Distributed Data Aggregation Scheduling in Wireless Sensor Networks

Bo Yu, Jianzhong Li, School of Computer Science and Technology,
Harbin Institute of Technology, China
Email: bo yu@hit.edu.cn, lijzh@hit.edu.cn
Yingshu Li, Department of Computer Science,
Georgia State University, USA
Email: yli@cs.gsu.edu

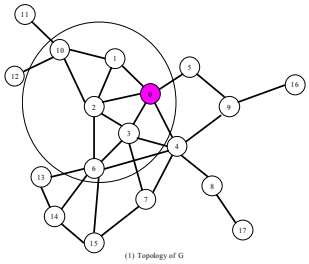
IEEE INFOCOM 2009, April 19–25, 2009, Rio de Janeiro, Brazil

Outline

Preliminaries-Transmission Collision Model

- Assumption in this paper is that communication is deterministic and proceeds in synchronous rounds controlled by a global clock.
- In each time round
  - any node can send data (be a sender) or receive data (be a receiver) but cannot do both;
  - data sent by any sender reaches simultaneously all its neighbors;
  - **all nodes have the same transmission range *r*=1**
  - a node receives a data only if the data is the only one that reaches it (in this round); and
  - each receiver updates its data as the combination of its old data and the data received (according to the aggregation function).

Preliminaries-Transmission Collision Model



all nodes have the same transmission range *r*=1.

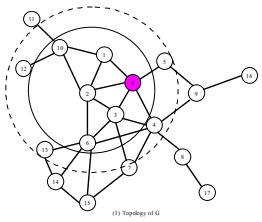
ρ is interference radius
ρ=*r*=1

2→0 conflicts with 11→10

11→10 conflicts with 12→10

2→0 conflicts-free with 7→4

Preliminaries-Transmission Collision Model



all nodes have the same transmission range.
$\rho > r$=1

2→0 conflicts with 11→10

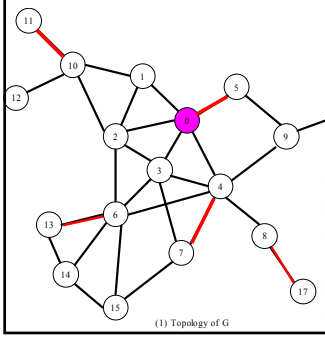
11→10 conflicts with 12→10

2→0 conflicts with 7→4

Research Problem(MLAS)

- The problem studied in this paper is the *Minimum-Latency Aggregation Scheduling* (MLAS) .
- MLAS Problem is defined as follows:
  - Given a wireless sensor network that consists of a number of sensors and a sink, supposing each sensor node has a piece of data to be aggregated, the MLAS problem is to design a transmission schedule for all sensor nodes such that there is no collision between any two concurrent transmissions and the total number of timeslots for all data to reach the sink is minimized.

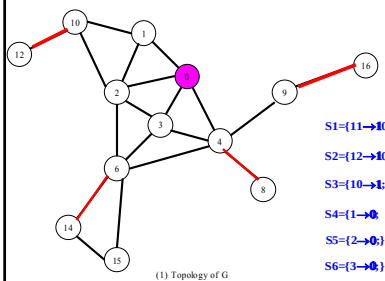
Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }
S2={12→10; 14→6; 8→4; 16→9; }
S3={10→1; 15→6; 9→4;}
S4={1→0; 6→4; }
S5={2→0;}
S6={3→0;}
S7={4→0;}

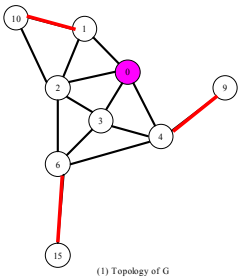
Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }
S2={12→10; 14→6; 8→4; 16→9; }
S3={10→1; 15→6; 9→4;}
S4={1→0; 6→4; }
S5={2→0;}
S6={3→0;}
S7={4→0;}

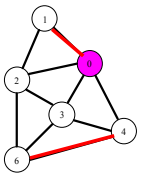
Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }
S2={12→10; 14→6; 8→4; 16→9; }
S3={10→1; 15→6; 9→4;}
S4={1→0; 6→4; }
S5={2→0;}
S6={3→0;}
S7={4→0;}

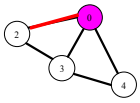
Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }
S2={12→10; 14→6; 8→4; 16→9; }
S3={10→1; 15→6; 9→4;}
S4={1→0; 6→4; }
S5={2→0;}
S6={3→0;}
S7={4→0;}

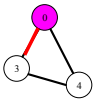
Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }
S2={12→10; 14→6; 8→4; 16→9; }
S3={10→1; 15→6; 9→4;}
S4={1→0; 6→4; }
S5={2→0;}
S6={3→0;}
S7={4→0;}

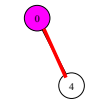
Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }
S2={12→10; 14→6; 8→4; 16→9; }
S3={10→1; 15→6; 9→4;}
S4={1→0; 6→4; }
S5={2→0;}
S6={3→0;}
S7={4→0;}

Data Aggregation Schedule



(1) Topology of G

S1={11→10; 13→6; 7→4; 17→8; 5→0; }

S2={12→10; 14→6; 8→4; 16→9; }

S3={10→1; 15→6; 9→4;}

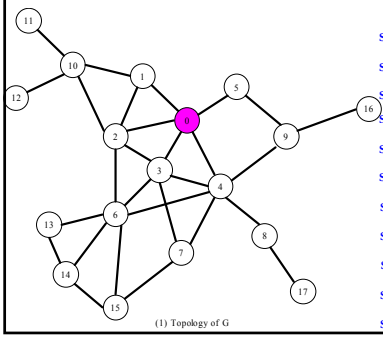
S4={1→0; 6→4; }

S5={2→0;}

S6={3→0;}

S7={4→0;}

Another Data Aggregation Schedule



(1) Topology of G

S1={5→0; 11→10; 13→6; 17→8; }

S2={16→9; 14→6; 12→10;}

S3={15→6}

S4={6→2; }

S5={10→1;}

S6={7→3;}

S7={7→4; }

S8={1→0;}

S9={2→0;}

S10={3→0;}

S11={4→0;}

The Data Aggregation Schedule

A data aggregation schedule is a sequence of sender sets $S_1, S_2, \cdots, S_l$ satisfying the following conditions:

1) $S_i \cap S_j = \emptyset, \forall i \neq j$;
2) $\bigcup_{i=1}^{l} S_i = V - \{r\}$;
3) Data are aggregated from S_k to $V - \bigcup_{i=1}^{k} S_i$ at time slot k, for all $k = 1, 2, \cdots, l$ and all the data are aggregated to the sink r in l time slots.

The Abstraction of MLAS Problem

- We further assume that all nodes have the same transmission range.
- We use a graph **G=(V, E)** to model a WSN.
  - **V** is a set vertex of **G**. It denotes sensor nodes in the WSN.
  - **E** is a edge set of **G**. For ***u***, ***v*∈V,** an edge exists from ***u*** to ***v*** if and only if ***v*** lies in ***u***'s transmission area (which is a disk).

The Abstraction of MDAT Problem

- Given a graph **G=(V, E),** a sink ***s*∈V** and interference ρ=r=1, find a distributed data aggregation schedule with minimum latency.

Back

Motivation

- All the previous aggregation scheduling algorithms for generating collision-free schedules are centralized which require the sink to compute the schedule and disseminate it to the sensors. Once all the sensor nodes receive the schedule, they work according to the schedule.
- Since topology changes often occur in sensor networks such as node failures, the sink has to gather new topology information from the network, recompute a schedule and disseminate it frequently.
- These processes consume lots of energy, which makes centralized algorithms inefficient.

Back

DAS-Solution

- This paper proposed the first distributed aggregation scheduling algorithm, named DAS, consists of two phases.
  - One is to construct a distributed aggregation tree(DAT). An existing distributed method [19] is adopted for the first phase.
  - Another one is to perform the distributed aggregation scheduling (SCHDL for short). The second phase is the key part of this algorithm.

Phase 1: Construct a distributed aggregation tree

[19] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *INFOCOM*, 2002.

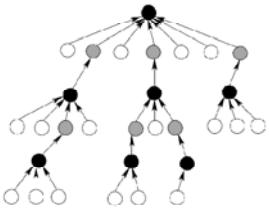
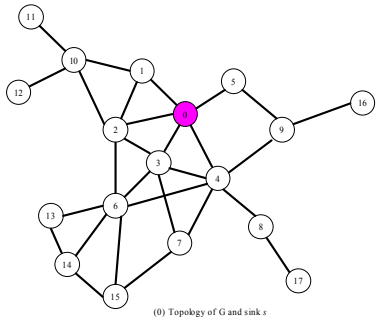


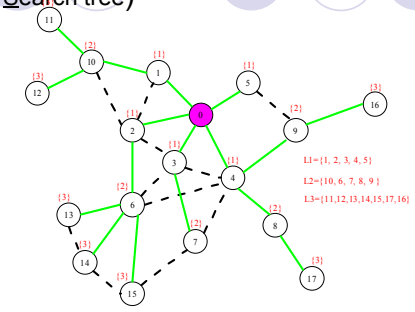
Fig. 1. The form of aggregation tree

Construct DAT (1- Input)



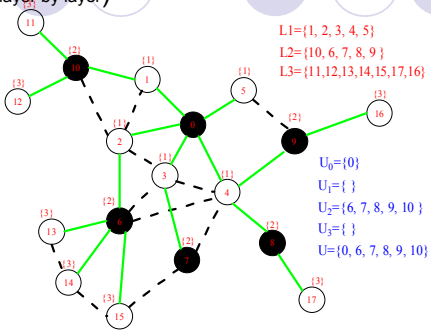
(0) Topology of G and sink *s*

Construct DAT (Step1 Construct Breadth First Search tree)



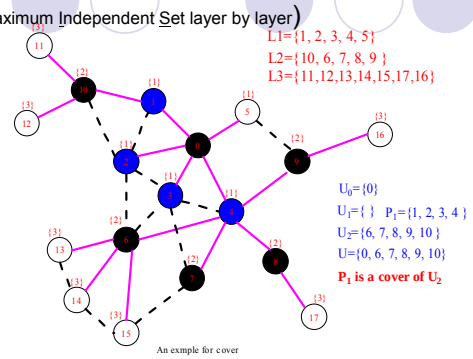
(1) BFS (Breadth First Search) tree of G

Construct DAT (Step 2 Construct Maximum Independent Set layer by layer)



(2) Construct a MIS (Maximal Independent Set) layer by layer

Construct DAT (Step 3 Find gray nodes to connect Maximum Independent Set layer by layer)



An exmple for cover

Phase 2: Distributed aggregation scheduling algorithm (SCHDL)

Definition 1: For any node u, a node is called a *competitor* of u if it cannot send data while u is sending data due to the collision. The set of all the competitors of u is called u's *competitor set*.

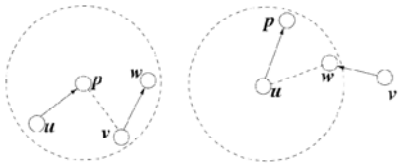


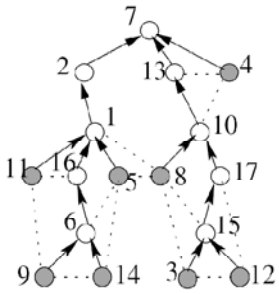
Fig. 2. Two cases of a collision

Compute u's competitor set

Proposition 1: For each node u, let $p(u)$ be u's parent. Let $N(u)$ be the set of u's 1-hop neighbors except $p(u)$ and let $Ch(u)$ be the set of u's children, then u's *competitor set* $= N(p(u)) \bigcup (\bigcup_{v \in N(u) \setminus Ch(u)} Ch(v)) \setminus \{u, p(u)\}$. ■

Easy to understand: For each node u, u's parent p's 1-hop neighbors and u's 1-hop neighbors' children constitute u's *competitor set*.

An Example for compute 11's competitor set



P(11)=1;
N(P(11))={2, 11, 16, 5, 8}
N(11)={1, 16, 9}
Ch(11)={ }
N(11)\Ch(11)={1, 16, 9}
Ch(1)={11, 16, 5}
Ch(16)={6}
Ch(9)={ }
M(11)=N(P(11))∪Ch(1) ∪Ch(16) ∪Ch(9)-{11, 1} ={2, 11, 16, 5, 8, 6}-{11,1}={2, 16, 5, 8, 6}

Proof the proposition

Proof: Suppose u is sending data to its parent p and a collision occurs. The collision must occur at node p or at another node w. Both cases are shown in Figure 2.

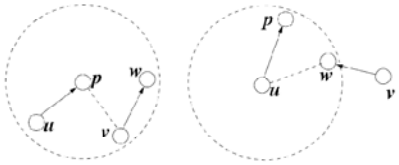


Fig. 2. Two cases of a collision

u's maintained info-(ID, M(u), K, $R_1(u)$, $R_2(u)$, x)

In the rest of the paper, we assume that each node u maintains the following information.

- u's unique ID.
- u's *competitor set*, denoted by $M(u)$.
- u's *earliest possible sending slot*, K, initialized to 1.
- $R(u)=R_1(u) \bigcup R_2(u)$, where $R_1(u) = \{v | v \in M(u)$ and is ready to make schedule$\}$, $R_2(u) = \{v | v \in M(u)$ and finished scheduling$\}$, and $R_1(u)$ and $R_2(u)$ are both initialized to a null set. For $\forall v \in R_1(u)$, v's *earliest possible sending slot* is maintained and for $\forall w \in R_2(u)$, w's schedule is maintained.
- The number x of u's children that have not scheduled, which is initialized to the number of u's children.

Sensor node behavior

a MARK message containing *u*'s ID and *K*

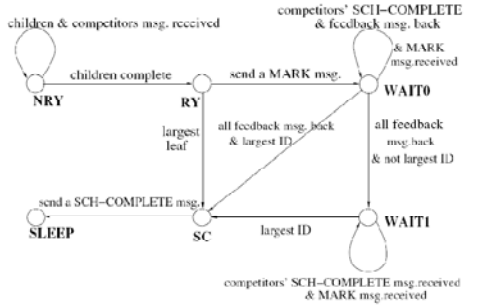


Fig. 3. Automaton of node behavior

The description of the SCHDL algorithm

1) Initially all the leaf nodes are set to state RY and the rest nodes are set to state NRY.
2) For each leaf u, if its ID is larger than IDs of all the nodes in $M(u) \cap \{leafs\}$, then multicast a SCH-COMPLETE msg. to $M(u) \cup \{p\}$, and u turns to sleep, else send a MARK msg. to $M(u)$ and turn to state WAIT0.

a MARK message containing *u*'s ID and *K*

3) For each node u, upon receiving a SCH-COMPLETE msg. from node v,
   a) if u is in state WAIT0, record the msg. as a feedback message and $R_2(u) \leftarrow R_2(u) \cup \{v\}$.
   b) if u is in state NRY, do
      i) if $v \in M(u)$, do $M(u) \leftarrow M(u) \setminus \{v\}$, $R_2(u) \leftarrow R_2(u) \cup \{v\}$.
      ii) if u is parent of v, do $x \leftarrow x-1$, $K \leftarrow max\{K, msg.K+1\}$.
      iii) if $x = 0$, do
         A) if $M(u)$ is empty, send a SCH-COMPLETE msg. to its parent p and turns to sleep;
         B) else send a MARK msg. to $M(u)$ and turn to state WAIT0.
   c) if u is in state WAIT1, do $R_1(u) \leftarrow R_1(u) \cup \{v\}$, $R_2(u) \leftarrow R_2(u) \cup \{v\}$ and check if u's ID is larger than IDs of all the nodes in $R_1(u)$. If so, call FIX-SCH(u); else u keeps in state WAIT1.

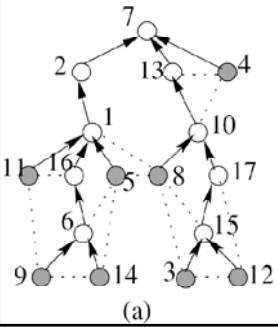
4) For a node u, upon receiving a MARK msg. from node v,
   a) $R_1(u) \leftarrow R_1(u) \cup \{v\}$;
   b) if $x = 0$, send a READY msg. containing K back to v;
   c) else send a NOT-READY msg. back to v.

5) For a node u, upon receiving a READY or NOT-READY msg. from node v,
   a) record the message as feedback.
   b) if it is a READY msg., do $R_1(u) \leftarrow R_1(u) \cup \{v\}$.
   c) if all feedback messages are received, check if u's ID is larger than IDs of all the nodes in $R_1(u)$. If so, call FIX-SCH(u); else turn to state WAIT1.
6) If the root w comes into RY state, SCHDL ends.

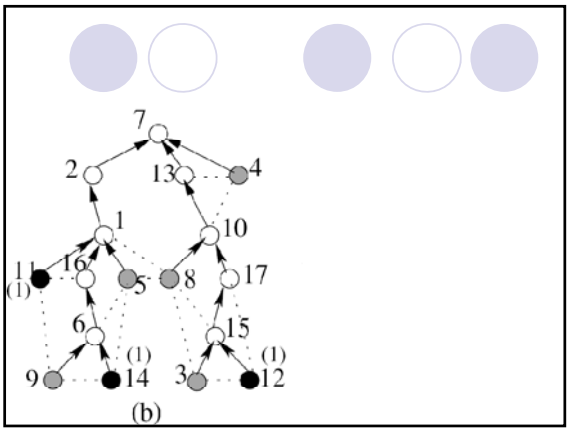
Algorithm 1 FIX-SCH(u)

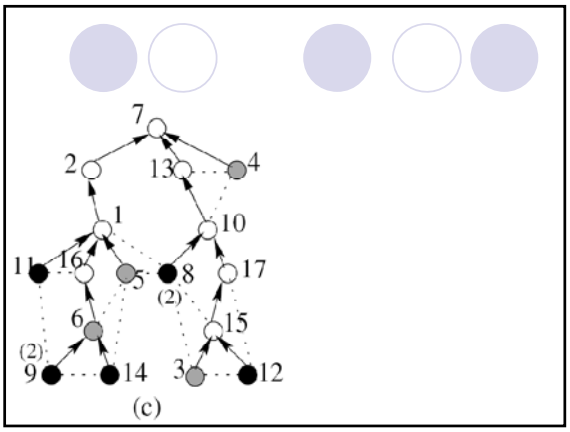
```
1: sch ← u.K
2: while (true) do
3:    if ∃ node v ∈ R2(u), v.schedule = sch then
4:       sch ← sch + 1
5:    else
6:       u.schedule ← sch
7:       Send a SCH-COMPLETE msg. and u.schedule to M(u)∪
         {p}, then let u go to sleep and return;
8:    end if
9: end while
```

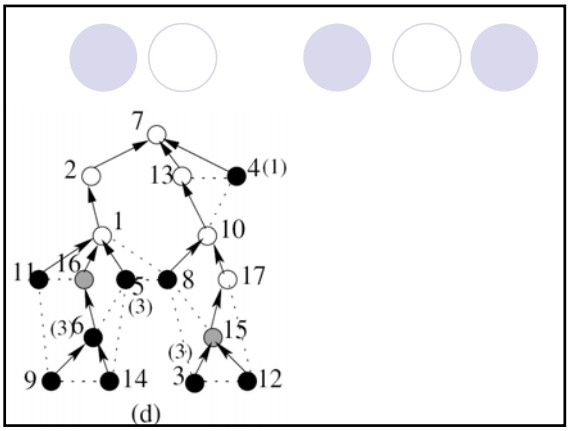
A example of SCHDL

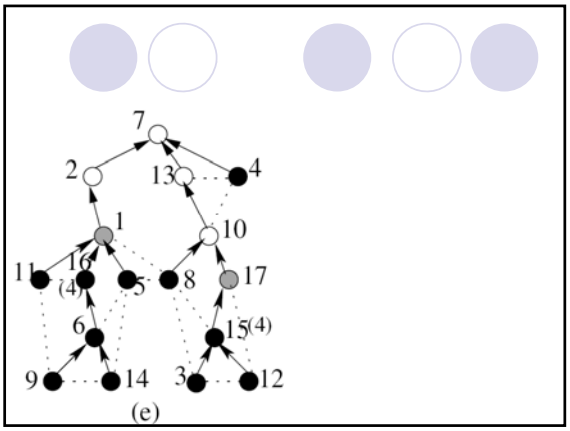


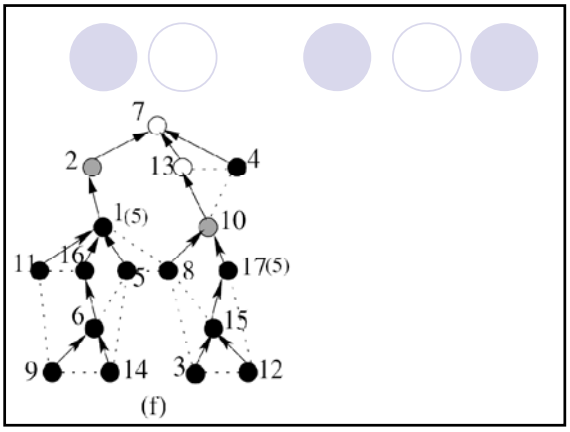
(a)

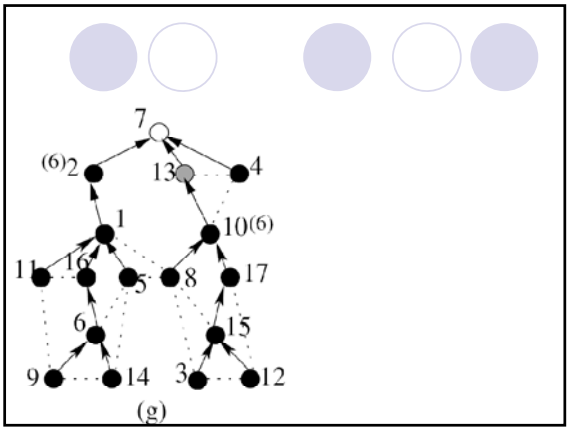

7
2
13
4
1
10
11
16
5
8
17
(1)
6
15
(1)
(1)
9
14
3
12
(b)

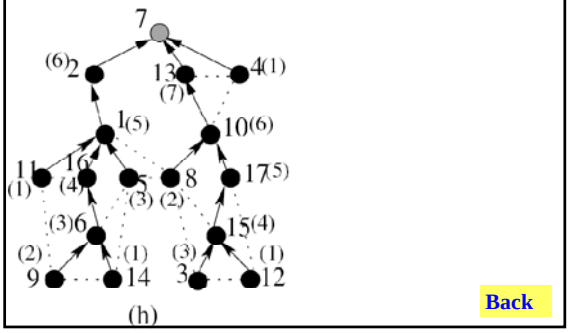

7
2
13
4
1
10
11
16
5
8
17
(2)
6
15
(2)
9
14
3
12
(c)


7
2
13
4(1)
1
10
11
16
5
8
17
(3)
(3)6
15
(3)
9
14
3
12
(d)


7
2
13
4
1
10
11
16
(4)
5
8
17
6
15(4)
9
14
3
12
(e)


7
2
13
4
1(5)
10
11
16
5
8
17(5)
6
15
9
14
3
12
(f)


7
(6)2
13
4
1
10(6)
11
16
5
8
17
6
15
9
14
3
12
(g)



Time Latency Analysis

Lemma 1: For any node u, if u's schedule is set to t by SCHDL and its *earliest possible sending slot* is K when it is ready in SCHDL, then $t \leq K + |M(u)|$.

Proof: Suppose a node u's *earliest possible sending slot* is K when it is ready in SCHDL and its schedule is set to t by SCHDL. According to the SCHDL algorithm, when FIX-SCH(u) is called, the first available sending slot after K is assigned to u's schedule. Referring to FIX-SCH, a time slot is available for u if and only if it is not equal to the schedule of any node in $R_2(u)$. Thus u's schedule $t \leq K + |R_2(u)|$ since the case for u's latest schedule is that the schedules of the nodes in $R_2(u)$ are $K, K+1, \cdots, K+|R_2(u)|-1$. As $R_2(u)$ is a subset of $M(u)$, $t \leq K + |M(u)|$. ■

Let t_i be the latest schedule of the nodes in layer i, $i = 1, 2, \cdots, m$, where m is the number of the deepest layer. It is obvious that the whole latency $T = t_1$.

The largest size, Z, of the *competitor sets* of all the white nodes.

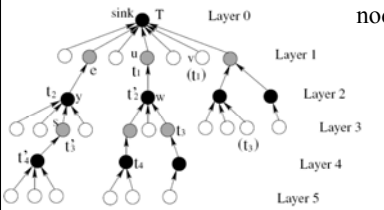


Fig. 5. Analysis on an aggregation tree

$$t_1 \leq \begin{cases} 20 + t_2, & \text{all white nodes finish sending at } t_2 + 1; \\ 20 + Z, & \text{otherwise.} \end{cases}$$

The largest size, Z, of the *competitor sets* of all the white nodes.

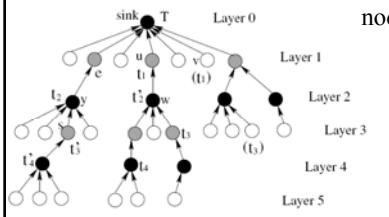


Fig. 5. Analysis on an aggregation tree

$$t_2 \leq \begin{cases} 5 + t_3, & \text{all white nodes finish sending at } t_3 + 1; \\ 5 + Z, & \text{otherwise.} \end{cases}$$

The largest size, Z, of the *competitor sets* of all the white nodes.

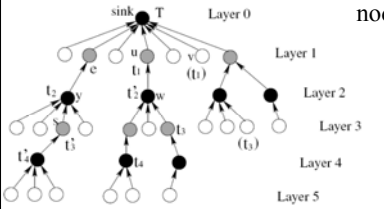


Fig. 5. Analysis on an aggregation tree

$$t_{2k-1} \leq \begin{cases} 19 + t_{2k}, & \text{all white nodes finish sending at } t_2 + 1; \\ 19 + Z, & \text{otherwise.} \end{cases}$$

The largest size, Z, of the *competitor sets* of all the white nodes.

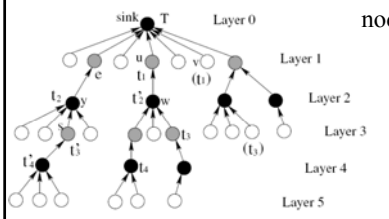


Fig. 5. Analysis on an aggregation tree

$$t_{2k} \leq \begin{cases} 5 + t_{2k+1}, & \text{all white nodes finish sending at } t_2 + 1; \\ 5 + Z, & \text{otherwise.} \end{cases}$$

Estimation of Z

the largest size, Z, of the *competitor sets* of all the white nodes

Lemma 2: In a UDG model, for an arbitrary white node u in the aggregation tree, the size of its *competitor set* is at most $20 - 6\Delta$, where Δ is the maximum node degree.

Back

An Adaptive Scheduling Method

- Maintenance of Aggregation Tree
  - New Nodes Joining
  - Node Failures
- Adaptive Scheduling using DAS

Back

New Nodes Joining

- When a node u joins in a network, it sends a JOIN message.
- All the nodes in u's transmission area receive the message.
  - For any node v receiving the JOIN message, it sends back an ACK message including its node color.
  - After u collects all the ACK messages from its neighbors, it checks if there are messages from black nodes.
    - If so, it picks any one, say w, as its parent and sends a CHILD message to w. Node u becomes a leaf and its color is set to white.
    - Otherwise u has no black neighbors, and u makes itself a black node. Since the network is connected, u must have at least a white or gray neighbor. u can randomly picks one as its parent and sends a CHILD message. Upon receiving a CHILD message, a node records the sender as its child and checks if it is a white node. If so, it turns itself to a gray one.

Back

Node Failures

- Typically node failures are detected by the mechanism of sending confirming messages periodically.
  - If node v sends a confirming message to node u and has not received u's response for a predefined time duration, v believes that u fails.

Node Failures

- Assume u fails.
- If u is a leaf node, its parent p will find out that u fails after a while. In this case, p removes u from its child list.

Node Failures

- If u is a gray node, its parent p and its black children will find out after a while.
  - p deletes u from its child list.
  - For any black child v of u we adopt the following strategy to find a parent for v.
    - First, v finds a white or gray neighbor w, which is not its child, as its parent and sends a CHILD message to node w.
    - If v cannot find one, v turns its color to white and finds its parent.
    - v's white children also need to find parents for themselves. The white nodes can be seen as the new joining nodes during the finding of their parents.
    - v's gray children find their parents using the strategy proposed as follows.

Node Failures

- If u is a black node, u's parent p deletes u from its child list and checks if it has any other black children. If not, p turns itself to a white node.
- For any white child of u, again we take it as a new node joining in.
- For any gray child v of u, v finds its parent in the following way.
  - First, v checks if there are black neighbors who are not its children. If so, v randomly chooses one from them as its parent and sends to it a CHILD message.
  - Otherwise, v randomly chooses one of its children w, which must be black, as its parent and sends to w a CHILD message.
  - v's black children lose their parents and use the strategy above of finding the parent for a black node to find their parents. As the process is recursive, any node who loses its parent can find a new one as long as the network is connected.

Back

Adaptive Scheduling using DAS

- After the aggregation tree has been updated, the nodes who have changed their parents compute their new *competitors* locally and send their updated information to their competitors.
- All the nodes whose parents have changed form a set Upd.
- For each node u in Upd, it marks itself *renewed* and sends a RENEW message to its parent.
- For each node v receiving a RENEW message, if it is not a child of the sink, it marks itself *renewed* and sends a RENEW message to its parent. This is because if a node has some new child, it may need to change its schedule and so do its parent and grandparent and so on.
- Then the *unrenewed* nodes send their schedules to their *renewed competitors*.
- After these processes, the *renewed* nodes starts to run SCHDL of DAS.

Back

Simulation Results

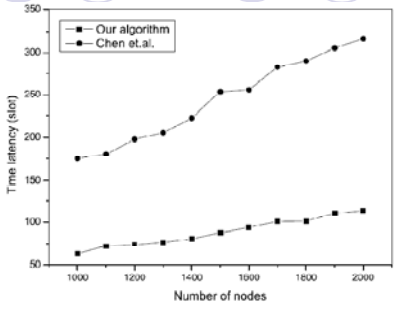


Fig. 6. Latency with different number of nodes

Simulation Results

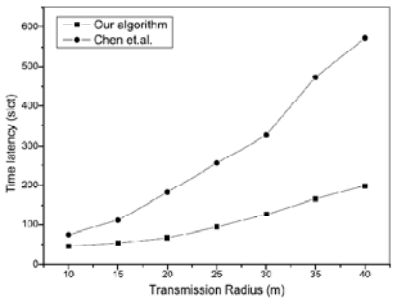


Fig. 7. Latency with different transmission radius

Simulation Results

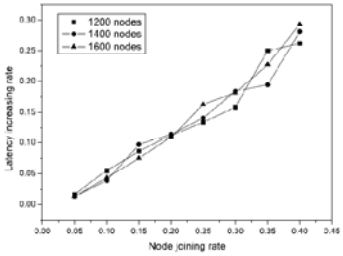


Fig. 10. Latency increments with different joining rate

Simulation Results

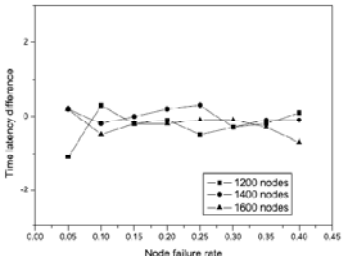


Fig. 12. Latency differences with different failure rate

Simulation Results

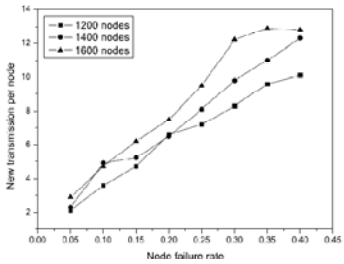


Fig. 13. Transmission increments with different failure rate

Back

Related Reference

- **[1] X. H. X. Chen, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," presented at 1st Int'l Conference on Mobile Ad-hoc and Sensor Networks–MSN'05, 2005.**
- **[2] P.-J. W. S. C.-H. Huang, C. T. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," presented at Proc. IEEE INFOCOM, 2007, 2007.**
- **[3]S. K. S. G. V. Annamalai, and L. Schwiebert, "On tree-based convergecasting in wireless sensor networks," presented at IEEE Wireless Communications and Networking–WCNC'03, 2003.**
- **[4] BO YU, J. L., AND LI, Y. Distributed Data Aggregation Scheduling in Wireless Sensor Networks. In *IEEE INFOCOM* (2009).**
- **[5] P.-J. Wan, C.-H. Huang, L. Wang, Z.-Y. Wan, and X. Jia, Minimum-Latency Aggregation Scheduling in Multihop Wireless Networks, ACM MOBIHOC 2009, pp. 185-194.**

Related Work

- Time complexity of [1] is (Δ-1)R
- Time complexity of [2] is 23R+Δ-18
- Time complexity of [3] is log(n), where n is the number of sensor nodes in WSNs. In their model, they assume each node can vary its transmission range to reduce links.
- Time complexity of [4] is $24D+6\Delta+16$, where D is the network diameter.
- Time complexity of [5] is
  - $\rho=1$
    - SAS (Sequential Aggregation Scheduling)
      - $15R+\Delta-4$
    - PAS (Pipelined Aggregation Scheduling)
      - $2R+\Delta+O(\log R)$
    - E-PAS (Enhanced Pipelined Aggregation Scheduling)
      - $(1+O(\log R)/R^{1/3})R+\Delta$
  - $\rho>1$
    - $\beta_{\rho+1}l$

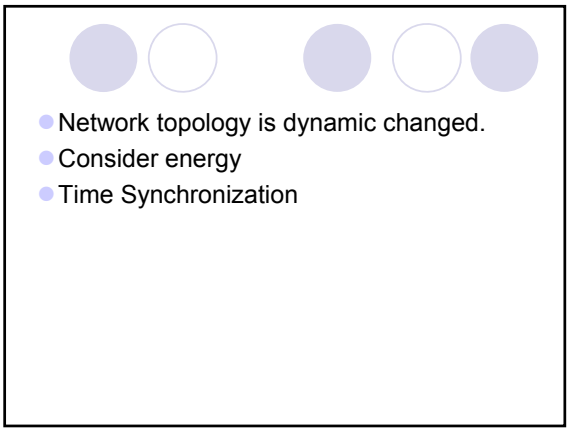
Back

Conclusion

- This paper studies the problem of minimum-latency aggregation scheduling and proposes the first distributed scheduling algorithm with latency bound $24D + 6\Delta + 16$.
- Motivation
- Solution and Theoretical Analysis
  - SCHDL
  - Time Latency Analysis
  - An Adaptive Scheduling Method
- Simulation Results
- Related Work

Q&A

Thank you very much!



- Network topology is dynamic changed.
- Consider energy
- Time Synchronization