Opportunistic Source Coding for Data Gathering in Wireless Sensor Networks

Tao Cui, Lijun Chen, Tracey Ho, Steven H. Low, and Lachlan L. H. Andrew
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA, USA 91125
Email: {taocui@,chen@cds.,tho@,slow@,lachlan@}caltech.edu

***Abstract*—We propose a jointly opportunistic source coding and opportunistic routing (OSCOR) protocol for correlated data gathering in wireless sensor networks. OSCOR improves data gathering efficiency by exploiting opportunistic data compression and cooperative diversity associated with wireless broadcast advantage. The design of OSCOR involves several challenging issues across different network protocol layers. At the MAC layer, sensor nodes need to coordinate wireless transmission and packet forwarding to exploit multiuser diversity in packet reception. At the network layer, in order to achieve high diversity and compression gains, routing must be based on a metric that is dependent on not only link-quality but also compression opportunities. At the application layer, sensor nodes need a distributed source coding algorithm that has low coordination overhead and does not require the source distributions to be known. OSCOR provides practical solutions to these challenges incorporating a slightly modified 802.11 MAC, a distributed source coding scheme based on network coding and Lempel-Ziv coding, and a node compression ratio dependent metric combined with a modified Dijkstra's algorithm for path selection. We evaluate the performance of OSCOR through simulations, and show that OSCOR can potentially reduce power consumption by over 30% compared with an existing greedy scheme, routing driven compression, in a 4×4 grid network.**

I. Introduction

Data gathering is a common function of sensor networks, where information sampled at sensor nodes needs to be transported to central base stations for further processing and analysis. In view of the severe energy constraints of sensor nodes and the limited transport capacity of multihop wireless networks, an important topic addressed by wireless sensor networks community has been in-network data aggregation. The idea is to pre-process sensor data in the network by sensor nodes endowed with computational power, so as to reduce expensive data transmission.

This work has been supported in part by DARPA grant N66001-06-C-2020, Caltech's Lee Center for Advanced Networking, a gift from Microsoft Research and the Australian Research Council.

In this paper we consider those data-gathering scenarios where data is sampled at a number of distributed correlated sources and needs to be routed to one or a few base stations or sinks. Data aggregation in this context involves in-network data compression, see, e.g., [1]–[3]. Such compression and its interaction with routing has been the subject of several previous studies, some of which are briefly reviewed in Section II.

Much of the existing work on correlated data gathering implicitly assumes routing techniques similar to those in wireline networks, neglecting the characteristics of wireless transmission. On the one hand, wireless transmission is error-prone. Sequential forwarding of packets along a fixed path may incur many retransmissions, and thus exhaust scarce network resources such as energy and capacity. On the other hand, wireless transmission is broadcast in nature. The chance that all the neighboring nodes fail to receive the packet is small (multiuser diversity in packet reception). Moreover, multiple receptions of a packet by different nodes can also be exploited for opportunistic data compression. By leveraging the wireless broadcast advantage and multiuser diversity, we can reduce the number of wireless transmissions needed for data gathering.

We propose a jointly opportunistic source coding and opportunistic routing (OSCOR) protocol for correlated data gathering in wireless sensor networks, which exploits the broadcast nature of wireless transmission. OSCOR broadcasts each packet, which is received by possibly multiple sensor nodes, and opportunistically chooses a receiving neighbor to forward the packet, with the goal of obtaining a path online with highest possible compression and best possible link quality. Opportunistic forwarding with opportunistic compression allows OSCOR to exploit multiuser diversity in packet reception, data compression and path selection, resulting in high expected progress per transmission.

The design of OSCOR involves several challenges. First, sensor nodes need to coordinate wireless transmission and packet forwarding so as to exploit multiuser

diversity in packet reception. Second, sensor nodes need a distributed source coding algorithm that does not require full knowledge of the joint source distributions or too much coordination overhead. Finally, in order to achieve high diversity and compression gain, routing (or more precisely, forwarding decisions) must be based on a metric that is dependent on not only link-quality but also compression opportunities, which is nontrivial because the effect of data compression is not additive along a path and the source distributions are not known *a priori* but are learned online. In this paper, we develop practical solutions to these challenging issues. Our main contributions are

- By slightly modifying 802.11 MAC, we design a low overhead consensus protocol to coordinate wireless transmission and packet forwarding. Although it needs coordination between nodes to choose a single forwarder out of multiple receiving nodes, our protocol is "local" and flexible enough to allow good spatial reuse and to allow easy extension to applications with multicast traffic and multiple sessions.
- We propose a practical distributed source coding scheme that combines and takes advantage of both Lempel-Ziv code and network coding. Lempel-Ziv code does not require the knowledge of the statistics of the data, while network coding is well-suited to distributed compression of information in networks.
- We propose to use expected transmission count discounted by node compression ratio (cETX) and expected opportunistic transmission power discounted by node compression ratio (cOETP) along a path as the path metrics for routing. These two path metrics cannot be simply described as the summation of some link metric over the links in a path. So, existing routing algorithms are not directly applicable. We propose modified Dijkstra's algorithms to update the path metrics cETX and cOETP from a node to the sink and select the shortest path, which is used to prioritize the neighboring nodes and update the forwarding candidate set of a node.

An interesting aspect of OSCOR is the way that opportunistic source coding interacts beneficially with opportunistic routing to route packets over paths with high compression and good link quality. We evaluate the performance of OSCOR and find that OSCOR provides both opportunistic compression and opportunistic routing gains.

The remainder of the paper is organized as follows. Section II introduces sensor network model and data compression, and discusses related work and motivation for this work. Section III describes the idea behind OSCOR and gives the details of its design. Section IV presents a performance evaluation of OSCOR through simulations. Section V concludes this paper with some discussion on future work.

II. Preliminaries

A. Sensor Network Model

A sensor network is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges in $\mathcal{G}$. An edge from node i to node j is denoted either by a single index e or by the directed pair (i, j). We restrict our attention to a single session associated with a number of data sources $s_1, \ldots, s_m \in \mathcal{V}$ and a single sink t, i.e., t attempts to gather information from the sources $s_1, \ldots, s_m$. Our proposed protocol can be readily extended to handle multiple sessions with a single sink or multiple sinks.

Each source s_i periodically measures a continuous random observation X_i. The joint source vector $X = \{X_1, \ldots, X_m\}$ is characterized by a joint probability distribution $p(X_1 = x_1, \ldots, X_m = x_m) = p(x_1, \ldots, x_m)$. Let $\{X(\tau)\}$ be a stationary random process, where $X(\tau) = \{X_1(\tau), \ldots, X_m(\tau)\}$ corresponding to the set of random variables observed at all sources at time-slot τ. We assume that $X(\tau)$ is both spatially and temporally correlated. Each source s_i quantizes $X_i(\tau)$ to generate a discrete random variable $\hat{X}_i(\tau)$. $\hat{X}_i(\tau)$ is compressed into bits using source coding. The bits are packetized and transmitted over the sensor network.

To compare and evaluate different data gathering schemes, we need a common metric. Our focus is on energy expenditure, and we therefore choose to use the expected number of MAC layer transmissions that is needed for successfully delivering a packet from each source to the sink. Each edge e is associated with a cost $c_e \geq 0$ that relates to its communication cost. In this paper, we choose c_e to be the expected transmission count (ETX) [4], which is a metric used in link-quality-aware routing. The ETX of a wireless link is the average number of transmissions necessary to transfer a packet successfully over this link. We will see later that the path metric cETX used in this paper is a sum of ETXs discounted by the node compression ratio along the path.

B. Quantization and Compression

To quantify the performance of a particular scheme, we need to quantify the amount of information generated by the sources and by the aggregation points after compression. In this subsection, for the convenience of presentation we drop the time index τ. Let $h(X_{\mathcal{I}})$ denote the joint entropy of $\{X_i | i \in \mathcal{I}\}$, i.e.,

$$h(X_{\mathcal{I}}) = -\int p(X_{\mathcal{I}}) \log_2 p(X_{\mathcal{I}}) \mathrm{d}X_{\mathcal{I}}. \qquad (1)$$

If X_i are individually quantized with a uniform quantizer with stepsize δ, high-resolution analysis shows that the joint entropy of $\hat{X}_{\mathcal{I}} = \{\hat{X}_i | i \in \mathcal{I}\}$ is [5]

$$H(\hat{X}_{\mathcal{I}}) \approx h(\hat{X}_{\mathcal{I}}) - |\mathcal{I}| \log_2 \delta, \tag{2}$$

where $\hat{X}_i$ is the sample of X_i and $|\mathcal{I}|$ denotes the cardinality of $\mathcal{I}$. For example, for a Gaussian m-dimensional multivariate process with full-rank covariance matrix $\mathbf{\Sigma}$

$$h(X_1, \ldots, X_m) = \frac{1}{2} \log_2 (2\pi e)^m |\mathbf{\Sigma}|, \tag{3}$$

where $|\mathbf{\Sigma}|$ is the determinant of $\mathbf{\Sigma}$. When $\mathbf{\Sigma}$ is singular with rank $\kappa(\mathbf{\Sigma}) < m$, let $|\mathbf{\Sigma}|^+$ denote the product of $\mathbf{\Sigma}$'s non-zero eigenvalues. The joint entropy of $X_1, \ldots, X_m$ is

$$h(X_1, \ldots, X_m) = \frac{1}{2} \log_2 (2\pi e)^{\kappa(\mathbf{\Sigma})} |\mathbf{\Sigma}|^+, \tag{4}$$

and the joint entropy of $\hat{X}_1, \ldots, \hat{X}_m$ can be written as

$$H(\hat{X}_1, \ldots, \hat{X}_m) \approx \frac{1}{2} \log_2 (2\pi e)^{\kappa(\mathbf{\Sigma})} |\mathbf{\Sigma}|^+ - \kappa(\mathbf{\Sigma}) \log_2 \delta. \tag{5}$$

We can write the joint entropy of $\hat{X}_{\mathcal{I}} = \{\hat{X}_i | i \in \mathcal{I}\}$ similarly.

C. Existing Data Gathering Schemes and Motivation

Existing data gathering schemes proposed in the literature can be classified into four classes:

(1) Distributed Source Coding (DSC) [6]–[9]: If the sources have perfect knowledge about their correlations, they can encode/compress data by using distributed source coding [10] (e.g., Slepian-Wolf coding [11]) so as to avoid transmitting redundant information. In [6], it was shown that each source can send its data to the sink along the shortest path without the need for intermediate aggregation. Sources need to coordinate to operate at a certain point within the Slepian-Wolf region such that the total cost is minimized. In [7], a suboptimal hierarchical difference broadcasting scheme is proposed without requiring knowledge of joint entropy of sources. But it works for single sink case only. The scenario of multi-sink is considered in [8], where a suboptimal distributed scheme is proposed and it also requires the information exchange between sources. In [9], we proposed a fully decentralized algorithm without requiring the coordination of sources, which works for both single sink and multi-sink cases. However, this scheme still requires the knowledge of joint entropy of sources for decoding purpose, which is difficult and complicated to estimate in practice. Nevertheless, this scheme provides a baseline for evaluating the other schemes.

(2) Routing Driven Compression (RDC) [1], [2]: In this scheme, the sources do not have any knowledge about their correlations and send data along the shortest paths to the sink while allowing for opportunistic aggregation wherever the paths overlap. Such shortest path tree aggregation techniques are described, for example, in [1], [2], where the tree is generated greedily.

(3) Compression Driven Routing (CDR) [3]: This was motivated by the scheme in [12]. As in RDC, the sources have no knowledge of the correlations but the data is aggregated close to the sources and initially routed so as to allow for maximum possible aggregation at each hop. Eventually, this leads to the collection of compressed data at a central node, which are sent to the sink along the shortest possible path.

(4) Hybrid Clustering [3]: In this scheme, sources form small clusters and data is aggregated within them at a cluster head which then sends data to the sink along the shortest path. Opportunistic aggregation is also allowed wherever the paths overlap. This scheme can be considered as a combination of both RDC and CDR. The optimal cluster size depends on the source correlations, which is unknown in advance. This scheme also requires nodes' coordination to find a cluster head.

In [6]–[9], it is assumed that any edge in the network is error-free and can transmit information at the rate of its channel capacity. In [1]–[3], only joint design of source coding and routing is considered on top of the MAC layer and the routing metric is hop distance, which does not take into account the link quality. None of [1]–[3], [6]–[9] considers exploiting the broadcast advantage and cooperative diversity of wireless networks.

In this paper, we consider joint design of application, network and MAC layers taking advantage of wireless broadcast and cooperative diversity. Practical wireless radios such as the ones based on various IEEE 802 standards (e.g., 802.11, 802.15, etc.) employ only a simple coding strategy, mostly for error detection. Nodes transmit at one of a discrete set of power levels, and rely on a small number of link-layer packet retransmissions to overcome errors. Also, nodes can only transmit at a predetermined set of rates. Our work focuses on developing practical data gathering schemes over sensor networks comprised of radios similar to 802.11.

III. Opportunistic Source Coding

A. Basic Idea

The basic idea of OSCOR works as follows. Each node chooses a set of forwarding candidates with different priorities (we will describe how to decide priority in Section III-B). In each time step, each source attempts to broadcast a packet subject to 802.11 MAC. The nodes within a source's forwarding candidate set that actually receive the packet run a protocol to agree on that the highest priority node keeps the packet and all the other nodes drop the packet to prevent unnecessary multiple

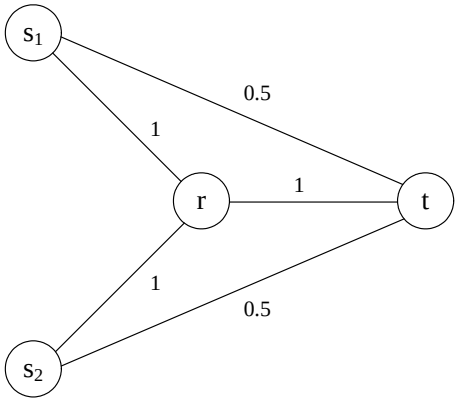

Fig. 1. Example of OSCOR with link delivery probabilities shown along the edges of the graph. The entropy rates of s_1, s_2, and (s_1, s_2) after quantization are $H(\hat{X}_1) = 1$, $H(\hat{X}_2) = 1$, and $H(\hat{X}_1, \hat{X}_2) = 1.5$, respectively.

forwarding of the same packet. If the packet is not received by any node in the source's candidate set, the source broadcasts the packet again until it is received by at least one node in the candidate set or the maximum number of trials is reached. Each node other than the sink waits for a period of time to create opportunity for receiving multiple packets from different sources, which are then compressed, packetized, and forwarded. At the next time step, each source has a new packet to deliver. Intermediate nodes which have received packets to forward are also considered as new sources. The original and new sources repeat the same process. Note that at any time, several nodes may have packets to transmit, which could result in packet collision. We just apply 802.11 MAC to resolve this issue. After an appropriate period of time, the forwarding candidate set of each node is updated by using the information collected in the past.

Fig. 1 gives an example on how OSCOR works. Link delivery probabilities are shown along the edges of the graph. The entropy rates of s_1, s_2, and (s_1, s_2) after quantization are $H(\hat{X}_1) = 1$, $H(\hat{X}_2) = 1$, and $H(\hat{X}_1, \hat{X}_2) = 1.5$, respectively. Source s_i has a packet b_i to deliver, $i = 1, 2$. The forwarding candidate sets for s_1, s_2, r are $\{t, r\}, \{t, r\}, \{t\}$, respectively, where the node listed earlier has higher priority. s_1 first broadcasts b_1. If b_1 is received by t, the transmission finishes (as t has higher priority than r) and s_1 is ready to transmit another, new packet. If b_1 is received only by r, r waits for a period of time. In case that r receives b_2 later and b_2 is not received by t, r compresses b_1 and b_2 and sends the resulting packet to t. Otherwise, r sends b_1 to t directly. We now analyze the average number of transmission required by different schemes. For DSC, we can compress the data at s_1, s_2 such that s_1 sends 1 packet and s_2 only sends 0.5 packets along their respective shortest paths $s_1 \rightarrow t$ and $s_2 \rightarrow t$. If we assume that 0.5 packets require 0.5 transmissions on average, DSC requires $1/0.5 + 0.5/0.5 = 3$ transmissions. For RDC, without compression at sources, it requires $1/0.5 + 1/0.5 = 4$ transmissions. For OSCOR, with probability 0.25 both b_1 and b_2 are received by t; with probability 0.25 b_1 is received by r only and b_2 is received by t; with probability 0.25 b_1 is received by t and b_2 is received by r only; with probability 0.25 both b_1 and b_2 are received by r only, where after compression 1.5 packets ($H(\hat{X}_1, \hat{X}_2) = 1.5$) are needed to deliver. Therefore, the average number of transmissions is $0.25(2 + 3 + 3 + 3.5) = 2.875$. Surprisingly, OSCOR outperforms not only RDC but also DSC.

There are two reasons why OSCOR might outperform existing schemes. First, with OSCOR each transmission can have multiple independent chances of being received, which reduces the number of retransmissions. In Fig. 1, without opportunistic source coding, each packet is received by t with only probability 0.5 and the fact that r can always receive the packet is not taken into account. With opportunistic source coding, each packet can always be received by t and/or r. Another reason is that OSCOR takes advantage of the opportunity for two correlated packets to be received by the same node and hence to be compressed, which again can reduce the number of transmissions. As we will see later, the way our protocol chooses and prioritizes each node's forwarding candidate set can actually increase this opportunity.

Note that our opportunistic routing component in OSCOR is similar to ExOR proposed in [13]. But there are several key differences. First, the path cost metric for routing used in OSCOR is a combination of expected transmission count (ETX) and compression ratio, which makes the calculation of lowest cost path from a node to the sink more complicated. Second, ExOR improves performance by taking advantage of long-distance links, while the opportunistic routing in OSCOR improves performances mainly by reducing multiple retransmissions through multiple-reception gain. Third, in ExOR, only the source specifies the forwarding candidate set and all the nodes use the same candidate set. It leads to a special MAC protocol on top of 802.11 hardware, which goes in rounds and reserves the medium for a single forwarder at any time. This prevents the forwarders from exploiting spatial reuse. Moreover, this highly structured approach to medium access makes it very difficult to coordinate the transmissions of packets of different sources or sinks. In contrast, in our opportunistic routing, each node has its own candidate set and only requires local coordination, and transmissions are scheduled by a slightly modified 802.11 MAC. Therefore, our scheme can enjoy the basic features available to 802.11 MAC.

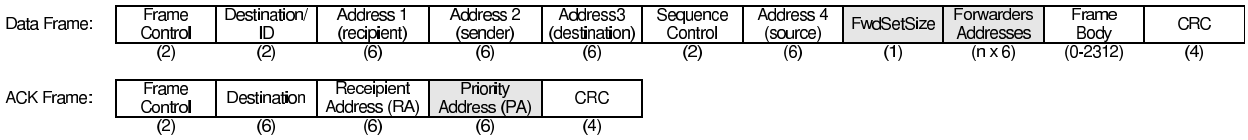

Fig. 2. Data and ACK frame formats in our OSCOR protocol. Lengths in bytes are shown in brackets, and new fields are shaded.

According to the way the received packets are acknowledged and the tolerance of delay, OSCOR is described in two variants in the following.

B. OSCOR with Per-Packet Acknowledgement

In this protocol, each node sends acknowledgement after receiving a packet.

1) Packet Format: Fig. 2 depicts the data and ACK frame formats in the OSCOR protocol. The formats are similar to the 802.11 standard, with the addition of three fields. In the data frame, the FwdSetSize and Forwarders Address fields are added before the frame body, where FwdSetSize indicates the number of forwarding candidates in the forwarder set and Forwarders Address includes the addresses of all the candidates in the forwarder set except the highest priority candidate. The Forwarders Address is in priority order, where candidates with higher priority appear earlier in Forwarders Address. The maximum number of forwarders is denoted `max_fwd_size`. Address 1 is always the address of the highest priority candidate. Address 2 is always the sender address. All the other fields in the data frame are the same as those in 802.11. In the ACK frame, a new PA field is added before the CRC, which indicates the address of the highest priority forwarding candidate that the sending node has known before this ACK is sent.

2) Packet Reception and Acknowledgement: One of the major challenges of OSCOR is how to make the nodes in a node's forwarding candidate set agree on which of them should forward the packet. We propose to use a modified version of the 802.11 MAC which reserves multiple time-slots for receiving nodes to send acknowledgements. This idea shares similarity with the acknowledgement scheme in the preliminary version of ExOR [14].

Let `ack_tx_time` denote the time required to transmit an ACK packet, and let SIFS denote the short interframe space in 802.11. Nodes listen to all transmissions. Each node remembers the priority of the highest-priority ACK it has overheard so far for a particular packet as $\texttt{highest\_ACK\_rx} \in \{1, 2, \ldots, \texttt{max\_fwd\_size}\}$. When a node v hears a data packet, it checks whether its address is in the packet's Address 1 or Forwarders Address field. If so, v checks its priority in the forwarder list denoted i (if v's address appears in Address 1, it sets $i = 1$; if v's address is the k-th address in Forwarders Address, it sets $i = k + 1$), sets its `highest_ACK_rx` to i, and waits $\text{SIFS} + (i-1) \cdot (\texttt{ack\_tx\_time} + \text{SIFS})$ before sending its acknowledgement. During its waiting time, if it overhears an ACK with RA the same value as Address 2 in the data packet it has received, v checks PA field of the ACK to see whether the node u with address PA has a priority greater than its `highest_ACK_rx`. If so, `highest_ACK_rx` is set to the priority of node u. When it is the time for node v to transmit the ACK, it sets the RA field to be Address 2 of its received data packet, and it sets PA field to be the address of the node with priority `highest_ACK_rx`. After transmitting ACK, node v continues to hear possible ACKs and performs the same update on `highest_ACK_rx`. After waiting for all the nodes in the forwarder set to transmit ACKs, node v compares its priority i with its current `highest_ACK_rx`. If the former is less than or equal to the latter (indicating that node v thinks that it is the highest priority recipient), the received packet is kept for further compression and forwarding. Otherwise, the packet is dropped since another node with higher priority also received the data packet.

If the sender does not hear the ACK from any nodes in its forwarder set after time `time_out`, it retransmits the packet. After `max_retry` retransmissions, if the sender still does not get any ACK, it drops the packet and transmits another packet.

Including PA field in ACK helps suppress duplicate forwarding. An example of acknowledgement is shown in Fig. 3. Suppose that node A hears a transmission, that A is the highest-priority candidate, and that A sends an ACK. Node B, the second highest priority candidate, does not hear the ACK, but node C does hear the ACK.

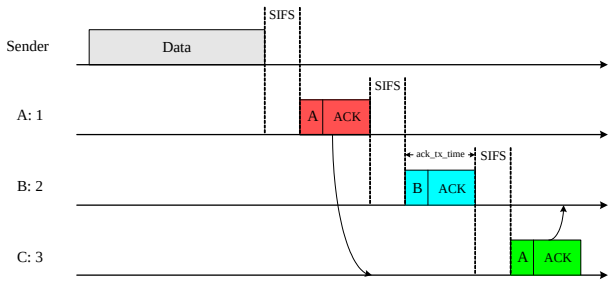

Fig. 3. Example of acknowledgement in packet acknowledgement protocol. The number besides each node indicates the priority of this node.

Suppose further that node B hears node C's ACK. If PA were not added in ACKs, node B would forward the packet, since it is the highest-priority recipient to its knowledge. The fact that node C's ACK indirectly notifies B that node A did receive the packet and it did not need to transmit the packet.

Even though we use this acknowledgement scheme, there still exist chances that the same packet is transmitted by different nodes. According to the rule (described below) for choosing each node's candidate set, there is a high probability that any two nodes in a node's candidate set can hear each other, and thus with high probability that only one copy of a packet is transmitted. If duplicate packets are indeed transmitted, they may be received by the same node later and compressed into a single packet by using source coding.

3) Scheduling: OSCOR uses 802.11's basic access mechanism (i.e., without RTS/CTS) to schedule the nodes' transmissions unlike ExOR which uses a special scheduler on top of 802.11. In 802.11, when a node detects that the medium has been free for more than DCF interframe space (DIFS), it starts backoff and transmits its packet when the backoff counter becomes zero. In 802.11, usually DIFS $=$ SIFS$+2\cdot$`slot_time`, where `slot_time` is the duration of a time slot, an 802.11 parameter. Since OSCOR generates multiple ACKs per packet, this must be extended. Suppose that node A's candidate set contains nodes B and C, that B's priority is higher than C, and that another node D waits for transmission. Suppose further that node C receives a packet from A but node B does not. As node B has higher priority than node C, node C needs to wait for 2·SIFS+`ack_tx_time` before sending its ACK. During node C's waiting time, as node B does not send ACK, node D may detect that the medium is free and its backoff counter may return to zero. Node D then sends its packet, which may collide with node C's ACK at node A. The problem arises because of our packet acknowledgement mechanism and the short DIFS. To avoid this problem, we propose to increase the DIFS to `max_fwd_size`·(SIFS+`ack_tx_time`)+2·`slot_time`. Thus, all the nodes wait for a packet acknowledgement to be accomplished before entering backoff.

4) Source Coding: To increase opportunities for data compression, each node delays received packets for a period of time T_c before compressing and sending them. This allows multiple packets to be received and jointly compressed. The parameter T_c should be chosen based on the application or other system factors. For example, in delay sensitive applications, it is preferable to choose a small T_c, while in power constrained applications, it is preferable to choose a large T_c to allow for maximum possible data compression. Clearly, choosing T_c gives a tradeoff between delay and compression. Let $L_{rx}(k)$ and $L_{cp}(k)$ denote the number of bits before and after the k-th round of compression. We record the compression ratio $\rho_i(k) = L_{cp}(k)/L_{rx}(k)$ at node i.

After time T_c, each node compresses its received packets using any universal source code that does not require knowledge of the statistics of the packets, e.g., Lempel-Ziv [15]. The Lempel-Ziv encoding algorithm is a sequential algorithm, which can compress a packet immediately after it is received without waiting for compression until the end of T_c. The compressed data is then packetized and transmitted. The disadvantage of Lempel-Ziv coding is that it is complicated to extend to the network case, where the packets formed by compression of data at a node may be received by different next hop nodes and undergo joint compression with other packets. To recover the original packets, the sink would have to run the Lempel-Ziv decoding algorithm once for each coding step in reverse order. Moreover, Lempel-Ziv is prone to packet loss. Network coding offers a more elegant solution.

Network coding allows nodes to algebraically combine packets before forwarding them. The use of network coding can significantly improve the ability of the network to transfer information in multicast or lossy settings [16]; practical implementations of such network codes, e.g. [17], are based on distributed random linear network coding [18]. Each coding node forms its output transmissions as a random linear combination of its input transmissions in some finite field $\mathbb{F}_{2^m}$. It is also recognized in [18] that random linear coding can be used to perform distributed compression in a network. However, network coding needs *a priori* knowledge of packets' joint entropies to determine how many coded packets to generate, which may not be available in practice. We thus combine both Lempel-Ziv coding and network coding to take advantage of both. The idea is to use Lempel-Ziv[1] to obtain an estimate of the number of coded bits to generate, denoted as n. Random linear network coding is then applied to generate n coded bits. The coded bits formed by network coding are packetized and sent. This process can also be executed sequentially. Let n_i denote the number of bits generated by Lempel-Ziv after receiving the i-th packet. The output data of Lempel-Ziv is then discarded. Suppose that we have n_i network coded bits, which are generated by using the bits in the first i packets. After receiving the $i+1$-th packet, we add a random linear combination of the bits in the $i+1$-th packet to the n_i network coded bits and form another $n_{i+1} - n_i$ network coded bits by using all the bits in the received $i+1$ packets. This allows parallelization of the coding process.

[1]Note that any entropy estimator such as Burrows-Wheeler (or block-sorting) transform based estimators can replace the Lempel-Ziv encoder here.

The decoding at the sink can be performed by using the polynomial-time minimum-entropy decoding algorithm in [19]. However, this algorithm requires the sink to know the coding vector associated with each packet it receives. Since the size of the coding vector is at least the number of bits in a block, for large blocks it is impractical to include the coding vector in the header of each packet as in traditional network coding [18]. We thus propose to generate the coding coefficients at each node using a pseudo-random number generator with a prespecified random seed known to the sink. Each coded packet is identified by the node at which it was created and a sequence number. Each coding node periodically transmits control packets informing the sink of which packets were coded together to form each of its output packets. This allows the sink to recover the coding vectors of transmitted packets. As the control packet is transmitted every T_c seconds, with a large T_c, the overhead is not significant.

When packet length is fixed in the protocol, the number of bits after source coding may not be an integral multiple of the packet length. In this case, we just append zeros after the encoded sequence. Sometimes it is also wasteful to append zeros as it may happen that after packetization, a packet only contains one useful bit and all the other bits are zero. In this case, the node may wait for more packets until the wasted bits are not many or send part of the bits and leave the rest bits for further compression.

5) Forwarding Candidate Set Generation: After a period of time T_{gen}, each node has done $N_{cp} = \lfloor T_{gen}/T_c \rfloor$ rounds of compression. For each node i, we compute the average compression ratio as $\bar{\rho}_i = \sum_{k=1}^{N_{cp}} \rho_i(k)/N_{cp}$ (initialized as 1). Each node estimates the average link packet delivery rate $\bar{p}_{i,j}$ from i to j and average ACK delivery rate $\bar{a}_{i,j}$ from j to i over time T_{gen}. Let $\bar{\rho}_i(k)$, $\bar{p}_{i,j}(k)$, and $\bar{a}_{i,j}(k)$ denote the estimates in the k-th round. To improve estimation accuracy, we estimate $\bar{\rho}_i$, $\bar{p}_{i,j}$, and $\bar{a}_{i,j}$ using an exponentially weighted moving average

$$\bar{\rho}_i \longleftarrow (1-\alpha)\bar{\rho}_i + \alpha\bar{\rho}_i(k), \tag{6}$$

$$\bar{p}_{i,j} \longleftarrow (1-\beta)\bar{p}_{i,j} + \beta\bar{p}_{i,j}(k), \tag{7}$$

$$\bar{a}_{i,j} \longleftarrow (1-\beta)\bar{a}_{i,j} + \beta\bar{a}_{i,j}(k), \tag{8}$$

where parameters $\alpha, \beta \in [0,1]$. According to [4], the ETX is then estimated as $c_{i,j} = 1/(\bar{p}_{i,j}\bar{a}_{i,j})$.

To update the forwarding candidate set for each node i, we need to first compute the least average number of transmissions required to transmit a packet from node i to sink t, denoted as w_i, which is also called the expected transmission count discounted by node compression ratio (cETX). Note that $\bar{\rho}_i$ means that on average each packet received by node i is compressed into $\bar{\rho}_i$ packets. So, the effect of data compression is not additive along a path, and existing routing algorithms are not directly applicable.

If we use a flow model in which a packet on edge (i,j) means one unit of flow on this edge, this implies that the total outgoing flow of node i is equal to $\bar{\rho}_i$ times of the total incoming flow. Let $f_{i,j}$ denote the flow on edge (i,j). For each node v, we need to solve the following min-cost flow problem:

$$\begin{aligned} w_v = \min_f & \sum_{(i,j)\in\mathcal{E}} c_{i,j} f_{i,j} \\ \text{s.t.} & \sum_j f_{i,j} - \bar{\rho}_i \sum_j f_{j,i} = \begin{cases} \bar{\rho}_i, & \text{if } i = v, \\ -y, & \text{if } i = t, \\ 0, & \text{otherwise,} \end{cases} \\ & y \geq 0. \end{aligned} \tag{9}$$

If $\bar{\rho}_i = 1$ for all $i \in \mathcal{V}$, (9) reduces to the classic min-cost network flow problem in an uncapacited graph or the shortest path problem [20],[2] which can be solved distributedly by using Dijkstra's algorithm or Bellman-Ford algorithm. The coefficient $\bar{\rho}_i$ reflects data compression at each node. The problem (9) with arbitrary value of $\bar{\rho}_i$ is a linear program and can be solved in polynomial time, if all the information on the objective function and constraints is given, which is impractical in real networks. We find that (9) can also be solved distributedly using a modified Dijkstra's algorithm as follows. Let $\mathcal{T}$ denote the set of nodes whose w_v is definitively known. Initially, $\mathcal{T} = \{t\}$ where t is the sink node and $w_t = 0$. Add one node to $\mathcal{T}$ in each iteration. Initially, $w_v = \bar{\rho}_v c_{v,t}$ for all nodes v adjacent to t, and $w_v = \infty$ for all other nodes $v \in \mathcal{V}$. Do the following:

Algorithm 1:

1) ***loop***
2) *Find v not in $\mathcal{T}$ with the smallest w_v;*
3) *Add v to $\mathcal{T}$;*
4) *Update w_u for all u adjacent to v and not in $\mathcal{T}$:*

$$w_u = \min\{w_u, \bar{\rho}_u(w_v + c_{u,v})\}; \tag{10}$$

5) ***until*** *all nodes are in $\mathcal{T}$.*

Let $\mathcal{L}(v)$ denote the forwarding candidate set of node v. For any $u \in \mathcal{L}(v)$, it must satisfy the following conditions:

i) The ETX $c_{v,u}$ should be less than or equal to max_retry, the maximum number of retransmissions, i.e., $c_{v,u} \leq$max_retry;
ii) Node u should be closer to sink t than node v, i.e., $w_v > w_u$.

[2]Shortest path routing is an integer optimization problem. However, what we care is only the cost of the shortest path, which can be obtained by solving (9).

Among those nodes satisfying conditions i) and ii), only the first max_fwd_size lowest ($c_{v,u} + w_u$)-value nodes are added into $\mathcal{L}(v)$. If node u cannot find any node satisfying conditions i) and ii), it adds the node u with minimum $c_{v,u} + w_u$ and $w_v > w_u$ into $\mathcal{L}(v)$. Condition i) ensures that a packet transmitted by node v can be received with high probability at node u. Condition ii) guarantees that packet is always transmitted towards the sink. Next, all nodes u in the forwarding candidate set $\mathcal{L}(v)$ of node v are prioritized according to w_u. The smaller w_u is, the higher priority u has. As we rank the nodes according to w_u, the path with fewer expected number of transmissions is preferable, which may be due to both a shorter distance to the sink and a higher opportunity of data compression on this path. Note that as we adapt $\bar{\rho}_i$ and $c_{i,j}$ over time, the proposed protocol adapts to network change, e.g., nodes dying or moving. When $c_{i,j}$ is fixed, nodes initially have no idea which path has more opportunity to have data compressed. With time, nodes learn the opportunity of compression through $\bar{\rho}$, and they will gradually prefer the paths with high chance of data compression. This is in contrast to the existing data gathering schemes, in which data compression and routing are actually uncoupled.

Algorithm 1 is simple to implement, but does not take into account either the fact that opportunistic routing is employed instead of shortest path routing or the power consumption of ACKs. The following algorithm considers both of these factors. Let P_{data} and P_{ack} denote the energy consumption by sending a data packet and an ACK, respectively. We need to compute the average energy required to transmit a packet from node i to sink t, denoted $\tilde{w}_i$, which is also called the expected opportunistic transmission power discounted by node compression ratio (cOETP). cOETP can also be obtained by solving a linear program (LP) as in (9). However, in this case, the LP is hard to solve distributedly. Alternatively, as in Algorithm 1, let $\mathcal{T}$ denote the set of nodes whose $\tilde{w}_v$ is definitively known, except that $\mathcal{T} = \emptyset$ initially. One node is added to $\mathcal{T}$ in each iteration. Let $\mathcal{L}(v)$ denote the forwarding candidate set of node v, where nodes in $\mathcal{L}(v)$ are in increasing order of $\tilde{w}$. Initially, $\tilde{w}_v = \infty$ and $\mathcal{L}(v) = \emptyset$ for all nodes $v \in \mathcal{V} - t$ and $\tilde{w}_t = 0$, where t is the sink node. Let n_i denote the i-th entry of $\mathcal{N}$. Do the following:

Algorithm 2:

1) ***loop***
2) *Find v not in $\mathcal{T}$ with the smallest $\tilde{w}_v$;*
3) *Add v to $\mathcal{T}$ and to the end of $\mathcal{L}(u)$ for all nodes u adjacent to v and not in $\mathcal{T}$;*
4) *for all u adjacent to v and not in $\mathcal{T}$, add v to the end of $\mathcal{L}(u)$, update $\tilde{w}_u$ according to (11) at the top of next page;*
5) ***until** all nodes are in $\mathcal{T}$.*

The $\tilde{w}_u$ computed by (11) is the average energy consumption by sending a packet from node u to t, where the nominator of (11) is the probability that at least one node in $\mathcal{N}$ receives the packet and we neglect the effect caused by ACK packet loss. Opportunistic routing is counted through $\bar{p}_{u,n_i} \prod_{j=1}^{i-1}(1-\bar{p}_{u,n_j})$, which is the probability that the i-th node in $\mathcal{N}$ receives the packet from node u while all the other $i-1$ higher priority nodes in $\mathcal{N}$ do not. The energy consumption of ACK is counted through $P_{\text{ack}}\left(\sum_{i=1}^{k} \bar{p}_{u,n_i}\right)$. Note that in (11) we implicitly assume that ACK will never lost and duplicated packet forwarding is completely eliminated. As ACK is usually short, the error probability of ACK is small. Also the ACK mechanism of OSCOR discussed in Section III-B2 can effectively prevent ACK loss and packet duplication. Both factors indicate that (11) is a good approximation to the real case. Note that (11) also automatically determines the size of forward set.

The complexity of Algorithm 2 is high as computing (11) has a complexity exponential in the size of $\mathcal{L}(u)$. In large networks, this complexity is not acceptable. We combine Algorithm 1 and Algorithm 2 to get Algorithm 3. In Algorithm 3, we first apply Algorithm 1 to generate $\mathcal{L}(u)$ for each u. According to the order that u is added into $\mathcal{T}$, we compute $\tilde{w}_u$ for each u. First, each $\mathcal{L}(u)$ is reordered according to increasing order of $\tilde{w}$. $\tilde{w}_u$ is then computed by setting $\mathcal{N} = \mathcal{L}(u)$ in (11) directly without performing the $\min$ operation.

Remarks:

- Note that when `max_fwd_size`=1 and $\bar{\rho}_i = 1$, $\forall i \in \mathcal{V}$, OSCOR reduces to a variant of RDC which uses ETX instead of hop count as the path metric. When `max_fwd_size`> 1, our scheme takes advantage of both cooperative diversity and opportunistic aggregation.
- In [7], it was shown that allowing nodes to broadcast does not reduce the cost of data gathering in networks with lossless channel. However, in a network with lossy channels, as indicated in Fig. 1, the data gathering cost may be reduced by exploiting the broadcast advantage or cooperative diversity of wireless medium even with perfect DSC.
- Different from existing data gathering schemes [1]–[3], [6]–[9], which only consider the interaction between application and network layer. Our proposed protocol can be considered as a joint design across application layer, network layer and MAC layer, which does source coding in application layer, runs modified Dijkstra's algorithm at network layer, and handles scheduling and packet forwarding at MAC layer. By using universal source coding and opportunistic routing, our proposed protocol can be

$$\tilde{w}_u = \min_{1\le k\le|\mathcal{L}(u)|}\ \min_{\mathcal{N}\subseteq\mathcal{L}(u),|\mathcal{N}|=k} \frac{\bar{\rho}_u\left(P_{\rm data}+\sum_{i=1}^{k}\tilde{w}_{n_i}\bar{p}_{u,n_i}\prod_{j=1}^{i-1}(1-\bar{p}_{u,n_j})\right)+P_{\rm ack}\left(\sum_{i=1}^{k}\bar{p}_{u,n_i}\right)}{1-\prod_{i=1}^{k}(1-\bar{p}_{u,n_i})}. \quad (11)$$

implemented in a fully distributed fashion.

- We have assumed that all the packets entering a node i roughly have the same contribution to $\bar{\rho}_i$. We do not account for the possibility that different packets may have different impacts on the compression ratio. For example, the compression ratio of compressing only two packets entering i may be less than that of compressing three packets. It will make the protocol complicated by considering this effect.
- Note that DSC can also work with opportunistic routing. However, it requires not only the coordination of the sources but also the statistics of the sources. This approach is not practical so we do not discuss here. Our proposed protocol can also be combined with other existing schemes, e.g., the hybrid clustering scheme in [3], and can be extended to the scenario that only a few nodes can perform data compression.
- In some applications, e.g., [1], sophisticated source coding is not used, and only duplicated packets are removed at each node. OSCOR can be readily modified in this situation.
- By replacing power consumption in (11) with time duration, Algorithm 2 can also be used to improve the throughput of opportunistic routing.

C. OSCOR with Per-Batch Acknowledgement

In the OSCOR protocol with per-packet acknowledgement, each packet is acknowledged after being sent and received. Considering that each node needs to wait time T_c before compression and transmission, it is not power- and time-efficient to acknowledge each packet immediately after receiving it. In the following, we discuss a variant that sends acknowledgements after receiving a batch of packets instead of a single packet. All the components are same as the OSCOR protocol with per-packet acknowledgement except the packet acknowledgement part.

Each sender puts a batch of packets into the transmitting buffer and broadcasts these packets one by one all together. All the nodes in the sender's forwarding candidate set try to receive those packets. After time T_b, each node in the candidate set acknowledges its received packets by following the same way (from high priority node to low priority node) as in the per-packet acknowledgement protocol. The only difference is that each ACK contains a reception report indicating which packets have been received by this node. Each packet in the reception report is labeled by the priority of this node. When another node in the candidate set overhears this ACK, it updates each packet's priority in the reception report in the same way as in the per-packet acknowledgement protocol. Also, whether a packet is kept by a node is decided similarly as in Section III-B. Upon receiving the ACK, the sender removes the packets in the reception report from its buffer. The unacknowledged packets are kept in the transmitting buffer for the next batch until it has been sent `max_retry` times. New packets are put into the transmitting buffer to make a full batch, and a new transmission cycle starts.

As the ACK from one node in the forwarding candidate set may not be received by another node in the set, different from the per-packet acknowledgement protocol where missing one ACK may only result in duplicating one packet, missing one reception report may cause the duplication of many packets. To resolve this problem, after receiving all the ACKs, the sender sends a summary of received reception reports to all the nodes in the candidate set, which indicates for each packet the highest priority node that has received this packet. This prevents possible packet duplication.

Another problem with the per-batch acknowledgement protocol is that each node cannot encode packet immediately after it is received as it does not know whether this packet is also received by a higher priority node. Note that from the reception reports in the previous batches each node can estimate the probability that a received packet is also received by a higher priority node, denoted as p. Each node can also estimate that on average each received packet is compressed into $\bar{\rho}$ packets. On receiving a new packet, with probability $1-p$, a random linear combination of the bits in the received packet is added to the already coded bits and this packet is marked. Also additional $\bar{\rho}$ coded packets are generated by using random linear combinations of the marked packets. After receiving the summary report, the node checks whether an unmarked packet is not received by a higher priority node. If so, a random linear combination of the bits in this packet is added into the existing coded bits.

IV. Evaluation

In this section we report some preliminary evaluation results of OSCOR. To evaluate the performance of

TABLE I
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Path Loss Exponent	2	Slot Time	20 μs
Lognormal fading	0.1 dB	SIFS	10 μs
Transmit Power	23 dBm	DIFS	980 μs
Noise Power	-55 dBm	MAC Header	34 bytes
Data Rate	6 Mbps	PLCP Header	24 bytes
Modulation	BPSK	MAC ACK	14 bytes
`max_retry`	3	`max_fwd_size`	3
T_{gen}	1 s	T_c	74.5 ms

OSCOR, we develop a packet-level simulator that implements our approach, DSC and RDC. Our simulations are based on IEEE 802.11b standard, with some modification as described in Section III-B2. We only implement the OSCOR protocol with per-packet acknowledgement. The values for the parameters used in simulations are summarized in Table I. In all simulations, each source transmits 3000 packets. After every 1s, $\bar{\rho}_i(k)$, $\bar{p}_{i,j}(k)$, and $\bar{a}_{i,j}(k)$ are updated according to (6)-(8), and each node's candidate set is updated by using the algorithms in Section III-B5. We consider a jointly Gaussian data model. The differential joint entropy of the sources is given by (3), where the elements of the covariance matrix $\mathbf{\Sigma}$, $\sigma_{i,j}$, depend on the distance between the corresponding nodes and the degree of correlation. In our simulations, we assume that $\sigma_{i,j} = e^{-d_{i,j}/c}$, where $d_{i,j}$ is the distance in meters between nodes i and j and c is a correlation parameter, in meters. Uniform quantizers with stepsize $\delta = 1$ are used at all sources. The joint entropy of the sources is given by (5). For evaluation simplicity, we assume that $H(\hat{X}_i(\tau), \hat{X}_j(\tau')) = H(\hat{X}_i(\tau), \hat{X}_j(\tau''))$, $\forall \tau', \tau'', i \neq j$, and samples from a given node at different times are independent. We also assume the use of ideal data compression with network coding, where each node knows how many coded packets are needed to send (can be obtained by assuming perfect knowledge of each packet's joint entropies). OSCOR with Algorithm i in Section III-B5 is denoted OSCORi, $i = 1, 2, 3$.

We evaluate the performance of different schemes on a 4×4 grid network shown in Fig. 4. In Fig. 4, we only give the coordinates of nodes 1 and 16 in meters. Fig. 5 shows the average power consumption per bit versus the correlation parameter c with different schemes. We assume that the sources know the perfect knowledge of joint entropy in DSC. To compare the performance of different schemes on the same ground, we use ETX as path metric in both DSC and RDC instead of using hop count. In OSCOR, we choose smoothing parameters $\alpha = \beta = 0.1$ in (6) and (7). As source correlation c increases, the average power consumption reduces because of higher correlation between the packets from different sources. DSC outperforms both RDC and OSCOR as it can remove the redundancy

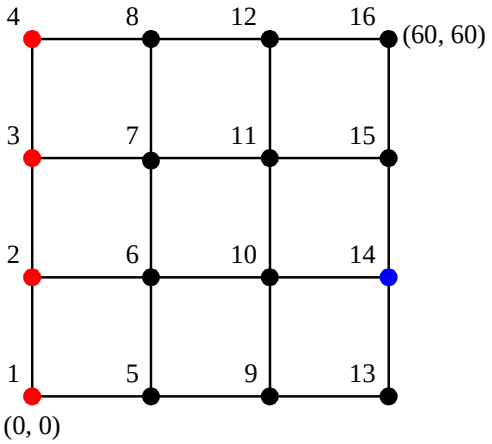

Fig. 4. A 4×4 grid network, where nodes 1, 2, 3, and 4 are sources and node 14 is the sink.

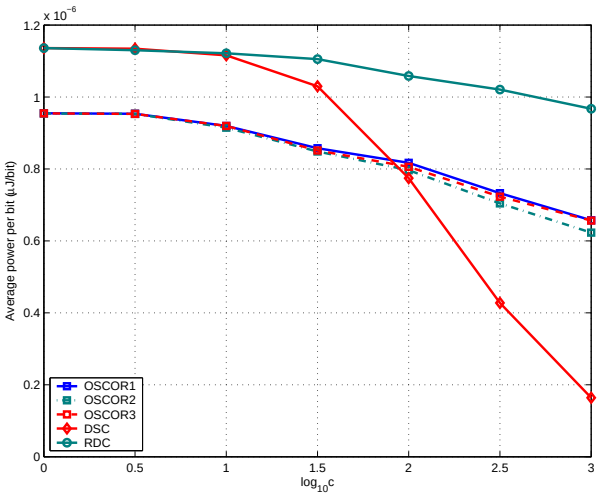

Fig. 5. Average power consumption versus correlation parameter c in the grid network in Fig. 4 with OSCOR, RDC, and DSC. The quantization step size $\delta = 1$.

in the packets perfectly. When $c = 10^3$, OSCOR1 reduces the power consumption by 32% as compared with RDC as OSCOR uses opportunistic compression, compression ratio learning and path adaptation. When $c = 1$, OSCOR1 achieves a 16% power saving over both RDC and DSC, which is due to multiuser diversity and spatial reuse with opportunistic routing. From Fig. 5, we can also see that both OSCOR2 and OSCOR3 have a less power consumption than OSCOR1. OSCOR2 achieves the least power consumption, and OSCOR3 lies between OSCOR1 and OSCOR3. When $c = 10^3$, OSCOR2 attains 5% power saving over OSCOR1. When c is small, the power saving by using OSCOR2 is fairly small. However, in large sensor networks, a large gain may be obtained by OSCOR2.

Fig. 6 shows the evolution of compression ratio $\bar{\rho}$

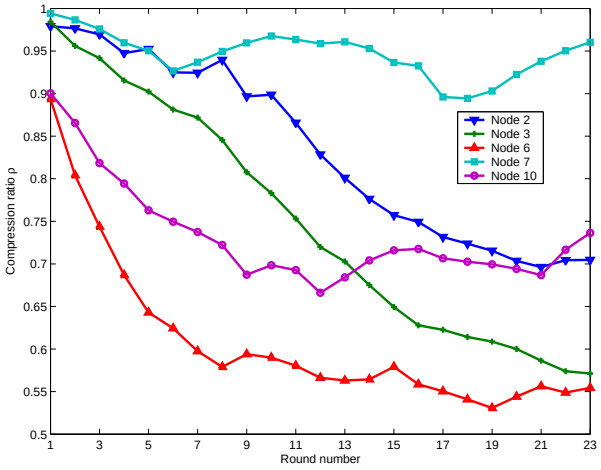

Fig. 6. Evolution of compression ratio ρ at nodes 2, 3, 6, 7 and 10 versus round number in the grid network in Fig. 4 with OSCOR1. The quantization step size $\delta = 1$ and correlation ratio $c = 10^3$.

as a function of rounds with OSCOR1. We only show the nodes with compression ratio less than 0.95. Nodes 2, 3, and 6's compression ratios reduce gradually with round. It is interesting to see that both nodes 7 and 10's compression ratios first decrease and then increase. At first, node 7 is the highest priority node in node 4's forwarding candidate set. Later node 4 finds that its packets have a better chance to be compressed more at node 3. Node 4 then puts node 3 as the highest priority node in its candidate set. The compression ratio at node 7 then increases. Finally, node 4 prefers to send to node 3, node 3 prefers node 6, nodes 1 and 2 both prefer node 6. The same analysis holds for node 10. In RDC, the path is pre-determined and fixed during data-aggregation, and it does not take path adaptation into account.

V. Conclusion

We have presented a jointly opportunistic source coding and opportunistic routing protocol, OSCOR, for correlated data gathering in wireless sensor networks. OSCOR achieves efficient data gathering by exploiting both opportunistic data compression and cooperative diversity associated with wireless broadcast advantage. OSCOR uses a slightly modified 802.11 MAC which allows good spatial reuse, different from that of ExOR. A practical source coding scheme was proposed by using Lempel-Ziv code and network coding. OSCOR also allows path adaptation according to not only link quality but also compression opportunities. Modified Dijkstra's algorithms were proposed to solve the resulting min-cost routing problem. Simulation results showed that OSCOR can potentially reduce the power consumption by 32% compared with RDC in a 4×4 grid network. Further work includes evaluation of our protocol in larger networks, where we expect greater performance gains due to more spatial reuse that allows more diversity in packet reception and more chances in data compression. A close analysis and evaluation of the impact of network topology and traffic pattern on OSCOR is also of interest.

References

[1] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," in *Proc. of International Conference on Distributed Computing Systems*, 2002, pp. 457–458.

[2] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. of International Conference on Distributed Computing Systems*, 2002, pp. 575–578.

[3] S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," in *Proc. of International Conference on Information Processing in Sensor Networks*, April 2004, pp. 28–35.

[4] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. of ACM MobiCom*, 2003, pp. 134–146.

[5] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, 1st ed. Springer, 1991.

[6] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "Networked Slepian-Wolf: Theory, algorithms and scaling laws," *IEEE Trans. Inform. Theory*, vol. 51, no. 12, pp. 4057–4073, Dec. 2005.

[7] J. Liu, M. Adler, D. Towsley, and C. Zhang, "On optimal communication cost for gathering correlated data through wireless sensor networks," in *Proc. of ACM MobiCom*, 2006, pp. 310–321.

[8] K. Yuen, B. Li, and B. Liang, "Distributed data gathering in multi-sink sensor networks with correlated sources," in *Proc. of IFIP Networking*, May 2006, pp. 868–879.

[9] T. Cui, T. Ho, and L. Chen, "On distributed distortion optimization for correlated sources," in *Proc. of IEEE International Symposium on Information Theory*, Jun. 2007.

[10] T. Cover and J. Thomas, *Elements of Information Theory*, 1991.

[11] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. 19, no. 4, pp. 471–480, July 1973.

[12] A. Scaglione and S. Servetto, "On the interdependence of routing and data compression in multi-hop sensor networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 149–160, Jan. 2005.

[13] S. Biswas and R. Morris, "ExOR: Opportunistic routing in multi-hop wireless networks," in *Proc. of ACM SIGCOMM*, Aug. 2005, pp. 133–144.

[14] ——, "Opportunistic routing in multi-hop wireless networks," in *Proc. of Workshop on Hot Topics in Networks*, Nov. 2003.

[15] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. 24, no. 5, pp. 530–536, Sept. 1978.

[16] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[17] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. of Allerton Conf. Comm., Control and Comp.*, Oct. 2003.

[18] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[19] T. Coleman, M. Medard, and M. Effros, "Towards practical minimum-entropy universal decoding," in *Proc. of IEEE Data Compression Conference*, March 2005, pp. 33–42.

[20] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.