Steady and Fair Rate Allocation for Rechargeable Sensors in Perpetual Sensor Networks

Kai-Wei Fan Zizhan Zheng Prasun Sinha
Department of Computer Science and Engineering
The Ohio State University
{fank, zhengz, prasun}@cse.ohio-state.edu

ABSTRACT

Renewable energy enables sensor networks with the capability to recharge and provide perpetual data services. Due to low recharging rates and the dynamics of renewable energy such as solar and wind power, providing services without interruptions caused by battery runouts is non-trivial. Most environment monitoring applications require data collection from all nodes at a steady rate. The objective of this paper is to design a solution for fair and high throughput data extraction from all nodes in presence of renewable energy sources. Specifically, we seek to compute the lexicographically maximum data collection rate for each node, such that no node will ever run out of energy. We propose a centralized algorithm and an asynchronous distributed algorithm that can compute the optimal lexicographic rate assignment for all nodes. The centralized algorithm jointly computes the optimal data collection rate for all nodes along with the flows on each link, while the distributed algorithm computes the optimal rate when the routes are pre-determined. We prove the optimality for both the centralized and the distributed algorithms, and use a testbed with 155 sensor nodes to validate the distributed algorithm.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Algorithms,Design

Keywords

Rechargeable Sensor Networks, Rate Assignment

1. INTRODUCTION

Recent years have witnessed a growing number of prototype sensor network deployments for environment monitoring such as earthquake monitoring [1], structural monitoring [2, 3], soil monitoring [4], and glacial movement monitoring [5]. Unattended operability for long periods is highly desirable due to the harshness of the environment, which is typical in such applications. However, the limitations of fixed battery sources often require operation at low data rates, which may even be unacceptable to some applications.

Energy harvesting from natural sources such as solar [6, 7], wind [8], thermal [9], and vibration [10, 11], have been shown to be effective in addressing the above problem. The amount and source of energy that can be leveraged depends on the environment and the application. For example, in a deployment for monitoring vibration of industrial machines, vibration energy could be readily available. Similarly, for a deployment for monitoring the air-quality in the AC-ducts, thermal energy can be harvested when hot air is blowing. For outdoor deployments, solar energy is readily available for a wide range of application scenarios.

Environment monitoring applications often require periodic collection of data from all nodes. *The objective of this paper is to design a solution for fair and high throughput data extraction from all the nodes in the network in presence of renewable energy sources. Specifically, we seek to compute the lexicographically maximum data collection rate for each node in the network, such that no node will ever run out of energy.* A rate assignment is lexicographically maximum if it is impossible to increase the rate of a node without decreasing the rate of another one with a lower rate.

The time varying profile of recharging rates poses challenges in the computation of the optimum data rates. If the rate of data collection is high, a node can end up depleting its battery, which can hurt coverage as well as network connectivity. If the network gets partitioned due to a node running out of energy, the amount of data collected during that period also gets impacted. Similarly, if the data collection rate is low, the battery may reach the highest level which results in missed recharging opportunities. The determination of optimum data rates must take into account the profile of recharging, together with the energy cost of sensing, transmission, and packet reception.

The contributions of the paper are as follows:

- We propose a centralized algorithm to compute the optimum data collection rate for each node, along with the amount of flow on each link in the network.
- We propose an optimum distributed and asynchronous algorithm assuming that the routing tree is pre-determined.

- We prove the optimality of the centralized and the distributed algorithms.
- Using experimentation with solar panels and a sensor network testbed, we evaluate the performance of the proposed approaches under various realistic scenarios.

The organization of the paper is as follows. Section 2 presents the centralized algorithm to compute the optimal lexicographic rate assignment and corresponding routing paths. Section 3 presents the distributed algorithm. The evaluation of the protocols using experiments with solar energy harvesting and experiments on a testbed are presented in Section 4. We discuss some issues not addressed in this paper and our future works in Section 5. Section 6 presents background and related work. Finally, Section 7 concludes the paper.

2. OPTIMAL LEXICOGRAPHIC RATE ASSIGNMENT

The objective of this paper is to find a fair rate assignment for sensors that can best utilize renewable energy sources while providing continuous monitoring services. Consider a network with four nodes as shown in Figure 1. Each node is equipped with a solar cell to collect solar energy and store it in a rechargeable battery. The amount of energy collected by each node may differ due to their exposure to sunlight. In addition, the size of the solar panels may be different for different nodes. Each node is collecting data and sending packets to the sink at a certain rate. If nodes work at low rates such that their battery will never be depleted, they can provide continuous service. For example, assuming the four curves in Figure 2 are the recharging rates for these four solar cells, Figure 3(a) shows the battery levels of these four nodes when they collect readings and send five packets per second. However, the collected energy is not well utilized since we can increase the rate of each node without losing continuous service. But if the rate is too high, some nodes may run out of energy before they can be recharged. For example, Figure 3(b) shows the battery level of these four nodes when their working rates are 10 packets per second. The batteries of nodes A and C are depleted for 3 to 4 hours. The objective is to find a rate assignment for these four nodes that is fair, can best utilize the collected energy, and is able to provide uninterrupted service, as shown in Figure 3(c).

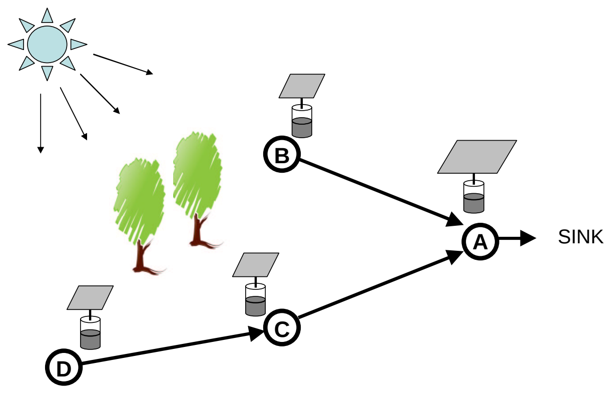


Figure 1: A network with four nodes, each equipped with a solar cell and a rechargeable battery.

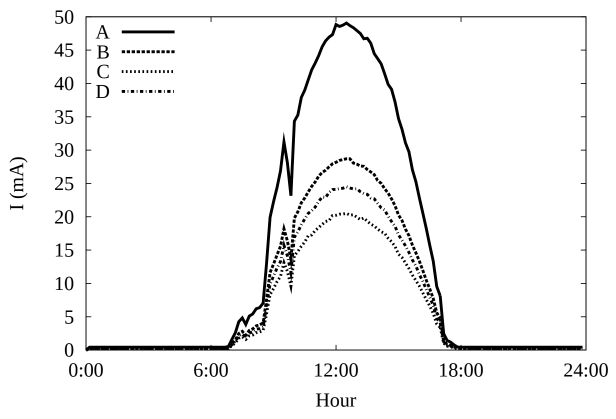


Figure 2: Recharging profile for nodes in Figure 1

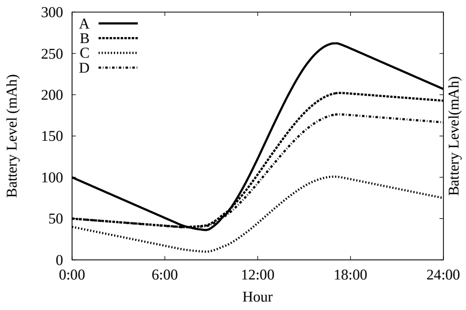
In this section we propose a centralized algorithm to solve the optimal joint lexicographic rate and flow assignment problem given the recharging profiles for all nodes and limited battery capacity. We assume that the link capacity is not a constraint. For a sensor network to support a continuous monitoring service, sensors should not consume more energy then they can collect. Therefore, the data rate is constrained by the amount of energy they can collect. Usually energy collected from a renewable source is too small to support high data rate applications. Take solar energy as an example. Figure 4 is the measured current collected from a $37mm \times 33mm$ solar cell over a 48 hours period that includes a sunny day and a partly cloudy day during the Spring. The total energy collected is $655.15mWh$ for the sunny day and is $313.70mWh$ for the partly cloudy day. For a wireless module, such as TelosB from Crossbow [12], the current drawn in receiving mode is $23mA$, in transmitting mode is $21mA$ at 0 dBm, and is $1.8mA$ for the micro control unit (MCU) in active mode, which can be converted to $69mW$, $63mW$, and $5.4mW$, respectively. Therefore, a node can only spend 9.2 hours forwarding packets a day ($69\text{mW}\times t/2 + 63\text{mW}\times t/2 + 5.4\text{mW}\times t = 655.15\text{mWh}$, $t = 9.2\text{h}$), which is about 38% of its time. The energy drawn by the attached sensors can further reduce it. Therefore, sensor nodes can only support low data rates and link capacity is typically not a constraint.

2.1 Problem Formulation

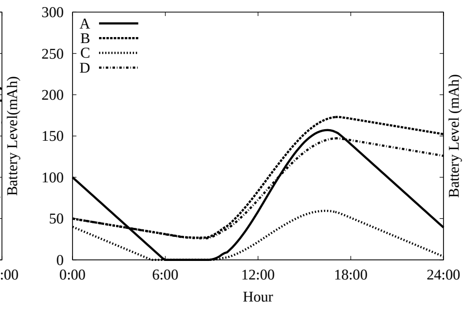
We seek to compute the optimal lexicographic rate assignment which is defined as follows:

DEFINITION 1. *Let L_1 and L_2 be two rate assignments. A rate vector R^L is a sorted rate vector of a rate assignment L if R^L is the result of sorting L in non-decreasing order, and R_i^L is the i^{th} rate in R^L. We say $L_1 = L_2$ if $R^{L_1} = R^{L_2}$, $L_1 > L_2$ if there exists an i such that $R_i^{L_1} > R_i^{L_2}$ and $R_j^{L_1} = R_j^{L_2}$, $\forall j < i$, and $L_1 < L_2$ otherwise. L^* is an optimal lexicographic rate assignment if there is no other rate assignment $L' > L^*$.*

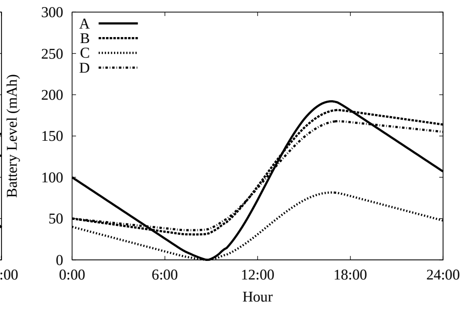
The goal of our paper is to find L^* such that given the battery constraints and energy recharging profiles for each node, no node ever runs out of energy. First, we define constant parameters that will be used in our formulation in



(a) $\vec{r} = (5, 5, 5, 5)$ pkt/s



(b) $\vec{r} = (10, 10, 10, 10)$ pkt/s



(c) $\vec{r} = (8.6, 8.6, 6.4, 6.4)$ pkt/s

Figure 3: Battery levels of four nodes in Figure 1 working at different data collection rates.

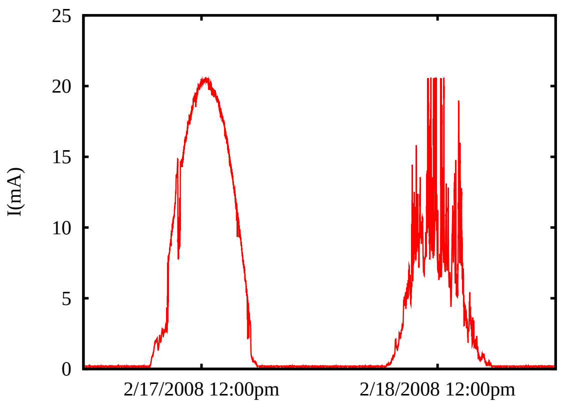


Figure 4: Current measured from a $37mm \times 33mm$ solar cell over a 48 hours period starting from 6:15pm 2/16/2008. The first day is a sunny day and the second day is a partly cloudy day.

Table 1, and variables for flows and energy constraints in Table 2.

With the given parameters and variables, we can formulate the lexicographic rate assignment problem as follows:

Problem: *LP-Lex*
Objective: *Lexicographically Maximize* $L = \{r_1, r_2, ..., r_n\}$

subject to

V	the set of all nodes in the network.
n	number of nodes in the network.
$\pi_{i,t}$	amount of energy collected by node i in time slot t.
Π_i	maximum battery capacity for node i.
W_i	initial battery level for node i.
λ_{tx}	energy cost for per packet transmission.
λ_{rx}	energy cost for per packet reception.
λ_{sn}	energy cost for sensing.
N_i	node i's neighbors.

Table 1: Constant parameters used in the formulation.

r_i	data collection rate for node i.
$f_{ij,t}$	flow from node i to node j in time slot t.
$w_{i,t}$	amount of energy available for node i at the beginning of time slot t.
$x_{i,t}$	total energy consumption for node i in time slot t.
$l_{i,t}$	a surplus variable that represents the amount of energy not being collected when the battery is full.

Table 2: Variables used in formulation.

$$r_i + \sum_{j \in N_i} f_{ji,t} = \sum_{j \in N_i} f_{ij,t} \tag{1}$$

$$x_{i,t} = \lambda_{sn} r_i + \lambda_{tx} \sum_{j \in N_i} f_{ij,t} + \lambda_{rx} \sum_{j \in N_i} f_{ji,t} \tag{2}$$

$$w_{i,t+1} = w_{i,t} + \pi_{i,t} - x_{i,t} - l_{i,t} \tag{3}$$

$$\sum_{k=1}^{T} x_{i,k} \le \sum_{k=1}^{T} \pi_{i,k} \tag{4}$$

$$0 \le w_{i,t} \le \Pi_i \tag{5}$$

$$w_{i,1} = W_i \tag{6}$$

$$l_{i,1} \ge 0 \tag{7}$$

for all $i \in V$ and $1 \le t \le T$ where T is the number of slots after which the recharging pattern is expected to repeat itself.

Constraint 1 ensures that the inflow equals the outflow. Constraint 2 specifies total energy consumption in time slot t, which includes the energy consumption in sensing, packet transmissions, and packet receptions. Constraint 3 states that the available energy in the next time slot is equal to the available energy in the current slot plus the energy collected minus the energy consumed in the current time slot. Constraint 4 ensures that nodes do not consume more energy than they collect. Constraint 5 states that the available energy does not go below zero and does not go above the battery capacity. Constraint 6 states that the available energy in the first time slot is the initial battery level W_i. Constraint 7 states that the surplus variable should be greater than or equal to 0.

2.2 Optimal Lexicographic Rate Assignment

The optimal lexicographic rate assignment problem can be solved using a similar approach as proposed in [13]. The approach in [13] first finds a maximum common rate r^* that is feasible for all the nodes with given constraints. It then computes the maximum feasible rate r_i for each node i assuming that the rates of other nodes are fixed at r^*. If $r_i = r^*$, the rate of node i cannot be increased any further even if all other nodes are assigned with the rate r^*; therefore, it will be assigned with the rate r^*. We call these nodes the constrained nodes. The algorithm finds all constrained nodes with $r_i = r^*$, fixes their rates to r^*, and repeats the process for the rest of the nodes. Due to the uniqueness of the optimal solution, in each iteration at least one constrained node exists. Therefore, iteratively the rate of each node will be determined and the algorithm converges to the optimal lexicographic rate.

In this paper we show that in our setting the optimal lexicographic rate assignment is also unique, and therefore we can use a similar approach to solve the problem. The differences between our approach and [13] are as follows. First, in [13], the resources are static and do not change while in our case the recharging rate changes over time. Second, we use a more general proof to show the uniqueness of the optimal lexicographic rate assignment that is suitable for both static as well as dynamic resources, as in our scenario.

To solve this problem iteratively, first we need to find the maximum common rate for the nodes. This can be achieved by modifying the objective and constraints of the formulation in problem *LP-Lex*. Instead of using different r_i variables for each node, we use the same rate r for all the nodes and try to maximize r. Therefore, the problem becomes:

Problem: *LP-MaxminRate*
Objective: *Maximize r*

subject to

$$r + \sum_{j\in N_i} f_{ji,t} = \sum_{j\in N_i} f_{ij,t} \tag{8}$$

$$x_{i,t} = \lambda_{sn} r + \lambda_{tx} \sum_{j\in N_i} f_{ij,t} + \lambda_{rx} \sum_{j\in N_i} f_{ji,t} \tag{9}$$

< Constraints 3 to 7 >

Using an LP solver we can compute a maxmin rate r^* for all nodes. After the maxmin rate r^* is computed, we can compute *Maximum Single Rate* (MSR) for each node to determine whose rate should be fixed at r^* in the optimal lexicographic rate assignment. The MSR problem for a node u can be modeled as a linear programming problem and be solved by an LP solver [13]. The formulation is similar to *LP-MaxminRate*, but for nodes other than u, we use the following constraints to replace constraints 8 and 9:

$$r^* + \sum_{j\in N_i} f_{ji,t} = \sum_{j\in N_i} f_{ij,t} \tag{10}$$

$$x_{i,t} = \lambda_{sn} r^* + \lambda_{tx} \sum_{j\in N_i} f_{ij,t} + \lambda_{rx} \sum_{j\in N_i} f_{ji,t} \tag{11}$$

Node u still uses constraints 8 and 9. By solving this LP formulation we can get the MSR rate of node u. We use *LP-MSR*(V, r^*, u) to represent the MSR rate for node u assuming all other nodes in V are assigned with the rate r^*. By solving the *LP-MaxminRate* and *LP-MSR* problems iteratively, the optimal lexicographic rate can be determined. The iterative algorithm is shown in Algorithm 1.

We will show that the solution to Algorithm 1 is unique.

Algorithm 1 Optimal Lexicographic Rate Assignment

```
procedure LexRateAssignment()
 1: $S \leftarrow V$
 2: while $S \neq \{\}$ do
 3:     $r^* \leftarrow$ Solve LP-MaxminRate($S$)
 4:     for each $i$ in $S$ do
 5:         $r_i \leftarrow$ Solve LP-MSR($S, r^*, i$)
 6:         if $r_i = r^*$ then
 7:             $D \leftarrow D \cup \{i\}$
 8:         end if
 9:     end for
10:     $S \leftarrow S - D$
11: end while
12: return $< r_1, r_2, ..., r_n >$
```

Due to the uniqueness, in each iteration of the while loop the set D with node i such that $r_i = r^*$ will be non-empty. Therefore, the while loop (Lines 2-11) will have at most n iterations. In each loop, we solve one $LP\text{-}MaxminRate$ problem and at most n $LP\text{-}MSR$ problems. Therefore, the complexity of $LexRateAssignment$ is $O(|N|^2 C_{LP}(|N|, |E|, T))$ where $C_{LP}(n, |E|, T)$ is the complexity of solving an LP problem with $O(nT)$ constraints and $O(|E|T)$ variables where E is the set of edges of the network.

Theorem 1. *LexRateAssignment computes the optimum lexicographic rate assignment.*

Proof. See Appendix 9.1. □

In the proof we first show that the optimal lexicographic rate assignment is unique. Due to the uniqueness of the solution, we can always find some nodes with MSR rate equal to the maxmin rate r^*. As we cannot increase the rates of these nodes even if we assign r^* to all other nodes, the only way to increase the rate of these constrained nodes is to decrease the rate of some nodes to be lower than r^*. Therefore, in the optimal lexicographic rate assignment, these constrained nodes will be assigned with a rate of r^*.

3. DISTRIBUTED LEXICOGRAPHIC RATE ASSIGNMENT

In Section 2 we use a centralized algorithm to solve the lexicographic rate assignment problem using the global knowledge of the network. The algorithm involves solving this problem using multiple executions of an LP solver, which is computation intensive and is not suitable for sensor networks. In this section, we present a distributed algorithm that does not require an LP solver to solve this problem for the case when the routes are known. We leave the more general problem of distributed joint routing and rate computation in presence of recharging for future work.

There are many studies on finding the maxmin rate in the literature. Most of them employ the feedback-based flow control mechanism. In particular, Charny et al. [14] use a technique called *Consistent Marking* to achieve maxmin rate assignment using a distributed algorithm. We use a similar technique to determine the rates of nodes when their routes pass through a node, but the way we compute the rate is different. We make an assumption as in [14] that each flow uses a fixed route to forward packets. This makes the problem more tractable for distributed computation. In this work, each node has exactly one flow connected to the sink, and we use flow i to represent the flow originating from node i.

Each node j maintains two variables, $r_j^{max}(i)$, and $r_j(i)$, for each flow i passing through it, where $r_j^{max}(i)$ is the current maximum achievable data rate and $r_j(i)$ is the computed data rate for flow i. For each flow i at any node, it must satisfy $r_j(i) \leq r_j^{max}(i)$, $\forall j$. Each node i starts with computing its maximum achievable rate $r_i^{max}(i)$ according to its recharging process assuming it only generates and forwards its own packets. For a non-steady energy resource such as solar energy, the maximum achievable rate depends on the available energy in the battery and the variations in the recharging rate. In order to support a perpetual sensor network, nodes should not consume more energy than they can collect. Furthermore, to provide continuous monitoring services, a node should not deplete its battery before it can be recharged, i.e., the total energy consumption should not be greater than the summation of the battery level and the total energy it collected up to a given time. Algorithm 2 computes the maximum rate at which a node can collect readings and forward them without running out of energy in any time slot.

Line 1 computes the rate in each time slot by computing the average energy collected per time slot divided by the energy consumption for collecting and forwarding a reading (e_s). Lines 3 to 14 consider cases when battery is depleted or full, given the data collection rate, energy recharging rate, and battery capacity. Variable w_t is the amount of energy in the battery in the beginning of time slot t, s is the last time slot when the battery is full, and E is the summation of the amount of energy collected from time slot $s+1$ to t and the available energy in the battery at the end of time slot s.

Algorithm 2 Maximum Rate

```
procedure $MaximumRate()$
 1: $r \leftarrow \frac{\sum_{i\leftarrow 1}^{T} \pi_{u,i}}{T} \times \frac{1}{e_s}$
 2: $E \leftarrow W_u$, $w_1 \leftarrow W_u$, $s \leftarrow 0$
 3: for $t \leftarrow 1$ to $T$ do
 4:     $w_{t+1} \leftarrow w_t + \pi_{u,t} - re_s$
 5:     $E \leftarrow E + \pi_{u,t}$
 6:     if $w_{t+1} > \Pi_u$ then
 7:         $E \leftarrow \Pi_u$
 8:         $w_{t+1} \leftarrow \Pi_u$
 9:         $s \leftarrow t$
10:     else if $w_{t+1} < 0$ then
11:         $r \leftarrow \frac{E}{t-s} \times \frac{1}{e_s}$
12:         $w_{t+1} \leftarrow 0$
13:     end if
14: end for
15: return $r$
```

First, we consider the case when the battery is full. If at time slot t the battery is full, the extra energy collected cannot be put into the battery. Therefore, E has to be adjusted and w_{t+1} is set to the maximum battery capacity Π_u (Lines 7 and 8). In addition, we know that if the battery is full at time slot t when working at rate r, the battery will still be full at time slot t even if the node works at a rate lower than r, and any rate that is smaller than r will still be feasible from time slot 1 to t. Observing that r will only become smaller each time it is updated in Line 11 (we will show it later), we can check if the newly computed rate is feasible by considering only the time slots after t. Therefore, we set s to t.

If at some time slot t the available energy w_{t+1} becomes negative, it means that node u cannot support the rate r. In this case, we should evenly distribute the energy collected from time slot $s+1$, plus the energy originally in the battery at the end of time slot s, to all slots from $s+1$ to t. It is clear that the newly computed rate r will be smaller than the previous one because Line 4 can be expressed as

$$w_{t+1} = w_{s+1} + \sum_{i=s+1}^{t} \pi_{u,i} - re_s(t-s) = E - re_s(t-s)$$

Because $w_{t+1} < 0$, $E - re_s(t-s) < 0$, $\frac{E}{(t-s)e_s} < r$. We know that as node u can support the original rate r from time slot 1 to $t-1$, it can also support the newly computed rate, which is lower than r, from time slot 1 to $t-1$, and also in time slot t.

After $r_i^{max}(i)$ is computed, node i sends a control packet containing the flow id i and rate $r_i = r_i^{max}(i)$ to its next hop. Node j receiving the control packet from node i first assigns the r_i from the control packet to $r_j^{max}(i)$ and then computes a new rate $r_j(i)$ based on flows passing through it. Node j then sends a control packet containing the flow id i and the newly computed rate $r_j(i)$ to its next-hop node. The process is repeated at each node from leaves to the sink. Once the control packet reaches the sink, the control packet will contain the maximum rate achievable for node i in the optimal lexicographic rate assignment, and the sink can send a feedback packet notifying node i of its assigned rate.

To compute the rate for each flow passing through node j, we define two types of flows for node j: restricted flows (R_j) and unrestricted flows (U_j). A flow f is in R_j if $r_j^{max}(f)$ is smaller than $r_j(j)$, i.e., its computed rate is restricted by some node before it reaches node j; otherwise f is in U_j. Note that for $f \in U_j$, $r_j(f) = r_j(j)$ in the optimum solution. Given the sets R_j and U_j, node j can compute the assigned rate $r_j(j)$ by evenly allocating the remaining rate not used by flows in R_j to all flows in U_j and node j itself as:

$$r_j(j) = \frac{C_j - e_f \sum_{i \in R_j} r_j^{max}(i)}{e_f(n_j - |R_j|) + e_s} \tag{12}$$

where $C_j = r_j^{max}(j)e_s$, e_f is the cost of forwarding a packet for other nodes, which includes the cost of receiving and transmitting a packet, and $n_j = |R_j| + |U_j|$ is the total number of flows, excluding flow j, passing through node j. If $r_j^{max}(i)$ of any flow i in R_j becomes higher than the new $r_j(j)$, or $r_j^{max}(i)$ of any flow i in U_j becomes smaller than the new $r_j(j)$, R_j and U_j are updated accordingly and Equation 12 is repeated until R_j and U_j do not change. Algorithm 3 shows the pseudocode for the distributed lexicographic rate assignment algorithm for node j.

THEOREM 2. *The UpdateRate rate computation using Equation 12 converges and computes the optimal lexicographic rate assignment.*

PROOF. We show the convergence by showing that after the first round of computation, the $r_j(j)$ will only increase and this moves at least one flow from U_j to R_j. As the number of flows is finite, *UpdateRate* will eventually converge. For the optimality of the *UpdateRate* algorithm, we

Algorithm 3 Distributed Lexicographic Rate Assignment

```
Require: $\pi_{1..T}$: recharging rate from time slot 1 to $T$

procedure $InitializeRate()$
 1: $r_j^{max}(j) \leftarrow MaximumRate()$
 2: $r_j(j) \leftarrow r_j^{max}(j)$

// $i$: received flow id
// $r_i$: maximum achievable rate of flow $i$
procedure $UpdateRate(i, r_i)$
 1: $r_j^{max}(i) \leftarrow r_i$
 2: $R_j' \leftarrow R_j \setminus \{i\}$
 3: $U_j' \leftarrow U_j \cup \{i\}$
 4: repeat
 5:     $R_j \leftarrow R_j'$
 6:     $U_j \leftarrow U_j'$
 7:     Compute $r_j(j)$ using Equation 12
 8:     $R_j' \leftarrow \{i | r_j^{max}(i) < r_j(j)\}$
 9:     $U_j' \leftarrow \{i | r_j^{max}(i) \geq r_j(j)\}$
10: until $R_j = R_j'$
11: $r_j(i) \leftarrow r_j^{max}(i), \forall i \in R_j$
12: $r_j(i) \leftarrow r_j(j), \forall i \in U_j$
```

first assume that there is an optimal solution that is better than the result computed by the *UpdateRate* algorithm, and then prove that this can not happen. See Appendix 9.2 for details. □

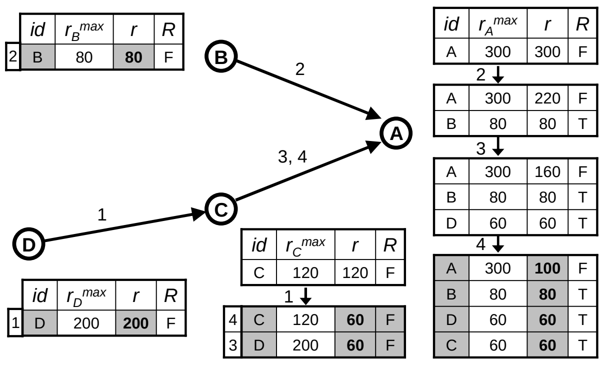


Figure 5: Distributed lexicographic rate computation for four nodes. The numbers on links represent the order of packet transmissions in this example. They also represent the rows that are transmitted and the instances that trigger the recomputation of rates of corresponding nodes. The R field is T if the corresponding flow is a restricted flow, otherwise it is an unrestricted flow. The items that are marked grey are data actually transmitted in the packet. Other fields are for internal reference only. The final solution is $(100, 80, 60, 60)$.

Figure 5 shows an example of the distributed rate computation for four nodes in steps. For ease of understanding we assume $e_s = e_f$. Nodes first compute their maximum rate r^{max} using the *InitializeRate* procedure in Algorithm 3 (The first table for each node). The maximum rates for node A through D are 300, 80, 120, and 200, respectively. After the nodes compute r^{max}, they send a control packet containing the flow id and the maximum achievable rate $r = r^{max}$ to their next-hop nodes. Note that the control packets do not need to be transmitted in a synchronous fashion from leaves to the root. However, transmitting control packets in this order can reduce the number of control packets, as when the rate of a flow is updated, the next-hop node has to update and compute a new rate accordingly.

When a node receives the control packet from its children, it uses *UpdateRate* to compute the rates. For example, when node C receives the rate 200 from node D in the first step, node C sets $r_C^{max}(D) = 200$, $R_D = \{\}$ and $U_D = \{D\}$. Therefore, $r_C(C) = 120e_s/(e_f + e_s) = 60$. When node A receives the rate 60 from node C for flow D at step 3, node A sets $r_A^{max}(D) = 60$, $R_A = \{B\}$, and $U_A = \{D\}$. Therefore, $r_A(A) = (300e_s - 80e_f)/(e_f + es) = 110$. Since $110 > r_A^{max}(D)$, D will be put into R_A, and in the next round $r_A(A) = (300e_s - (80 + 60)e_f)/(es) = 160$, and $R_A(D) = R_A^{max}(D) = 60$.

4. EVALUATION

We evaluate our distributed algorithm, called DLEX, on MoteLab [15], a testbed with more than 150 TmoteSky sensor motes deployed over a 3-floor building. TmoteSky consists of TI MSP430 processor running at 8MHz, has 10KB RAM, and uses CC2420 radio operating at 2.4GHz. Since the TmoteSky nodes in the testbed are not equipped with solar cells, we use the recharging model collected on a sunny day, as shown in Figure 4, as a baseline, and generate a recharging model in which the whole recharging profile is varied by a random amount that is −10% to 10% of the baseline for each sensor. Each sensor node stores the randomly generated charging rate for 24 hours, and uses one hour as the length of a slot to compute the rate.

4.1 Optimality

Figure 6 shows the rate assignments computed by an LP solver and our distributed algorithm for a shortest path routing tree. The X-axis is node id sorted in non-decreasing order according to their assigned rates. The two curves overlap and therefore we only see one curve in Figure 6. Figure 7 shows the difference between these two rate assignments. The difference between these two rate assignments is less than 0.03%. The difference comes from the difference in precision in arithmetic operation between sensor nodes and the LP solver running on a PC. The CPU of the sensor nodes has limited computation power and floating point operations are extremely slow on them. Therefore, we use integer operations to replace the floating point operations, and this leads to the loss of precision resulting in the difference.

4.2 Recharging Profile

We evaluate the effect of different recharging profiles on the rate assignment. Based on the data shown in Figure 4, we extract two different recharging profiles, one for a sunny day and one for a partly cloudy day.

Figure 8 shows the rate assignments obtained from our distributed algorithm using the two different recharging profiles as shown in Figure 4. We start the rate assignment protocol at 12:00am in the midnight and use 24 hours as the length of the recharging profile. The total energy collected on the sunny day is 655.15mWh, which is about twice of 313.7mWh, the energy collected on the partly cloudy day. Since in our protocol nodes do not consume more energy

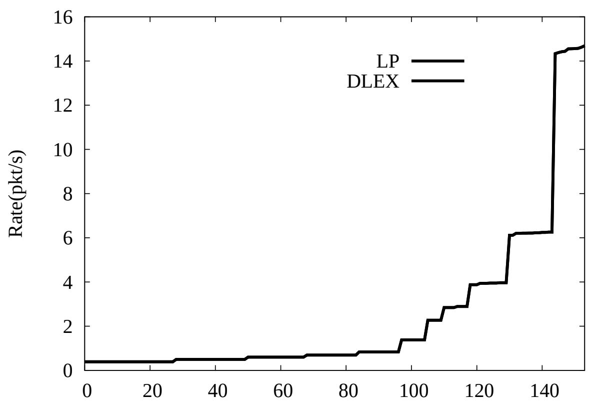


Figure 6: Rate assignments computed by an LP solver and our distributed protocol on a shortest path tree (These two curves overlap with each other).

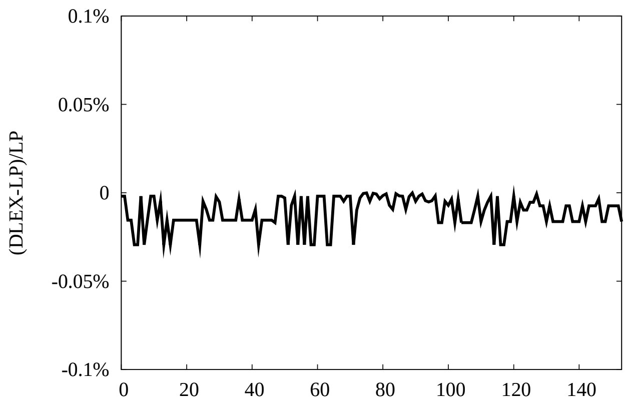


Figure 7: The difference between rate assignments computed by a centralized LP solver and our distributed protocol.

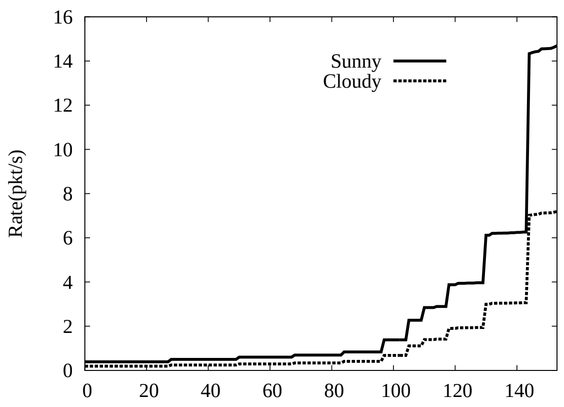


Figure 8: Rate assignments for energy collected on a sunny day and partly cloudy day shown in Figure 4.

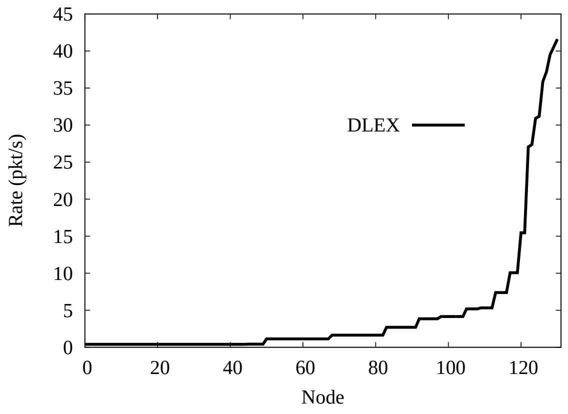


Figure 9: The rate assignments for six solar profiles obtained from [16].

than they can collect, the rates assigned to nodes are directly related to the amount of energy that they collect.

Next, we assign nodes with solar profiles obtained from [16] that are collected from six locations in California. This experiment represents the scenario where different sensors may have different recharging patterns. The solar radiation is sampled every two seconds. Every five minutes these two-second samples are averaged to obtain five-minute values. Each hour the average, maximum, and minimum values are recorded. To be consistent, we only use one day, or 24 hours, of data, from these six locations because the weather condition might be different on different days.

For each sensor node, we randomly select one solar profile, and randomly choose a number between the maximum and minimum energy collected from the solar cell as the recharging rate for that sensor. The result is shown in Figure 9.

4.3 Protocol Overhead

Figure 10 shows the number of control packets of the distributed protocol and also the size of subtrees rooted at the corresponding nodes. The X-axis is node id sorted in non-decreasing order according to their assigned rates. The number of control packets for the entire network is 1285, including the retransmitted packets due to packet losses. Assume T_i is the subtree rooted at node i. In our distributed algorithm, node i has to forward control packets from all nodes in T_i to the sink, and the responses from the sink to all nodes in $T_i - \{i\}$. For a node that is the root of a large subtree, it has to forward more control packets than other nodes, and this can be seen clearly in Figure 10 (nodes 27, 59, 62, 178, 160, 124). Ideally, the number of control packets sent by a node i is $2|T_i| - 1$. However, due to asynchronous operation and packet losses, the number of control packets is higher.

The size of a control packet payload is 9 bytes, which include 4 bytes for rate, 2 bytes for flow id, 2 bytes for forwarder id, and 1 byte for control message. The size of the control packet, including 7 bytes of packet header, is 16 bytes. For a network with 155 nodes, the maximum overhead for a node is around $2.5KB$. Note that the overhead can be further reduced by combining multiple control packets into one packet to save the overhead of packet header.

From Figure 10 we can also observe a trend that among those nodes that have larger subtrees (nodes 27, 59, 62, 178, 160, 124), their rates are lower if their subtrees are larger.

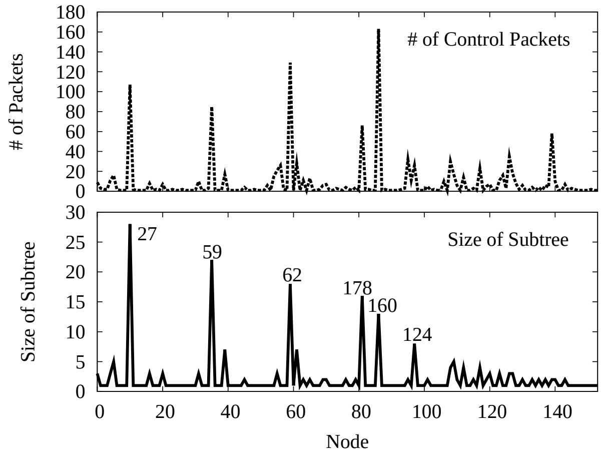


Figure 10: The number of control messages for DLEX and the number of children of corresponding nodes.

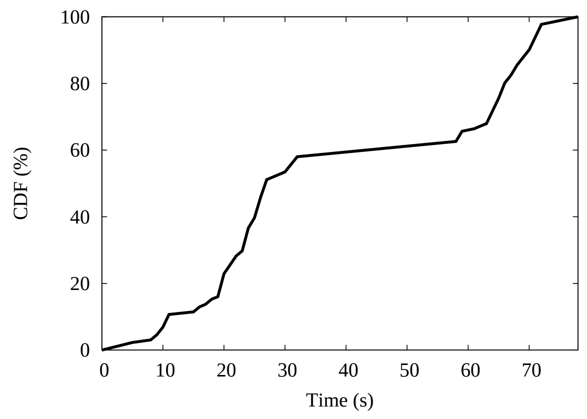


Figure 12: CDF of the number of nodes assigned with the optimal rate versus protocol running time.

From Figure 11 we can clearly see the trend. This is because the nodes closer to the sink are usually the bottleneck nodes since they have to forward packets for others. From the logs of experiments we do find that the nodes under the subtree rooted at these nodes are assigned with the same rate as these nodes. If we want to increase the total throughput or improve the lexicographic rate assignment, it is sufficient to increase the recharging rate of first hop nodes.

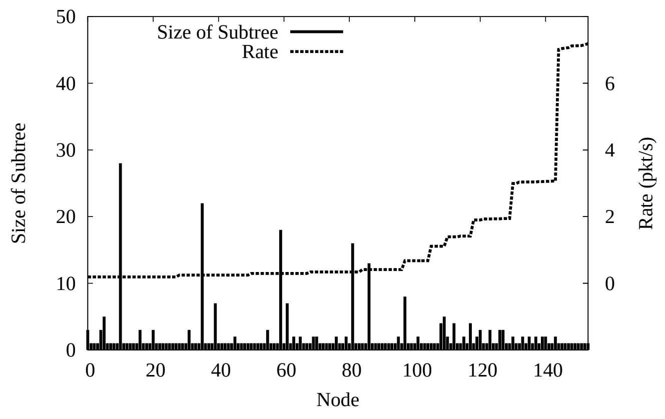


Figure 11: The size of a subtree v.s. the rate.

From our experiments, we observed that the running time of our distributed protocol varies from 50 seconds to 224 seconds. Figure 12 shows the CDF of the number of nodes assigned with the optimal rate versus protocol running time from one set of results where the total running time is 78 seconds. We observed that most of the time is wasted on timeouts (0.25 second + 0.1 second random backoff in our experiment) before retransmission due to packet losses because of unstable link quality. If the link quality can be considered while creating the routing tree, we should be able to reduce the running time significantly.

4.4 Initial Battery Level

In this section we study the performance of rate assignment when the battery level is low. When the battery level is low, how the maximum achievable rate is determined has big impact on the system performance since the battery does not have enough energy to serve as buffer when the energy collected is not sufficient.

We evaluate our protocol and compare it with a variation and a naive approach. In the variation, called DLEX-A, instead of using Algorithm 2 to compute the maximum achievable rate, we simply use the average recharging rate per time slot to compute the maximum achievable rate, and use the distributed algorithm to compute the rate assignment. In the naive approach, NAVG, nodes use the average rate per time slot to compute the maximum achievable rate, and use that rate as their working rate. We set the initial battery level to 30mAh for this experiment.

Figure 13 shows the actual rate assigned to each node. The X-axis is the node id, sorted in non-decreasing order according to their rates assigned by DLEX. The rates computed by DLEX are slightly smaller than DLEX-A because using Algorithm 2, DLEX has to lower the rate to prevent a node from running out of energy when the battery level is low. NAVG has the highest rate since it uses the maximum rate as the working rate. However, the higher working rate does not necessarily result in higher system performance.

Figure 14 shows the number of packets received at the sink for each node in the simulation. Because of the policy of the testbed, we are not allowed to run the experiment for 24 hours, we use a simulator and plug the rates we obtained from the testbed into the simulator to simulate packet transmissions and energy consumption in the rechargeable network. From the figure we can see that the number of packets received at the sink in NAVG does not correspond to the high working rate in Figure 13. This is because nodes may run out of energy before their battery is recharged due to the dynamics of the recharging process. Nodes that run out of energy can not generate packets anymore. In addition they can not forward packets for others either.

Figure 15 shows the percentage of time each node runs out of energy in 24 hours. We can see that in NAVG, all nodes run out of energy for some duration, and over 30% of nodes run out of energy for over 50% of the time during the simulation (13% of nodes run out of energy over 90% of the time during the simulation). DLEX-A only has 28 nodes that have reached 0 available energy for 10% to 15% of the simulation time. DLEX has 24 nodes that run out of energy

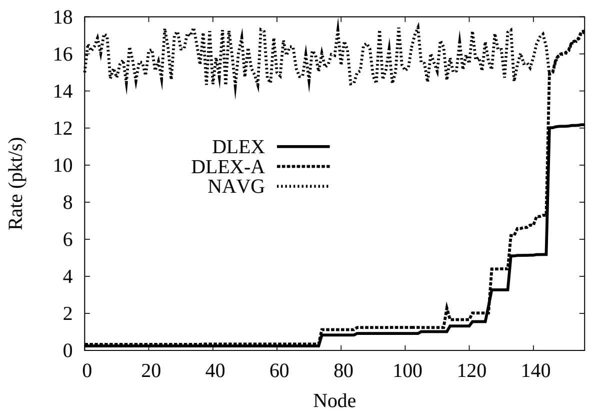


Figure 13: Rates of nodes in different rate assignment approaches.

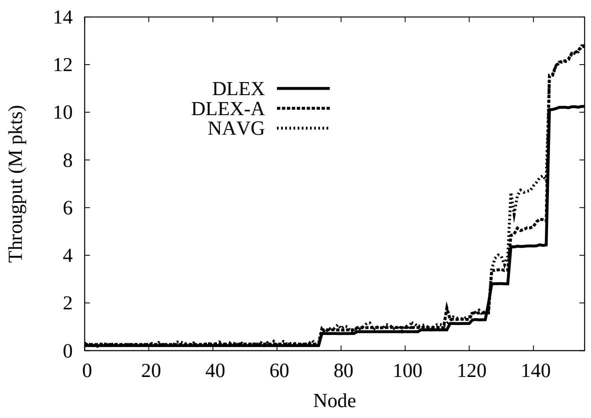


Figure 14: Number of packets received for each source.

for less than 3% of the simulation time.

Ideally, DLEX should not have any node running out of energy. The reason that nodes run out of energy in DLEX is because we use one hour as the unit to store the recharging profile, and use that to compute the maximum achievable rate. However, the recharging rate may vary within one hour, and the variation results in occasional energy runouts. We have conducted experiments using one minute as the unit to store the recharging profile and no nodes have ever run out of energy. However, storing recharging profile with a finer resolution will consume more memory or storage space, and therefore it is a design trade-off. To prevent nodes from running out of energy using only coarse grained recharging profile, nodes may reserve a small amount of energy as the buffer when the battery level is low. We leave this as future work and do not investigate it further in this paper.

Figure 16 shows the number of packets received at the sink and the percentage of nodes that ran out of energy during the 24 hours of simulation. NAVG can hardly receive any packets for about 2.5 hours of the time, from 5:50 to 8:15 due to many nodes running out of energy during that period. DLEX-A is better, however, its throughput is also affected severely during the same period because the nodes that run out of energy are usually close to the sink. When they run

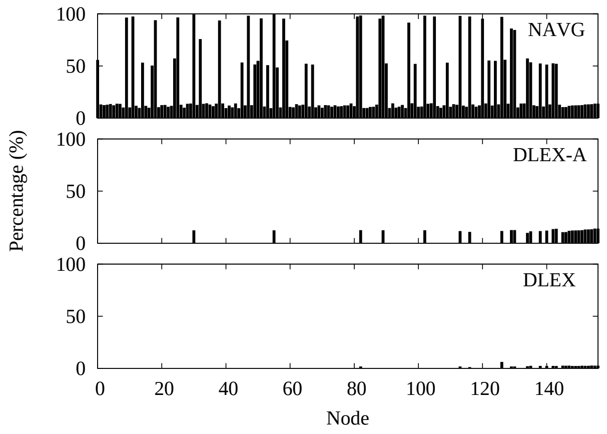


Figure 15: Percentage of time a node runs out of energy.

out of energy, they cannot forward packets for other nodes, thus resulting in severe throughput degradation. DLEX is affected only for a small portion of the time because it has fewest nodes running out of energy for very short duration. Again, if we store finer grained recharging profile on sensors, we can maintain a stable throughput for the entire simulation for all nodes. This shows that DLEX performs better in terms of uniformly collecting data across time.

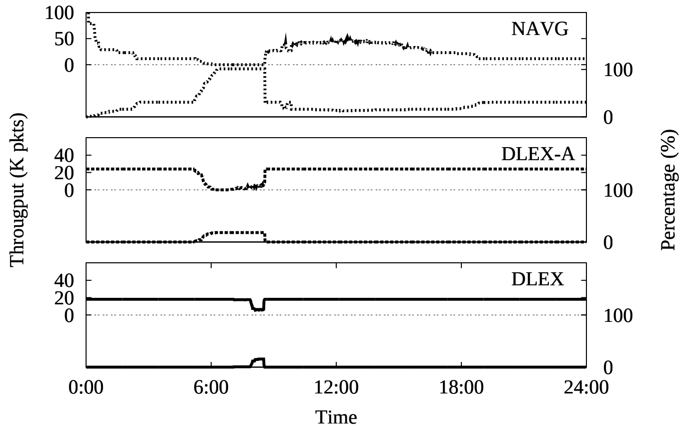


Figure 16: Total number of packets received at the sink (top) and the percentage of nodes running out of energy (bottom) in different time slots.

4.5 Topology

In this experiment we vary the transmission power to create networks with different sizes and densities. With lower transmission power, nodes have few choices for selecting the routing paths in shortest path tree, and the diameter, i.e., the maximum hop count of the network, will increase too. Figure 17 shows the results of three different transmission powers. The higher number represents higher transmission power. We can see that with higher transmission power, we can get better lexicographic rate assignment. This is due to the higher density of the network. In a high density network, there will be more nodes that are one hop away from the sink. The size of subtrees rooted at these one hop nodes will be smaller, compared with the subtrees of one hop nodes in a low density network which has fewer number of one hop nodes. Since the first hop nodes are usually the bottleneck nodes, fewer nodes in the subtree implies higher share of the available energy, thus resulting in higher achievable rate.

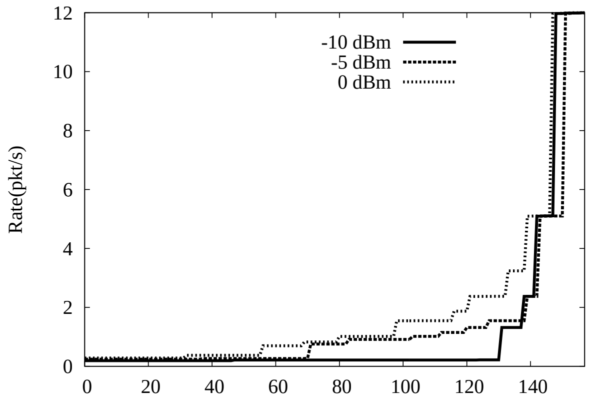


Figure 17: Rate assignment in different topology.

5. DISCUSSION

5.1 Impact of Tree Imbalance

Our distributed algorithm finds the optimal lexicographic rate assignment when each source has a fixed route and the routing path is known. As the optimal rates of nodes depend on which nodes their flows pass through, different routing paths may result in different optimum rate assignments. If some nodes have much larger subtree than others, nodes under those subtrees will typically be assigned with smaller rates. As the shortest path tree does not consider the load on the nodes, some nodes might have many more nodes in their subtrees than others. Nodes under such subtrees can only share a small portion of resources, and therefore will be assigned only a small rate. If we can create a more balanced tree such that each subtree has roughly the same number of nodes, we can improve the lexicographic rate assignment.

As computing a load balanced tree is an NP-hard problem [17], we use an offline heuristic to create a balanced tree from the connectivity graph we obtained from the testbed. The heuristic starts at any arbitrary spanning tree T of the connectivity graph G and greedily checks each edge of G connecting two different subtrees of T. An edge connecting two different subtrees is added to T if, by removing certain edge of T, the resulting set of subtrees has a lower standard deviation in terms of their sizes. The heuristic is applied to the subtrees of first hop nodes, and recursively applied to lower levels of subtrees.

We compare the rate assignment on the balanced tree resulting from our heuristic with the shortest path tree. We also compare it with the optimal solution where routes are not fixed and flows can be split over multiple routes.

Figure 18 shows the results we obtained from an LP solver for a dense network and Figure 19 shows the results for a sparse network. In both cases, the balanced tree does improve the lexicographic rate assignment over the shortest path tree, though in the sparse network the improvement is marginal since nodes do not have many choices for selecting a parent. This shows that in a dense network, a tree may not be the best routing structure, and multipath routing schemes are preferable for balancing the traffic load.

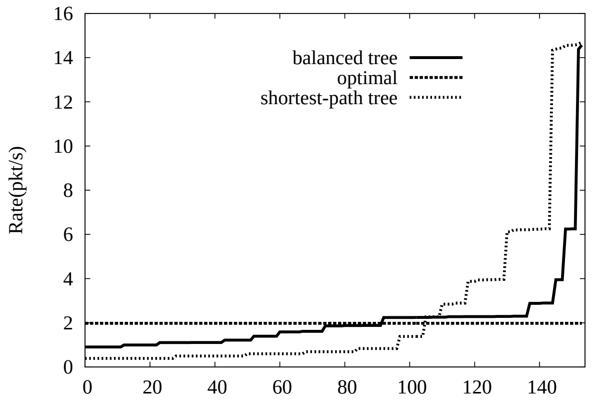


Figure 18: Rate assignment on shortest path tree, balanced tree, and the optimal in a dense network.

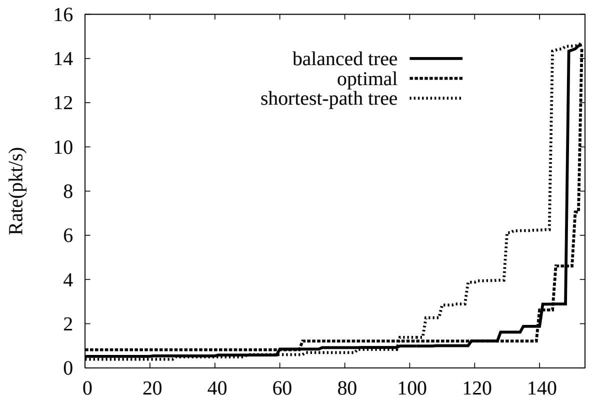


Figure 19: Rate assignment on shortest path tree, balanced tree, and the optimal in a sparse network.

We seek to design a better algorithm to create a balanced tree distributedly and also investigate algorithms that can exploit multi-path routing to achieve better lexicographic rate assignment. We leave these as our future work.

5.2 Granularity of the Recharging Profiles

In this paper we assume that the recharging profile of the renewable resources are known by each sensor. This can be achieved by keeping track of the recharging rate the sensors observed in the previous recharging cycles. To maintain a steady data collection rate, the recharging rate can be computed using the average of previous recharging cycles. As long as the average recharging rate does not change dramatically, sensors do not need to invoke the distributed algorithm to compute new rate assignments.

In experiments we use one day as one recharging cycle to compute the optimal rate. It is possible to use one month, one quarter, or one year as the recharging cycle to lower the dynamics. However, when the length of the recharging cycle increases, the initial battery level and when to invoke the protocol become more important. For example, if the recharging cycle is one year, during the winter season a higher battery level is desired so the sensors will have larger energy buffer to boost the data collection rate.

5.3 Link Quality and Battery Leakage

In our algorithm we use the cost of sensing, receiving, and transmitting data to compute the optimal rate, and assume there is no battery leakage. However, in reality packet loss due to poor channel conditions, interference, and collisions are inevitable, and battery leakage is a common phenomenon. Therefore, packet retransmission and battery leakage have to be considered.

For the centralized algorithm, we can take the retransmission probability into consideration when deciding the value of e_s and e_f to reflect the expected cost. For example, if the retransmission probability is p_{ij} for link (i, j), the cost of forwarding on link (i, j) can be expressed as $e_{rx} + \frac{e_{tx}}{p_{ij}}$ where e_{rx} is the receiving cost and e_{tx} is transmitting cost (note $e_{rx} + e_{tx} = e_f$). In the distributed algorithm, the retransmission probability of the link can be included in the control packet and be forwarded to the sink along with the maximum achievable rate. However, this will slightly increase the control overhead. Similarly, battery leakage can be modeled by an extra variable for the discharging rate. In each slot a fixed amount of energy is discharged from the battery. More complex models based on battery level, humidity, and temperature can also be used.

However, when the change of link quality results in disconnection of some links, rates have to be reassigned. For the centralized algorithm, rates have to be recomputed. For the distributed algorithm, first, a new routing tree is required. However, rates do not need to be recomputed for all nodes. Only nodes that change their parent node need to initiate the rate recomputation. If a change results in a new rate assignment, only nodes whose rates are different need to be notified with their new rates, though it is possible that a change of link quality may change the rate assignment of all nodes.

6. RELATED WORK

There are many works on developing sensors with capability of harvesting energy from solar or wind resources [8, 18–20]. There are also many studies on exploiting renewable energy to increase system performance or network lifetime [21–28]. In [22], the authors consider solar-aware routing in rechargeable sensor networks. They use a simple heuristic that preferably routes packets through solar-powered nodes, and the extra harvested energy is only a means to boost the network lifetime. In [24], the authors measure the environmental energy properties and renewable opportunities, and use the information to schedule tasks to increase the network lifetime. In [25] and [26], the authors further consider maximizing the system performance while maintaining Energy-Neutral operation, i.e., the energy used is always less than the energy harvested so that the system can continue to operate perennially. In [27], the authors study how sensor nodes should be activated dynamically so as to maximize a utility function based on the coverage area of the sensors. In [28], in addition to adjusting the duty cycle of the sensors to achieve Energy-Neutral operation, the authors also consider the variability of environmental energy resource and attempt to reduce the variation of the duty cycle using adaptive control theory. However, these works either only consider the workload of individual sensors and do not consider the influence on overall network performance by the individual decisions, or only try to maximize the system performance but do not consider the impact on individual sensors.

To maximize the system performance while balancing the workload, fairness has to be considered. Maxmin fairness, or lexicographic fairness, has been widely used to define the fairness of a system. In [14], a distributed maxmin rate computation algorithm for fixed flows routed through capacity constrained switches in wired networks is proposed. The proposed algorithm computes a maxmin rate assignment for each flow and guarantees quick convergence. In [29], the authors generalize the problem by adding maximum and minimum rate requirement for each flow, and propose a centralized algorithm similar to the one proposed in [30] that identifies bottleneck links first, and assigns rates equally to all flows passing through these bottleneck links. A distributed algorithm is also proposed that is based on the algorithm proposed in [14]. In [13], a centralized algorithm is proposed that iteratively uses linear programming to find lexicographic rate assignment for all sensor nodes that periodically report readings to the sink. The proposed algorithm does not require these flows to be forwarded through fixed routes. In [31] a distributed congestion control scheme is proposed to achieve maxmin rate allocation through overhearing and propagating congestion announcement, but it requires sophisticated parameter tuning to achieve stable operation and the rates oscillate up and down after converging even when the topology does not change. All these works solve the fairness problem with static constraints, such as based on switches or battery capacity, and do not consider dynamic resources such as changing harvested energy.

7. CONCLUSION

In this paper we study the rate assignment problem for rechargeable sensor networks. We propose a centralized algorithm and a distributed algorithm for optimal lexicographic rate assignment. The centralized algorithm computes the optimal rate assignment along with determining the amount of flow on each link. The distributed algorithm computes the optimal rate when the routing tree is predetermined. We prove the optimality of the centralized and the distributed algorithms. To evaluate the proposed distributed algorithm, we conduct experiments using a testbed with 155 sensor nodes under various scenarios. The rates computed by the distributed algorithm are dependent on the routing tree. As part of our future work, we plan to explore distributed solutions that can jointly determine the optimum rates for all nodes and the flows on each link.

8. REFERENCES

[1] M. Lukac, V. Naik, I. Stubailo, A. Husker, and D. Estrin, “In Vivo Characterization of a Wide area 802.11b Wireless Seismic Array,” Center for Embedded Network Sensing. Papers. Paper 100., 2007. [Online]. Available: http://repositories.cdlib.org/cens/wps/100

[2] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, “A Wireless Sensor Network for Structural Monitoring,” in *Proc. of the ACM Sensys*, Nov. 2004, pp. 13–24.

[3] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, “Health Monitoring of Civil

Infrastructures Using Wireless Sensor Networks," in *Proc. of IPSN*, Cambridge, MA, Apr. 2007.

[4] T. E. O. Systems, "AMARSS and NIMS - Networked Minirhizotron and Arrayed Rhizosphere Sensing Systems." [Online]. Available: http://research.cens.ucla.edu/projects/2007/Terrestrial/AMARSS-NIMS/AMA%RSS

[5] K. Martinez, R. Ong, and J. Hart, "Glacsweb: a Sensor Network for Hostile Environments," in *Proc. of The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, 2004.

[6] J. Hsu, A. Kansal, J. Friedman, V. Raghunathan, and M. Srivastava, "Energy Harvesting Support for Sensor Network," in *Proc. of IEEE IPSN Demo*, 2005. [Online]. Available: http://www.ee.ucla.edu/~mbs/ipsn05/demo/18_JHsu.pdf

[7] V. Raghunathana, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design Considerations for Solar Energy Harvesting Wireless Embedded Systems," in *Proc. of IPSN (SPOT Track)*, Apr. 2005, pp. 457–462.

[8] C. Park and P. H. Chou, "AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes," in *SECON*, Sept. 2006, pp. 168–177.

[9] I. Stark, "Thermal Energy Harvesting with Thermo Life," in *Proc. of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2006.

[10] M. P. Release, "MicroStrain Wins Navy Contract for Self Powered Wireless Sensor Networks," http://www.microstrain.com/news/article-29.aspx, 2003.

[11] S. Meninger, J. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang, "Vibration-to-electric energy conversion," *IEEE Transactions on VLSI Systems*, vol. 9, no. 1, pp. 64–76, Feb. 2001.

[12] Crossbow, "Crossbow," http://www.xbow.com.

[13] S. Chen, Y. Fang, and Y. Xia, "Lexicographic Maxmin Fairness for Data Collection in Wireless Sensor Networks," in *IEEE Transactions on Mobile Computing*, vol. 6, no. 7, July 2007, pp. 762–776.

[14] A. Charny, D. D. Clark, and R. Jain, "Congestion Control With Explicit Rate Indication," in *IEEE International Conference on Communication*, vol. 3, June 1995, pp. 1954–1963.

[15] "MoteLab," http://motelab.eecs.harvard.edu/.

[16] "National Climatic Data Center." [Online]. Available: http://www.ncdc.noaa.gov/oa/climate/uscrn

[17] J. Gao and L. Zhang, "Load Balanced Short Path Routing in Wireless Networks," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 4, Apr. 2006, pp. 377–388.

[18] X. Jiang, J. Polastre, , and D. Culler, "Perpetual environmentally powered sensor networks," in *The 4th International Conference on Information Processing in Sensor Networks*, Apr. 2005, pp. 463–468.

[19] P. Dutta, J. Hui, J. Jeong, S. Kim, C. Sharp, J. Taneja, G. Tolle, K. Whitehouse, and D. Culler, "Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments," in *The 5th International Conference on Information Processing in Sensor Networks*, Apr. 2006, pp. 407–415.

[20] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, and J. Porter, "LUSTER: Wireless Sensor Network for Environmental Research," in *SenSys*, Nov. 2007, pp. 103–116.

[21] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware Design Experiences in ZebraNet," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 227–238.

[22] T. Voigt, H. Ritter, and J. Schiller, "Utilizing Solar Power in Wireless Sensor Networks," in *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, Oct. 2003, pp. 416–422.

[23] M. Rahimi, H. Shah, G. S. Sukhatime, J. Heideman, and D. Estrin, "Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network," in *IEEE International Conference on Robotics and Automation*, vol. 1, Sept. 2003, pp. 19–24.

[24] A. Kansal and M. B. Srivastava, "An Environmental Energy Harvesting Framework for Sensor Networks," in *Proceedings of the International Symposium of Low Power Electronics and Design*, 2003, pp. 481–486.

[25] A. Kansal, D. Potter, and M. B. Srivastava, "Performance Aware Tasking for Environmentally Powered Sensor Networks," in *ACM SigMetrics*, June 2004, pp. 223–234.

[26] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power Management in Energy Harvesting Sensor Networks," in *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 4, Sept. 2007.

[27] K. Kar, A. Krishnamurthy, and N. Jaggi, "Dynamic Node Activation in Networks of Rechargeable Sensors," in *INFOCOM*, vol. 3, Mar. 2005, pp. 1997–2007.

[28] C. M. Vigorito, D. Ganesan, and A. G. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *The 4th Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks*, June 2007, pp. 21–30.

[29] Y. T. Hou, S. S. Panwar, and H. H.-Y. Tzeng, "On Generalized Max-Min Rate Allocation and Distributed Convergence Algorithm for Packet Networks," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 5, May 2004, pp. 401–416.

[30] D. Bertsekas and R. Gallagher, *Data Networks*. Prentice Hall, 1992.

[31] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-Aware Fair Rate Control in Wireless Sensor Networks," in *SIGCOMM*, vol. 36, no. 4, Oct. 2006, pp. 63–74.

9. APPENDIX

9.1 Proof of Theorem 1

THEOREM 1. *LexRateAssignment computes optimal lexicographic rate assignment.*

First, we define some terms that will be used in the proof.

DEFINITION 2. *Let $G(V, E, S)$ be a network with nodes $V = \{v_1, v_2, ..., v_n\}$, wireless links E, and node constraints $S = \{s_1, s_2, ..., s_n\}$. A node constraint s_i is a 2-tuple of (Π_i, R_i) that specifies the maximum battery capacity and energy recharging rate for v_i. We define $G'(N', E', S') = G^2(N, E, S)$ where $N' = N$, $E' = E$, $S' = \{s'_1, s'_2, ..., s'_n\}$, and $s'_i = 2 \times s_i = (2\Pi_i, 2R_i)$, $1 \le i \le n$. In short, G^2 is a network with the same topology as G, but the maximum capacity and energy recharging rate of each node is doubled in G^2. We also define $G = G^2/2$.*

DEFINITION 3. *Let L_1 and L_2 be any two rate assignments for nodes $\{v_1, v_2, ..., v_n\}$ where $L_1 = \{r_1^{(1)}, r_2^{(1)}, ..., r_n^{(1)}\}$ and $L_2 = \{r_1^{(2)}, r_2^{(2)}, ..., r_n^{(2)}\}$. We define $L' = L_1 + L_2$ where $L' = \{r'_1, r'_2, ..., r'_n\}$ and $r'_i = r_i^{(1)} + r_i^{(2)}$, $1 \le i \le n$. We also define $L'' = L/2$ where $L'' = \{r''_1, r''_2, ..., r''_n\}$ and $r''_i = r_i/2$, $1 \le i \le n$.*

LEMMA 1. *The lexicographic optimal rate assignment is unique.*

PROOF. We prove the uniqueness by contradiction. We show that if there are more than one optimal solutions, say L_1 and L_2, there must exist a rate assignment that is better than L_1 and L_2 so they can not be optimal.

Suppose there are two different optimal rate assignments, L_1 and L_2, for a network G. L_1 and L_2 are identical as sorted vectors but their rate assignments for some nodes differ. Let M be the set of nodes whose rates are different in L_1 and L_2. Among all the nodes in M, let v_k be a node with the smallest rate in L_1. Let $S_1 = \{v_i \mid r_i^{(1)} < r_k^{(1)}\}$, $Q_1 = \{v_i \mid r_i^{(1)} = r_k^{(1)}\}$, and $X_1 = \{v_i \mid r_i^{(1)} > r_k^{(1)}\}$, i.e., S_1 contains all nodes whose rates are smaller than v_k's rate in L_1, Q_1 contains all nodes whose rates are equal to v_k's rate in L_1, and X_1 contains all nodes whose rates are larger than v_k's rate in L_1. Next, we define $S_2 = \{v_i \mid r_i^{(2)} < r_k^{(1)}\}$, i.e. S_2 contains all nodes whose rates in L_2 are smaller than v_k's rate in L_1. We know that

$$r_i^{(1)} = r_i^{(2)}, \forall v_i \in S_1 \tag{13}$$

because v_k is the node with the smallest rate in L_1 that differs in its rate assignment in L_2. Therefore,

$$S_1 \subseteq S_2 \tag{14}$$

Furthermore,

$$|S_1| = |S_2| \tag{15}$$

otherwise L_1 and L_2 will have different number of nodes whose rates are smaller than $r_k^{(1)}$, which contradicts that $L_1 = L_2$. By (14) and (15), we have

$$S_1 = S_2 \tag{16}$$

Now we construct a new network G^2. Observe that $L^2 = L_1 + L_2$ is a feasible rate assignment for G^2. Moreover $L^2/2$ is a feasible rate assignment for $G^2/2$. Therefore, $L' = L^2/2 = (L_1 + L_2)/2$ is a feasible rate assignment for G since $G^2/2 = G$. We show that L' is lexicographically greater than L_1, and therefore L_1 can not be optimal.

First, consider the rates in L' for nodes in S_1. By Definition 3 and (13) we have

$$r'_i = (r_i^{(1)} + r_i^{(2)})/2 = r_i^{(1)} < r_k^{(1)}, \forall v_i \in S_1 \tag{17}$$

Considering the rates in L' for nodes in Q_1 and X_1. First

$$r_i^{(2)} \ge r_k^{(1)}, \forall v_i \in Q_1 \cup X_1 \tag{18}$$

otherwise v_i would be in S_2, which indicates $v_i \in S_1$ and is a contradiction that $v_i \in Q_1 \cup X_1$. Therefore, by Definition 3 and (18) we have

$$r'_i = (r_i^{(1)} + r_i^{(2)})/2 \ge (r_k^{(1)} + r_k^{(1)})/2 = r_k^{(1)}, \forall v_i \in Q_1 \tag{19}$$

$$r'_i = (r_i^{(1)} + r_i^{(2)})/2 > (r_k^{(1)} + r_k^{(1)})/2 = r_k^{(1)}, \forall v_i \in X_1 \tag{20}$$

Now we define $S' = \{v_i \mid r'_i < r_k^{(1)}\}$, $Q' = \{v_i \mid r'_i = r_k^{(1)}\}$, and $X' = \{v_i \mid r'_i > r_k^{(1)}\}$. By (17), (19), and (20) we know

$$S_1 \subseteq S' \tag{21}$$

$$X_1 \subseteq X' \tag{22}$$

Because $S_1 \cup Q_1 \cup X_1 = S' \cup Q' \cup X'$, by (21) and (22)

$$Q_1 \supseteq Q' \tag{23}$$

By (19) and (20)

$$Q_1 \cup X_1 \subseteq Q' \cup X' \tag{24}$$

Because $S' \cap (Q' \cup X') = \phi$ and $S_1 \cap (Q_1 \cup X_1) = \phi$, we have

$$|S'| + |Q' \cup X'| = |S_1| + |Q_1 \cup X_1| \tag{25}$$

From (21) and (24) we know that $|S_1| \le |S'|$ and $|Q_1 \cup X_1| \le |Q' \cup X'|$. By (25), we know $|S_1| = |S'|$ and this together with (21) gives us

$$S_1 = S' \tag{26}$$

Furthermore, $r_k^{(2)} > r_k^{(1)}$ otherwise v_k would be in S_2 which indicates $v_k \in S_1$, a contradiction of the definition of S_1. Therefore, $r'_k = (r_k^{(1)} + r_k^{(2)})/2 > (r_k^{(1)} + r_k^{(1)})/2 = r_k^{(1)}$. Since $v_k \in Q_1$ and $v_k \in X'$, using (23) we further have

$$Q_1 \supset Q' \tag{27}$$

$$X_1 \subset X' \tag{28}$$

By (26), (27), and (28), $L' > L_1$, which is a contradiction that L_1 is an optimal lexicographic rate assignment.

□

9.2 Proof of Theorem 2

THEOREM 2. *The UpdateRate rate computation using Equation 12 converges and computes optimal lexicographic rate assignment.*

PROOF. First we show the convergence of the *UpdateRate* algorithm. Let $r_j(j)^{(x)}$ be the $r_j(j)$ at the end of round x. There are only three possibilities considering R_j and U_j: (1) R_j and U_j do not change. (2) Some flows in U_j become restricted and are moved to R_j. (3) Some flows in R_j become unrestricted and are moved to U_j. For case 1, the algorithm terminates, so we only consider cases 2 and 3.

- **Case 2**: If a non-empty subset of flows, say Z, in U_j is moved to R_j, the $r_j(j)^{(2)}$ will be computed in next round. Since

$$\begin{aligned} r_j(i)^{(1)} &= \frac{C_j - e_f \sum_{i \in R_j} r_j^{max}(i)}{e_f(n_j - |R_j|) + e_s} \\ r_j(i)^{(2)} &= \frac{C_j - e_f \sum_{i \in R_j \cup Z} r_j^{max}(i)}{e_f(n_j - (|R_j| + |Z|)) + e_s} \end{aligned}$$

Therefore,

$$\begin{aligned}
&r_j(j)^{(2)}(e_f(n_j-(|R_j|+|Z|))+e_s)\\
&=C_j-e_f\sum_{i\in R_j\cup Z} r_j^{max}(i)\\
&=C_j-e_f\sum_{i\in R_j} r_j^{max}(i)-e_f\sum_{i\in Z} r_j^{max}(i)\\
&=r_j(j)^{(1)}(e_f(n_j-|R_j|)+e_s)-e_f\sum_{i\in Z} r_j^{max}(i)\\
&>r_j(j)^{(1)}(e_f(n_j-|R_j|)+e_s)-r_j(j)^{(1)}e_f|Z|\\
&=r_j(j)^{(1)}(e_f(n_j-(|R_j|+|Z|))+e_s)
\end{aligned}$$

and $r_j(j)^{(2)} > r_j(j)^{(1)}$. The above argument could be generalized for round x and $x+1$ to show that $r_j(j)^{(x+1)} > r_j(j)^{(x)}$. Since the number of flows in U_j is finite, the process will eventually converge.

- **Case 3**: If a non-empty subset of flows, say Z, in R_j are moved to U_j, the $r_j(j)^{(2)}$ will be computed as

$$\begin{aligned}
r_j(j)^{(2)} &= \frac{C_j-e_f\sum_{i\in R_j-Z} r_j^{max}(i)}{e_f(n_j-|R_j-Z|)+e_s}\\
&=\frac{C_j-e_f\sum_{i\in R_j} r_j^{max}(i)+e_f\sum_{i\in Z} r_j^{max}(i)}{e_f(n_j-|R_j|+|Z|)+e_s}\\
&>\frac{C_j-e_f\sum_{i\in R_j} r_j^{max}(i)+e_f\sum_{i\in Z} r_j(j)^{(1)}}{e_f(n_j-|R_j|+|Z|)+e_s}\\
&=r_j(j)^{(1)}
\end{aligned}$$

Using the same argument as in case 2, we know $r_j(j)^{(x+1)} > r_j(j)^{(x)}$, and the process will converge.

Now we show that the *UpdateRate* algorithm computes the optimal lexicographic rate assignment. We show this by assuming that there is an optimal rate assignment D^* which is better than the rate assignment D computed by *UpdateRate*, and show that given D^*, *UpdateRate* can compute a rate assignment better than D^*, and hence a contradiction.

Let $D=(r(1),...,r(n))$ be the rate assignment computed from the distributed algorithm and $D^*=(r^*(1),...,r^*(n))$ be the optimal lexicographic rate assignment for flows 1 to n, and $D^*>D$. Among those flows whose rates are different in D and D^*, let f be the node that is assigned with the smallest rate in D; therefore, $r(f)<r^*(f)$.

First, as $r(f)<r^*(f)\le r_f^{max}(f)$, flow f must be restricted at some node j, i.e., $r(f)=r_j(f)=r_j(j)<r_f^{max}(f)$ (it is possible that $f=j$). We define R_j and U_j as the set of flows passing through node j and whose rates are smaller than and greater or equal to $r(f)$ in D respectively. Thus $f\in U_j$. We also define R_j^* and U_j^* as the set of flows that pass through node j and whose rates are smaller than and greater or equal to $r^*(f)$ in D^* respectively.

We first show the following three properties.

$$r_j(i) = r^*(i), \forall i\in R_j \tag{29}$$
$$r(f) \le r^*(i), \forall i\in U_j \tag{30}$$
$$r(f) \le r^*(j) \tag{31}$$

Because $r(f)<r^*(f)$, it is clear that

$$R_j\subseteq R_j^* \tag{32}$$

Furthermore, since flow f is the flow whose rates are different in D and D^* and is the smallest one in D, $r_j(i)=r^*(i), \forall i\in R_j$. This proves Equation 29.

Second, for $i\in U_j$, if $r^*(i)<r(f)$, from (29) and the fact that $i\in U_j$, the number of flows whose rates are smaller than $r(f)$ in D^* will be greater than the number of flows whose rates are smaller than $r(f)$ in D, which contradicts that D^* is the optimal lexicographic rate assignment, and this proves Equation 30. Similarly if $r^*(j)<r(f)$, it contradicts D^* is the optimal lexicographic rate assignment and this proves Equation 31.

Note that $C_j=e_f\sum_{i\in R_j\cup U_j} r_j(i)+e_s r_j(j)$. Because $r(j)=r(f)$, and together with (29), (30), and (31), if $f\ne j$, we have

$$\begin{aligned}
r(f) &= \frac{C_j-e_f\sum_{i\in R_j\cup U_j-\{f\}} r_j(i)-e_s r_j(j)}{e_f}\\
&= \frac{C_j-e_f\sum_{i\in R_j} r_j(i)-e_f\sum_{i\in U_j-\{f\}} r_j(i)-e_s r_j(j)}{e_f}\\
&= \frac{C_j-e_f\sum_{i\in R_j} r_j(i)-e_f\sum_{i\in U_j-\{f\}} r_j(j)-e_s r_j(j)}{e_f}\\
&\ge \frac{C_j-e_f\sum_{i\in R_j} r^*(i)-e_f\sum_{i\in U_j-\{f\}} r^*(i)-e_s r^*(j)}{e_f}\\
&= \frac{C_j-e_f\sum_{i\in R_j\cup U_j-\{f\}} r^*(i)-e_s r^*(j)}{e_f}\\
&= r^*(f)
\end{aligned}$$

which contradicts that $r(f)<r^*(f)$.

If $f=j$, we have

$$\begin{aligned}
r(f) &= \frac{C_j-e_f\sum_{i\in R_j\cup U_j} r_j(i)}{e_s}\\
&= \frac{C_j-e_f\sum_{i\in R_j} r_j(i)-e_f\sum_{i\in U_j} r_j(i)}{e_s}\\
&= \frac{C_j-e_f\sum_{i\in R_j} r_j(i)-e_f\sum_{i\in U_j} r_j(j)}{e_s}\\
&\ge \frac{C_j-e_f\sum_{i\in R_j} r^*(i)-e_f\sum_{i\in U_j} r^*(i)}{e_s}\\
&= \frac{C_j-e_f\sum_{i\in R_j\cup U_j} r^*(i)}{e_s}\\
&= r^*(f)
\end{aligned}$$

which also contradicts that $r(f)<r^*(f)$. Therefore, D must be the optimal lexicographic rate assignment, and this completes the proof.

□