Efficient Slot Assignment for the Many-to-One Routing Pattern in Sensor Networks

Volker Turau, Christoph Weyer, and Christian Renner
Institute of Telematics
Hamburg University of Technology
Schwarzenbergstrasse 95, 21073 Hamburg, Germany
Email: turau@tuhh.de

***Abstract*—In this paper an efficient TDMA slot assignment for the many-to-one routing pattern in wireless sensor networks is presented and analyzed. This assignment scheme is based on a simple to implement heuristic, named spatial path-based reuse (SPR). By considering several metrics, the superiority of this heuristic with respect to energy consumption and runtime compared to proposals found in the literature is demonstrated.**

I. Introduction

There is a substantial interest in the design of the many-to-one routing pattern in sensor networks. This pattern naturally arises in situations, where all nodes need to report their sensor readings to a central location called sink. In many cases a reliable implementation of this pattern is required, i.e., all readings must reach the sink. Reliability in harsh environments is a challenging task for several independent reasons. One problem is rooted in wireless communication: high loss rates, asymmetric links, weak correlation between quality and distance, and hidden terminal problems. The other main problem comes from the constrained resources of sensor nodes: limited energy supply, small computational power and memory space, and narrow communication bandwidth. Consequently, nodes cannot buffer large amounts of data nor implement complex compression algorithms.

This paper considers a special variant of the many-to-one routing pattern, where all nodes have stored a certain amount of data packets that need to be routed reliably via a routing tree towards a sink. The main objectives of the current work are to minimize energy consumption and total completion time. The ease of implementing the solution is also considered. It is assumed, that nodes have limited memory for storing packets in transit, but also for necessities such as maintaining neighborhood tables. Data compression is not considered in this paper, it is regarded as an orthogonal approach that can be combined with this work.

The major sources of energy waste in wireless sensor networks that have been identified in the pertinent literature are idle listening, collisions and control packet overhead. It is well known that schedule-based MAC protocols effectively reduce the first two issues. Therefore, Time Division Multiple Access (TDMA) is frequently considered. To provide a reliable service such MAC protocols are also favored over contention based MAC protocols.

The main challenge of current research into TDMA protocols is to devise a simple, distributed algorithm for an interference-free time-slot allocation. This paper analysis the suitability of different slot assignment algorithms for the many-to-one routing pattern. In particular it is shown, that classical schemes that assign a single slot to each node are not the best choice in this case. The main contribution of the paper is a new slot assignment scheme, called *spatial path-based reuse* (SPR), dedicated to the needs of the many-to-one pattern. The new scheme is compared with another scheme using extensive simulations, which also account for the limited resources of sensor nodes. These simulations yield that using this scheme allows for a faster and more energy-efficient solution for the many-to-one routing problem under consideration. The paper also contains recommendations for the parameterization of the scheme with respect to the size of the network, the average node degree and other practical aspects of wireless sensor networks.

The rest of this paper is organized as follows: Section II presents the state of the art in TDMA protocols for tree routing and Section III discusses the related issues. The following section contains the main contribution of the paper, the slot assignment scheme SPR for the many-to-one routing pattern. Section V evaluates the scheme using simulations.

II. State of the Art

The many-to-one routing problem solves the most significant traffic pattern in WSN that is data gathering from sensor nodes of a network. It is well known that this problem is best supported by a tree structure: a data gathering or routing tree. Flows in this tree are unidirectional from sensor nodes to a sink. All nodes except the sink forward any packets they receive to the next hop, i.e., to the parent. Dozer [1] and LUSTER [2] are two environmental monitoring systems making use of this pattern. Both approaches are using a routing tree and a TDMA schedule for forwarding the measurements in an energy-efficient way.

The communication topology of a wireless sensor network can be described as a graph $G = (V, E)$, where $V = \{v_0, \ldots, v_n\}$ are the nodes of the network and E contains a tuple (v_i, v_j) if both nodes are in communication range r_{com} of each other and a reliable bidirectional communication link exists between these nodes. The distance between two nodes

v_i and v_j is denoted as $d_{i,j}$. The data is forwarded along the routing tree $T \subseteq E$, which is rooted in the sink v_0.

TDMA protocols differ in whether they use a static or a dynamic slot schedule. A dynamic slot schedule usually comes with a high overhead in the form of additional control packets. The dynamics they provide is not needed for periodical data or predictable traffic. Thus, in these cases a static schedule is advantageous. The slot assignment of TDMA protocols that are used in routing trees is described by the link-scheduling problem. Each link $l_i \in T$ is assigned a slot $s_u \in S$ and $S = \{s_0, \ldots, s_k\}$, whereby slots can be spatially reused. The communication roles of the two nodes are known a priori from their position in the tree, since the data is sent from the children to the parents towards the sink. For each packet an acknowledgment is sent back to the corresponding child in the same slot. Therefore, each link is used in both directions. Hence, it is important to determine the possible interference between links.

Link-scheduling or slot assignment can be solved via coloring by using k-hop neighborhoods or interference knowledge. Most existing TDMA slot allocation algorithms are using k-hop neighborhood information such that a slot is uniquely used within k hops. In order to decrease complexity and memory consumption a 2-hop neighborhood is used in most cases [3], [4]. Available algorithms use different approaches concerning the order in which nodes get colored, as this order affects the total number of colors used. E.g., the Color Constrained Heuristic (CCH) [4] is based on the weighted sum of a node's already colored 1- and 2-hop neighbors. By recalculating this sum after each color assignment, the next node is determined.

The main problem of using a k-hop neighborhood is shown in Fig. 1. The link between node v_4 and v_2 causes interference, when nodes v_9 or v_6 are sending to v_5 (because they are in the interference range of each other). Therefore, the links between $(v_4, v_2), (v_9, v_5)$, and (v_6, v_5) must use different slots. The example shows that the 2-hop knowledge generates a schedule that leads to collisions. If the packet rate is high, consecutive collisions have the effect that parts of the tree get completely stalled. In general a k-hop neighborhood cannot provide an interference free schedule [5]. Furthermore, distributed computation within the k-hop neighborhood for larger values of k is not feasible.

Several different approaches for modeling interference in wireless communication exist. The Fixed Power Protocol Interference Model (fPrIM) introduced in [6] assumes that each node v_k has an interference range r_{int} so that any node v_j will be interfered during a reception from v_i by v_k if $d_{k,j} < r_{int}$. The ratio between interference and communication range $\gamma = \frac{r_{int}}{r_{com}}$ is often assumed as $2 \leq \gamma \leq 4$. The interference depicted in Fig. 1 with $\gamma = 2$ shows the large influence of this physical parameter for an interference-free schedule. But the main problem of this model is that the interference depends not only on the distance between v_k and v_j.

A more realistic model considers the difference between the interference signal (caused by v_k) and the resulting signal strength of the communication between v_i and v_j. The signal is sent with power P_i^t by node v_i and is damped by the path loss exponent α, so that the receiving power at v_j can be expressed as $P_{ij}^r = \frac{P_i^t}{{d_{i,j}}^\alpha}$. When the quotient of the received signals from v_i and v_k is below a given threshold λ, the packet from v_i cannot be received successfully. Therefore, it is important that in the given scenario each packet is acknowledged. However, both communications can be interfered by node v_k if at least one of the following equations is satisfied:

$$\frac{P_{ij}^r}{P_{kj}^r} < \lambda \quad or \quad \frac{P_{ji}^r}{P_{ki}^r} < \lambda$$

This model is used in NS-2, where λ is commonly set to 10, which depends on the sensitivity of the used radio transceiver. In [7], [8] the dependency between the positions of sender, receiver, and jammer are investigated by field tests. The used model is a simplification of the physical interference model that considers the Signal-To-Interference-And-Noise-Ratio (SINR) that accumulates all interference at the receiver side [6].

Using interference for assigning slots to links is a complex task. The radio interference detection protocol (RID) described in [7] uses high-power communication to estimate the interference during the data transfer in low-power mode. This assumes that the transmitting power can be adjusted to simulate the interference radius, which is difficult due to the spatial irregularities of the signal strength. The interference values are shared within the interference area. Storing the SINR for all nodes in that area does not scale with network density. Another approach [9] uses a special random access slot at the beginning of each round for competition for slot acquirement depending on the depth in the tree. The number of data slots is fixed a priori and must be configured globally. Therefore, slot acquirement is delayed by the ratio between data slots and the random access slot since only these slots can be used for slot assignment.

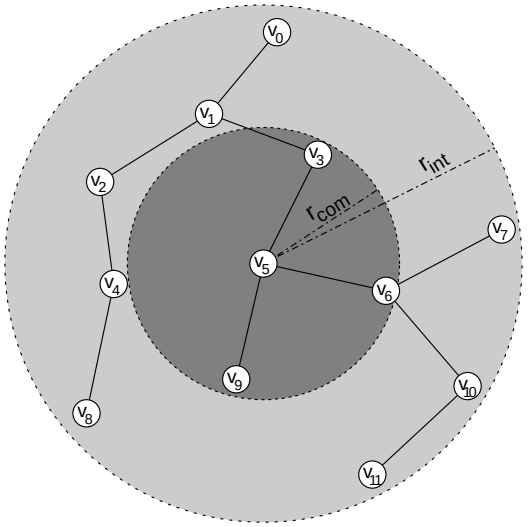


Fig. 1. Interference radius (r_{int}) vs communication radius (r_{com})

III. TDMA for Tree-based Routing

In this section the different challenges of using TDMA in the many-to-one routing problem for data-gathering in wireless sensor networks are discussed. Given the problem at hand, TDMA schemes with static slot schedules are regarded as the first choice in order to achieve reliability and energy-efficiency. However, finding an efficient schedule for this task is still an open issue, as has been outlined in Section II.

The first step in many-to-one routing is the construction of a routing tree rooted in the sink. Zhou et al. [10] discuss different procedures to build such trees. The disadvantage of their approaches is, that they have no means to limit the number of children of a node in the tree. Considering the memory limitations of the sensor nodes, this is an imperative. Therefore, an appropriate routing tree construction method is needed. The second step is the static assignment of the time slots to the links. After that the routing of packets from nodes to the sink can start.

A. Channel Congestion

State-of-the-art slot assignment algorithms optimized to generate a minimum total number of slots are likely to produce collision-afflicted TDMA schedules. As shown in Section II, this is because k-hop neighborhoods do not carry any direct information about the interference radius r_{int}.

In order to analyze the influence of this deficiency, we have implemented the CCH slot assignment algorithm. Because in our application links are used for bidirectional communication, slots have been uniquely assigned to links of T within 3 hops. Simulation results have revealed that the obtained slot assignments produce a considerable amount of collisions. A detailed analysis of this can be found in Section V-B.

In consideration of these findings, a comparison of our approach using this assignment method is impossible. As stated in Section II, increasing the number of hops is not a feasible solution in terms of storage space, and choosing a different k is difficult, since k obviously depends on the network density and node placement.

As a result, we decided to use a 2-hop, CCH-based coloring algorithm provided with global knowledge, i.e., the interference radius $r_{int} = \gamma r_{com}$. This new algorithm firstly computes the interference neighborhood from the given topology, r_{com}, and γ. Secondly, it generates a unique coloring of nodes within 2 hops and finally assigns slots to the links of T by using a child's color for its link to the father. Of course, this approach produces a slightly higher total number of slots (on average 1.5 as many), but it leads to completely collision-free TDMA schedules. However, it is practically impossible to implement this assignment algorithm in a real network due to the exploitation of global knowledge (i.e., node position) and general assumptions (e.g., choice of γ). Hence, our schedules are compared against a potential optimal solution.

B. Buffer Congestion

Tree routing based on a TDMA scheme, where each node sends a single packet during each round, directly leads to buffer congestion: during a round each node transmits a single packet while at the same time it receives a packet from every child. Thus, nodes close to the sink are likely to become bottlenecks since they have to transmit packets generated locally as well as those generated by all downstream nodes. Hence, TDMA protocols of this type do not avoid the buffer congestion problem.

A remedy for the buffer congestion problem is to explicitly coordinate the sending of packets by sensor nodes such that, over a period of time, the sink receives equal amounts of data from each node. To meet this end, each node must forward on the average as many packets as it receives during each round. Rate-based congestion control using the traditional AIMD approach (additive increase multiplicative decrease) relies on periodic rate adjustment. But these approaches suffer from complicated signaling traffic increasing the packet overhead [11].

TDMA schemes suitable for the many-to-one pattern do not equally distribute slots over all nodes, but preferably assign slots to nodes near the sink. Furthermore, more than one slot is assigned per node and round. TDMA schemes with these properties have been proposed in [12]. It was shown, that they considerably reduce energy consumption and total completion time. The main drawback of these algorithms is that the administration of a large number of assigned slots is complex and requires storage proportional to the number of assigned slots. This number can be in the order of the network size, making this approach feasible only for very small networks.

C. Reliability

To provide for the required reliability a scheme with hop-by-hop acknowledgments was employed. A node successfully receiving a packet sends within the same time slot an acknowledgment to the sender. Upon receiving such an acknowledgment the child can remove the data packet from its buffer. If no acknowledgment is received the packet is resent in the next assigned slot. The mechanism of acknowledgments is used to carry information from a parent to a child. In particular this is used to signal buffer congestion. If a node receives a packet from a child, it checks its available memory. In case the fill level of the buffer is above a specified limit, it advises the child with the acknowledgment to suspend sending for a specific number of slots. Unless the node's buffer is completely full, the node accepts the packet. When the buffer is full, the node sets a flag in the acknowledgment. If the child receives this acknowledgment the packet is not removed from the buffer, instead it is resent at a later time.

D. Energy Efficiency

The situation that a child keeps sending packets to its parent, despite a permanent interruption of the communication link or a node failure, must be avoided. A simple solution is to regard a link as permanently interrupted, if a node does not receive a packet or an acknowledgment for a given number of consecutive slots assigned to that link. The node will then stop

sending packets or stop listening and switch off its transceiver during these slots.

In the used TDMA scheme, nodes always start sending new packets at the beginning of a slot. In addition, acknowledgments are sent as an immediate response to a received data packet. Therefore, a listening node switches off its transceiver, if it does not detect a packet or an acknowledgment within a specified time limit. To allow for synchronization errors and propagation delays, we have set this limit to 10% the length of a slot.

IV. Our Approach

In consequence of the findings reported in Section III-A, we have developed the SPR slot assignment, a light-weight heuristic that does not explicitly rely on any neighborhood information, but on the routing tree only. The basic idea is to regard the routing tree as an overlay of the paths from the sink to each leaf. Each of those paths is handled separately during the slot assignment procedure and the assigned sets of slots are pairwise disjoint. The final slot assignment consists of the union of the individual slots. Thus, if a node v is on p_v paths, it will get assigned p_v slots. Note that p_v equals the number of leafs among the predecessors of v. This approach restricts the spatial reuse of slots to nodes that are on a common path.

In the following the basic principle will be introduced, the details of the final algorithm will be presented later. In its basic form we allot fixed sets of κ slots to each path from a leaf to the sink. Since the sets of slots do not overlap, inter-path collisions are completely avoided. The slot assignment starts at the sink and slots are reused after every κ hops on a path. By reusing slots on a path, the total number of slots is reduced. Yet, reusing slots may cause intra-path collisions, which can be avoided by choosing κ carefully in context of the network density and depth of the routing tree.

In general, if κ is equal to the length of the path, then trivially no intra-path collisions can occur. Thus, for the example depicted in Fig. 1, κ need not be larger than 6. However, with v_0 being the sink, $\kappa = 3$ will most likely lead to collisions at v_5, as v_1 and v_9 would share the same slot and v_5 is within the interference radius of v_1. A less predictable situation arises, in case a path bends around a void. If, e.g., v_8 were the sink, v_4 and v_9 would interfere at v_8, although κ were as high as 5. Albeit this phenomenon cannot be completely prevented, simulation has shown that it is likely in sparse topologies only.

Our approach can be efficiently implemented using the information gathered during tree construction. With low effort only, this method equips every non-leaf node in the network with exactly as many sending slots as it has receiving slots. This has an outstanding advantage: Buffer usage is considerably reduced, as between every pair of sending slots there is only a single receiving slot. In the absence of packet loss, this implies that buffer usage will never exceed the initial fill level by more than one. Even if there is packet loss, buffer fill levels will not increase rapidly, as the balance between actually incoming and outgoing packets can be expected to vary only slightly over time.

Even so a node gets assigned as many slots as there are leafs among its children, there is no need to explicitly store these slots. The following slot assignment procedure guarantees, that each node v may use p_v slots with distance κ beginning with slot s_v^0.

- Let $\{P_0, \ldots, P_M\}$ be the set of all paths from the sink to the leafs ordered according to a depth-first search.
- Assign to the nodes of P_i beginning at the sink the slots $i\kappa, i\kappa+1, \ldots, i\kappa+\kappa-1, i\kappa, i\kappa+1, \ldots$

Thus, a node v can calculate the sequence of its slots from the smallest slot s_v^0, p_v, and κ. However, in order to identify interrupted links as described in III-D, v has to map its receiving slots to its c children. This can be achieved by v storing the number of paths p_j its child labeled j is on; ordered according to the P_i. Finally, v can calculate the slots of its children. Because p_v is the sum of the p_j, it does not have to be stored explicitly.

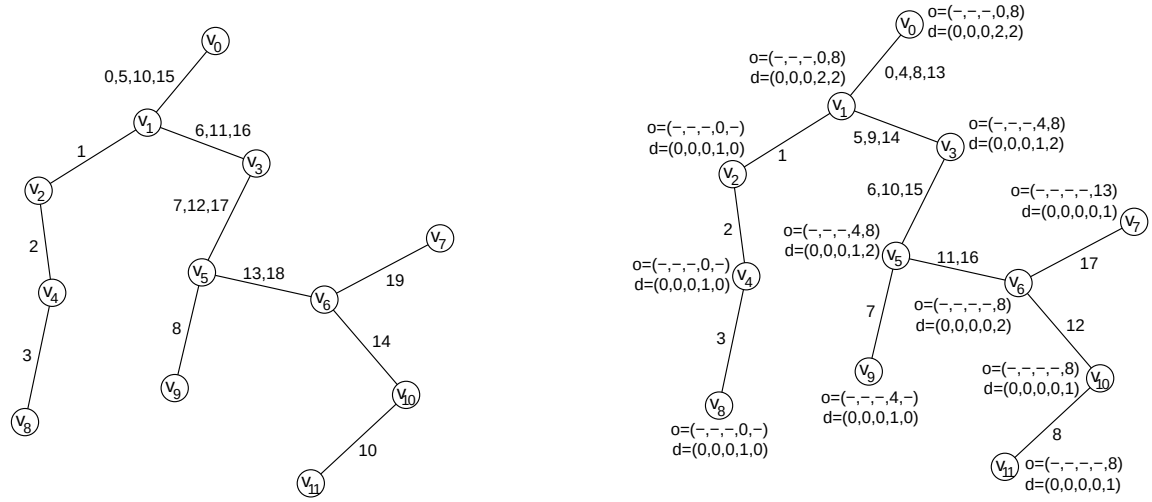
This distribution scheme exhibits a disadvantage. When there are paths having less than $\kappa + 1$ nodes, some slots are not used; e.g., the path from the sink v_0 to v_8 in Fig. 2(a) has length 4, while $\kappa = 5$. Even so, this will not lead to an increase in energy consumption, but it will prolong the total completion time.

A simple remedy would be the following. If a path P_i has less than $\kappa + 1$ nodes, unused slots can be used for the next path P_{i+1}. Consequently, the sets of slots assigned to a path no longer have the form $\{i\kappa, i\kappa+1, \ldots, i\kappa+\kappa-1\}$, and the slots of a single node no longer have the constant displacement κ. A detailed analysis of the slot displacements shows, that they are not completely random, but depend on the number of leafs in the subtree on the different depth levels of the routing tree.

In order to assign only i slots to paths with length $i < \kappa$, every node v calculates its displacement vector $\mathbf{d}[1], \ldots, \mathbf{d}[\kappa]$ as follows: $\mathbf{d}[i]$ is the number of leafs in its subtree at depth i of T if $i < \kappa$. For $i = \kappa$ the vector denotes the number of leafs in the subtree at depth κ or higher. Each node needs to store the displacement vectors of its children along with a slot offset for each entry in $\mathbf{d}$. Those offsets are required, because paths are firstly ordering by length and secondly in a depth-first equivalent as before. As will be shown in the following section, this information together with the length of the round and the depth of the nodes will be sufficient to compute the slots assigned to a node. The space required for this information only depends on κ and the number of children a node has.

A. Effective Implementation

As our slot assignment relies on the routing tree, we developed a three-step breadth-first search. Firstly, the sink starts generating the tree T via a breadth-first search. Here, the sink invites its neighbors in order of descending link-quality to become its children. If the sink has successfully added a given maximum number of children, or if no neighbors are left, its children proceed accordingly. Whenever a node is invited to



(a) Naive SPR slot assignment

(b) SPR slot assignment

Fig. 2. Example slot assignment ($\kappa = 5$)

become a child, it will only accept the first invitation and all others must be declined. Finally, the recursion terminates, after all nodes have been added to T.

Secondly, the leafs of the just constructed tree T reflect the construction wave up to the sink. During this contraction phase, every non-leaf node v in the network stores the displacement vectors $\mathbf{d}_0, \ldots, \mathbf{d}_{c-1}$ for each of its c children. Leaf nodes have no children and their own $\mathbf{d}$ is uniquely determined by their own depth in T. Finally, the sink acquires the displacement vectors of its children and thus its own. As a result, it gains knowledge about the round length R, i.e., the total number of slots, by means of its own displacement vector $\mathbf{d}$: $R = \sum_{i=1}^{\kappa} i\,\mathbf{d}[i]$.

Thirdly, the slots are actually assigned. Given its displacement vector $\mathbf{d}$, the sink calculates an offset vector $\mathbf{o}$, where $\mathbf{o}[i]$ represents the smallest of all slots belonging to a path with displacement i.

$$\mathbf{o}[1] = 0\,, \quad \mathbf{o}[i+1] = \mathbf{o}[i] + i\,\mathbf{d}[i] \quad (1 \le i \le \kappa - 1)$$

Next, the sink calculates the offset vectors $\mathbf{o}_0, \ldots, \mathbf{o}_{c-1}$ using $\mathbf{d}_0, \ldots, \mathbf{d}_{c-1}$ and $\mathbf{o}$.

$$\mathbf{o}_j[i] = \mathbf{o}[i] + i \sum_{m=0}^{j-1} \mathbf{d}_m[i] \quad (1 \le i \le \kappa,\ 0 \le j < c)$$

Here, the values $\mathbf{o}_j[i]$ represent the smallest of all slots belonging to a path with displacement i within the subtree of the child labeled j. Note that values $\mathbf{o}[i]$ and $\mathbf{o}_j[i]$ have no meaning, if $\mathbf{d}[i] = 0$ or $\mathbf{d}_j[i] = 0$, respectively.

The sink passes each $\mathbf{o}_j$ to the corresponding child. In addition, each child receives the total number of slots, which is equivalent to the round length. Every node v that receives an offset vector from its parent proceeds accordingly. First, it adopts the received offset vector as its own $\mathbf{o}$ and generates for each of its children a new offset vector $\mathbf{o}_j$ using the same method as the sink. Then, the total number of slots and the calculated $\mathbf{o}_j$ are sent to the child j of v. It suffices that v additionally stores $\mathbf{o}$, because it can always reconstruct the $\mathbf{o}_j$ from $\mathbf{o}$ and the $\mathbf{d}_j$, when needed. Finally, v can calculate its own slots by means of the $\mathbf{o}_j$ and $\mathbf{d}_j$. The algorithm terminates, if all leafs have received an offset vector.

B. Explicit Slot Calculation

Each node has explicit knowledge about its depth h in T, its own offset vector $\mathbf{o}$ and its childrens' displacement vectors $\mathbf{d}_j$. The vectors $\mathbf{o}_j$ and $\mathbf{d}$ can be calculated from those. While $\mathbf{d}$ is directly available on leaf nodes, inner nodes obtain $\mathbf{d}$ by

$$\mathbf{d}[i] = \sum_{j=0}^{c} \mathbf{d}_j[i] \qquad (1 \le i \le \kappa)$$

Putting these pieces of information together, a node can determine its own set of slots

$$S_{send} = \{\ s \ \mid\ 1 \le i \le \kappa,\ 0 \le e < \mathbf{d}[i] : \\ s = \mathbf{o}[i] + i\,e + (h-1) \bmod i\ \}$$

and the slot sets of its children

$$S^j_{recv} = \{\ s \ \mid\ 1 \le i \le \kappa,\ 0 \le e < \mathbf{d}_j[i] : \\ s = \mathbf{o}_j[i] + i\,e + h \bmod i\ \}$$

Note that the $\bmod$ operator is effective and thus required only for $i = \kappa$, because, by construction, paths with length $i < \kappa$ are assigned i slots only. In addition, the sink must not compute S_{send}, because there are no slots assigned to it.

C. Example

In order to illustrate the explicit slot calculation as outlayed in Section IV-B, Fig. 2(b) shows an example slot assignment for $\kappa = 5$. Displacement and offset vectors have been calculated as described in Section IV-A, invalid values of the latter are marked with dashes for clarity reasons.

Now say, v_5 wants to calculate the set of receiving slots $S^1_{recv} = \{s^1_0, s^1_1\}$ for its second child v_6. Using its own depth $h = 3$, all values $\mathbf{d}_1[i] > 0$, and the corresponding $\mathbf{o}_1[i]$, they evaluate to

$$s^1_0 = 8 + 5 \cdot 0 + 3 \bmod 5 = 11$$
$$s^1_1 = 8 + 5 \cdot 1 + 3 \bmod 5 = 16$$

D. Enhancements

Providing every node with the same amount of sending and receiving slots imposes a slight drawback. If the routing tree is not balanced, inner nodes will experience situations where the subtrees of one or more children have been emptied, while there exists at least one child with a non-empty subtree. In this case, there are less incoming packets per round than sending slots available. If the buffer of such a node runs empty, only a fraction of sending slots, i.e., the number of receiving slots used for the children with non-empty subtrees, is needed to forward incoming data packets. This leads to idle listing and unnecessary energy consumption, but there is a more severe problem.

Since a parent will give up receiving from a child, if no packet is received during a given number of consecutive slots, simply skipping slots during phases of temporarily empty buffers is not an option. Hence, a counter is included into the last packet, before the buffer may transitionally run empty. Its value will be set to the difference between the number of sending slots and receiving slots in use.

If the parent sees a packet with such a counter u, it will not expect packets within the next u sending slots of the corresponding child. To notify the child about having accepted this counter, it will be included into the acknowledgment. Only on reception of this acknowledgment, a child may skip u sending slots. Otherwise, it has to send a keep-alive packet, which may carry a new counter, if there is no packet available at the next sending slot. This procedure is mandatory, because the original packet may be lost. In this case, the child would skip slots, which the parent would not be aware of. As a result, the parent might accidentally give up receiving from that child. Due to the same issue, the parent must keep listening at the beginning of every slot of the child. Although it has just allowed the child to skip some slots, the parent does not know, whether its acknowledgment has been received. If it were lost, the child would keep sending, as its skipping request has no been agreed on, but the parent would not reply. Hence, the child might give up sending.

V. Simulation

The in this paper developed spatial path-based reuse (SPR) slot assignment scheme is extensively simulated and compared with the CCH $\gamma = 1.9$ slot assignment algorithm. For all simulations the NS-2 simulator is used. First the simulation setup is described in detail. After that the simulation results are presented. The section concludes with a discussion of results and recommendations for concrete networks.

	Interference [%] by Network Size				
Slot Assignment	100	300	500	700	900
CCH, 3-hop	7.9	9.2	9.9	10.0	10.3
CCH, $\gamma = 1.9$	0.0	0.0	0.0	0.0	0.0
SPR, $\kappa = 4$	5.0	5.7	5.9	5.7	6.0
SPR, $\kappa = 5$	0.9	0.7	0.6	0.6	0.7
SPR, $\kappa = 6$	0.4	0.2	0.1	0.2	0.2

TABLE I
AVERAGE PERCENTILE OF NODES SUFFERING FROM COLLISIONS IN NETWORKS WITH DENSITY $\rho = 6$

A. Simulation Setup

In order to avoid any influence of packet loss, we chose the two-ray ground propagation model with a communication radius of $r_{com} = 40\,m$. In addition, we implemented the interference model described in Section II. By setting $\lambda = 10$, we obtained $\gamma = 1.9$, which leads to $r_{int} = 76\,m$.

NS-2 has been used to simulate the data gathering process only. All topologies, routings trees, and slot assignments have been precalculated in advance, so that for each different slot assignment algorithm, the same topology and tree settings have been used. Hence, all results have been obtained under equal preconditions.

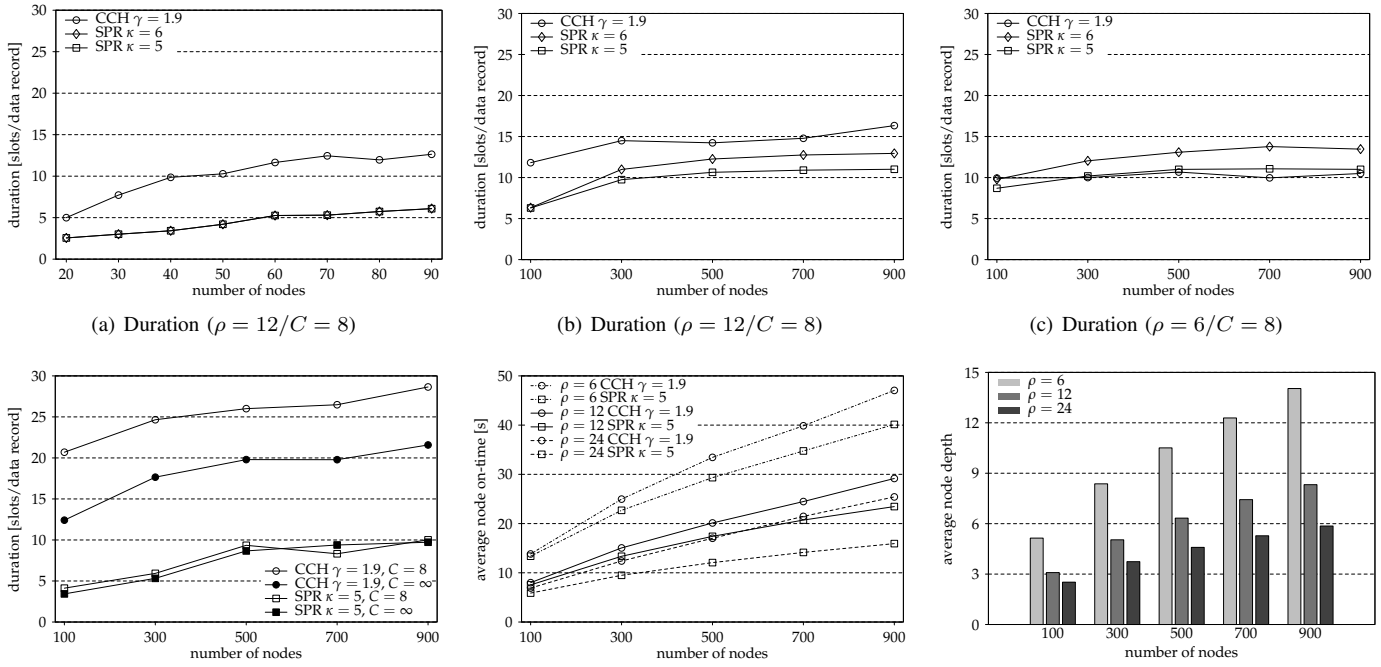
Network density is an important characteristic of a sensor network. In particular, it influences communication neighborhoods and thus the shape of the routing tree, i.e., the depth, the number of children per parent and the number of leafs. Additionally, density affects the total number of slots assigned. On account of this, we have developed a topology generator that can be configured to create topologies of a given density ρ. Here, ρ specifies the average number of nodes within an area of the size given by the communication range of a node. Using this generator, we have created topologies with 20 to 900 nodes and three different densities: $\rho = 6, 12$, and 20. For each combination of these parameters, 50 topologies have been generated.

Routing trees have been built by a bread-first search according to the scheme sketched in Section IV-A. Furthermore, two different trees have been built for every topology. In the first tree, the number of children per parent is unlimited ($C = \infty$). In the second tree, this number is restricted to $C = 8$, which is a trade-off between ensuring connectivity and saving memory.

Finally, buffers have been restricted to contain at most 200 packets to account for the constrained storage space available on sensor nodes. At the beginning of each experiment, every node in the network had the same amount of 40 packets in its buffer. Different initial fill levels have been simulated as well, without changing the findings report in the following.

B. Simulation Results

As discussed in Section III-A, slot assignment algorithms relying purely on k-hop knowledge will in general not completely rule out collisions. Table I lists the average percentile of nodes suffering from collisions in networks with density $\rho = 6$



(a) Duration ($\rho = 12/C = 8$)

(b) Duration ($\rho = 12/C = 8$)

(c) Duration ($\rho = 6/C = 8$)

(d) Duration ($\rho = 24/C = 8, \infty$)

(e) Energy consumption ($\rho = 6, 12, 24/C = 8$)

(f) Node depth in tree ($\rho = 6, 12, 24/C = 8$)

Fig. 3. Simulation results

using different scheduling assignment algorithms. CCH based on 3-hop knowledge led to about 9% collisions. In comparison, SPR with $\kappa = 4$ or 5 led to a collision rate below 1%. Note that SPR with $\kappa = 3$ did not produce satisfactory results for the topologies under consideration. In topologies with higher densities ($\rho = 12, 24$), SPR with $\kappa \geq 5$ creates interference-free slot assignments. Contrarily, CCH 3-hop assignments still suffer from collisions.

As stated earlier, buffer congestion is completely avoided with SPR. Furthermore, the required storage per node is independent of the network size and is proportional to the number of children in the routing tree. For this reason, the simulations to evaluate the efficiency of SPR for the many-to-one pattern were focused on two metrics: total completion time and energy consumption.

1) Total Completion Time: This metric evaluates the total completion time of the considered variant of the many-to-one routing pattern, where all nodes have stored a certain amount of data records that need to be routed towards a sink. This metric is calculated by dividing the total number of slots needed to collect all data from all nodes by the total number of data records. Additionally, this value can be used to compute the maximum rate, at which nodes can generate data records that can be reliably routed to the sink. Fig. 3(a) and 3(b) depict the duration in relation to the number of nodes for a medium density ($\rho = 12$). For all sizes of networks SPR is considerably faster than CCH. For networks up to 100 nodes SPR requires about 50% less time than CCH. Note that for networks with less than 100 nodes there is almost no difference between SPR with $\kappa = 5$ and $\kappa = 6$. The reason for this is that at density $\rho = 12$, the average depth of a node in the routing tree in a network with 100 nodes is approximately 3 (see Fig. 3(f)). Hence, both versions of SPR assign in these networks roughly the same slots. This changes when the number of nodes increases. Then the average node depth raises to about 8 and SPR with $\kappa = 5$ gets faster.

The simulations also revealed that the completion time of SPR is independent of the network density. Fig. 3(b) to 3(d) show that the completion times for SPR with $\kappa = 5$ vary only slightly. On the other hand the completion time of CCH increases considerably with the density. At the low density CCH is even marginally better than SPR with $\kappa = 5$. This may come as a surprise, because Fig. 3(f) shows that the average depth of a node increases strongly with decreasing density and this leads to a high reuse of slots within a path. But at low densities the depth of leaves has a high variance. As a consequence, if a node has among its predecessors leaves with low and high depth, then the slot corresponding to the leave with low depth is only needed for a short period of time. This prolongs the total completion time for low densities.

Another issue is the number of children a node is allowed to have in the routing tree. So far, we have considered routing trees, where the number of children was limited to 8. Fig. 3(d) compares the completion times for the cases unlimited number of children and maximal 8 children. As can be seen from this figure, SPR is not sensitive with respect to this number. An analysis of the average depth of the nodes showed that this number depends only on the size and the density of the network, but not on the number of children. This explains the behavior of SPR. CCH performs significantly better in case

the number of children is not limited. The reason is that in this case the sink will have more children, especially at high densities. Thus, the main bottleneck of CCH is attenuated.

2) Energy Consumption: Fig. 3(e) depicts the average time a node had turned on its transceiver in the relation to the size of the network for different densities. This time can be regarded as a measure for the energy consumption. Since CCH and SPR were evaluated with the same topologies, the number of packets to be forwarded to the sink is equal for both algorithms. Thus, the on-time adequately reflects energy consumption. The figure shows that SPR requires in all cases less energy than CCH. It also shows that energy consumption decreases with increasing density. This dependency reflects the factor, that the average depth of a node decreases with increasing density (see Fig. 3(f)). Simulation has shown that using unlimited buffers leads to equal energy consumption of both schemes. The increased energy usage of CCH is caused by the additional communication overhead that occurs in case of buffer overflow.

C. Lessons Learned

An important result of this work is that the density of a network has the largest influence on the performance and scalability of slot assignment schemes. Generally the memory consumption raises in dense networks due to the increasing number of neighbors. Especially those algorithms that use k-hop information do not scale with density even if they rely purely on local knowledge. Limited memory requires to bound the number of potential neighbors, and therefore each node can only have a limited number of children in the routing tree. In general, slot assignment schemes for the many-to-one routing pattern have the problem that their performance scales with the degree of balance of the communication tree. Restricting the number of children results in unbalanced trees and thus in a low performance of those slot assignment algorithms, where each node has a single sending but multiple receiving slots.

A general problem for slot scheduling algorithms based on k-hop knowledge are voids. These are areas with a diameter between r_{com} and r_{int}, which internally have no nodes but are surrounded by nodes. If a path of the routing tree bends around such a void, then interferences will occur. These cannot be resolved by slot schedules that based on k-hop knowledge. Even the performance of SPR is influenced by interference in these topologies as discussed in Section III-A. In networks with a low density and a small number of nodes other solutions to the many-to-one routing problem might be useful, e.g., assignment of unique slots for each node, which completely rule out collisions.

In wireless sensor networks often localized algorithms are preferred with regard to scalability. However, the above described problems with purely localized algorithms, i.e., only based on k-hop knowledge, reveal a basic phenomenon that local knowledge is not sufficient for an efficient slot assignment algorithm. The SPR algorithm makes local decisions but needs some global knowledge, i.e., the depth of the node and the depth information of the leaves in its subtree. This information can be easily gathered by a three-step breadth-first search, but it must be ensured that all nodes have the same view upon the whole network. κ is another global parameter that must be chosen based on the network density and the resulting topology. Extensive simulations have shown that topologies with a medium density or higher need at least a value of 5 for κ to ensure a nearly interference free slot assignment. A spatial reuse of slots can only be interference free with global knowledge (e.g., as used in CCH $\gamma = 1.9$) or global information (e.g., setting κ to highest path length). In this paper a fixed value of κ is used. However, the sink can adopt this value at the beginning of each slot distribution phase, based on the success rate during the last data-gathering phase, so that an optimal κ can be used by an iterative approximation.

VI. Conclusion

The spatial path-based reuse (SPR) is an ideal slot assignment procedure for a TDMA protocol to implement the many-to-one communication pattern. SPR completely avoids buffer congestion, a serious problem in tree-routing. If appropriately set up, it also avoids collisions on the wireless channel. The distributed allocation of slots is far simpler than in traditional slot assignment algorithms. Slot assignments created by SPR and CCH $\gamma = 1.9$ have been compared in a many-to-one routing tree by using the NS-2 simulator. Simulation results have shown that SPR slot assignments lead to significantly lower energy-consumption and total runtime. Furthermore SPR can be implemented distributedly with low effort and a comparably small memory footprint. This holds in almost all network configurations except in sparse networks with a small number of nodes. For such topologies, more investigations are required in order to find efficient slot assignment schemes.

References

[1] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-Low Power Data Gathering in Sensor Networks," in *IPSN'07*, Apr. 2007.

[2] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter, "LUSTER: Wireless Sensor Network for Environmental Research," in *SenSys'07*, Nov. 2007.

[3] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad-hoc Networks," in *MobiHoc'06*, May 2006.

[4] K. L. Bryan, T. Ren, L. DiPippo, T. Henry, and V. Fay-Wolfe, "Towards Optimal TDMA Frame Size in Wireless Sensor Networks," University of Rhode Island, Tech. Rep., Mar. 2007.

[5] J. Grönkvist, "Comparison Between Scheduling Models for Spatial Reuse TDMA," in *AWSI'04*, Jun. 2004.

[6] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder, "Efficient Interference-Aware TDMA Link Scheduling for Static Wireless Networks," in *MobiCom'06*, Sep. 2006.

[7] G. Zhou, T. He, J. A. Stankovic, and T. F. Abdelzaher, "RID: Radio Interference Detection in Wireless Sensor Networks," in *INFOCOM'05*, Mar. 2005.

[8] A. Rowe, R. Mangharam, and R. Rajkumar, *FireFly: A Time Synchronized Real-Time Sensor Networking Platform*. CRC Press, Nov. 2006.

[9] M. S. Nunes, A. Grilo, and M. Macedo, "Interference-Free TDMA Slot Allocation in Wireless Sensor Networks," in *LCN'07*, Oct. 2007.

[10] C. Zhou and B. Krishnamachari, "Localized Topology generation mechanisms for wireless sensor networks," in *GLOBECOM'03*, Dec. 2003.

[11] S. Chen and N. Yang, "Congestion Avoidance based on Light-Weight Buffer Management in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, Sep. 2006.

[12] V. Turau and C. Weyer, "TDMA-Schemes for Tree-Routing in Data Intensive Wireless Sensor Networks," in *PARIS'07*, Oct. 2007.