SenZip: An Architecture for Distributed En-route Compression in Wireless Sensor Networks [*]

Sundeep Pattem, Godwin Shen, Ying Chen, Bhaskar Krishnamachari, Antonio Ortega
Department of Electrical Engineering - Systems
University of Southern California, Los Angeles, CA, USA.

ABSTRACT

In-network compression is essential for extending the lifetime of data gathering sensor networks. The progress made in designing distributed schemes for en-route compression has not been followed by their adoption in deployments. This can be attributed to the lack of development of software that permits code-reuse and interoperability, while also retaining the flexibility to incorporate future developments. To address this gap, we propose SenZip, an architectural view of compression as a service that interacts with standard networking components. SenZip is designed for achieving completely distributed en-route compression and its utility is illustrated by presenting (a) details of how it helps map specific algorithms to software modules, and (b) results from mote experiments for data gathering with two different compression schemes, DPCM and 2D wavelets.

1. INTRODUCTION

Sensor networks are aiding the evolution of monitoring systems for earth and space science applications [7, 15, 20]. Frequently, these systems require continuous gathering of correlated data. It is possible to exploit the correlations for efficient and long-lived operation via in-network compression, and since this compression will happen en-route to the sink, the relationships between compression and routing have been studied extensively [3, 4, 10, 11, 13, 18, 22].Algorithms for in-network compression of sensor data are by definition spatially distributed, i.e., the data-set itself is distributed across sensors, so that each sensor performs different coding tasks as data is transmitted. Recent work has shown that it is possible to achieve distributed compression with elementary computations and over well-known routing structures [13, 21]. However, only limited efforts have been devoted to understanding the problems associated with *distributed node configuration* for compression. For efficiency and scalability, only a small amount of "local" communications should be needed to determine which nodes exactly perform which compression computations, over what data and how the data is then routed to them. Distributed configuration is also desirable as it can help reduce initialization and reconfiguration times since it is not necessary for a sink node to first gather information about all nodes.

Most earlier work has focused on theory and simulations to understand performance limits. These studies, and some limited system implementations (e.g., [21]), have therefore had limited impact on technology adoption and sensor network software development because they have not yielded modular and inter-operable software. In this paper we move towards addressing this problem by (i) proposing a novel architecture, SenZip, that fits into the overall networking software architecture for sensor networks and (ii) demonstrating that a practical design based on this architecture can be deployed on motes and can achieve distributed configuration and modularity.

The SenZip architecture specifies a *compression service* that can encompass different compression schemes and its modular *interactions* with standard networking services such as routing. This architecture enables a distributed node configuration for compression, just as existing systems make it possible for sensors to configure themselves for routing in a distributed manner. The architecture proposal is based on (a) lessons from overall architectural principles for sensor networks [14], (b) our own experience in implementing a practical wavelet-based distributed compression system, and (c) identifying common patterns in existing compression schemes.

[*]This work was supported in part by NASA under grant AIST-05-0081 and by the NSF through CAREER award CNS-0347621.

ESSA Workshop '09, April 16, 2009, San Francisco, California, USA
Copyright ©2009 ACM 978-1-60558-533-8/09/04... $5.00

To concretely illustrate the utility of the architecture, we show how it can incorporate two different compression schemes, DPCM and 2D wavelets and present results from mote experiments for data gathering in which nodes can configure themselves for compression under different routing conditions. This is the first system demonstration of wavelet-based distributed compression for multi-hop sensor networks.

2. MOTIVATION AND APPROACH

In order to achieve the potential gains from existing schemes in practice, we believe that a *heuristic but principled view of a common architecture for distributed en-route compression* has to be arrived at. The key motivation is to spur the development of modular and interoperable software that will lead to the rapid adoption and deployment of efficient data gathering sensor networks. We also believe that such an architecture will allow the development of novel approaches and schemes for distributed compression. Our approach is informed by a system implementation that is described in Section 5 and principles of an overall sensor network architecture [5, 14]. With this background, we first obtain a broad, minimal abstraction that encompasses several existing compression schemes and their routing structures. This abstraction forms the core of our proposal for an architecture that defines a compression service and its modular interactions with standard routing components, possibly with well-defined, modest extensions.

2.1 Proposed Abstraction

In what follows, we focus only on compression tools to exploit spatial correlation. Note that exploiting temporal correlation is a simpler problem, since this can generally be achieved via local processing at each sensor. We now abstract components that are *necessary* for a range of spatial compression system.

First, spatial compression requires *data exchange between nodes* in the vicinity of each other. In what follows we call **aggregation graph** a graph specifying those data transfers between nodes that are needed to perform joint encoding (different from those data transfers whose sole purpose is to relay already encoded data to the sink). For example, in a predictive compression scheme data at a certain node is encoded after subtracting a prediction based on data from neighboring nodes (those specified by the graph).

Second, nodes that "collaborate" to perform joint encoding will have to carry out computations that in many cases, e.g., distributed wavelets, will be different for different sets of nodes. Thus, we will discuss a notion of **role assignment**: each node has to determine which part of the computation it has to perform in order to achieve a common compression goal. A role assignment is also necessary, otherwise nodes will not know how to properly process their own data nor data of their neighbors. For example, a distributed wavelet transform will not be invertible unless the roles of the nodes are properly assigned.

Third, in a distributed compression system the goal is to accumulate all compressed data in one or more fusion centers. Thus, since data eventually has to flow towards the fusion center, an **ordering** of communication and compression operations along the aggregation graph is necessary to carry out distributed compression correctly and efficiently. Thus, a transmission schedule[1] must be defined to allow each node to collect data from its neighbors in the *aggregation graph* before compressed data can even be generated, let alone forwarded to the sink. Moreover, since compression is done in a distributed manner, compression computations are not done at a central node but instead must be carried out in a particular order across multiple nodes. The order in which communications and computations are carried out also impacts the overall efficiency of a distributed compression scheme.

Finally, once data from a given node is fully compressed it is then routed to the sink along a **routing tree** using standard routing protocols like CTP [8]. Thus, an efficient routing tree is also necessary to forward compressed data to the sink. The generality of this abstraction is discussed as part of related work.

3. THE SENZIP ARCHITECTURE

We propose and detail SenZip, an architecture for distributed en-route compression in sensor networks. The primary goals of SenZip are are flexibility, modularity, and distributed configuration and reconfiguration. In addition to the lessons from the principles of an overall architecture for sensor networks and common abstraction identified for existing compression schemes, our design of the SenZip architecture is based on a system implementation effort.

3.1 SenZip Specification

The SenZip architecture specifies:

1. a *compression service* that can encompass different compression schemes and,
2. its *interactions* with standard routing and other networking services.

Figure 1 is a block diagram representation of the SenZip architecture. It needs to be emphasized that a system based on SenZip would be completely distributed and components shown in Figure 1 would reside on each

[1]A transmission schedule defines the order in which nodes transmit data

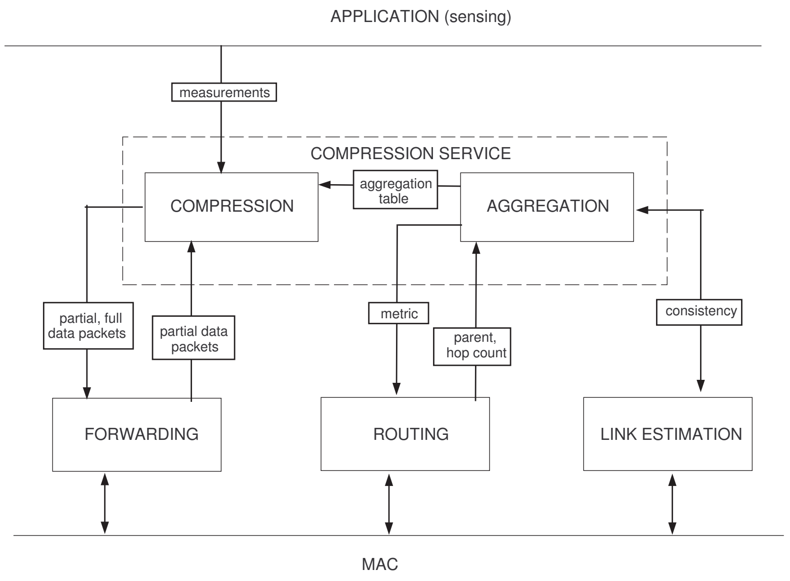


Figure 1: The SenZip architecture. A completely distributed compression service is enabled by having the interacting components shown here at each network node.

network node. Of course, compressed data from all nodes in the network finally reaches the base station where it is jointly reconstructed. We now describe the services, their responsibilities and interactions.

3.1.1 *Compression Service*

The compression service consists of the aggregation module and the compression module.

1. Aggregation module: The aggregation module disseminates and gathers information for maintaining the local aggregation tree by exchanging messages. This information is collated in an aggregation table. The aggregation graph abstraction allows the definition of a generic table that works for different compression schemes. Pseudo-code for such a table is shown in Figure 2.

2. Compression module: This module has the following functions: (a) From the aggregation tree structure provided by routing, this module obtains the role played by the node - which computations to perform and for which nodes, the parameters involved in computation and ordering information - the sequence in which nodes process and forward data. (b) It receives raw measurements from the application and packets with data that needs further processing from forwarding. (c) This module performs further processing over the partially processed data in storage and initiates processing for data of the node itself. The computations will be specific to the compression scheme and based on the role and parameter information. (d) Data that is still partially processed is packetized and sent to forwarding. For data that is fully processed, it checks if enough has been buffered in storage to fill a packet. If yes, performs quantization and bit reduction operations, and sends the packet to forwarding.

```
 struct attributes {
    int upstreamOnehopNeighborhoodSize;
    int downstreamOnehopNeighborhoodSize;
    :
} weight_attributes;

struct entry {
    int node_id;
    weight_attributes weights;
    int further_hops;
    tableEntry *neighborEntry[MAX_NHOOD_SIZE];
} tableEntry;

AggregationTable tableEntry[MAX_NHOOD_SIZE];
```

Figure 2: Aggregation table example. The recursive entry structure allows the same definition for different compression schemes.

3.1.2 *Networking components*

1. Routing engine: In addition to the standard rout-

ing functionality, this component in SenZip has an extra interface to the compression service. It reports information of path routing that is relevant for the local aggregation, for example, the parent and hop count in a tree topology. Optionally decisions on changing parent can be coordinated with the compression service, which can also provide a specific metric for the routing cost.

2. Forwarding engine: While partially processed data from nodes in the local aggregation tree is allowed to be intercepted by the compression service, fully processed data is forwarded directly along the route to the sink. Optionally, it might apply different settings, such as power, number of retries etc., for the different types of packets.

3. Link estimator: Efficient link estimation requires a limited choice of links to monitor [6]. To remove a link (or node in the neighbor table) that is part of the current aggregation tree, joint decision has to be made with the compression service to maintain consistency in the data processing.

3.2 Discussion

We emphasize that the configuration of roles, parameters and ordering is to be achieved purely locally from the aggregation graph and based on the compression scheme. There is no centralized decision and dissemination. This is a design criteria for compression schemes that can fit into the architecture. There is an overhead cost for the exchange of beacons to maintain the aggregation table. Whether the overhead is acceptable or not depends on the relative frequency of measurement versus the frequency of topology changes. If the frequency of topology changes is very high, the potential gains from compression might be overwhelmed by the cost of packet exchanges to maintain the table.

Which component is best suited for constructing and maintaining the local aggregation graph? One option is to give this additional responsibility to the routing engine, which already generates and receives messages to setup path routing. However, we believe it is much better for the compression service to handle the aggregation graph operations. This will aid code-reuse and flexibility by restricting the changes to the routing engine to providing a single extra interface.

To ensure flexibility and extensibility, important goals for an overall sensor network architecture [5, 14], SenZip only details the interactions between compression and networking services and not the interfaces. The components within the compression service also follow the larger goal of "meaningful separation of concerns". The abstraction helps avoid over-specification, by ensuring that the compression components are required by most existing schemes. Overall, the specification of SenZip has the features of a desirable *programming paradigm* described by Tavakoli et al. [14].

4. ALGORITHMS TO ARCHITECTURE

We now discuss two compression schemes that work over tree routing topologies - a simple differential encoding scheme, DPCM, and a more sophisticated 2D wavelet scheme developed by Shen and Ortega [13]. We describe how these schemes fit into the SenZip architecture.

4.1 Algorithm details

Assume a given graph $G(V, E)$ with vertices defined by node locations and edges defined by communication links between nodes. Assume a tree graph $T(V, R)$ $(R \subset E)$ rooted at a single sink node. Suppose every node is indexed by an integer $n \in V$, $\mathcal{C}_n$ is the set of child indices of n, and $\rho(n)$ is the parent index of n in T. We say that that node n has depth k when it is k-hops from the sink. Also let x_n denote the data measured at node n. For simplicity, we assume data is forwarded and compressed along the same tree T, i.e., the *aggregation graph* is T. In both schemes, we define the following transmission schedule. Initially, nodes without any children (*leaf* nodes) forward raw data to their parents in T. Then, every node n waits until it receives data from all children $m \in \mathcal{C}_n$ before it transmits its own data. This induces an *ordering* of the communications which is necessary for nodes to compress data as it is forwarded to the sink.

4.1.1 DPCM

Leaf nodes first forward raw data to their parents. Each node n waits to receive raw measurements from all its children in T and then computes residual prediction errors as differences from its own measurement as follows:

$$\begin{aligned} d_m &= x_m - x_n \quad \forall m \in \mathcal{C}_n \\ s_n &= x_n. \end{aligned} \tag{1}$$

Node n then forwards the compressed prediction residuals of its children (and other descendants) and its own raw measurements to its parent $\rho(n)$.

4.1.2 2D wavelet

This transform is constructed as follows for a single level of decomposition. First, vertices of G are assigned *roles* by being split into disjoint sets of predicts (odd depth) and updates (even depth) based on depth in T. Next, a high-pass "detail" coefficient d_m for each predict node m is computed by subtracting from the data at node m, x_m, a prediction that is based on information available at neighboring nodes (where neighbors are

table entry element	DPCM	2D wavelet
weight attributes	not needed	upstreamOnehopNeighborhoodSize ≡ number of children in tree downstreamOnehopNeighborhoodSize ≡ 1 (for parent in tree)
further_hops	1 (upstream only)	2 for upstream node, 1 for downstream
neighborEntry[].further_hops	0	1 upstream node, 0 for downstream

Table 1: Aggregation table initialization

defined as nodes that are 1-hop away in the aggregation graph):

$$d_m = x_m - \frac{1}{(|\mathcal{C}_m|+1)} \sum_{k \in \mathcal{C}_m} x_k - \frac{1}{(|\mathcal{C}_m|+1)} \cdot x_{\rho(m)} \quad (2)$$

Finally, a low-pass "smooth" coefficient s_n for each update node n is computed by adding to x_n a correction term based on the detail coefficients of neighboring nodes:

$$s_n = x_n + \frac{1}{2(|\mathcal{C}_n|+1)} \sum_{k \in \mathcal{C}_n} d_k + \frac{1}{2(|\mathcal{C}_n|+1)} \cdot d_{\rho(n)} \quad (3)$$

Under the given transmission schedule, each node only has access to data from its descendants and only forwards its own data and data from its descendants. Since each node n uses data from its parent, transform computations for n cannot be completed at n. However, note that terms corresponding to children $\mathcal{C}_m$ and parent $\rho(m)$ are explicitly separated in the computations. This allows us to compute *partial wavelet coefficients* and to update partial coefficients as data flows towards the sink to make them *full wavelet coefficients* as described in [3, 13].

This process is summarized as follows. Leaf nodes first forward raw data. Each predict node m waits to receive data from its children, then generates a partial coefficient $d_p(m)$ using data from its children as $d_p(m) = x_m - \frac{1}{(|\mathcal{C}_m|+1)} \sum_{k \in \mathcal{C}_m} x_k$. Then m forwards its partial $d_p(m)$ (and data from descendants) and $\rho(m)$ completes the computation as $d(m) = d_p(m) - \frac{1}{(|\mathcal{C}_m|+1)} \cdot x_{\rho(m)}$. Each update node performs similar operations. This process is illustrated in Figure 3. Note that this induces an *ordering* of the computations.

4.2 Relating algorithms to SenZip

We now describe the operation overview for SenZip based systems deploying the two algorithms.

4.2.1 Initialization

The aggregation component configures aggregation table entries and initiates message exchanges (with its neighbors) in order to gather information needed to build the aggregation table. The specifics of table entries for each scheme are shown in Table 1. This is

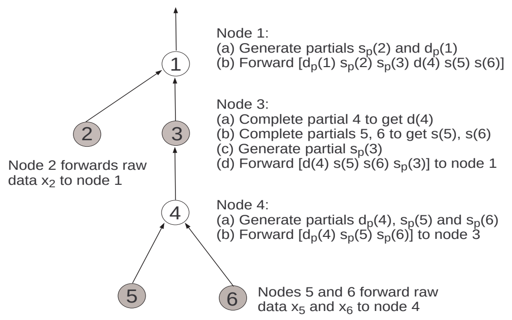

Node 1:
(a) Generate partials $s_p(2)$ and $d_p(1)$
(b) Forward [$d_p(1)$ $s_p(2)$ $s_p(3)$ d(4) s(5) s(6)]
Node 3:
(a) Complete partial 4 to get d(4)
(b) Complete partials 5, 6 to get s(5), s(6)
(c) Generate partial $s_p(3)$
(d) Forward [d(4) s(5) s(6) $s_p(3)$] to node 1
Node 2 forwards raw data x_2 to node 1
Node 4:
(a) Generate partials $d_p(4)$, $s_p(5)$ and $s_p(6)$
(b) Forward [$d_p(4)$ $s_p(5)$ $s_p(6)$] to node 3
Nodes 5 and 6 forward raw data x_5 and x_6 to node 4
1
2
3
4
5
6

Figure 3: Partial computations for 2D wavelet. Gray (white) circles denote even (odd) nodes. Operations at each node are done in the order listed.

shared with the compression component which can then identify their role, parent in the tree and children in the tree, and ordering of computations, to configure each compression scheme as follows:

DPCM: The roles are uniform i.e. all nodes have the same role. The ordering is that leaf nodes start forwarding and intermediate nodes wait for all one-hop upstream descendants (children) in aggregation tree.

2D wavelet: The roles are decided based on depth in tree from root, odd depth nodes are predicts nodes and even depth nodes are updates. The parameters in computation are equal to the weights, the number of one-hop (children) and two-hop (grandchildren) upstream descendants. The ordering is that leaf nodes start forwarding and intermediate nodes wait for partial coefficients of one-hop (children) and two-hop (grandchildren) upstream descendants in the aggregation tree.

4.2.2 Data forwarding and compression

DPCM: At each node n, the partially processed data to be received is raw data from children and that to be sent is raw data for node n and fully processed data of the children is the differentials according to Equation 1.

2D wavelet: At each node, partially processed data is raw data from children and grandchildren. Sent partially processed data is raw data for node n and all children, and fully processed data is the coefficients for all grandchildren according to Equations 2 and 3.

4.2.3 *Reconfiguration*

The routing engine informs aggregation component of a change in parent (and hop count) in the tree.

DPCM: When parent changes at node n, send an explicit parent change message to the old parent $\rho_{old}(n)$ and initiate a message to the new parent. When a parent change message is received by $\rho_{old}(n)$, remove child form table. The number of children is decremented, so waiting criteria in ordering changes.

2D wavelet: When parent of node n changes, send explicit delete message to ex-parent $\rho_{old}(n)$ and add message to new parent. If the hop count changes parity from before, propagate the change to all upstream nodes (descendants in subtree). When a parent change message is received by $\rho_{old}(n)$, remove child from table. The number of children and grandchildren is decremented, so waiting criteria in ordering changes. $\rho_{old}(n)$ sends a grandparent change message to $\rho(\rho_{old}(n))$ where changes in ordering are made.

5. SYSTEM IMPLEMENTATION

We now present results from the implementation of a TinyOS [16] system for distributed en-route compression based on the SenZip architecture. This implementation effort has informed the design of the SenZip architecture and in turn, concretely demonstrates it in software. The results provide proof of distributed configuration of compression, illustrate the flexibility for deploying different compression schemes within the same framework and demonstrate for the first time a sophisticated distributed compression scheme (based on 2D wavelets) for sensor networks.

5.1 Experimental Evaluation

An in-lab testbed with 15 Tmote Sky nodes is used for the evaluation. Ambient temperature is the sensed phenomenon and we introduce temperature gradients switching hot lamps on and off. The setting and topology are illustrated in Figure 4. Raw measurements use 16 bits, packet size is 28 bytes, 18 bytes available for data, i.e., 9 measurements, the time for 9 measurements defines an epoch, experiments last 24 epochs (216 samples) and a uniform bit allocation for all nodes is used.

The same experiments (sequence of switches) are repeated for the two different tree topologies in Figure 4 with different bit allocations per sample. To enforce a particular global tree, we bypass the link estimator and set $ETX = 1$ for intended parent and $ETX =$ inf for all other nodes. Also, the maximum power setting is used and packet losses are not considered.

1.*Distributed configuration of compression:* In these experiments, nodes were turned on and exchanged compression beacons and self-configured the roles, parameters and ordering according to the topology. Figures 5 (a) and (b) show the reconstruction at node 7 which has different depth and hence roles, in the two trees. Figures 5 (c) and (d) compare the reconstruction error at the nodes for each topology. The RMS error ranges between .01^{o}C to .16 ^{o}C over the temperature range of 20^{o}C to 28^{o}C for 3 bit quantization of coefficients for original sample of 16 bits. Since good and similar reconstruction is obtained, it is verified that the compression operations were correctly configured in a completely distributed manner.

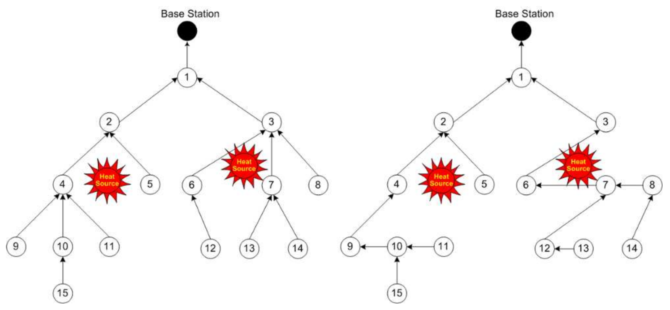


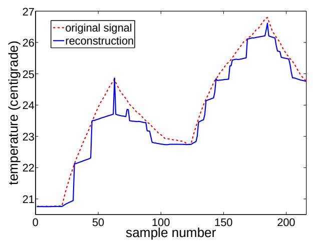
Figure 4: Experiment setting and two different routing trees, (a) tree 1 and (b) tree 2.

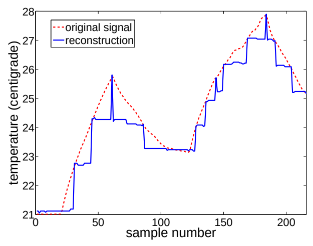
2. *Modularity and Flexibility:* We are able to successfully demonstrate correct operation and good reconstruction with the separate DPCM and 2D wavelet compression components over the same routing and forwarding components. This shows the modularity and flexibility of the implementation based on the SenZip architecture.

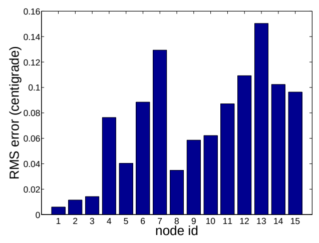
6. RELATED WORK

Work by Scaglione and Servetto was the first to explicitly consider the problem of joint routing and compression in sensor networks [11]. The correlated data gathering problem and the need for jointly optimizing the coding rate at nodes and routing structure were then established [4, 10]. Pattem *et al.* analyzed the relative performance of various routing and compression schemes based on using an empirically motivated model for the joint entropy as a function of inter-source distances [10] and showed that there exist efficient correlation independent routing structures. Zhu *et al.* have shown that under many network scenarios, a shortest path tree has performance that is comparable to an optimal correlation aware routing structure [22]. The work discussed above is mostly theoretical in nature and ignores details of how compression is actually achieved. We now discuss practical compression schemes and how they fit the abstraction described in this paper.

One particular class of methods only send trend data or data models within a given cluster. In Ken [1] and


(a) node 7, tree 1, 2 bits


(b) node7, tree2, 2 bits


(c) tree1, all nodes, 3 bits

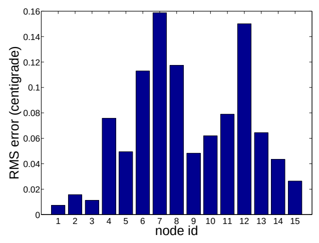

(d) tree2, all nodes, 3 bits

Figure 5: Node 7 has different roles in (a) tree 1, as predict node and (b) tree 2, as update node. Similar and good reconstruction performance for (c) tree 1 and (d) tree 2 verifies correct and distributed self-configuration of roles, parameters and ordering for 2D wavelet over different topologies.

PAQ [17] nodes are separated into clusters and assigned *roles* as cluster head or non-cluster head. Then, nodes forward data to cluster heads on some *aggregation graph*, model parameters for data in each cluster are estimated at cluster heads and only model parameters are forwarded to the sink along a *routing tree*. Note that an *ordering* of communications is implicit in this process. Another simple form of distributed data compression is differential encoding. For example, in DOSA [21], nodes are assigned *roles* as either correlating (C) or non-correlating (NC) nodes, NC nodes forward data to C nodes and C nodes compute and forward differentials of their NC neighbors. More sophisticated techniques have also been proposed [9]. Distributed computation of differentials must be done in a predefined *order* on an *aggregation graph*. Differentials are forwarded along a *routing tree*. A variety of distributed 2D wavelet transforms have also been developed [2, 3, 12, 13, 19]. The computations are defined by a given *role assignment* ("even" and "odd" nodes) on a *routing tree* (or *aggregation graph*) and an *ordering* of computations and communications is needed for unidirectional computations.

Culler *et al.* [5] advocate the need for an overall sensor network architecture. In a follow up paper by Tavakoli *et al.* [14], a set of design principles is proposed for the development of elements of the networking software architecture. In addition to the traditional goals of code reuse and interoperability, these include extensibility. This requirement arises in view of the relative immaturity of the field, where a rigid and complete modularization stifles innovation. They recommend a hybrid approach, with modularity for low level components (*underlying infrastructure*) and flexibility and extensibility at the higher layer (*programming paradigm*). The compression service in SenZip is viewed as a programming paradigm and the specification does not make strict definitions of modules and interfaces to allow for flexibility. The aggregation graph abstraction allows for newer compression schemes, providing extensibility.

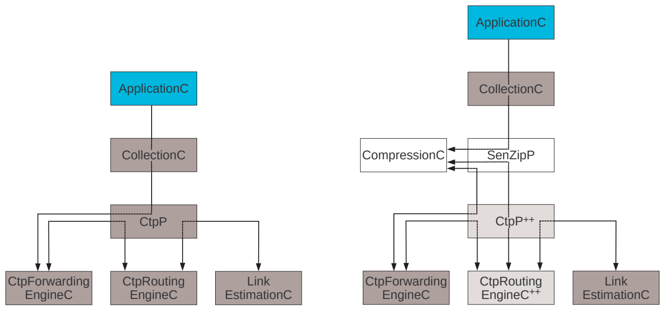


Figure 6: CTP extension to support a compression service.

7. CONCLUSIONS AND FUTURE WORK

We argue that distributed configuration is a key requirement for distributed compression in sensor networks. Motivated by this need, we propose the SenZip architecture, and present preliminary results from an ongoing system implementation based on it. These results indicate the potential of SenZip in developing reusable, flexible and extensible software for data gathering applications. Our current efforts are on extending and improving the implementation for a public software release. A critical milestone is to demonstrate that a SenZip system can achieve distributed reconfiguration in the face of network dynamics. Compression schemes simpler to reconfigure than the 2D wavelet are also being investigated. We are working on an extension of the Collection Tree Protocol (TinyOS proposal TEP 123 [8]) to include a modular compression service, along with a library of compression schemes that can be readily deployed. The code structure of CTP and the extension are illustrated in Figure 6. For further optimization of distributed compression, the system needs to include locally adaptive bit allocation, temporal and entropy coding. To be fully operational, the system needs to be integrated with existing synchronization, sleep scheduling and low power listening components.

8. REFERENCES

[1] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *IEEE Intl. Conf. on Data Engineering*, April 2006.

[2] A. Ciancio and A. Ortega. A distributed wavelet compression algorithm for wireless multihop sensor networks using lifting. In *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, March 2005.

[3] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari. Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm. In *Proc. of the ACM/IEEE Intl. Symp. on Information Processing in Sensor Networks*, April 2006.

[4] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *Proc. of the 23rd Conf. of the IEEE Comm. Society*, March 2004.

[5] D. Culler, P. Dutta, C. T. Eee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. Towards a sensor network architecture: Lowering the waistline. In *Proc. of the 10th Workshop on Hot Topics in Operating Systems*, June 2005.

[6] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four bit wireless link estimation. In *Proc. of the 6th Workshop on Hot Topics in Networks*, November 2007.

[7] GBROOS. Great barrier reef ocean observing system. `http://imos.org.au/gbroos.html/`.

[8] TinyOS 2.0 Network Protocol Working Group. Collection tree protocol, tinyos enhancement proposal (tep) 123. `http://www.tinyos.net/tinyos-2.x/doc/`.

[9] H. Luo, Y. Tong, and G. Pottie. A two-stage dpcm scheme for wireless sensor networks. In *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, April 2005.

[10] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Trans. on Sensor Networks*, 4(4), August 2008.

[11] A. Scaglione and S.D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *Proc. of The 8th ACM Intl. Conf. on Mobile Computing and Networking*, pages 140–147, August 2002.

[12] G. Shen and A. Ortega. Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting,. In *Proc. of the ACM/IEEE Intl. Symp. on Information Processing in Sensor Networks*, April 2008.

[13] G. Shen and A. Ortega. Optimized distributed 2D transforms for irregularly sampled sensor network grids using wavelet lifting. In *Proc. of the IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, April 2008.

[14] A. Tavakoli, P. Dutta, J. Jeong, S. Kim, J. Ortiz, P. Levis, and S. Shenker. A modular sensornet architecture: Past, present, and future directions. In *Proc. of the Intl. Workshop on Wireless Sensornet Architecture*, April 2007.

[15] A. Terzis, R. Musaloiu-E., J. Cogan, K. Szlavecz, A. Szalay, J. Gray, S. Ozer, M. Liang, J. Gupchup, and R. Burns. Wireless sensor networks for soil science. *Intl. Jrnl. on Sensor Networks on Environmental Sensor Networks*, January 2009.

[16] TinyOS. An operating system for wireless embedded sensor networks. `http://www.tinyos.net/`.

[17] D. Tulone and S. Madden. PAQ: Time series forecasting for approximate query answering in sensor networks. In *Proc. of the European Conf. in Wireless Sensor Networks*, February 2006.

[18] P. von Rickenbach and R. Wattenhofer. Gathering correlated data in sensor networks. In *Proc. of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, pages 60–66, October 2004.

[19] R. Wagner, R. Baraniuk, S. Du, D.B. Johnson, and A. Cohen. An architecture for distributed wavelet analysis and processing in sensor networks. In *Proc. of the ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks*, April 2006.

[20] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of the 7th Symp. on Operating Systems Design and Implementation*, December 2006.

[21] Y. Zhang, S. Chatterjea, and P. Havinga. Experiences with implementing a distributed and self-organizing scheduling algorithm for energy-efficient data gathering on a real-life sensor network platform. In *First IEEE Intl. Workshop on From Theory to Practice in Wireless Sensor Networks*, June 2007.

[22] Y. Zhu, K. Sundaresan, and R. Sivakumar. Practical limits on achievable energy improvements and useable delay tolerance in correlation aware data gathering in wireless sensor networks. In *Proc. of the 2nd IEEE Comm. Soc. Conf. on Sensor and Ad Hoc Communications and Networks*, September 2005.