On Optimization of E-Textile Systems Using Redundancy and Energy-Aware Routing

Jung-Chun Kao, *Student Member*, *IEEE*, and Radu Marculescu, *Member*, *IEEE*

Abstract—Recent advances in the electronic device manufacturing technology have opened many research opportunities in pervasive computing. Among the emerging design platforms, "electronic textiles" (or *e-textiles*) make possible a wide variety of novel applications, ranging from consumer electronics to aerospace devices. Due to the harsh environment of e-textile components and battery size limitations, low-power and redundancy techniques are critical for obtaining successful e-textile applications. In this paper, we consider a platform which consists of dedicated components for e-textiles, including computational modules, dedicated transmission lines, and thin-film batteries on fiber substrates. As a theoretical contribution, we address the issue of the energy-aware routing for e-textile platforms and propose an efficient algorithm to solve it. Furthermore, we derive an analytical upper bound for determining the maximum number of achievable jobs over all possible e-textile routing frameworks. From a practical standpoint, for the Advanced Encryption Standard (*AES*) cipher, the routing technique we propose achieves close to or more than 75 percent of this theoretical upper bound. Moreover, compared to the non-energy-aware counterpart, the new routing technique increases the number of encryption jobs by one order of magnitude.

Index Terms—Electronic textile, energy-aware routing, pervasive computing.

1 Introduction

The scaling of device technologies opens many research opportunities in the area of pervasive computing [4]. In particular, it is now possible to fabricate flexible materials with sophisticated computing and communication devices embedded within the material. Such computational fabrics, or *e-textiles*, have many applications, ranging from consumer electronics to military, health, security, and aerospace devices.

A typical e-textile system has to be not only light and mobile, but also withstand wear-and-tear due to frequent washing. Due to the potential stress upon e-textile interconnects, one solution is to use a communication *network* instead of the traditional bus-based architecture. Besides reliable operation, e-textile platforms have strict processing, storage, energy, and size constraints per computational node. Thus, in order to be able to handle complicated applications over an extended period of time, e-textiles need to rely on low-power techniques, as well as exploit redundancy. The target application that runs on an e-textile platform must be appropriately partitioned to meet the strict resource requirements per computational node and be able to self-manage as a whole. In addition, each device must cooperate with its (redundant) duplicates, especially when there is little residual energy left. These additional requirements lead to an interesting and unique design problem, which is the main objective of this paper.

We select the Advanced Encryption Standard (AES) [7] as the main driver application running on an e-textile platform for two reasons. First, AES is a robust encryption algorithm; since data secrecy plays a critical role in pervasive computing, exploring the potential of AES for distributed implementations on e-textile fabrics becomes very important. Second, in June 2004, IEEE ratified the next generation of wireless local area network (WLAN) standard, 802.11i, which requires AES for data encryption. This further increases the potential of AES for future pervasive applications.

In order to handle complicated applications over an extended period of time, electronic textiles need to rely on low-power techniques and exploit redundancy. To this effect, we propose an *e-textile routing framework* (*ERF*) as a means to extend the system lifetime by utilizing redundancy. The focus and main contribution are reflected in the energy-aware routing (*EAR*) portion of this newly proposed framework. More precisely, for this online routing algorithm, a routing controller monitors the residual energy capacity at each processing node and periodically sends the routing commands to each node in order to avoid creating energy hot-spots over an extended period of operation. Other contributions to this framework include topology selection,[1] task mapping, and multiple access network control features.

To validate the ERF in the context of real technology constraints, we study the distributed implementation of AES on e-textiles as follows: First, we partition the AES cipher into several tasks (each performing a unique function) which we fully design and synthesize in Verilog. This helps us use realistic power and performance numbers in the experimental part. A cycle-accurate network simulator, *et_sim*, has been developed as a byproduct of this

• *The authors are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: {jckao, radum}@cmu.edu.*

Manuscript received 28 June 2005; revised 29 Nov. 2005; accepted 26 Jan. 2006; published online 21 Apr. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0215-0605.

1. Mesh topology is discussed in this paper because of the availability of the weaving technology of embedding copper wires and circuits into garments in a crisscross pattern (see [18]).

work for determining the power and performance figures of e-textile platforms. The electrical characteristics of the dedicated components extracted from our Verilog modules, plus the data from [6] and [10], are fed into *et_sim* and used to compare the *EAR* results against the non-energy-aware routing counterpart.

From a theoretical standpoint, our contributions are twofold. The first contribution is the formulation of the *ERF* problem for e-textile applications and the online energy-aware routing algorithm (*EAR*) which can be used to solve it. The second contribution is the derivation of an analytical upper bound for the achievable number of completed jobs using arbitrary topologies. According to our simulation results, *EAR* achieves approximately 75 percent of the analytical upper bound and exceeds its non-energy-aware counterpart by a factor between 6 to 21 times. We point out that the methodology and theoretical results presented here can be applied to any e-textile applications, either wearable or nonwearable (e.g., e-textiles hanging on the walls, carpets, etc.).

The remainder of the paper is organized as follows: In Section 2, we summarize related work. We formulate the problem of the e-textile routing framework in Section 3. In Section 4, we present the analytical upper bound for all possible e-textile routing frameworks. Section 5 outlines the architectural issues, the battery model, and detailed platform description. The online energy-aware routing algorithm is discussed in Section 6. We present the experimental results for the AES cipher and compare them with the analytical upper bound in Section 7. In Section 8, we conclude by summarizing our main contribution and suggest future extensions of this work.

2 Related Work

There has been some recent work on embedding electrical components into wearable fabrics. For instance, the authors of [1] and [2] demonstrate the idea of attaching off-the-shelf electrical components to traditional clothing materials and also provide means to weave user interfaces (and even chip packages) directly into fabric during textile manufacturing.

On the other hand, dedicated e-textile devices such as textile transmission lines (e.g., [6]) and batteries (e.g., thin-film batteries [10], [11]) are currently under development. The routing of electrical power and communication through a wearable fabric is addressed in [3]. It provides a detailed account of the physical and electrical components for routing electricity through suspenders made of fabric and embedded conductive strands, as well as a data-link layer protocol on a Controller Area Network (CAN) bus.

A complete apparel with embedded computing elements is described in [19]. The authors describe a survival clothing prototype designed to provide the sensors (e.g., heart rate monitors and thermometers), as well as the communication and position aids. In [26], the authors present a prototype of the acoustic beamforming array, a prototype of the shape-sensing e-textile garment and a simulator, Ptolemy II, which can model the motion of a person wearing the e-textile garment and the effect of fault in e-textile systems. Besides, the "wearable motherboard" project [17] is a substrate that permits the attachment of computation and sensing devices in much the same manner as a conventional PC motherboard. Its proposed use is to monitor vital signs of its wearer and perform some data processing. All components of the above prototypes obtain power from a centralized power source and the user interacts with them through a single user interface.

Several architectures with distributed deployment of batteries are proposed in [5]. Adaptive techniques, such as code migration and remote execution, applied to redundantly deployed nodes have also been discussed to increase the operational lifetime of the e-textile applications. However, these techniques do not apply to ASIC-style e-textile implementations, as is our case, due to the requirement of using reprogrammable devices (e.g., processors and memories) as computational nodes.

In contrast to these previous research efforts, we investigate techniques for energy-aware routing meant to increase the operational lifetime of an e-textile system, as well as the achievable number of completed jobs. These techniques apply to both programmable devices and fixed devices. We note that, although they look similar, the problem of breaking a task down into pieces and assigning them to various nodes on an e-textile platform cannot be simply recast as a problem of minimizing the execution time (and/or energy consumption) in the field of parallel computing (where, typically, a task is partitioned into pieces and each piece is assigned to nodes such that execution time is minimized). The incompatibility between these two problems comes from the fact that the functionality of the devices used in parallel computing is homogeneous, while the ASIC-style e-textile devices are essentially heterogeneous, each performing a fixed, unique function.

In the area of wireless ad-hoc/sensor networks, there has been a considerable amount of work on energy-aware routing algorithms (e.g., [12], [13], [20], [21], [22], [23], [24]) aimed at extending network lifetime. These algorithms can be viewed as different attempts to combine the key elements of two basic (complementary) routing approaches: Minimum Energy (ME) routing, which selects the route with the least total link energy cost, and Max-min routing, which selects the route with the least residual node energy. However, these two approaches and their variations *cannot* be applied to e-textile platforms for two reasons. First, they consider *wireless communication* as the major source of power consumption, while computation cost dominates the power consumption for *wired* e-textile applications. Second, their resource requirements (e.g., memory, processing speed, and power) during searching an optimal/suboptimal path are often too high to be implemented on an e-textile platform. These major differences motivate the need for developing a resource-efficient energy-aware routing algorithm specifically designed for e-textile applications; this is precisely one of the main objectives of this paper.

To this end, we follow the same general distributed deployment scheme of batteries as in [5], but, due to potential link failures, we target a *network* architecture instead of a bus-based architecture. Also, unlike [5], dedicated devices (e.g., ASICs) and reprogrammable devices (e.g., processors) are supported as computational

nodes in our architecture. These differences make our routing framework quite unique.

3 Problem Description

The main objective of this paper is to propose an *e-textile routing framework* (*ERF*) which can increase the number of completed jobs for e-textile applications. This e-textile routing framework consists of four distinct steps, namely:

1. Selecting the appropriate network topology.
2. Mapping from application tasks to network nodes.
3. Defining an online routing algorithm.
4. Defining an appropriate control mechanism.

As such, the *ERF* we propose refers to a complete *methodology* consisting of several design steps. One should also note that, while the network topology and the task mapping are determined during the design stage and remain fixed afterward, the online routing algorithm adapts to the current state of the system (e.g., residual energy) in order to enable lifetime[2] maximization.

Obviously, such a problem space is too big to be completely explored in practice. Therefore, we approach this problem as follows: We first formulate the *ERF* problem (Section 3) and then derive the upper bound for the system lifetime an *arbitrary ERF* can achieve. Afterward, we focus on the online routing step of this framework and propose the energy-aware routing (*EAR*) algorithm (Section 6). Finally, given the resource constraints, we show the performance gains due to the proposed *EAR* algorithm and compare the system lifetime achievable under the *EAR* algorithm against the analytical upper bound (Section 7).

To define the problem formulation for the e-textile routing framework, we first assume that 1) the target application is prepartitioned into several application tasks, which are implemented by customized modules or are available as IP cores in a library, and 2) the target e-textile system is composed of a wired communication network which connects many active and idling nodes. Each application task performs a unique (fixed) function and cooperates with other application tasks to complete the job by exchanging packets of fixed length. An *instance* is an entity of the hardware module which implements exactly one application task; it is possible that each application task has *multiple* instances across the network. Each in-stance resides at a physical location, namely, a node, in the network.

The hardware redundancy is exploited to extend the system lifetime. However, the level of redundancy (in terms of the total number of the tasks' instances) is bounded above by the number of physically available nodes in the network. Let us consider Fig. 1 as an example. There are 16 physically available nodes in the network: 12 of them (shown as solid circles) are instances of three application tasks (shown in white, light gray, and dark gray colors), while four of them (the dotted circles) remain unused. In theory, the existence of unused nodes is possible, but, as we shall discuss in Section 4, using all available nodes helps

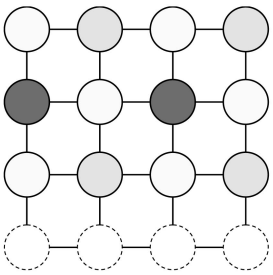

Fig. 1. A 4-by-4 mesh network with 12 instances (solid circles) and four unused nodes (dotted circles).

maximize system lifetime. Therefore, in practice, "node" and "instance" are synonyms in the sense that each instance physically resides at a node and each node has exactly one instance residing at it.

It is assumed that each node has its own attached battery and, for simplicity, all batteries have the same initial capacity. In this paper, a fiber-shaped thin-film battery [10] is used in the simulation setup. However, the theoretical results hold also for any other battery that does not get recharged.[3] A node is considered dead when its attached battery is completely depleted, while the target system dies when a node which is running an operation (but hasn't finished the operation yet) becomes dead.

An *operation* is defined as the act of computation of any application task and the subsequent act of communication (e.g., buffering and switching/forwarding) until the originated packet arrives at the next node which can be either an intermediate node or the final destination node. Typically, a *job* is completed after several operations. For instance, the encryption of a 16-byte block using 128-bit AES algorithm (shown in Fig. 2) takes 10 operations of task *SubBytes/ShiftRows*, 9 operations of task *MixColumns*, and 11 operations of task *AddRoundKey*.

In general, the hardware modules which implement the application tasks can be either fixed or reprogrammable, but the remapping techniques (e.g., code migration and remote execution [5]) on reprogrammable modules are not considered in this paper. Instead, it is assumed that, once decided, the application mapping onto architecture remains fixed. This avoids not only the difficult problem of predicting the overhead caused by the remapping techniques (which is highly dependent on architecture and application itself), but also the requirement of using reprogrammable components.

For simplicity of exposition, in the remainder of this paper, the attributes of a task are referred to as the corresponding attributes of the hardware module implementing that task. For example, "*task 1 consumes 120.1pJ*" means that the corresponding hardware module consumes 120.1pJ per act of computation.

Given a network topology and a fixed mapping from application tasks to the network nodes, the routing algorithm addresses the issue of routing each network packet from the source(s) to the destination(s). One routing option is to route the packets along the shortest physical distance, ignoring other factors such as the residual battery

2. Throughout the remainder of this paper, the term "lifetime" refers to the number of completed jobs.

3. Energy scavenging is beyond the scope of this paper.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey(state, w[0, Nb-1])
    for round = 1 step 1 to Nr–1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
    out = state
end
```

Fig. 2. The pseudocode of the AES cipher.

energy. Another option is to broadcast the packets over all links and then flood the network. In general, a routing algorithm implies making decisions at each routing node as to *where* to forward a packet and *when* to do so.

Due to the limited processing, memory, and energy resources, an efficient online routing algorithm for e-textile platforms needs to have two features. The first feature is the ability to exploit redundancy for lifetime maximization. Indeed, such an online routing algorithm needs to decide at runtime where to forward a packet without any prior knowledge of where exactly the packet destination is.[4] The second feature is the requirement of relying on moderate resources. In this work, we propose the energy-aware routing (*EAR*) algorithm which takes the routing decisions at runtime based on the remaining battery capacity, as well as the available functionality at different nodes. The detailed description of our routing algorithm is given in Section 6.

To present the formulation of the e-textile routing framework, we summarize the basic parameters in Table 1.

As we can see, some parameters in Table 1 depend only on application and architecture. Take 128-bit AES as an example. An encryption job (shown in Fig. 2) involves 10 operations of task 1 (*Subbytes/Shiftrows*), 9 operations of task 2 (*Mixcolumns*), and 11 operations of task 3 (*Addroundkey*); therefore f_1, f_2, and f_3 are 10, 9, and 11, respectively. Other parameters also depend on the routing algorithm and the control mechanism. For instance, the induced overhead, OH_j, and the total energy consumption during communication, C_j, are such examples.

With these notations available, the problem of determining the e-textile routing framework for lifetime maximization (defined as the number of completed jobs), under resource constraints, can be formulated as follows:

Given:

Data flow of the target application, specifically p and $f_i, \forall i$.
Tasks energy consumption per act of computation, $E_i, \forall i$.
The battery budget, B, and the node budget, K.

4. This is because each task has multiple instances across the network and the packet destination can be any of these instances.

TABLE 1
Parameter Notation

Parameter	Description
ERF	E-textile routing framework which consists of network topology selection, task mapping technique, control mechanism and online routing algorithm
$J^{(ERF)}$	The number of jobs completed under a given *ERF*
B	Initial capacity of each battery (that is, the available battery budget)
K	The number of (physically) available nodes (that is, the node budget)
P	The number of (distinct) application tasks
f_i	The number of operations that application task i has to run before a job is completed, where $1 \le i \le p$
E_i	The energy consumed by application task i per act of computation, where $1 \le i \le p$
S_i	The set of nodes which are instances of application task i, where $1 \le i \le p$
n_i	The number of instances of application task i in the target topology (*i.e.* n_i nodes are mapped to application task i, where $1 \le i \le p$)
c_i	Energy consumption per act of communication originated from application task i, where $1 \le i \le p$
C_j	Total energy consumption at node j during communication (either transmitting the packet originated from node j or relaying packets originated from other nodes) before the target system dies, where $1 \le j \le K$
OH_j	The induced overhead in terms of energy consumption by the routing framework *ERF* at node j,where $1 \le j \le K$
ε_i	The normalized energy consumption of application task i, which is defined as the total energy consumed at the instances of application task i by a job completion

Determine:

The optimal e-textile routing framework which maximizes the number of jobs completed over *all possible* e-textile routing frameworks, that is,

$$\max_{ERF}(J^{(ERF)})$$

such that

$$1) \quad \sum_{i=1}^{p} n_i \le K$$

$$2) \quad \sum_{j \in S_i} x_j / f_i - J^{(ERF)} \ge 0, \ \forall i = 1, 2, \ldots, p$$

$$3) \quad E_{i(j)} x_j + C_j + OH_j \le B, \ \forall j = 1, 2, \ldots, K,$$

where x_j is the actual number of operations at node j before the target system dies and the subscript $i(j)$ denotes the type of application task that node j implements.

The first condition in the above formulation restricts the total number of instances of application tasks up to the number of available nodes because a node is not allowed to implement more than one application task. The second condition ensures that $J^{(ERF)}$ jobs are completed under the e-textile routing framework *ERF*. The last condition guarantees that the target system remains alive before $J^{(ERF)}$ jobs are completed. Among all possible e-textile routing frameworks, the one which satisfies all these conditions and maximizes the number of completed jobs represents the optimal solution we are looking for.

4 Analytical Results

The objective of this section is to derive the upper bound for the number of completed jobs which an arbitrary e-textile routing framework can achieve, given a set of resource constraints. To this end, we first define the *ideal* e-textile routing framework (ERF^*), which has the following four features:

1. The topology of ERF^* is chosen to perfectly match the data flow of the target application.
2. The mapping from application tasks to the network nodes is considered optimal. Specifically, for each i, the number of instances of application task i (n_i) is optimal. Furthermore, for each i, we expand the domain of n_i from positive integers to positive real numbers for the sake of mathematical tractability.
3. An incomplete operation due to node battery depletion can continue the remaining fraction of operation at a living node that performs the same functionality without incurring any cost.
4. The overhead of the ideal routing framework caused by the control mechanism is negligible.

To give a little intuition, feature 1 above implies that ERF^* consumes the least amount of energy during communication. Feature 2 ensures that an application task with higher power consumption has more instances to distribute that task's load. Feature 3 implies that the target system under ERF^* is alive as long as there exists (at least) a running instance for each application task. Taken together, all four features imply that no routing framework outperforms ERF^* in terms of maximizing system lifetime.

We note that features 1, 3, and 4 reduce the complex *ERF* problem to the problem of optimizing the number of instances for each application task. To complete a job, the application task i has to perform f_i operations, each consuming $E_i + c_i$, while doing computation and communication, so the energy consumption sums up to the normalized energy consumption, $\varepsilon_i = f_i(E_i + c_i)$. The target system under the e-textile routing framework ERF^* dies when all instances of some application task run out of their batteries. Therefore, the number of completed jobs under ERF^*, assuming the number of instances of task i is n_i, where $1 \le i \le p$, is $\lfloor \min(n_1 B/\varepsilon_1, n_2 B/\varepsilon_2, \ldots, n_p B/\varepsilon_p) \rfloor$. Since no other e-textile routing framework outperforms ERF^*, the achievable number $J^{(ERF)}$ of completed jobs under an arbitrary e-textile routing framework *ERF* is bounded above by

$$\begin{aligned} J^{(ERF)} &\le J^{(ERF^*)} \\ &= \left\lfloor \max_{\underline{n} \in (R^+)^p : \sum_{i=1}^{p} n_i \le K} \min\left(\frac{n_1 B}{\varepsilon_1}, \frac{n_2 B}{\varepsilon_2}, \cdots, \frac{n_p B}{\varepsilon_p}\right) \right\rfloor \qquad (1) \\ &\le \max_{\underline{n} \in (R^+)^p : \sum_{i=1}^{p} n_i \le K} \min\left(\frac{n_1 B}{\varepsilon_1}, \frac{n_2 B}{\varepsilon_2}, \cdots, \frac{n_p B}{\varepsilon_p}\right), \end{aligned}$$

where $\underline{n}$ is a vector[5] defined as $(n_1, n_2, \ldots, n_p)$, R^+ denotes the positive real numbers, and $(R^+)^p$ denotes the set of vectors consisting of p ordered positive real numbers.

5. For simplicity of exposition, the ordered sequence of positive variables $(n_1, n_2, \ldots, n_p)$ is denoted by $\underline{n}$. Similarly, $\underline{x}$ and $\underline{y}$ are defined in the same way.

Now, we are ready to derive the upper bound but need to first give a few helpful results. The following lemma implies that, given a set of nodes, using all of them gives a better or equal performance compared to just using a subset of them.

Lemma 1. *For any positive constant a_i and any positive variable x_i, where $i = 1, 2, \ldots, p$, the following equality:*

$$\max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i \le K} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p) = \max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i = K} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p)$$

always holds.

Proof. Denote the domain $\{\underline{x} \in (R^+)^p : \Sigma_i x_i = K\}$ by γ and the domain $\{\underline{x} \in (R^+)^p : \Sigma_i x_i \le K\}$ by Γ. The following inequality holds:

$$\max_{\underline{x} \in \Gamma} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p) \ge \max_{\underline{x} \in \gamma} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p)$$

because the domain γ is a subset of the domain Γ.

Consider the mapping from Γ to γ. For any element $\underline{x} \in \Gamma$, the corresponding element in γ, $\underline{x}' = (x_1', x_2', , \ldots, x_p')$, is defined as

$$(x_1, x_2, \ldots, x_{p-1}, K - x_1 - x_2 - \ldots - x_{p-1}).$$

Since $x_i \le x_i'$ and, hence, $a_i x_i \le a_i x_i'$ for all i, we get

$$\max_{\underline{x} \in \Gamma} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p) \le \max_{\underline{x}' \in \gamma} \min(a_1 x_1', a_2 x_2', \cdots, a_p x_p').$$

Therefore, we have proven that the equality

$$\max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i \le K} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p) = \max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i = K} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p)$$

always holds. □

Lemma 2 below gives us a hint on how to solve the max-min problem by providing an upper bound for it.

Lemma 2. *If the constants a_is and variables x_is, where $i = 1, 2, \ldots, p$, are all positive, then*

$$\max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i = K} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p) \le J^*,$$

where

$$J^* = \frac{K}{\sum_{i=1}^{p} 1/a_i}.$$

The equality holds when $x_i = J^*/a_i$, for all i.

Proof. We prove this inequality by contradiction. Assume that

$$\max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i = K} \min(a_1 x_1, a_2 x_2, \cdots, a_p x_p) > J^*.$$

By subtracting J^* from both sides of the above inequality, we get:

$$\max_{\underline{x} \in (R^+)^p : \sum_{i=1}^{p} x_i = K} \min(a_1 x_1 - J^*, \cdots, a_p x_p - J^*) > 0. \quad (2)$$

Now, define $y_i = x_i - J^*/a_i$, for $1 \le i \le p$, and use the $\underline{y}$ notation (defined in footnote 5). We know that y_is sum up to zero since

$$\sum_{i=1}^{p} y_i = \sum_{i=1}^{p} x_i - J^* \sum_{i=1}^{p} \frac{1}{a_i} = \sum_{i=1}^{p} x_i - K = 0.$$

So, (2) is equivalent to

$$\max_{\underline{y} : \sum_{i=1}^{p} y_i = 0} \min(a_1 y_1, a_2 y_2, \cdots, a_p y_p) > 0.$$

Since a_is are all positive, the above inequality implies there exists (at least) one $\underline{y}$ such that all of its components y_is are positive. However, this is impossible because y_is sum up to zero. Therefore, the initial assumption is false. Note that, by inspection, the equality in Lemma 2 holds when $x_i = J^*/a_i$, for all i. So we have proven Lemma 2.□

Now, we are able to derive an upper bound for system lifetime, in terms of the number of completed jobs, over all possible e-textile routing frameworks.

Theorem 1 (Upper bound for the achievable number of completed jobs). *Given the application parameters p and f_is, the energy consumption for each application task E_i, the battery budget B, and the total number of available nodes K, the maximum number of completed jobs is given by*

$$J^* = \frac{K\,B}{\sum_{i=1}^{p} \varepsilon_i},$$

where ε_i is the normalized energy consumption of application task i. Furthermore, the optimal number of instances of application task i is given by

$$n_i^* = \frac{K\,\varepsilon_i}{\sum_{j=1}^{p} \varepsilon_j}$$

for $1 \le i \le p$.

Proof. From (1), we have the following upper bound for the number of completed jobs under an arbitrary e-textile routing framework *ERF*:

$$J^{(ERF)} \le \max_{\underline{n} \in (R^+)^p : \sum_{i=1}^{p} n_i \le K} \min\left(\frac{n_1 B}{\varepsilon_1}, \frac{n_2 B}{\varepsilon_2}, \cdots, \frac{n_p B}{\varepsilon_p}\right) = \max_{\underline{n} \in (R^+)^p : \sum_{i=1}^{p} n_i = K} \min\left(\frac{n_1 B}{\varepsilon_1}, \frac{n_2 B}{\varepsilon_2}, \cdots, \frac{n_p B}{\varepsilon_p}\right),$$

where the last equality follows from Lemma 1. Using Lemma 2 and substituting a_i with B/ε_i and x_i with n_i for each i, it is easy to derive the upper bound

$$\max_{\underline{n} \in (R^+)^p : \sum_{i=1}^{p} n_i \le K} \min\left(\frac{n_1 B}{\varepsilon_1}, \frac{n_2 B}{\varepsilon_2}, \cdots, \frac{n_p B}{\varepsilon_p}\right) = \frac{K\,B}{\sum_{i=1}^{p} \varepsilon_i} = J^*$$

and show that n_i^* is the optimal number of instances for application task i, where $i = 1, 2, \ldots, p$. □

Theorem 1 not only gives us a tight upper bound for the achievable number of completed jobs, but also reveals an important design rule: For each i, the optimal number n_i^* of instances of application task i is proportional to the corresponding value of the normalized energy consumption, ε_i. We will come back to this design issue later in Section 5. For now, let us see the details of the platform we consider.

5 Platform Description

Although our proposed *e-textile routing framework* applies to *any* e-textile application, successful research requires validating the methodology in the context of real constraints and application requirements. Hence, we focus on the AES cipher as an illustrative example of our work. For a fair comparison, the proposed energy-aware routing framework and its non-energy-aware counterpart are kept identical except for their routing algorithms, which are the energy-aware routing algorithm (*EAR*) and the shortest-distance routing algorithm (*SDR*), respectively.

We first address the battery modeling issue for the thin-film batteries. We also present the energy models of the computational hardware modules and the transmission lines for the AES cipher implemented on e-textiles. Using measured data, we find out that, for the AES cipher, the power consumed on the transmission lines is *not* negligible compared with the power consumed in the computational modules. This suggests that, for e-textiles, the remaining battery capacity, as well as the distances of the routing paths should be considered when taking the routing decisions.

5.1 AES Partitioning and Computation Energy Consumption

We outline the pseudocode of the AES cipher in Fig. 2. For the AES version with 128-bit key, $Nb = 4$ and $Nr = 10$. Due to the limited capabilities for raw processing and limited battery capacity, the entire system has to be partitioned into several tasks and, obviously, none of these tasks should consume a large amount of power. Finally, the following partitioning scheme is used:

- Task 1: $SubBytes(\,)/ShiftRows(\,)$.
- Task 2: $MixColumns(\,)$.
- Task 3: KeyExpansion/AddRoundKey( ).

We specified all these tasks in Verilog and synthesized them with the Synopsys Design Compiler using a 0.16μm technology library. While these tasks[6] can operate at clock frequencies up to 233MHz, the power consumption measured at 100MHz is used a reference. The energy

6. For simplicity of exposition, the property of the hardware implmentation of some application task is referred to as the property of that task.

TABLE 2
The Energy Consumption per Act of Computation for Different Tasks Measured Directly on Verilog Implementations

	Energy consumption per computation
Task 1	120.1 pJ
Task 2	73.34 pJ
Task 3	175.55 pJ

Tasks 1, 2, and 3 perform SubBytes/ShiftRows, MixColumns, and KeyExpansion/AddRoundKey, respectively.

consumption values, per act of computation, are shown in Table 2.

5.2 Electrical Characteristics of Transmission Lines and Communication Energy Consumption

The electrical characteristics of dedicated e-textile transmission lines of various lengths are taken from [6]. The conductive fabrics contain polyester yarns that are twisted with one copper thread of 40μm diameter and are insulated with a polyesterimide coating. Our SPICE simulations help estimate the energy consumption values per bit-switching activity (Table 3). These values, multiplied by the packet size, represent the energy required to transmit a packet over these transmission lines and, consequently, are fed into our network simulator, *et_sim*.

5.3 Battery Modeling

E-textile systems have to be light, flexible, and slim so that they can be easily embedded into the fabric. Due to these reasons, the novel thin-film batteries [10], [11] are a good candidate to be used for e-textile platforms. Fig. 3 shows the discharge voltage profile of a Li-free thin-film battery [10]. The discharging characteristic, together with the discrete-time model in [8], is implemented in our network simulator, *et_sim*.

The reasons for adopting such a battery modeling approach are twofold. The first reason is the need for speeding up the simulation process. Indeed, instead of building a slow, mixed-mode simulator and accurately simulating the continuous-time discharging characteristics of the thin-film battery and the discrete-time digital circuit, we adopt the discrete-time battery modeling, where a simulation cycle $\Delta t = \tau/5$ is sufficient to model the first and second-order effects and the transient behavior of the battery. This facilitates the implementation of a fast, discrete-time network simulator. The second reason for discrete battery modeling is determined by the levels of accuracy (typically within 15 percent [8]) acceptable in this kind of simulation. Since the actual capacity of any group of

TABLE 3
Energy Consumption Values of Dedicated Transmission Lines

	Length of transmission lines			
	1cm	10cm	20cm	100cm
Energy per bit-switching	0.4472pJ	4.4472pJ	11.867pJ	53.082pJ

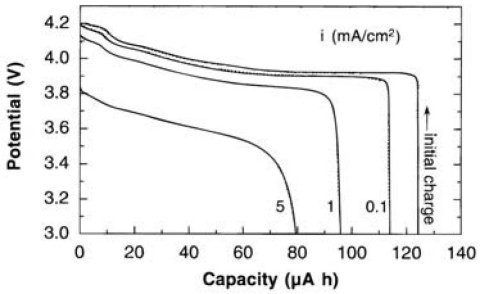

Fig. 3. Discharge curve of thin-film batteries (from [10]).

cells may vary by as much as 20 percent (even between identical units), a level of 15 percent accuracy is very reasonable.

To reduce the simulation time, the initial (nominal) capacity of the thin-film battery is limited to $B = 60\text{nJ}$. We also assume that the corresponding discharging profile shrinks proportionally in the horizontal direction and a node is dead after the output voltage of its attached battery drops below 3.0 Volts. This conservative threshold avoids getting incorrect execution data, which may be produced due to a significant drop in the supply voltage.

5.4 Mapping the AES Application Tasks onto the Mesh Architecture

As Theorem 1 shows, the normalized energy consumption per task (ε_i) is a key factor for mapping the application tasks to the network nodes. Based on Theorem 1, task 3, which consumes the highest normalized energy among all three application tasks, has a large number of instances, so these instances can share the heavy workload.

Consider any node with coordinates (x, y) where x and y are positive integers. We map task 1 to the (x, y) node if $m(x) + m(y) = 2$ or task 2 if $m(x) + m(y) = 0$ or task 3 if $m(x) + m(y) = 1$, where $m(x)$ is defined as $x \; modulo \; 2$. This task mapping process is mostly a trial-and-error process, as illustrated in Fig. 4. In this representation, Fig. 4a shows a smart shirt with several blocks connected through a wired network, while Fig. 4b zooms in to the region where the AES tasks are mapped using a 4×4 mesh network.

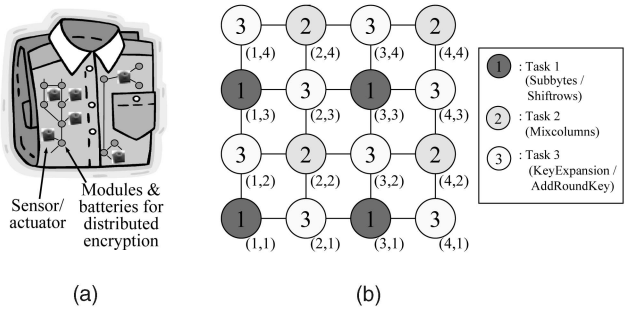

Fig. 4. (a) Network-based e-textile architecture. (b) The mapping of the AES using a 4-by-4 mesh network with a few redundant instances for each application task. Here, the number n inside a circle indicates that node is a hardware module (i.e., an instance) performing application task n.

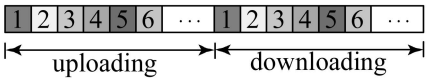

Fig. 5. The TDMA scheme used to implement the control mechanism.

5.5 The Control Mechanism

A control mechanism for energy-aware routing is necessary to allow exchanging the control information (namely, the battery remaining capacity and the routing information) among the network nodes. For e-textile applications, the energy efficiency of the control mechanism is a must due to the limited battery resources. At the same time, due to limited storage resources, the routing table stored at each node in the network has to be as small as possible.

Each node may or may not decide the routing path on its own. Considering the energy efficiency issue, a centralized control mechanism is preferred because of its simplicity and energy efficiency in wired networks. Indeed, the potential overhead associated to gossiping the control information and also the complicated router design make a distributed control mechanism highly unsuited for e-textile systems.

In our implementation of the AES cipher, a time division multiple access (*TDMA*) scheme is used to schedule the control information because, under this policy, the AES nodes can go to idle or sleeping states (and then save energy) most of the time. In addition, we designed a central controller that makes the routing decisions for all the AES nodes and distributes them through a shared communication medium.

The centralized control mechanism based on the TDMA scheme (see Fig. 5) consists of several active and idle centralized controllers, several AES nodes, and a shared communication medium. During their own upload slots, the AES nodes report their status to the controllers. Based on the reported information, the active controller decides the routing paths for all AES nodes and then the information about the next hops is sent to the corresponding AES nodes through the shared medium during the upcoming download phase.

As such, the shared communication medium is used only for exchanging the control information. Under the TDMA scheme, the width of the shared medium is typically very narrow (e.g., 2 bits wide as a typical value). In addition, compared to the significant performance gains (see Section 7.1), the percentage of energy consumed on the shared medium is small (ranging from 2.8 percent to 11.6 percent in our experiments). Therefore, backing up the information sent over the shared medium, in case of failures, is feasible even under the strict resource constraints that characterize e-textile platforms.

As we have seen in Section 5.2, a shorter shared medium consumes less energy while exchanging the control information. In our simulator, *et_sim*, the length of the shared medium is set to the length of the shortest curve crossing over all nodes in a mesh network. More precisely, assuming that all nodes in an *r*-by-*c* mesh network are 1cm away from their nearest neighboring nodes, the length of the shared medium is $(rc-1)$ cm. Given the length of the shared medium, the energy per bit consumed on transmitting the control information can be linearly interpolated using data in Table 3.

To make our solution complete, we also provide a deadlock recovery mechanism for the TDMA scheme: When a job stays at a node for more than a preset threshold period, that particular node reports the occurrence of deadlock during its next upload slot. The controller then sends the new routing instruction to that node to redirect the job along an unlocked path during the next download phase.

In our design of the TDMA scheme, assuming that a node has up to N_P output ports and the remaining battery capacity is partitioned into N_B levels, a time frame takes $K(\lceil \log_2 N_B \rceil + \lceil \log_2(p+1) \rceil)$ bits for the upload phase and $Kp(\lceil \log_2 N_P \rceil)$ bits for the download phase. The number of bits transmitted (per time frame) over the shared medium is linear with the total number of the AES nodes in the network K, so the TDMA scheme is scalable.

In terms of occupied area, the routing tables at nodes are very small: Each AES node has p entries in its routing table and each entry has the length of $\lceil \log_2 N_P \rceil$ bits. To simplify the hardware implementation, only one node has the read/write access to the shared medium in each cycle. For the AES cipher in a mesh network (i.e., $N_P = 4$) with K nodes and an n-bit shared medium, a full time frame takes $K(\lceil 4/n \rceil + \lceil 6/n \rceil)$ cycles. In our implementation, the size of the routing table at each node is 6-bit.

6 Comparison between the EAR and SDR Algorithms

The *EAR* algorithm and its shortest distance routing counterpart (*SDR*) have been developed to decide the routing paths *online*, based on the monitored information about the system. Both *EAR* and *SDR* have the capability of recovery from deadlock.

When the currently reported system information differs from the previous one, the controller executes the routing algorithm in order to instruct the nodes on how to modify their routing tables. *SDR* generally selects the shortest path, while *EAR* determines the path based on the reported battery information. Both *SDR* and *EAR* consist of three phases that are detailed next.

In the first phase, for both *SDR* and *EAR*, a *weight* is assigned to each directed interconnect. The network topology is represented as a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of edges.

For convenience, our algorithms use an adjacency-matrix representation. The edge weight matrix, $W^{(SDR)} = [W_{ij}^{(SDR)}]$, of *SDR* is set to

$$W_{ij}^{(SDR)} = \begin{cases} 0 & \text{if } i = j \\ L_{ij} & \text{if } i \neq j \text{ and } (i,j) \in E \\ \infty & \text{if } i \neq j \text{ and } (i,j) \notin E, \end{cases}$$

where L_{ij} is the length of the directed interconnect. On the other hand, the edge weight matrix $W^{(EAR)} = [W_{ij}^{(EAR)}]$ of *EAR* is set to

```
1  D^(0) = W
2  for n = 1 to K
3     for i = 1 to K
4        for j = 1 to K
5           D_ij^(n) = min(D_ij^(n-1), D_in^(n-1) + D_nj^(n-1))
6           S_ij^(n) = { S_ij^(n-1)   if D_ij^(n-1) ≤ D_in^(n-1) + D_nj^(n-1)
                       { S_in^(n-1)   if D_ij^(n-1) > D_in^(n-1) + D_nj^(n-1)
7  return D^(K) and S^(K)
```

Fig. 6. The second phase of the *EAR* and *SDR* algorithms.

$$W_{ij}^{(EAR)} = \begin{cases} 0 & if\ i = j \\ f(N_B(j))\ L_{ij} & if\ i \neq j\ and\ (i,j) \in E \\ \infty & if\ i \neq j\ and\ (i,j) \notin E, \end{cases}$$

where $N_B(j) \in Z^+$ is the reported battery level of node j, $0 \leq N_B(j) < N_B$, and $f(\cdot)$ is a weighting function. In our experiments, $f(n)$ is set to $2^{Q(N_B-1-n)}$, where $Q > 0$ is a constant used to strengthen the impact of the remaining battery capacity.

During the second phase, for all pairs of nodes, we compute the *K*-by-*K* shortest-path matrix $D = [D_{ij}]$ and the *K*-by-*K* successor matrix $S = [S_{ij}]$, for $1 \leq i, j \leq K$, where K denotes the budget of nodes, D_{ij} is the shortest distance from node i to node j, and S_{ij} is the successor of node i along the shortest path to node j. A variation of the Floyd-Warshall algorithm of complexity $O(n^3)$ (see [9]) is used to compute all-pairs shortest paths and their successors; the pseudocode is given in Fig. 6. In line 1 of the pseudocode in Fig. 6, the initial edge weight matrix W is set to $W^{(SDR)}$ for *SDR* and is set to $W^{(EAR)}$ for *EAR*.

The third phase has to determine the destination node (among several candidate nodes with the same functionality) and avoid using the ports which are currently in a deadlock state. The pseudocode of this phase is shown in Fig. 7. Note that the recovery from deadlock is ensured due to the *if-branch* in line 5 of the pseudocode.

The third phase yields a running time of $O(n^2)$. This is because the total number of executions of line 5 is $(n_1 + n_2 + \ldots + n_p) = K^2$. The hidden constant of the third phase is close to zero when few deadlock and/or congestion situations occur (lines 5-8 in Fig. 7). Even in case of severe deadlock/congestion, the hidden constant in the third phase is at most equal to that in the second phase. From the above arguments, we conclude that, for either *EAR* or *SDR*, the complexity is $O(n^3)$, the hidden constants are small, and most of the running time is spent in the second phase. Thus, *EAR* and *SDR* are practical algorithms for routing on graphs consisting of tens to a few hundred nodes.

7 Experimental Results

As justified in Section 3, due to its complexity, it is impossible to fully explore the design space of the e-textile routing framework. Therefore, given the network topology, mapping technique, and control mechanism (as in Section 5), this section focuses on the performance gains of the online energy-aware routing algorithm, *EAR*.

```
RT_j(i) ≡ the successor of node j on a shortest path to
          S_i in the routing table at node j
1  for n = 1 to K
2     for i = 1 to p
3        dist = ∞
4        for j ∈ S_i
5           if node n is not in deadlock or S_nj ≠ RT_n(i)
6              if dist > D_nj
7                 suc = S_nj
8                 dist = D_nj
9  RT_n(i) = suc
```

Fig. 7. The third phase of the *EAR* and *SDR* algorithms.

To evaluate the performance gains that can be achieved under *EAR*, we simulate the AES cipher using mesh networks of various sizes. The results are compared with those obtained by using its non-energy-aware counterpart, *SDR*. The experimental results are also compared with the analytical upper bound derived in Section 4 to illustrate how much room for further improvement still exists. In addition, multiple concurrent jobs are fed into the target system to see the effectiveness of the developed deadlock recovery mechanism.

Throughout Sections 7.1 to 7.3, it is assumed that a single controller with infinite energy resource is deployed and, therefore, the system dies only when some critical node dies. In order to study the impact of controller failures on the system lifetime, this single-controller architecture with infinite-energy will be replaced in Section 7.4 by the *finite-energy multiple-controller* architecture. In the multiple-controller case, each controller cooperates with other controllers by using remote execution [25]. When the remaining battery capacity of the active controller is below a threshold, it requests remote execution and another alive controller, if one exists, takes over its role.

The cycle-accurate network simulator, *et_sim*, supports, in the default mode, any 2D mesh network with the task mapping technique described in Section 5.4. Since many factors (e.g., energy consumed on computational components and transmission lines, the discharging characteristics of batteries, etc.) have a significant impact on the performance, *et_sim* models them accurately using their actual values (see Section 5 for details).

7.1 EAR versus SDR

In the first set of experiments, a new job is launched when the previous one is completed. In other words, there is exactly one job running in the target system and, consequently, no buffering at nodes is needed. As discussed in Section 5.3, the battery model is based on the discharging characteristics of a thin-film battery together with a discrete-time approximation. The performance data is collected when the target system becomes dead.

The experimental results of performance gain are shown in Fig. 8. It is observed tha, *EAR* does better than *SDR* by a factor between 6 to 21 times, depending on the size of the network.

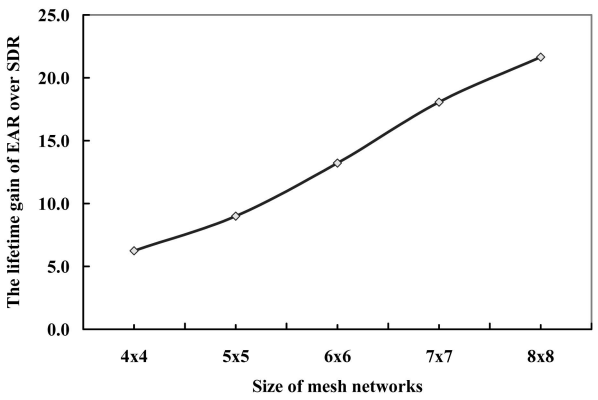

Fig. 8. The system lifetime (in terms of the number of completed jobs) under *EAR* is shown comparatively against its non-energy-aware counterpart, *SDR*. The x axis represents the size of the mesh network, while the y axis shows the ration between the lifetime under *EAR* and the lifetime under *SDR*.

In addition, we also tested two low-overhead configurations where the remaining battery capacity is partitioned into four levels, i.e., $N_B = 4$. The performance comparison between *EAR* and *SDR* in this configuration is shown in Fig. 9. Compared with the significant performance gains ranging from 5 to 15 times, the overhead caused by the control mechanism in the two configurations is indeed very small: As shown in Fig. 10, the percentage of energy consumed on exchanging the control information divided by the total energy consumption in 4×4, 5×5, 6×6, 7×7, and 8×8 mesh networks is only 2.8 percent, 3.1 percent, 4.1 percent, 9.3 percent, and 11.6 percent, respectively.

It should be noted that, in the experiments for *EAR* with 1-bit wide shared medium in the 7×7 and 8×8 networks, the performance drops to the same level as in *SDR* (see Fig. 9). This is because exchanging the control information through the 1-bit shared medium is too slow to compensate for the actual battery depletion: The durations of the time frames in the TDMA scheme for the 7×7 and 8×8 networks are approximately equal to the lifetime of some critical nodes in the mesh networks. Increasing the width of the shared medium to more than 1 bit makes *EAR* perform better than *SDR* in large networks, which is also obvious from data in Fig. 9. In practice, since the capacity of a fully charged battery is, by several orders of magnitude, larger than the value used in our experiments, the degradation will either not happen at all or will only occur when the total number of nodes is very large.

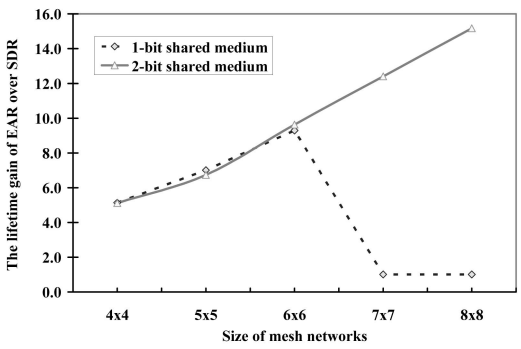

Fig. 9. *EAR* versus *SDR* under two low-overhead configurations.

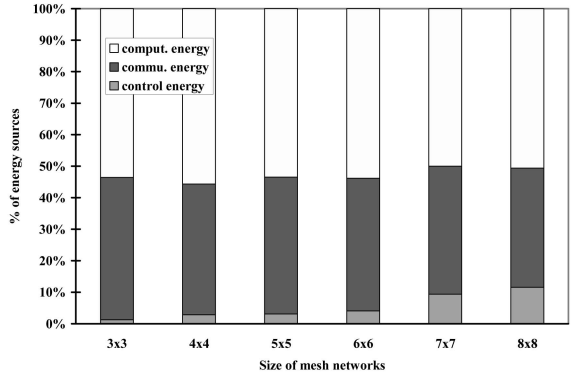

Fig. 10. The percentage of computational energy, communication energy, and energy consumed on exchanging control information (over a 2-bit shared medium) at AES nodes under *EAR*.

7.2 EAR versus Theoretical Upper Bound

In Table 4, the number of jobs completed under *EAR* is compared against the corresponding theoretical upper bound given in Theorem 1. For a fair comparison, the battery model of the Li-free thin-film battery is replaced with the ideal battery model, which outputs constant voltage with 100 percent efficiency until depletion.

The parameters used in calculating the theoretical upper bound are as follows: The packet size is 260-bit with the following breakdown: 128 bits reserved for the data block, 128 bits reserved for the encryption key,[7] and the last 4 bits reserved for the round number.[8] We know that the energy consumption per bit-switching over a 1cm transmission line is 0.4472pJ (see Table 3), so the energy consumption per act of communication, under the ideal e-textile routing framework ERF^*, is $c_i = 0.4472 * 260 = 116.272$ pJ for all i. Using the formula $\varepsilon_i = f_i(E_i + c_i)$, where the values of E_is can be found in Table 1, and substituting f_1, f_2, and f_3 by 10, 9, and 11, respectively, we get the values of normalized energy consumption for the three tasks (described in Section 5.1): $\varepsilon_1 = 2,363.72$ pJ, $\varepsilon_2 = 1,706.508$ pJ, and $\varepsilon_3 = 3,221.042$ pJ. Further, substituting these values and the initial battery capacity $B = 60\text{nJ}$ into Theorem 1, we get the theoretical upper bounds shown in Table 4.

It can be observed in Table 4 that *EAR* achieves 70.0 to 80.1 percent of the maximum achievable number of completed jobs. The gap between *EAR* and the theoretical upper bound is due to 1) the imperfect match between the mesh topology and the application flow, 2) the additional packet transmission generated by going around the nodes with low remaining battery capacity, and 3) the overhead caused by exchanging control information. Therefore, for all practical purposes, it is expected that *EAR* actually performs much better than it appears from the data in Table 4.

7. The field of the encryption key is necessary because the system is designed in such a way that it allows multiple encryption jobs with multiple keys running concurrently.

8. The round number is the value of the *round* variable in Fig. 2.

TABLE 4
EAR versus the Theoretical Upper Bound

# of jobs completed		Simulation results, $J^{(EAR)}$	Theoretical upper bound, J^*	$J^{(EAR)}/J^*$
Network size	4×4	99	131.4	75.2%
	5×5	144	205.2	70.0%
	6×6	223	295.7	75.3%
	7×7	323	402.4	80.1%
	8×8	396	525.7	75.2%

7.3 Latency under Multiple Concurrent Jobs

In this section, we examine the efficiency of the deadlock recovery mechanism. In this set of experiments, multiple jobs are injected into the target system when all previous injected jobs are ejected. In this paper, the numbers of buffer slots available at individual nodes (also called buffer depth) are identical,[9] varying from 1 to 8. Having n slots at some node means that at most n packets can be served, buffered, or relayed at that node. The main objective of this section is to evaluate the efficiency of our deadlock recovery mechanism through latency-throughput bar diagrams.

In Fig. 11, we plot the measured average latency (considered from the moment when a job is injected until the moment when that job is ejected) given the number of buffer slots at each node and the number of concurrent jobs. As we can see, the latency is high when the number of concurrent jobs is large but the buffer depth is small. In that case, congestion and/or deadlock appear very frequently.

The simulation results show that the common phenomenon of phase transition in data networks [14], [15], [16] also happens in the distributed implementation of the AES cipher. Increasing the buffer depth to a certain level drastically decreases the latency in the network, sometimes by several orders of magnitude. Further increasing the buffer depth beyond a certain threshold does not further decrease the latency significantly due to the rare occurrence of congestion and/or deadlock. As one can see in Fig. 11, setting the buffer depth to half of the maximum number of concurrent jobs can prevent the occurrence of severe congestion and/or deadlock.

7.4 Effect of Controller Failures on System Lifetime

In this section, the effect of controller failures due to battery depletion is investigated. We implemented in Verilog several controllers for mesh networks of various sizes and measured their power consumption. For example, the controller of a 4×4 mesh network (operating at 100MHz clock frequency) consumes a dynamic power of 6.94mW and a leakage power of 0.57mW, while a controller designed for a 5×5 mesh network dissipates a dynamic power of 11.4mW and a leakage power of 1.07mW.

In Fig. 12, we present the simulation results for a different number of controllers, each controller having attached a battery as in Section 5.3. The system lifetime is

9. The customization of the buffer sizes at different individual nodes is beyond the scope of this paper and is left as future work.

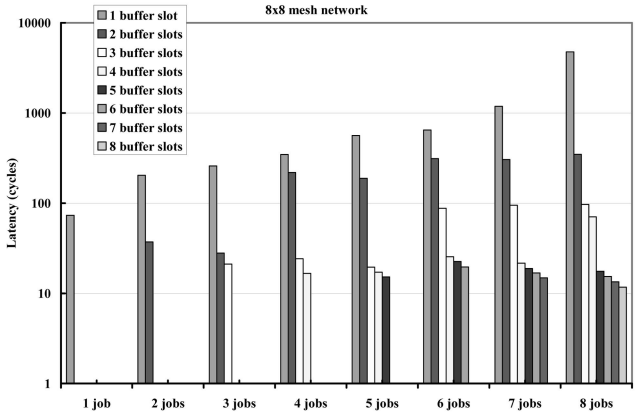

Fig. 11. Latency under multiple concurrent jobs.

determined by either the lifetime of the AES nodes or the lifetime of the controllers, whichever is smaller. For a fixed-size mesh network, increasing the number of controllers extends the system lifetime up to a threshold, beyond which the lifetime of AES nodes becomes the dominant factor for system lifetime. Given a fixed number of available controllers, say l, the head of the curve corresponding to l controllers in Fig. 12 is increasing because there are more and more available AES nodes. On the contrary, the tail of the corresponding curve is decreasing because a controller for a bigger mesh network consumes more power than a controller for a smaller network.

8 Conclusion and Future Work

In this paper, the issue of energy-aware routing in electronic textiles has been addressed. As main theoretical contributions, we have derived an analytical upper bound and developed a general-purpose energy-aware routing algorithm, *EAR*, which considers the concrete limitations of e-textile platforms (e.g., limited battery capacities, long wire effects, etc.). For the AES cipher, *EAR* achieves approximately 75 percent of the analytical upper bound. Furthermore, in a mesh network with less than or equal to 64 nodes, the performance gains of *EAR* over its non-energy-aware counterpart ranges from 6 to 21 times, depending on the

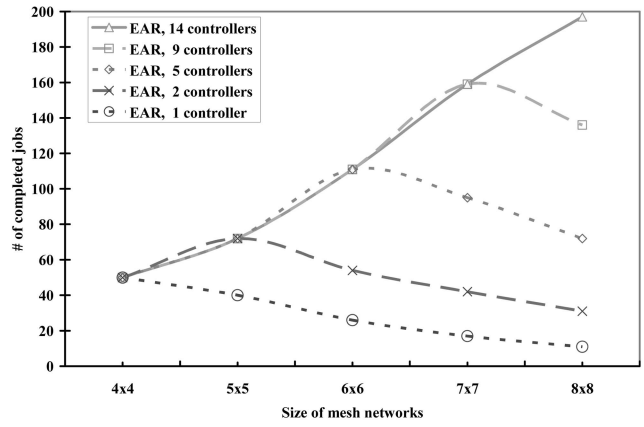

Fig. 12. The effect of the number of controllers on system lifetime in networks of various sizes.

size of the network. It is expected that the performance gain will be even higher in larger networks.

The technology of embedding electronic devices and conductive wires into garments in a crisscross pattern is already available nowadays. This is the reason that the experimental setup in this paper is mesh-based. However, as the technology of weaving in arbitrary patterns becomes available in the near future, significant performance gains may be achieved by using application-specific customized topology, scheduling policy, and buffer allocation. While, in principle, the theoretical results proposed in this paper apply to systems based on any topology, we foresee a need for algorithms optimizing the application-customized e-textile systems in such general cases. The design of such customized algorithms, as well as performance analysis based on the results derived in this paper, are part of our future research investigations.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers and the System-Level Design group of Carnegie Mellon University for many helpful suggestions. This research is supported by CMU CyLab ARO grant no. 9097.60.5.

REFERENCES

[1] E.R. Post and M. Orth, "Smart Fabric, or Wearable Clothing," *Proc. Int'l Symp. Wearable Computers,* pp. 167-168, Oct. 1997.
[2] E.R. Post, M. Orth, P.R. Russo, and N. Gershenfeld, "E-Broidery: Design and Fabrication of Textile-Based Computing," *IBM Systems J.,* vol. 39, nos. 3-4, pp. 840-860, 2000.
[3] M. Gorlick, "Electric Suspenders: A Fabric Power Bus and Data Network for Wearable Digital Devices," *Proc. Int'l Symp. Wearable Computers,* pp. 114-121, Oct. 1999.
[4] D. Marculescu, R. Marculescu, N. Zamora, P. Stanley-Marbell, P.K. Khosla, S. Park, S. Jayaraman, S. Jung, C. Lauterbach, W. Weber, T. Kirstein, D. Cottet, J. Grzyb, G. Troester, M. Jones, T. Martin, and Z. Nakad, "Electronic Textiles: A Platform for Pervasive Computing," *Proc. IEEE,* vol. 91, no. 12, pp. 1995-2018, Dec. 2003.
[5] P. Stanley-Marbell, D. Marculescu, R. Marculescu, and P.K. Khosla, "Modeling, Analysis and Self-Management of Electronic Textiles," *IEEE Trans. Computers,* vol. 52, no. 8, pp. 996-1010, Aug. 2003.
[6] D. Cottet, J. Grzyb, T. Kirstein, and G. Tröster, "Electrical Characterization of Textile Transmission Line," *IEEE Trans. Advanced Packaging,* vol. 26, no. 2, pp. 182-190, May 2003.
[7] FIPS 197, announced by Nat'l Inst. of Standards and Technology, Nov. 2001.
[8] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Discrete-Time Battery Models for System-Level Low-Power Design," *IEEE Trans. VLSI Systems,* vol. 9, no. 5, pp. 630-640, Oct. 2001.
[9] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms,* chapter 26.2, pp. 558-564, first ed. McGraw-Hill, 1990.
[10] B.J. Neudecker, N.J. Dudney, and J.B. Bates, "Lithium-Free Thin-Film Battery with In Situ Plated Li Anode," *J. Electrochemical Soc.,* vol. 147, no. 2, pp. 517-523, 2000.
[11] B.J. Neudecker, M.H. Benson, and B.K. Emerson, "Power Fibers: Thin-Film Batteries on Fiber Substrates," http://www.darpa.mil/dso/thrust/matdev/smfm/Present.html, Nov. 2005.
[12] M. Maleki, K. Dantu, and M. Pedram, "Lifetime Prediction Routing in Mobile Ad Hoc Networks," *Proc. IEEE Wireless Comm. and Networking Conf.,* pp. 1185-1190, Mar. 2003.
[13] J.-H. Chang and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks," *IEEE/ACM Trans. Networking,* vol. 12, no. 4, pp. 609-619, Aug. 2004.
[14] T. Ohira and R. Sawatari, "Phase Transition in a Computer Network Traffic Model," *Physical Rev. E,* vol. 58, no. 1, pp. 193-195, July 1998.
[15] S. Valverde and R.V. Sole, "Self-Organized Critical Traffic in Parallel Computer Networks," *Physica A,* vol. 312, no. 4, pp. 636-648, Sept. 2002.
[16] R.V. Sole and S. Valverde, "Information Transfer and Phase Transitions in a Model of Internet Traffic," *Physica A,* vol. 289, no. 4, pp. 595-605, Jan. 2001.
[17] S. Park, K. Mackenzie, and S. Jayaraman, "The Wearable Motherboard: A Framework for Personalized Mobile Information Processing (PMIP)," *Proc. ACM/IEEE Design Automation Conf.,* pp. 170-174, June 2002.
[18] I. Locher, T. Kirstein, and G. Troester, "Routing Methods Adapted to e-Textiles," *Proc. Int'l Microelectronics and Packaging Soc.,* Nov. 2004.
[19] J. Rantanen, T. Karinsalo, M. Makinen, P. Talvenmaa, M. Tasanen, and J. Vanhala, "Smart Clothing for the Arctic Environment," *Proc. Int'l Symp. Wearable Computers,* pp. 15-23, Oct. 2000.
[20] Q. Li, J.A. Aslam, and D. Rus, "Online Power-Aware Routing in Wireless Ad-Hoc Networks," *Proc. Int'l Conf. Mobile Computing and Networking,* pp. 97-107, July 2001.
[21] V. Rodoplu and T.H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.,* vol. 17, no. 8, pp. 1333-1344, Aug. 1999.
[22] S. Singh, M. Woo, and C.S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks," *Proc. Int'l Conf. Mobile Computing and Networking,* pp. 181-190, Oct. 1998.
[23] J. Wieselthier, G. Nguyen, and A. Ephremides, "Energy Limited Wireless Networking with Directional Antennas: The Case of Session-Based Multicasting," *Proc. IEEE INFOCOM,* pp. 190-199, June 2002.
[24] C.-K. Toh, "Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks," *IEEE Comm. Magazine,* vol. 39, no. 6, pp. 138-147, June 2001.
[25] D.S. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou, "Process Migration," *ACM Computing Surveys,* vol. 32, no. 3, pp. 241-299, Sept. 2000.
[26] T. Martin, M. Jones, J. Edmison, T. Sheikh, and Z. Nakad, "Modeling and Simulating Electronic Textile Applications," *Proc. ACM SIGPLAN/SIGBED Conf. Languages, Compilers, and Tools for Embedded Systems,* pp. 10-19, June 2004.

Jung-Chun Kao received the BS degree from National Taiwan University in 1999 and the MS degree from University of Southern California in 2003, both in electrical engineering. He is currently a PhD student at Carnegie Mellon University, Pittsburgh, Pennsylvania. His research interests include analysis and optimization techniques in networked systems, ambient intelligence, wireless ad hoc networks, and wireless sensor networks. He is a student member of the IEEE.

Radu Marculescu received the PhD degree in electrical engineering from the University of Southern California in 1998. He is currently an associate professor in the Department of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, Pennsylvania. He is a recipient of the US National Science Foundation's CAREER Award (2001) in the area of design automation of electronic systems. He received the 2005 *IEEE Transactions on Very Large Scale Integration Systems* (*TVLSI*) Best Paper Award from the IEEE Circuits and Systems (CAS) Society, two Best Paper Awards from the Design Automation and Test in Europe (DATE) Conference in 2001 and 2003, and a Best Paper Award from the Asia and South Pacific Design Automation Conference (ASP-DAC) in 2003. He was also awarded the Carnegie Institute of Technology's Ladd Research Award in 2002. His current research focuses on developing design methodologies and software tools for SoC design, on-chip communication, and ambient intelligence. He is a member of the IEEE and the ACM.