

Data Funneling: Routing with Aggregation and Compression for Wireless Sensor Networks

Dragan Petrović, Rahul C. Shah, Kannan Ramchandran, Jan Rabaey

{dragan,rcshah,kannanr,jan}@eecs.Berkeley.edu

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley

Abstract—This paper considers the problem of minimizing the amount of communication needed to send readings from a set of sensors to a single destination in energy constrained wireless networks. Substantial gains can be obtained using packet aggregation techniques while routing. The proposed routing algorithm, called Data Funneling, allows the network to considerably reduce the amount of energy spent on communication setup and control, an important concern in low data-rate communication. This is achieved by sending only one data stream from a group of sensors to the destination instead of having an individual data stream from each sensor to the destination. Doing so also reduces the probability of packet collisions in the wireless medium because the same amount of information can be transmitted by having fewer nodes send longer packets. Additional gains can be realized by efficient compression of data. This is achieved by losslessly compressing the data by encoding information in the ordering of the sensors' packets. This "coding by ordering" scheme compresses data by suppressing certain readings and encoding their values in the ordering of the remaining packets. Using these techniques together can more than halve the energy spent in communication.

I. INTRODUCTION

There has been a lot of interest lately in the research community about sensor networks and the applications they enable. There is a multiplicity of scenarios in which such networks might find uses, such as environmental control in office buildings, the monitoring of seismic activity, robot control and guidance in automatic manufacturing environments, interactive toys, smart homes providing security, identification, and personalization, and interactive museums.

In particular, the PicoRadio project [1] at the Berkeley Wireless Research Center aims to build tiny meso-scale devices that are cheap enough to enable dense deployment with the aim of monitoring and controlling the environment. In order to be practical, these devices must be smaller than one cubic centimeter, weigh less than 100 grams, and cost substantially less than one dollar. Even more important, the nodes must use ultra-low power to eliminate frequent battery replacement. The power-dissipation level is targeted to be below 100 microwatts, as this would enable self-powered nodes using energy extracted from the environment, an approach called energy scavenging or harvesting.

Such aggressive power goals can only be achieved by optimizing all components of the PicoRadio system – protocols, radio designs and chip architectures. The

PicoRadio network consists of three kinds of nodes – sensors, controllers and actuators. The sensors, which are the most numerous, sense physical phenomena in the environment and generate readings based on those measurements. The controllers, which are fewer in number, observe the readings from multiple sensors. Based on these readings, they command the actuators to take certain actions to control the environment.

Due to the large number of sensor nodes, it is not practical to expect the nodes to be plugged into an electrical outlet or to have their batteries replaced often; therefore, it is crucial to minimize the amount of energy expended by the nodes. Energy consumption determines the lifetime of a sensor network, and communicating wirelessly consumes more power at the nodes than any other activity. Hence, it is crucial to design protocols so as to minimize the amount of communication required by the sensor nodes.

In this paper, two methods are discussed which improve the lifetime of energy constrained networks by reducing the amount of communication needed to send readings from a set of sensors to a controller. The first scheme is a packet aggregation technique while the second scheme performs data compression. While either of the two schemes can be used separately, using them together provides maximum gain. Both schemes use characteristics of the environment-sensing application. Designing application-aware communication layers can be very effective in reducing the power consumption at nodes, consequently increasing the network lifetime [2].

The motivation behind the work is the following. Packet headers, which include training sequences for transceiver clock synchronization, framing information, destination address, and error control codes, typically make up a large component of the sensor data packets, especially at low data rates such as those in environmental sensing applications. These headers are necessary to make communication in a multi-hop network possible, but they do not carry any information about the phenomenon being sensed. Since a large number of sensors periodically generate data and send it to only a few controllers in the system, great savings can be realized if these different sensor readings can be combined into a single packet; however, this has to be achieved using a robust, decentralized method with a minimum of control overhead and storage requirements. While these are obvious requirements, they are not easy to achieve in practice, as all

the sensors have to coordinate scheduling and routing their packets so that the data can be aggregated along the way.

The algorithm that is presented in this paper, called Data Funneling achieves that, leading to substantial increases in the network lifetime. The protocol is a reactive routing protocol that tries to manage energy usage of nodes to improve network lifetime. Other routing protocols that have been proposed for sensor networks include [3].

The second algorithm that enables further energy savings is data compression of the sensor readings at the aggregation points. Even if every sensor node individually compresses its data down to its entropy, further lossless compression can be achieved by taking advantage of the fact that the relative order in which those readings, generated at approximately the same time, reach the controller is unimportant to the application. Some of the sensor readings may, therefore, be omitted by having the values within those packets implied in the ordering of the readings that are explicitly transmitted to the controller.

The rest of the paper is organized as follows. Section II introduces the Data Funneling algorithm, which reduces the overhead due to packet headers in data packets and reduces the probability of packet collisions in the wireless medium. Simulation results that confirm the savings achievable due to funneling are presented in Section III. Section IV presents a method for further compressing the sensors' data at the aggregation points, and Section V concludes the paper.

II. DATA FUNNELING

The sensor networks envisioned by PicoRadio consist of a few controller nodes and many sensor nodes that periodically send their readings to the controllers. Since the controller nodes are required to have much greater computational and communication capabilities than the sensor nodes, the cost of controller nodes can be much greater than the cost of sensor nodes. Also, the controllers have to decide what actions to take based upon collated readings from a large region of space. For these reasons, there are many more sensor nodes than controller nodes, and each controller receives the readings of many sensors, while each sensor sends its data to only one or two controllers.

Further, the amount of data in each reading is low, at most a few bytes of light, temperature, acoustic, seismic or other measurements along with the time and place where those measurements were made. However, packet headers include training sequences for clock synchronization, framing information, destination address and error control codes and can be large relative to the packet size. Since many sensors report their data to the controller at approximately the same time and have similar headers, considerable savings can be realized by combining different packets into one large packet with a single header.

This combining can range from signal processing and/or source coding to a simple concatenation of the readings. This section describes a routing algorithm designed to perform dynamic clustering allowing *packet aggregation*. It can be used with any method of *data processing and*

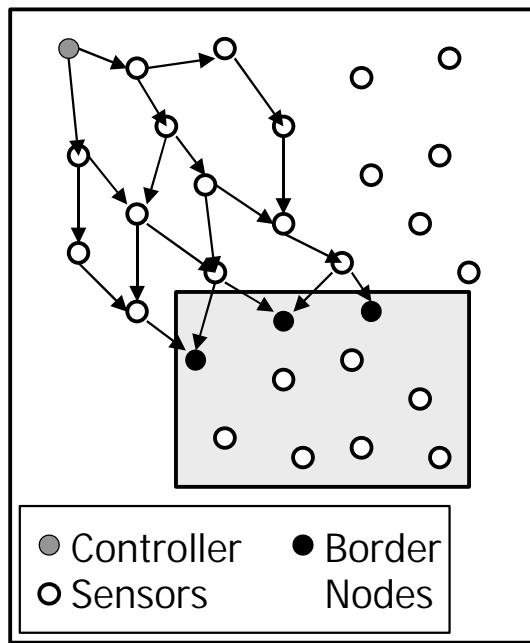


Figure 1. Control node, target region, directional flood and border nodes in Data Funneling.

compression and provides a significant reduction in energy consumption by the network by reducing the overhead of packet headers even when used with a simple concatenation of sensor readings. The algorithm also decreases the probability of packet collisions because it allows the same amount of information to be transmitted by fewer nodes sending larger packets, thereby reducing the number of nodes vying for access to the communication medium.

The main idea behind the algorithm, called Data Funneling, is the following. The controller breaks up the space into different regions (e.g. cuboids) and sends interest packets to each region, as shown in Figure 1. Upon receiving the interest packet, each node in the region will start periodically sending its readings back to the controller at an interval specified in the interest packet, usually every few minutes. Since many or all of the nodes within the region will be sending their readings back to the controller at the same time, it would be much more efficient to combine these readings into a single packet, so that only one packet with only one header travels from the region to the controller. The question is how can all these reading be collected at a single point and combined into a single packet.

The Data Funneling algorithm works as follows. The interest packets are sent toward the region using directional flooding. Each node that receives the interest packet checks if it is in the target region. If it is not, it computes its cost for communicating back to the controller, updates the cost field within the interest packet, and sends it on toward the specified region [3]. This is the directional flooding phase.

When a node that is in the target region receives the interest packet from a neighbor node that lies outside the target region, the directional flooding phase concludes. The node realizes that it is on the border of the region and designates

itself to be a border node as shown in Figure 1. Each border node computes its cost for communicating with the controller in the same manner as was done by the nodes outside the region during the directional flooding phase. It then floods the entire region with a modified version of the interest packet. The “cost to reach the controller” field is reset to zero and becomes the “cost to reach the border node field.” Within the region, each node only keeps track of its cost for communicating with the border node, not its cost for communicating with the controller. Intuitively, it is as if the border node becomes the controller of the specified region. It is at one of the border nodes that all the readings from within the region will be collated into a single packet.

In addition, two new fields are added to the modified interest packet. One field keeps track of the number of hops that have been traversed between the border node and the node currently processing the packet. The other field specifies the border node’s cost for communicating with the controller, and this field, once defined by the border node, does not change as the packet travels from one node to another.

Once the nodes within the region receive the modified interest packet from the border nodes, they will then route their readings to the controller via each of the border nodes in turn. Since there are several border nodes within the region, maximizing aggregation of sensor readings requires all the nodes within the region to agree to route their data via the same border node during every given round of reporting back to the controller. This is accomplished by having every node compute an identical schedule of which border node to use during each round of reporting. This is achieved by each node in the region applying the same deterministic function to the vector of costs to reach the controller seen by each border node. Since all the nodes apply the same function to the same inputs, they will all compute the same schedule, allowing them to collect all of their data at one border node during each round of reporting. The function used to compute the schedule can be similar to the function used to compute the probabilities for selecting different paths in probabilistic routing [3]. This allows border nodes with a low cost for communicating to the controller to be used more frequently than the ones with a high cost.

As data flows within the region from the sensors to the border nodes it can be aggregated along the way as shown in Figure 2. When the time comes to send a new round of observations back to the controller, the sensor nodes do not immediately start sending their packets. Instead, they wait an amount of time inversely proportional to their distance (in number of hops) to the border node that will be used in that round of reporting before sending their readings toward that border node. This allows the nodes that are far away from the border node to send their data earlier than the nodes that are closer to the border node. This way, nodes close to the border will first receive the readings from upstream nodes and bundle those readings with their own. In the end, all of

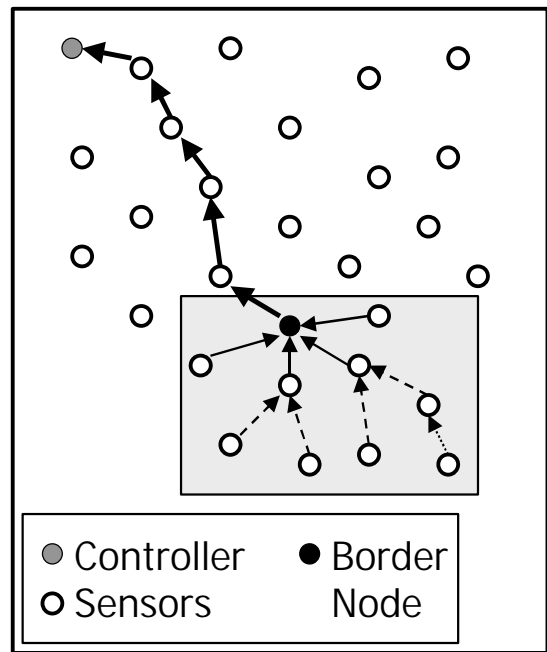


Figure 2. Funneling of data within the region and reporting back to the controller.

the data to be sent out by all the nodes within the region will be collated at one border node and sent back to the controller in a single packet as shown in Figure 2.

The protocol can be summarized as follows:

Setup Phase:

1. The controller divides the area it wishes to monitor into cuboids.
2. It then initiates a directional flood towards each region.
3. Each intermediate node records the cost of reaching the controller, as in [3].
4. When the packet reaches the region, the first node in the region that receives the packet designates itself as a border node.
5. The border node adds two new fields to the packet, the cost for reaching the border node and the number of hops to the border node. It then floods the region with this modified packet.
6. All nodes within the region receive packets from all the border nodes. Based on the energy required by each border node to reach the controller, they compute a schedule of border nodes where data is to be aggregated.

Data Communication Phase:

1. When a sensor has a data sample it needs to send back to the controller, it uses the schedule to figure out the border node to use in the current round of reporting.
2. It then waits for a time inversely proportional to the number of hops from the border node before sending out the packet.
3. Along the way to the border node, the data packets are joined together until they reach the border node.

4. The border node collects all the packets and then sends one packet with all the data back to the controller, using probabilistic routing.

Data Funneling creates clusters within the sensor network, but its advantage is that the clusters it creates have a dynamic hierarchy. There isn't a single cluster-head, whose failure can be devastating to the functionality of the network. Instead, the border nodes take turns acting as the cluster-head spreading out the responsibility and the load (i.e. energy consumption) among them. Also, the controller can redefine the regions into which its area of interest is divided thereby forcing the nodes to divide themselves into new clusters and elect new sets of border nodes. The controller can redefine the regions based on the data it receives from the nodes and/or the energy remaining in the nodes so as to ensure that nodes that have the greatest energy reserves act as border nodes.

III. SIMULATION RESULTS

To demonstrate its feasibility, the Data Funneling algorithm was implemented in the OpNet network simulator. The simulation was performed just for the network layer; lower layers were abstracted away. Energy consumed at all the nodes for transmission, reception and computation was measured. One sample topology is shown in Figure 3. The controller node, shown within the dark circle queried a region, shown as the large rectangle, containing 15 sensor nodes. Copies of the interest packet propagated toward the region, and the four nodes shown within the squares were determined to be the border nodes. Each of the sensors sent its readings to the controller every 10 seconds, and the packets were aggregated along the way. The simulation measured the number of sensor readings contained within each transmitted packet.

For the topology shown in Figure 3, the average number of sensor readings per transmitted packet was 7. This means that the energy expended by the network on transmitting packet headers was reduced by 86%. In general, the larger the region and the further away it is from the controller, the greater the savings due to funneling. If there are n sensors in the region, and the region is far away from the controller, the energy spent on transmitting headers will be reduced by a factor of approximately n .

Let α be the ratio of bits in a packet header to the total number of bits in a packet containing the header and a single sensor reading for a particular application, and let m be the average number of sensor readings per transmitted packet when Data Funneling is employed. Then, the total energy expended by the network on communication is reduced by $\alpha \times \frac{m-1}{m} \times 100$ percent due to Data Funneling if no compression of the sensor readings is done at the aggregation points. Performing compression on the sensor readings at the aggregation points within a region, as discussed in the next section, would result in even greater energy savings.



Figure 3. Sample topology for Data Funneling.

IV. CODING BY ORDERING

A. Intuition

The main idea behind ‘‘Coding by Ordering’’ is that when transmitting many unique pieces of data, and the order in which the data is sent is not important to the application (i.e. the transmitter may choose the order in which to send those pieces of data), then the choice of the order in which those pieces of data are sent can be used to convey additional information to the receiver. In fact it is possible to avoid explicitly transmitting some of those pieces of data, and use the ordering of the other information to convey the information contained in the pieces of data that were not sent.

Consider the case of the Data Funneling algorithm. In each round of reporting, the border node receives the packets containing sensor readings from n sensors in its region. It then places each node’s packet (containing the node ID, which may be just the node’s position, and payload) into a large super-packet containing the data of all the nodes and sends the super-packet to the controller. The border node has to include the ID of each node, which is unique, along with the node’s sensor reading so as to make it clear which payload corresponds to which node. Since all of the sensor readings from the region will reach the controller at the same time and the ordering of the packets within the super-packet does not affect the application, the border node has the freedom to choose the ordering of the packets within the super-packet. This allows the border node to choose to ‘‘suppress’’ some of the packets (i.e. choose not to include them in the super-packet), and order the other packets within the super-packet in such a way as to indicate the values contained within the suppressed packets.

For example, consider the case when there are four nodes with ID’s 1,2,3, and 4 in the region. Each of the four sensors generates an independent reading, which is a value from the set $\{0, \dots, 5\}$. The border node can choose to suppress the packet from node 4 and, instead, choose the appropriate ordering among the $3! = 6$ possible orderings of the packets from nodes 1, 2, and 3 to indicate the value generated by node 4. TABLE I shows a possible mapping from the permutations of three objects to the integers in the range $\{0, \dots, 5\}$. Note

that in this case the border node need not encode the ID of the suppressed node because that information can be derived from the fact that there are only four nodes and the packets of three of them were explicitly given in the super-packet.

What is of interest is to find out how many packets can be suppressed. Let n be the number of packets present at the encoder, k be the range of possible values generated by each sensor (e.g. if each sensor generates a 4-bit value, then $k = 2^4$), and d be the range of node ID's of the sensor nodes. Given n , k and d , what is the largest number of packets, l , that can be suppressed?

B. Achievable with Simple Codec

One strategy is to have the encoder (located at the border node) throw away any l packets and appropriately order the remaining $n-l$ packets to indicate what values were contained in the suppressed packets. A total of $(n-l)!$ values can be indexed by ordering $n-l$ distinct objects. Each of the suppressed packets contains a payload that can take on any of the k possible values and an ID, which can be any value from the set of d valid ID's except for the ones that belong to the packets included in the super packet. The values contained within the suppressed packets can be regarded as symbols from a $(d-n+l) \times k$ -ary alphabet, giving $(d-n+l)^l \times k^l$ possible values for the suppressed packets. In order for it to be possible to suppress l out of n packets in this manner, the following relationship must be satisfied:

$$(n-l)! \geq (d-n+l)^l k^l \quad (1)$$

Since this inequality cannot be converted into one in which l is expressed in terms of n , d and k , numerical methods must be used to compute which values of l satisfy the inequality; however, for large values of d and n , evaluating the relationship in (1) involves computing very large quantities that may exceed the precision of a computer. To alleviate this problem, Stirling's approximation, $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$ [11], may be used to convert (1) to the following equivalent inequality involving values of smaller magnitude:

TABLE I
MAPPING FROM PERMUTATIONS OF THREE OBJECTS TO INTEGERS.

| Permutation | Integer |
|-------------|---------|
| 123 | 0 |
| 132 | 1 |
| 213 | 2 |
| 231 | 3 |
| 312 | 4 |
| 321 | 5 |

$$(n-l)(\ln(n-l)-1) + 0.5 \ln(2\pi(n-l)) - l \ln k - l \ln(d-n+l) \geq 0 \quad (2)$$

If this inequality is satisfied, then it is possible to suppress l packets.

Rather than having to use a brute-force approach involving large lookup tables, one can use a computationally efficient algorithm for a bijection, a one-to-one and onto mapping, of integers from the range $\{0, \dots, n!-1\}$ to the possible permutations of n objects [12]. The mapping can be done by expressing any integer b in "base factorial" as

$$b = \sum_{i=1}^n a_i (i-1)! \quad (3)$$

where a_i is the number of objects lying to the right of the i^{th} object and smaller than the i^{th} object. The rightmost object is considered to be the 1st object, the one to its left is the 2nd object, and so on. The mapping shown in TABLE I is done according to this formula.

C. Achievable Rate

In the scheme proposed in Section IV.A, the suppressed packets are treated as if they can contain identical values. While their payloads may be identical, each packet has to have a unique ID. Since each packet has to be identified with a unique ID from among the d possible ID's, and $n-l$ of the possible ID's are taken up by the transmitted packets, there are $\binom{d-n+l}{l}$ possible combinations of ID's that the l suppressed packets can take on; therefore, when enumerating the possible values contained within the suppressed packets, the $(d-n+l)^l$ term in (1) should be replaced by $\binom{d-n+l}{l}$ giving

$$(n-l)! \geq \binom{d-n+l}{l} k^l \quad (4)$$

as the relationship that must be satisfied in order for it to be possible to suppress l out of n packets.

Again, this expression involves terms of high magnitude, but Stirling's approximation can be used to convert the inequality to the following equivalent relationship with more manageable terms:

$$\begin{aligned} & \ln(2\pi) + l + (l+0.5)\ln l + (n-l+0.5)\ln(n-l) \\ & + (d-n+0.5)\ln(d-n) \\ & - l \ln k - (d-n+l+0.5)\ln(d-n+l) - n \geq 0 \end{aligned} \quad (5)$$

| ID | Payload |
|----|---------|
| 1 | 3 |
| 2 | 5 |
| 3 | 0 |

Figure 4. Sample packet values that cannot achieve the bound.

The rate promised by (4) is achievable; however, it is not clear that this can be done with a simple algorithm (one that does not involve large look-up tables) at the encoder and decoder, as is the case with the scheme presented in Section IV.B. A low complexity encoding and decoding algorithm achieving this rate is the subject of future research.

D. Upper Bound

The two schemes presented above assume that the encoder will suppress l packets without giving much consideration to which l packets are suppressed; however, since the encoder has the freedom to choose which l packets to suppress, the number of values that may be indexed by dropping l out of n packets and ordering the remaining $n-l$ packets increases by a factor of $\binom{n}{l}$. Combining this with (4) gives the following relationship, which must be satisfied if it is to be possible to suppress l out of n packets:

$$\frac{n!}{l!} \geq \binom{d-n+l}{l} k^l \quad (6)$$

As before, applying Stirling's approximation and some manipulation can reduce the inequality to an equivalent one:

$$\begin{aligned} & (n+0.5)\ln n + (d-n+0.5)\ln(d-n) \\ & + 0.5\ln(2\pi) - d - 0.5\ln(d-n+l) \\ & + (d-n+l)[\ln(d-n+l) - 1] + l\ln k \geq 0 \end{aligned} \quad (7)$$

The condition in (6) and (7) is necessary, but it is not sufficient. Consider the case where $n=d=3$, and $k=6$.

Here, (6) is satisfied for $l=1$ because $\frac{3!}{1!} \geq \binom{1}{1} 6^1$; however,

depending on the values contained in the payloads of the packets, it is not always possible to suppress one of the three packets. If the packets contain values as shown in Figure 4 and the mapping is as in TABLE I, then it is not possible to suppress any of the packets because each packet's payload can be encoded only by suppressing one of the other packets. Therefore, the largest l that satisfies (6) is an upper bound on the number of packets

that can be suppressed, but the bound is not necessarily achievable. It may be possible that for large n the bound can be attained because there is more freedom to choose which packets are to be suppressed, but this issue remains the subject of future research.

The compression rates of the two achievable schemes as well the upper bound for $n \leq 100$, when $d=2^7$ and $k=2^4$, are shown in Figure 5. For example when $n=30$, using the low-complexity scheme allows the encoder to suppress $l=6$ packets, a 20% savings in energy spent on transmitting sensor data. The bound on the number of packets that can be suppressed at $n=30$ is 10. As n grows, the savings also increase. When $n=100$, the low-complexity scheme provides 32% savings, the higher-complexity scheme guarantees 44% savings, while the bound is 53%.

V. CONCLUSION

This work proposes a routing algorithm, called Data Funneling, for wireless sensor networks in which many sensor nodes have to communicate their data to a single controller node that does data-gathering and processing, makes decisions about how to influence the environment based on the sensor measurements, and possibly acts as an interface between the sensor network and the outside world by relaying the network's data to a human user or another network or device. The algorithm allows the sensor nodes to aggregate their data before sending it to the controller. This can reduce the amount of energy spent on communication setup and control by an order of magnitude because the sensor nodes need not all communicate with the controller individually. The scheme also reduces the probability of packet collisions in the wireless medium because, instead of having many nodes trying to communicate small amounts of data through the wireless medium, the scheme allows the same amount of information to be transmitted by many fewer nodes sending larger packets.

The aggregation of data also allows compression to be done on the sensors' data. Even if the sensors compress their data individually, further compression of 50% or more is possible due to the fact that the sensors generate their readings at approximately the same time, but there is freedom to choose the order in which these readings reach the controller. The compression savings due to the reduction of overhead imposed by communication maintenance and the savings due to the Coding by Ordering are orthogonal.

It is possible to do packet aggregation without doing further data compression at the aggregation points. Similarly, Coding by Ordering can be done in any communication scenario in which there are distinct pieces of data to be communicated, but the order in which they are communicated is unimportant. In the case of sensor networks such as that being developed by the PicoRadio project, it makes sense to take advantage of both of these types of gains.

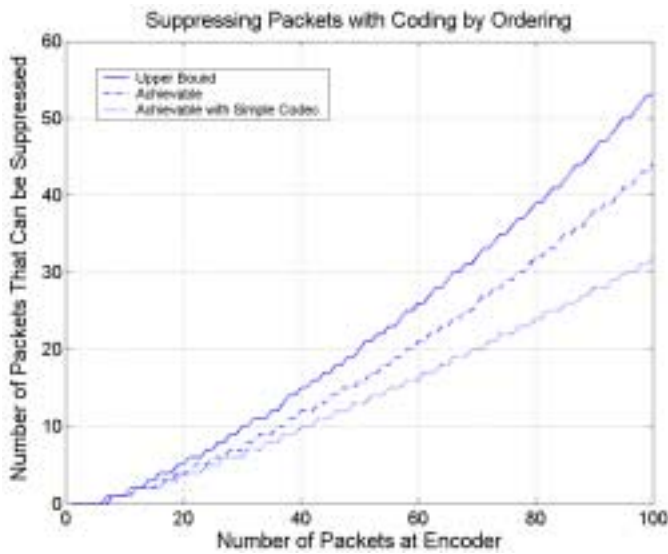


Figure 5. Compression rates for Coding by Ordering.

REFERENCES

- [1] Jan M. Rabaey et al., "PicoRadio supports ad hoc ultra-low power wireless networking", IEEE Computer, July 2000, pp. 42-48.
- [2] A. Goldsmith and S. Wicker, "Design challenges for energy-constrained ad hoc wireless networks", IEEE Wireless Communications Magazine, Aug.2002.
- [3] R. C. Shah and J. Rabaey, "Energy aware routing for low energy ad hoc sensor networks", IEEE Wireless Communications and Networking Conference, March 2002.
- [4] C. K. Toh, "Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks", IEEE Comm. Mag., June 2001, pp. 138-147.
- [5] S. Singh, M. Woo and C. S. Raghavendra, "Power aware routing in mobile ad hoc networks", IEEE/ACM MobiCom, Oct. 1998, pp. 181-190.
- [6] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A scalable and robust communication paradigm for sensor networks", IEEE/ ACM Mobicom, 2000, pp. 56-67.
- [7] E. Royer, C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks", IEEE Personal Comm., April 1999, pp. 46-55.
- [8] C. Perkins and E. Royer, "Ad hoc On demand distance vector routing", Proc. 2nd IEEE Wksp Mobile Comp. Sys. & Apps., Feb 1999.
- [9] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks", Mobile Computing, Kluwer, 1996, pp. 153-181.
- [10] J. Chang and L. Tassiulas, "Energy conserving routing in wireless ad hoc networks", IEEE Infocom, 2000, pp. 22-31.
- [11] T. Cormen et al., Introduction to Algorithms, The MIT Press, Cambridge, MA, 2002.
- [12] Sedgewick, R. "Permutation Generation Methods." *Comput. Surveys* 9, 137-164, 1977.