Summary on Directed Diffusion for Wireless Sensor Networking

Jussi Nikander

7th February 2005

1 Introduction

Distributed micro-sensing is an activity where a network of cheap, small nodes coordinate to achieve a larger sensing task. In "Directed Diffusion for Wireless Sensor Networking" [1] a new protocol for node communication called Directed Diffusion is introduced and evaluated. Since the nodes in a network doing micro-sensing tasks are typically small and have limited processing capabilities and power sources the article presents a protocol that aims for robustness, scaling energy–efficiency. In this paper, the main points of the Directed Diffusion paper are summed up. In addition, some critique at the paper is given.

To achieve these goals Intanagonwiwat, et. al. adopt an approach, where nodes require only knowledge of their immediate surroundings, and where all communication is data centric. There are no messages that are only for finding neighbour nodes or establishing network topology. Messages in the network are either announcements for *interest* for named data, or data messages generated by nodes that have sensed *interests*. The requirement for energy–efficiency requires nodes that have a very low energy dissipation when not sending or receiving data.

The original paper describes the Directed Diffusion method, gives an analytic evaluation for it, as well as describes the results of several simulations. Also, an implementation for several different platforms is discussed briefly.

The rest of this paper is organised as follows. The second section gives an overview of Directed Diffusion and briefly describes how it works. Then, the original paper's analysis on the method is summarised, and finally some critique is given.

2 Overview of the Directed Diffusion Method

There are four basic elements in Directed Diffusion. *Interest* messages are announcements by nodes of the network that they wish to receive certain type of

data. In essence, an interest is a named task description that is disseminated through the network to the relevant nodes. Interests create *gradients* in nodes that recieve a particular interest. A gradient describes the direction of data flow and tells a node where to forward data it receives. *Data* messages are replies to interests from nodes that are sensing the desired type of data. *Data* flows through the network according to gradients related to the *interests* that have requested particular data.. Typically data are descriptions of events sensed by a node. Finally *reinforcement* is done on a small number of gradient paths once data starts to flow to a destination, in order to decrease the load on the network and pull in data more efficiently.

2.1 Naming

In directed diffusion each data gathering task must be named. A name can be any identifier that describes the data gathering task adequately, and differentiates between different data gathering tasks. The method suggested in the original paper uses a list of name–value pairs to name the task. These pairs can be used describe what type of entities are to be sensed (e.g. wheeled vehicles in the original example, which describes a vehicle–tracing system), at what geographic area, for how long, at what intervals, etc.. This method has the advantage that the name clearly describes what the task originator is interested in, which some arbitrarily chosen name might not do. In the example, not all of the attribute–value pairs are part of the task name. The name can consist, for example, of the type and area fields. Other fields just give additional information. Each distinct task description defines an *interest*. If two nodes send *interest*s with identical names, the two interests are considered to be identical.

In addition to interests, data messages must also be named in order to allow nodes to connect a data message they recieve to an *interest* (or a number of *interests*). Without such connection data messages could not be transmitted through the network to the nodes that are interested in certain types of events. The further exploration of possible naming schemes was, however, deemed to be out of the scope of the Intanagonwiwat paper.

2.2 Interest Propagation and Gradients

When the user (the human operator) wishes to gather sensor data, he or she inserts an *interest* to the network. The interest is inserted at some, possibly arbitrary, node and propagates through the network to all relevant nodes. The node where the interest is inserted to the network is called a *sink*.

In order to insert an *interest* to the network, a *sink* broadcasts an interest message to its neighbour nodes. The neighbouring nodes forward the interest, until the whole network, or all relevant nodes, have received the interest. In the model described in the original paper, an interest is sent to all nodes in the network. More sophisticated ways to propagate interests are not discussed.

Each node in the network maintains a cache of active interests. In the cache each entry corresponds to a distinct interest. An entry in the cache includes the

interest, and information about previous hop (and not about the sink). If the node has received the same interest several times from different nodes, the cache entry has several previous hops associated with it. Each previous hop entry is a *gradient*, which tells where the node should forward received data messages. Each gradient includes also information about requested data rate (how often event data should be sent), and task duration (for how long the data should be gathered). The nature of gradients requires the nodes to be be able to identify their neighbours. Therefore, some locally unique naming scheme, for example 802.11 MAC addresses or Bluetooth, is required.

When a node receives an interest message, it compares it to the interests stored in the cache. If no match is found, a new cache entry is created from the received interest. The new cache entry is given one gradient, which points to the node where the interest message was received from. The event interval and task duration of the gradient are also instantiated from the interest. If a matching interest is found, the node checks if gradient to the sender node exists. If not, a new gradient is created. Finally the duration and data rate are updated, if required. A node may, after updating the cache, send the interest to some, or all, of its neighbour nodes. To all nodes that recieve this broadcast, the interest appears to originate from the sending node. Not all interests need to be resent, however. A node can, for example, decide to not send an interest if it recently sent a matching one.

It should be noted that interests are soft states. The sink must periodically re-send an interest message in order for the intermediate nodes to keep the interest alive. Also, the interests are originally *exploratory*, where the data gathering interval is rather large (for example one second). When an interesting event is first sensed, the sink can renew an interest with shorter interval in order to draw in real data. This mechanism is called path reinforcement.

2.3 Data Propagation and Path Reinforcement

When a node receives an *interest* that defines a data gathering task the node is required to participate in (since, for example, data to gathered from the area the node is in), it will turn on its sensors. Such a node is called a *source*. The source will then generate events at the highest requested event rate among all interests that it must satisfy. For example, if a source has two interests it must satisfy, one requires data with 10ms intervals and the other with 50ms intervals, the node will generate events once every 10ms. However, events that are part of the second interest are sent only every 50ms. After each event generation the source sends the data message to all those neighbours it has a *gradient* connected to an interest the data message is a reply to.

When a node receives a data message, it checks its interest cache for a matched interest. If no matching interests are found, the message is dropped. If an interest is found, the message is compared to the node's data cache. If the message is already in the node's cache, it is a duplicate message this node has already heard, and is dropped. A node can hear duplicate messages because of two reasons. First, two intermediate nodes can forward the same data message

from the same source, or two sources can sense identical event. If the data message isn't in the node's cache, it is added there and and resent according to the gradients.

If a node receives data faster than any gradient it has requires it to resend, it has several options. First, the node may just drop the "extra" packets that arrive between the required intervals. It can, however, also do some application specific data interpolation of the events, and forward the interpolated data (by, for example, selecting the most interesting event for forwarding).

Once a node detects a target that matches an active interest, it starts sending data. Through the gradients established in the net, the data diffuses towards the sink that originated the interest. When a target is first detected, the interests are *exploratory*, with low data rates. Therefore, in order to gain useful data, with short enough intervals, the sink can *reinforces* one neighbour by sending it – and only it – a new interest, which has a shorter data gathering interval but is otherwise identical to the original exploratory interest. Several different methods can be used to select the neighbour to reinforce. A simple, but useful method is to reinforce any neighbour from which the node receives a new data message (a message it hasn't seen before, that is).

When the neighbouring node receives the new interest, it checks its cache, and sees that it already has a matching interest and gradient towards the sink. It then updates the gradient, since the new interest has different event interval than before. If the new interval is shorter than that of any gradient the node has, it must then reinforce at least one neighbouring node in order to gain data often enough. The node can use its data cache to select the neighbour it wishes to reinforce. The node can use the same rules to reinforce its neighbours than what the sink uses.

It is also possible to negatively reinforce unwanted paths, which is required since the positive reinforcement mechanism may result in several paths bringing in the same data. The mechanism elaborated in the original paper uses explicit negative reinforcement messages. Such message is similar to a positive reinforcement message described above, but instead of shorter event interval (and thus higher data rate) it has a longer event interval, which will degrade the gradient. In the original paper, negative reinforcement will eventually degrade the gradient into an exploratory one.

As with positive reinforcement there are several different methods to choose how and when to negatively reinforce a gradient. The method chosen by Intanagonwiwat is to send negative reinforcement to those nodes that send no new information inside a certain time window.

There are other possible path reinforcement mechanisms, but those are ruled to be out of the scope of the original paper. Since the described mechanisms work with multiple sources and sinks, can be used to repair failed and degraded paths and remove loops in the network (by using negative reinforcement), this decision is justified since the described reinforcement mechanisms work.

3 Evaluation

To justify the usefulness of Directed Diffusion, the original paper compares it to two idealised alternate schemes, using both mathematical analysis and experiments with the ns2 [2] network simulator. The results are encouraging for Directed Diffusion.

The two idealised models to which Directed Diffusion is compared are flooding and Omniscient Multicast.

In flooding each message is sent to all nodes in the network. All sources send each of their events to every node in the network. Comparing Directed Diffusion to flooding works as a reality check. A data propagation scheme that is not as good as flooding cannot be considered viable.

In Omniscient Multicast each source has a shortest path multicast tree to all sinks that are listening to its events. Each message is sent once between two nodes of the multicast tree, and the scheme represents the best possible performance that an IP–based sensor network could achieve.

3.1 Mathematical Analysis

In the mathematical analysis the network is assumed to consist of N nodes in a square grid (therefore a side of the grid is $\sqrt{N}$ nodes). Each node in the network can communicate with its eight neighbouring nodes (excluding the nodes in the corners and the sides, which have less than eight neighbours). Sources are set at the left side of the grid and sinks at the right side. Both are placed at regular intervals, with the first source/sink being in the middle. There are n sources and m sinks, with d_n hops between sources and d_m hops between sinks. This leads to $\sqrt{N}$ being no less than $max(nd_n, md_m)$.

With the flooding scheme, the cost is defined as one unit for message transmission and one unit for message reception. When N, n, m, d_n and d_m are all known, the cost for flooding C_f can be calculated as

$$C_f = nN + 4n(\sqrt{N} - 1)(2\sqrt{N} - 1)$$

The transmission cost for flooding n events is nN, since each node transmits the event once. The cost for reception is therefore $2n$ times the number of links in the network. Asymptotically, this leads to $O(nN)$ total cost for flooding. This is asymptotically higher than the cost for the two other schemes.

For Omniscient Multicast the cost is the sum of the costs of the n multicast trees. If the cost of the tree from source j is denoted by $C(T_j)$, the cost to transmit event from there can be expressed as a function of $C(T_1)$, the cost to transmit from source 1, since $C(T_j) = C(T_1) + C(T_j - T_1) - C(T_1 - T_j)$. Here $C(T_j - C(T_k)$ is the cost of the tree constructed by removing from T_j the links that are also in T_k.

Furthermore, if cost of horizontal links in the network is denoted by $H(T_k)$ and diagonal links by $D(T_k)$, the total cost C_o can be expressed as

$$C_o = \sum_{j=1}^{n} (D(T_1) + H(T_j) + D(T_j - T_1) - D(T_1 - T_j))$$

For brevity how the different costs are calculated is left out of this summary. In the end, however, the result is that the asymptotic cost for Omniscient Multicast is $O(n\sqrt{N})$, when $m \ll \sqrt{N}$.

In the analysis of the Directed Diffusion method, the path constructed by the algorithm is assumed to be an "union" of the shortest path trees rooted at each source. The trees constructed by the Directed Diffusion method are very similar to those of Omniscient Multicast. The delivery costs are different, however, mostly because of data processing in the intermediate nodes. In Directed Diffusion the intermediate nodes are able to suppress a portion of data messages as duplicates.

The cost for Directed Diffusion is the union of all trees, or

$$C_d = C(UT_{1\to n}) = C(T_1) + \sum_{j=2}^{n} (H(T_j - UT_{1\to(j-1)}) + D(T_j - UT_{j-(j-1)}))$$

Similar to C_o, C_d is $O(n\sqrt{N}$ when $m \ll \sqrt{N}$. However, the total cost of C_d is less than C_o, since $D(T_1) - D(T_1 - T_j) \geq 0$ and $D(T_j - T_1) \geq D(T_j - UT_{j-(j-1)})$. Or, in words, the total cost of diagonal links in Omniscient Multicast is more than the cost of diagonal links in Directed Diffusion. The total cost of the horizontal links is the same, and therefore the total cost of Directed Diffusion is less than the total cost of Omniscient Multicast.

3.2 Simulation

To further evaluate the Directed Diffusion method and justify its usefulness the authors of the original paper performed a number of simulations using the ns2 network simulator. Again, Directed Diffusion was compared to flooding and Omniscient Multicast. In addition to this separate simulations were done where some aspects of the Directed Diffusion were evaluated.

Three metrics were used to compare the protocols:

- Average dissipated energy is the ratio of total dissipated energy per node in the network to the number of distinct events seen by the sink nodes.
- Average delay is the average one–way latency between sending an event and seeing it at each sink
- Distinct–event delivery ratio is the ratio of distinct events seen by sinks to the number of originally sent distinct events.

Five different network sizes were simulated, ranging from 50 nodes to 250 nodes, with 50 node increases in the size. Each node's radio had a range of 40 meters, with idle time power dissipation of 35mW, recieve power dissipation of 395mW and send dissipation of 660mW. Communication used (modified) 802.11 MAC layer.

The size of the network area depended on the number of nodes. For the 50 node case the size was 160mx160m, and the area was scaled up enough to keep average node density the same in all simulations.

In each simulation there were five source and five sink nodes. The sources were randomly selected from 70mx70m area, and sinks were uniformly scattered through the network. Each source node was configured to send two events per second. Interests were generated at five second intervals, and each interest had a duration of 15 seconds. Negative reinforcement time window was set to be two seconds. Each source node sent identical data to all interested sinks.

Number of simulations or the length of each simulation is not mentioned.

The summary of the results of the comparative simulations is as follows. First, flooding is clearly the most inefficient method, as it should be according to the mathematical analysis. It consumes more than twice as much energy, and has delays that are several times larger than the more efficient methods. The differences are more emphasised in larger networks. The poor performance of flooding is explained as a combination of high network traffic (all nodes transmit all messages they recieve) and the properties of the 802.11 MAC layer. The large energy consumption stems from the large number of transmissions, which consumes a lot of power both in the transmitting node and in all nodes that hear the transmission. Large average delay is a side–effect of the exponential back-off algorithm used in the MAC layer upon broadcast collision.

The average dissipated energy of Directed Diffusion is noticeably less than that of Omniscient multicast. The difference lessens, as network size grows, however. Intanagonwiwat et. al. state that Directed Diffusion uses only 60% of the energy Omniscient Multicast requires, but this figure is probably true only for the network with 50 nodes. The reason stated for the lower energy consumption is Directed Diffusion's ability to suppress duplicate data messages in the network, which causes less transmissions than in Omniscient Multicast.

Average delay for Directed Diffusion and Omniscient Multicast is is almost the same. In the original paper's figures Omniscient Multicast seems to have slightly lower average delays, but without seeing the actual results, it is impossible to say whether the difference is statistically significant.

The comparative evaluation was also repeated with radio power dissipations similar to that of AT&T Wavelan, with 1.6W transmission, 1.2W recieve and 1.15W idle time power consumption. This caused the differences between average power dissipation of the schemes to disappear, since the high idle time power consumption dominated all results.

In addition to the simulations comparing Directed Dissipation with the other schemes, a number of simulations were done to evaluate the various aspects of the Directed Diffusion.

One test simulated Directed Diffusion where parts (10% or 20% of the nodes) of the network were shut down for 30 seconds at a time. This caused some increase in average delay and worse distinct–event delivery ratio when compared to network with no node failures. However, the average dissipated energy was somewhat lower. The reason for lower energy consumption is stated to be the failure of some high–quality paths between sources and sinks. It should be noted that in this simulation all sources sent different data.

Another test investigated the effects of data suppression and aggregation in intermediate nodes. Here normal Directed Diffusion was compared to a variant where intermediate nodes were not allowed to suppress and aggregate data. This caused the average energy dissipation of the network rise to at least three times as much as in normal Directed Diffusion. The difference was largest in small networks (approximately five times more), but very significant even in a network of 250 nodes.

In the last simulation Directed Diffusion was compared to a variant where negative reinforcement was not implemented. Again, average energy dissipation raised to be over twice that of normal Directed Diffusion.

4 Critique

The Intanagonwiwat et. al. paper describes a novel method for transferring data through a wireless sensory network. For the most part, the work appears to be rather solid but, as always, there are shortcomings, and points that can be critiqued.

First, all the analysis ignores one fact of how the protocol is said to work. The negative reinforcement mechanism elaborated in the original paper degrades the unwanted data paths to exploratory path level. This feature is not, however, mentioned anywhere in the analysis. From all the analysis it seems that data is sent from sources to the sinks using only those paths that have been reinforced. Whether the data sent through exploratory gradients has any real impact in the efficiency of the Directed Diffusion method is doubtful, but it not a minor detail that can be completely ignored in the analysis.

Also, the mathematical analysis of the different schemes is rather brief in all, and a lot of the mathematics it contains are not really explained, leaving the reader to figure out the rather complex equations without much support. This causes the mathematical analysis to be very hard to comprehend for anyone, who is not very good at mathematics. Especially if the analysis of Omniscient Multicast and Directed Diffusion had contained some more explanation on the computations involved, the reader might have had a chance to gain some intuition on the calculations. Now, in order to affirm the results presented in the paper, a lot of work is required. An alternative is to take the results as they are, but this leaves the reader in doubt whether the results are correct or not.

Furthermore, in the mathematical analysis, the original paper includes several graphs, which show how the data delivery costs of Directed Diffusion and Omniscient Multicast grow, when a) number of sinks increases and b) number of

sources increases. However, when the parameters of the graphs are examined, in case a) the number of sources (which is constant) is set to 6 and in case b) the number of sinks (which is constant) is set to 10. These are the maximum values compared when the said parameter is a variable. As the performance of Directed Diffusion becomes ever better compared to Omniscient Multicast when the number of sinks/sources increases, the graphs apparently present a worst case for Omniscient Multicast. It would have been interesting to see the figures when number of sinks or sources is a smaller constant. Interestingly, when network sizes are compared, the number of sinks and sources is both 5.

Moreover, in the simulation where the different schemes are compared, all source nodes send identical data. This is a very good case for Directed Diffusion, since it can suppress the maximum amount of event transmissions in intermediate nodes. This probably gives the Directed Diffusion an edge in the simulation, since it seems rather far–fetched that all source nodes would send identical data all the time. This casts a doubt on whether Directed Diffusion really has as low average dissipated energy as the simulation suggests. Perhaps a more realistic comparison would have been a case where the sources send different data.

Last, the paper seems to imply that Directed Diffusion is usable even when the nodes are mobile since some features of the method are specifically said to be usable in an immobile network. There paper does not specifically mention applicability to mobile networks, however.

5 Conclusions

In addition to the parts covered in this paper, Intanagonwiwat et. al. briefly describe their implementation for Directed Diffusion. The implementation part of the original paper gives an overview of the two API (network and filter) the team has implemented for several platforms.

All in all, the paper describes an interesting method for data dissemination in sensor networks. The mathematical analysis and simulation results seem to indicate that the scheme can be used in real–world applications where low energy consumption is required.

References

[1] C. Intanagonwiwat, R. Givindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trasactions on Networking*, 11(1), 2003.

[2] T. V. Project. Network simulator – ns–2. http://www.isi.edu/nsnam/ns/.