Reliable and Energy Efficient Transport Layer for Sensor Networks

Petar Djukic and Shahrokh Valaee
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto, 10 King's College Road, Toronto, ON, M5S 3G4, Canada
e-mail:{djukic,valaee}@comm.utoronto.ca

***Abstract*— We present Diversity Coded Directed Diffusion (DCDD), a reliable and energy efficient transport protocol for sensor networks. In DCDD, the sink uses a number of receivers—called "prongs"—that connect to it with reliable links. Sensors split observations into many fragments and generate parity fragments with an FEC algorithm. The fragments are then distributed over the paths and simultaneously sent to the sink. The sink can reconstruct the observations if it receives a portion of the fragments that is of the same size as their original observation.**

We use the ns-2 simulator to examine the ability of DCDD to increase end-to-end reliability, as well as the effect of DCDD on energy consumption in the network. Our simulations show that the network where DCDD is used outperforms the network in which the sensors use only MAC retransmissions to increase reliability. DCDD makes the energy use in the network more fair and at the same time it increases the end-to-end reliability in the network. DCDD also decreases the delay in the network.

***Index Terms*— Sensor networks, directed diffusion, fault tolerance, energy efficient protocols**

I. Introduction

Wireless sensor networks are used to observe natural phenomena such as seismological and weather conditions, collect data in battlefields, and monitor traffic in urban areas. The observed data is usually in the order of several bytes per observation and the receipt of every observation is not essential. In the future, sensor networks will be made of video and image enabled sensors. The observed data will be in the order of several kilobytes per observation and the correct reception of every piece of data will be essential.

In this paper, we consider a security system implemented with image enabled sensors. Sensors would be placed throughout a building and occasionally take pictures of their environment. If an incident is detected, the security management center (the sink) requests the sensors in the vicinity of the incident to send their most recent images and to start taking new images more frequently and report them at regular intervals. Once the incident is over, the sink requests that sensors stop sending their images.

The security system's network layer is implemented with directed diffusion [1]–[3] . Unlike the IP network layer, which routes datagrams based on unique node names, directed diffusion routes datagrams based on data naming. This type of routing is perfectly suited for our sensor network since the identification of sensors is not important, but the location and the content of observations is. However, directed diffusion delivers datagrams from the sensors to the sink with best effort and it does not take into account reliability or energy characteristics of sensor networks.

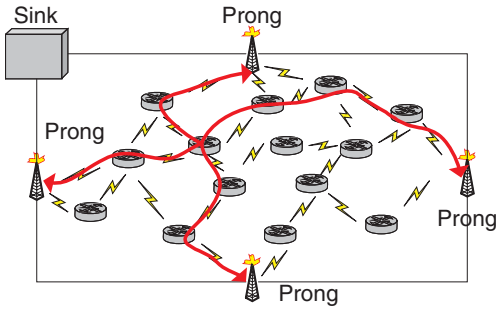


Fig. 1. Sensor Network with Multiple Prongs

It has been shown previously that radio links in sensor networks are extremely unreliable [4]. For example, [4] shows that even when sensor node distances are fixed, the range of packet loss can vary almost from close to 1 to close to 0. This makes it necessary to add a transport layer to directed diffusion, especially if the sensors are transmitting large observations split into smaller datagrams. In this case, it is essential for the sink to receive the complete sequence of datagrams to be able to retrieve the whole observation.

The traditional end-to-end transport layer approaches do not work in sensor networks since they rely on end-to-end and hop-by-hop retransmission for reliable delivery of large data. The end-to-end retransmissions make the system unscalable since the number of sensor in the network is very large and it may be impossible for the sink to track hundreds or thousands of connections from the reporting sensors. At the same time the hop-by-hop retransmissions are energy inefficient since wireless transmission is the biggest energy consumer on the sensors [5].

We propose diversity coded directed diffusion (DCDD) as a transport layer on top of directed diffusion. DCDD uses multiple network paths and forward erasure codes (FEC) to increase reliability and improve energy use in the network. In DCDD, the sink uses a number of receivers—called "prongs"—that connect to it with reliable links. The prongs are set sufficiently apart from each other, so that a sensor

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

can reach each prong through a path independent of the paths to other prongs (Fig. 1). Sensors split each observation into many fragments and generate parity fragments with an erasure code [6]. The fragments are then distributed over the paths and simultaneously sent to the sink. The sink can reconstruct the packet if it receives a portion of the fragments which is of the same size as the original observation.

We examine performance of DCDD with ns-2 simulations [7]. We show that DCDD increases the end-to-end reliability by about 10% compared to when the reliability is provided solely by retransmissions in the MAC layer, and that DCDD significantly lowers the delay in the network. We also show that DCDD introduces significant load-balancing of energy consumption in the network and therefore can increase the network lifetime.

A. Related Work

We first review two transport layer approaches designed specifically for sensor networks, and then we review diversity coded routing approaches. Reliable multi-segment transport (RMST) protocol for directed diffusion was proposed in [8]. The protocol uses a combination of end-to-end retransmissions, hop-by-hop retransmissions in the MAC layer, and caching on intermediate nodes to provide reliable delivery of large observations. The pump slowly, fetch quickly (PSFQ) protocol was designed for sensor networks where the MAC layer does not retransmit packets [9]. The protocol uses the transport layer to provide reliable end-to-end transmissions, making the hop-by-hop transmissions unecessary. The reliability is achived by letting intermediate nodes request retransmissions of packets from their upstream neighbours if loss of packets is detected. Both in the case of RMST and PSFQ the cost of energy is not taken into account. In contrast, our protocol is designed to load-balance the energy use in the network and decrease the need to retransmit packets lost in hop-by-hop transmissions.

Diversity coded routing (DCR) was first proposed in [10] for use in wired networks. The focus of the work was to analyze the decrease in delay due to load balancing (introduced with diversity coded routing). In [11] and [12] the authors investigate DCR for highly mobile wireless multihop networks. In that work, the authors present optimization algorithms for DCR that minimize packet loss.

We introduced diversity coded routing (DCR) for sensor networks in [13] and [14]. In that work, we proposed energy optimization algorithms that maximize network lifetime with a lower bound on reliability in the network. We showed, that it is possible to use diversity coded routing to increase network lifetime and reliability in the network. In the present work, we propose a protocol that implements DCR over directed diffusion, and show that indeed DCR is a good approach for sensor networks.

II. Diversity Coded Directed Diffusion

In this section, we describe how the DCDD layer works and suggest the changes necessary in directed diffusion to allow the use of multiple prongs and FEC. We start by reviewing directed diffusion and then describe how DCDD is added to the directed diffusion network stack.

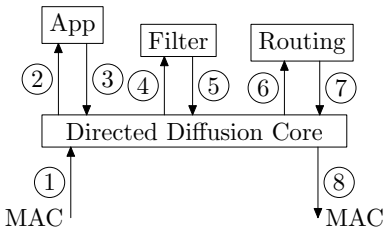


Fig. 2. Directed Diffusion Network Stack

A. Directed Diffusion

Directed diffusion uses named data and the `subscribe/publish` API to provide communication channels between sensors and sinks. On the sensor, the sensing tasks publish their data and on the sink the collecting applications subscribe to specific observations. Naming of the data is necessary so that directed diffusion can filter and steer the data to the sinks that subscribe to it. Together, named data and the `subscribe/publish` API allow the sensors and the sink to create a dynamic network stack [3]. We first describe how data is named and then how the network stack is built using the `subscribe/publish` API.

1) Named Data: All data observed by sensors is named with attribute-value pairs. An observation from our security system can be named as follows:

```
type        = data
task        = image
location    = [10, 20]
time        = 10:02:34
data        = blob.
```

The attributes of the data are on the left and the values of the attributes are on the right. The observed data are from the imaging task, at the location $x = 10$m and $y = 20$m, collected at 10:02:34 with the actual image stored as a blob.[1]

A similar scheme is used by the sink to name the data it is interested in:

```
type        = query
task        = image
rect        = [0, 50, 0, 50]
time        > 10:00:00.
```

The difference is that the sink specifies a range of locations and times rather than a specific location and time.

2) Subscribe/publish API: In addition to named data, directed diffusion uses a `subscribe/publish` API to allow sensors and sinks to build the network stack. A task (application) on a sensor registers for the data of the *query* type with the `publish` call and the sink registers for the

[1]We make the location attribute a coordinate to simplify exposition.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

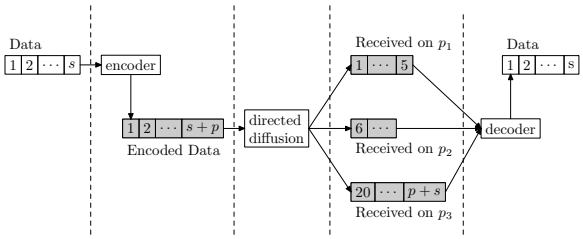


Fig. 3. DCDD Operation

TABLE I
SEGMENTID FIELDS

Field	Size (bits)	Description
`flowid`	8	Identifies the sensor
`seqid`	5	Identifies the observation
`maxid`	4	$s = 2^{(1+\text{maxid})}$
`inseqid`	15	Identifies a segment of the observation

data of the *data* type with the `subscribe` call. It is also possible to register filters, which can modify the passing data, so that the stack is more elaborate. Directed diffusion keeps track of the registered applications and filters and uses a callback mechanism to indicate if data matching the registered description is encountered.

We show the directed diffusion network stack in Figure 2. Data flow is shown as arrows to and from the directed diffusion core. Data can come from the application (entry point 3) or the MAC layer (entry point 1) and is then passed to applications and filters that match the data. The order in which data traverse applications and filters is established with a priority scheme, known before any publish or subscribe calls are made. The diffusion core matches the data to the descriptions provided with `subscribe`/`publish` calls and passes it to components that registered for that data.

The subscribe and publish calls also cause directed diffusion to send control traffic in the network [2]. In the case of routing protocols, the subscribe call causes interests (queries) to propagate through the network and set up gradients that aid with routing of the data back to the sink. If an interest reaches a sensor where a matching publish call was made the task that registered for the query is notified and the sensor can start reporting its data back to the sink.

The prototype of DCDD, presented in this paper, is tightly coupled within our sensing application. In the future, we may also implement DCDD as a filter that modifies the data stream into a FEC encoded stream, however since we only deal with one collecting task, that was not necessary for the prototype.

B. DCDD Additions

DCDD adds two new attributes to the directed diffusion data namespace, FEC encoder and decoder, and a protocol sensors and prongs use to start DCDD enabled communications.

We show the operation of DCDD in Figure 3. The application on the sensor generates an image which is passed as a blob of data to DCDD. DCDD splits the data into s segments and uses an FEC algorithm to encode the set into a new set of $s+p$ segments. The FEC allows the sink to reconstruct the original data if it receives at least s encoded segments. In our implementation, we use an XOR-based Reed-Solomon erasure code [6]. The encoder function is 72 lines of C code long and compiles to about 1Kb of Intel x86 binary code.

The encoded segments are passed to directed diffusion so they can be sent to the prongs (shown as p_1 to p_3 in the figure). Directed diffusion allows the use of multiple prongs, however if the protocol is used as is, diffusion would send all $s+p$ segments to each prong. This would add unnecessary redundancy to our protocol. To reduce the redundancy, DCDD needs to distribute the $s+p$ segments among multiple prongs and send only a portion of the segments to each prong.

We add `prongid` attribute to the directed diffusion data namespace so that DCDD can distribute the segments among the prongs. The new attribute is added to the queries so that interests from each prong are unique. For example, queries from prong 1 will be named as follows:

```
type        = query
task        = image
rect        = [0, 50, 0, 50]
time        > 10:00:00.
prongid     = 1.
```

Given this query, directed diffusion can set up gradients so that only the observations containing `prongid=1` is steered to prong 1, allowing DCDD to distribute the segments among the prongs.

Once the segment arrives at a prong, the prong forwards it to the sink via a reliable link. The sink keeps the cache of all the received segments and once it receives all s segments from the observation it can reconstruct the observation using the decoder function of the FEC algorithm.

We also add a `segmentid` attribute to the directed diffusion data namespace so that the sink can uniquely identify the segments sent by the sensors. We pack the `segmentid` as a 32-bit integer since this is the smallest attribute size in directed diffusion. The fields of the `segmentid` attribute and their sizes are given in Table I.

The first field, `flowid`, identifies a sensor's response to a query. This field allows the sink to receive 255 unique query responses at a time.[2] It can be made unique for each sensor by using a hash of the sensor's MAC address, or by assigning the `flowids` to the sensors before deployment in the field. The second field, `seqid`, is used to distinguish observations sent by the sensor. This field allows the sink to receive more than one observation at a time from each sensor. The third field, `maxid` indicates the number of fragments in the observation s. The fourth field, `inseqid` identifies the segment of the observation. This field takes values from 0 to $s+p$.

[2]This number can be easily increased by introducing a separate 32-bit field to the directed diffusion data namespace.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

At the beginning of the operation each sensor subscribes to queries coming from a list of predefined prongs. Once the sensor receives queries from its list of prongs it is able to start transmitting information back to the sink. The registration of prongs with the sensors is not necessary for successful operation of DCDD, however it allows us to control which prongs each sensor uses for communications with the sink. For example, the network can have hundreds of prongs, but each sensor can only communicate with a few of the prongs closest to its own location. In order to decrease the complexity, sensors do not need to register with the sink. The sink can identify the packets from each sensor with the `segmentid` attribute.

III. Simulation Results

We have implemented DCDD in the ns-2 simulator [7]. At the physical layer we have used 802.15.4 in the beacon-enabled mode. In this mode of operation, some of the sensors are full-function devices (FDDs) while others are reduced-function devices (RFDs). The FFDs transmit beacons used by the RFDs to synchronize. All data packet frames are transmitted using slotted CSMA-CA, and the MAC layer data transmissions can be acknowledged or unacknowledged. In the case of acknowledged transmissions, lost data packets are retransmitted twice before they are dropped at the sender. In all of the simulations, we have used a single hop packet loss rate of 0.01.

We used a network scenario with 100 sensors in an area $120\text{m}\times\ 120\text{m}$. The sensors were arranged in a grid topology, with distance chosen such that a sensor can only communicate with the sensors one hop away on the grid. The sink prongs were put in the four corners of the network to achieve the maximum network coverage, and the source of information are the twelve sensors in the middle of the topology.

We have extended directed diffusion to include `prongid` and `segmentid` attributes as described in Section II-B. We split observations into $s = 32$ fragments, where each fragment is 30 bytes long. We vary the number of parity fragments p from 0 to 24. The sensor transmits all $s + p$ fragments in $T_{\text{ON}} = 90$ seconds with interarrival times for each fragment exponentially distributed with equal means. After each transmission of $s + p$ fragments, the sensors sleep for an exponentially distributed time with the mean $T_{\text{OFF}} = 120$ seconds.

The sink collects the received fragments in its cache. If more than 32 fragments for a particular observation are collected, the sink reconstructs the observation. Otherwise, the fragments are collected for fixed amount of time and discarded after that time expires. The cache timeout is set to be smaller than the time sensors take to transmit $2^{\text{seqid}+1} - 1$ observations, so that the observations from a particular sensor can be uniquely identified with the `seqid` field.

A. Simulation Results

Table II shows the end-to-end reliability depending on the number of parity segments p and the number of prongs. The end-to-end reliability is defined as the number of recovered observations at the sink. The first column shows performance of a single prong solution with retransmissions. The other columns show performance of single or multiple prongs without retransmissions. First we observe that it is possible to increase the end-to-end reliability without retransmissions in the network. In fact, as the number p increases, the end-to-end reliability also increases. Second, as the number of prongs increases, the percentage of recovered observations also increases. We have shown this theoretically in [13] and [14].

TABLE II
End-to-end Reliability

	Number of Prongs			
	Acks	No Acks		
Parity p	1	1	4	8
0	78%	–	–	–
8	–	60%	63%	77%
16	–	83%	89%	90%
24	–	89%	89%	91%

TABLE III
End-to-end Delay

Number of Prongs			
Acks	No Acks		
1	1	4	8
25ms	20ms	18ms	15ms

We have also observed that the delay in delivering the observations is significantly lower when DCDD is used (Table III). The delay for the 1 prong using DCDD is lower than the delay 1 prong not using DCDD because there are no MAC layer acknowledgements when DCDD is used.

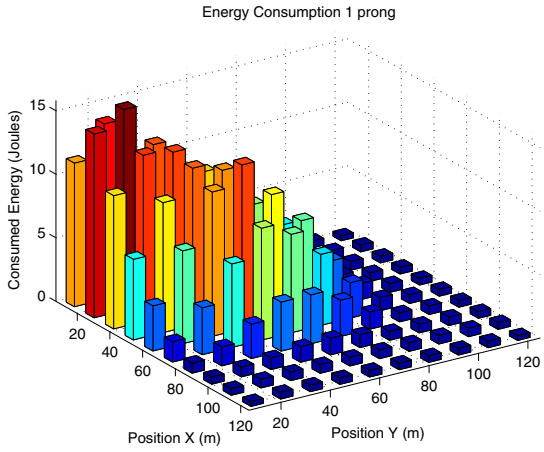
Fig. 4 shows how energy is consumed in the network. Fig. 4a shows energy consumption when only one prong is used with acknowledgments at the MAC layer and Fig. 4b shows the energy consumption when four prongs are used without any acknowledgments at the MAC layer. We first note that most of the energy in Fig. 4a is consumed by sensors which do not generate any information. Second, we note that the energy consumption is distributed more evenly when multiple prongs are used (Fig. 4b) and that the sensors consuming the most energy are the ones generating the observations. This shows that DCDD introduces load-balancing and fairness in the network.

Fig. 5 shows the percentage of energy used by the group of lowest n consumers. We see that in the case of one prong with acknowledgments 50% of the energy is consumed by the top 15 nodes, whereas in the case of four and eight prongs 50% of the energy is consumed by the top 38 nodes. This also shows that DCDD distributes energy consumption, which should increase the lifetime of the network.

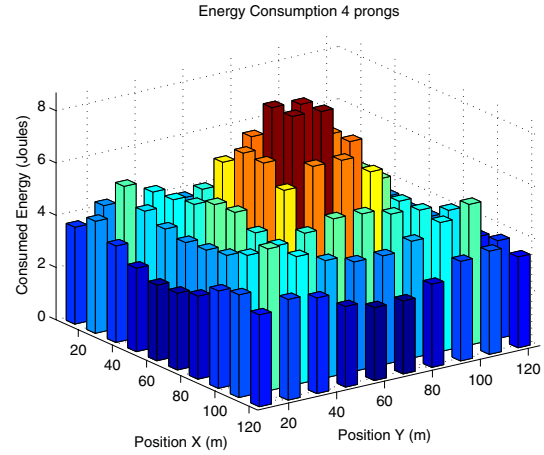
IV. Conclusions and Future Work

We have introduced DCDD, a reliable transport layer for sensor networks. DCDD runs transparently on top of directed

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.



(a) Energy Consumption 1 Sink with acknowledgments



(b) Energy Consumption 4 Sinks no acknowledgments $p = 16$

Fig. 4. Energy consumption

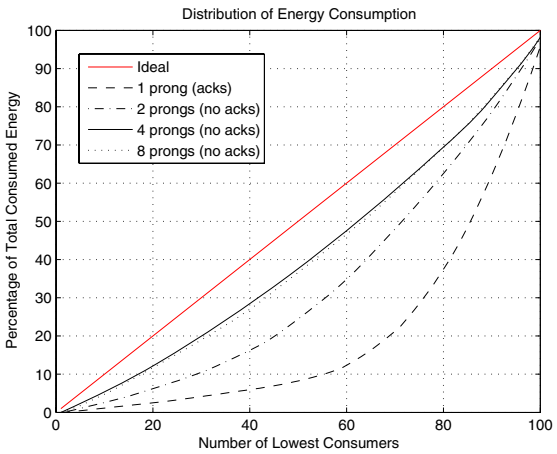


Fig. 5. Energy Distribution

diffusion and provides an increased end-to-end reliability compared to an approach that only relies on MAC retransmissions for increased reliability. DCDD also distributes energy use in the network, increasing the network lifetime.

We have used a very simple algorithm to distribute observation fragments among the prongs. However, we have shown in [13] and [14] that the performance of a scheme like DCDD can be improved significantly if more information is known about the paths in the network.

REFERENCES

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 2–16, 2003.

[2] F. Silva, J. Heidemann, R. Govindan, and D. Estrin, "Directed diffusion," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2004-586, January 2004, to appear in *Frontiers in Distributed Sensor Networks*, S. S. Iyengar and R. R. Brooks, eds.

[3] D. Coffin, D. V. Hook, R. Govidan, J. Heidemann, and F. Silva, "Network routing application programmer's interface (API) and walk through 8.0," 2001.

[4] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges for reliable multihop routing in sensor networks," in *SenSys'03*. ACM Press, 2003.

[5] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, March 2000.

[6] J. Blomer, M. Kalfane, R. Karp, M. Karpinski, M. Luby, and D. Zuckerman, "An XOR-based erasure-resilient coding scheme," Berkley, CA, Technical Report TR-95-048, 1995.

[7] "The network simulator - ns-2." [Online]. Available: http://www.isi.edu/nsnam/ns/

[8] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," in *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*. Anchorage, Alaska, USA: IEEE, April 2003, pp. 102–112.

[9] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: a reliable transport protocol for wireless sensor networks," in *Proceedings of the first ACM international workshop on Wireless sensor networks and applications*. ACM Press, 2002, pp. 1–11.

[10] N. F. Maxemchuk, "Dispersity routing," in *Proceedings of IEEE Internationl Communications Conferenece ICC'75*, Sna Francisco, CA, June 1975, pp. 41.10–41.13.

[11] A. Tsirigos and Z. J. Haas, "Analysis of multipath routing-part I: The effect on the packet delivery ratio," *IEEE Trans. Wireless Commun.*, vol. 3, no. 1, pp. 138–146, January 2004.

[12] ——, "Analysis of multipath routing, part 2: Mitigation of the effects of frequently changing network topologies," *IEEE Trans. Wireless Commun.*, vol. 3, no. 2, pp. 500–511, March 2004.

[13] P. Djukic and S. Valaee, "Minimum energy fault tolerant sensor networks," in *Globecom Workshops*, 2004, pp. 22–26.

[14] ——, "Maximum network lifetime in fault tolerant sensor networks," in *Globecom*, 2005.

1-4244-0357-X/06/$20.00 ©2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.