# Poster Abstract: A Distributed Algorithm to Compute Spatial Skyline in Wireless Sensor Networks [*]

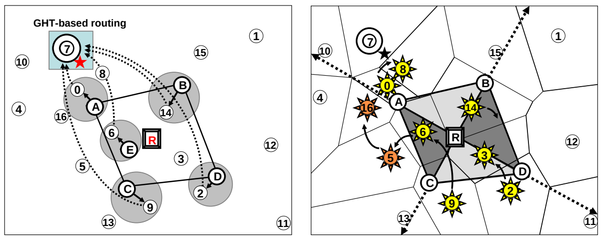
SunHee Yoon and Cyrus Shahabi
Department of Computer Science, University of Southern California
sunheeyo@usc.edu, shahabi@usc.edu

## ABSTRACT

Spatial skyline queries can be used in wireless sensor networks for *collaborative positioning* of multiple objects. However, designing a distributed spatial skyline algorithm in resource constrained wireless environments introduces several research challenges: how to combine multi-dimensional data, (*e.g.* distances to multiple events) to compute the skylines efficiently, accurately, quickly, progressively, and concurrently while dealing with the network and event dynamics. We address this challenge by designing Distributed Spatial Skyline (DSS) algorithm. DSS is the first *distributed* algorithm to compute spatial skylines. In a network of 554 nodes, DSS reduces the communication overhead by up to 91% over a centralized algorithm with 100% accuracy.

## 1. INTRODUCTION

Given a set of data points $P$ and a set of query points $Q$, spatial skylines are those members of P that are not spatially dominated by any other point in $P$ with respect to $Q$ [2]. A point $p1$ spatially dominates a point $p2$ with respect to $Q$, if and only if $p1$ is closer to at least one query point as compared to $p2$ and has the same distance as $p2$ to the rest of the query points. The spatial skyline query can be utilized to collaboratively position multiple targets in wireless networks. For example, suppose from a set of police cars (data points $P$) we want to identify a candidate subset to be dispatched to multiple incident points (query points $Q$). This candidate subset includes those cars that are not dominated by any other police car with respect to all the incidents, and hence they are the spatial skylines. Spatial skyline query can be performed using either centralized or distributed approaches.



(a) Phase 1: Computing the query convex hull and $R$. (b) Phase 2 and 3: Search internal, external skylines.

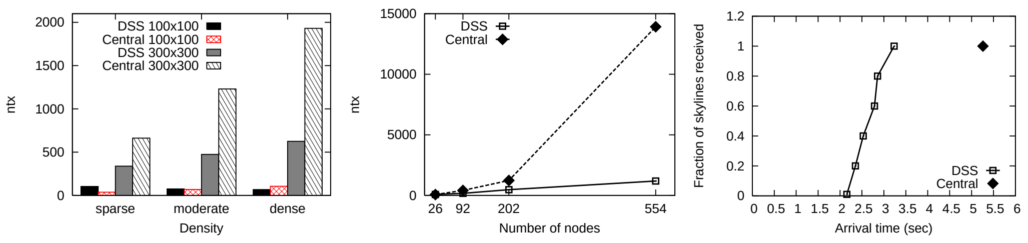
**Figure 1: DSS algorithm. {A,B,C,D,E} are query points and quadrilateral $ABDC$ is the query convex hull. Star is a synchronization point, $R$ is a rendezvous point, and sunflowers are spatial skylines {0,8,14,2,3,9,6,5,16}.**

The centralized approach assumes a complete set of data in space and time; if we use TAG-like data collection tree [1], it requires high transmission overhead for resource constrained wireless networks. For the time-critical applications, centralized approach cannot meet the promptness requirement because it does not provide any partial results early. We address these challenges by proposing a efficient Distributed Spatial Skyline (DSS) algorithm to collaboratively find the spatial skylines in WSN. DSS parallelizes the search for skylines by partitioning the search space recursively based on the geometric properties of the nodes and the topology. This approach enables an efficient, scalable, progressive, fast, and concurrent search while providing correct results. Our DSS algorithm is the first *distributed* algorithm to compute *spatial skylines*.

## 2. DSS ALGORITHM

The DSS algorithm partitions the network into two sets of nodes: nodes inside or outside of query convex hull $CH(Q)$ which is the convex hull of query points $Q$.

[*]This research has been funded in part by NSF grants IIS-0238560 (PECASE) and CNS-0831505 (CyberTrust), the NSF Center for Embedded Networked Sensing (CCR-0120778) and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.



(a) Number of transmissions by network size and density. (b) Number of transmissions by number of nodes. (c) Progressiveness and delay to compute skylines.

**Figure 2: Message transmission overhead (a), scalability (b), and delay (c) of DSS.**

DSS secondarily partitions those two sets of nodes by triangulation and enables parallel search per *triangle*, the triangle region divided by two consecutive points in $CH(Q)$ and rendezvous point $R$ which is the centroid of $CH(Q)$. DSS algorithm operates in these three phases:

**Phase 1:** DSS computes $CH(Q)$ and $R$ for partitioning. The node closest to each query point reports the <coordinate of the query point> using GHT-based routing to the synchronization point (star in Fig 1(a)), where all the query points across the network are gathered. GHT home node (node 7 in Fig 1(a)) receives those coordinates, and computes $CH(Q)$ and $R$. DSS then transmits the $<CH(Q), R>$ to each triangle home $t$ which is the node closest to each point in $CH(Q)$.

**Phase 2:** DSS searches the internal skylines per triangle which are definite skylines located within or near $CH(Q)$. Each $t$ (nodes 0, 14, 2, and 9 in Fig 1(b)) naturally becomes a spatial skyline by the definition [2]. $t$ subsequently searches for other undiscovered internal skylines among all Voronoi neighbors [3] by applying two rules [2] with $CH(Q)$ replaced by triangle. This change does not impact the correctness of the algorithm. DSS determines a node to be an internal skyline either 1) if a node is within a triangle or 2) if the Voronoi cell of a node intersects the edge not adjacent to $R$ in the triangle (*e.g.* edge $AB$ in the triangle $ABR$ in Fig 1(b)). When $t$ receives all the internal skylines from its triangle, it sends this list to $R$.

**Phase 3:** DSS searches external skylines, the skylines that are located outside of $CH(Q)$. Each $t$ traverses the internal skylines of corresponding extended triangle (union of triangle and area enclosed by dotted line in Fig. 1(b)) by sending $<CH(Q)$, triangle points, internal skylines> to them. While this message traverses each skyline, DSS discovers a new skyline if any of its Voronoi neighbors is not spatially dominated by the skylines in the packet. DSS appends the new skyline to the list, and sends it to the next skyline node in the list. This traversal terminates when no new skylines are discovered [2]. Then, DSS sends the list of skylines to the clockwise $t$. When the clockwise $t$ receives all the skylines, it sends this list to $R$. In Fig. 1(b), the clockwise $t$ 0, 14, 2, and 9 report aggregated skylines {9,6,5,16}, {0,8}, {14}, and {2,3} to $R$, and $R$ aggregates them into the final skyline list {0,8,14,2,3,9,6,5,16}.

## 3. PERFORMANCE EVALUATION

We evaluate the DSS algorithm compared to the centralized algorithm running over TAG [1]-like data gathering tree using simulation in TinyOS 1.1.15.

**Efficiency:** The DSS algorithm is more efficient than the centralized algorithm. Fig. 2(a) shows that, in a network with 292 nodes, DSS incurs 68% fewer transmission compared to the centralized algorithm. **Scalability:** The DSS algorithm is more scalable than the centralized algorithm. Fig. 2(b) shows that communication overhead of DSS scales linearly with the size of the network while the overhead of centralized algorithm scales exponentially. **Accuracy:** In all of our experiments, the DSS algorithm provides 100% accurate results just like the centralized algorithm. **Delay:** DSS algorithm provides progressive results during the execution of algorithm and returns the final results faster than the centralized algorithm. Fig. 2(c) shows that DSS starts to return the partial results starting 2.16 second and the complete results at 3.23 second. The centralized algorithm cannot return the progressive result. It produces the final results at 5.26 second.

## 4. CONCLUSIONS AND FUTURE WORK

We designed the first distributed algorithm to compute spatial skyline for collaborative positioning in WSN. We showed that DSS is more efficient, scalable, and faster than the centralized algorithm while providing 100% accurate skylines. We plan to extend our work to compute distributed skylines with mobile query points.

## 5. REFERENCES

[1] S. Madden, M. J. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *OSDI*, Dec. 2002.

[2] M. Sharifzadeh and C. Shahabi. The spatial skyline queries. In *VLDB*, Sept. 2006.

[3] M. Sharifzadeh and C. Shahabi. Approximate Voronoi cell computation on spatial data streams. In *VLDB Journal Vol.18 No.1*, 2009.