

SODaR: Sensor-Aided Overlay Deployment and Relocation Protocol for Vast-Scale Sensor Networks

Guanqun Yang

Iowa State University, Ames, IA 50011

{gqyang}@iastate.edu

Abstract

In this short report, we will present the theoretical proof and the maintenance protocol, which are omitted in our Infocom'08 paper.

I. SODAR – THEORETICAL ANALYSIS

In this section, we first explore the problem of syphon tree formulation when the number of syphons to be deployed in the field is sufficiently large by conducting asymptotic analysis. Then, we derive practical bounds on the probability of syphon tree formulation which provide a good guidance when deploying finite-size overlay syphon networks.

We assume that the sensing field has a disk shape with unit area (hence the radius of the sensing field is $\frac{1}{\sqrt{\pi}}$) and a single sink sits at center of the field. Sensing fields with irregular shapes, presence of obstacles, or multiple sinks will be studied in the future work. We use n to denote the number of syphons to be deployed in the sensing field. Moreover, we write $g(n) = o(f(n))$ if and only if $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = 0$, and $g(n) = \mathcal{O}(f(n))$ if and only if $\limsup_{n \rightarrow \infty} \left| \frac{g(n)}{f(n)} \right| < \infty$.

A. Asymptotic Analysis

We study the sufficient condition for SODaR to formulate a connected syphon tree in the sensing field. Consider the following mapping function from a 2-D sensing field (shown in Fig. 1(a)) to a 1-D line segment \overline{OB} (shown in Fig. 1(b)):

$$\mathcal{M} : \mathcal{R} \times \Theta \rightarrow \mathcal{R}, \quad (1)$$

where the sink is mapped to point O and each point in the 2-D sensing field with polar coordinates of (r, θ) is mapped to a point on \overline{OB} with distance r to point O . As shown in the figure, point s in the field is mapped to point t on \overline{OB} , and points belonging to the shaded annulus region in the field are mapped to points between t_1 and t_2 on \overline{OB} .

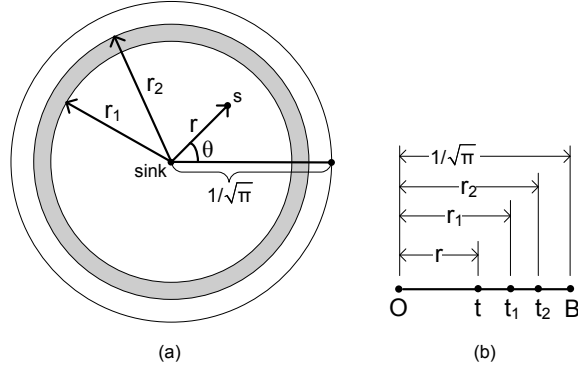


Fig. 1. Illustration of (a) a 2-D disk sensing field with unit area and (b) its 1-D mapping segment \overline{OB} . The sink is mapped to point O .

Definition 1 A point on the mapping segment \overline{OB} is said to be 1-covered with coverage radius of ℓ if there is at least one syphon¹ or sink that is within distance ℓ to the point. \overline{OB} is said to be 1-covered with a certain coverage radius if each point on it is 1-covered with the same coverage radius.

Lemma 1 SODaR formulates a connected syphon tree in a 2-D sensing field if and only if the sensing field's 1-D mapping segment is 1-covered with coverage radius of $\frac{R_c}{2}$.

Recall that R_c is the communication range of an 802.11 interface. This lemma is true according to a well-known fact [1] that if a 2-D space is 1-covered by a disk Boolean point coverage process with radius of $\frac{R_c}{2}$ and if two points are connected to each other when the distance between them is less than or equal to R_c , this point process must form a connected component that includes all the points; this fact also applies to the 1-D case. In other words, a syphon tree formulation problem in a 2-D sensing field is equivalent to a coverage problem of the corresponding 1-D mapping segment with coverage radius of $\frac{R_c}{2}$.

1) *Sufficient Condition for Syphon Tree Formulation:* We start our exploration by showing that if a certain set of points on the mapping segment \overline{OB} is 1-covered with a certain coverage radius, then the entire mapping segment is 1-covered with a slightly larger coverage radius. We call such set of points *virtual points* that are shown as the empty dots in Fig. 2. We consider L virtual points and they are evenly distributed between D to B , where $OD = \frac{R_c}{2}$. Hence, the distance between adjacent virtual points is $d = \frac{1/\sqrt{\pi} - R_c/2}{L}$.

Proposition 1 If all virtual points are 1-covered with coverage radius of $r = \frac{R_c - d}{2}$, the entire mapping segment \overline{OB} is 1-covered with coverage radius of $\frac{R_c}{2}$.

Proof: Since $OD = \frac{R_c}{2}$, we know that \overline{OD} is always covered by the sink. Therefore, we only need to consider the coverage of \overline{DB} . Fig. 3 shows a portion of \overline{DB} where

¹For simplicity, we reuse the term "syphon" to refer to a syphon's mapping point on \overline{OB} in the rest of this section.

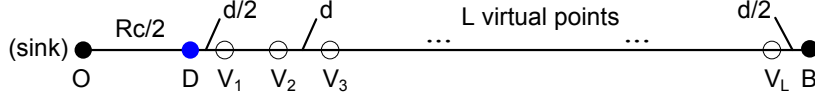


Fig. 2. Illustration of virtual points on the mapping segment \overline{OB} .

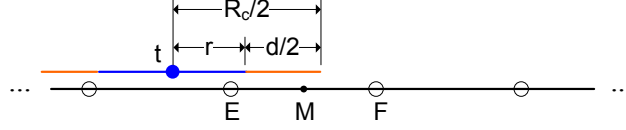


Fig. 3. Illustration of a syphon t and two virtual points E and F . M is the middle point of \overline{EF} .

empty dots represent virtual points. So if virtual point E is covered by syphon t with coverage radius of r , we know that \overline{EM} must be covered by t with a larger radius of $\frac{R_c}{2} = r + \frac{d}{2}$ simply because $EM = \frac{d}{2}$. Similarly, we can prove the coverage of \overline{MF} and subsequently the entire \overline{DB} . ■

Lemma 2 Given two monotonically-increasing functions $\phi(n)$ and $\varphi(n)$, where $\lim_{n \rightarrow \infty} \phi(n) = \infty$, $\lim_{n \rightarrow \infty} \varphi(n) = \infty$, and $\varphi(n) = o(\phi(n))$. If r and n satisfy $r^2 n = \phi(n)$, then for sufficiently large n , all virtual points are almost always 1-covered with coverage radius of $r = \frac{R_c - d}{2}$ where $d = \frac{1}{\sqrt{n\varphi(n)}}$.

Proof: Consider the i -th virtual point (V_i) in Fig. 2. It is easy to verify that $OV_i = \frac{R_c}{2} + \frac{d}{2} + (i-1)d$ and $\forall i, OV_i < \frac{1}{\sqrt{\pi}}$. Let P_i^v denote the probability that V_i is not 1-covered with coverage radius of r , i.e., the probability that there are no syphons mapping to locations between $V_i - r$ and $V_i + r$. It is the same as the probability that there are no syphons deployed in the annulus bounded by the radii of $(OV_i - r)$ and $\min\left(\frac{1}{\sqrt{\pi}}, OV_i + r\right)$ in the original 2-D sensing field. Since $\min\left(\frac{1}{\sqrt{\pi}}, OV_i + r\right) > OV_i$, the area of such annulus is always larger than $\pi[OV_i^2 - (OV_i - r)^2] = \pi r(r + 2id)$. Hence we have

$$P_i^v < (1 - \pi r(r + 2id))^n < e^{-n\pi r(r + 2id)}, \quad (2)$$

where n is the syphon density in the sensing field. Let P_c denote the probability that all virtual points are 1-covered with coverage radius of r . Applying Janson's inequality (Theorem 8.1.1 in [2]), we have $P_c \geq \prod_i (1 - P_i^v)$. Moreover, since

$$\begin{cases} nr^2 = \phi(n), \\ d = \frac{1}{\sqrt{n\varphi(n)}} \implies nrd = \sqrt{\frac{\phi(n)}{\varphi(n)}}, \end{cases} \quad (3)$$

both nr^2 and nrd go to infinity as $n \rightarrow \infty$. Hence for sufficiently large n , $P_i^v \leq 0.99$. Using the fact that $(1 - a) \geq e^{-5a}$ when $a \leq 0.99$, we know that for sufficiently large n ,

$$P_c \geq \prod_i e^{-5e^{P_i^v}} > \prod_i e^{-5e^{(-n\pi r(r + 2id))}} = e^{-5A \frac{1-q^L}{1-q}}, \quad (4)$$

where $A = e^{-n\pi r(r + 2id)}$ and $q = e^{-2n\pi rd}$. Based on (3), we know that, as $n \rightarrow \infty$, $A \rightarrow 0$, $(1 - q^L) \rightarrow 1$, and $(1 - q) \rightarrow 1$. Hence, $e^{-5A \frac{1-q^L}{1-q}} \rightarrow 1$ and $P_c \rightarrow 1$. ■

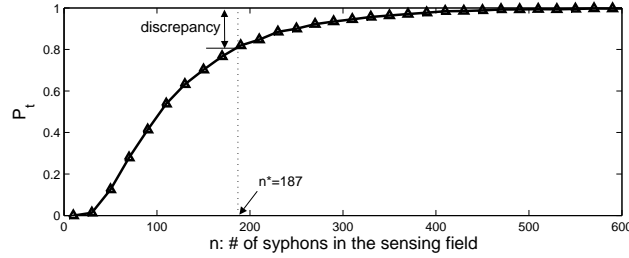
Theorem 1 If R_c and n satisfy $R_c^2 n = \phi(n)$ where $\phi(n)$ is a monotonically-increasing function and $\lim_{n \rightarrow \infty} \phi(n) = \infty$, then for sufficiently large n , SODaR is almost always able to formulate a connected syphon tree.

Proof: Recall that $r = \frac{R_c - d}{2}$, where $d = \frac{1}{\sqrt{n\varphi(n)}}$, $\varphi(n)$ is a monotonically-increasing function, $\lim_{n \rightarrow \infty} \varphi(n) = \infty$, and $\varphi(n) = o(\phi(n))$. Hence as $n \rightarrow \infty$, $d \rightarrow 0$ and $r \rightarrow R_c$. Therefore, we have

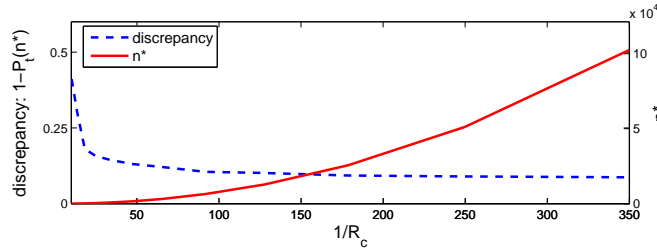
$$r^2 n = \frac{\phi(n)}{4} - \frac{d^2 n}{4} - r d n = \frac{\phi(n)}{4} - \frac{1}{4\varphi(n)} - o(r^2 n), \quad (5)$$

meaning that $r^2 n = \Theta(\phi(n))$ and hence $\varphi(n) = o(r^2 n)$. Using Lemma 2, we know that as $n \rightarrow \infty$, $P_c \rightarrow 1$. Let P_t denote the probability of syphon tree formulation. Then as $n \rightarrow \infty$, $P_t \rightarrow 1$ since $P_t \geq P_c$ which is a result of direct application of Lemma 1 and Proposition 1. ■

2) *Simulation Study of Asymptotic Results:* We use simulation to study the above asymptotic results. We consider a unit-area disk sensing field and use $\sqrt{\log_{10} \log_{10} n}$ for $\phi(n)$. We first fix $R_c = \frac{1}{10\sqrt{\pi}}$ and vary the number of syphons to study the probability of syphon tree formulation (P_t). Simulation results are plotted in Fig. 4(a) and each point is averaged over 100 simulation runs.



(a) n vs. P_t . Sensing field is a unit-area disk and $R_c = \frac{1}{10\sqrt{\pi}}$. $n^* = 187$ is the root of $R_c^2 n = \sqrt{\log_{10} \log_{10} n}$ and is the critical syphon density for SODaR according to asymptotic analysis. Discrepancy is measured as $(1 - P_t(n^*))$.



(b) $1/R_c$ vs. n^* vs. discrepancy.

Fig. 4. Simulation study of asymptotic results.

$n^* = 187$ shown in Fig. 4(a) is a reasonable root of the equation $R_c^2 n = \sqrt{\log_{10} \log_{10} n}$. Theorem 1 predicts that if there are more than n^* syphons deployed in the sensing field, SODaR should be able to form a connected syphon tree. Hence we call n^* the critical

syphon density for SODaR according to asymptotic analysis and its value varies with $\phi(n)$ and R_c . However, we see from the figure that this prediction is not so accurate. At this value of n^* , P_t is only about 0.8. The discrepancy ($1 - P_t(n^*)$) arises due to the asymptotic nature of results in Theorem 1 and we expect this discrepancy to become smaller as the value of $R_c^2 n$ grows, which is verified by the simulation results shown in Fig. 4(b). On the other hand, we observe that the discrepancy decreases and approaches zero very slowly. By choosing a slower growing $\phi(n)$ may help in reducing the discrepancy but the general trends of slow convergence (of discrepancy to zero) are similar. This observation implies that, although asymptotic results give us some insights about the problem and may be useful when the number of syphons is sufficiently large, they do not provide a good guidance when the number is small.

In order to answer practical questions such as “*how many syphons do we need to deploy in a given sensing field to guarantee syphon tree formulation with 91% probability,*” we provide in next section an upper and a lower bound for P_t , which are useful when deploying finite-size syphon networks.

B. Practical Bounds

1) *Lower Bound:* We use P_l to denote the lower bound for P_t . Consider a series of events T_i ($1 \leq i \leq k$). Each of the k events represents a sufficient condition for syphon tree formulation. Hence, $P_t \geq P[T_1 \cup \dots \cup T_k]$ and we let $P_l = P[T_1 \cup \dots \cup T_k]$. Event T_i is illustrated in Fig. 5, where \overline{OB} is divided into smaller segments. Segments with a dot inside them have at least one syphon, while segments marked with a cross do not have any syphon inside them. If a segment is not marked with either dot or cross, it may or may not have syphons inside it. Without loss of generality, we assume that the radius of the sensing field is at least $\frac{3R_c}{2}$. This is reasonable since we are considering vast-scale sensing field whose size is typically much larger than the communication range of an 802.11 wireless interface.

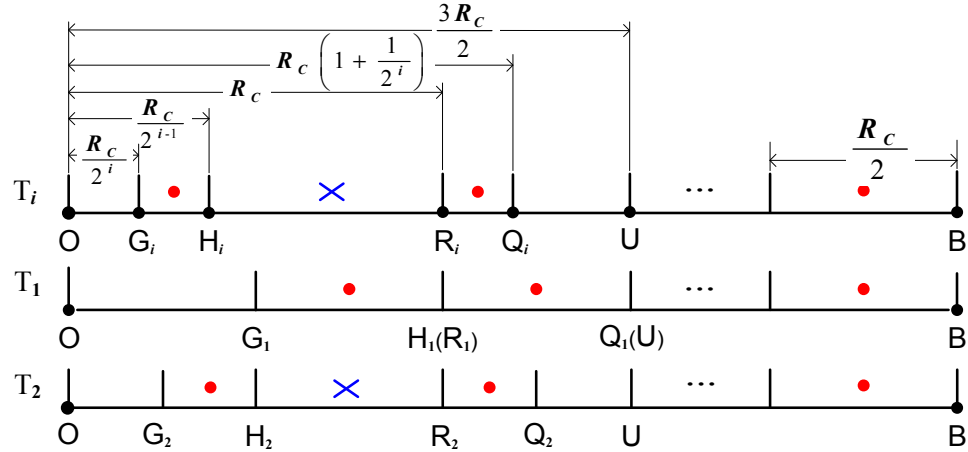


Fig. 5. Illustration of events T_i , T_1 and T_2 .

As shown in the figure, in case of T_i , the lengths of $\overline{OG_i}$, $\overline{OH_i}$, $\overline{OR_i}$, $\overline{OQ_i}$, and \overline{OU}

are $\frac{R_c}{2^i}$, $\frac{R_c}{2^{i-1}}$, R_c , $R_c(1 + \frac{1}{2^i})$, and $\frac{3R_c}{2}$, respectively. Moreover, segment \overline{UB} is evenly divided into smaller segments, each with a length of $\frac{R_c}{2}$. Events T_1 and T_2 are also shown in Fig. 5. We can see that, in case of T_1 , points H_1 and R_1 overlap while points Q_i and U overlap. It is easy to see that these k events are constructed carefully to guarantee mutual exclusiveness among them. Hence, $P_l = P[T_1 \cup \dots \cup T_k] = \sum_{i=1}^k P[T_i]$. $P[T_i]$ can be calculated as follows:

$$P[T_i] = \begin{cases} \left(1 - e^{-\frac{3\pi R_c^2 n}{4}}\right) \left(1 - e^{-\frac{5\pi R_c^2 n}{4}}\right) \hat{P}, & i = 1, \\ \left(1 - e^{-\frac{3\pi R_c^2 n}{4^i}}\right) \left(1 - e^{-\frac{\pi R_c^2 n(1+2^i+1)}{4^i}}\right) e^{-\frac{\pi R_c^2 n(4^i-1-1)}{4^i-1}} \hat{P}, & 1 < i \leq k, \end{cases} \quad (6)$$

where \hat{P} is the probability that each segment between points U and B contains at least one syphon. The tightness of P_l varies with the value of k and a larger k yields a tighter lower bound. Although it is difficult to get a closed-form expression for P_l , we find out that $P[T_i]$ decreases faster than a geometric series as i increases. Hence even a small value of k (e.g., $k = 10$) results in a fairly tight lower bound, as will be shown in Section I-B3.

2) *Upper Bound:* We use P_u to denote the upper bound for P_t and let $P_u = 1 - P_o$ where P_o is the probability that no syphons are deployed within R_c distance from the sink. This is intuitively true since existence of a connected syphon tree requires one or more syphons with R_c distance from the sink.

Theorem 2 *If R_c and n satisfy $R_c^2 n = \phi(n)$ where $\phi(n)$ is a monotonically increasing function and $\lim_{n \rightarrow \infty} \phi(n) = \infty$, then the difference between P_t and its upper bound $P_u = 1 - P_o$ is $P_o \mathcal{O}\left(\max\left\{\frac{1}{\varphi(n)}, e^{-2\pi\sqrt{\frac{\phi(n)}{\varphi(n)}}}\right\}\right)$, where $\lim_{n \rightarrow \infty} \varphi(n) = \infty$ and $\varphi(n) = o(\phi(n))$.*

Proof: First, we know that $P_o = (1 - \pi R_c^2)^n$. Then, applying the ‘‘virtual point’’ technique that has been used in the asymptotic analysis (recall that $d = \frac{1}{\sqrt{n\varphi(n)}}$), we have

$$\begin{aligned} P_1^v &= (1 - \pi(R_c^2 - d^2))^n = P_o \left(1 + \frac{\pi d^2}{1 - \pi R_c^2}\right)^n \\ &= P_o \left(1 + \mathcal{O}\left(\frac{n\pi d^2}{1 - \pi R_c^2}\right)\right) = P_o \left(1 + \mathcal{O}\left(\frac{1}{\varphi(n)}\right)\right) \end{aligned} \quad (7)$$

and

$$\begin{aligned} P_2^v &= (1 - \pi((R_c + d)^2 - 4d^2))^n = P_o \left(1 + \frac{\pi(3d^2 - 2R_c d)}{1 - \pi R_c^2}\right)^n \\ &= P_o \mathcal{O}\left(e^{\frac{n\pi(3d^2 - 2R_c d)}{1 - \pi R_c^2}}\right) = P_o \mathcal{O}\left(e^{-2\pi\sqrt{\frac{\phi(n)}{\varphi(n)}}}\right). \end{aligned} \quad (8)$$

Then, we have

$$\begin{aligned} P_t &\geq P_c \geq \prod_i (1 - P_i^v) \\ &= 1 - P_1^v - \mathcal{O}(P_2^v) = 1 - P_o + P_o - P_1^v - \mathcal{O}(P_2^v) \\ &= 1 - P_o - P_o \mathcal{O}\left(\frac{1}{\varphi(n)}\right) - P_o \mathcal{O}\left(e^{-2\pi\sqrt{\frac{\phi(n)}{\varphi(n)}}}\right). \end{aligned} \quad (9)$$

Hence,

$$\begin{aligned}
P_u - P_t &= 1 - P_o - P_t \\
&\leq P_o \left(\mathcal{O} \left(\frac{1}{\varphi(n)} \right) + \mathcal{O} \left(e^{-2\pi\sqrt{\frac{\phi(n)}{\varphi(n)}}} \right) \right) \\
&= P_o \mathcal{O} \left(\max \left\{ \frac{1}{\varphi(n)}, e^{-2\pi\sqrt{\frac{\phi(n)}{\varphi(n)}}} \right\} \right).
\end{aligned} \tag{10}$$

■

This theorem offers an interesting fact that the difference between $P_u = 1 - P_o$ and P_t decreases much faster than P_o . Hence, by choosing proper $\phi(n)$ and $\varphi(n)$, P_u could be very tight for practical network configurations with finite number of syphons.

3) *Simulation Study of Practical Bounds:* We use simulation to evaluate the tightness of the derived bounds. Similar to in Section I-A2, we consider a unit-area circle sensing field, fix $R_c = \frac{1}{10\sqrt{\pi}}$, and vary the number of syphons from 0 to 600. Simulation results are plotted in Fig. 6 and each point is averaged over 100 simulation runs.

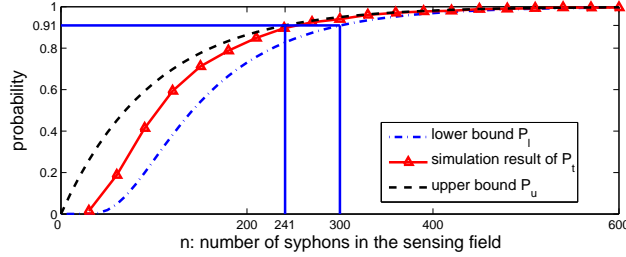


Fig. 6. Simulation study of practical bounds.

The value k is set to 10 when calculating and plotting the lower bound in the figure. We see from the figure that the derived bounds are fairly tight, particularly when the target probability of syphon tree formulation (P_t) is high, which is typically the scenario of our interest in practice. Moreover, it is interesting to observe that the upper bound becomes very tight when P_t is beyond 90%, which confirms our analysis in Section I-B2 that the difference between P_u and P_t decreases much faster than P_o .

Given the tightness of the derived bounds, when the target P_t is high, we could use either upper or lower bound to answer practical questions such as the one mentioned in Section I-A2: “how many syphons do we need to deploy in a given sensing field to guarantee syphon tree formulation with 91% probability.” The answer to that is, by looking up Fig. 6, deployment of 300 syphons (according to the lower bound) guarantees the target P_t of 91%, while deployment of 241 syphons (according to the upper bound) will also most likely result in acceptable performance. In SODaR, the lower bound P_l is used when making the syphon deployment decisions.

II. PRACTICAL CONSIDERATIONS AND DISCUSSIONS

We now discuss some practical issues that should be taken into account when deploying overlay syphon networks.

A. Choice of \hat{R}_c

In order to ensure that the advertisement message sent by an on-tree syphon reaches off-tree syphons near the ring belt it belongs to, we choose \hat{R}_c slightly larger than R_c : $\hat{R}_c = \alpha R_c$, where $\alpha \geq 1$ is a design parameter.

B. Connecting Off-Tree Syphons to On-Tree syphons

During the “Sensor-Aided Relocation of Off-Tree Syphons” step described in our paper, it is possible that when an off-tree syphon arrives at its destination for joining the syphon tree, it cannot hear the on-tree syphon. In this case, the off-tree syphon should continue to move towards the on-tree syphon. Since all sensors within the on-tree syphon’s 802.11 transmission range maintain paths to the syphon, these sensors can serve as landmarks for the off-tree syphon. The off-tree syphon stops as soon as it is able to communicate with the on-tree syphon.

C. Boundary Conditions

In practice, the sensing field may not be a disk, or the sink may not be at center of the field. In both cases, the propagation of advertisement messages or probing messages may be blocked at the boundary of the field, and hence some off-tree syphons may not be able to join the syphon tree using our protocol. Fig. 7 shows an example of a square sensing field, where X is an on-tree syphon but syphon Y is off-tree. Y receives X ’s advertisement message, but its own probing message is blocked by the field boundary hence cannot be propagated to X . Such problem can be addressed as follows. During the “Setting up Data Forwarding Path to Syphon Tree” phase, each off-tree syphon that is not able to join the tree during the syphon tree expansion will receive multiple Voronoi-setup-messages. Based on received messages, it will select the nearest on-tree syphon, and move towards the syphon. The movement can be aided by the landmarks, i.e. sensors on the path towards the on-tree syphon. Note that such path has been established during the propagation of Voronoi-setup-messages.

D. Slow Moving Syphons

During the “Sensor-Aided Relocation of Off-Tree Syphons” step, some off-tree syphon may move slower than expected, and hence reach its destination after the Voronoi diagram has been formed. If this happens, the on-tree syphon to which the off-tree syphon connects will notify the off-tree syphon to set up a Voronoi cell immediately. This requires a scoped broadcast of a Voronoi-setup-message, and only the sensors located in the newly connected syphon’s Voronoi cell process the message.

III. SODAR – SYPHON TREE MAINTENANCE

After the network operates for some time, some syphons may stop functioning because of various reasons such as physical damage, energy depletion and so on. In this case, the syphon tree may be partitioned. To address this issue, we propose a scheme to repair a partitioned syphon tree. Our scheme assumes that there is no simultaneous failure of two or more syphons at the same time. This is reasonable since syphons are more powerful and reliable than sensors.

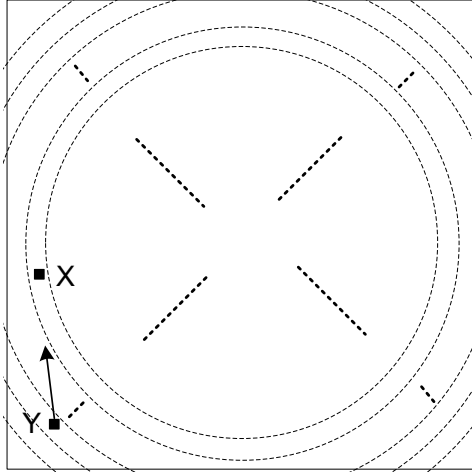


Fig. 7. Example of a square sensing field. X and Y are two syphons.

A. Detection of Syphon Failure

We propose two mechanisms for detecting syphon failure: *detection by sensors* and *detection by syphons*. When a sensor generates a data message, it is forwarded hop by hop towards the nearest syphon. Particularly, after data arrives at a sensor that is a one-hop neighbor to the syphon, the sensor forwards data to the syphon and waits for an ACK from the syphon. If the syphon fails, the sensor will not get the ACK and thus can detect the failure. In this case, the sensor may either wait till the syphon tree recovers and send again, or forward data hop-by-hop directly towards the sink (i.e., fall back to the no-syphon situation). Failure of a syphon may also be detected by its child syphon as follows. When a syphon forwards data to the sink along the syphon tree, its parent syphon shall acknowledge each data reception. So if a syphon fails to receive an ACK from its parent, the parent syphon is detected as failed.

Once a syphon failure has been detected, related syphons shall be notified to heal the syphon tree. If the failure is detected by a sensor, the sensor will initiate a broadcast within a radius \hat{R}_c centered at the failed syphon. Each child syphon of the failed syphon will hear this broadcast. On the other hand, if the failure is detected by a syphon, it sends a message towards the failed parent syphon, and the sensor along the route that is closest to the failed syphon will detect the failure. The sensor will then initiate a broadcast as described above.

B. Self-Healing of Syphon Tree

A failed syphon could be either a leaf node or an intermediate node on the syphon tree. If it is a leaf node, the tree is not broken and the position of the failed syphon will not be filled by any other syphon. In this case, SODaR goes to the next step of *Voronoi Diagram Adjustment*. Otherwise, the syphon tree is partitioned, and a self-healing process shall be performed. Specifically, a child of the failed syphon moves up to fill its position

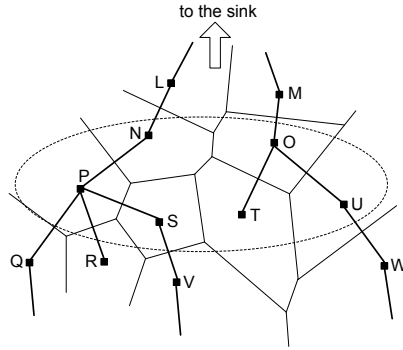


Fig. 8. Syphons (black squares) and Voronoi Diagram.

and inherit its ID, the position left by the moving child is filled and inherited by its own child, and so on and so forth.

In order to minimize the number of syphons that need to move and thus affect the topology of the syphon tree the least, SODaR chooses the syphons on the shortest branch from the failed syphon to the leaf to complete the cascaded self-healing process. For example, in Fig. 8, if syphon L fails, branch $L - N - P - R$ is the shortest branch from L to the leaf, and hence syphons N , P and R should move up to repair the tree. To enable each syphon to decide whether it should move during self-healing, we require that each syphon on the syphon tree to be aware of the topology of the subtree rooted at its parent. This can be achieved as follows: after the final syphon tree is formed, the sink collects the topology of the syphon tree and communicates it to every on-tree syphon; each syphon will maintain the topology of the sub-tree rooted at its parent node on the syphon tree. With the knowledge of the subtree topology, each child of the failed syphon can decide whether it should move. In case it should move, it will also notify its child on the shortest branch of its subtree. Such process continues until the notification is propagated to a leaf node. Note that, as these syphons move, the topology of the syphon tree changes. Hence the change should be reported to the sink as soon as the syphon tree recovers, and then the sink notifies the change to all syphons.

Once the branch of moving syphons has been determined, these syphons may move in various orders. To efficiently relocate these syphons, SODaR requires the leaf node to move first since its parent can provide guidance for the movement; that is, the parent will transmit signals periodically using its 802.11 interface, and the leaf node will use its smart antenna to determine and adjust its moving direction. After the leaf arrives at its destination, its parent may start to move, and so on and so forth, until it is the turn for the syphon whose parent is failed. The movement of this syphon cannot be guided by its parent; however, the sensors on the route between the syphon and its parent can provide the required guidance.

C. Voronoi Diagram Adjustment

Observe that no matter whether the failed syphon is a leaf or intermediate node, after the above self-healing process, it is always the case that the position of an original leaf

becomes void. Thus, sensors in this leaf's Voronoi cell have to find a new nearest syphon to send their data. In other words, the Voronoi diagram must be adjusted. The adjustment is triggered in one of the two ways:

- (i) If the failed syphon is a leaf, some sensors within its one sensor-hop range will detect the failure, and send out a *Voronoi adjustment* message.²
- (ii) If the failed syphon is an intermediate node, the leaf syphon on the shortest branch of its sub-tree will leave to replace its parent. Right before its departure, this leaf syphon sends out a *Voronoi adjustment* message.

Algorithm 1 depicts how sensors react upon receipt of the Voronoi adjustment message.

Algorithm 1 Processing *Voronoi-adjustment-message* $msg\langle SyID \rangle$

```

For sensor (with ID  $i$ )
1: if syphon  $msg.SyID$  is in its  $list$  then
2:   remove the entry from  $list$ 
3:   rebroadcast  $msg\langle SyID \rangle$ 
4:   if syphon  $msg.SyID$  is the nearest syphon then
5:     if  $list = nil$  then
6:        $Node_i.SHC \leftarrow \infty$ 
7:        $Node_i.SyID \leftarrow nil$ 
8:        $Node_i.SeID \leftarrow nil$ 
9:     else
10:      find the entry  $j$  in  $list$  with the smallest  $SHC$ 
11:       $Node_i.SHC \leftarrow list[j].SHC$ 
12:       $Node_i.SyID \leftarrow list[j].SyID$ 
13:       $Node_i.SeID \leftarrow list[j].SeID$ 
14:      broadcast Voronoi-setup-message
            $msg\langle Node_i.SyID, Node_i.SHC + 1, Node_i.SeID \rangle$ 
15:     end if
16:   end if
17: end if

```

Specifically, on receiving the message which carries the ID of the syphon whose position becomes void, each sensor checks its list of syphons. If the list contains an entry of this syphon, the entry is removed and the received Voronoi adjustment message is rebroadcast; moreover, if this syphon was the nearest syphon to a receiving sensor, the sensor selects the nearest syphon from its updated syphon list as the replacement and announces this event by broadcasting a Voronoi-setup-message. Such newly-generated Voronoi-setup-messages are handled in a similar way as that in Algorithm 2.

REFERENCES

- [1] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proc. of ACM SenSys'03*, Los Angeles, CA, 2003, pp. 28–39.
- [2] N. Alon and J.H. Spencer, *The Probabilistic Method*, John Wiley & Sons, 2000.

²In SODaR, a syphon needs to inform its one-hop neighbor sensors whether it is a leaf node or not. If it is an intermediate node, the detecting sensors will not send out the *Voronoi adjustment* message.

Algorithm 2 Setting Up Forwarding Paths Between Sensors and On-Tree Syphons

For every syphon (with ID i)

- 1: broadcast *Voronoi-setup-message* $msg(SyID = i, SHC = 1, SeID = SyID)$
/* $SyID$: syphon ID,
 SHC : number of sensor hops away from this syphon,
 $SeID$: ID of sensor forwarding this message */

For every sensor (with ID j)

Initialization:

- 1: $Node_j.SHC \leftarrow \infty$
 /* number of sensor hops to the nearest on-tree syphon */
- 2: $Node_j.list \leftarrow nil$
 /* used for recording syphon info */

Upon receiving *Voronoi-setup-message* $msg(SyID, SHC, SeID)$

- 3: **if** $Node_j.SHC > msg.SHC$ **then**
 - 4: $Node_j.SHC \leftarrow msg.SHC$
 /* record sensor hop count to the nearest on-tree syphon */
 - 5: $Node_j.SyID \leftarrow msg.SyID$
 /* record the ID of the nearest on-tree syphon */
 - 6: $Node_j.SeID \leftarrow msg.SeID$
 /* record the ID of the next-hop sensor towards the nearest on-tree syphon */
 - 7: forward $msg(Node_j.SyID, Node_j.SHC + 1, j)$
 - 8: **end if**
 - 9: **if** no $\langle msg.SyID, *, * \rangle$ in $Node_j.list$ **then**
 - 10: put $\langle msg.SyID, msg.SHC, msg.SeID \rangle$ to $Node_j.list$
 - 11: **end if**
 - 12: **if** there is $\langle msg.SyID, x, * \rangle$ ($x > msg.SHC$) in $Node_j.list$ **then**
 - 13: replace $\langle msg.SyID, x, * \rangle$ with
 $\langle msg.SyID, msg.SHC, msg.SeID \rangle$
 - 14: **end if**
-