Efficient k-Coverage Algorithms for Wireless Sensor Networks

Mohamed Hefeeda
(joint work with Majid Bagheri)

INFOCOM Minisymposium 07
7 May 2007
Motivations

- Wireless sensor networks have been proposed for many real-life monitoring applications
  - Habitat monitoring, early forest fire detection, …

- *k*-coverage is a measure of quality of monitoring
  - *k*-coverage $\equiv$ every point is monitored by $k+$ sensors
  - Improves reliability and accuracy

- *k*-coverage is essential for some applications
  - E.g., intruder classification, object tracking
Our $k$-Coverage Problem

- Given $n$ already deployed sensors in a target area, and a desired coverage degree $k \geq 1$, select a minimal subset of sensors to $k$-cover all sensor locations

- **Assumptions**
  - Sensing range of each sensor is a disk with radius $r$
  - Sensor deployment can follow any distribution
  - Nodes **do not** know their locations
  - Point coverage approximates area coverage (dense sensor network)
Our $k$-Coverage Problem (cont’d)

- $k$-coverage problem is NP-hard [Yang 06]

- Proof: reduction to minimum dominating set problem
  - Model network as graph
  - An edge between any two nodes if they are within the sensing range of each other
  - Finding the minimum number of sensors to 1-cover yields a minimum dominating set

Mohamed Hefeeda
Our Contributions: $k$-Coverage Algorithms

- **We propose two approximation algorithms**
  - Randomized $k$-coverage algorithm (RKC)
    - Simple and efficient
  - Distributed RKC (DRKC)

- **Basic idea:**
  - Model $k$-coverage as a hitting set problem
  - Design an approximation algorithm for hitting set
    - Prove its correctness, verify using simulations
  - Decentralize it
Set Systems and Hitting Set

- Set system \((X, R)\) is composed of
  - set \(X\), and
  - collection \(R\) of subsets of \(X\)

- \(H\) is a hitting set if it has a nonempty intersection with every element of \(R\):

\[
H \subseteq X \\
\forall s \in R, \quad H \cap s \neq \emptyset
\]
Set System for $k$-Coverage

- $X$: set of all sensor locations
- For each point $p$ in $X$, draw circle of radius $r$ (sensing range) centred at $p$
- All points in $X$ which fall inside that circle constitute one set $s$ in $R$
- The hitting set must have at least one point in each circle

- Thus all points are covered by the hitting set
Example: 1-Coverage
Example: $k$-Coverage ($k = 3$)

Elements of the hitting set are centers of $k$-flowers

Mohamed Hefeeda
Centralized Algorithm (RKC)

- **Build an approximate hitting set**
  1. Assign weights to all points, initially 1
  2. Select a **random** set of points, referred to as $\varepsilon$-net
     - Selection biased on weights
  3. If current $\varepsilon$-net covers all points, terminate
  4. Else double weight of one under-covered point, goto 2 if number of iterations is below a threshold ($\sim \log |X|$)
  5. Double size of $\varepsilon$-net, goto 1
\( N \) is an \( \varepsilon \)-net for set system \((X,R)\) if it has nonempty intersection with every element \( T \) of \( R \) such as \( |T| \geq \varepsilon |X| \)

Thus, \( \varepsilon \)-net is required to hit only large elements of \( R \)

- (hitting set must hit every element of \( R \))

Idea:

- Find \( \varepsilon \)-nets of increasing sizes (decreasing \( \varepsilon \)) till one of them hits all points
\( \varepsilon \)-net Construction

- \( \varepsilon \)-nets can be computed efficiently for set systems with finite VC-dimension [Bronnimann 95]
  - We prove that our set system has VC-dimension = 3

- **Randomly** selecting

  \[
  \max \left\{ \frac{4}{\varepsilon \log \frac{2}{a}}, \frac{8d}{\varepsilon \log \frac{8d}{\varepsilon}} \right\}
  \]

  points of \( X \) constitutes an \( \varepsilon \)-net with probability \( 1-a \) for \( 0 < a < 1 \) where \( d \) is the VC-dimension
Details of RKC

Randomized K-Coverage: RKC(X, r, k)

1. $c = 1$; \(//\) sets the initial size of $\epsilon$-net
2. while (net-size($\frac{1}{c}$) $\leq n$) do
3. set weights of all points to 1;
4. $\epsilon = 1/c$;
5. for $i = 1$ to $4c \log \frac{n}{c}$
6. $N = net$-finder($X$, $k$, $\epsilon$, $r$);
7. $u = verifier (X, N, k, r)$;
8. if ($u ==$ null)
9. return $N$;
10. else
11. double weight of $u$;
12. $c = 2 \times c$;
13. return $\emptyset$;
Theorem 1: RKC …

- ensures that every point is $k$-covered,
- terminates in $O(n^2 \log^2 n)$ steps, and
- returns a solution of size at most $O(P \log P)$, where $P$ is the minimum number of sensors required for $k$-coverage.
Distributed Algorithm: DRKC

- RKC maintains only two global variables:
  - size of $\varepsilon$-net
  - aggregate weight of all nodes

- Idea of DRKC: Emulate RKC by keeping local estimates of global variables
  - Nodes construct $\varepsilon$-net in distributed manner
  - Nodes double their weights with a probability
  - Each node verifies its own coverage
Theorem 2:

The average number of messages sent by a node in DRKC is $O(1)$, and the maximum number is $O(\log n)$
Performance Evaluation

- Simulation with thousands of nodes
- Verify correctness (k-coverage is achieved)
- Show efficiency (output size compared optimal)
- Compare with other algorithms
  - LPA (centralized linear programming) and PKA (distributed based on pruning) in [Yang 06]
  - CKC (centralized greedy) and DPA (distributed based on pruning) in [Zhou 04]
Correctness of RKC

- RKC achieves the requested coverage degree

**Requested k = 1**

**Requested k = 8**
Efficiency of RKC

- Compare against necessary and sufficient conditions for $k$-coverage in [Kumar 04]
Correctness of DRKC

- DRKC achieves the requested coverage degree

Requested $k = 1$

Requested $k = 8$

Mohamed Hefeeda
Efficiency of DRKC

- DRKC performs closely to RKC, especially in dense networks

Mohamed Hefeeda
DRKC consumes less energy and prolongs network lifetime.
Conclusions

- **Presented a centralized $k$-coverage algorithm**
  - Simple, and efficient (log-factor approximation)
  - Proved its correctness and complexity

- **Presented a fully-distributed version**
  - Low message complexity, prolongs network lifetime

- **Simulations verify that our algorithms are**
  - Correct and efficient
  - Outperform other $k$-coverage algorithms
Thank You!

Questions??

- Details are available in the extended version of the paper at:

http://www.cs.sfu.ca/~mhefeeda