

# A comparison of epidemic algorithms in wireless sensor networks <sup>☆</sup>

Mert Akdere <sup>a</sup>, Cemal Çağatay Bilgin <sup>a</sup>, Ozan Gerdaneri <sup>a</sup>, Ibrahim Korpeoglu <sup>a,\*</sup>,  
Özgür Ulusoy <sup>a</sup>, Uğur Çetintemel <sup>b</sup>

<sup>a</sup> Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey

<sup>b</sup> Department of Computer Science, Brown University, Providence, RI 02912, USA

Available online 3 March 2006

## Abstract

We consider the problem of reliable data dissemination in the context of wireless sensor networks. For some application scenarios, reliable data dissemination to all nodes is necessary for propagating code updates, queries, and other sensitive information in wireless sensor networks. Epidemic algorithms are a natural approach for reliable distribution of information in such ad hoc, decentralized, and dynamic environments. In this paper we show the applicability of epidemic algorithms in the context of wireless sensor environments, and provide a comparative performance analysis of the three variants of epidemic algorithms in terms of message delivery rate, average message latency, and messaging overhead on the network.

© 2006 Elsevier B.V. All rights reserved.

**Keywords:** Wireless sensor networks; Epidemic algorithms; Data dissemination; Performance evaluation

## 1. Introduction

Recent advances in digital and analog electronics and wireless radio communications have enabled wireless sensor networks to be used for many interesting and new applications such as real-time remote monitoring and control, military surveillance, environmental monitoring, healthcare management, and construction safety. The realization of wireless sensor networks, however, poses a lot of challenges in system and network design, algorithm and protocol design, and query language and database design. One such challenge in wireless sensor networks is efficient and effective dissemination of information to all or a group of sensor nodes in the network. This is not an easy task since the number of nodes in a sensor network can be quite huge and the environment is dynamic, i.e., nodes can die or move, and thus topology can change constantly. Also

depending on the application, the information to be disseminated can originate at a single node, such as the base station, or at multiple nodes, such as sensor nodes themselves.

A class of algorithms, called *epidemic algorithms*, have been successfully used in disseminating information in the context of various other systems. These systems are usually decentralized and distributed systems such as replicated databases. The distributed and decentralized nature of wireless sensor networks implies that the epidemic algorithms can also be applied in the context of sensor networks. However, unlike the powerful computing devices of today, sensor nodes in a sensor network are much restricted in terms of computation power and communication capability.

The goal of this paper is to investigate the use and performance of some classes of epidemic algorithms in the context of wireless sensor networks for a specific application scenario where all nodes are interested in receiving information messages originating from sensor nodes. Our evaluations are based on simulations, and we have used the TOSSIM sensor network simulator for this purpose. TOSSIM is an event-driven simulation tool; an

<sup>☆</sup> This work is partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) with grant numbers EEEAG-103E014, 104E028, and 105E065.

\* Corresponding author.

E-mail address: [korpe@cs.bilkent.edu.tr](mailto:korpe@cs.bilkent.edu.tr) (I. Korpeoglu).

application code written in the TOSSIM environment can also work on real sensor nodes running TinyOS operating system.

In a TOSSIM environment, we implemented three different types of epidemic algorithms: a pull-based algorithm, a push-based algorithm, and a pull–push based algorithm. We did extensive simulation experiments to compare the performance of these three types of epidemic algorithms. The performance metrics that we have used for this comparison include message delivery rate, average message latency, and the number of redundant messages seen in the network.

The paper is organized as follows. In Section 2, various forms of epidemic algorithms are explained and discussed briefly. In Section 3, we outline the existing approaches to dissemination of information in sensor networks. In Section 4 we provide our methodology and in Section 5 we provide our simulation results and the related discussions. Finally, in Section 6 we give our conclusions.

## 2. Epidemic algorithms

Epidemic algorithms [1] are related to the theory of epidemics, which describes a theoretical model for the spread of diseases. Epidemic algorithms follow the model of nature to spread information and define simple rules for information to flow between nodes of a network. An *epidemic round* (or cycle) is defined as the interval of time during which two nodes exchange information. In each epidemic round, a node chooses a communication partner. In our study the choice of communication partner is done randomly from the set of neighboring nodes.

Epidemic algorithms can be differentiated from each other by their style of communication between neighboring nodes. Following are the three main styles of communication used by epidemic algorithms:

- *Pull based epidemic algorithms.* A node asks a selected neighbor for new information. The node will receive new information only if the neighbor has new information.
- *Push based epidemic algorithms.* A node with new information sends the information to a selected neighbor.
- *Pull–push based epidemic algorithms.* These algorithms are a combination of the two models described above. A node employing such an algorithm sends information to a selected neighbor when it has some information available; it also asks and receives new information from the selected neighbor if the neighbor has new information.

The information to be disseminated from a node is first stored at a cache in the node. Even though the information has been transmitted to a neighbor, it stays there for a while before getting deleted. Each class of epidemic algorithms can further be categorized with respect to the scheme used in deciding when to delete the information from the cache. The following are some schemes that can be used for this purpose:

- *Blind deletion scheme.* A node deletes the information message from its message cache based only on its internal node state and based on a predetermined probability of deletion.
- *Counter scheme.* A node deletes an information message after  $k$  rounds. The parameter  $k$  is the value of the counter maintained at a sensor node.
- *Coin scheme.* A node deletes information with a probability of  $1/k$  in each epidemic round. The parameter  $k$  is the value of the counter maintained at a sensor node and is decreased by one at every round.

## 3. Related work

Epidemic algorithms have been used in various applications and environments to disseminate information in a robust manner. These applications include maintenance of replicated databases [1], updating code changes [2], discovering resources [3], and achieving ad hoc routing [4]. There is also data dissemination problem in the context of computer networks and a work in this area is the Network News Transfer Protocol [5]. Similar studies in wireless sensor networks area include SPIN [6] and directed diffusion [7].

Some techniques to disseminate data in the context of replicated databases for providing consistency are discussed in the work Epidemic Algorithms for Replicated Database Maintenance [1]. These techniques include direct mailing, anti-entropy, and rumor mongering. In direct mailing, any change in a database is separately mailed to all other databases. Obviously, this is a simple method but it is inefficient in terms of the traffic incurred over the network. In the anti-entropy technique, a database chooses another database regularly to exchange data. This method requires resolving the differences between two databases and is therefore costly. In rumor mongering, when a database receives an update for the first time, the update becomes a “hot rumor”. A database tends to deliver hot rumors as much as it can. If a hot rumor is known by most other databases, the algorithm stops spreading it. The anti-entropy method was implemented on the Xerox Corporate Network and resulted in impressive performance improvements, both in maintaining consistency and in reducing network traffic overhead [1].

Trickle [2] is a mechanism designed for propagating and maintaining code updates in wireless sensor networks. In a sensor network using this scheme, tasks of sensor nodes are assigned through code updates, and all nodes in the network will have the same code to execute. A new code is propagated hop by hop to all nodes in the network. Since wireless sensor nodes have limited energy, maintenance costs of the code updates must be low. Another requirement is rapid propagation of updates, because some tasks may have to be activated as soon as possible and newly assigned tasks make the older ones obsolete. The update process should also be scalable and should work in a dynamic environment, which is the case for sensor net-

works. The Trickle mechanism addresses all these issues by self-regulation and by observing code changes using metadata information. The metadata can be considered as the code summary of each sensor node, and the summary is exchanged regularly by using a technique called polite gossiping.

In ad hoc networks, since the topology may constantly change, flooding and its variants can be a good alternative for routing data messages from a single node to some other nodes, or for routing route discovery messages that are required during the route discovery phase of many ad hoc routing protocols like DSR [8] and AODV [9]. Gossiping [4] is also an alternative data dissemination strategy for this environment. In gossiping, nodes decide whether to forward a data message or not by looking to the result of tossing a coin with a predetermined probability.

Directed diffusion [7] is one of the data dissemination methods that is proposed for use in sensor networks. Directed diffusion is an example of data-centric routing where the content stored in a packet determines the route of the packet. The content stored in a packet is represented as a sequence of attribute–value pairs. A node that is interested in receiving data with some attributes indicates its interest to the network and this interest is disseminated. During this process, reverse paths are also set up. These reverse paths are then used to direct the information from the sensor nodes having the desired data towards the nodes interested in the information.

Another work on data dissemination on sensor networks is sensor protocols for information via negotiation (SPIN) [6]. It is an adaptive protocol and it tries to overcome some problems of flooding and gossiping based protocols. The main problems associated with flooding and gossiping are: implosion (i.e., a node is forced to get information twice from two different nodes), overlapping (i.e., a node is forced to get some subset of the information twice from two different nodes because of their overlapping regions), and resource blindness (i.e., nodes do not adopt energy saving schemes) [6]. SPIN uses data descriptors called meta-data for negotiation. These descriptors are smaller in size than actual data. In this way the overhead on the network is reduced both in terms of bandwidth and energy consumption.

The studies mentioned above are concerned with information dissemination in ad hoc and dynamic environments and in providing some schemes for efficient and effective dissemination of information in these kind of environments. There are also studies about the performance evaluation of the suggested schemes. A study similar to our work was conducted by Ganesan et al. [10]. In that study, empirical analysis of *flooding* is done for large scale multi-hop wireless sensor networks. The work also shows the effects of various layers of the protocol stack on the complexities of flooding as an epidemic algorithm. For this purpose, the authors define useful metrics for each layer. The metrics include providing some statistics on packet loss, effective communication range and link asymmetry for

physical and link layers, contention rate, collision ratio and latency for the MAC layer. The structures of routing trees constructed as a result of flooding technique are examined to analyze their effects on network and application layer performance. The results have implications for algorithm design and measurement. One of the conclusions states that simple protocols like flooding may incur unanticipated complexity due to the complex physical world. Another conclusion states that it is useful to make vertically integrated measurements to understand the contribution of the physical, medium access control, and application layers to application behavior. An interesting result also explains that the designed algorithms should use a probabilistic abstraction to model the connectivity in an ad hoc network. The last important conclusion is that protocols must be robust to asymmetry, because asymmetry is an expected issue in those environments. The work presented in [10] focuses on a simple epidemic algorithm, flooding. However, in this paper we are focusing on the performance of other epidemic algorithms.

#### 4. Methodology and simulation model

We have evaluated the use of different types of epidemic algorithms in wireless sensor networks through simulations. For this purpose, we have used the TOSSIM simulator [11,12]. We have implemented the epidemic routing protocols in nesC, which is the programming language of TinyOS [13].

TOSSIM is a bit-level simulator for TinyOS applications and it can simulate the entire TinyOS network stack. TinyOS [14] is an operating system designed for wireless sensor nodes. TOSSIM can compile directly from the TinyOS application code and in this way can enable code written for real sensor nodes to be used for simulations as well. We also used TinyOS's scripting language, *Tython* [15], during the development and running of our simulations. *Tython* was helpful in setting up the test scenarios.

We have used the following metrics in comparing three types of epidemic algorithms: message delivery rate, message latency, and messaging overhead. We have observed the values of these metrics depending on the values of the following factors: maximum transmission range of sensor nodes, number of sensor nodes, buffer size in a sensor node, and epidemic interval length.

The scenario that we have used in our simulation experiments consists of a grid, covered with wireless sensor nodes. The grid structure used is a square with length of one side equal the square-root of the number of nodes used in the experiment. There is at most one node placed per grid cell. Each cell in the grid is a  $10 \times 10$  unit<sup>2</sup>. Periodically, a randomly selected sensor node generates an information message to be disseminated to all other nodes in the network. We assume that only one node at a time generates new information. We also assume that all nodes in the network are interested in receiving the new information.

We have involved three different transmission ranges for the sensor nodes in our simulation experiments: 10, 15, and 20 units. We have used the disc radio model provided by TinyOS in defining the range of a sensor node. In this model, a mote receives messages from and sends messages to other motes within its communication range without any error. The transmission range is specified as the radius of this disc. With a transmission range of 10 units and a grid topology of  $10 \times 10$  for each cell, only the adjacent nodes hear each other. With a transmission range of 15 units, the nodes at the corners of each other can also hear each other. That is, with a radius of 10 units a sensor node has a maximum of four neighbors (left, right, up, and down), whereas with a radius of 15 units a sensor node has a maximum of eight neighbors. We have used the default MAC layer protocol in TOSSIM, CSMA, which introduced message drops due to collisions during the simulations.

In our application scenario, a randomly selected sensor node generates a new information message every 6 s. A new information message is generated periodically between 18 and 120 s of the simulation lifetime. When the cache of a node is full and a new message arrives at the node (or the node generates a new message), one of the existing messages in the cache is deleted and the new message is put into this place.

Unless otherwise stated, we have used the *counter based* cache item deletion scheme with a counter of 4, an epidemic interval of 2048 ms, a buffer size of 8 messages, and a neighbor cache size of 15 entries. In the neighbor cache, we keep information about the neighboring nodes. Neighbor cache provides a list of neighbors from which a communication partner can be selected. The information in a neighbor cache entry includes the id of neighbor node, and some other node specific information. The epidemic interval specifies the length of one round of information exchange in the epidemic algorithm. In every epidemic round, each node selects a random time at which it executes the epidemic algorithm by pulling or pushing data from its neighbor nodes.

Our scenario, of course, is a scenario that can be valid for some type of applications, but definitely not for all. There are certainly other sensor network application scenarios where not all nodes but only a small subset of them are interested in a data item generated by a sensor node. There are also application scenarios where more than one node can generate information messages at the same time. For this paper we have focused on a single application scenario and evaluated the three epidemic algorithms for this single sample scenario. We basically aim to study the relative performance of these algorithms in a sensor network for the application scenario that we have used in this paper.

## 5. Simulation results

In the experiments, we first observed the delivery rate and message latency for the three types of epidemic algorithms over the course of the simulation period. We define the *delivery rate* as the percentage of nodes that received an

information message. In finding this value, we take the average over all information messages disseminated in the network during a given time interval.

The *average message latency* is defined as the average amount of time between the start of disseminating a data item and its arrival at a node interested in receiving the data. Since some messages may not arrive at some nodes, their latency is set at infinity and we do not consider those messages in the calculation of the average message latency.

We compute the values of these metrics in a scenario where the number of nodes is 50 and the transmission range of nodes is 15 units. Fig. 1 shows the observed delivery rate of the three types of algorithms with respect to the simulation time, and Fig. 2 displays the average message latency of the algorithms.

The results show that the push-based epidemic algorithm starts the dissemination faster than the pull-based algorithm. This is understandable since we have a single source injecting a new packet to the network at a given time. As time passes, however, the pull based algorithm reaches a faster delivery rate than the pull–push based algorithm and close to the delivery rate of the push based algorithm. When enough simulation time has passed and the delivery rate becomes stabilized, we see that the push based algorithm performs slightly better than the pull based algorithm but the difference is not very significant.

Fig. 1 also shows that the delivery rate of the pull–push based algorithm is much less than to the delivery rate of the pull and push based algorithms. We think that this result is due to the buffer management strategy used during a message exchange operation in the pull–push based algorithm. In the pull phase of the pull–push based algorithm, the messages that will be sent in the push phase may be overwritten because of buffer overflow. This causes a reduction in the delivery rate of the pull–push based algorithm.

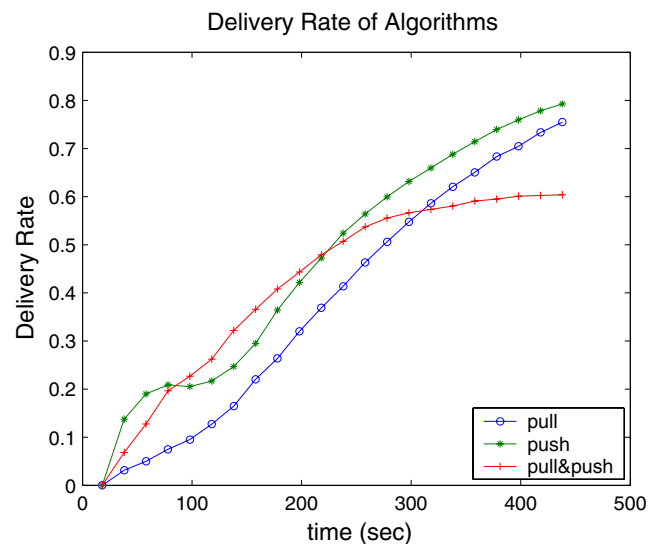


Fig. 1. Delivery rate of three different type of epidemic algorithms: pull based, push based, and pull–push based.

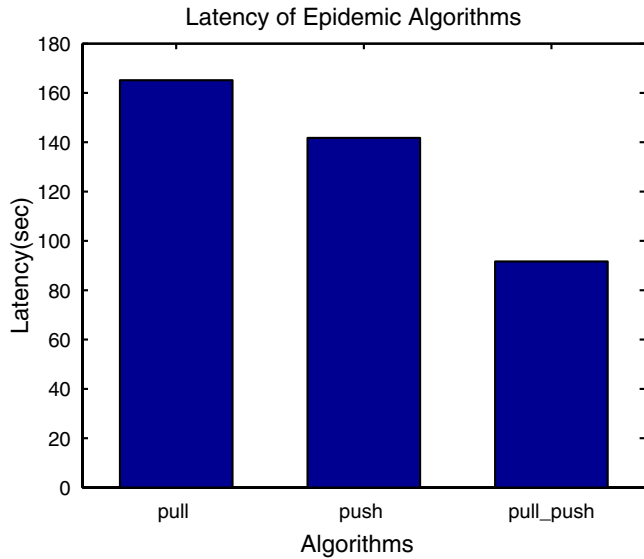


Fig. 2. Message latency of the three different types of epidemic algorithms.

The delivery rate of messages, however, is not 100% for any of the algorithms, as opposed to the observation of Vahdat and Becker [16]. They report a 100% delivery rate since they assume a buffer size of 2000 and dissemination of a total of 1980 messages. Under this situation, no deletion is required.

Fig. 2 shows the average message latency of the algorithms. The figure shows that the latency of the pull–push based algorithm is the lowest; the latency values should be interpreted together with the delivery rates. The pull and push algorithms have high delivery rates and it is reasonable for them to have high latencies since they distribute information for a longer time. However, the pull–push based algorithm exhibits a lower delivery rate which means that it stops disseminating information quicker. As a result, the observed latencies are lower for the pull–push based algorithm. In addition, the pull–push based algorithm can disseminate information faster since both parties in communication can send and receive information at the same time. The figure also shows that the push-based algorithm has lower latency than the pull-based algorithm. This is also expected since the push-based algorithm can exchange data immediately when data is available, and in this way available data can reach neighbors and other nodes more quickly.

We have also investigated the reaction of the algorithms to the size of the network. For this experiment, we have set the transmission range of each node to 15 units and changed the size of the network to 10, 20, and 50 nodes. The nodes are again placed on a grid. Figs. 3–5 display the observed delivery rate for each of these algorithms for different network sizes. We can see that as the number of nodes increases, both pull based and push based algorithms attain similar delivery rates, whereas the pull–push based algorithm’s delivery rate is much less. This suggests that for some type of applications using the scenario we have defined in this paper, pull-based and push-based algo-

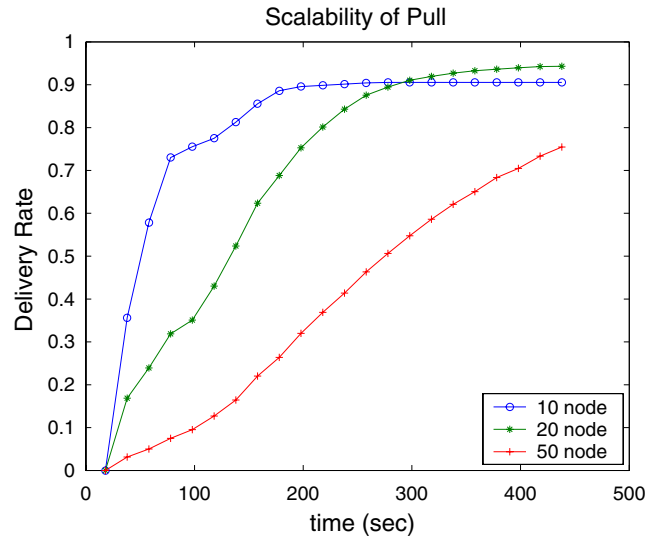


Fig. 3. Delivery rate of pull based algorithm with respect to the number of nodes in the network.

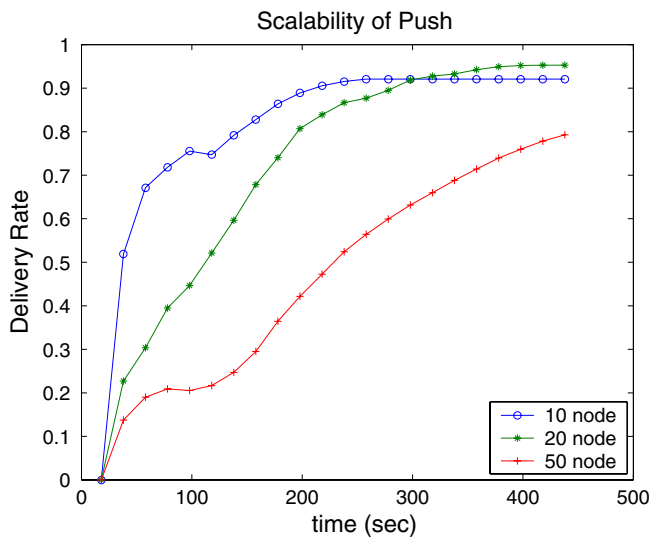


Fig. 4. Delivery rate of push based algorithm with respect to the number of nodes in the network.

rithms can perform better when the number of nodes is increased in a sensor network. In other words, they are more scalable to the size of the network.

It is also important to investigate the bandwidth overhead of the epidemic algorithms over the network they are running. The bandwidth overhead can be due to various reasons: large packet headers, retransmissions, flooding, or control messages required during the exchange of data between nodes, etc.

The epidemic algorithms we use in the paper first resolve the differences between the data available at two nodes that would like to exchange data, and then the data is exchanged. In this way, a node does not receive an information message that it already has. This reduces the num-



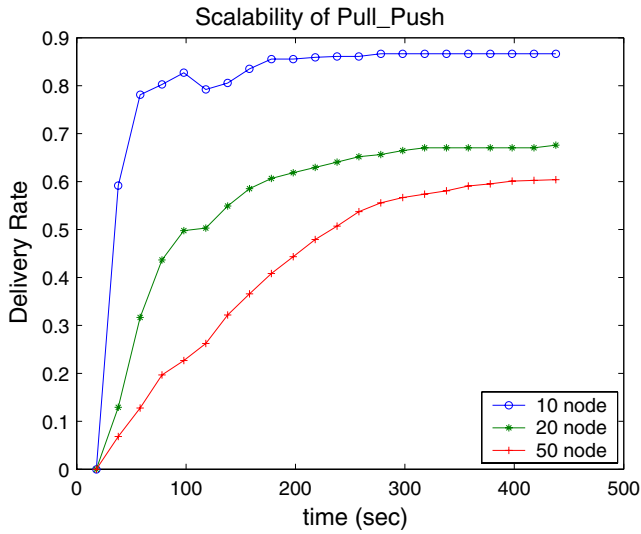


Fig. 5. Delivery rate of pull–push based algorithm with respect to the number of nodes in the network.

ber of redundant data messages in the network. But there is still bandwidth overhead due to the control messages used in epidemic rounds, while data messages are exchanged between neighbors. For example, the pull based and pull–push based algorithms periodically check for a new data message from a neighboring node even if they are fully synchronized.

Therefore, rather than relating the redundancy in the network to the unnecessary data transmissions, we relate the redundancy to the total number of messages transmitted in the network. Fig. 6 presents the overhead observed for various transmission ranges for 50 nodes. We see that

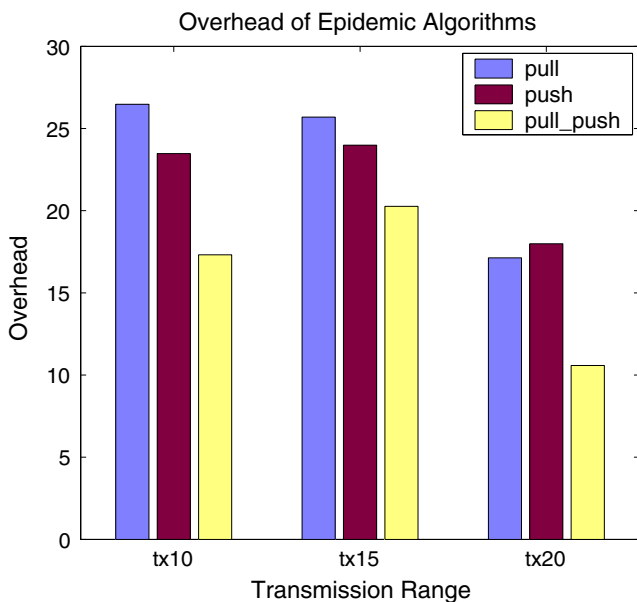


Fig. 6. Overhead of pull based, push based, pull–push based epidemic algorithms in terms of number of redundant messages they cause to be transmitted in the network.

the pull–push based algorithm has the least overhead compared to the others. This is because after resolving the difference, the pull–push based algorithm can send and receive more data messages. In pull or push based algorithms, however, data messages flow only in one direction. This behavior allows the pull–push based algorithm to synchronize faster than the others, and therefore the overhead of the pull–push based algorithm is less than the others. From the same figure we also see that the push based algorithm performs better than the pull based algorithm in our application scenario since the push based algorithm does not have to periodically poll neighbors for new message availability. The pull based algorithm on the other hand periodically checks to see if there is any new data available from its neighbors and therefore introduces more overhead.

We also wanted to see the impact of transmission range on the delivery rate of the algorithms. Fig. 7 shows the delivery rate of the algorithms for three different transmission ranges: 10, 15, and 20 units. Again the scenario involves 50 nodes deployed on a grid.

We observed that the delivery rate of each epidemic algorithm improves as the transmission range increases. For push based epidemic algorithms, for example, the delivery rates are 73%, 79%, and 89% for transmission ranges of 10, 15, and 20 units, respectively. As the transmission range increases, the connectivity of a node (i.e., the number of its neighbors) increases as well. And as the number of neighbors of a node increases, the delivery rate also increases. This is because the probability of a node receiving a message is higher when there are more neighbors. However, the contention in the network and the power consumption also increase with increasing transmission range. As a result, transmission range should be optimized with respect to the application scenarios.

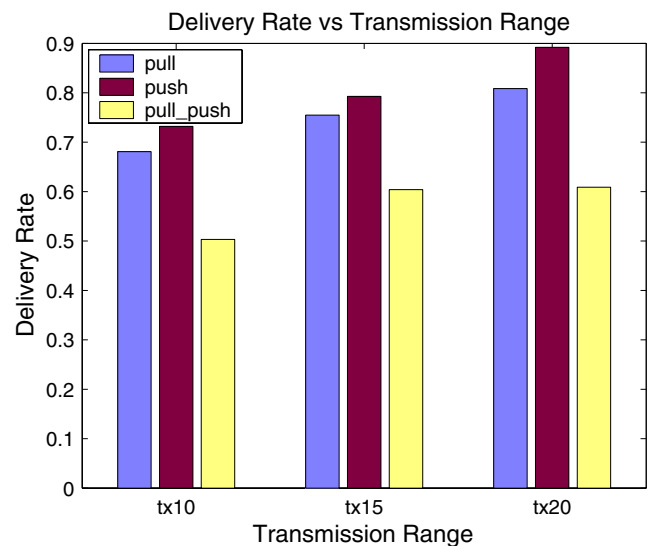


Fig. 7. Effect of transmission range on delivery rate of three types of epidemic algorithms.

We have also investigated the effect of buffer size on the performance of the epidemic algorithms. We do not expect sensor nodes to have much memory and therefore the buffer capacity is a very precious resource; it can affect performance in terms of the delivery rate, since a message may have to be deleted when a buffer becomes full. To observe the effect of buffer size, we have taken measurements for three different buffer sizes: 4, 8, and 10. We have fixed the transmission range at 15 units, and the number of nodes in the network has been set to 20.

As can be seen in Fig. 8, we have a better delivery rate for all of the algorithms as the buffer size increases. However, the difference is more pronounced for the pull based and push based algorithms.

We have observed that there is also a relationship between the buffer size and the length of the epidemic interval. This observation has been obtained with the push based epidemic algorithm. In Fig. 9, results for the push-based epidemic algorithm with 4 and 8 message buffer sizes are provided for the following epidemic interval lengths: 1024, 2048, 4096, and 6144 ms. We have observed that with a buffer size of 8, the maximum delivery rate is 94% at 2048 ms interval length. However, we have had a maximum delivery rate of 70% with a buffer size of 4 at 1024 ms. We can draw the following conclusions from the results of these experiments. First, with a fixed buffer size, epidemic interval length has a specific value for which the delivery rate is maximum. For example, with a buffer size of 4, 1024 ms is the optimum epidemic interval length. Second, with larger buffer sizes, the optimum value of epidemic interval length is larger. For example with a buffer size of 8, the value is 2048. This means that if we want to extend the epidemic interval length to consume less power, and at the same time maintain a high delivery rate, we might extend the buffer size. Therefore, we can conclude that with a larger buffer

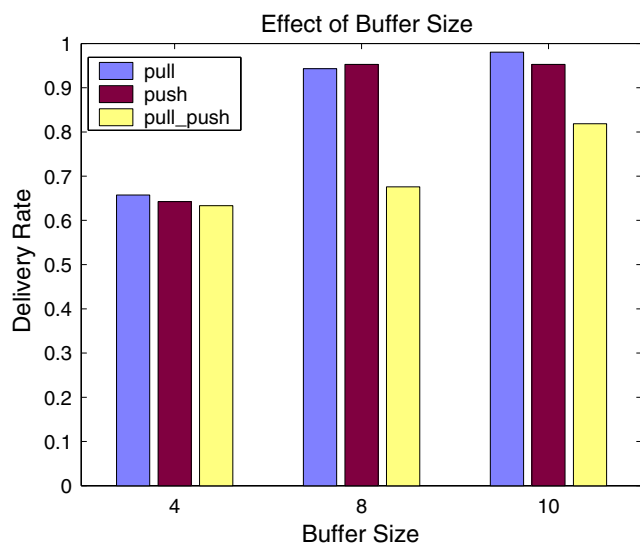


Fig. 8. Effect of the buffer size in sensor nodes on the delivery rate of the algorithms.

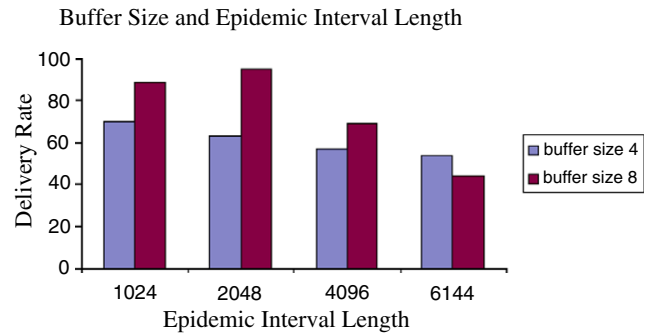


Fig. 9. The relation between the buffer size in a node and the length of the epidemic round during which a data exchange can be performed between two nodes.

size we can extend the interval length and still obtain similar delivery rates compared to lower buffer sizes.

## 6. Conclusions

In this study, we have made a comparative analysis concerning the performance of epidemic algorithms in the context of wireless sensor networks. Our results show the applicability of epidemic algorithms in this context.

The experiment results show that the pull based and push based algorithms perform better than the pull–push based algorithm in terms of delivery rate and scalability. The weak performance of the pull–push based algorithm observed in our simulations is in contrast to its good performance in some other application domains like replicated databases. We believe that the restricted memory resource of sensor devices is the cause of the relatively poor performance of the pull–push based algorithm.

As future work, we are planning to compare epidemic algorithms against protocols like SPIN which exploit the broadcast nature of the wireless medium. Another possible research direction is the investigation of power consumption by the epidemic algorithms discussed in this paper.

## References

- [1] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinchart, D. Terry, epidemic algorithms for replicated database maintenance, in: Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing, Vancouver, Canada, August 1987.
- [2] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: a self-regulating algorithm for code maintenance and propagation in wireless sensor networks, in: First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI), 2004.
- [3] D. Johnson, Routing in ad hoc networks of mobile hosts, in: Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, December 1994.
- [4] L. Li, J. Halpern, Z. Haas, Gossip-based ad hoc routing, in: Proceedings of the 21st Conference of the IEEE Communications Society (INFOCOM'02), 2002.
- [5] Network News Protocol, RFC 967.

- [6] J. Kulik, W. Rabiner, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks. Proceedings of the Fifth ACM/IEEE Mobicom Conference, Seattle, WA, August 1999.
- [7] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'2000), Boston, MA, August 2000.
- [8] D.B. Johnson, D.A. Maltz, Dynamic Source Routing in Ad hoc Wireless Networks, Kluwer Academic Publishers, Dordrecht, 1996.
- [9] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, February 1999, pp. 90–100.
- [10] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, S. Wicker, An empirical study of epidemic algorithms in large scale multihop wireless networks, UCLA Computer Science Department, Technical Report UCLA/CSD-TR-02-0013, 2002.
- [11] P. Levis, N. Lee, TOSSIM: a simulator for TinyOS Networks, September 17, 2003.
- [12] P. Levis, N. Lee, M. Welsh, D. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, in: Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys), 2003.
- [13] nesC 1.1 Language Reference Manual David Gay, Philip Levis, Eric Brewer, David Culler, May 2003.
- [14] The TINYOS web site. Available from: <<http://www.tinyos.net/>>.
- [15] Tython: Scripting TOSSIM Michael Demmer and Philip Levis, February 5, 2004.
- [16] Amin Vahdat, David Becker, Epidemic routing for partially connected ad hoc networks, Technical Report, Duke University, April 2000. Available from: <<http://issg.cs.duke.edu/epidemic/>>.



**Mert Akdere** received his B.S. in Computer Engineering from Bilkent University, Turkey in 2005. He is currently a Ph.D. student in the Department of Computer Science at Brown University. His studies focus on data management issues in distributed information systems and design and analysis of distributed computing infrastructures.



**Cemal Çağatay Bilgin** received his B.S. in Computer Engineering from Bilkent University, in 2005. He is a Ph.D. student in the Department of Computer Science at Rensselaer Polytechnic Institute. His current research interest is automated cancer diagnosis.



**Ozan Gerdaneri** received his B.S. in Computer Engineering from Bilkent University, Turkey in 2005. He is currently an M.S. student in the Department of Computer Engineering at Bilkent University. He is also working at Siemens PSE, Turkey, as a software engineer. His studies focus on parallel and distributed computing.



**Ibrahim Korpeoglu** received his Ph.D. and M.S. degrees from University of Maryland at College Park, both in Computer Science. He is currently an Assistant Professor in the Computer Engineering, Department of Bilkent University, Ankara, Turkey. Prior to joining Bilkent University, he also worked in Ericsson, IBM T.J. Watson Research Center, Bell Laboratories, and Telcordia Technologies, in USA. He has served on the program committees of several conferences and published numerous papers in the area of networking. His research interests include wireless ad hoc and sensor networks, P2P networks, Bluetooth, and mobile computing.



**Özgür Ulusoy** received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. He is currently a Professor in the Computer Engineering Department of Bilkent University in Ankara, Turkey. His research interests include web querying, multimedia database systems, data management for mobile systems, and real-time and active database systems. Prof. Ulusoy has served on numerous program committees for conferences and he was the program cochair of the International Workshop on Issues and Applications of Database Technology that was held in 1998. He coedited a special issue on Real-Time Databases in Information Systems journal and a special issue on Current Trends in Database Technology in the Journal of Database Management. He also coedited a book on Current Trends in Data Management Technology. He has published over 70 articles in archived journals and conference proceedings.



**Uğur Çetintemel** is an Assistant Professor of Computer Science at Brown University. His work focuses on the architecture and performance of distributed information systems and databases. Dr. Çetintemel has published numerous papers in leading databases and systems conferences, primarily in the areas of data stream processing, distributed data storage, and replication. He won the prestigious CAREER award from the National Science Foundation in 2004. Dr. Çetintemel received his doctorate degree in Computer Science from the University of Maryland, College Park and his undergraduate and masters degrees in Computer Engineering from Bilkent University, Ankara.