Maximizing Data Preservation Time in Linear Sensor Networks

Ryan Hausen[1], Bin Tang[2], Samuel Sambasivam[1], and Simon Lin[1]
[1]Department of Computer Science, Azusa Pacific University, USA
[2]Department of Computer Science, California State University Dominguez Hills, USA
Email: {rhausen11,ssambasivam,slin}@apu.edu, btang@csudh.edu

***Abstract*— We study a new algorithmic problem called *data preservation problem with maximum preservation time*. It preserves data inside sensor networks (due to absence of the base station) by offloading overflow data from source node into the network, such that the data preservation time is maximized. We present an optimal and efficient algorithm for linear sensor networks.**

Keywords – Data Preservation, Sensor Networks

I. Background and Motivation

Many emerging sensor network applications are deployed in challenging environments such as underwater, remote or inhospitable regions, or under extreme weather. With very limited accessibility of such networks, the generated sensory data is first stored inside the network, and then uploaded to the remote base station by data mules, or through low rate satellite link when they become available.

In our network model, there is one sensor node that generates large amounts of sensory data (due to its proximity to the event of interest), therefore it has exhausted its storage capacity. This sensor node with exhausted data storage while still generating new data is *source node*. The newly generated data that can not be stored locally is *overflow data*. Other sensor nodes, which all have some available storages, are *storage nodes* (sensor node whose generated data has not exceeded its storage capacity is considered as a storage node). To prevent data loss, the overflow data is offloaded to the storage nodes to be stored, then collected when above uploading opportunities become available. The storage nodes that finally store offloaded data are *destination nodes*. We refer to the process that overflow data is offloaded from the source node to destination nodes as *data preservation in sensor networks*. In this paper, we study data preservation in linear sensor networks, which have been well adopted in applications such as water pollution monitoring along the river bank and underwater seismic monitoring along the seashore.

II. Problem Formulation

Network Model. The sensor network is represented as an undirected linear graph $G(V, E)$, where $V = \{n, n-1, ..., 2, 1, 0\}$ is a set of $n+1$ nodes (from left to right) and E is a set of edges. Let S be the single source node and $V_s = \{V - \{S\}\}$ be the set of storage nodes. There are a equal-size overflow data items at source node, denoted as $D = \{D_1, D_2, ..., D_a\}$. Let m_i be the available free storage space at storage node $i \in V_s$, measured in number of data items. Sensor node i has a finite and unreplenishable initial energy E_i. For each sensor node, sending or receiving a data item each costs 0.5 unit of its energy. Therefore, energy consumption of offloading a data item from the source node to a destination node equals the number of hops the data item traverses.

Problem Formulation. We define *preservation function* as $r : D \rightarrow V_s$, indicating $D_j \in D$ is offloaded from S to its destination node $r(j) \in V_s$. Let $P_j : S, ..., r(j)$ be the sequence of sensor nodes along which D_j is offloaded from S to $r(j)$. Let $x_{i,j}$ be the energy cost incurred by node i when offloading data item D_j from S to $r(j)$, then $x_{i,j} = 0.5$ if $i \in \{S, r(j)\}$, 1 if $i \in P_j - \{S, r(j)\}$, and 0 otherwise. Let $E_i^{'}$ denote i's remaining energy after all the a data items are offloaded, $E_i^{'} = E_i - \sum_{j=1}^{a} x_{i,j}, \forall\ i \in V$.

Definition 1: (**Data Preservation Time.**) Given a source node with a data items to be preserved, *data preservation time* of the sensor network is defined as the sum of remaining energy of the destination nodes of all the a data items, that is, $\sum_{j=1}^{a} E_{r(j)}^{'}$. It equals $\sum_{i \in V_s} \big(E_i^{'} \times \xi(i)\big)$, where $\xi(i)$ is the number of data items offloaded to node i. □

Considering all the nodes are constantly draining their battery powers, it is preferred that the data is stored at storage nodes with high energy levels. Data preservation time therefore indicates the network's $overall$ achievable effort to preserve the a data items. The objective is to find a preservation function r to maximize the data preservation time $\sum_{j=1}^{a} E_{r(j)}^{'}$, under the energy constraint $E_i^{'} \geq 0, \forall\ i \in V$ and the storage capacity constraint $|\{j \,|\, r(j) = i, 1 \leq j \leq a\}| \leq m_i, \forall\ i \in V$.

III. Algorithms

We first consider that the source node is at one end of the linear topology then consider that it could be at any position.

Source Node at One End. Assume node 0 is the source node with a amount of data items to offload.

Theorem 1: If node i finally stores $\xi(i)$ data items, the data preservation time equals $\sum_{i=1}^{n} \big(E_i \times \xi(i)\big) - a^2/2$.

Proof: Recall that $\xi(i)$ is the number of data items stored at node i post offloading. Since all the a data items are offloaded, we have $a = \sum_{i=1}^{n} \xi(i)$, and

$$a^2 = \sum_{i=1}^{n} \xi(i) \sum_{i=1}^{n} \xi(i) = 2 \times \sum_{i=1}^{n} \xi(i) \sum_{j=i+1}^{n} \xi(j) + \sum_{i=1}^{n} \xi^2(i).$$

Recall $E_i^{'}$ is the remaining energy of node i post offloading:

$E_i^{'} = E_i - \sum_{j=i+1}^{n} \xi(j) - \xi(i)/2$. The data preservation time

$$\begin{aligned}
&\sum_{i=1}^{n} \left(E_i^{'} \times \xi(i)\right) \\
=\ & \sum_{i=1}^{n} \left(E_i - \sum_{j=i+1}^{n} \xi(j) - \xi(i)/2\right) \times \xi(i) \\
=\ & \sum_{i=1}^{n} E_i \times \xi(i) - \sum_{i=1}^{n} \xi(i) \sum_{j=i+1}^{n} \xi(j) - \sum_{i=1}^{n} \xi^2(i)/2 \\
=\ & \sum_{i=1}^{n} \left(E_i \times \xi(i)\right) - a^2/2. \qquad \blacksquare
\end{aligned}$$

To maximize data preservation time $\sum_{i=1}^{n} E_i^{'} \times \xi(i)$, it therefore needs to maximize $\sum_{i=1}^{n} \left(E_i \times \xi(i)\right)$.

Algorithm 1: Optimal algorithm for source node 0.

```
Input: Linear sensor network $\{n, n-1, ..., 2, 1, 0\}$,
       $m_i$, $E_i$, $a$ data items $D$ at source node 0
Output: $r : D \to \{n, n-1, ..., 2, 1\}$;
1. Sort storage nodes $\{n, n-1, ..., 2, 1\}$ in non-ascending
   order of their initial energy: $E_{v_1} \geq E_{v_2} \geq ... \geq E_{v_n}$;
2. Find the top $k+1$ highest energy nodes: $v_1, ..., v_k, v_{k+1}$
   such that $\sum_{i=1}^{k} m_{v_i} < a \leq \sum_{i=1}^{k+1} m_{v_i}$;
3. Offload $m_i$ data items to each node $i \in \{v_1, ..., v_k\}$,
   and $a - \sum_{i=1}^{k} m_{v_i}$ data items to $v_{k+1}$;
4. RETURN  Data preservation time $\sum_{i=1}^{n} \left(E_i^{'} \times \xi(i)\right)$.
```

Algorithm 1 is optimal when source node is at one end (the proof is omitted due to space constraints). That is, it gives the maximum data preservation time among all the data offloading strategies. Its time complexity is $O(n\log n + a)$.

Source Node in Arbitrary Position. Now source node is k ($0 \leq k \leq n$). Let $t_l(x)$ and $t_r(x)$ denote the maximum data preservation time when x data items are offloaded to k's left and right respectively, with k being at one end.

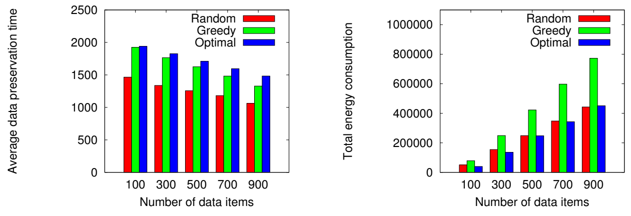
Algorithm 2: Optimal algorithm for source node k.

```
Input: Linear sensor network $\{n, n-1, ..., 2, 1, 0\}$,
       $m_i$, $E_i$, $a$ data items $D$ at source node $k$
Output: $r : D \to \{n, n-1, ..., k+1, k-1, 2, 1, 0\}$;
0. $max = 0$; $max$ is the maximum preservation time
1. for $(0 \leq a_1 \leq \min\{a, \sum_{i=k+1}^{n} m_i\})$
2.     if $t_l(a_1) + t_r(a - a_1) > max$, where $t_l(a_1)$
         and $t_r(a - a_1)$ are calculated by Algorithm 1;
3.     then $max = t_l(a_1) + t_r(a - a_1)$;
4. end for;
5. RETURN  $max$.
```

Algorithm 2 is optimal for arbitrary position of source node (the proof is omitted due to space constraints). Its time complexity is $O(n\log n + a^2)$. We emphasize that the optimality of Algorithms 1 and 2 is non-trivial. Below we design a greedy algorithm (Algorithm 3).

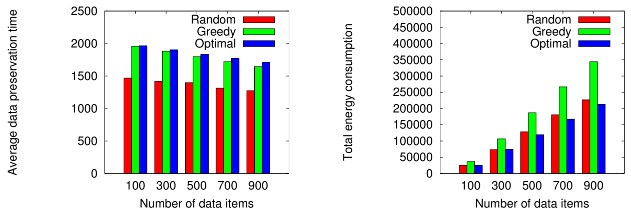
Algorithm 3: Greedy algorithm for source node k.

```
Input: Linear sensor network $\{n, n-1, ..., 2, 1, 0\}$,
       $m_i$, $E_i$, $a$ data items $D$ at source node $k$
Output: $r : D \to \{n, n-1, ..., 2, 1\}$;
1.  for (each of the $a$ data items)
2.      Offload it to the storage node with highest energy
        and available storage (tie is broken randomly);
3.  end for;
4. RETURN  Data preservation time $\sum_{i=1}^{n} \left(E_i^{'} \times \xi(i)\right)$.
```



(a) Average preservation time. (b) Total energy consumption.

Fig. 1. Performance comparison when source node is 0.



(a) Average preservation time. (b) Total energy consumption.

Fig. 2. Performance comparison when source node is 499.

The time complexity of Algorithm 3 is $O(n\log n + an)$. By offloading each data item to the highest energy node with storage in each iteration, it works more meticulously than Algorithm 2 to maximize the data reservation time. However, we show in Section IV that it is not optimal.

IV. Performance Evaluation

We consider a linear sensor network with 1000 nodes. We randomly set the initial energy level of each node between 1000 and 2000, and the storage capacity of each storage node between 1 and 5. In Figure 1, the source node is node 0 (at one end). It compares Algorithm 1, Algorithm 3 (referred to as Greedy), and a random algorithm (referred to as Random), while varying a from 100, 300, ..., to 900. In Random, we randomly choose a storage node with available storage to offload each data item. In Figure 2, the source node is 499 (in the middle). It compares Algorithm 2, Algorithm 3 and Random. It shows that Algorithm 1 and Algorithm 2 perform the best in either case in terms of average data preservation time (the total data preservation time divided by a), due to their optimality. We also observe that they both cost the least amount of total energy consumption.

V. Conclusion and Future Work

We identify, formulate, and solve a new algorithmic problem in sensor networks. We study how to offload overflow data items from source node to storage nodes to maximize the data preservation time, so that data can be preserved for the longest amount of time. We solve it optimally in linear topologies. As future work, we will study the problem under star, tree, and general graph topologies.