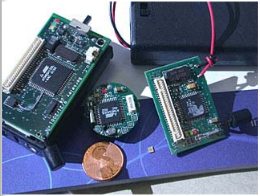
Minimum-Latency Aggregation Scheduling for Wireless Sensor Networks under the SINR Model

Hongxing Li, Qiangsheng Hua, Chuan Wu and Francis C.M. Lau

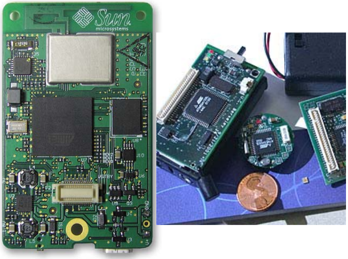
The University of Hong Kong

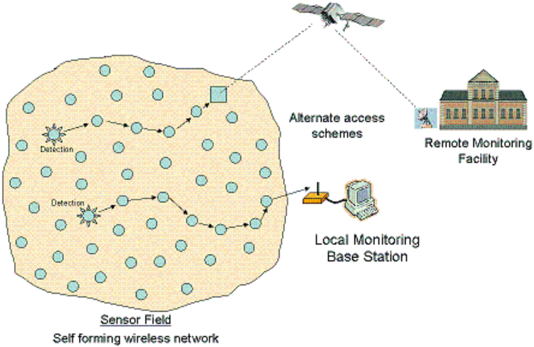
October 22, 2009

Wireless Sensor Network



Wireless Sensor Network





Applications of Wireless Sensor Networks

- **Military Applications**: distinguish ally and enemy, monitor the battle field, et.al.

Applications of Wireless Sensor Networks

- **Military Applications**: distinguish ally and enemy, monitor the battle field, et.al.
- **Environmental Applications**: forest fire, chemical pollution, temperature, humidity, et.al.

Applications of Wireless Sensor Networks

- **Military Applications**: distinguish ally and enemy, monitor the battle field, et.al.
- **Environmental Applications**: forest fire, chemical pollution, temperature, humidity, et.al.
- Others.

Data Aggregation in Wireless Sensor Networks

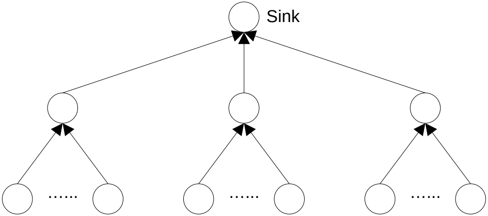


Figure: Data Aggregation.

Data Aggregation in Wireless Sensor Networks

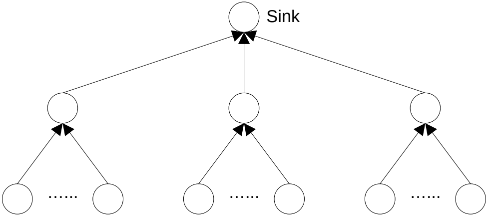


Figure: Data Aggregation.

Aggregation should be done in a timely fashion!

Definition (Minimum-Latency Aggregation Scheduling)

How shall one effectively schedule the aggregation transmissions in a wireless sensor network, such that no interference may occur and the total slots of time used for aggregation (referred to as *aggregation latency* hereinafter) is minimized?

Definition (Minimum-Latency Aggregation Scheduling)

How shall one effectively schedule the aggregation transmissions in a wireless sensor network, such that no interference may occur and the total slots of time used for aggregation (referred to as *aggregation latency* hereinafter) is minimized?

Graph Model

Binary interference relationship among concurrent transmitters: one transmission is successful if and only if its receiver is in the transmission range (R_t) of its transmitter and out of the interference range (R_i) of any other concurrent transmitter.

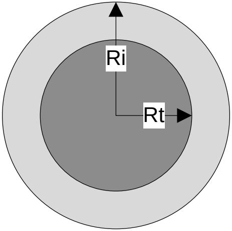


Figure: Graph Model

Graph Model

Binary interference relationship among concurrent transmitters: one transmission is successful if and only if its receiver is in the transmission range (R_t) of its transmitter and out of the interference range (R_i) of any other concurrent transmitter.

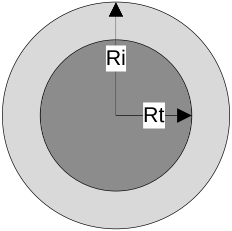


Figure: Graph Model

Problem: Too ideal, not realistic!

Signal-to-Interference-plus-Noise-Ratio (SINR) Model

$$SINR_i = \frac{P_i/d_{ii}^{\alpha}}{N_0 + \sum_{e_j \in \Lambda - \{e_i\}} P_j/d_{ji}^{\alpha}} \geq \beta$$

Here, Λ denotes the set of links that transmit simultaneously with e_i. P_i and P_j denote the transmission power at the transmitters of link e_i and e_j, respectively. d_{ii} (d_{ji}) is the distance between transmitters of link e_i (e_j) and the receiver of link e_i. α represents the path loss ratio, with a typical value between 2 and 6. N_0 is the ambient noise. β is the SINR threshold for a successful transmission, which is at least 1.

Signal-to-Interference-plus-Noise-Ratio (SINR) Model

$$SINR_i = \frac{P_i/d_{ii}^{\alpha}}{N_0 + \sum_{e_j \in \Lambda - \{e_i\}} P_j/d_{ji}^{\alpha}} \geq \beta$$

Here, Λ denotes the set of links that transmit simultaneously with e_i. P_i and P_j denote the transmission power at the transmitters of link e_i and e_j, respectively. d_{ii} (d_{ji}) is the distance between transmitters of link e_i (e_j) and the receiver of link e_i. α represents the path loss ratio, with a typical value between 2 and 6. N_0 is the ambient noise. β is the SINR threshold for a successful transmission, which is at least 1.

Challenge: Global information is required!

Current works on MLAS problem are all conducted on the basis of graph model.

- Chen et al. proved the NP-hardness of MLAS problem and proposed an aggregation scheduling algorithm with latency bound of $(\Lambda - 1)R$.
- Huang et al., for the first time, converted Λ from a multiplicative factor into an additive one. The scheduling algorithm builds on the basis of maximal independent set.
- Yu et al. presented the first distributed aggregation scheduling algorithm for MLAS problem. The aggregation latency is bounded by $O(\Lambda + R)$.

Here, Λ is the maximal node degree and R is the network radius.

Although there is no work on MLAS problem under the SINR model, there are a bunch of interesting results considering the Minimum-Length link Scheduling (MLS) problem with SINR constraints.

- Moscibroda, for the first time, gave a scaling law that describes the achievable data rate in worst-case sensor networks. A data gathering algorithm integrated with link scheduling, which maintains a $O(\log^2 n)$, is presented.
- In another work, Moscibroda et al. proposed a new measurement called "disturbance" to address the difficulty of finding a short schedule.
- Goussevskaia et al. proved the NP-completeness of a special case of the MLS problem.
- Fu. et al. introduced consecutive transmission constraints into MLS problem and proved the NP-hardness of that issue.

Problem Model

Definition (Minimum-Latency Aggregation Scheduling)

Given an arbitrarily located set of nodes V and sink node v_n, construct an aggregation tree $G = (V, E)$ and a link schedule $S = \{S_0, S_1, \ldots, S_{T-1}\}$, which meet the constraints that $\bigcup_{t=0}^{T-1} S_t = E$, for each $i \neq j$, $S_i \cap S_j = \emptyset$ and for each $i < j$, $T(S_i) \cap R(S_j) = \emptyset$, such that T is minimized and there is no collision under the SINR model.

The aggregation scheduling algorithm should be composed of two parts.

- Data aggregation tree construction.
- Link Scheduling.

Nearest-Neighbor Aggregation Scheduling (*NN-AS*) algorithm

Nearest-Neighbor Aggregation Scheduling (*NN-AS*) algorithm

- Executed phase by phase.

Nearest-Neighbor Aggregation Scheduling (*NN-AS*) algorithm

- Executed phase by phase.
- In each phase, every node v_i in V checks the status of its nearest neighbor v_j.

Nearest-Neighbor Aggregation Scheduling (*NN-AS*) algorithm

- Executed phase by phase.
- In each phase, every node v_i in V checks the status of its nearest neighbor v_j.
  - If v_j has been involved in a link in current phase, skip v_i for next node.

Nearest-Neighbor Aggregation Scheduling (*NN-AS*) algorithm

- Executed phase by phase.
- In each phase, every node v_i in V checks the status of its nearest neighbor v_j.
  - If v_j has been involved in a link in current phase, skip v_i for next node.
  - Otherwise, add link e_{ij} into E.

Nearest-Neighbor Aggregation Scheduling (*NN-AS*) algorithm

- Executed phase by phase.
- In each phase, every node v_i in V checks the status of its nearest neighbor v_j.
  - If v_j has been involved in a link in current phase, skip v_i for next node.
  - Otherwise, add link e_{ij} into E.
- At the end of each phase, all nodes been selected as transmitter in this phase are removed from V.

Algorithm 1 Centralized Aggregation Scheduling (NN-AS)

Input: Node set V with sink v_n.
Output: Set of link sets E and link schedule S.

$m := 1$; $E := S := \emptyset$;
while $(|V/\{v_n\}| \neq 1)$
  $E_m := \emptyset$;
  for$(\forall v_i \in V/\{v_n\})$
    Find v_i's nearest-neighbor $v_j \in V/\{v_n\}$;
    if$(v_j \in T(E_m) \cup R(E_m))$
      continue;
    $E_m := E_m \cup \{e_{ij}\}$;
  $V := V/T(E_m)$; $E := E \cup E_m$; $m := m + 1$;
  $S := S \cup$ Phase-Scheduler(E_m);
$v_i :=$ only node in $V/\{v_n\}$; $E := E \cup \{\{e_{in}\}\}$; $S := S \cup \{\{e_{in}\}\}$;
Return E and S;.

Figure: Phase by phase tree construction with nearest-neighbor mechanism: phase 1.

Figure: Phase by phase tree construction with nearest-neighbor mechanism: phase 2.

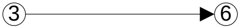


Figure: Phase by phase tree construction with nearest-neighbor mechanism: phase 3.

Cell Aggregation Scheduling (*Cell-AS*) algorithm

Cell Aggregation Scheduling (*Cell-AS*) algorithm

- Construct aggregation tree and schedule links phase by phase in ascending order of link length category.

Cell Aggregation Scheduling (*Cell-AS*) algorithm

- Construct aggregation tree and schedule links phase by phase in ascending order of link length category.
- For each link length category of k, the network is divided into numerous disjoint cells with side length of 3^k.

Cell Aggregation Scheduling (*Cell-AS*) algorithm

- Construct aggregation tree and schedule links phase by phase in ascending order of link length category.
- For each link length category of k, the network is divided into numerous disjoint cells with side length of 3^k.
  - One node is selected as head for each cell and all other nodes aggregation data to the head.

Cell Aggregation Scheduling (*Cell-AS*) algorithm

- Construct aggregation tree and schedule links phase by phase in ascending order of link length category.
- For each link length category of k, the network is divided into numerous disjoint cells with side length of 3^k.
  - One node is selected as head for each cell and all other nodes aggregation data to the head.
  - At the end of each phase, only head nodes remain in V for next phase.

Cell Aggregation Scheduling (*Cell-AS*) algorithm

- Construct aggregation tree and schedule links phase by phase in ascending order of link length category.
- For each link length category of k, the network is divided into numerous disjoint cells with side length of 3^k.
    - One node is selected as head for each cell and all other nodes aggregation data to the head.
    - At the end of each phase, only head nodes remain in V for next phase.
    - In the very end, only one node is left and it aggregate all data to the sink.

Algorithm 2 Aggregation Scheduling (Cell-AS)

Input: Node set V with sink v_n.
Output: Set of link sets E and link schedule S.

$k := 0$; $V := V/\{v_n\}$;
while $(|V| \neq 1)$
  Cover the network with cells of side length 3^k and color them with 16 colors;
  for$(i := 1$ to $16)$
    $E_i := \emptyset$;
    for(Each cell j with color i)
      Randomly select one node v_h in cell j as head;
      Connect all other nodes in cell j to v_h, add links to E_i and E, remove all nodes but v_h from V;
    $S := S \cup$ Cell-Scheduler(E_i);
  $k := k + 1$;
$v_h :=$ only node in V; $E := E \cup \{\{e_{hn}\}\}$; $S := S \cup \{\{e_{hn}\}\}$;
Return E and S;.

Algorithm 3 Cell Scheduler

Input: Link Set E_i.

Output: Link schedule S_i.

```
Define constant c such that c := Nβ / (1 − I_sum 2^α β);
t := 1; S_i := ∅;
while (E_i ≠ ∅)
    S_t := ∅;
    for(Each cell j with color i)
        Choose one non-scheduled link e_l in cell j; S_t := S_t ∪ {e_l}; E_i := E_i/{e_l};
        P_l := c × d_ll^α;
    S_i := S_i ∪ {S_t}; t := t + 1;
Return S_i;.
```

Where: $c := \frac{N\beta}{1 - I_{sum} 2^{\alpha}\beta}$; $t := 1$; $S_i := \emptyset$; while $(E_i \neq \emptyset)$: $S_t := \emptyset$; for(Each cell j with color i): Choose one non-scheduled link e_l in cell j; $S_t := S_t \cup \{e_l\}$; $E_i := E_i/\{e_l\}$; $P_l := c \times d_{ll}^{\alpha}$; $S_i := S_i \cup \{S_t\}$; $t := t + 1$; **Return** S_i;.

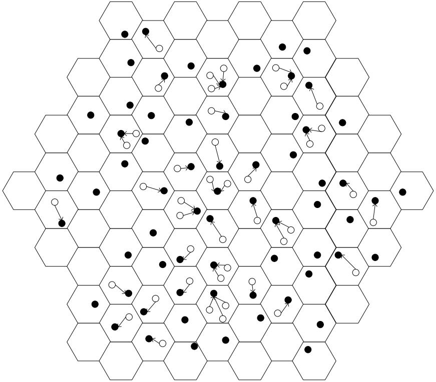


Figure: Tree construction with cells of different link length categories: category 0.

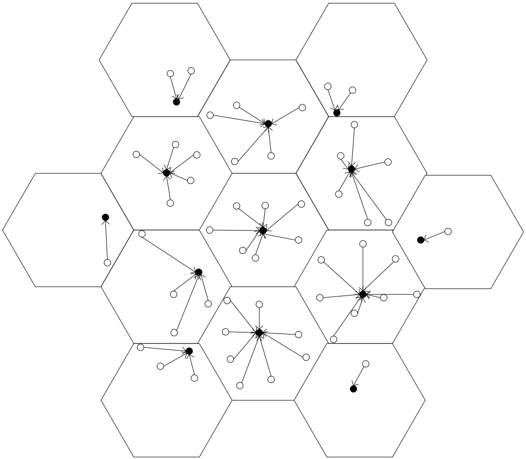


Figure: Tree construction with cells of different link length categories: category 1.

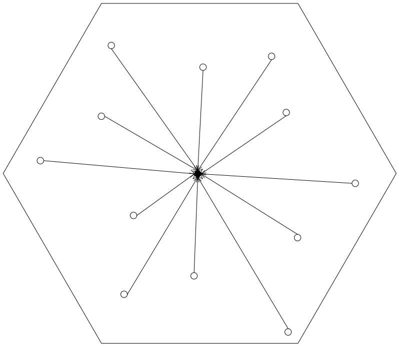


Figure: Tree construction with cells of different link length categories: category 2.

Theorem (Optimal Aggregation Scheduling Latency)

The optimal aggregation scheduling latency under any interference model is bounded by $\lceil \log n \rceil$.

Theorem (Centralized NN-AS Aggregation Latency)

The aggregation scheduling latency for centralized NN-AS is bounded by $O(\log^3 n)$ and the approximation ratio is bounded by $O(\log^2 n)$.

Theorem (Distributed Cell-AS Aggregation Latency)

The aggregation scheduling latency for distributed Cell-AS is bounded by $192K - 83$ and the approximation ratio is bounded by $(192K - 83)/\log n$.

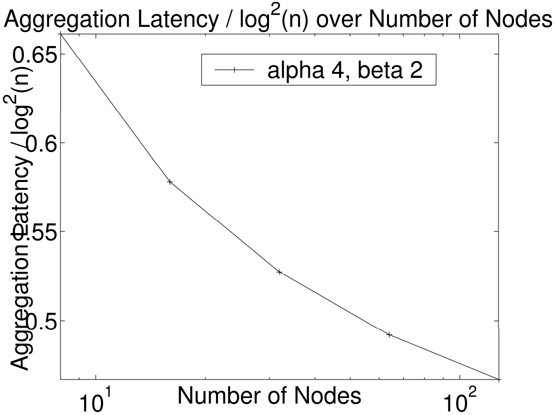


Figure: Aggregation Latency / $log^2 n$ over Number of Nodes (*NN-AS*).

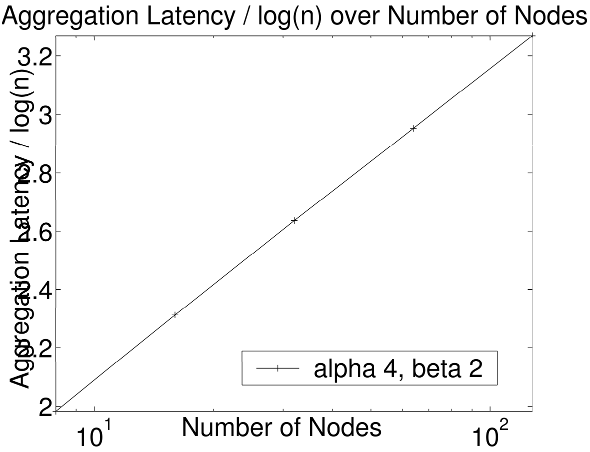


Figure: Aggregation Latency / $logn$ over Number of Nodes (*NN-AS*).

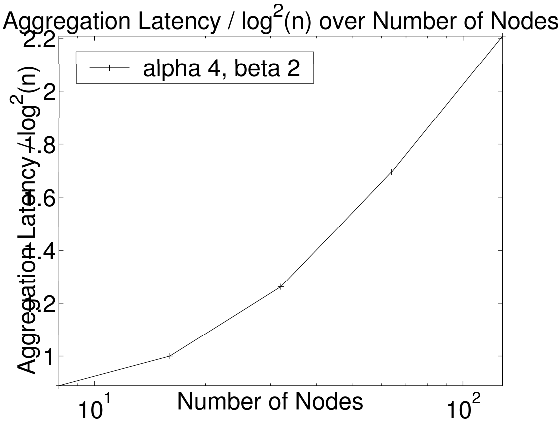


Figure: Aggregation Latency / $log^2 n$ over Number of Nodes (*Cell-AS*).

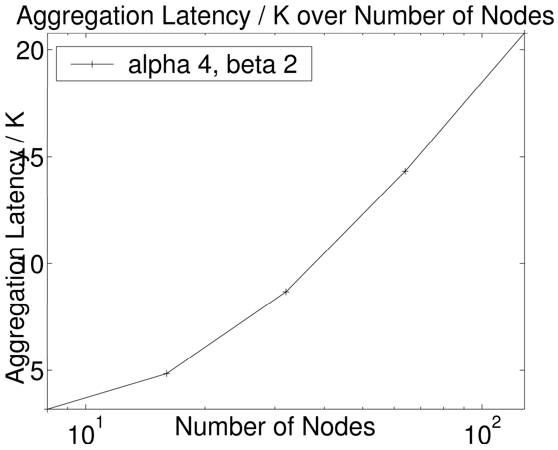


Figure: Aggregation Latency / $log^2 n$ over Number of Nodes (*Cell-AS*).

1. **Contributions:**
   - *Centralized joint aggregation tree construction and link scheduling algorithm*: Aggregation latency $O(\log^2 n)$, approximation ratio $O(\log^3 n)$.
   - *Distributed joint aggregation tree construction and link scheduling algorithm*: Aggregation latency $192K - 83$, approximation ratio upper-bounded by $(192K - 83)/\log n$.

1. **Contributions:**
   - *Centralized joint aggregation tree construction and link scheduling algorithm*: Aggregation latency $O(\log^2 n)$, approximation ratio $O(\log^3 n)$.
   - *Distributed joint aggregation tree construction and link scheduling algorithm*: Aggregation latency $192K - 83$, approximation ratio upper-bounded by $(192K - 83)/\log n$.
2. **Future Works:** Reduce the approximation ratio of centralized and distributed aggregation scheduling algorithms to $O(\log n)$ and $O(\log^2 n)$ respectively.

Thank You!