Battery-Aware Router Scheduling in Wireless Mesh Networks *

Chi Ma, Zhenghao Zhang and Yuanyuan Yang
State University of New York, Stony Brook, NY 11794, USA

Abstract

Wireless mesh networks recently emerge as a flexible, low-cost and multipurpose networking platform with wired infrastructure connected to the Internet. A critical issue in mesh networks is to maintain network activities for a long lifetime with high energy efficiency. As more and more outdoor applications require long-lasting, high energy efficient and continuously-working mesh networks with battery-powered mesh routers, it is important to optimize the performance of mesh networks from a battery-aware point of view. Recent study in battery technology reveals that discharging of a battery is nonlinear. Batteries tend to discharge more power than needed, and reimburse the over-discharged power later if they have sufficiently long recovery time. Intuitively, to optimize network performance, a mesh router should recover its battery periodically to prolong the lifetime. In this paper, we introduce a mathematical model on battery discharging duration and lifetime for wireless mesh networks. We also present a battery lifetime optimization scheduling algorithm (BLOS) to maximize the lifetime of battery-powered mesh routers. Based on the BLOS algorithm, we further consider the problem of using battery powered routers to monitor or cover a few hot spots in the network. We refer to this problem as the Spot Covering under BLOS Policy problem (SCBP). We prove that the SCBP problem is NP-hard and give an approximation algorithm called the *Spanning Tree Scheduling* (STS) to dynamically schedule mesh routers. The key idea of the STS algorithm is to construct a spanning tree according to the BLOS Policy in the mesh network. The time complexity of the STS algorithm is $O(r)$ for a network with r mesh routers. Our simulation results show that the STS algorithm can greatly improve the lifetime, data throughput and power consumption efficiency of a wireless mesh network.

Keywords: wireless mesh networks, mesh routers, power scheduling, energy efficiency, battery models, battery-awareness, lifetime optimization.

*The research work was supported in part by NSF grant numbers CCR-0207999 and ECS-0427345 and ARO grant number W911NF-04-1-0439.

1 Introduction

Recently wireless mesh networks have emerged as a flexible, low-cost extension to the wired network infrastructure [1, 2, 3]. A mesh network is a hybrid network which consists of a mix of fixed routers and mobile clients interconnected via access points. For example, in MIT's roofnet mesh network [4], a neighborhood can easily build a community mesh network by setting up a few mesh routers with flexible mesh connectivities among houses to support distributed storage, data access, and video streaming. Among the applications of wireless mesh networks, one of the most important applications is the outdoor mesh networks. Outdoor mesh networks are usually setup in Disneyland, outdoor assemblages and stadiums to reduce the cost of setting up Ethernet cables [2].

In general, a wireless mesh network is composed of three components: access points (AP), mesh routers and mesh clients. Fig.1 illustrates the architecture of a wireless mesh network. Unlike a traditional ad hoc network, which is an isolated self-configured wireless network, the mesh network architecture introduces a hierarchy with wireless routers communicating between mesh clients and APs [3]. A typical wireless mesh network usually has 30 to 100 mesh routers. The fixed APs are wired to connect to the Internet to provide high-bandwidth connections to the Internet backbone. The meshing among wireless routers and APs creates a wireless *backhaul* communication system [1]. The backhaul provides each mobile client with a limited number of entry points connected to the Internet [1]. These entry points, along with the APs, are usually referred to as *Hot Spots*. As the middle layer between the APs and mesh clients, mesh routers must cover all these hot spots. Mesh clients have more varieties of devices compared to mesh routers. These devices can be laptops, tablet PCs, PDAs, IP phones, RFID (Radio Frequency ID) readers, BACnet (Building Automation and Control networks) controllers, and many other types of widely used wireless devices.

A critical issue in mesh networks is to maintain the network activities for a long lifetime with high energy efficiency. As more and more outdoor applications require long-lasting, high energy efficient and continuously-

1-4244-0054-6/06/$20.00

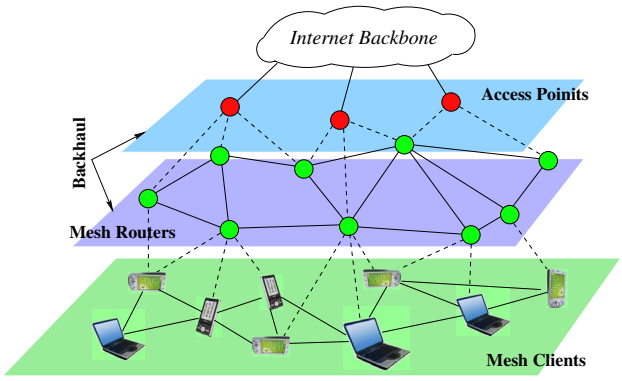

Figure 1. Architecture of a wireless mesh network.

working mesh networks with battery-powered mesh routers, it is important to prolong the lifetime of wireless routers and optimize their performance. Nowadays the batteries on most mesh routers can work for at most a few hours. As an example, the HotPort [5] series outdoor mesh router can continuously work for about two hours on battery. On the other hand, most outdoor applications, such as Disneyland, require a mesh network with a fairly long lifetime. Improving battery performance in mesh routers can greatly improve the overall network communication performance. Thus, carefully scheduling and budgeting battery power in mesh networks has become an urgent and important issue in mesh network design.

In this paper, we propose an approach to maximizing the lifetime of mesh networks from a battery-aware point of view. Recent study in battery technology helps us better understand the battery behavior. Unlike what people used to believe, the energy consumed from a battery is not equivalent to the energy dissipated in the device. When discharging, batteries tend to consume more power than needed, and can reimburse the over-consumed power later. The process of the reimbursement is often referred to as *battery recovery*. [6, 7, 8] conducted experiments on nickel-cadmium battery and lithium-ion battery, which are two commonly used types of batteries on wireless mesh routers. The results show that the over-consumed energy might take up to 30% of the total battery capacity. In other words, by carefully capturing the behavior of batteries and employing battery-aware mesh router scheduling algorithms, we may dramatically increase the lifetime of mesh routers, and as a result, the lifetime of wireless mesh networks. Therefore, the first important issue here is how to precisely capture and predict battery behavior with an accurate mathematical model.

Several analytical models on battery discharging behavior have been developed in recent years [6, 9]. Although these battery models are computational approaches and independent of battery chemistry, they are not quite suitable for implementation in mesh networks due to their low accuracy and high computational complexity. More recently, an on-line computable battery model was proposed in [7]. This model gave a simple way to measure the battery discharging behavior by using the recovery length. However, so far no study has been done on optimizing battery lifetime based on the relationships between discharging time and total battery lifetime. In this paper, we study these relationships, and present a battery lifetime optimization scheduling algorithm (BLOS) to maximize the lifetime of battery-powered mesh routers.

Based on the BLOS algorithm, we further consider the problem of using battery powered routers to monitor or cover hot spots. When all mesh routers follow the BLOS policy, our goal is to keep the mesh network covering all hot spots for as long as possible. We refer to this problem as the *Spot Covering under BLOS Policy problem (SCBP)*. We first reduce the SCBP problem to the Vertex Covering (VC) problem, and prove that the SCBP problem is NP-hard. We then give an approximation algorithm to dynamically schedule mesh routers. The key idea of the algorithm is to construct a spanning tree in the mesh network to ensure all hot spots covered. Thus the algorithm is referred to as the *Spanning Tree Scheduling* (STS) algorithm. The time complexity of the STS algorithm is $O(r)$ for a network with r mesh routers. We also validate the algorithm through simulations. Our results demonstrate that the STS algorithm can greatly improve the lifetime, data throughput and power consumption efficiency of a wireless mesh network.

2 Battery Discharging and Recovery

Nickel-cadmium and lithium-ion batteries are the most commonly used batteries in wireless mesh routers and other outdoor computing and communication devices. Such a battery consists of cells arranged in series, parallel, or a combination of both. Two electrodes: an anode and a cathode, separated by an electrolyte, constitute the active material of each cell. When the cell is connected to a load, a reduction-oxidation reaction transfers electrons from the anode to the cathode. To illustrate this phenomenon, Fig. 2 shows a simplified symmetric electrochemical cell. In a fully charged cell (Fig. 2(a)), the electrode surface contains the maximum concentration of active species. When the cell is connected to a load, an electrical current flows through the external circuit. Active species are consumed at the electrode surface and replenished by diffusion from the bulk of the electrolyte. However, this diffusion process cannot keep up with the consumption, and a concentration gradient builds up across the electrolyte (Fig. 2(b)). A higher load electrical current I results in a higher concentration gradient and thus a lower concentration of active species at the electrode surface [10]. When this concentration falls, the battery voltage drops. When the voltage is below a certain cutoff threshold, the electrochemical re-

action can no longer be sustained at the electrode surface, and the battery stops working (Fig. 2(e)). The electro active species that have not yet reached the electrode are not used. We refer to the unused charge as *discharging loss*. The discharging loss is not physically "lost," but simply unavailable due to the lag between the reaction and the diffusion rates. Before the battery dies, if the battery current I is reduced to zero or a very small value, that is, in battery recovery (Fig. 2(c)), the concentration gradient flattens out after a sufficiently long time, reaching equilibrium again. The concentration of active species near the electrode surface following this recovery period makes unused charge available again for extraction (Fig. 2(d)). Effectively recovering the battery can reduce the concentration gradient and recover discharging loss, hence prolong the lifetime of the battery (Fig. 2(f)). Experiments on nickel-cadmium battery and lithium-ion battery show that the discharging loss might take up to 30% of the total battery capacity [6]. Hence, precisely modeling battery behavior is essential for optimizing system performance in wireless mesh networks.

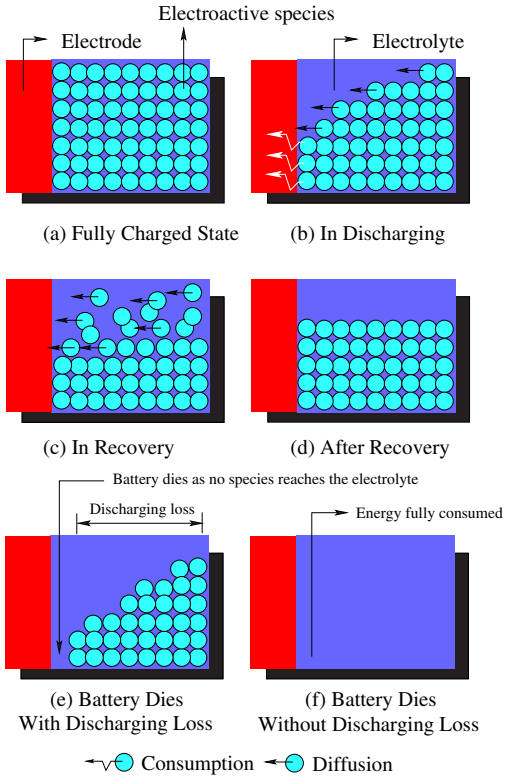

Figure 2. Battery operation at different states.

The key idea of battery awareness is to use a mathematical model to capture the special behavior of the battery, and then schedule the battery-powered device according to its discharging loss. By recovering the battery, we can reimburse the discharging loss and in turn improve the lifetime of the device. Several mathematical battery models have been introduced in recent years [6, 7, 8, 11, 12] to capture the battery behavior. We will discuss the battery modeling in the next section.

3 Modeling Battery Discharging Behavior

In this section we first briefly analyze and compare existing battery models. Then we introduce our battery model based on the scenario of epoch time discharging and recovery. We also give a method to simplify the computation of discharging loss in the model.

Mathematical models that can capture the battery discharging behavior have been developed [6, 7, 8, 11, 12]. These models are independent of battery chemistry. [11] provided an abstract model to describe battery recovery behavior. This model treats discharging and recovery as a negative exponential function and represents them as a transient stochastic process. However, as pointed out in [12], this method requires considerable effort to configure, and its accuracy and computational complexity are barely acceptable. Therefore, it has limited utility for the implementation in wireless mesh networks. [6] proposed an analytical battery model which estimates the battery recovery behavior. The analysis is based on a one-dimensional model of diffusion in a finite region. However, this model is very complex and requires long computing time. An on-line computable discrete time battery model was introduced in [7, 8] for ad hoc wireless networks, in which the battery lifetime is divided into a sequence of discrete time slots with a fixed slot length. This model can effectively capture the effect of battery discharging and recovery.

Although all previous analytical battery models can fairly describe the battery discharging behavior in mathematical expressions, none of them consider optimizing the battery lifetime based on the relationships among the discharging time, recovery time and overhead to switch the device from active to idle. In this section, we give a battery model based on continuous epoch time discharging and recovery for optimization in mesh router scheduling.

Consider a scenario where a battery is turned active for δ_i time, and then turned idle for τ_i time ($i = 1, 2, \dots$). The active-idle period is repeated until the battery dies. Note that by turning a battery into idle, we let it "sleep" with very low current on it. During its idling the battery recovers its over-charged capacity.

In our battery model, battery is discharged in each duration i with length δ_i, where δ_i may not be equal to δ_j if $i \neq j$. α is the initial battery capacity. A duration δ_i is called an epoch. We use I_i, α_i, α'_i and ζ_i to denote the discharging current through the battery, the battery capacity at the beginning of the i_{th} epoch, the battery capacity at the

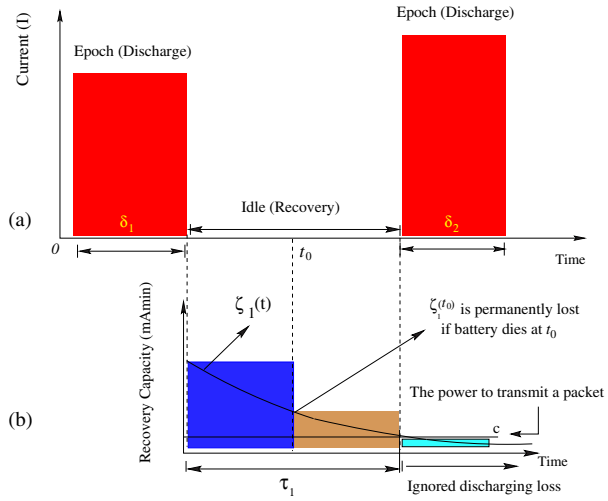

Figure 3. (a) Discharging of a battery in δ_1 and δ_2 epochs. Battery is idle between epochs. (b) The capacity $\zeta_1(t)$ is discharged in epoch δ_1 and recovered gradually after that. $\zeta_1(t)$ after time $\delta_1 + \tau_1$ is ignored. If this battery dies at t_0, $\zeta_1(t_0)$ is permanently lost.

end of the i_{th} epoch, and the discharging loss in the i_{th} epoch, respectively. We use T to denote the entire lifetime of the battery. An epoch δ_i is followed with a recovery period of length τ_i. Without loss of generality, we assume that the discharging current I is a constant in a certain epoch.

The condition of a battery at the i_{th} epoch is measured by its discharging loss at that time. A high discharging loss indicates a "fatigue" battery which needs some recovery, while a battery with low discharging loss is well recovered. Intuitively, an energy efficient scheduling algorithm should always choose routers with well recovered batteries. Therefore, a good battery model should be able to calculate the discharging loss at any epoch.

The following analytical model can be used to compute the battery discharging loss at an epoch. The model that computes the energy dissipated by the battery during the i_{th} epoch $[\bar{t}, \bar{t} + \delta_i]$ is

$$\alpha_i - \alpha_i' = I_i \times F(T, \bar{t}, \bar{t} + \delta_i, \beta) \qquad (1)$$

where

$$F(T, \bar{t}, \bar{t} + \delta_i, \beta) = \delta_i + 2 \sum_{m=1}^{\infty} \left[\frac{e^{-\beta^2 m^2 \bar{t}} - e^{-\beta^2 m^2 (\bar{t} + \delta_i)}}{\beta^2 m^2} \right]$$

This model can be interpreted as follows. The dissipated energy during the i_{th} epoch is $\alpha_i' - \alpha_i$ in (1). It contains two components: The first term, $I_i \times \delta_i$, is simply the energy consumed in the device during $[\bar{t}, \bar{t} + \delta_i]$. The second term, $2I_i \times \sum_{m=1}^{\infty} \left[\frac{e^{-\beta^2 m^2 \bar{t}} - e^{-\beta^2 m^2 (\bar{t} + \delta_i)}}{\beta^2 m^2} \right]$ is the amount of battery discharging loss in the epoch. It can be seen that the discharging loss decreases as the lifetime T increases. β (> 0) is a constant, which is an experimental chemical parameter and may vary from battery to battery. The larger the β, the faster the battery diffusion rate, hence the less the discharging loss. Next we show how the model in (1) can be used to calculate the discharging loss at a given epoch.

As defined earlier, ζ_i is consumed in the i_{th} epoch and recovered in the next τ_i time. Clearly,

$$\zeta_i(t) = 2I_i \times \sum_{m=1}^{\infty} \left[\frac{e^{-\beta^2 m^2 (\bar{t} + t)} - e^{-\beta^2 m^2 (\bar{t} + \delta_i + t)}}{\beta^2 m^2} \right] \qquad (2)$$

where $\zeta_i(t)$ is the residual discharging loss at time t. It should be mentioned that discharging loss $\zeta_i(t)$ is only a potential type of energy. For example, in Fig. 3, if the battery dies at time t_0, the battery permanently loses the energy $\zeta_i(t_0)$.

As can be observed, the recovery of $\zeta_i(t)$ continues from $t + \delta_i$ to ∞. In practice, we can simplify the computation of ζ_i as follows. Assume c is a fairly small constant, which is the power to transmit a packet. By observing (2), if $\zeta_i(\tau_i)$ is less than c, we can ignore the discharging loss after time $\bar{t} + \delta_i + \tau_i$. Thus, after τ_i time of recovery, the battery can be considered to be well-recovered.

In summary, in this section we first briefly summarize and compare previous work on battery modeling. We then introduce our battery model based on the δ epoch time discharging and τ time recovery scenario. We also give a method to simplify the computation of discharging loss ζ. Next we will apply this model to battery lifetime optimization in wireless mesh networks.

4 Battery Lifetime Optimization Scheduling (BLOS)

As discussed earlier, given a battery with initial capacity α discharged under current I from 0 to δ_1, the battery capacity decreases nonlinearly during time $[0, \delta_1]$, and afterwards it gradually recovers the capacity to the value of $\alpha - I \times \delta_1 - \zeta(t)$, where $\zeta(t)$ is the discharging loss at time $\delta_1 + t$. By periodically recovering the battery, we can reduce $\zeta(t)$, and in turn increase the battery lifetime. Fig. 4 gives an example that shows the simulated lifetime prolonged by considering the battery recovery. In this case we assume the battery capacity is $\alpha = 4.5 \times 10^4 mA\text{min}$ and $\beta = 0.4$, which are typical values of a chemical battery [6]. The discharging current is $I_d = 900mA$. Under the greedy mode, the battery is continuously discharged until the battery dies. The total lifetime is 116 minutes. In the battery-aware mode, the battery is discharged in each epoch, and recovered in the next recovery period. The total working time is increased by 14.7%.

Now given a battery, in order to optimize the battery lifetime we need to determine when an epoch should start and

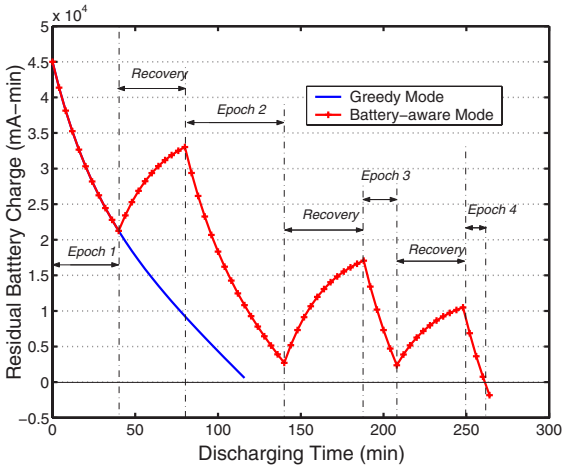

Figure 4. Simulated lifetime under the greedy mode and battery-aware mode. The lifetime under the greedy mode is 116 **minutes. Under the battery-aware mode, the battery is discharged in each epoch, and recovered in the subsequent recovery time. The total working time is increased by** 14.7%**.**

how long the epoch should last. We adopt an iterative way to find an optimal scheduling policy for battery-powered mesh routers.

First consider a battery being discharged in $[\overline{t}, \overline{t}+\delta_i]$, after that period it takes τ_i time to fully recover the discharging loss. From (2) we know the length of recovery time τ_i is only dependent on δ_i if given α, I and β. Therefore, there is a function to calculate τ_i, and we call it *GetTau*. That is,

$$\tau_i = GetTau(\alpha, I, \beta, \delta_i)$$

Under the greedy mode, where the battery is continuously discharged under current I until it dies, we define the total lifetime function as

$$Lifetime(\alpha, \beta, I)$$

This lifetime function can be easily obtained from (1). Now let's consider the battery discharging behavior. Our goal is to maximize the total battery discharging time. We assume the mesh router is turned active n times. Let δ_i be the length of the i_{th} active time, $1 \leq i \leq n$. The problem can be formulated as: Given an initial battery capacity, how to choose the lengths of δ_1, δ_2, ..., δ_n such that the battery has the longest working time.

In mathematical terms, let $\overline{T}_i = \sum_{j=i}^{n} \delta_j$. A policy P is defined as a schedule for a router. P describes the lengths of time $\delta_1, \tau_1, \delta_2, \tau_2, \ldots$ until the battery is used up. An optimal policy is a policy by which $\overline{T}_1 = \sum_{i=1}^{n} \delta_i$ is maximized. Let α_i be the residual charge at the beginning of the i_{th} active time. $\overline{T}_i$ depends on α_i and the policy P. Given α_i, let $\overline{T}_i$ under the optimal policy be

$$\overline{T}_i = P_i(\alpha_i) \tag{3}$$

Then what we need to find is $P_1(\alpha_1)$.

Suppose that $P_i(\alpha_i)$ has been found, that is, for any given α_i, we know the optimal lengths of δ_i, δ_{i+1}, ..., δ_n such that $\overline{T}_i$ is maximized. Now we want to find $P_{i-1}(\alpha_{i-1})$. Define the overhead to switch between active and idle is ϵ. Note that

$$\overline{T}_{i-1} = \delta_{i-1} + \overline{T}_i \tag{4}$$

Also note that if α_{i-1}, δ_{i-1} and τ_{i-1} are given, α_i is determined and can be written as

$$\alpha_i = f(\alpha_{i-1}, I, \delta_{i-1}, \tau_{i-1}) - \epsilon \tag{5}$$

where function $f(\alpha_x, I, \delta_x, \tau_x)$ describes the residual battery power after being discharged for δ_x time under current I and being recovered for τ_x time.

In this case $\overline{T}_{i-1}$ can be maximized by adopting the optimal policy for the $\overline{T}_i$ we have already found. From (4), (3) and (5), we obtain the maximum value of $\overline{T}_{i-1}$

$$\overline{T}_{i-1} = \delta_{i-1} + P_i(f(\alpha_{i-1}, I, \delta_{i-1}, \tau_{i-1}) - \epsilon) \tag{6}$$

Note that (6) is only a function of α_{i-1} and δ_{i-1}. By varying δ_{i-1}, the maximum value of $\overline{T}_{i-1}$ under a given α_{i-1} can be found, which is the $P_{i-1}(\alpha_{i-1})$ we want to find. In practice, we can increase δ_i by a constant $\overline{\delta}$ step by step. Since after each τ_i time recovery, the battery is well-recovered, function f has a very simple form

$$\alpha_i = f(\alpha_{i-1}, I, \delta_{i-1}, \tau_{i-1}) = \alpha_{i-1} - I * \delta_{i-1}$$

Now we are in the position to describe the Battery Lifetime Optimization Scheduling (BLOS) algorithm. As defined earlier, n is the number of epochs during the lifetime of a battery. Clearly, $\overline{T}_1$ is not maximized when $n = 1$, because this is the greedy mode. As n increases, the battery periodically gets recovery, hence $\overline{T}_1$ is also increased. However, this increasing is not monotonic because the accumulation of overhead ϵ also increases. The accumulated overhead in turn reduces $\overline{T}_1$. Therefore, there must exist an n such that $\overline{T}_1$ obtains its maximum value. The BLOS algorithm is used to find the optimum $\overline{T}_1$. The algorithm can be described as follows. Initially, we let $n = 1$, and calculate the optimum policy for the given n. Each step we increase n by 1, and calculate the new policy for this n until there is an optimum $\overline{T}_1$ with a peak value.

The BLOS algorithm employs an iterative approach to finding $P_1(\alpha_1)$ for a given n. We call it $GetBlos$. Table 1 gives the details of the $GetBlos$ procedure. At the beginning, we find $P_n(\alpha_n)$. In the subsequent steps, $P_i(\alpha_i)$

Table 1. Finding Optimal Policy P for Given n

Procedure *GetBlos* (α, i)

begin
  if $(i = n)$
    begin
      δ_n = *lifetime*(α, β, I);
      return δ_n;
    end
  else
    begin
      $T = 0$;
      for $j = 1$ **to** $\lfloor \frac{\alpha}{I \times \overline{\delta}} \rfloor$
        begin
          calculate $\delta_{i+1}, \delta_{i+2}, \ldots, \delta_n$
            by calling *GetBlos* $(\alpha - j \times \overline{\delta} \times I - \epsilon, i + 1)$;
          $T_i = \sum_{k=i+1}^{n} \delta_k + j \times \overline{\delta}$;
          if $T < T_i$ *then* $T = T_i$;
        end
      return T;
    end
end

can be determined according to the results of $P_{i+1}(\alpha_{i+1})$. This way we can finally find $P_1(\alpha_1)$. Since in the last active time the battery works until exhausted and there is no policy involved, it can be simply obtained from the battery model.

Finally, we give the complete BLOS algorithm based on procedures $GetBlos$, $GetTau$ and $Lifetime$. Table 2 describes the algorithm in pseudo-codes. From $n = 1, 2, \ldots$, procedure *GetBlos* is called to calculate the optimal lifetime $\overline{T}_1$ for the given n, then n is increased until a peak value of $\overline{T}_1$ is obtained. The policy P at this n is the optimal policy we want. Finally we can use *GetTau* function to specify each τ_i for δ_i, to ensure the battery is well recovered after δ_i time discharging.

5 Hot Spot Covering Under BLOS Policy

In Section 4 we gave an optimal scheduling algorithm to maximize the lifetime of mesh routers. In this section we consider the problem of using battery powered mesh routers to monitor or cover hot spots. Our goal is to let routers all follow the optimal battery active-idle policy, while keeping all hot spots covered for as long as possible. We call this problem the *Spot Covering under BLOS Policy* problem, and refer to it as the *SCBP* problem.

Like many other covering problems, the SCBP problem is NP-hard. In the following we give a proof of it. We show that even in the simplest case in which the optimal policy is to "use till exhausted," the decision version of the SCBP problem is NP-hard.

The decision version of the SCOP problem can be formally written as follows. Given a set of routers and a set of spots, where each router may cover several spots and each spot may be covered by several routers. If all routers have the same battery life T, does there exist a schedule by which all spots are covered in $[0, t']$, where t' is a positive constant? Such a schedule is called a *valid* schedule. Note that since all routers, by the optimal policy, must work till the battery is exhausted after turned on, the only thing we can control is when to turn on each router.

Table 2. Battery Lifetime Optimization Scheduling (BLOS)

Procedure *BLOS*

begin
  $n = 0$;
  $T = 0, \overline{T}_1 = 0$;
  repeat
    $T = \overline{T}_1$;
    $n = n + 1$;
    calculate $\overline{T}_1$ by calling *GetBlos*(α, n);
  until $\overline{T}_1 < T$;
  calculate τ_i by calling $GetTau(\alpha_i, I, \beta, \delta_i)$,
   for $i = 1, 2, \ldots, n$;
end

Lemma 1 *The decision version of the SCBP problem is equivalent to the* Subset Partition *problem (SP).*

Proof. First we give some properties of a valid schedule. Since all spots must be covered at time 0, some routers must have been turned on at time 0 and these routers must collectively cover all spots. This can be seen without difficulty. Now suppose under a valid schedule, a router is turned on at time t where $0 < t < T$. We can safely delay the turn on time of this router to T, since all spots it covers must be covered during $[t, T]$ by the routers turned on at time 0, and all spots it covers during $[T, T + t]$ are still covered by itself. As a result of this fact, if there is a valid schedule, there will be a valid schedule by which all routers are turned on at times which are multiples of T. Therefore, the problem reduces to partitioning the routers into groups, where each group should collectively cover all spots and the number of groups multiplied by T must be larger than t'. Thus the decision version of the SCBP problem is equivalent to the following problem, which we call the *Subset Partition* problem (SP): Given a whole set (the spots) and some subsets (the routers), can the subsets be partitioned into k groups where each group of subsets covers all the elements in the whole set? ■

Now we have proved that the SCBP problem is equivalent to the SP problem. Next we show that the SP problem is NP-hard, and as a result, the SCBP problem is NP-hard.

Lemma 2 *The SP problem is NP-hard.*

Proof. Consider the *Vertex Covering* problem which is known to be NP-hard [14, 13]: Given a graph and k colors, can each vertex be given a color such that no adjacent vertices have the same color? The SP problem can be shown to be NP-hard as follows.

Given any instance of the VC problem G, construct an instance of the SP problem in 3 steps.

1. For each edge E_i in G, create an element called e_i.

2. For each vertex V_a in G, create a subset called S_a. Subset S_a contains element e_i if V_a is incident to E_i in G. Note that at this time each element is in exactly 2 subsets, i.e., covered by two subsets.

3. Then for each e_i, create $k-2$ identical subsets s_1^i, s_2^i, $\ldots$, s_{k-2}^i, where each of these subsets contains only one element which is e_i.

Note that now each element is covered by exactly k subsets. If all the subsets created can be partitioned into k groups where each group collectively covers all the elements, the subsets covering the same element must belong to different groups. Give color C_j to vertex V_a if subset S_a is in the j_{th} group in the partition. Apparently, two vertices V_a and V_b will be given different colors if they are adjacent to each other in graph G. As a result, the partition determines a k-coloring in the VC instance.
■

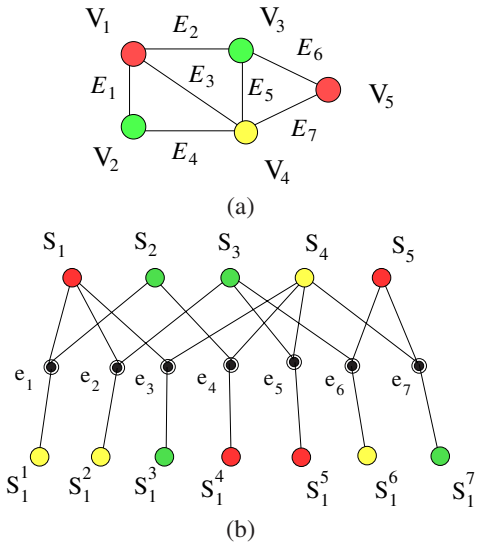

Figure 5. An example to show the SP instance for a graph G. (a) Graph G. (b) SP instance for (a).

As an example, the SP instance for the graph shown in Fig. 5(a) is given in Fig. 5(b). Letting $k = 3$, the subsets in the SP instance can be partitioned into 3 groups where subsets in the same group are shown in the same color: $\{S_1, S_5, s_1^4, s_1^5\}$, $\{S_2, S_3, s_1^3, s_1^7\}$ and $\{S_4, s_1^1, s_1^2, s_1^6\}$. It determines a valid 3-coloring in the VC instance, as shown in Fig. 5(a).

Theorem 1 *The SCBP problem is NP-hard.*

Proof. Given the proofs of *Lemma* 1 and *Lemma* 2, we can draw the conclusion that the SCBP problem is NP-hard. ■

Since there is no optimal SCBP decision in polynomial time, in the next section, we will give an approximation algorithm for the SCBP problem under BLOS scheduling.

6 Spanning Tree Mesh Router Scheduling under BLOS Policy

In this section we present an approximation algorithm to ensure wireless mesh routers to cover hot spots under BLOS policy. The key idea of our algorithm is to construct a spanning tree in the mesh network to ensure all spots to be covered. The spanning tree is reconstructed periodically according to the BLOS policy. A new tree is constructed to recover the mesh routers in the old tree. We refer to this algorithm as the *Spanning Tree Scheduling* (STS).

We assume that there are r mesh routers in the network. The STS algorithm consists of two steps. First, each mesh router computes its optimal policy by BLOS. Then a spanning tree is constructed from all spots to be covered. $m_1, m_2, \ldots, m_s$ are the s nodes selected in this tree. They are turned to active for δ_1 time, where $\delta_1 = \min\{\delta_1^{m_1}, \delta_1^{m_2}, \ldots, \delta_1^{m_s}\}$. All other nodes are turned into idle during this time. After δ_1 time, a new tree is reconstructed. The STS algorithm turns all nodes not on the spanning tree into idle for recovery. To avoid a router being selected again in the next round, STS gives a weight to each of them. When we select spanning tree nodes, a node with lower weight has higher priority. Table 3 gives the STS algorithm in detail. For a network with r routers, it is easy to see that the time complexity to construct a spanning tree is $O(r)$. In order to verify whether a spanning tree is constructed in the STS algorithm, we let each node broadcast a short packet via selected routers. Such packets overhead is very minor considering the number of mesh routers is fairly small ($30 - 100$), and the radius of a router is fairly long (300 feet) in a mesh network [1, 2]. Fig. 6 gives an example of how the STS algorithm works.

7 Performance Evaluations

In this section we evaluate the performance of the STS algorithm under BLOS through simulations. The simulation consists of two parts: stand-alone router performance

Table 3. Spanning Tree Scheduling Algorithm (STS)

Procedure *STS*
begin
Initially each mesh router is assigned a weight;
$\tau = 0, h = 1$;
Each mesh router computes its policy by BLOS;
repeat
Each hot spot is colored as a *black* node;
Each mesh router is colored as a *white* node;
while *the spanning tree is not connected* **do**
begin
Each *black* node k selects a *white* node i,
where Distance(i, k) <Radius(i),
and weight(i)=
$\min\{$weight$(j)|$distance(j, k) <Radius$(j)\}$;
i is colored *black*;
end
Obtain a spanning tree with s routers: $m_1, m_2, \ldots, m_s$;
Mesh routers not in the spanning tree are
turned idle for recovery;
This spanning tree works for
$\max\{\tau, \min\{\delta_h^{m_1}, \delta_h^{m_2}, \ldots, \delta_h^{m_s}\}\}$ time;
$\tau = \max\{\tau_h^{m_1}, \tau_h^{m_2}, \ldots, \tau_h^{m_s}\}$;
Each node $m_l (l = 1 \ldots s)$ does:
Weight(m_l)=Weight(m_l)+δ_{weight};
$h = h + 1$;
until no spanning tree can be constructed;
end

(Section 7.1), and mesh network performance (Section 7.2). We evaluate the performance with respect to router lifetime, data throughput, network lifetime and power consumption.

7.1 Stand-Alone Router Performance

In this subsection we evaluate the lifetime of a stand-alone mesh router under BLOS policy. For comparison purpose we also simulated two other battery scheduling algorithms: *Greedy Scheduling* (GS) and *Fixed-Time Scheduling* (FTS), on the same router battery. Under the GS a battery is discharged without recovery during its lifetime. On the other hand, the FTS blindly turns the battery between active and idle in simple fixed time slots. Our simulation results show that these two algorithms are much less energy efficient than the BLOS algorithm. To further strengthen our conclusion, we also evaluated battery lifetime performance for different batteries with various initial capacity α, chemical parameter β and discharging current I. Fig. 7 shows the results. As can be seen, battery lifetime can be prolonged by up to 21% under BLOS.

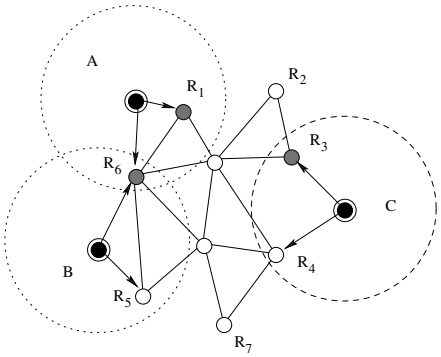

Figure 6. An example to show one step of constructing a spanning tree under the STS algorithm. A, B **and** C **are the hot spots to be covered. Weight**$(R_1) = 0$**, Weight**$(R_3) = 0$**, Weight**$(R_4) = 10$**, Weight**$(R_5) = 20$**, and Weight**$(R_6) = 10$**. At this step,** R_1, R_3 **and** R_6 **are selected based on their weights.**

7.2 Mesh Network Performance

In this subsection we evaluate the performance of the STS algorithm under BLOS in mesh networks. We consider the number of alive nodes, network lifetime, power dissipation and data throughput in the simulation. We assume that the mesh network is setup in a 150×150 field. Wireless mesh routers and hot spots are randomly distributed in the field. Fig. 8 shows an example network with 50 routers and 15 hot spots.

In our simulation we consider several possible mesh networks with different numbers of mesh routers and hot spots. To model real-world applications, we also evaluated our algorithm for heterogeneous mesh networks, that is, networks with mesh routers having various initial battery capacity α and various β. This may be the case when a mesh network is implemented using mesh routers of different brands. The routers from different companies come with very different α and β values. For comparison purpose we implemented two approaches: (i) Greedy Scheduling mode (GS), where all routers are continuously discharged; (ii) Greedy Spanning Tree mode (GST), where a spanning tree is constructed without considering their battery status. After at least one router in the spanning tree exhausts its power, a new spanning tree is constructed by alive routers. We evaluated the performance in terms of alive routers, power dissipation and data throughput.

Alive Routers. We first consider the number of alive routers during the network lifetime. Since the BLOS algorithm enables routers to use up battery power gradually, there should be more alive routers. The decreasing of the number of alive nodes is shown in Fig. 9 for various numbers of routers and hot spots, and α and β values. We can see that since the battery-aware scheduling is sensitive to

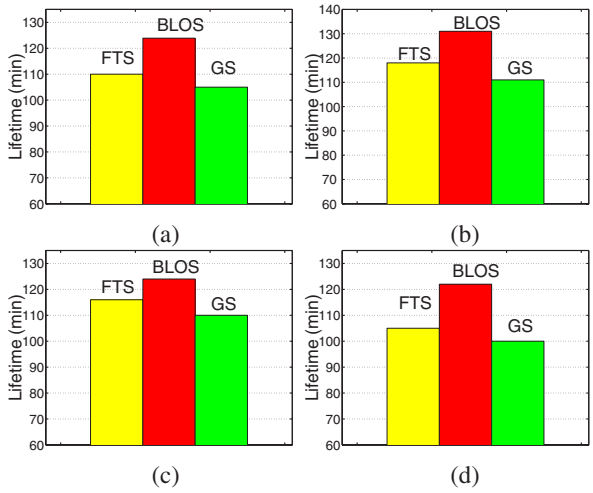

Figure 7. Simulated lifetime under BLOS, Greedy Scheduling and Fixed-Time Scheduling for various α, β, and I. (a) $\alpha = 4.5 \times 10^4 mA$min, $\beta = 0.4$, $I = 900mA$, $\epsilon = 100mA$min; (b) $\alpha = 5 \times 10^4 mA$min, $\beta = 0.4$, $I = 900mA$, $\epsilon = 100mA$min; (c) $\alpha = 4.5 \times 10^4 mA$min, $\beta = 0.5$, $I = 900mA$, $\epsilon = 100mA$min; (d) $\alpha = 4.5 \times 10^4 mA$min, $\beta = 0.4$, $I = 1300mA$, $\epsilon = 100mA$min.

battery status, the decreasing under the STS is slower. Also note that the network lifetime is extended as well.

Power Dissipation. We evaluated the network with various number of heterogeneous routers. Fig. 10 shows the power distribution of the routers in the middle of network transmission (at the $60_{th} min$). The X and Y axes show the geographic positions of routers in the network. The Z axis stands for the residual battery power of routers. It can be seen that by adopting the STS, routers can preserve higher battery energy.

Data Throughput. We also evaluated the normalized gross data throughput of the network under three scheduling algorithms. We simulated two networks with different numbers of hot spots and β values. Fig. 11 shows the normalized data throughput. As can be seen, The STS improves the total data throughput because it prolongs the entire network lifetime.

8 Conclusions

In this paper, we have addressed the energy efficient router scheduling problem in wireless mesh networks from a battery-aware point of view. We showed that in practice batteries tend to discharge more power than needed, and reimburse the over-discharged power later if they have sufficiently long recovery time. Given this fact, we studied the relationships between discharging duration and battery lifetime, and introduced a battery lifetime optimization scheduling algorithm (BLOS) to maximize the lifetime of

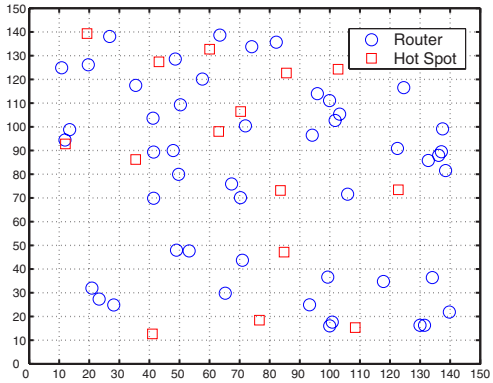

Figure 8. An example of a wireless mesh network with 50 mesh routers and 15 hot spots distributed.

battery-powered mesh router. Based on the BLOS algorithm, we further considered the SCBP problem for monitoring or covering hot spots under BLOS algorithm. We proved that the SCBP is NP-hard, and give an approximation algorithm (STS) with time complexity of $O(r)$ for a network with r mesh routers. Our simulation results show that the STS can greatly improve the lifetime, data throughput and power consumption efficiency of a wireless mesh network.

References

[1] I. Akyildiz, X. Wang and W. Wang, "Wireless Mesh Networks: a Survey," *Computer Networks*, 2004.

[2] R. Bruno, M. Conti and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks," *IEEE Communications*, pp. 123-131, 2005.

[3] R. Draves, J. Padhye and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," *Proc. ACM MobiCom '04*, pp. 114-128, 2004.

[4] D. Aguayo, J. Bicket, S. Biswas, D.S.J. De Couto and R. Morris, "MIT Roofnet Implementation," http://pdos.lcs.mit.edu/roofnet/design/

[5] Firetide Networks Inc., http://www.firetide.com/

[6] D. Rakhmatov and S. Vrudhula, "Energy Management for Battery-Powered Embedded Systems", *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 3, pp. 277-324, Aug. 2003.

[7] Y. Yang and C. Ma, "Battery Aware-Routing in Wireless Ad Hoc Networks – Part I: Energy Model," *Proc. of 19th International Teletraffic Congress (ITC-19)*, pp. 293-302, Sept. 2005.

[8] C. Ma and Y. Yang, "Battery Aware Routing in Wireless Ad Hoc Networks - Part II: Battery-Aware Routing," *Proc. of 19th International Teletraffic Congress (ITC-19)*, pp. 303-312, Sept. 2005.

[9] D. N. Rakhmatov and S.B.K. Vrudhula, "An Analytical High-Level Battery Model for Use in Energy Management of Portable Electronic Systems," *Proc. 2001 IEEE/ACM Int'l Conf. Computer-Aided Design,* pp. 488-93, 2001.

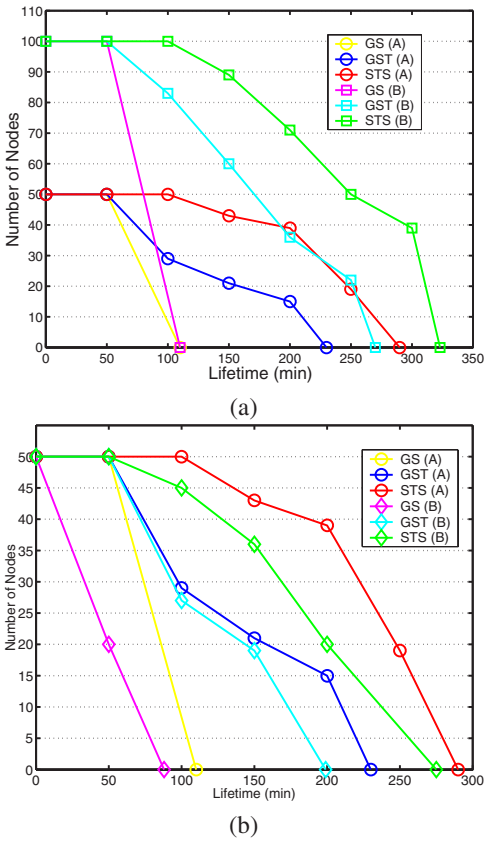

Figure 9. Number of alive nodes in the mesh network. (a) Simulation results for various numbers of routers and hot spots. Network A has 50 routers and 15 hot spots; Network B has 100 routers and 40 hot spots. (b) Simulation results for various α and β values. Network A has identical routers with $\alpha = 4.5 \times 10^4 mA$min and $\beta = 0.4$; Network B is heterogeneous with $\alpha \in [0, 4.5 \times 10^4] mA$min and $\beta \in [0, 0.4]$.

[10] M. Doyle, T. F. Fuller and J. Newman, "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell," *J. Electrochemical Soc.* vol. 140, no. 6, 1993, pp. 1526-33.

[11] D. Panigrahi, et al., "Battery Life Time Estimation of Mobile Embedded Systems," *Proc. of 14th Int'l Conf. VLSI Design*, pp. 57-63, 2001.

[12] R. Rao, S. Vrudbula and D.N. Rakbmatov, "Battery Modeling for Energy-Aware System Design," *IEEE Computer*, vol. 36, pp. 77-87, Dec. 2003.

[13] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, 1978.

[14] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms,* 2nd edition, The MIT Press, Sept. 2001.

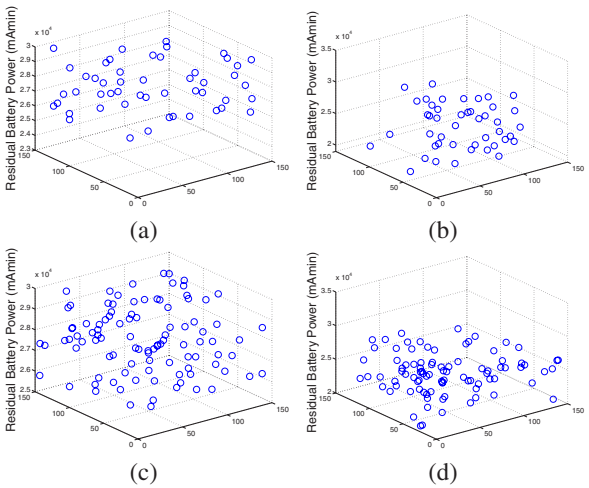

Figure 10. Power dissipation at the 60th minute. (a) 50 routers, STS algorithm; (b) 50 routers, GST algorithm; (c) 100 routers, STS algorithm; (d) 100 routers, GST algorithm; Mesh routers are heterogeneous with random α and β values.

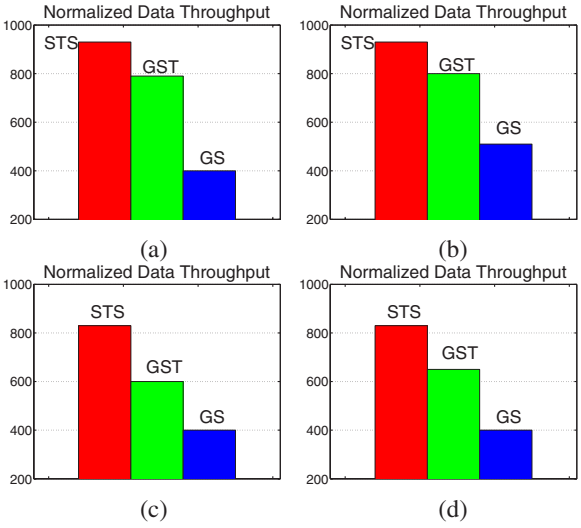

Figure 11. Normalized data throughput in the mesh network. (a) 50 routers and 15 hot spots, routers have identical $\alpha = 4.5 \times 10^4 mA$min and $\beta = 0.5$; (b) 50 routers and 15 hot spots, routers have identical $\alpha = 4.5 \times 10^4 mA$min and $\beta = 0.4$; (c) 50 routers and 20 hot spots, routers have identical $\alpha = 4.5 \times 10^4 mA$min and $\beta = 0.5$; (d) 50 routers and 20 hot spots, routers have identical $\alpha = 4.5 \times 10^4 mA$min and $\beta = 0.4$.