of IEEE Int. Conference on Communications*, Helsinki, June 2001.