

Dynamic Data Compression in Multi-hop Wireless Networks*

Abhishek B. Sharma[†], Leana Golubchik^{†*}, Ramesh Govindan[†] and Michael J. Neely^{*}

[†]Computer Science, ^{*}Electrical Engineering
University of Southern California, Los Angeles
{absharma, leana, ramesh, mjneely}@usc.edu

ABSTRACT

Data compression can save energy and increase network capacity in wireless sensor networks. However, the decision of whether and when to compress data can depend upon platform hardware, topology, wireless channel conditions, and application data rates. Using Lyapunov optimization theory, we design an algorithm called SEEC that makes joint compression and transmission decisions with the goal of minimizing energy consumption. A practical distributed variant, DSEEC, is able to achieve more than 30% energy savings and adapts seamlessly across a wide range of conditions, without explicitly taking topology, application data rates, and link quality changes into account.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Design studies, Performance attributes

General Terms

Theory, Algorithms, Performance

Keywords

Stochastic Network Optimization, Energy efficiency

1. INTRODUCTION

Wireless sensor networks, in which nodes collect sensor data and transmit them over multiple hops to a sink, have enabled unprecedented visibility in many natural and built environments. Data compression has been considered for these networks in two contexts. First, compression can reduce transmission costs and thereby save energy resources. Second, for applications that generate significant data, compression can increase the effective network capacity.

*This material is supported in part by: NSF Grant Nos. CNS-0540420 and OCE 0520324, the DARPA IT-MANET program grant W911NF-07-0028, and the NSF Career grant CCF-0747525.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMETRICS/Performance'09, June 15–19, 2009, Seattle, WA, USA.
Copyright 2009 ACM 978-1-60558-511-6/09/06 ...\$5.00.

Until recently, it was assumed that compression was always energy-efficient so that the decision to compress could be made statically. However, Sadler and Martonosi [15] showed that, for many commonly used platforms, whether compressing data at a node is energy-efficient or not depends upon the platform's components, as well as the position of the node in the topology. Specifically, they showed that the balance between the energy required for compression and the energy required for transmission is such that compression wins only if a node is several hops away from the sink (the actual distance depends upon topology). One can extend the arguments in [15] to show that the decision to compress at a node can also depend upon the wireless channel conditions along the path from the node to the sink: a noisy channel, because it consumes energy in retransmissions, can make it favorable to compress at nodes closer to the sink. Moreover, this decision can also depend upon the application data rate — when the network is operating close to capacity, it might be more energy-efficient to compress a fraction of packets, rather than all of them.

In practice, this means that the decision of whether (and when) to compress data for energy-efficient operation must be made *dynamically*. It is possible, of course, to devise a heuristic decision algorithm for this task, that explicitly takes platform energy consumption, topology, channel characteristics and application data rates into consideration.

In this paper, we take a more principled approach. Using tools from Lyapunov optimization theory [8], which explores the design of stable transmission schedulers that optimize a given objective function, we design an algorithm called SEEC (Stable Energy-Efficient joint Compression and transmission scheduling, Section 4). SEEC ensures network stability (queues at nodes are finite) and minimizes time average energy expenditure, while dynamically deciding whether and when to compress data in order to achieve energy-efficiency. It is parsimonious, in the sense that a single parameter V governs the performance of the algorithm.

We are able to provide a performance bound for SEEC. Specifically, we show that SEEC's energy consumption is within an additive factor of the optimal. It can achieve an energy consumption arbitrarily close to the optimal at the expense of increased queuing delay.

SEEC is a centralized algorithm and makes idealized assumptions about wireless medium access (Section 2), so we also explore a more practical distributed variant called DSEEC (Section 5). This variant runs on CSMA-based MACs (such as those used with 802.11 and 802.15.4 radios). It uses the same compression decision algorithm as SEEC (which re-

quires only local information), and uses queue backlog information from its local neighborhood to implement SEEC’s transmission scheduling algorithm.

DSEEC’s chief advantage is that it is able to achieve energy-savings, while adapting dynamically to platform changes, channel variability, diverse application data rates, and topology diversity. We demonstrate (Section 6) its adaptivity through extensive simulations in Qualnet [1] (a widely-used high-fidelity wireless simulator), and show that DSEEC can achieve more than 30% energy savings in some settings. DSEEC’s elegance stems from the fact that it is able to make the compression decisions without *explicitly* considering topology, application data rates, or channel qualities. Instead, these characteristics manifest themselves as changes in queue backlog at a node, which triggers compression decisions. Finally, we show that DSEEC’s performance is relatively insensitive to the choice of its parameter V : changing V by three orders of magnitude affects overall energy consumption by less than 10%.

2. SYSTEM MODEL

In this section we describe our model for a wireless (sensor) network deployed for data collection. This model provides a framework for describing our joint compression and transmission scheduling algorithm and analyzing its performance. For ease of exposition, we assume that time is slotted; our system model and algorithms can be easily generalized to the continuous time case.

Data acquisition. Consider a network of N nodes. Let $\mathcal{N} = \{1, 2, \dots, N\}$ represent the set of nodes. Each node $n \in \mathcal{N}$ has a sensor attached to it, which collects measurements periodically. Let $A_n[t]$ represent the number of measurement samples collected by the sensor attached to node n during timeslot t . We assume that the size of each sample is b bits. In a wide range of sensing applications, data is collected at a fixed sampling frequency. For these applications, $A_n[t] = a$ for all timeslots t . However, if the sensing application adapts the sampling frequency in response to an event [4], then $A_n[t]$ may vary across time slots. We model $A_n[t]$ as a discrete random variable with maximum value m . Each node has to deliver the measurement data collected by its sensor to a sink (possibly) over multiple wireless hops. In the design and analysis of SEEC, we assume a single sink but possible dynamic routing over different outgoing links. However, the evaluation in Section 6 is done for the special case of a fixed routing tree rooted at the sink.

Data Compression. The data collected by a node is first processed by a *compression module*. The compression module can compress the data using one of the K compression algorithms available to it. Every timeslot t , the compression module picks a compression option $k_n[t]$. Option $k_n[t] = 0$ represents no data compression. This is the *central focus of the paper*: the decision about whether to compress, and which compression option to use, is made *jointly* with transmission scheduling (discussed below), taking channel conditions and interference into account. The compressed data is then delivered to the queue at the *transmission module* (discussed below).

For each node n , we model the size of the data after compression, $R_n[t]$, as a random variable. In practice, the output of the compression module will depend on the values of measurement samples being compressed. However, for

analytical tractability, we assume that $R_n[t]$ are i.i.d. conditionally on the value of $A_n[t]$ and $k_n[t]$. As we show in our trace-based evaluations (in Section 6), our results hold even for real datasets, indicating that this assumption is not restrictive in practice.

Let $E_C^n[t]$ denote the energy consumed at node n while compressing data using option $k_n[t]$; with $E_C^n[t] = 0$ for $k_n[t] = 0$. We model $E_C^n[t]$ as a random variable whose value is dependent upon the compression option $k_n[t]$. We assume that $E_C^n[t]$ are i.i.d. for all t with the same compression option $k_n[t]$.

Compression Algorithms. We characterize the different compression algorithms in terms of their expected compression ratios and energy consumption. For every compression algorithm $k \in \mathcal{K}$, we define the expected size of the compressed output $m(a, k)$ and the energy consumption $\phi(a, k)$ for $A_n[t] = a$ as follows:

$$m(a, k) \triangleq \mathbb{E}\{R_n[t] \mid A_n[t] = a, k_n[t] = k\} \quad (1)$$

$$\phi(a, k) \triangleq \mathbb{E}\{E_C^n[t] \mid A_n[t] = a, k_n[t] = k\} \quad (2)$$

Given our definition of $m(a, k)$, the expected compression ratio achieved by option k is $\frac{m(a, k)}{ab}$.

We assume that $m(a, k)$ and $\phi(a, k)$ are known for all compression algorithms $k \in \mathcal{K}$. In practice, these quantities can be empirically determined from a large enough sample dataset prior to network operation. If such quantities are not available prior to network operation they can be estimated on-line during an initial learning period during which data is always compressed to collect statistics. In our evaluations, we use the former approach for simplicity.

Scheduling Transmissions. Let \mathcal{L} denote the set of all wireless links in the network. Each link $l \in \mathcal{L}$ is characterized by a pair of nodes (n_1, n_2) where n_1 transmits data to n_2 over the link l . For each node n , let Ω_n represent the set of all outgoing links on which node n can transmit data and Θ_n represent the set of all incoming links on which node n can receive data.

The transmission rate over a wireless link depends on the transmit power and the channel state. The channel state of a wireless link can be characterized using several different metrics: the Signal to Interference plus Noise Ratio (SINR) for the link, the expected transmission count (ETX) metric [7], the LQI (Link Quality Indicator threshold) [18] and the RSSI (Received Signal Strength Indicator) values, *etc.* In this paper, we use the ETX metric to capture link quality in our evaluation (Section 6). If we model each packet transmission attempt as a Bernoulli trial, then $\text{ETX} = 1/(p_f \times p_r)$, with p_f and p_r equal to the successful transmission probability for a packet and its acknowledgment, respectively.

We represent the current channel state at time t as a vector $\vec{S}[t] = \{S_l[t]\}_{l \in \mathcal{L}}$ where $S_l[t]$ denotes the channel state at time t for link l . For all links $l \in \mathcal{L}$, we assume that $S_l[t]$ takes values from a finite set \tilde{S} for all t . We represent the (finite) set of all possible channel state vectors by $\mathcal{S} = \tilde{S}^{|\mathcal{L}|}$, and hence, $\vec{S}[t] \in \mathcal{S}$ for all t .

A centralized¹ transmission scheduler determines the set of nodes that can transmit at time t based on the current channel state $\vec{S}[t]$. For each link $l \in \mathcal{L}$, it determines a transmit power $P_{tran}^l[t]$. We assume that $P_{tran}^l[t] \in \{0, P_{max}\}$ for

¹In a later section, we discuss a decentralized scheduler whose decisions approximate this centralized scheduler.

all links l and time t (it is possible to extend our framework to multiple transmit power levels). A node n transmits at time t if and only if for at least one of its outgoing links $l \in \Omega_n$ is assigned $P_{tran}^l[t] > 0$. We represent the transmit power allocation by the scheduler at time t as a vector $\vec{P}_{tran}[t]$.

We can incorporate different scheduling constraints into the transmission scheduler. For example, if nodes have a single radio, and they cannot transmit and receive (or transmit on more than one link) simultaneously, we can impose the following *feasibility* constraint on $\vec{P}_{tran}[t]$:

$$\sum_{l \in \Omega_n} I(P_{tran}^l[t]) + \sum_{l \in \Theta_n} I(P_{tran}^l[t]) \leq 1 \quad \forall \text{ nodes } n \quad (3)$$

where $I(P_{tran}^l[t])$ is equal to 1 if $P_{tran}^l[t] > 0$ and zero otherwise.

Transmission capacity-power curve Let $\mu_l[t]$ denote the number of bits that can be transmitted over link l during time slot t . We model $\mu_l[t]$ as a function,

$$\mu_l[t] = C_l(\vec{P}_{tran}[t], \vec{S}[t]), \quad (4)$$

where $\vec{P}_{tran}[t] = \{P_{tran}^l[t]\}_{l \in \mathcal{L}}$, $\vec{S}[t]$ is the current channel state, and $C_l(\vec{P}_{tran}[t], \vec{S}[t])$ is the *capacity-power* curve for link l defined by the modulation and coding schemes used for transmission on link l . We assume that there exists a constant μ_{max} , such that $\mu_l[t] \leq \mu_{max}$ for all t . We also assume that, for all l , $C_l(\vec{P}, \vec{s})$ is piecewise continuous in \vec{P} , and is monotonically increasing in the transmission power for link l , i.e.,

$$C_l((P^l, \{P^j\}_{j \neq l}), \vec{s}) \geq C_l((\tilde{P}^l, \{P^j\}_{j \neq l}), \vec{s}) \quad \text{for } P^l > \tilde{P}^l$$

for each channel state \vec{s} . Additionally, for every link l , $C_l((0, P^j_{j \neq l}), \vec{s}) = 0$ for every channel state \vec{s} , i.e., zero transmission power always yields zero transmission rate. Thus, at time t , $\mu_l[t] > 0$ only for those links l that are allocated $P_{tran}^l[t] > 0$ by the transmission scheduler.

Example of Capacity-Power Curve: For a channel subject to additive white Gaussian noise, the capacity-power curve for link l , $C_l(\vec{P}_{tran}, \vec{s})$ can be approximated using a $\log()$ formula for channel capacity. Hence, we can define $C_l(\vec{P}_{tran}, \vec{s})$ for link l as follows:

$$C_l(\vec{P}, \vec{s}) = \log \left(1 + \frac{\alpha_l P_{tran}^l}{N_l + \sum_{i \neq l} \alpha_i P_{tran}^i} \right)$$

where N_l and α_l represent the Noise and fading coefficients associated with the particular channel state \vec{s} .

Queueing dynamics. The output of the compression module, $R_n[t]$, is held in a queue at the transmission module awaiting transmission. Once data is passed on to the transmission module, it is not considered for compression again.

Let $U_n[t]$ denote the number of bits in queue at node n . The following equation captures the dynamics of the queue backlog:

$$U_n[t+1] \leq \max \left(U_n[t] - \sum_{l \in \Omega_n} \mu_l[t], 0 \right) + R_n[t] + \sum_{l \in \Theta_n} \mu_l[t] \quad (5)$$

where $\mu_l[t]$ is given by equation (4) and $\sum_{l \in \Theta_n} \mu_l[t]$ represents the maximum possible exogenous arrivals at node n

Symbol	Description
$A_n[t]$	Samples collected; ($\leq m$)
b	size of each sample in bits
$k_n[t]$	Compression decision
$R_n[t]$	Data size after compression
$m(\cdot, k)$	Expected size of compressed data for compression algorithm k (eq. 1)
$E_C^n[t]$	Energy consumed by data compression
$\phi(\cdot, k)$	Expected energy consumption for compression algorithm k (eq. 2)
\mathcal{L}	All links in the network
Ω_n	Set of outgoing links
Θ_n	Set of incoming links
$\vec{S}[t]$	Current channel state; $\vec{S}[t] = \{S_l[t]\}_{l \in \mathcal{L}}$
$\vec{P}_{tran}[t]$	Transmit power allocation; $\vec{P}_{tran}[t] = \{P_{tran}^l[t]\}_{l \in \mathcal{L}}$
$\mu_l[t]$	# bits that can be transmitted on link l during slot t (eq. 4)
$U_n[t]$	Queue backlog
$E_T^l[t]$	Energy consumed at node transmitting data on link l
$E_R^l[t]$	Energy consumed at node receiving data on link l
$E_B^l[t]$	Energy consumed due to nodes overhearing the transmission on link l

Table 1: System Model Notation: symbols t and n denote the current time and a node, respectively.

due to other nodes routing their data through node n . The equality in (5) achieved only when the actual endogenous arrivals from other nodes is equal to $\sum_{l \in \Theta_n} \mu_l[t]$; the actual endogenous arrival can be less than $\sum_{l \in \Theta_n} \mu_l[t]$ if the other nodes have little or no queue backlog.

Table 1 summarizes the notation used to describe our model. Our framework can naturally incorporate several extensions to this model, which we discuss briefly as part of future work in Section 8.

3. OPTIMIZATION GOAL

In this paper, our aim is to design a compression and transmission scheduling algorithm that minimizes the total system power expenditure while maintaining network stability. Using our system model (Section 2), we now formally define total system power expenditure and network stability.

The main sources of energy expenditure in our system are data compression, data transmission, and data reception. At time t , the energy consumed by the compression module at node n is $E_C^n[t]$, with $E_C^n[t] = 0$ if the “no compression” option is chosen ($k_n[t] = 0$).

If the transmission scheduler allocates power $P_{tran}^l[t] = P_{max}$ for link $l = (n, \tilde{n})$, then node n has the opportunity to transmit data for the duration of one time slot, T_{slot} .

Now, at time t , node n can transmit $\mu_l[t]$ bits over link l . However, as later described in Section 4, in our scheme transmission schedules are determined based on the queue backlog at different nodes. In most cases, when it is the turn of node n to transmit, its queue will have more data than it can possibly transmit during one slot, i.e., $U_n[t] \geq \mu_l[t]$. Thus, the energy consumed (at node n) by the data transmission on link l during slot t is

$$E_T^l[t] = P_{tran}^l[t] = P_{max} \times T_{slot} \quad (6)$$

When $U_n[t] < \mu_l[t]$, we assume (for analytical tractability) that the transmitter stays up for the entire slot duration. In practice, of course, the transmitter does not have to do this.

Node \hat{n} consumes energy while receiving the data packet transmitted by node n . Suppose the wireless interface at each node dissipates a constant amount of power P_{recv} when in receive mode. The energy consumed (at node \hat{n}) for packet reception on link l during time slot t , under the same assumptions as above, is:

$$E_R^l[t] = P_{recv} \times T_{slot} \quad (7)$$

We assume that the wireless interface at a node does not consume any energy when it is not transmitting or receiving data. This is an idealized assumption, and requires the existence of a well-designed network duty-cycling protocol. Such a protocol coordinates packet transmissions in a way that nodes can turn their radios off in order to conserve energy. In general, these protocols [3, 10] do not achieve the ideal we have assumed, but incur a very small amount of overhead in determining whether it is safe to go back to sleep. We examine this deviation from the ideal in our experiments in Section 6.

Finally, nodes can expend significant energy *overhearing* packets, when omnidirectional radios are used (as we have assumed). A standard method [22] for overhearing-avoidance is to use the RTS/CTS message exchange to determine whether a node can go to sleep or not. Before transmitting a data packet for node \hat{n} , a node n broadcasts an RTS control packet that identifies \hat{n} as the intended receiver and the data packet's transfer duration. All the nodes that are not identified as the receiver can then safely turn off their radios for the duration of the data packet transfer.

We account for this overhearing-avoidance in our analysis in the following manner. Node n broadcasting a packet is equivalent to node n transmitting that packet on all its outgoing links $l \in \Omega_n$. We assume that a fraction $\alpha < 1$ of the $\mu_l[t]$ bits is consumed by a control message. The energy consumed at all the neighbors of node n , except \hat{n} , due to these nodes receiving a control message is

$$\begin{aligned} E_B^l[t] &= \sum_{\hat{i} \in \Omega_n, \hat{i} \neq l} \alpha (P_{recv} \times T_{slot}) \\ &= \alpha \times (|\Omega_n| - 1) \times (P_{recv} \times T_{slot}) \end{aligned} \quad (8)$$

In our analysis, we assume that all timeslots are of the same duration. Without loss of generality, we define T_{slot} to be of unit length for the rest of the paper.

Since node n transmits $\mu_l[t]$ total (control and data) bits on link l , the total network-wide energy consumed due to node n 's compression and transmission activity during time slot t , E_{tot}^n , can be written as,

$$E_{tot}^n[t] = E_C^n[t] + \sum_{l \in \Omega_n} \left(E_T^l[t] + E_R^l[t] + E_B^l[t] \right) I(P_{tran}^l[t])$$

where $I(P_{tran}^l[t])$ is the indicator function.

We define the total system energy expenditure during slot t as the sum of the energy expenditure at each node. The time average total system power expenditure is given by

$$\liminf_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=1}^N E_{tot}^n[\tau] \quad (9)$$

Our goal is to design an algorithm *for making compression and transmission scheduling decisions to minimize the time average total system power expenditure (9) while en-*

*surging network stability*². Formally, we define a network to be stable if:

$$\limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=1}^N \mathbb{E} \{U_n[\tau]\} < \infty \quad (10)$$

Note that network stability implies finite average queue backlog, and hence, finite average delay at each node.

4. THE SEEC ALGORITHM

In this section, we present our first contribution: the design and analysis of a *centralized* algorithm, SEEC that minimizes the system energy expenditure while adapting to topology, network dynamics, application data rates, and platform differences. In subsequent sections, we explore the design and performance of a distributed variant, DSEEC.

In designing SEEC, we impose an additional requirement: our algorithm must result in a *stable* network (Equation 10). This requirement suggests a starting point, the Lyapunov optimization framework [8, 9]. This framework can incorporate performance metrics such as energy expenditure, fairness, etc., into the *Lyapunov drift*, a well-known technique for developing stabilizing control algorithms. The key idea is to define a non-negative, scalar function, called a *Lyapunov function*, that measures the aggregate congestion of all the queues in the network during timeslot t . The Lyapunov drift represents the expected change in the Lyapunov function from one timeslot to the next. Under the Lyapunov optimization framework, control algorithms designed to minimize the Lyapunov drift over time are guaranteed to stabilize the network and achieve *near-optimal* performance for a given optimization objective (energy expenditure in case of SEEC).

4.1 Design of SEEC

We first describe SEEC, designed using the Lyapunov drift technique. We present the details of the derivation and then, in next subsection, analyze SEEC's optimality characteristics. SEEC decouples the choice of the compression algorithm and the transmission power allocation into two separate algorithms. Both algorithms involve a single parameter $V > 0$ that controls the trade-off between energy consumption and delay.

Compression Algorithm. Every timeslot t , each node $n \in \mathcal{N}$ observes the data collected by its sensor, $A_n[t]$, and its current queue backlog, $U_n[t]$. It then chooses a compression option $k_n[t] \in \mathcal{K}$ as follows:

$$k_n[t] = \arg \min_{k \in \mathcal{K}} (U_n[t]m(A_n[t], k) + V\phi(A_n[t], k)) \quad (11)$$

We break ties arbitrarily if multiple compression options satisfy Equation (11). We describe the intuition for this algorithm below.

Transmission Algorithm. For each link $l \in \mathcal{L}$, we define the *differential queue backlog* $U_l[t]$ during timeslot t as $U_l[t] = U_n[t] - U_{\hat{n}}[t]$, where $l = (n, \hat{n})$ is a link from node n to node \hat{n} .

Let $\vec{U}[t] = (U_n[t])$ be the vector of queue backlog at all nodes. Every timeslot t , the transmission scheduler observes

²It may make sense, from the perspective of our application, to consider adding other constraints (for example, constraining average energy usage on individual nodes). We have left this to future work.

the current queue backlogs $\vec{U}[t]$ and the channel state $\vec{S}[t]$, and allocates a power vector $\vec{P}_{tran}[t] = (P_{tran}^l[t])_{l \in \mathcal{L}}$ that solves the following optimization problem:

$$\begin{aligned} & \text{Maximize} && \sum_n \sum_{l \in \Omega_n} (U_l[t] \mu_l[t] - V \tilde{P}[t]) && (12) \\ & \text{subject to:} && \\ & \tilde{P}[t] = P_{tran}^l[t] + I(P_{tran}^l[t])[1 + \alpha(|\Omega_n| - 1)]P_{recv} \\ & \vec{P}_{tran}[t] = (P_{tran}^l[t])_{l \in \mathcal{L}} \in \mathcal{P} \end{aligned}$$

In Section 2, we assumed that \mathcal{P} is finite and that the *capacity-power* curves $C_l(\vec{P}, \vec{S})$ (that determine $\mu_l[t]$) are piecewise continuous. Hence, there exists a maximizing power allocation. We break ties arbitrarily if multiple maximizing power vectors exist.

At a high level, these two algorithms work together as follows. For each time slot, the compression algorithm chooses the option $k_n[t]$ that minimizes a *weight function* which depends on the current queue backlog at a node (Equation (11)). In order to maximize the summation in Equation (12), the transmission algorithm will schedule transmissions only on links for which the *link-weight* $d_l[t] = (U_l[t] \mu_l[t] - V \tilde{P}[t]) > 0$. If transmission on two links with positive link-weight cannot be scheduled simultaneously, then the transmission algorithm picks the link with larger link-weight.

Next, we describe how we derive SEEC (which attempts to jointly make compression and scheduling decisions in a stable fashion) using the Lyapunov drift technique [8].

Lyapunov analysis. Before we embark upon our analysis, we need a closed form expression for the time evolution of queue lengths in the system. Equation (5) provides this, but does not include the overhead³ of RTS/CTS control messages (Section 3). The modified equation is as follows:

$$\begin{aligned} U_n[t+1] & \leq \max \left(U_n[t] - \sum_{l \in \Omega_n} \tilde{\mu}_l[t], 0 \right) \\ & + R_n[t] + \sum_{l \in \Theta_n} \tilde{\mu}_l[t] \end{aligned} \quad (13)$$

where $\tilde{\mu}_l[t] = (1 - \alpha) \times \mu_l[t]$.

We define a quadratic Lyapunov function of queue backlogs as:

$$L(\vec{U}[t]) \triangleq \frac{1}{2} \sum_{n=1}^N (U_n[t])^2 \quad (14)$$

and the one-step conditional Lyapunov drift $\Delta(\vec{U}[t])$ as:

$$\Delta(\vec{U}[t]) \triangleq \mathbb{E} \left\{ L(\vec{U}[t+1]) - L(\vec{U}[t]) \mid \vec{U}[t] \right\} \quad (15)$$

The Lyapunov drift for our system is given by the following lemma. (See [8] for further details on Lyapunov drift.)

Lemma 1 Suppose the r.v. $A_n[t]$ at each node n , and the channel states $\vec{S}[t]$ are i.i.d. over timeslots. For the queue

³Accounting for such overhead may not be strictly necessary in case of SEEC since it uses a centralized transmission scheduler in a time-slotted system (i.e., α may be 0). However, it is needed in the case of a distributed transmission scheduler (DSEEC, Section 5), so we introduce it here and carry it forward in the derivations that follow.

evolution, given in Equation (13), and the Lyapunov function, defined in Equation (14), the one-step Lyapunov drift for our system satisfies the following constraint for all t and all $\vec{U}[t]$:

$$\begin{aligned} \Delta(\vec{U}[t]) & \leq BN - \sum_n \mathbb{E} \left\{ \sum_{l \in \Omega_n} \tilde{\mu}_l[t] U_l[t] \mid \vec{U}[t] \right\} \\ & + \sum_{n=1}^N U_n[t] \mathbb{E} \left\{ R_n[t] \mid \vec{U}[t] \right\} \end{aligned} \quad (16)$$

with the constant B defined as follows.

$$\begin{aligned} B & \triangleq (R_{max} + \mu_{max}^{in})^2 + (\mu_{max}^{out})^2, \quad R_{max} \triangleq \max_n (\mathbb{E} \{ R_n[t] \}) \\ \mu_{max}^{in} & \triangleq \max_{(n, \vec{s} \in \mathcal{S}, \vec{P} \in \mathcal{P})} \sum_{l \in \Theta_n} \tilde{\mu}_l[t], \quad \mu_{max}^{out} \triangleq \max_{(n, \vec{s} \in \mathcal{S}, \vec{P} \in \mathcal{P})} \sum_{l \in \Omega_n} \tilde{\mu}_l[t] \end{aligned}$$

PROOF. given in [16], due to lack of space. \square

Incorporating the energy constraint. Recall that our goal is to design an algorithm that makes joint compression and transmission decisions while minimizing energy usage. To do this, we use the Lyapunov optimization framework of [8, 9]. We add a weighted cost (total system energy consumed during slot t) to the Lyapunov drift, in Equation (16), to get:

$$\begin{aligned} \Delta(\vec{U}[t]) + V \sum_{n=1}^N \mathbb{E} \left\{ E_{tot}^n[t] \mid \vec{U}[t] \right\} & \leq \\ BN - \sum_n \mathbb{E} \left\{ \sum_{l \in \Omega_n} \tilde{\mu}_l[t] U_l[t] \mid \vec{U}[t] \right\} & \\ + \sum_{n=1}^N U_n[t] \mathbb{E} \left\{ R_n[t] \mid \vec{U}[t] \right\} + V \sum_{n=1}^N \mathbb{E} \left\{ E_{tot}^n[t] \mid \vec{U}[t] \right\} & \end{aligned} \quad (17)$$

In this inequality, we can expand the term $\mathbb{E} \left\{ E_{tot}^n[t] \mid \vec{U}[t] \right\}$ using Equations (6), (7), and (8), and substitute the value of $\mathbb{E} \left\{ R_n[t] \mid \vec{U}[t] \right\}$ (Equation (1)) to get:

$$\begin{aligned} \Delta(\vec{U}[t]) + V \sum_{n=1}^N \mathbb{E} \left\{ E_{tot}^n[t] \mid \vec{U}[t] \right\} & \leq \\ BN - \sum_n \mathbb{E} \left\{ \sum_{l \in \Omega_n} \tilde{\mu}_l[t] U_l[t] - V \tilde{P}[t] \mid \vec{U}[t] \right\} & \\ + \sum_{n=1}^N \mathbb{E} \left\{ U_n[t] m(A_n[t], k_n[t]) + V \phi(A_n[t], k_n[t]) \mid \vec{U}[t] \right\} & \end{aligned} \quad (18)$$

where $\tilde{P}[t] = P_{tran}^l[t] + I(P_{tran}^l[t])[1 + \alpha(|\Omega_n| - 1)]P_{recv}$.

SEEC is designed to minimize the RHS of (18). There are three salient points to note from 18. First, because the inequality incorporates the Lyapunov drift, SEEC is stable. Second, comparing the second term on the RHS of (18) and Equation (12), SEEC's transmission algorithm contributes to minimizing the RHS of (18) by maximizing this negative term. Finally, comparing the third term on the RHS of (18) and Equation (11), we see that SEEC's compression algorithm minimizes this positive term (in order to minimize the Lyapunov drift). Taken together, SEEC ensures stable, joint compression and transmission scheduling, with the goal of minimizing energy consumption.

4.2 Performance Bounds on SEEC

Next, we provide an analytical bound on the system energy expenditure achieved by SEEC compared to an *optimum* value. The optimum value is characterized by the class of stationary *randomized* algorithms that make the compression and transmission scheduling decisions in a multi-hop network according to a fixed probability distribution. We then make the following contributions. For this restricted class of algorithms, Lemma 2 describes the minimum system power consumption, P_{av}^* , required to achieve network *stability*. Theorem 1 shows that *any* joint compression and transmission scheduling algorithm for multi-hop networks (and therefore SEEC) that stabilizes the system will require a system power expenditure of at least P_{av}^* . In Theorem 2, we show that SEEC can achieve an average system power consumption arbitrarily close to P_{av}^* .

Stationary randomized algorithms. Consider the class of stationary randomized algorithms for making compression and transmission scheduling decisions. Such algorithms choose a compression option (based only on $A_n[t]$) and the transmit power allocation (based only on $\vec{S}[t]$) for each time slot t according to a fixed probability distribution. For example, one policy can be to pick a compression option *uniformly* at random as well as, based on the current channel state $\vec{S}[t]$, similarly choose the transmit power vector $\vec{P}_{tran}[t]$ according to a fixed probability distribution. Note that these algorithms do not consider the queue backlogs $U_n[t]$ while making their decisions.

Condition for achieving stability. Suppose that the data arrival process, i.e., the sequence $A_n[t]$, $t \geq 0$, is ergodic with a steady state distribution p_A ⁴. For a given stationary randomized compression decision policy, let the output data rate (in bits/slot) of the compression module be $r_n = \mathbb{E}\{R_n[t]\} \leq \mathbb{E}\{bA_n[t]\}$. We define a lower bound on r_n , denoted by r_{min}^n as follows:

$$r_{min}^n \triangleq \mathbb{E} \left\{ \min_{k \in \mathcal{K}} m(A_n[t], k) \right\} \quad \text{for all } n \in \mathcal{N} \quad (19)$$

where the expectation is taken over $A_n[t]$. Thus, r_{min}^n is the minimum average bits/slot delivered to the output queue by the compression module at node n , assuming that the compression option that results in the largest expected data size reduction is used in every timeslot. Clearly, $r_n \geq r_{min}^n$.

Suppose the process representing the time varying channel state, i.e., the sequence $\vec{S}[t]$, $t \geq 0$, is ergodic with a steady state probability distribution π_s . Under a given stationary randomized transmission scheduling (and transmit power allocation) policy, let $\vec{f} = (f_l)_{l \in \mathcal{L}}$ where $f_l = \mathbb{E}\{\mu_l[t]\}$. A combination of compression and transmission scheduling policy will stabilize the network if and only if $\vec{f} = (f_l)_{l \in \mathcal{L}}$ define a network-flow satisfying the following conditions⁵.

$$f_l \geq 0 \quad \forall l \in \mathcal{L}; \quad \epsilon > 0 \quad (20)$$

$$\sum_{l \in \Omega_n} f_l - \sum_{l \in \Theta_n} f_l = r_n + \epsilon \quad \forall n \neq \text{sink} \quad (21)$$

$$\sum_{n=1}^N r_n = \sum_{l \in \Theta_{\text{sink}}} f_l \quad (22)$$

⁴For ease of exposition, we assume that $p_{A_n} = p_A$ for all n .

⁵ ϵ in Equation (21) is needed to produce an appropriate randomized policy; we omit the details due to lack of space.

Let Λ denote the set of rates for which there exists an *achievable* network-flow \vec{f} . Clearly, if $\vec{r}_{min} = (r_{min}^n)$ does not belong to Λ , then for the arrivals $A_n[t]$, it is not possible to stabilize the network, even by always compressing data. In our subsequent analysis, we assume that $\vec{r}_{min} \in \Lambda$. A formal definition of Λ can be found in [12], where it is defined as the *Network Capacity Region*.

For the class of stationary randomized policies, we define the *minimum-energy compression function* $h_n^*(r)$ (for all nodes n) and the *minimum energy transmission function* $g^*(\vec{f})$ as follows.

Definition 1 For each node n , for any value of r_n such that $r_{min}^n \leq r_n \leq \mathbb{E}\{bA_n[t]\}$, the minimum-energy compression function $h_n^*(r_n)$ is defined as the infimum value h for which there exist probabilities $(\gamma_{a,k})$ for $a \in \{1, 2, \dots, m\}$, $k \in \mathcal{K}$, such that:

$$\mathbb{E}\{E_C^n[t]\} = \sum_{a=0}^m \sum_{k=1}^K p_A(a) \gamma_{a,k} \phi(a, k) = h \quad (23)$$

$$r_n = \mathbb{E}\{R^n(t)\} = \sum_{a=0}^m \sum_{k=1}^K p_A(a) \gamma_{a,k} m(a, k) \quad (24)$$

$$\gamma_{a,k} \geq 0 \quad \forall a, k, \quad \sum_{k=1}^K \gamma_{a,k} = 1 \quad \forall a$$

Given $\vec{S}[t]$ with distribution π_s and *capacity-power* curves $\vec{C}(\vec{P}, \vec{S}) = (C_l(\vec{P}, \vec{S}))_{l \in \mathcal{L}}$, Neely et al. define the *Network Graph Family*, Γ , as the set of average transmission rates $\vec{w} = (w_l)$ that can be achieved [12]. Different power allocation algorithms will lead to different \vec{w} .

Definition 2 For any $\vec{w} = (w_l)_{l \in \mathcal{L}} \in \Gamma$, we define the *minimum energy transmission function* $g^*(\vec{w})$ as the *infimum value* g for which there exists a stationary randomized power allocation policy that chooses transmit power vector $\vec{P}_{tran}[t]$ as a random function of the observed channel state vector $\vec{S}[t]$, and independent of the current queue backlog, such that:

$$\sum_{l \in \mathcal{L}} \mathbb{E}\{\tilde{P}^l[t]\} = g, \quad \mathbb{E}\{\mu_l[t]\} = w_l \quad \forall l \in \mathcal{L}$$

where $\tilde{P}^l[t] = P_{tran}^l[t] + I(P_{tran}^l[t])[1 + \alpha(|\Omega_n| - 1)]P_{recv}$ and $P_{tran}^l[t]$ is the power allocated for transmission on link l during timeslot t .

The following lemma shows that the infimum values $h_n^*(r_n)$ and $g^*(\vec{w})$ are *achievable*.

Lemma 2 Consider $\vec{r} = (r_n)$ with $r_{min}^n \leq r_n \leq \mathbb{E}\{bA_n[t]\}$ $\forall n$ and $\vec{f} = (f_l) \in \Gamma$ satisfying conditions (20)-(22). For such a scenario, there exists a stationary randomized policy that chooses compression option $k_n^*[t]$ and transmit power allocation $\vec{P}^*[t]$ for timeslot t based only on $A_n[t]$ and $\vec{S}[t]$ (and independent of queue backlogs) such that:

$$\mathbb{E}\{E_C^n[t]\} = \mathbb{E}\{\phi(A_n[t], k_n^*[t])\} = h_n^*(r) \quad \forall n \quad (25)$$

$$\mathbb{E}\{R_n[t]\} = \mathbb{E}\{m(A_n[t], k_n^*[t])\} = r_n \quad (26)$$

$$\sum_{l \in \mathcal{L}} \mathbb{E}\{\tilde{P}^{l*}[t]\} = g^*(\vec{f})$$

$$\mathbb{E}\{\mu_l[t]\} = f_l \quad \forall l \in \mathcal{L}$$

PROOF. In [16]. \square

Optimum system power consumption. The *minimum-energy* functions h_n^* and g^* defined for stationary, randomized algorithms can be used to characterize the minimum system energy consumption for the larger class of joint compression and transmission scheduling algorithms.

Theorem 1 At each node n , let the r.v. $A_n[t]$, $t \geq 0$, be i.i.d. and the corresponding data arrival process be ergodic with a steady state probability distribution p_A . Let the stochastic process representing the time varying channel state ($\vec{S}[t]$) be ergodic with a steady state probability distribution π_s . We assume that $\vec{r}_{min} = (r_{min}^n)$ is within the network capacity region. Then any joint compression and transmission scheduling algorithm that stabilizes the queues $\vec{U}[t] = (U_n[t])$ requires an average system power expenditure that satisfies:

$$\liminf_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=1}^N E_{tot}^n[\tau] \geq P_{av}^* \quad (27)$$

where P_{av}^* is the optimal solution to the following optimization problem:

$$\text{Minimize: } \sum_{n=1}^N (h_n^*(r_n)) + g^*(\vec{f}) \quad (28)$$

subject to:

$$r_{min}^n \leq r_n \leq b \mathbb{E}\{A_n[t]\} \quad \forall n \in \mathcal{N}$$

$$\vec{f} = (f_l)_{l \in \mathcal{L}} \text{ defines a valid network flow}$$

PROOF. In [16]. \square

In practice, solving the optimization problem in (28) might not be possible as it requires exact knowledge of functions h_n^* and g^* , which in turn requires complete *a priori* knowledge of the distributions p_A and π_s . However, this formulation is useful in showing the optimality characteristics of SEEC, as discussed next.

SEEC Performance. The following theorem shows that SEEC can achieve *near-optimal* performance, i.e., achieve power consumption arbitrarily close to P_{av}^* while maintaining network stability and trading-off delay (as described below).

Theorem 2 Suppose the arrival process $A_n[t]$ and the channel state $\vec{S}[t]$ are i.i.d. across timeslots with distributions p_A and π_s , respectively. We assume that it is possible to stabilize the network, i.e., \vec{r}_{min} is strictly interior to the network capacity region Λ . For any control parameter $V > 0$, the compression and transmit power allocation algorithms (Equations (11) and (12)) achieve average power consumption and queue backlogs that satisfy the following constraints:

$$\overline{P_{tot}} = \liminf_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \left(\sum_{n=1}^N E_{tot}^n[\tau] \right) \leq P_{av}^* + \frac{BN}{V} \quad (29)$$

$$\begin{aligned} \overline{\sum_n U_n} &\triangleq \limsup_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_n \mathbb{E}\{U_n[\tau]\} \\ &\leq \frac{BN + VN(\phi_{max} + \tilde{P}_{max})}{\epsilon_{max}} \end{aligned} \quad (30)$$

where the constant B is defined in Equation (17), and $\tilde{P}_{max} = P_{max} + P_{recv}(1 - \alpha + \alpha \tilde{\Omega}_{max})$ with $\tilde{\Omega}_{max} \triangleq \max_n \Omega_n$. The constant ϕ_{max} denotes the maximum expected power consumption for data compression across all nodes when, for each timeslot t , the compression algorithm that is expected to consume maximum power is chosen. It is defined as:

$$\phi_{max} \triangleq \max_n \left(\mathbb{E} \left\{ \max_{k \in \mathcal{K}} [\phi(A_n[t], k)] \right\} \right) \quad (31)$$

where the expectation is over the random arrival process $A_n[t]$. ϵ_{max} is the largest ϵ such that $\hat{r}_n = (r_{min}^n + \epsilon) \in \Lambda$, i.e., if we increased the minimum possible compressed data rate at node n , r_{min}^n , by ϵ_{max} , the resulting rate vector lies on the boundary of the network capacity region Λ .

PROOF. In [16]. \square

Power consumption vs. delay trade-off. We can choose a large value for control parameter V to make B/V arbitrarily small, and hence, achieve time average power consumption $\overline{P_{tot}}$ arbitrarily close to the optimal value P_{av}^* . However, the total average queue backlog $\sum_n \overline{U_n}$ grows linearly in V . Thus, reducing the average power expenditure by choosing a large value for V causes larger queue backlogs resulting in longer delay in delivering data to the base-station. This $O(1/V, V)$ power consumption vs. delay trade-off is inherent in control algorithms designed based on Lyapunov optimization techniques [8].

5. DSEEC: DISTRIBUTED ALGORITHM

SEEC's performance bounds are derived for a timeslotted system. Under general SINR constraints, finding the optimal transmission schedule for a timeslot is NP-hard [8]. However, for certain scenarios, for example, a cell partitioned network [8], finding the optimal transmission schedule is equivalent to finding a *maximum weight matching* in a graph with link-weights $d_l[t] = U_l[t]\mu_l[t] - V\tilde{P}[t]$ [8]. Given the knowledge of the complete network topology and the link-weights, a maximum weight matching can be found in polynomial time.

In practice, most wireless systems (e.g., 802.11 or 802.15.4-based systems) are not time-slotted. In the rest of this paper, we consider a distributed variant of SEEC, called DSEEC, for multi-hop wireless networks based on practical MACs. DSEEC uses the same compression algorithm as SEEC, since that algorithm only requires local information. The key difference between DSEEC and SEEC is the transmission algorithm, in which a node uses only information about queue backlogs from its neighbors. Specifically, our transmission heuristic lets only nodes n with *positive* link-weights, i.e., $d_l[t] > 0$ for some link $l = (n, \tilde{n})$, contend for the wireless channel. Several other *heuristics* have been proposed implementing such backpressure-based scheduling with CSMA-based MACs [20, 21]; Sridharan et al. [17] show that the positive link-weight based heuristic performs as well as others. Because DSEEC is not analytically tractable, we evaluate its performance through simulation.

6. EVALUATION

In this section we evaluate the performance of our algorithm in simulation, on topologies derived from real wireless testbeds and deployments. This section first discusses our experimental methodology, then presents our results.

6.1 Methodology

We begin by describing our experimental methodology.

Implementation in Qualnet. Qualnet [1] is a widely-used, high-fidelity, packet-level wireless simulator. It has been used extensively for the evaluation of mobile ad-hoc networks. In this paper, we use it to mimic a multi-hop wireless data gathering network.

A constant bit rate (CBR) application in Qualnet drives our evaluation of DSEEC. The CBR application generates data periodically. These data bytes are passed on to the compression module that decides whether to compress the data or not. The output of the compression module is queued in a buffer at a wireless link/interface awaiting transmission.

Hardware platform. The power consumption for data transmission and/or compression depends on the hardware platform. In this paper, we model each node as a LEAP2 [19], an embedded networked sensor platform⁶ optimized for low power processing. The two main reasons behind our choice were: (1) the LEAP2 platform can provide detailed, fine-grained, real-time energy usage information, and (2) it provides significant computing resources (fast CPU speeds up to 624 MHz, as well as a large memory and storage subsystem). Hence, it is an ideal platform for developing energy aware applications and system components such as the one described in this paper.

Data compression model parameters. To compute the average compression energy $\phi(a, k)$, and the average data size after compression $m(a, k)$, we use the zlib compression libraries to compress the data collected during a real-world sensor network deployment [13]. This deployment measured vibrations on a large suspension bridge. In all our simulations, we use a single compression algorithm ($K = 1$) (we have left an evaluation of $K > 1$ to future work). We model the data arrival process at each node n , $A_n[t]$, as a CBR application generating 600 bytes of data periodically. With each node n in our simulation, we associate a sensor node \tilde{n} from the deployment in [13]. We partition the data collected by the node \tilde{n} during this deployment into slices of size 600 bytes, and compress each slice separately using the compression function in zlib. We then set $m(A_n[t], 1)$ to be equal to the average data size achieved after compression across these slices (with values ranging from 430 to 470 bytes).

Each simulation run is trace-driven, where each node periodically transmits successive 600 byte slices from the corresponding trace in [13]. If it decides to compress a slice at time t , in simulation we actually compress the slice, instead of (say) using the average.

To estimate $\phi(a, 1)$, we ran our compression program on a LEAP2 node, with our traces as input. We measured the average CPU and memory (SDRAM) energy consumption across several runs. We did not notice any significant variability in the energy consumed for data compression on the LEAP2 node across different runs with data from different nodes. For example, with default CPU speed settings for the LEAP2 node, the average energy consumed for compressing 600 bytes of data, by the CPU and the memory combined,

⁶We expect our results to generalize to other sensor platforms like the motes, although we have not evaluated these platforms. Indeed, in [15] these platforms are used to point out that always compressing data is not necessarily energy-efficient.

was 3.35 mJ. Hence, in our simulations, we model the energy consumed for data compression as a constant equal to 3.35 mJ for each node n (thus, $\phi(A_n[t], 1)$ is equal to 3.35 mJ for each node n).

Data Transmission model parameters. The LEAP2 platform supports 802.11 radios, with a measured radio power consumption of 400 mW, regardless of the wireless interface state (transmitting, receiving or idle) [19]. We use the 802.11b MAC and physical layer implementation in Qualnet in our simulations. We set the maximum transmission rate to 2 Mbps, and the radio power consumption to 400 mW when the wireless interface is in transmit or receive state. We assume the existence of a scheme for radio duty cycling that turns off the radio whenever possible, and discuss its impact in Section 6.2.

We activate RTS/CTS in 802.11. In our energy accounting, we assume simple overhearing avoidance using RTS/CTS. Before transmitting a data packet, a node n sends a request-to-send (RTS) control packet that identifies the intended destination \hat{n} . We assume that all nodes other than \hat{n} turn off their radios and do not waste energy trying to receive the data packet transmitted by node n . Similarly, all the nodes receiving the clear-to-send (CTS) packet sent by node \hat{n} to n indicating that \hat{n} is expecting a packet from node n , turn off their radios as well. In our simulations, each data packet is 684 bytes (600 bytes of payload plus 84 bytes of headers). RTS and CTS packets are 20 and 18 bytes, respectively. In addition, the 802.11b implementation in Qualnet defines a 192 μ s synchronization time overhead associated with each transmission. For these values of packet sizes and synchronization time overhead, the energy consumption due to receiving RTS or CTS packets is within 10% of the energy consumed by the wireless interface for receiving a packet. Thus, with RTS/CTS enabled, we set the parameter α in our transmission decision algorithm (Equation (12)) to 0.1.

Network topologies. For a fixed energy consumption by the radio and for data compression, the network topology determines whether compressing data saves energy or not. Two network topology dependent factors can have an impact on the energy savings due to data compression: (a) multi-hop routing and (b) the average neighborhood size.

Intuitively, compressing data at a node that is multiple hops away from the sink can reduce the total energy consumption in the network. Compression reduces the total size of the data (hence, the number of packets) a node needs to transmit. The energy saved from transmitting fewer packets over multiple hops can offset the energy consumed for data compression, leading to a lower total energy consumption (compared to not compressing data before transmission) [15].

In addition, each data packet transmission incurs an energy consumption not only at the intended receiver but also at nodes that are within the radio range of the sender. A node with a large number of neighbors can reduce the total energy consumption in the network by compressing data – the energy consumed for data compression can be offset by the lower energy consumption for packet reception (due to fewer packets being transmitted).

We evaluate the performance of our algorithm using three qualitatively different data collecting trees: a *cluster-tree* (Figure 1), a *shallow-tree* (Figure 2), and a *deep-tree* (Figure 3). These topologies represent different combinations of

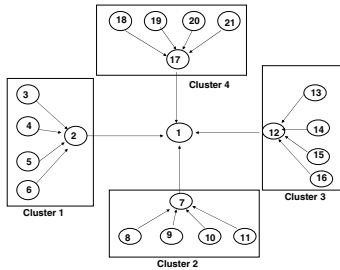


Figure 1: Cluster-tree

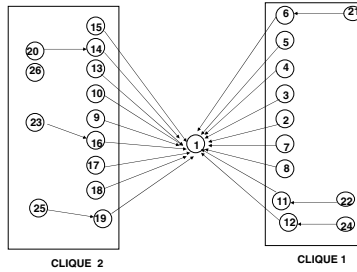


Figure 2: Shallow-tree

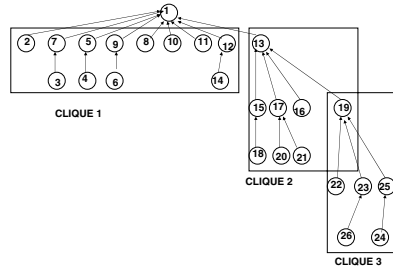


Figure 3: Deep-tree

the two factors – multi-hop routing and neighborhood sizes, as summarized in Table 2.

	# source nodes	avg. # neighbors	Multi-hop paths
Cluster-tree	20	small (4)	short (≤ 2)
Shallow-tree	25	large (11)	short (≤ 2)
Deep-tree	25	medium (8)	long (≤ 4)

Table 2: Data Collection Trees

6.2 Energy Savings

In this section, we provide evidence which illustrates the need for *dynamic compression decisions*. We first show that realistic data collection topologies exist where statically always compressing data may not be the most energy efficient strategy. Then, we demonstrate DSEEC’s adaptability to varying wireless channel conditions, application data rate changes, and diverse platform settings.

Parameters, Metrics, Alternatives. In our simulations, a CBR application at each node generates 600 bytes of data every 5 seconds. In this regime, the data rate is low enough that nodes compress data only to minimize energy, not to stabilize queues. Incidentally, this is the rate at which data is generated at each node in the original deployment. Each simulation run is 100 minutes long, so the CBR application at each node generates ≈ 1200 data packets. Unless otherwise specified, the control parameter V is set to 10^6 (we discuss DSEEC’s sensitivity to V in Section 6.4), and the compression and data transmission parameters are set to the values given in Section 6.1. Finally, each simulation is averaged over 10 runs; the 95% confidence intervals are too small to see on the graphs, so we omit those.

We compare DSEEC against two baseline strategies: (1) *always compress* and (2) *never compress*. These are both static compression decisions that do not take the (energy) cost of data compression relative to the cost of transmission into account. To equalize the effect of transmission scheduling on energy consumption, we use the same transmission decision algorithm (Equation (12)) for all strategies.

We use the *time averaged total system power consumption* as the performance metric. This is computed as the total energy consumption across all nodes divided by the duration of the experiment.

Motivating Dynamic Compression. Figure 4 depicts the total power consumed by the different compression decision strategies, where the *always compress* strategy dissipates more power than DSEEC for all three topologies – 34%

for cluster-tree, 17% for shallow-tree, and 10% for deep-tree. In these topologies, the energy consumed for compressing data is higher than the savings resulting from having to transmit fewer packets as a result of data compression, so with DSEEC nodes never compress data, and its system power consumption is the same as *never compress*.

Across topologies, two facts stand out. First, *always compress* consumes less power in transmitting and receiving packets. This is expected since compressing data reduces the number of packets each node has to transmit. Second, the power savings achieved by DSEEC for the cluster-tree is significantly higher than for the other two topologies. Nodes in the cluster-tree have both smaller neighborhood sizes and shorter multi-hop paths than in the other topologies. In the shallow-tree each node has a large number of neighbors. Thus, a significant fraction of the energy consumed for data compression by *always compress* is offset by the lower energy consumed for receiving data and control (RTS/CTS) packets. The deep-tree topology has nodes with a large number of neighbors as well as nodes that are more than 2 hops away from the sink, and the benefits of DSEEC are the least in this case.

Finally, recall that DSEEC differs from SEEC as follows: it uses only local information for transmission scheduling. To understand the performance hit caused by local scheduling, we implemented SEEC without time slots (SEEC-WT), which makes global scheduling decisions on top of a CSMA MAC. For our topologies, the total energy consumed by DSEEC is comparable to that of SEEC-WT. The difference is highest (about 9%) for the deep-tree topology, where transmissions over multiple-hops can be better scheduled centrally. In practice, most wireless deployments more closely match cluster-tree or shallow-tree, where DSEEC performs extremely well.

Overall, this experiment demonstrates the effect of network topology on power consumption and the benefits of using a *dynamic* algorithm for compression decisions over statically always compressing data. DSEEC is able to determine this *without any explicit information about the multi-hop path used for delivering the data to the sink or about the application data arrival process*. In order to determine *a priori* that nodes should not compress data, a system would need complete information about dynamic quantities such as the network topology, the routing tree, the data arrival rate, and the wireless link qualities.

Duty-cycling overhead. The results shown in Figure 4 assume *ideal* duty-cycling, i.e., that the radio is turned off whenever it is not transmitting or receiving. In practice, due

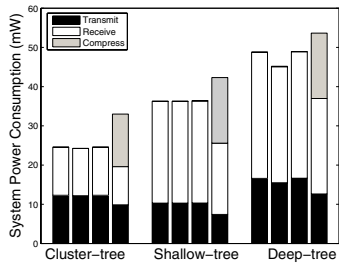


Figure 4: Energy consumption: never compress (first), SEEC-WT (second), DSEEC (third), always compress (fourth)

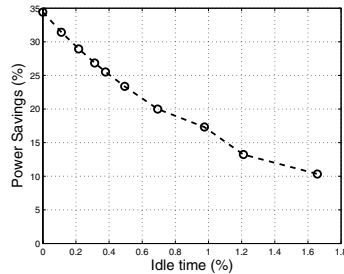


Figure 5: Duty-cycling overhead

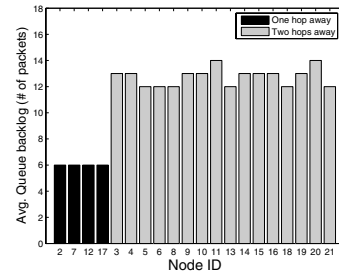


Figure 6: Avg. Queue backlog: Cluster-tree

to the overhead incurred by the duty-cycling protocols, the radio will remain in *idle-state* from time-to-time before it is turned off.

Figure 5 shows how the power savings achieved by DSEEC over *always compress* varies as a function of the overhead of duty-cycling. This overhead is measured by the fraction of the total experiment time that a node could have slept but stayed awake to determine if it was appropriate to sleep, averaged over all nodes. For a carefully-designed duty-cycling protocol, this overhead is topology-dependent but is typically about 1-1.5% [3]. As the figure shows, DSEEC achieves 12-17% savings over the static decision in this regime.

6.3 Adapting to Dynamics

In practice, a static compression decision is undesirable not only because of the detailed information needed for making such a decision, but also because the system needs to adapt to changes in wireless channel conditions, application traffic demand or platform settings. If the wireless link qualities and/or the application data rate change significantly, then a static decision would have to be recomputed. In contrast, DSEEC can dynamically adapt compression decisions.

Change in link quality. To show that DSEEC can adjust to changes in wireless link quality we use the deep-tree topology and vary the quality of the link between node 13 and the sink node 1 in two experiments. In the first experiment, termed “good link”, none of node 13’s transmissions encounter channel errors. In the second experiment, termed “bad link”, 20% of the transmissions by node 13 get corrupted and not received successfully at node 1 (requiring node 13 to retransmit these packets).

Table 3 depicts the fraction of packets compressed, by DSEEC, at all the nodes in the deep-tree. For nodes in cluster 3 (nodes 19, 22-26), there is a significant increase in the fraction of packets compressed in the “bad” link scenario. The rest of the nodes did not compress any packet in either scenario. The packet drops at node 13 result in retransmissions at the MAC layer. Packet retransmissions decrease the *effective* transmission rate on the 13 → 1 link, which increases transmission energy cost for all nodes routing data through that link. This increase in transmission energy cost results in higher compression at nodes 22 – 26 that are 3 or more hops away from the sink. However, the increase in transmission energy consumption is not large enough to trigger data compression at nodes that are 1 or 2 hops away

Node ID	Light	Heavy	Good link	Bad link
Clique 1				
2-12,14	0	0	0	0
Clique 2				
13	0	0	0	0
15	0	2.7%	0	0
16	0	0.35%	0	0
17	0	31%	0	0
18	0	39.7%	0	0
20	0	1.8%	0	0
21	0	76.8%	0	0.7%
Clique 3				
19	0	46.8%	0	0.5%
22	0	99.8%	0.1%	24.9%
23	0	99.8%	0.45%	34.3%
24	0	99.7%	3.2%	40.2%
25	0	99.3%	0	9.3%
26	0	42.4%	0.87%	2.4%

Table 3: DSEEC, deep-tree: % of packets compressed; change in application load and link quality

from the sink. Note that DSEEC makes these decisions without any *explicit* information about the location of the “bad” link and the extent of packet drops on it.

Compression for queue stability. If the application data rate at a node is greater than the rate R at which that node can transmit data to its parent, then, in order to maintain a stable queue size, the node should compress data to match the transmit rate. If R is small enough that every application packet must be compressed, or if compression is energy-efficient, then the only possible decision is to compress every packet. Otherwise, when compression is not energy-efficient and R is smaller than the application rate but larger than the rate resulting from always compressing the data, then a node can decide to compress a fraction of packets. Moreover, nodes further away from the sink should compress a greater fraction of packets than those closer to the sink. In the following simulations, we show that DSEEC is able to make such *fine-grain* decisions dynamically on a per-packet basis.

We steadily increased the application data rate at the nodes in the deep-tree topology until DSEEC started compressing each packet. We do this by decreasing the packet inter-arrival time. To illustrate DSEEC’s dynamic adaptation to traffic rates, we consider two data points: (1) Light load (300 milliseconds packet inter-arrival time), and (2) Heavy load (180 milliseconds packet inter-arrival time).

Table 3 gives the percentage of application data com-

pressed at each node for the Light and Heavy load scenarios. As the application traffic load changes from light to heavy, nodes in cliques 2 and 3 start compressing data, and more packets are compressed by nodes in clique 3. Nodes in clique 1 do not compress data in the two scenarios.

The level of data compression varies across different nodes due to the following reasons. The $13 \rightarrow 1$ link in the deep-tree topology is a bottleneck link for all the nodes in cliques 2 and 3 trying to send data to the sink. Nodes in clique 1 are outside the transmission range of nodes in clique 2 and 3. Hence, their transmission rate is not affected by the increased contention amongst the nodes in clique 2 and 3 when the application data rate is increased. As a result, the application data rate that can be supported without compression is smaller for nodes in cliques 2 and 3 compared to the nodes in clique 1. This is the reason why nodes in clique 1 never compress the data in both scenarios.

In the Heavy load scenario, nodes in clique 3 compress more data than nodes in clique 2 since they are further away from the sink, and hence, their data is transmitted over more hops as compared to the nodes in clique 2.

Insight into dseec’s adaptability. To understand why DSEEC is able to adapt to topology, link quality, and application data rate, without explicitly monitoring each of these, it helps to look at the heart of the algorithm: the queue backlog at a node determines the decisions made by DSEEC. As we discuss below, changing any of these quantities manifests itself as a change in the queue backlog, which drives the scheduling and compression decisions.

DSEEC’s transmission algorithm uses the *differential* queue backlog on a link (between a transmitter and its intended receiver) to schedule transmissions (Section 4, Equation (12)). Under this scheduling policy, the queue backlog at a node is directly proportional to its hop-count distance from the sink. Figure 6 depicts average queue sizes at the nodes in the cluster-tree topology, for DSEEC. The cluster head nodes 2, 7, 12, and 16 that are one hop away from the sink have an average queue backlog of 6 packets. The rest of the nodes that are two hops away from the sink have an average queue backlog between 12 – 14 packets. With DSEEC’s compression algorithm (Section 4, Equation (11)), nodes with higher queue backlog are more likely to compress data. Hence, with DSEEC, nodes that are multiple hops away from the sink have a higher likelihood of compressing data (due to their larger queue sizes) compared to the nodes that are only 1 or 2 hops away .

A degradation in link quality and/or increase in application data rate can also result in larger queue backlogs at all the nodes in network. An increase in queue backlogs (for example, when the application traffic load changes from Light to Heavy in Section 6.3) can cause DSEEC to change its compression decision. Moreover, the impact of any change in network conditions will always be more significant at nodes that are farther away from the sink because these nodes have a larger queue backlog compared to the nodes that are closer to the sink.

Different Platform Settings. DSEEC does not need to be modified in order to support a different platform or different settings of the same platform. In these cases, all we need to do is to specify the compression and transmission model parameters ($\phi(a, k)$, $m(a, k)$, P_{max}) to DSEEC, and it will adjust its decisions to optimize for that platform.

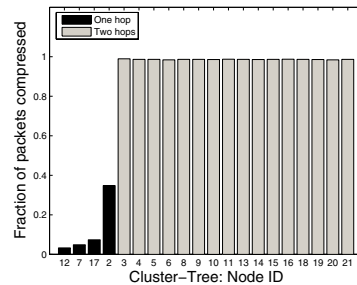


Figure 7: Different Platform

Figure 7 shows the fraction of packets compressed by the different nodes in the cluster-tree for the LEAP2 processor running at its lowest speed (104 Mhz) with the radio operating at 1 Mbps. While DSEEC did not compress any data packets earlier (cluster-tree, Figure 4), it now adapts its compression decision to optimize for the different platform. Nodes that are 2 hops away from the sink compress almost all their data packets, while nodes that are 1 hop away compress fewer data packets.

6.4 Sensitivity to V

The control parameter V impacts the *energy-vs-delay* trade-off. As discussed in Section 4, increasing V reduces the system power consumption but increases the average queue backlog in the system. We have found (results omitted for brevity) that *changing V even by 3 orders of magnitude results in a small impact on the total system power consumption*. Moreover, relative to a static strategy like *always compress*, when V is varied from 10^3 to 3×10^6 , DSEEC consumes from 8.7% to 10% less power. Intuitively, this relative insensitivity results from using the same transmission decision strategy for all three compression strategies.

7. RELATED WORK

Most relevant to our work is Neely’s [11] Lyapunov optimization based joint compression and transmission scheduling algorithm for a *single node* transmitting data to a base station over a *single wireless hop*. It does not consider energy consumption due to packet receptions, overhearing and duty-cycling overhead that must be accounted for in a multi-hop scenario, as we do. Also, that paper only presents a theoretical analysis of the algorithm. In this paper, we provide analytical bounds on the performance of SEEC and also evaluate a distributed version, DSEEC, using simulations.

Two other works related to ours, focusing on compression in wireless networks are [2, 15]. Specifically, Barr and Asanovic [2] consider a *single* wireless hop setting with *fixed* transmission cost (equivalent to *static* channel condition), and focus on estimating the communication to computation energy ratios for several compression algorithms. Sadler and Martonosi [15] consider a multi-hop *static* setting, where they experimentally demonstrate that compression can save significant amount of energy when data is transmitted over multiple hops. They also develop variants of popular compression algorithms more suited for sensor nodes with limited CPU processing speed and memory. In contrast to both these papers, we discuss a principled technique for making on-line compression and transmission decisions in a *dynamic*

environment. Our approach, based on the Lyapunov optimization techniques, also enables us to provide performance guarantees for the centralized version of our algorithm.

Several papers have considered the complementary problem of joint routing and data compression in wireless sensor networks [5, 6, 14]. The main focus of these papers is to design data correlation aware routing trees that enable nodes to achieve better compression efficiency (via in-network aggregation and compression) compared to routing trees that are *agnostic* to data correlation. A version of SEEC that incorporates *compression after in-network data aggregation* can be used on top of these joint routing and data compression schemes to enable the nodes to adapt their compression decisions to changes in the routing tree, in addition to link quality and application data rate.

Lyapunov optimization based techniques have also been used to design stable algorithms that optimize different performance metrics [8]. For example, a joint transmit power allocation and transmission scheduling algorithm (EECA) that minimizes the system energy expenditure is discussed in [9]. SEEC's transmission scheduling algorithm is similar to EECA in its use of a differential queue backlog associated with links for making the transmission decision. However, unlike that work, we consider the energy consumption due to data transmission, reception, and overhearing. Georgiadis et al. present Lyapunov optimization based algorithms that maximize the total throughput or achieve fair rate allocation across flows in addition to achieving network stability [8].

An alternative technique for designing algorithms that can optimize a performance metric and achieve network stability is discussed in [20]. It uses primal-dual gradient descent techniques to design a joint scheduling and congestion control algorithm that also maximizes a utility function.

Finally, several recent papers have used backpressure based transmission scheduling for distributed congestion control [17, 21]. The key contribution in these papers is to design mechanisms that enable backpressure based scheduling for CSMA based MACs (802.11 and 802.15.4). As discussed in Section 5, their heuristics can be used in DSEEC.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have described the design of SEEC, a stable energy-efficient compression and scheduling algorithm for multi-hop wireless networks. SEEC can achieve near-optimal energy performance, and its distributed variant DSEEC adapts to topology, link dynamics, platform settings, and application data rates without explicitly taking those factors into account. We intend to pursue several future directions, including extending SEEC to consider spatially correlated data, choice of routing paths, compression at intermediate nodes (not just at the source), multiple radio transmit power levels, and bounded-distortion lossy compression. We also intend to evaluate these extensions on DSEEC, and also evaluate DSEEC more extensively, using multiple compression levels ($K > 1$), different platforms, larger networks, and different compression algorithms.

9. REFERENCES

- [1] Qualnet. <http://www.scalable-networks.com/products>.
- [2] K. Barr and K. Asanović. Energy Aware Lossless Data Compression. In *Proceedings of MobiSys*, 2003.
- [3] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proceedings of IPSN*, pages 450–459, 2007.
- [4] D. Caron, A. Das, A. Dhariwal, L. Golubchik, R. Govindan, D. Kempe, C. Oberg, A. B. Sharma, B. Stauffer, G. Sukhatme, and B. Zhang. AMBROSia: An Autonomous Model-Based Reactive Observing System. In *Proceedings of ICCS, Invited paper*, 2007.
- [5] A. Ciancio, S. Patten, A. Ortega, and B. Krishnamachari. Energy Efficient Data-Representation and Routing for Wireless Sensor Networks Based on a Distributed Wavelet Compression Algorithm. In *Proceedings of the IPSN*, 2006.
- [6] T. Dang, N. Bulusu, and W. chi Feng. RIDA: A Robust Information-Driven Data Compression Architecture for Irregular Wireless Sensor Networks. In *Proceedings of the EWSN*, 2007.
- [7] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proceedings of Mobicom*, 2003.
- [8] L. Georgiadis and M. J. Neely and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006.
- [9] M. J. Neely. Energy Optimal Control for time varying wireless networks. *IEEE Transactions on Information Theory*, 52(7):2915–2934, 2006.
- [10] R. Musaloiu-E., C.-J. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of IPSN*, 2008.
- [11] M. J. Neely. Dynamic Data Compression for Wireless Transmission over a Fading Channel. In *Proceedings of the Conference on Information Sciences and Systems*, 2008.
- [12] M. J. Neely, E. Modiano, and C. E. Rohrs. Dynamic Power Allocation and Routing for Time Varying Wireless Networks. In *Proceedings of the INFOCOM*, 2003.
- [13] J. Paek, O. Gnawali, K.-Y. Jang, D. Nishimura, R. Govindan, J. Caffrey, M. Wahbeh, and S. Masri. A Programmable Wireless Sensing System for Structural Monitoring. In *Proceedings of the 4th World Conference on Structural Control and Monitoring(4WCSCM)*, 2006.
- [14] S. Patten, B. Krishnamachari, and R. Govindan. The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks. In *Proceedings of the IPSN*, 2004.
- [15] C. Sadler and M. Martonosi. Data Compression Algorithms for Energy-constrained devices in Delay Tolerant Networks. In *Proceedings of the ACM Sensys*, 2006.
- [16] A. B. Sharma, L. Golubchik, R. Govindan, and M. J. Neely. Dynamic Data Compression in Multi-hop Wireless Networks. Technical Report 09-905, Computer Science, University of Southern California, April 2009.
- [17] A. Sridharan, S. Moeller, and B. Krishnamachari. Investigating Backpressure based Rate Control Protocols for Wireless Sensor Networks. Technical Report CENG-2008-7, University of Southern California, July 2008.
- [18] K. Srinivasan and P. Levis. RSSI is Under Appreciated. In *Proceedings of EmNets Workshop*, 2006.
- [19] T. Stathopoulos, D. McIntire, and W. J. Kaiser. The Energy Endoscope: Real-Time Detailed Energy Accounting for Wireless Sensor Nodes. In *Proceedings of the IPSN*, 2008.
- [20] A. Umut, M. Andrews, P. Gupta, J. Hobby, I. Sanjeev, and A. Stolyar. Joint scheduling and congestion control in mobile ad-hoc networks. In *Proceedings of INFOCOM*, 2008.
- [21] A. Warrier, L. Le, and I. Rhee. Cross-layer optimization made practical. In *Proceedings of Broadnets, Invited paper*, 2007.
- [22] W. Ye, F. Silva, and J. Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *Proceedings of the ACM Sensys*, 2006.