DTM: A Distributed Traffic Management Algorithm for Sensor Networks

Yantao Pan, Xicheng Lu, Peidong Zhu

School of Computer, National University of Defense Technology, Changsha, P. R. China

Summary
Lifetime optimization is a key challenge of sensor networks. Since data transmission is the main energy consumer, it is important to make use of energy efficient routing protocols. We regard the lifetime optimization problem as a multi-source single-sink max flow problem in a directed graph with capacity powers on vertices. Then we propose a distributed algorithm to solve this problem. The method gives the value of maximum lifetime exactly.

Key words:
Sensor Networks, Traffic Management, Lifetime maximization

1. Introduction

The development of sensor networks provides us a new way to interact with physical environments. It is a key challenge to maximize the lifetime of a sensor network, which is usually defined as the time from its deployment to its partition when there exits a node that cannot send data to the sink. Literatures [1-4] investigate the upper bound and the expectation of a sensor network's maximum lifetime. Literatures [5-8] propose heuristic algorithms to maximize the lifetime approximately. However, none of those approaches provides the maximum lifetime exactly. In this paper, we exploit this problem by max flow model and propose a distributed algorithm to achieve a maximum lifetime routing.

The rest of this paper is organized as follows. In section 2, we propose the formulation of the problem. We give a centralized algorithm in section 3 and then a distributed algorithm in section 4. In section 5, some concluding remarks are made.

2. Problem Statement and Formulation

Consider a group of wireless static sensor nodes V randomly distributed in a region. Each node $v \in V$ has a limited battery energy supply which is mainly used for data communications. Assume that each node has a fixed communication power and a fixed transmission radius. Let $p(v)$ be the total quantity of data that vertex v can send. It is a function of the initial energy supply and the communication power. Let $w(v)$ be the data-generating rate at a source node v. A sensor network can be defined as a directed graph with powers on vertices.

Definition-1. A sensor transportation network (It will be showed as SensorNet for short in the rest of this paper.) is defined as the form of $N^s = (V, A, p, X, t)$, where

1. $G(V, A)$ is a connected simple directed graph, and
2. $p(v): V \mapsto \overline{\square^{-}}$ is the residuary transmission capacity of vertex v, and
3. X is the source set and t is the sink.

Let $\alpha(H) = \{(u,v) \in A \mid v \in H, u \in V \setminus H\}$ and $\beta(H) = \{(v,u) \in A \mid v \in H, u \in V \setminus H\}$ be the sets of arcs entering and leaving a node set H. For a function $f(a): A \mapsto \square^{-}$, we denote $f^{\alpha}(H) \mathrel{\hat{=}} \sum_{a \in \alpha(H)} f(a)$ and $f^{\beta}(H) \mathrel{\hat{=}} \sum_{a \in \beta(H)} f(a)$.

Definition-2. $f(a): A \mapsto \overline{\square^{-}}$ is defined as a flow of N^s if $f^{\beta}(v) \le p(v), \forall v \in V$ and $f^{\beta}(v) = f^{\alpha}(v), \forall v \in I$. Let $val(f) \mathrel{\hat{=}} f^{\beta}(X) - f^{\alpha}(X)$ be the value of f.

Definition-3. Let N^s be a SensorNet. Flow f is defined as a routing of N^s if there is a T such that $f^{\beta}(x) - f^{\alpha}(x) = T \cdot w(x), \forall x \in X$ and $p(v) = f^{\beta}(v), \exists v \in V$. T is called the lifetime of N^s under the routing f. If there is no f' and its corresponding T' so as to $T' > T$, f is a maximum lifetime routing and T is the maximum lifetime.

For a SensorNet $N^s = (V, A, p, X, t)$, we may add a virtual source s to it and add a virtual arc from it to each source. Additionally, we set the capacity of a virtual arc to be $T \cdot w(v)$ corresponding to vertex v, where T is a value not more than the maximum lifetime. Then we can transfer the maximum flow problem from a multi-source network to a single-source network.

Manuscript received August 5, 2006.
Manuscript revised August 25, 2006.

Definition-4. Let $N^s = (V, A, p, X, t)$ be a SensorNet and assume $T > 0$. $N' = (V', A', p', c', s, t)$ is defined as the transformed network of N^s about T , if following 4 conditions are satisfied.

1. $V' = V \bigcup \{s\}$.
2. $A' = A \bigcup A_s$, where $A_s = \{(s, x) \mid x \in X\}$.
3. $p'(v) = \begin{cases} p(v), v \in V, \\ \infty, otherwise. \end{cases}$.
4. $c'(u, v) = \begin{cases} T \cdot w(v), u = s, \\ \infty, otherwise. \end{cases}$

Lemma-1. Let N^s be a SensorNet and let T^* be its maximum lifetime. Suppose N' is the transformed network of N^s about T , and $f'_{\max}$ is a maximum flow of N' . Then, $T \le T^* \Leftrightarrow val(f'_{\max}) = \sum_{x \in X} T \cdot w(x)$.

Theorem-1. Suppose N' is the transformed network of N^s about T and $f'_{\max}$ is a maximum flow of N_T^s . Then T^* is the maximum T that satisfies $val(f'_{\max}) = \sum_{x \in X} T \cdot w(x)$.

Suppose $\{T_i\}, i = 0,1,2,...$ is a sequence, where $T_0 = 0$ and T_1 is an upper bound of the maximum lifetime. The rest elements are generated by BS algorithm, where ξ is the given precision and $\theta(T) = \begin{cases} 1, val(f'_{\max}) = \sum_{x \in X} T \cdot w(x), \\ 0, val(f'_{\max}) < \sum_{x \in X} T \cdot w(x). \end{cases}$.

```
BS ($T_0$,$T_1$)
$T_{low} \leftarrow T_0$ ; $T_{up} \leftarrow T_1$ ; $i \leftarrow 0$ ;
while $T_{up} - T_{low} > \xi$ {
    $T_{i+2} \leftarrow (T_{up} - T_{low})/2$ ;
    if $\theta(T_{i+2}) = 0$
        $T_{up} \leftarrow T_{i+2}$ ;
    else
        $T_{low} \leftarrow T_{i+2}$ ;
    $i = i + 1$ ;
}
return $(T_{up} - T_{low})/2$ ;
```

Fig.1 BS algorithm

From Theorem-1 and BS algorithm, we obtain $\lim_{i \to +\infty} \{T_i\} = T^*$. Therefore, we can take a binary search between the lower and upper bounds of the maximum lifetime and finally achieve a value exactly enough if only we could find the maximum flow of N' .

In order to transfer the constraints of N' from vertices to arcs, we divide a vertex into two virtual vertices and add a virtual arc between them. The capacity of a virtual arc is set to the residual transmission capacity of the corresponding vertex.

For a SensorNet $N^s = (V, A, p, X, t)$, we may add a virtual source s to it and add a virtual arc from it to each source. Additionally, we set the capacity of a virtual arc to be $T \cdot w(v)$ corresponding to vertex v

Definition-5. Let $N' = (V', A', p', c', s, t)$ be a transformed network of N^s . Then $N^T = (V^T, A^T, c^T, s_+, t_-)$ is defined as the VTA (Vertex to Arc) network of N^s , if the following 2 conditions are satisfied.

1. $V^T = V_+^T \bigcup V_-^T$, such that $v_+ \in V_+^T$ and $v_- \in V_-^T$ for each $v \in V'$.
2. $A^T = A_A^T \bigcup A_V^T$, such that
   1) $(u_+, v_-) \in A_A^T$, $c^T(u_+, v_-) = c'(u, v)$ for $\forall (u, v) \in A'$;
   2) $(v_-, v_+) \in A_V^T$, $c^T(v_-, v_+) = p'(v)$ for $\forall v \in V'$.

Theorem-2. Let $N^T = (V^T, A^T, c^T, s_+, t_-)$ be a VTA network and $N' = (V', A', p', c', s, t)$ be a transformed network of $N^s = (V, A, p, X, t)$. Assume f^T to be a maximum flow of N^T . Then f' is a maximum flow of N' , where $f'(u, v) \hat{=} f^T(u_+, v_-), \forall (u, v) \in A'$. And f is a maximum flow of N^s , where $f(u, v) \hat{=} f'(u, v), \forall (u, v) \in A$.

2. Centralized Algorithm

In this section, we propose a centralized algorithm based on preflow-push method introduced by Golberg-Tarjan [9] to solve the maximum flow problem in N^T .

Since there is only N^T in consideration in this section, we denote it as $N^T = (V, A, c, s, t)$ for short and denote the residual network of it as $N^0(f) = (V, A^0, c^0, s, t)$.

For a node $v \in V$, we call $ex(v) = f^\beta(v) - f^\alpha(v)$ the excess of v . It is the difference between the total flow entering and leaving node v . The flow conservation constraint is satisfied when all nodes have zero excesses except the source and the sink.

Definition-6. A function $\zeta : A \mapsto \overline{\mathbb{R}}$ is defined as a preflow if

1. $ex(v) \ge 0$ for $\forall v \in V \setminus \{s, t\}$, and
2. $0 \le \zeta(a) \le c(a)$ for $\forall a \in A$.

A node is active if its excess is positive.

The basic idea is to push the excess flow to the neighbors that are closer to the sink. To do this, a height function $h : V \mapsto \overline{\square^{-}}$ is assigned to each node. A valid height function satisfy: $h(t) = 0$ and $h(u) \le h(v) + 1$ for $\forall (u, v) \in A^0$. We only push flow from a higher node to a lower node.

Definition-7. A residual arc $(u, v) \in A^0, u \ne s$ is admissible if $h(u) = h(v) + 1$.

Let u be an active node and suppose (u, v) is admissible. We push Δ unit flow from u to v through (u, v), where $\Delta = \min\{ex(u), c^0(u, v)\}$. Then $ex(u) = ex(u) - \Delta$ and $ex(v) = ex(v) + \Delta$. If no more admissible arcs exist, we perform lift process to create admissible arcs. In this stage, a node that has no admissible outward arc sets its height to $\min\{h(v) + 1 | (u, v) \in A^0\}$. The algorithm ends when there is no active node. At that stage the flow is feasible for the first time and also maximized.

```
CA( $N^T$ )
$h(s) \leftarrow |V|$ ; $h(t) \leftarrow 0$ ;
Compute $h(v)$ for other nodes;
$ex(v) \leftarrow 0, \forall v \in V$ ; $\zeta(a) \leftarrow 0, \forall a \in A$ ;
for $\forall u \in Adj^+(s) \triangleq \{u | (s,u) \in A\}$
    $\zeta(s,u) \leftarrow c(s,u), ex(u) \leftarrow c(s,u)$ ;
while there exits active nodes{
    select an active node u;
    if there is admissible arcs{
        select an admissible arc $(u,v)$ ;
        $\Delta \leftarrow \min\{ex(u), c^0(u,v)\}$ ;
        $ex(u) \leftarrow ex(u) - \Delta$ ;
        $ex(v) \leftarrow ex(v) + \Delta$ ;
        $\zeta(u,v) \leftarrow \zeta(u,v) + \Delta$ ;
        $c^0(u,v) \leftarrow c^0(u,v) - \Delta$ ;
        $c^0(v,u) \leftarrow c^0(v,u) + \Delta$ ;
    }else
        $h(u) \leftarrow \min\{h(v) + 1 | (u,v) \in A^0\}$ ;
}//while
return $\zeta$
```

Fig.2 CA algorithm

Lemma-2. Let h be a valid height function. Then $h(u)$ is a lower bound on the distance from u to t in $N^0(\zeta)$.

Theorem-3. The CA algorithm is finite and terminates with the maximum s-t flow.

Proof. The algorithm is finite because the height value can increase at most $2|V|$ times each. The algorithm ends with a flow $f = \zeta$ because if there were any active node, the algorithm would not end. At the end, $h(s) = |V|$. Since $h(s)$ is a lower bound on the shortest $s - t$ path in $N^0(f)$, there is no $s - t$ path in $N(f)$. Then there is no $s - t$ admissible path in $N^0(f)$ and the flow is maximized.

Theorem-4. The time complexity of CA is $O(|V|^3)$ for $N^s = (V, A, p, X, t)$.

4. Distributed Algorithm

In this section, we propose an asynchronously distributed algorithm DA based on CA. The basic idea is introduced by [10]. The difference is that our approach can solve the maximum flow problem in multi-source networks with capacity constraints on vertices.

We assume that each node in the network executes a distributed program asynchronously. Each link is bidirectional. All processes are event-driven and work on local data. Communications are message based. We assume that any source node knows the number of sources $|X|$.

The execution of algorithm also proceeds in two phases. The first phase sets heights of all nodes. The second phase establishes the maximum flow. Different from the centralized algorithm presented in section 3, a residual arc (u, v) is admissible for node u if $h(u) \ge h(v) + 1$. Additionally, in lift process, we increase a node's height to a value which is just enough for it to push all its excess out.

Consider a SensorNet $N^s = (V, A, p, X, t)$. We regard a node $v \in V$ as two virtual nodes $v-$ and $v+$. $v-$ is called "In-node" and $v+$ is called "Out-node". They are connected to the neighbors of node v by incoming arcs and outgoing arcs respectively. At each node v, the program maintains the following information.

1. $Adj(v) = Adj(v-) \bigcup Adj(v+)$ is the neighbor set, where $Adj(v-) = \{u | (u, v) \in A\}$ and $Adj(v+) = \{u | (v, u) \in A\}$.
2. $h(v\pm)$ is the height of $v\pm$.
3. $ex(v\pm)$ is the excess of $v\pm$.
4. $h_{v\pm}(u)$ is u's height from the point of view of $v\pm$.
5. $ty(v)$ is the node's type.
6. $c_{v\pm}(u)$ is the residual capacity of arc $(v-, u+)$ or $(v+, u-)$. Especially $c_{v-}(v) = p(v)$ at the beginning.

If v is a source, it has to maintain following information about the virtual source s.

1. *InitOk* denotes if all sources are initialized already.

2. $h_{v-}(s)$ and $c_{v-}(s)$ for v.
3. $ex(s+)$ and $h(s+)$.
4. $h_{s+}(u)$ and $c_{s+}(u)$ for all $u \in V$.

The algorithm terminates at any of two situations. The first is when all nodes are inactive. This means there are no more messages will be exchanged. The second is when a source pushes its excess back to the virtual source. This means the source cannot transmit $T \cdot w$ data to the sink. As to BS algorithm, T is larger than the maximum flow and more computation is useless. Therefore, we terminate the algorithm. The details of DA are described as follows.

```
MainOnNode( $v$ )
Initialization( $v$ );
while (1){
  wait for incoming msg;
  process msg;
}//while
```

Fig.3 MainOnNode

```
Initialization( $v$ )
 $ex(v+) \leftarrow 0$ ; $ex(v-) \leftarrow 0$ ;
 $h(v+), h_{v-}(v) \leftarrow 0$ ; $h(v-), h_{v+}(v) \leftarrow 0$ ;
 for all $u \in Adj(v+)$
    $c_{v+}(u) \leftarrow M$ [1];
 for all $u \in Adj(v-)$
    $c_{v-}(u) \leftarrow 0$ ;
 $c_{v-}(v) \leftarrow p(v)$ ; $c_{v+}(v) \leftarrow 0$ ;
 $h_{v+}(u) \leftarrow 0$ , $h_{v-}(u) \leftarrow 0$ for all $u \in V$ ;
 if $ty(v) = Source$ {
   $InitOk \leftarrow 0$ ;
   $Adj(v-) \leftarrow Adj(v-) \cup \{s\}$ ;
   $h_{v-}(s) \leftarrow 0$ ; $c_{v-}(s) \leftarrow 0$ ;
   $ex(s+) \leftarrow 0$ ; $h(s+) \leftarrow 0$ ;
   $h_{s+}(u) \leftarrow 0$ , $c_{s+}(u) \leftarrow 0$ for all $u \in V$ ;
 }//if
 if $ty(v) = Sink$ {
   $h(v-) \leftarrow 0$ ;
   InitHight( $v$ );
 }//if
```

Fig.4 Initialization

```
InitHight( $v$ )
 for all $u \in Adj(v+)$
   Send(NEW-HEIGHT, $h(v+)$) to u;
 for all $u \in Adj(v-)$
   Send(INIT-HEIGHT, $h(v-)$) to u;
 if $ty(v) = Source$ {
    $ex(v-) \leftarrow T \cdot w(v)$ ;
    $c_{v-}(s) \leftarrow T \cdot w(v)$ ;
    Broadcast(ADD-SOURCE, $h(v-)$) to all;
    ReAS( $v, h(v-), v$ )
 }//if
```

Fig.5 InitHight

```
//when $v$ receive(NEW-HEIGHT, $h$) from $w$
ReNH( $v, h, w$ )
 if $w \in Adj(v+)$
    $h_{v+}(w) \leftarrow h$ ;
 if $w \in Adj(v-)$
    $h_{v-}(w) \leftarrow h$ ;
```

Fig.6 ReNH

```
//when $v$ receive(INIT-HEIGHT, $h$) from $w$
ReIH( $v, h, w$ )
 ReNH( $v, h, w$ );
 if$\left(h(v+) = 0 \vee h(v+) > h+1\right)$
   $h(v+), h_{v-}(v) \leftarrow h+1$ ; $h(v-), h_{v+}(v) \leftarrow h+2$ ;
 InitHight( $v$ );
```

Fig.7 ReIH

```
//when $v$ receive(ADD-SOURCE, $h$) from $w$
ReAS( $v, h, w$ )
 if $ty(v) = Source$ {
    $InitOk \leftarrow InitOk + 1$ ;
    $h_{s+}(w) \leftarrow h$ ;
    if $InitOk = |X|$ {
       $h(s+) \leftarrow 2|V|$ ;
       $h_{v-}(s) \leftarrow 2|V|$ ;
      PushLiftIn( $v$ , $v$ ,0);
    }//if
 }//if
```

Fig.8 ReAS

```
PushLiftIn( $v$ , $w$ , $\Delta$ )
 $c_{v-}(w) \leftarrow c_{v-}(w) + \Delta$ ; $ex(v-) \leftarrow ex(v-) + \Delta$ ;
 while$\left(\exists u \text{ s.t. } c_{v-}(u) > 0 \wedge h(v-) > h_{v-}(u)\right)$
       $\wedge ex(v-) > 0$
 { $\Delta \leftarrow \min\{ex(v-), c_{v-}(u)\}$ ;
   $ex(v-) \leftarrow ex(v-) - \Delta$ ; $c_{v-}(u) \leftarrow c_{v-}(u) - \Delta$ ;
   if $u = v$
     PushLiftOut( $v$ , $v$ , $\Delta$ );
   else
     Send(PUSH-REQUEST, $\Delta$) to $u$ ;
 }//while
 if $ex(v-) > 0$ {
   $h \leftarrow \min\left\{h \middle| \sum_{u | c_{v-}(u) > 0 \wedge h_{v-}(u) < h} c_{v-}(u) \geq ex(v-)\right\}$ ;
   $h(v-), h_{v+}(v) \leftarrow h$ ;
```

[1] M is a positive real number that is large enough.

```
if ( $ty(v) = Source \wedge h(v-) > h_{v-}(s)$ )
     terminate;
for all $u \in Adj(v-)$ {
   Send(NEW-HEIGHT, $h(v-)$) to u;
}//for
 PushLiftIn( $v$ , $v$ ,0);
}//if
```

Fig.9 PushLiftIn

```
PushLiftOut( $v$ , $w$ , $\Delta$ )
  $c_{v+}(w) \leftarrow c_{v+}(w) + \Delta$ ; $ex(v+) \leftarrow ex(v+) + \Delta$ ;
  while$\left(\exists u \text{ s.t. } c_{v+}(u) > 0 \wedge h(v+) > h_{v+}(u)\right)$
      $\wedge ex(v+) > 0$
  {  $\Delta \leftarrow \min\{ex(v+), c_{v+}(u)\}$ ;
     $ex(v+) \leftarrow ex(v+) - \Delta$ ; $c_{v+}(u) \leftarrow c_{v+}(u) - \Delta$ ;
     if $u = v$
        PushLiftIn( $v$ , $v$ , $\Delta$ );
     else
        Send(PUSH-REQUEST, $\Delta$ )to $u$ ;
  }//while
  if $ex(v+) > 0$ {
  $h \leftarrow \min\left(h \middle| \sum_{u|c_{v+}(u)>0 \wedge h_{v+}(u)<h} c_{v+}(u) \geq ex(v+)\right)$ ;
  $h(v+), h_{v-}(v) \leftarrow h$ ;
  for all $u \in Adj(v+)$
     Send(NEW-HEIGHT, $h(v+)$)to u;
  PushLiftOut( $v$ , $v$ ,0);
  }//if
```

Fig.10 PushLiftOut

```
//when $v$ receive(PUSH-REQUEST, $\Delta$ )from $w$
RePR( $v, \Delta, w$ )
 if $w \in Adj(v-) \wedge h_{v-}(w) > h(v-)$ {
   if $ty(v) \neq Sink$
      PushLiftIn( $v$ , $w$ , $\Delta$ );
   else{
      $c_{v-}(w) \leftarrow c_{v-}(w) + \Delta$ ;
      $ex(v-) \leftarrow ex(v-) + \Delta$ ;
   }//else
 }else if $w \in Adj(v+) \wedge h_{v+}(w) > h(v+)$
         PushLiftOut( $v$ , $w$ , $\Delta$ );
      else
         Send(PUSH-DENY, $\Delta$ )to w;
```

Fig.11 RePR

```
//when $v$ receive(PUSH-DENY, $\Delta$ )from $w$
RePD( $v, \Delta, w$ )
 if $w \in Adj(v-)$ {
     $c_{v-}(w) \leftarrow c_{v-}(w) + \Delta$ ;
     $ex(v-) \leftarrow ex(v-) + \Delta$ ;
    PushLiftIn( $v$ , $v$ ,0);
 }else{
     $c_{v+}(w) \leftarrow c_{v+}(w) + \Delta$ ;
     $ex(v+) \leftarrow ex(v+) + \Delta$ ;
    PushLiftOut( $v$ , $v$ ,0);
 }//else
```

Fig.12 RePD

The time complexity of DA is $O(|V|^2)$. The message complexity of DA is $O(|V|^2 \cdot |A|)$.

5. Conclusions

In this paper, we formulate the lifetime maximization problem of sensor networks as a max flow problem on a transportation network with powers on vertices and arcs. With BS algorithm, we can achieve maximum lifetime routing of a sensor network in any precision if only we can find the max flow on such a transportation network. After propose a centralized algorithm with time complexity of $O(|V|^3)$, we propose an asynchronously distributed algorithm with time complexity of $O(|V|^2)$ and message complexity of DA is $O(|V|^2 \cdot |A|)$.

References

[1]. M. Bhardwaj, A. Chandrakasan, and T. Garnett, “Upper Bounds on the Lifetime of Sensor Networks,” *IEEE International Conference on Communications, Helsinki, Finland, June 2001.*

[2]. M. Bhardwaj, A. Chandrakasan, “Bounding the Lifetime of Sensor Networks Via Optimal Role Assignments,” *IEEE INFOCOM’2002.*

[3]. E. J. Duarte-Melo, M. Liu, and A. Misra, “A Modeling Framework for Computing Lifetime and Information Capacity in Wireless Sensor Networks,” *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Cambridge, UK, March 2004.*

[4]. Vivek Rai, Rabi N. Mahapatra, “Lifetime Modeling of a Sensor Network,” *Design, Automation and Test in Europe, Munich, Germany, March 2005.*

[5]. J. H. Chang, L. Tassiulas, “Energy conserving routing in wireless ad-hoc networks,” *IEEE INFOCOM’2000.*

[6]. K. Dasgupta, K. Kalpakis, and P. Namjoshi, “Efficient Algorithms for Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks,” *Computer Networks, vol. 42, 2003.*

[7]. Sankar, Z. Liu, “Maximum Lifetime Routing in Wireless Ad-hoc Networks,” *INFOCOM’2004.*

[8]. R. Madan, S. Lall, “Distributed Algorithms for Maximum Lifetime Routing in Wireless Sensor Net-works,” *Global Telecommunications Conference, IEEE, volume 2, Nov 2004.*

[9]. Andrew V. Goldberg and Robert E. Tarjan, “A New Approach to the Maximum Flow Problem,” *Journal of ACM, 35(4):921-940, 1988.*

[10]. T. L. Pham, M. Bui, I. Lavallae, S. H. Do, “A Distributed Preflow-Push for the Maximum Flow Problem,” *Innovative Internet Community Systems, 5th International Workshop, IICS 2005, Paris, France, June, 2005.*