Ad Hoc Networks 7 (2009) 882–890

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

LOHD: Location-Oblivious Hybrid data Diffusion in wireless sensor networks

Xu Cheng, Feng Wang, Jiangchuan Liu *

School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada V5A 1S6

ARTICLE INFO

Article history:
Received 4 February 2008
Received in revised form 12 July 2008
Accepted 5 August 2008
Available online 23 August 2008

Keywords:
Wireless sensor networks
Data diffusion
Broadcast
Hybrid
PUSH
PULL
Location information

ABSTRACT

Data-centric design has been widely adopted in wireless sensor networks thanks to its efficiency, as PUSH and PULL are two common data dissemination algorithms for such networks. The two algorithms work well with only a few sources or a few sinks, respectively. However, when there are many sources and many sinks, both of them become inefficient. In this paper, we take advantage of these two algorithms, and propose a novel Location-Oblivious Hybrid PUSH–PULL data Diffusion (LOHD) algorithm, which suits a wide range of network settings. Different from most of the existing approaches, LOHD does not rely on any location information, as it adaptively selects an ultra-node in the middle of sources and sinks through a well-controlled flooding, and the ultra-node establishes and maintains the gradients between sources and sinks. LOHD also incorporates enhanced PUSH and PULL to deliver messages along the gradients instead of flooding. We model and analyze the algorithms and perform extensive simulations. The results show that LOHD performs much better than both PUSH and PULL, particularly when the number of sources and sinks increases. We also show that the initialization overhead well resists to such increase, and thus LOHD is highly scalable.

1. Introduction

Wireless sensor networks, consisting of sensor nodes with communication ability, have been successfully used in applications such as hazardous monitoring, enemy tracking, and traffic managing, etc. [1]. Many emerging sensor network applications involve dissemination of observed information to interested nodes and thus demand efficient diffusion mechanisms. In such applications, a data-centric design has been widely adopted, where data generated by sensor nodes are named by attribute values, as a node requests data by sending interests for named data instead of the identity of a certain node.

A pioneer work on data-centric routing is *direct diffusion* [2], which suggests a *two-phase pull* algorithm. *Sinks* identify data by a set of attributes and this information propagates in the *interest* messages that are flooded through the network. As the interests flow through the network, they establish *gradients*, which state the next hop direction of the other nodes that are interested in the data. When the interests arrive at *sources*, the sources generate and send data. The first data sent from sources are marked as *exploratory data* and are sent to all the neighbors that have the matching gradients. When receiving the exploratory data, the sinks send reinforcement messages back to sources to establish the reinforced gradients. Then sources send the remaining data through the reinforced gradients instead of to all the neighbors.

It is known that the original two-phase pull does not work well in certain network configurations. To address this problem, two distinct approaches, *push* and *one-phase pull* (referred to as "PUSH" and "PULL" in the rest of the paper) have been proposed [3]. In PUSH, sinks become passive, and do not flood interests; instead, sources periodically flood exploratory data. Sinks send reinforcement messages back when receiving the exploratory data.

* Corresponding author. Tel.: +1 778 782 4336; fax: +1 778 782 3045.
E-mail addresses: xuc@cs.sfu.ca (X. Cheng), fwa1@cs.sfu.ca (F. Wang), jcliu@cs.sfu.ca (J. Liu).

1570-8705/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2008.08.001

Since no interest is flooded throughout the network, PUSH behaves better when there are many sinks with only a few sources. While in PULL, sinks flood interests and sources send data after receiving the interests, and all the data are sent through the preferred gradients instead of flooding. Since there is no exploratory data and reinforcement messages, PULL works well with the networks where there are many sources but only a few sinks.

However, when the number of both sources and sinks increases, neither PUSH nor PULL can avoid the significant overhead increase. Recent proposals suggest that embedding location information can reduce the amount of flooding, i.e., GPSR [4], GEAR [5] and GHT [6]. Nevertheless, to obtain precise location information, sensor nodes have to be equipped with expensive hardware like GPS, and localization suffers from extra computation and message overhead. Therefore the benefit is limited, not to mention that many sensor networks do not provide such information.

In this paper, we propose a novel Location-Oblivious Hybrid PUSH–PULL data Diffusion algorithm (*LOHD*). LOHD first finds a rendezvous node called *ultra-node*, which is selected by the intersection of local floodings, respectively from sources and from sinks. The ultra-node then builds gradients between sources and sinks, so that the following control messages and data can be sent through the gradients instead of flooding. These operations adapt to diverse networks and sink/source distributions, and more importantly, do not rely on any location information. The performance of LOHD has been evaluated through both analytical modeling and simulations. The results demonstrate that LOHD works well for a wide range of network configurations, as its traffic is remarkably lower than that of PUSH and PULL, and the overhead is scalable with the number of sources and sinks.

The rest of the paper is organized as follows. In Section 2, we review the related works. We present our algorithm in Section 3. An analytical modeling comparison of the three algorithms is discussed in Section 4. We have performed the simulations and present the results in Section 5. Finally, Section 6 concludes the paper.

2. Related work

The original two-phase pull diffusion was first presented in [2], which was later enhanced in [3] by introducing the PUSH and the one-phase PULL diffusion.

To enhance the performance of these basic algorithms, a hybrid rendezvous approach which is similar to ours was proposed [7]. This approach pre-defines a rendezvous point (RP). Both sources and sinks send exploratory data and interests to the RP. However, the location of the RP is pre-defined and its selection is a challenging task, as we solve this problem by adaptively selecting the ultra-node.

A hybrid data dissemination framework was proposed in [8]. In their proposal, the sensor field is partitioned into several functional regions according to supply chain management methodology. Different routing mechanisms are applied in different functional areas. A "comb-needle" discovery support model also combines PUSH and PULL strategies for information discovery [9]. The main difference between our algorithm and the above three is that they are all location-based approaches. By building gradients, our algorithm does not require location information.

Rumor Routing is a routing mechanism without location information [10]. Since both queries and events are sent by random walks, the algorithm might fail. In [11], the authors proposed a geographic routing approach without location information by assigning virtual coordinates to nodes and applying geographic routing over these coordinates. In our algorithm, packets are sent through the established gradients, which have nothing to do with the locations.

3. Algorithm design for LOHD

We consider a typical application scenario, where the sensor nodes are distributed in a large field. On one side, there are source nodes monitoring the environment and generating raw data such as temperature and humidity; on the other side, there are sink nodes requiring and collecting such data. Between sources and sinks, there are intermediate nodes relaying the data from sources to sinks. We consider square fields, in which sources and sinks are clustered in the two diagonal corners.[1] The above scenario is same as many previous studies [3,7,8].

Our solution has three highlights:

- First, we introduce the idea of the ultra-node, which divides the field into two parts and acts as both a source and a sink simultaneously.
- Second, we utilize enhanced PUSH and PULL diffusion in the two parts, in which the control messages are sent through the established gradients instead of flooding.
- Finally, our solution is Location-Oblivious. Without location information, the ultra-node maintains the gradients so that the control messages and data do not have to be flooded.

The LOHD algorithm scheme consists of three stages: (a) ultra-node selection, (b) gradients establishment, and (c) data delivering. Please refer to Fig. 1 for an illustration.

3.1. Ultra-node selection

Our goal in this stage is to select a node in the middle of sources and sinks to serve as the ultra-node.

Since location information is unavailable, we use local flooding to find the nodes in the middle of the field. Specifically, all the sources broadcast the identical SOurce Searching messages (SOS) and all the sinks broadcast the identical SInk Searching messages (SIS). It is worth noting that the SOS/SIS messages do not contain any interest or data, as they only indicate the type of the messages and the number of hops already propagated, and thus the size is much smaller than that of the other message types such as interests and exploratory data. Moreover, these messages sent from different sources and sinks are considered

[1] The scenario of sources and sinks being distributed throughout the network is discussed in Section 5.4.

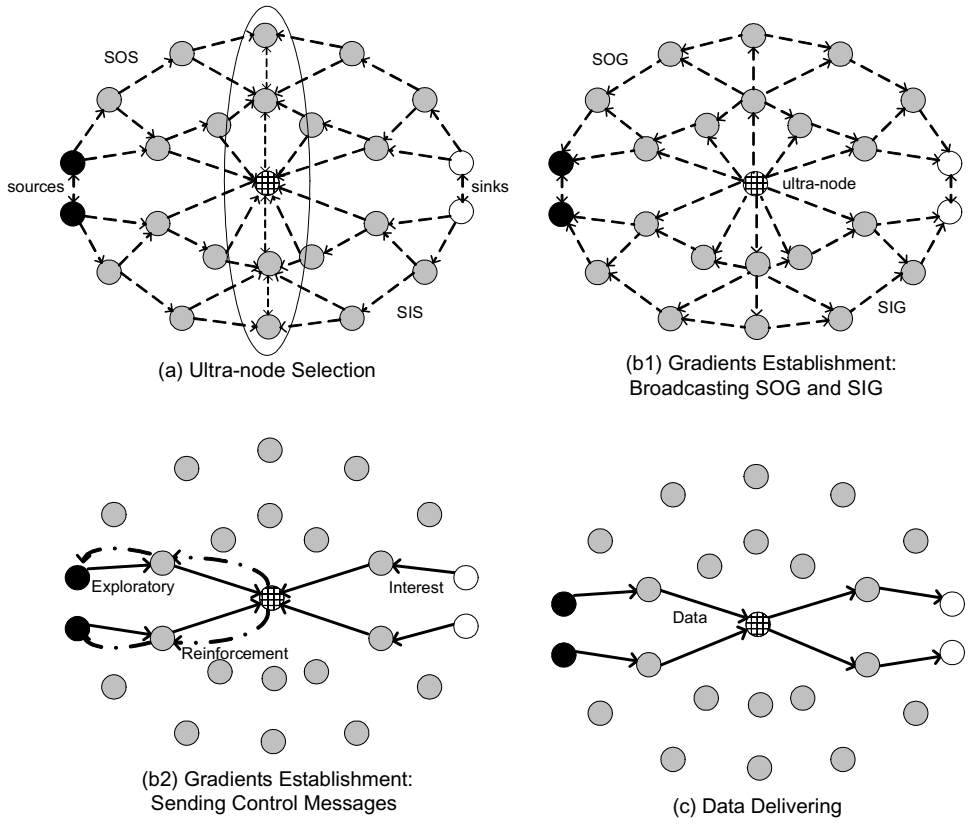

Fig. 1. An illustration of LOHD work flow.

identical, and the nodes only forward one copy unless the message has a smaller hop count (discussed later). Therefore, the flooding amount is independent on the number of sources and sinks, making our algorithm highly scalable. This will be demonstrated in Section 5.3.

We define a node's distance (in hop count) to its nearest sink as d_{SI} and its distance to its nearest source as d_{SO}. For each intermediate node, its d_{SI} and d_{SO} are initially set to infinity. When a node receives an SOS message, if the hop count contained in the message is less than its d_{SO}, the node will then update its d_{SO} with the new hop count and broadcast the message with the hop count $(d_{SO}+1)$; otherwise, it will simply discard the received SOS message. We use a similar mechanism to forward and update on receiving an SIS message. Furthermore, instead of flooding the searching messages throughout the whole network, the messages only propagate for limited hops according to the field size and the node density, as this will be discussed in Section 4.

After the SOS/SIS flooding is done, each node with $|d_{SO}-d_{SI}|<R$ randomly generates an integer number and broadcasts it. By local flooding strictly constrained within the nodes with distance $|d_{SO}-d_{SI}|<R$, the node with the greatest number will be selected as the ultra-node. We call R as the *load balance parameter*. By setting R to 1, the selected ultra-node will be exactly in the middle of the sources and sinks. When R increases, more nodes in the middle area can be the ultra-node candidates. Since the energy of the ultra-node may be consumed rapidly, by increasing R, more nodes can be selected as the ultra-node, and thus balancing the energy consumption. Moreover, the ultra-node selection can be performed periodically, in order to balance the energy consumption.

3.2. Gradients establishment

Since location information is not available, gradients are necessary for relaying the data. Our goal in this stage is to build the gradients between sources and sinks, which are maintained by the ultra-node.

When the ultra-node is determined, the field is then divided into two parts: (1) the nodes between sources and the ultra-node, and (2) the nodes between sinks and the ultra-node. The ultra-node then broadcasts SOurce Gradients messages (SOG) towards the first part and SInk Gradients messages (SIG) towards the second part. The SOG/SIG messages are similar to the SOS/SIS messages, do not contain any data, and have the same size of the SOS/SIS messages. On relaying the messages, all the nodes record the previous sender. Similar to relaying the SOS/SIS messages, the nodes only broadcast once unless receiving a message with a smaller hop count. When sources (sinks) receive the SOG (SIG) messages, the gradients from sources (sinks) to the ultra-node are established. Note that the gradients are unidirectional, and thus the gradients from the ultra-node to the sources and sinks have not been built at this moment.

Then sources send exploratory data and sinks send interests to the ultra-node through the established gradients instead of flooding. When the ultra-node receives the exploratory data and the interests, the gradients from the ultra-node to sources and to sinks are built. Therefore, the required gradients between sources and sinks are successfully established.

The ultra-node then sends the first received data to sinks according to their interests, and also sends the negative reinforcement messages back to the sources that no sink is interested in.

3.3. Data delivering

In this stage, the sources that have not been negatively reinforced send the remaining data to sinks, which propagate along the gradients via the ultra-node.

If some sinks are interested in the same data (as the data being from the same source in many cases), the source will send the data to the ultra-node once, and the ultra-node will replicate the data and send the replicas to different sinks. Notice that the sinks are not requesting the data according to the identity of the source, yet it turns out that the same data are probably from the same source. This offers another advantage of our algorithm, as it can reduce much traffic if many sinks are interested in the same source.

Fig. 2 shows the flowchart of the algorithm for an intermediate node.

For ease of exposition, we do not consider packet losses in the rest of this paper and assume they can be dealt with by under layer protocol.[2]

4. Modeling and analysis

We consider a particular case of the above scenario: N nodes including n_1 sources and n_2 sinks are distributed evenly in an $m \times m$ square field. Therefore we have the node density $\rho = N/m^2$. The sources and sinks are clustered in the diagonal corners. Suppose the nodes have a transmission range of r, and thus the number of neighbor nodes is $k = \rho\pi r^2$. Our analysis is based on densely deployment, however, we believe the results can also be applied to a moderate density deployment.

Let's consider the number of hops h needed to transfer a packet from sources to sinks. If the packet is sent along the diagonal, the hop count can be minimized as $h = \sqrt{2}m/r$. Since the nodes are distributed evenly, there are $\sqrt{N}$ nodes on one side, and thus the distance between two adjacent nodes is $\frac{m}{\sqrt{N}}$, and hence there are at most $\sqrt{2}m/\frac{m}{\sqrt{N}} = \sqrt{2N}$ nodes on the diagonal. Therefore, we use the geometric mean to calculate $h = \sqrt{\frac{2m\sqrt{N}}{r}}$.

[2] SOS/SIS or SOG/SIG message loss will not affect, because even if the messages are lost, the ultra-node can still be found and the gradient can still be established, as we can assume that the node which loses the message does not appear in the system. For interest and exploratory data, the destination node will not receive the lost message because there is only one message being delivered along the gradient.

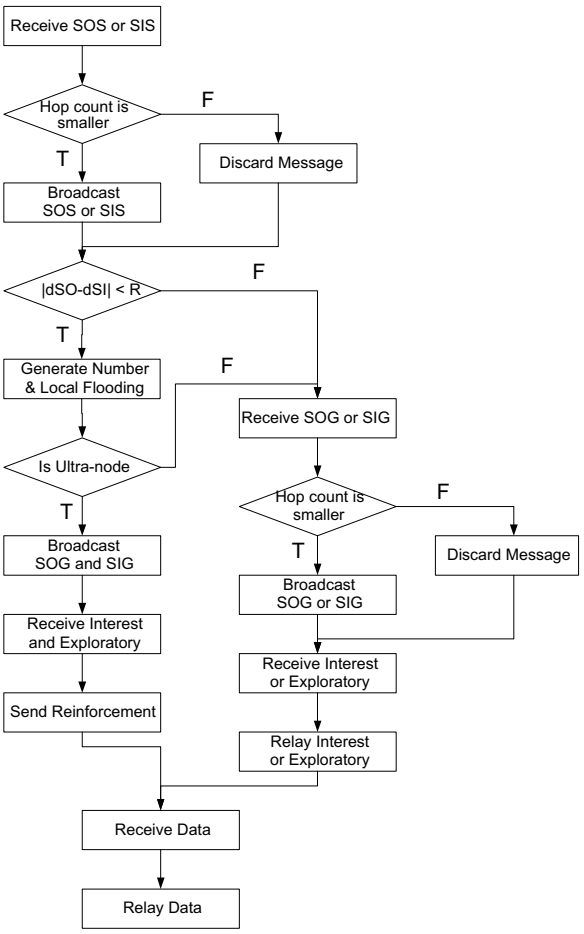

Fig. 2. The flowchart of the algorithm for an intermediate node.

The packet sizes of data, exploratory data, interests, reinforcement and LOHD messages are represented as S_D, S_E, S_I, S_R and S_L. The control intervals of exploratory data, interests and reinforcement are represented as I_E, I_I and I_L, and the data interval is I_D.

The metric we are particularly interested in is the traffic of transferred packet size per event, denoted by U.

To better understand these parameters, we provide a notation list in Table 1.

4.1. The original PUSH and PULL modeling

In PUSH, the exploratory data are flooded throughout the networks. Supposing all the nodes in the network receive and broadcast the exploratory data, and different sources will send different exploratory data, the traffic is $(n_1 \cdot N \cdot k \cdot S_E)$. The reinforcement messages are sent from all the sinks back to sources along the diagonal, so the transferred amount is $(n_2 \cdot h \cdot S_R)$. The subsequent data are sent from sources to all the sinks, also along the diagonal, so the traffic is $(n_2 \cdot h \cdot S_D)$. In an exploratory interval, there are (I_E/I_D) events. Therefore, the total traffic per event is

Table 1
List of notations which will be used in the model

Notation	Explanation
N	Total number of nodes in the network
n_1	Number of sources
n_2	Number of sources
m	Length of the square field
r	Transmission range
S_D	Packet size of data
S_E	Packet size of exploratory data
S_I	Packet size of interest
S_R	Packet size of reinforcement message
S_L	Packet size of LOHD control message
I_D	Interval of sending data
I_E	Interval of sending exploratory data
I_I	Interval of sending interest
I_L	Interval of sending LOHD control message
U	Transferred packet size per event

$$U_{\text{PUSH}} = \left(n_1 NkS_E + n_2 hS_R + n_2 hS_D\left(\frac{I_E}{I_D} - 1\right)\right) \Big/ \left(\frac{I_E}{I_D}\right)$$
$$= \frac{\pi r^2 N^2 I_D S_E}{m^2 I_E} \cdot n_1 + \frac{I_D S_R + (I_E - I_D) S_D}{I_E} \cdot \sqrt{\frac{2m\sqrt{N}}{r}} \cdot n_2$$

In PULL, the interest messages are flooded in the network in the same way as the exploratory data in PUSH, so the traffic is $(n_2 \cdot N \cdot k \cdot S_I)$. The data are sent from sources to all the sinks along the diagonal, so the traffic is $(n_2 \cdot h \cdot S_D)$. In an interest interval, there are (I_I/I_D) events. Therefore, the total traffic per event is

$$U_{\text{PULL}} = \left(n_2 NkS_I + n_2 hS_D \frac{I_I}{I_D}\right) \Big/ \left(\frac{I_I}{I_D}\right)$$
$$= \left(\frac{\pi r^2 N^2 I_D S_I}{m^2 I_I} + S_D \cdot \sqrt{\frac{2m\sqrt{N}}{r}}\right) \cdot n_2$$

4.2. LOHD modeling

In LOHD, supposing both SOS and SIS messages are flooded in half of the network, the traffic is $(2 \cdot \frac{N}{2} \cdot k \cdot S_L)$. For simplicity, we assume all the ultra-node candidates are located on the diagonal, so there are $(h+1)$ nodes. In the worst case, each candidate node will broadcast $(h+1)$ times for the ultra-node selection in the zone, so the traffic is $(k \cdot (h+1)^2 \cdot S_L)$; while in the best case, each candidate node will broadcast only once to find the ultra-node, so the traffic is $(k \cdot (h+1) \cdot S_L)$. We calculate the geometric mean to be $(k \cdot (h+1)^{\frac{3}{2}} \cdot S_L)$. Assuming the ultra-node is located at the middle of the diagonal, all the following control messages and data are sent along the diagonal. The ultra-node then broadcasts the SOG/SIG messages towards the two halves. Same as SOS/SIS, the total transferred amount is $(2 \cdot \frac{N}{2} \cdot k \cdot S_L)$.

Then sources send the exploratory data and sinks send the interests towards the ultra-node, and thus the traffic is $(n_1 \cdot \frac{h}{2} \cdot S_E)$ and $(n_2 \cdot \frac{h}{2} \cdot S_I)$, respectively. The ultra-node then sends the first data to certain interested sinks, transferring $(n_2 \cdot \frac{h}{2} \cdot S_D)$. We assume if the number of sources is greater than that of the sinks, the ultra-node will send the negative reinforcement messages back to $(n_1 - n_2)$ sources, and thus the traffic is $\left((n_1 - n_2) \cdot \frac{h}{2} \cdot S_R\right)$ if $n_1 > n_2$, or 0 otherwise.

Finally, sources send the subsequent data. If sources are fewer than sinks, some sinks will be interested in the same source, and that source will send the data once to the ultra-node, which then sends the replicas to different sinks. Therefore, the traffic is $(n_2 \cdot h \cdot S_D)$ if $n_1 > n_2$, or $(\frac{n_1+n_2}{2} \cdot h \cdot S_D)$ otherwise.

In a LOHD interval, there are (I_S/I_D) events. Considering the exploratory interval I_E and interest interval I_S, the total traffic per event is:

if $n_1 > n_2$,

$$U_{\text{LOHD}} = \left((2N + (h+1)^{\frac{3}{2}})kS_L + \left(n_1 \frac{h}{2} S_E + n_2 \frac{h}{2} S_I \frac{I_E}{I_I} + n_2 \frac{h}{2} S_D + (n_1 - n_2)\frac{h}{2} S_R + n_2 h S_D\left(\frac{I_L}{I_D} - 1\right)\right) \cdot \left(\frac{I_L}{I_E}\right)\right) \Big/ \left(\frac{I_L}{I_D}\right)$$
$$= \frac{I_D(S_E + S_R)}{2I_E} \cdot \sqrt{\frac{2m\sqrt{N}}{r}} \cdot n_1 + \left(\frac{I_D S_I}{2I_I} + \frac{2I_L S_D - I_D(S_R + S_D)}{2I_E}\right) \cdot \sqrt{\frac{2m\sqrt{N}}{r}} \cdot n_2 + \frac{\pi r^2 N\left(2N + \left(\sqrt{\frac{2m\sqrt{N}}{r}} + 1\right)^{\frac{3}{2}}\right) I_D S_L}{m^2 I_L}$$

if $n_1 \leqslant n_2$,

$$U_{\text{LOHD}} = \left((2N + (h+1)^{\frac{3}{2}})kS_L + \left(n_1 \frac{h}{2} S_E + n_2 \frac{h}{2} S_I \frac{I_E}{I_I} + n_2 \frac{h}{2} S_D + (n_1 + n_2)\frac{h}{2} S_D\left(\frac{I_L}{I_D} - 1\right)\right) \cdot \left(\frac{I_L}{I_E}\right)\right) \Big/ \left(\frac{I_L}{I_D}\right)$$
$$= \frac{I_D S_E + (I_L - I_D) S_D}{2I_E} \cdot \sqrt{\frac{2m\sqrt{N}}{r}} \cdot n_1 + \left(\frac{I_D S_I}{2I_I} + \frac{I_L S_D}{2I_E}\right) \cdot \sqrt{\frac{2m\sqrt{N}}{r}} \cdot n_2 + \frac{\pi r^2 N\left(2N + \left(\sqrt{\frac{2m\sqrt{N}}{r}} + 1\right)^{\frac{3}{2}}\right) I_D S_L}{m^2 I_L}$$

4.3. Scalability analysis and comparison

We next discuss how the total number of sensor nodes, including the number of sources and sinks, affects the total traffic, by observing the coefficients of the above equations. Supposing m, r, I and S are fixed, we can simplify the above three equations to:

$$U_{\text{PUSH}} = a_1 \cdot N^2 \cdot n_1 + a_2 \cdot \sqrt[4]{N} \cdot n_2$$
$$U_{\text{PULL}} = (b_1 \cdot N^2 + b_2 \cdot \sqrt[4]{N}) \cdot n_2$$
$$U_{\text{LOHD}} = c_1 \cdot \sqrt[4]{N} \cdot n_1 + c_2 \cdot \sqrt[4]{N} \cdot n_2 + c_3 \cdot N^2$$

The coefficients a, b and c are independent on the number of nodes. For $N > 1$, as N increases, N^2 is obviously much greater than $\sqrt[4]{N}$. Considering N as a dominative factor compared with m, r, I and S, U_{PUSH} and U_{PULL} mainly depend on n_1 and n_2 respectively, and U_{LOHD} depends on both, but

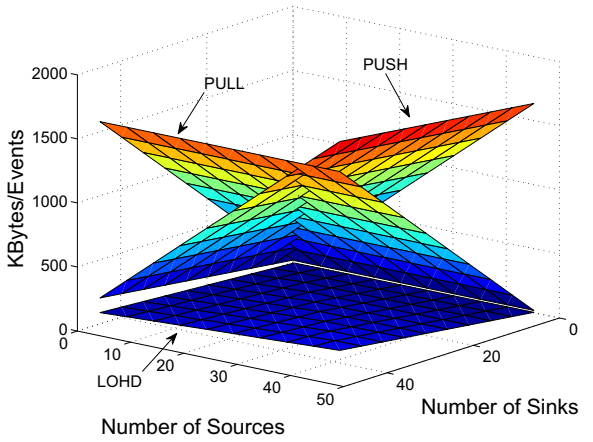

Fig. 3. Mathematic model of the traffic comparison of PUSH, PULL and LOHD.

neither affects much. This demonstrates that LOHD is highly scalable.

To further present numerical comparison results, we put the realistic parameter settings according to Section 5.1, and obtain the following equations:

$$U_{\mathrm{PUSH}} = 36,652n_1 + 4182n_2,$$
$$U_{\mathrm{PULL}} = 35,645n_2,$$
$$U_{\mathrm{LOHD}} = \begin{cases} 73n_1 + 4201n_2 + 11,563 & n_1 > n_2 \\ 2117n_1 + 2157n_2 + 11,563 & n_1 \leqslant n_2 \end{cases}.$$

We plot the above equations in Fig. 3. From this figure, we can see that PUSH mainly depends on the number of sources, and PULL depends on the number of sinks. Both PUSH and PULL have very high traffic compared with LOHD, especially when the number of sources and sinks is great, and this indicates the highly scalability of LOHD. We will show in the next section that our mathematic models correctly match the simulation results.

5. Simulation results

5.1. Configuration and method

We adopt a similar configuration settings used in [3,7], yet with a larger scale. We randomly place 500 nodes by uniform distribution in a 200 m × 200 m square area. The nodes do not move once placed, and we assume the nodes do not die during one period. We examine one period, that is, an exploratory interval in PUSH, an interest interval in PULL, and a LOHD interval in LOHD, and therefore the network is static in such a period. The sources and sinks are clustered in the diagonal corners as the scenario we mentioned above. The radio transmission range is 20 m.

The packet sizes and data/control intervals are assigned according to the configuration of the previous studies [3,7]. Please see Table 2 for the detail. For each situation of different number of sources and sinks, we generate 100 different valid topologies and compute the average results.

Table 2
Simulation configuration: packet sizes and intervals

Type	Packet size (Byte)	Interval (s)
Data	$S_D = 200$	$I_D = 2$
Exploratory	$S_E = 210$	$I_E = 90$
Interest	$S_I = 60$	$I_I = 30$
Reinforcement	$S_R = 100$	–
LOHD	$S_L = 30$	$I_L = 90$

5.2. Traffic comparison of the three algorithms

The results of PUSH and PULL are in accordance with the previous study [3].

We examine the performance of LOHD by varying the number of sources and sinks. In Fig. 4, the traffic is almost constant when sinks are fewer than sources, and increases linearly when sinks are more than sources. In Fig. 5, as sinks increase, the traffic increases linearly with faster trends than previous ones in Fig. 4. The increasing trends in both figures have clear inflexions when the number of the sources and sinks is equal. This can be well explained by the two cases of our LOHD model.

We next compare the three algorithms with different number of sources and sinks. First is the situation of one source and different number of sinks. In Fig. 6, all the three algorithms increase linearly, as PULL has the greatest increasing trend, and LOHD has the smallest trend. Fig. 7 plots the situation of one sink and different number of sources, as PUSH increases linearly and both PULL and LOHD are constant.

In one-source networks, all the three algorithms depend on the number of sinks because all the sinks request data from the sources. It is known that flooding in LOHD is independent on the number of sinks, so it behaves similar to PUSH. Since all the sinks broadcast interests in PULL, the traffic increases significantly as the sinks increase. While in one-sink networks, PULL and LOHD are almost constant, because flooding in both PULL and LOHD is independent on the number of sources, and PUSH behaves worst because all the sources broadcast different exploratory data.

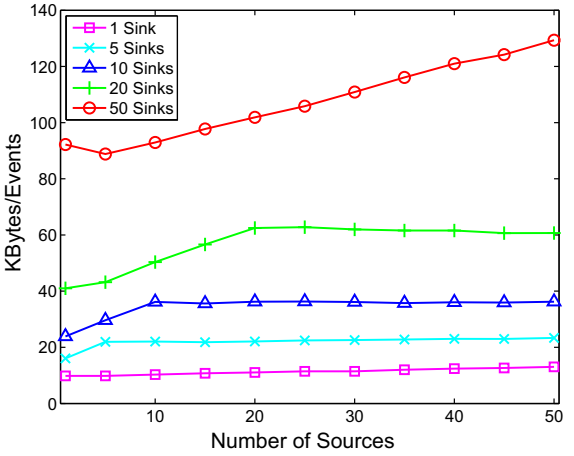

Fig. 4. Traffic of LOHD with different number of sources.

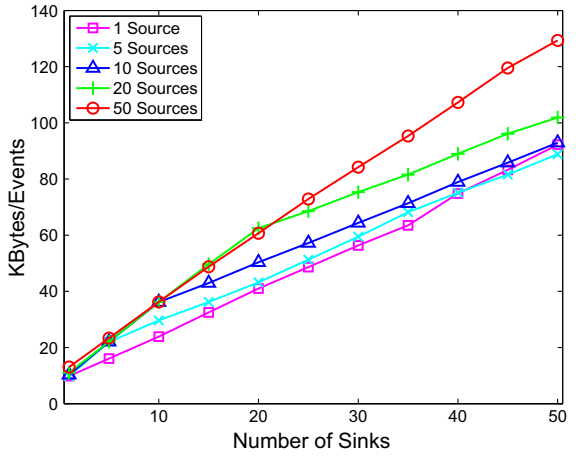

Fig. 5. Traffic of LOHD with different number of sinks.

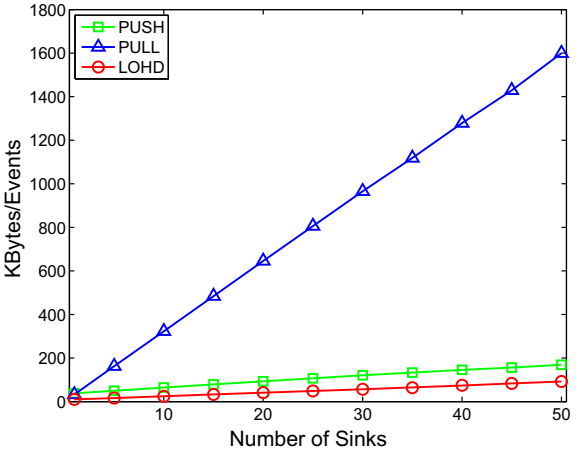

Fig. 6. Traffic comparison of PUSH, PULL and LOHD with one source and different number of sinks.

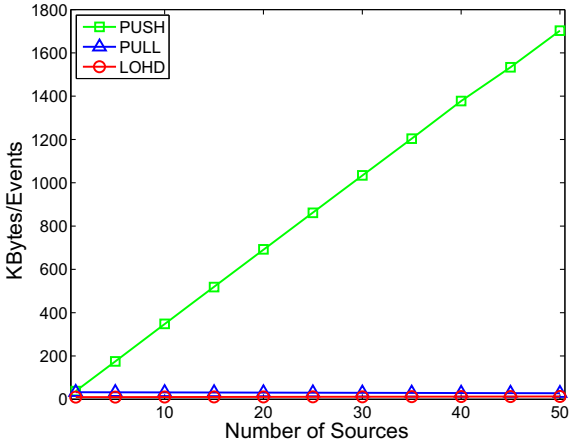

Fig. 7. Traffic comparison of PUSH, PULL and LOHD with one sink and different number of sources.

Fig. 8 compares the three algorithms in the situation of 50 sources and different number of sinks. PUSH is the highest and keeps increasing slowly, PULL is low at first but increases with the fastest trend and will surpass PUSH (when there are more than 50 sinks), and LOHD increases with a slow trend similar to PUSH. Fig. 9 is the situation of 50 sinks and different number of sources. PULL is much higher than the other two at first, yet has a slow decreasing trend, PUSH increases fast and finally surpasses PULL, and LOHD is the lowest and almost keeps constant.

When the number of sources and sinks is much greater, LOHD behaves much better than PUSH and PULL, because we assure that flooding is independent on the number of sources and sinks. We note that PULL has a slow decreasing trend in 50-sink networks, this is because in our simulation, we cluster the sources and sinks in the corners, and the cluster size depends on the number of nodes. If there are many sinks and sources, the paths from sources to some sinks are shorter, and thus the traffic becomes less.

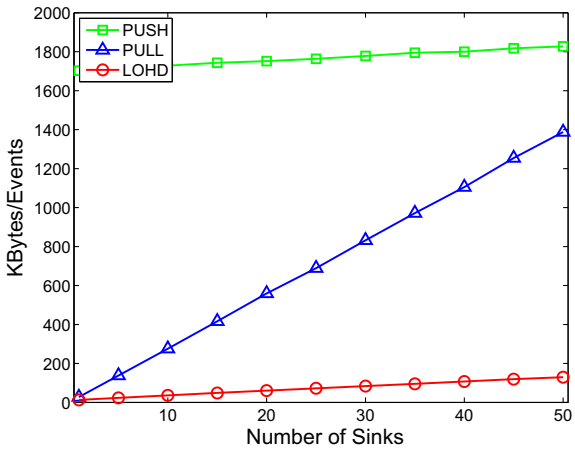

Fig. 8. Traffic comparison of PUSH, PULL and LOHD with 50 sources and different number of sinks.

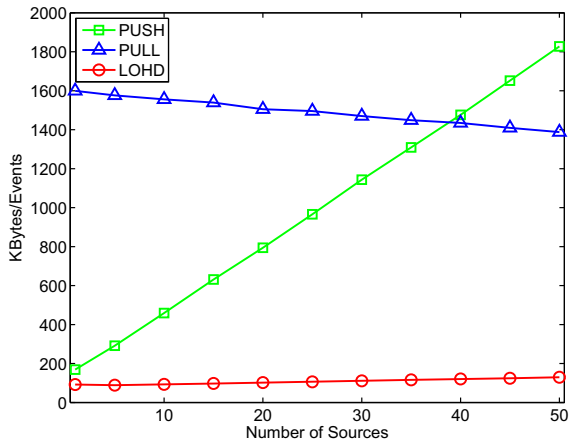

Fig. 9. Traffic comparison of PUSH, PULL and LOHD with 50 sinks and different number of sources.

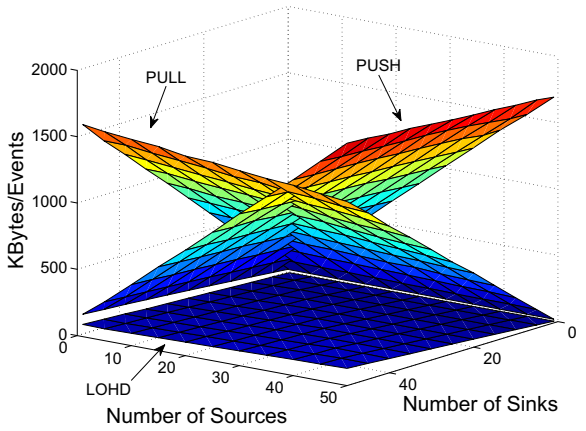

Fig. 10. Simulation result of the traffic comparison of PUSH, PULL and LOHD.

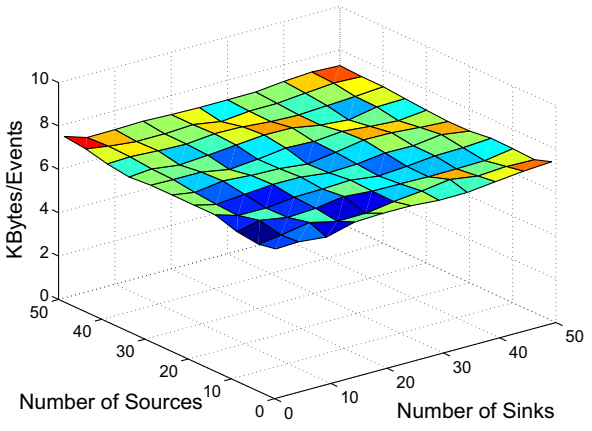

Fig. 11. Initialization overhead of LOHD.

We also plot the simulation results of the three algorithms in Fig. 10 to verify our model. We observe that the increasing trend of the three algorithms in both figures are almost the same. However, we applied some assumptions in the modeling, and hence there are some insignificant differences between the two figures, such as the intersection point. Nevertheless, our models can correctly indicate the increasing trend of the three algorithms, and demonstrate the significant advantage of LOHD against PUSH and PULL.

5.3. Initialization overhead of LOHD

Different from some other location-based diffusion algorithms, LOHD does not require location information. Without such information, in order to avoid flooding a large amount of control messages, LOHD first establishes gradients by a well-controlled local flooding for initialization in each period, and then the following packets can be sent through the gradients instead of flooding.

Although it introduces overhead to flood the SOS/SIS messages for the ultra-node selection and the SOG/SIG messages for the gradients establishment, our simulation results show that this initialization overhead is affordable and worth. For example, the overhead is less than 6% in a 50-source 50-sink network, and more importantly, the overhead is scalable with the number of sources and sinks, as shown in Fig. 11.

5.4. Further discussion

In this paper, we consider the scenario of sources and sinks being clustered in the corners (cluster scenario). However, there are situations that sources and sinks being distributed throughout the network (non-cluster scenario), which possibly leads the LOHD algorithm to failure. We hence conduct an experiment to see the robustness of our LOHD algorithm.

In a cluster scenario, the ultra-node divides the network into two parts and broadcasts SOG and SIG towards the two parts respectively. In a non-cluster scenario, the ultra-node is not necessary in the middle of the network and no longer divides the network into two parts. Therefore, we have to modify the algorithm to broadcast SOG and SIG throughout the network, assuring that all the sources can receive SOG and all the sinks can receive SIG. This obviously increases the overhead. In this scenario, the distance between a pair of source and sink is probably closer, since they are no longer clustered in the corner, and thus the traffic transferred might be smaller.

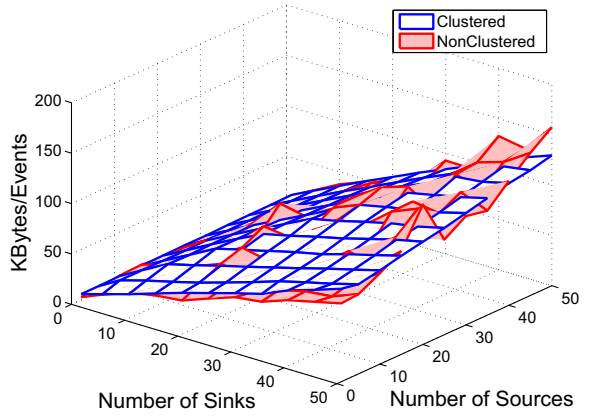

Fig. 12. Comparison of scenario of sources and sinks clustered in the corner and distributed throughout the network.

We conduct the experiment to implement the non-cluster scenario, and plot the result in Fig. 12. We can see that the two have similar results, as the non-cluster scenario has slightly smaller traffic, because the distance to transfer packets is smaller, and has greater deviation because the location of the ultra-node, sources and sinks is too random. The experiment can demonstrate that our LOHD algorithm is robust, and can adapt to different network situations.

6. Conclusion

We revisit the PUSH and PULL diffusion algorithms and propose our Location-Oblivious Hybrid PUSH–PULL data

Diffusion algorithm (LOHD), which can greatly reduce the amount of flooding, and thus the overhead is significantly lower. Furthermore, our algorithm does not require any location information, so that it eliminates the extra requirement for additional hardware and the overhead for complicated localization computation, and thus it is more adaptive to diverse networks. We mathematically model and analyze the three algorithms. We have also performed extensive simulations, which demonstrate that LOHD works much better than PUSH and PULL in relatively large-scale networks with many sources and many sinks.

Acknowledgement

This research is supported by a Canadian NSERC Discovery Grant and an NSERC Strategic Project Grant.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102–114.
[2] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the 6th ACM MobiCom, 2000, pp. 56–67.
[3] J. Heidemann, F. Silva, D. Estrin, Matching data dissemination algorithms to application requirements, in: Proceedings of the 1st ACM SenSys, 2003, pp. 218–229.
[4] B. Karp, H.T. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in: Proceedings of the 6th ACM MobiCom, 2000, pp. 243–254.
[5] Y. Yu, R. Govindan, D. Estrin, Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks, Tech. Rep. CSD-TR-01-0023, UCLA Computer Science Department, 2001.
[6] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker, GHT: a geographic hash table for data-centric storage, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), 2002, pp. 78–87.
[7] B. Krishnamachari, J. Heidemann, Application-specific modelling of information routing in wireless sensor networks, in: Proceedings of the 23rd IEEE International Performance Computing and Communications Conference (IPCCC'04), 2004, pp. 717–722.
[8] W. Liu, Y. Zhang, W. Lou, Y. Fang, A robust and energy-efficient data dissemination framework for wireless sensor networks, Wireless Networks 12 (4) (2006) 465–479.
[9] X. Liu, Q. Huang, Y. Zhang, Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks, in: Proceedings of the 2nd ACM SenSys, 2004, pp. 122–133.
[10] D. Braginsky, D. Estrin, Rumor routing algorithm for sensor networks, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), 2002, pp. 22–31.
[11] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, I. Stoica, Geographic routing without location information, in: Proceedings of the 9th ACM MobiCom, 2003, pp. 96–108.

Xu Cheng received the B.Sc. degree from Peking University, Beijing, China, in 2006, and the M.Sc. degree from Simon Fraser University, British Columbia, Canada, in 2008, both in computer science. He is currently a Ph.D. student and research assistant under the supervision of Dr. Jiangchuan Liu, at Simon Fraser University. His research interests include peer-to-peer networks, video streaming and sensor networks. He has been a Student Member of IEEE since 2008. He was awarded the SFU-CS Graduate Entrance Scholarship in 2006 and the Computing Science Graduate Fellowship in 2008 at Simon Fraser University.

Feng Wang received both his Bachelor's degree and Master's degree in Computer Science and Technology from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently a Ph.D. student in School of Computing Science at Simon Fraser University, conducting research on wireless sensor networks under the supervision of Prof. Jiangchuan Liu. His research interests include wireless sensor networks, Peer-to-Peer live streaming and distributed computing. He has been a Student Member of IEEE and IEEE Communications Society since 2007. In the summer of 2006, he interned in Wireless and Networking Group at Microsoft Research Asia, where he conducted research on Peer-to-Peer live video streaming. He also interned in Department of Computing at Hong Kong Polytechnic University in the spring of 2008, where his research was on data collections in wireless sensor networks. He was awarded Tsinghua University Scholarship for Excellent Student in 1998, 2000 and 2001. At Simon Fraser University, he was awarded the SFU-CS Graduate Entrance Scholarship in 2005 and the Graduate Fellowship in 2007, 2008 and 2009.

Jiangchuan Liu (Member, IEEE) received the B.Eng. degree (cum laude) from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong, China, in 2003, both in computer science. He is currently an Assistant Professor in the School of Computing Science, Simon Fraser University, BC, Canada, and was an Assistant Professor at The Chinese University of Hong Kong from 2003 to 2004. He is a corecipient of one European patent and two U.S. patents. His research interests include Internet architecture and protocols, media streaming, wireless ad hoc networks, and service overlay networks. He was a Guest Editors for ACM/Kluwer Journal of Mobile Networks and Applications (MONET) Special Issues on wireless sensor Networking and Wireless Mesh Networking. He has been a Technical Program Committee (TPC) member for many networking conferences, including IEEE INFOCOM and IWQoS. He was TPC Cochair for the First IEEE International Workshop on Multimedia Systems and Networking (WMSN'05) and Information System Cochair for IEEE INFOCOM'04. He is an Editor of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He received a Microsoft Research Fellowship (2000) and the Hong Kong Young Scientist Award (2003). He is an Associate Editor of IEEE Transactions on Multimedia.